

Trajectory Following Methods in Control System Design

Thomas L. Vincent
Aerospace and Mechanical Engineering
University of Arizona, Tucson, AZ 85721, USA
vincent@u.arizona.edu

Walter J. Grantham
Mechanical and Materials Engineering
Washington State University, Pullman, WA 99164, USA
grantham@mme.wsu.edu

A trajectory following method for solving optimization problems is based on the idea of solving ordinary differential equations whose equilibrium solutions satisfy the necessary conditions for a minimum. The method is “trajectory following” in the sense that an initial guess for the solution is moved along a trajectory generated by the differential equations to a solution point. With the advent of fast computers and efficient integration solvers, this relatively old idea is now an attractive alternative to traditional optimization methods. One area in control theory that the trajectory following method is particularly useful is in the design of Lyapunov optimizing feedback controls. Such a controller is one in which the control at each instant in time either minimizes the “steepest decent” or “quickest decent” as determined from the system dynamics and an appropriate (Lyapunov- like) decent function. The method is particularly appealing in that it allows the Lyapunov control system design method to be used “on-line.” That is, the controller is part of a normal feedback loop with no off-line calculations required. This approach eliminates the need to solve two-point boundary value problems associated with classical optimal control approaches. We demonstrate the method with two examples. The first example is a nonlinear system with no constraints on the control and the second example is a linear system subject to bounded control.

1 Introduction

In this paper we combine a **trajectory following method** (Vincent & Grantham 1997), (Vincent 2000) useful for solving a **nonlinear programming problem** (NPP) with a **Lyapunov optimizing control** (LOC) method (Vincent & Grantham 1997) to design feedback controllers for nonlinear dynamical systems. The LOC method requires that at each state an associated NPP be solved. The NPP problem is subject to nonlinear inequality constraints on the control variables. This optimization problem is solved approximately, using differential equations based on the trajectory following algorithm. In the following sections we review the NPP problem, use of the trajectory following method to solve it and the LOC method for designing controllers. We then show how trajectory following can be used with the LOC method followed by some examples.

2 Nonlinear Programming Problem (NPP)

In order to design controllers using the LOC method we must deal with an underlying nonlinear programming problem of the form:

$$\text{minimize } G(\mathbf{u}) \quad (1)$$

subject to a set of inequality constraints

$$\mathbf{u} \in \mathcal{U} = \{\mathbf{u} \mid \mathbf{h}(\mathbf{u}) \geq 0\}, \quad (2)$$

where $G(\mathbf{u})$ is a scalar cost function of an n_u -dimensional control vector \mathbf{u} , the set $\mathcal{U} \subseteq R^{n_u}$ is a specified constraint set, and $\mathbf{h}(\mathbf{u}) = [h_1(\mathbf{u}) \dots h_{n_h}(\mathbf{u})]^\top$ is an n_h -dimensional vector of inequality constraint functions, with $G(\mathbf{u})$ and each component of $\mathbf{h}(\mathbf{u})$ being continuous and continuously differentiable in \mathbf{u} .

The general **necessary conditions** for a local minimum to this problem are well known and are given by:

Minimization necessary conditions: *If $\mathbf{u}^* \in \mathcal{U}$ is a regular local minimizing point for $G(\mathbf{u})$ subject to the constraints (2), then there exists a vector $\boldsymbol{\gamma} = [\gamma_1 \dots \gamma_{n_h}]^\top$ such that*

$$\mathbf{0}^\top = \frac{\partial L(\mathbf{u}^*, \boldsymbol{\gamma})}{\partial \mathbf{u}} \quad (3)$$

$$\mathbf{0} \leq \mathbf{h}(\mathbf{u}^*) \quad (4)$$

$$\mathbf{0} \leq \boldsymbol{\gamma} \quad (5)$$

$$0 = \boldsymbol{\gamma}^\top \mathbf{h}(\mathbf{u}^*), \quad (6)$$

where $\boldsymbol{\gamma}$ is called a Lagrange multiplier vector and the Lagrangian function $L(\cdot)$ in (3) is defined as

$$L(\mathbf{u}, \boldsymbol{\gamma}) \triangleq G(\mathbf{u}) - \boldsymbol{\gamma}^\top \mathbf{h}(\mathbf{u}). \quad (7)$$

These necessary conditions incorporate all of the inequality constraint functions, both active ($h_i = 0$) and inactive ($h_i > 0$). Note that (4)–(6) imply that $\gamma_i = 0$ if $h_i(\mathbf{u}^*) > 0$, so that inactive constraints will not play a role.

When using the trajectory following method to solve this problem, it is useful to identify **simple inequality constraints**, of the form

$$\mathbf{u}_{\min} \leq \mathbf{u} \leq \mathbf{u}_{\max},$$

where every component of the vector \mathbf{u}_{\min} is less than every component of the vector \mathbf{u}_{\max} . One advantage of introducing simple inequality constraints even for a problem which can be formulated without any constraints is that we are now guaranteed that a global minimum (and a global maximum) will exist. The necessary conditions for a minimum, with only simple inequality constraints, reduce to a straightforward set of equations.

From (7)

$$L = G(\mathbf{u}) - \gamma_{\max}^\top (\mathbf{u}_{\max} - \mathbf{u}) - \gamma_{\min}^\top (\mathbf{u} - \mathbf{u}_{\min})$$

so that the necessary conditions for \mathbf{u}^* to be a minimum become

$$\begin{aligned} \mathbf{0} &= \left. \frac{\partial G}{\partial \mathbf{u}} \right|_{\mathbf{u}=\mathbf{u}^*} + \gamma_{\max}^\top - \gamma_{\min}^\top \\ \mathbf{0} &\leq \mathbf{u}_{\max} - \mathbf{u}^* \\ \mathbf{0} &\leq \mathbf{u}^* - \mathbf{u}_{\min} \\ \mathbf{0} &\leq \gamma_{\max} \\ \mathbf{0} &\leq \gamma_{\min} \\ 0 &= \gamma_{\max}^\top (\mathbf{u}_{\max} - \mathbf{u}^*) + \gamma_{\min}^\top (\mathbf{u}^* - \mathbf{u}_{\min}). \end{aligned}$$

It is easy to show that this set of necessary conditions for simple inequality constraints is equivalent to the following conditions:

$$\begin{aligned} \text{if } u_i^* = u_{i_{\max}} \quad &\text{then } \left. \frac{\partial G}{\partial u_i} \right|_{u_i=u_i^*} \leq 0 \\ \text{if } u_i^* = u_{i_{\min}} \quad &\text{then } \left. \frac{\partial G}{\partial u_i} \right|_{u_i=u_i^*} \geq 0 \\ \text{if } u_{i_{\min}} < u_i^* < u_{i_{\max}} \quad &\text{then } \left. \frac{\partial G}{\partial u_i} \right|_{u_i=u_i^*} = 0. \end{aligned} \tag{8}$$

3 Using trajectory following to solve the NPP

Most numerical methods for solving the NPP are based on iteratively solving discrete algebraic equations. The trajectory following method [Arrow and Hurwicz, 1977] is based on solving continuous differential equations, whose equilibrium solutions satisfy the necessary conditions for a minimum, maximum, or min-max, depending on the problem to be solved. This method has the advantage that no special programming skill is required to generate solutions and the differential equation approach fits in naturally with the LOC method. Trajectory following is also quite robust, in that many problems, known to be difficult to solve by other methods, can be easily solved. The same assumptions on G and \mathcal{U} are made as with the NPP.

3.1 Minimizing G Subject to No Constraints

The basic idea behind a “trajectory following” method is to form a set of differential equations from the gradient of the cost function. Consider first the problem of minimizing $G(\mathbf{u})$ subject to no constraints. Suppose that we use the local minimal point \mathbf{u}^* as an initial condition for integrating the differential equations

$$\dot{\mathbf{u}} = \mathbf{f}(\mathbf{u}), \tag{9}$$

where $\mathbf{f}(\cdot)$ is a function at our disposal, to be determined shortly. Suppose also that we calculate $G(\mathbf{u})$ along the trajectory generated by the solution to (9). Let \mathbf{u}^* be a local minimal point for $G(\mathbf{u})$. The basic necessary condition for minimizing $G(\mathbf{u})$ is that

$$\frac{\partial G(\mathbf{u}^*)}{\partial \mathbf{u}} \mathbf{e} \geq 0$$

for all vectors \mathbf{e} tangent to \mathcal{U} at \mathbf{u}^* . Since $-\mathbf{e} = \mathbf{f}(\mathbf{u}^*)$ is a tangent vector to the forward-time trajectory generated by (9) it follows that we need

$$\left. \frac{dG}{dt} \right|_{\mathbf{u}^*} = \left. \frac{\partial G}{\partial \mathbf{u}} \right|_{\mathbf{u}^*} \mathbf{f}(\mathbf{u}^*) \leq 0. \quad (10)$$

That is, along any feasible trajectory in a neighborhood of the minimal point, the time derivative of the cost must be non-positive. Since we are interested in a trajectory which will search for a minimum, the above observation suggests that we integrate (9) by choosing

$$\mathbf{f}(\mathbf{u}) = - \left[\frac{\partial G}{\partial \mathbf{u}} \right]^\top.$$

To see why this is a good choice, suppose that $G(\cdot)$ has a unique global minimum at \mathbf{u}^* . This means that $G(\mathbf{u}) > G(\mathbf{u}^*)$ for all $\mathbf{u} \neq \mathbf{u}^*$ and that, along any trajectory generated by the **unconstrained Trajectory Following equations**

$$\dot{\mathbf{u}} = - \left[\frac{\partial G}{\partial \mathbf{u}} \right]^\top, \quad (11)$$

we have

$$\frac{dG}{dt} = - \frac{\partial G}{\partial \mathbf{u}} \left[\frac{\partial G}{\partial \mathbf{u}} \right]^\top < 0$$

for all points along the trajectory except at $\mathbf{u} = \mathbf{u}^*$, where $dG/dt = 0$. Due to the assumed uniqueness of \mathbf{u}^* all trajectories from every point in the control space must asymptotically approach \mathbf{u}^* . Thus, in order to solve the problem of minimizing $G(\cdot)$ in an unconstrained control space, one need only choose an initial condition for \mathbf{u} and integrate (11) until the equilibrium solution is approached. Unfortunately, since we only have asymptotic stability, the equilibrium point will never be obtained in finite time. We must instead stop the integration process in the neighborhood of the equilibrium point when some accuracy criteria has been satisfied. We will not get an exact solution, but we can come as close to an exact solution as desired by specifying that the integration be stopped only when the difference between two consecutive values of G is less than a pre-specified amount. Note that in using (11) to search for a minimum it is usually more convenient to calculate the gradient using a numerical method such as central difference than to use an analytical gradient.

While the Trajectory Following method has some similarities with the well known gradient method, it is not the same since the gradient vector under Trajectory Following is continuously updated. Because of this, we avoid convergence difficulties the numerical gradient method has with some problem. Using Trajectory Following with such problems results in a **stiff system** of differential equations. The integration routine must be able to handle stiff systems in such situations.

3.1.1 Rosenbrock's Banana Function

Perhaps the most famous example of a problem in which traditional gradient methods have convergence difficulties is Rosenbrock's banana function [Grace, 1992] given by

$$G = 100 (u_1^2 - u_2)^2 + (1 - u_1)^2.$$

It is also an example of a problem in which use the trajectory following algorithm (11) will result in a stiff system of equations. However we have shown (Vincent & Grantham 1997), (Vincent 2000) that this problem may be readily solved using the trajectory following method provided that a stiff integration solver is used such as the Gear algorithm or any of the stiff integration solvers used by Matlab.

3.2 Minimizing G Subject to Simple Inequality Constraints

If the NPP problem has just simple constraints, we may modify (11) to incorporate the necessary conditions as given by (8). In particular the trajectory following algorithm becomes

$$\dot{u}_i = \begin{cases} 0 & \text{if } u_i = u_{i_{\max}} \text{ and } \frac{\partial G}{\partial u_i} \leq 0 \\ 0 & \text{if } u_i = u_{i_{\min}} \text{ and } \frac{\partial G}{\partial u_i} \geq 0 \\ -\frac{\partial G}{\partial u_i} & \text{otherwise,} \end{cases} \quad (12)$$

where $i = 1, \dots, n_u$. Note that any equilibrium solution generated by the Trajectory Following algorithm will satisfy (8) and hence will represent a local minimum candidate. This algorithm can handle initial conditions $\mathbf{u} \notin \mathcal{U}$, as well as any problems associated with integrating past a boundary if the integration routine includes a test for the inequality conditions being satisfied and setting u_i to the appropriate boundary value for any inequality not satisfied. In this way the first two conditions of (12) will be automatically satisfied.

It is well worth the extra coding effort to take care of situations where $\mathbf{u} \notin \mathcal{U}$ for any integration time step. The advantage gained is that if a minimal solution is on the boundary of \mathcal{U} then exact values can be obtained for all components of \mathbf{u} which lie on the boundary of \mathcal{U} . If G has a unique global minimum on \mathcal{U} , then the trajectory generated by (12) will asymptotically approach the minimal point starting from almost all $\mathbf{u} \in \mathcal{U}$. The exceptions would be any interior points of \mathcal{U} for which $\partial G / \partial \mathbf{u} = \mathbf{0}$.

If it is known that G has several local minimal points, the above algorithm can be modified to search for all of them. In modifying the algorithm to search for multiple minimal points, it turns out that this process is facilitated by searching for all local minimal points and all maximal points at the same time.

3.2.1 Quadratic Cost

A two-dimensional positive-definite or negative-definite quadratic cost function subject to simple inequality constraints will either have one global minimum point with one or more local maximal points or one global maximal point with one or more local minimal points. Consider the problem of minimizing

$$G = u_1^2 - 2u_1u_2 + 4u_2^2$$

subject to the constraints

$$-10 \leq u_i \leq 10.$$

Suppose that we wish to find all local minimal points using Trajectory Following. Since this problem could have more than one local minimal point we will attempt to find all

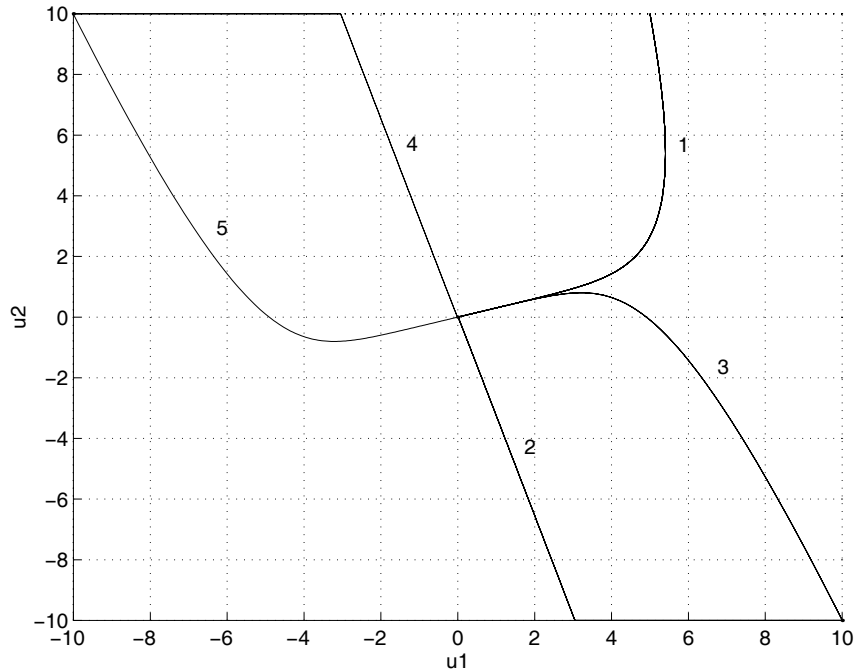


Figure 1: Using trajectory following to find multiple optimal points.

local minimal and maximal points. Rather than automating this process and just printing out the solutions obtained, the iterative process was done by hand in order to illustrate the trajectories obtained as shown in Figure 1. The first forward trajectory (#1) finds a local minimum at the origin. The first backward trajectory (#2) finds a local maximum at $u_1 = 10$, $u_2 = -10$, and so on, until the second backward trajectory (#4) finds the local maximum at $u_1 = -10$, $u_2 = 10$.

3.3 Minimizing G Subject to Non-Simple Constraints

The Trajectory Following method classifies all inequality constraints as being either ‘simple’ or ‘non-simple’. Since the simple constraints (i.e., upper and lower bounds) are handled by the algorithm itself, in this section, we will use the notation $\mathbf{h}(\mathbf{u}) \geq \mathbf{0}$ to denote any non-simple constraints which are included in the constraint set. Non-simple constraints can be included in the Trajectory Following algorithm through the use of a penalty function. In particular, we form a new function

$$W = G + \beta \sum_{i=1}^{n_h} \delta_i h_i^2,$$

where β is a large positive number, with $\delta_i = 1$ if $h_i < 0$ and $\delta_i = 0$ if $h_i \geq 0$. In this way whenever a given inequality constraint is satisfied, no penalty is applied. For every inequality constraint not satisfied, a positive penalty term is added to G . The minimum value of W should be very close to the minimum value of G provided that the penalty parameters are taken sufficiently large.

Trajectory following is used to find the minimum of W by replacing G in (12) with W . It should be noted that even if G by itself would not make (12) a stiff system, large penalty values certainly will. Most numerical procedures which use penalty functions for minimizing G recommend using a sequential procedure in which W is first minimized using small values for the penalty parameters. The solution obtained is used as a starting condition with larger values of the penalty parameters until some convergence criteria is obtained. Such a sequential procedure may not be needed to minimize W using Trajectory Following, provided that a stiff integration routine is used. In fact, very large values for the penalty functions can be used at the onset, and if the solution obtained satisfies pre-specified convergence criteria, only one run need be made. Note that convergence criteria should include the amount that the non-simple inequality constraints are allowed to be negative.

Equality constraints may also be handled in a similar fashion by simply writing them as two inequality constraints. That is the equality constraint

$$\psi(\mathbf{u}) = 0$$

maybe written as the two inequality constraints

$$\begin{aligned}\psi(\mathbf{u}) &\geq 0 \\ -\psi(\mathbf{u}) &\geq 0.\end{aligned}$$

3.3.1 Non-simple constraints

Using Trajectory Following to find all minimal points for (p.621 Reklaitis, Ravindran & Ragsdell 1983)

$$G = u_1^2 u_2^4,$$

subject to the constraints

$$0.25 - (u_1)^{\frac{3}{4}} u_2 = 0$$

$$u_1 u_2^4 + u_2^2 \sqrt{u_1} - 1 \geq 0$$

and the bounds

$$\begin{aligned}0 &\leq u_1 \leq 10 \\ 0 &\leq u_2 \leq 10.\end{aligned}$$

Using the trajectory following method as illustrated in Figure 2 we obtain the solution

$$\begin{aligned}G_{\min} &= 0.0386 \\ u_1 &= 0.1010 \\ u_2 &= 1.3945.\end{aligned}$$

Note how the trajectory following algorithm first drives the solution the neighborhood of equality constant and then to the solution point where the inequality constraint is active ($h = 0$).

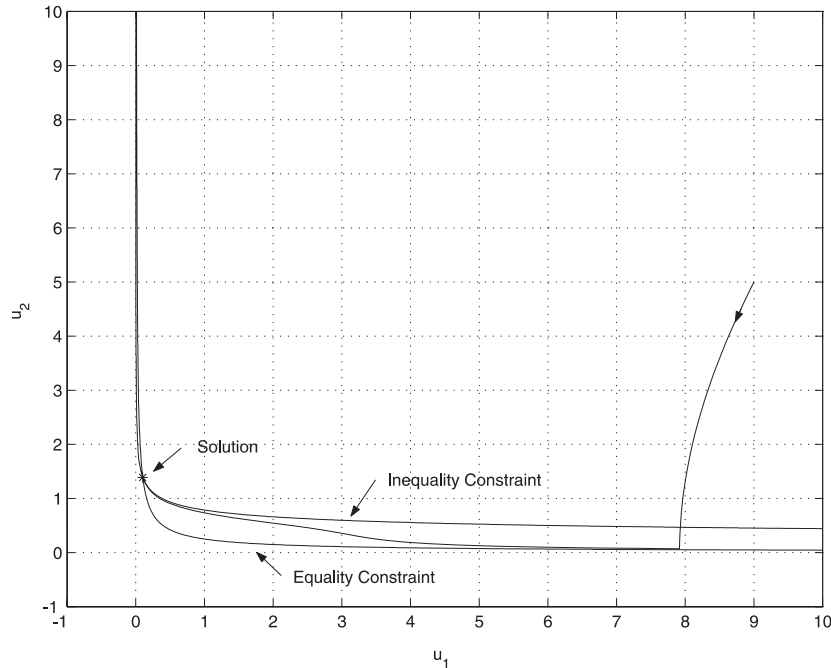


Figure 2: Trajectory following method with non-simple constraints.

3.4 The LOC method

The combination of function minimization and Lyapunov stability techniques represents a powerful approach to closed-loop nonlinear control design (Gutman & Leitmann 1976), (Grantham 1981), (Grantham 1982), (Grantham 1986), (Grantham & Chingcuanco 1984), (Anderson & Grantham 1989) for systems of the form

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}), \quad (13)$$

where $\mathbf{x} \in R^{n_x}$. The basic idea behind the LOC method is to choose a candidate Lyapunov-type function $W(\mathbf{x})$, such as distance to a target, and then choose $\mathbf{u}(\mathbf{x})$ so that $W[\mathbf{x}(t)]$ decreases along trajectories of (13). There is more freedom in this process than is normally available in standard Lyapunov stability analysis. This is because at every state point \mathbf{x} , the control system allows for a set of possible velocity vectors $\dot{\mathbf{x}} \in \mathbf{f}(\mathbf{x}, \mathcal{U})$, rather than a single velocity vector $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) \leftarrow \mathbf{f}[\mathbf{x}, \mathbf{u}(\mathbf{x})]$, for making $W[\mathbf{x}(t)]$ decrease.

A function $W(\mathbf{x})$ is a **descent function** for a target set $\mathcal{X} \subset R^{n_x}$ if the following conditions hold at points that are controllable to the target:

- i) $W(\mathbf{x})$ is continuous and continuously differentiable outside the target set and on the boundary of the target set.
- ii) The regions $W(\mathbf{x}) \leq c$ are nested. That is, for $c_1 < c_2$

$$\{\mathbf{x} \mid W(\mathbf{x}) \leq c_1\} \subset \{\mathbf{x} \mid W(\mathbf{x}) \leq c_2\}.$$

iii) The target is contained in one of the $W(\mathbf{x}) \leq c$ regions.

In addition, if the target set \mathcal{X} is bounded then we also require

iv) The regions $W(\mathbf{x}) \leq c$ are bounded.

Condition iv) is imposed to exclude the possibility that trajectories might decrease the value of $W[\mathbf{x}(t)]$ but with $\mathbf{x}(t)$ approaching infinity instead of approaching the target set \mathcal{X} .

Once a specific descent function $W(\mathbf{x})$ has been selected, the feedback control $\mathbf{u}(\mathbf{x})$ is chosen so that $W[\mathbf{x}(t)]$ decreases at each state \mathbf{x} , hopefully penetrating $W(\mathbf{x}) = \text{constant}$ surfaces. If the resulting controller does make $W[\mathbf{x}(t)]$ decrease everywhere outside the target, and the target is the origin, then $W(\mathbf{x})$ will be a Lyapunov function for the feedback control system which is a sufficient condition for asymptotic stability. However we will not require this. There are many applications where a Lyapunov optimizing control is effective even though $W[\mathbf{x}(t)]$ does not decrease everywhere.

There are two related methods that can be used to make $W[\mathbf{x}(t)]$ decrease: steepest descent and quickest descent (Vincent & Grantham 1997). We will only review the **quickest decent control** method here. This method attempts to make trajectories penetrate surfaces of constant $W(\mathbf{x})$ as quickly as possible (minimize $dW[\mathbf{x}(t)]/dt$). This is done by choosing the feedback control $\mathbf{u}(\mathbf{x})$ to minimize $dW[\mathbf{x}(t)]/dt$ at each state \mathbf{x} . A feedback control law $\mathbf{u}(\mathbf{x})$ is a Lyapunov optimizing **quickest descent** control for a specified descent function $W(\mathbf{x})$ if

$$\dot{W}[\mathbf{x}, \mathbf{u}(\mathbf{x})] \leq \dot{W}(\mathbf{x}, \mathbf{u}) \quad \text{for all } u \in \mathcal{U}, \quad (14)$$

where

$$\dot{W}(\mathbf{x}, \mathbf{u}) \triangleq \frac{\partial W(\mathbf{x})}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x}, \mathbf{u}). \quad (15)$$

The quantity $\dot{W}(\mathbf{x}, \mathbf{u})$ is the time rate of change of $W[\mathbf{x}(t)]$ along a trajectory $\mathbf{x}(t)$. Quickest descent control uses information about both direction and speed. It has the potential to produce an optimal feedback control law since it has exactly the same structure as the optimization function associated with Pontryagin's Minimum Principle for minimum-time optimal control problems. It is for this reason that the region of effectiveness under quickest descent control can always be made equal to the controllable set by using a suitable choice for the descent function. However, without solving the optimal control problem, it is not obvious how to do this. Despite this potential, if $W(\mathbf{x})$ is not carefully chosen, quickest descent control may result in an undesirable trade-off of direction of motion in exchange for an increase in speed.

4 Using trajectory following with the LOC method

The LOC method requires that a NPP problem be solved at every point of the trajectory. If an analytical solution is available a closed-loop controller is obtained directly. See (Vincent & Grantham 1997) for some examples where this is possible. However, if an analytical solution can not be found, then some numerical technique must be used in order to solve the minimization problem at every point of a trajectory. Depending on the application, this

might be difficult to do in a continuous fashion in order to produce the desired closed-loop control. One way to avoid continuously solving a numerical optimization problem is to use the Trajectory Following method. Using this method will produce what could be called Lyapunov sub-optimizing control.

The idea is to solve the numerical optimization problem only once, at the initial point of the trajectory. For all other points \mathbf{x} along a trajectory, the control is determined from the differential equations

$$\dot{\mathbf{u}} = -\frac{\partial \dot{W}}{\partial \mathbf{u}}$$

for quickest descent control. The initial conditions for the differential equations are determined from the initial numerical optimization problem and the gradients are calculated numerically so that only \dot{W} need be specified. How well these solutions agree with the actual solutions for the control, as the system moves along the trajectory, depends on how well these equations track the optimal solutions. Gains can be introduced into these equations to adjust the rates. Generally, they will do a good job if the control does not change too rapidly. However, even in cases where part of the Lyapunov optimizing control is bang-bang, good results can be obtained as we will show below.

One consideration that is not addressed here is the issue of local versus global optimization. If the initial control is indeed a global minimum solution, then the trajectory following method will generally continue to track a global minimum for the decent function. Likewise the trajectory following method will tend to follow a non-global minimum initial control. As illustrated below, one does not necessarily have to specify an initial optimal initial control. In this case the solution obtained will initially not be a Lyapunov optimizing control, however the trajectory following method will soon converge to at least a local minimum for the decent function.

5 Examples

We present two examples both of which have been previously solved analytically (Vincent & Grantham 1997). The solutions we obtain below are indistinguishable from the analytical solutions.

5.1 Quickest Descent Control for Zermelo's Problem

Zermelo's problem (Vincent & Grantham 1997) may be thought of as a swimmer fighting a strong current. The swimmer with a velocity 1 is swimming against a current of velocity 2. He can swim in any direction as given by the angle u , with the equations of motion given by

$$\begin{aligned} \dot{x}_1 &= -2 + \cos u \\ \dot{x}_2 &= \sin u. \end{aligned}$$

His objective is to reach circular buoy centered at the origin. Using the descent function

$$W(\mathbf{x}) = x_1^2 + x_2^2,$$

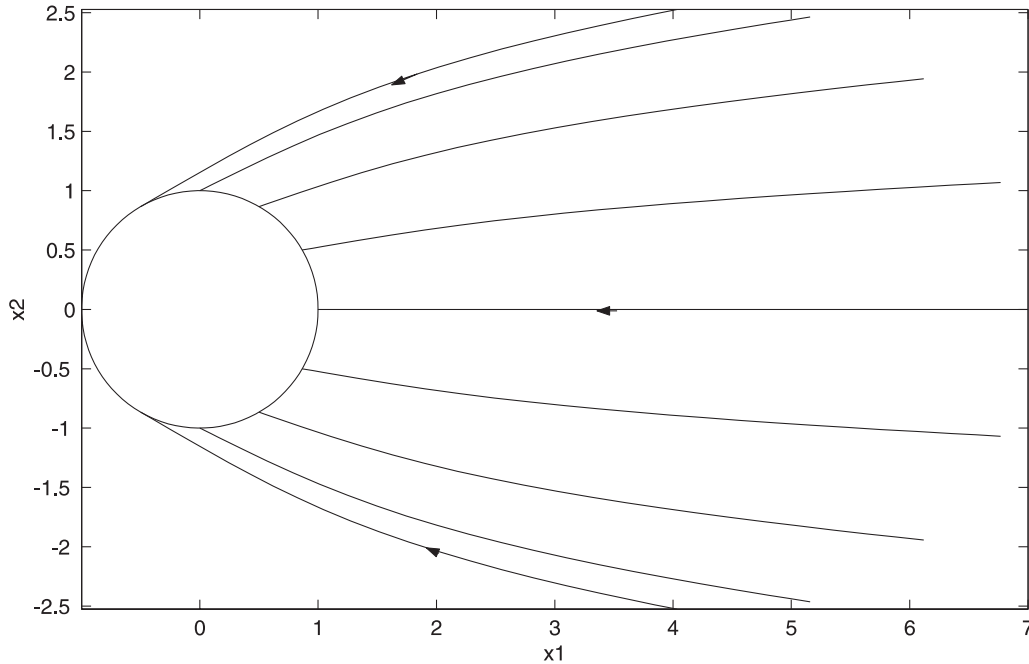


Figure 3: Trajectories for Zermello's problem as generated using trajectory following.

quickest descent control minimizes

$$\dot{W}(\mathbf{x}, u) = \frac{\partial W}{\partial x_1} f_1 + \frac{\partial W}{\partial x_2} f_2 = 2x_1(-2 + \cos u) + 2x_2 \sin u. \quad (16)$$

Under trajectory following, the swimmer chooses his direction according to

$$\dot{u} = -\frac{\partial \dot{W}}{\partial u} = 2x_1 \sin u - 2x_2 \cos u. \quad (17)$$

He can improve his chances of reaching the buoy if he chooses his choice for $u(0)$ at the initial time by minimizing (16) as given by

$$u(0) = \pi + \tan^{-1} \left[\frac{x_2(0)}{x_1(0)} \right], \quad (18)$$

where $\tan^{-1}(\cdot)$ is the Fortran `ATAN2`(\cdot, \cdot) two-argument arctangent function.

Figure 3 shows the quickest descent trajectories that reach the target using (17) with (18). The region of effectiveness for the quickest descent controller is a good portion of, but not all of the controllable set for this problem (Vincent & Grantham 1997). The reason for this is that quickest descent minimization of this descent function results in a trade off between the direction of motion and speed. As a result, trajectories starting in the controllable set but near the controllability boundary leave the controllable set.

5.2 Quickest Descent Control of the Linearized Inverted Pendulum

The linearized inverted pendulum is given by the system of equations

$$\begin{aligned}\dot{x}_1 &= x_2 \\ \dot{x}_2 &= x_1 + u\end{aligned}\tag{19}$$

where x_1 is the angular displacement from the vertical position and x_2 is the angular rate of change. The pendulum is attached to the shaft of a DC motor that can supply torque within the bounds

$$-1 \leq u \leq 1.$$

It is easy to show that for this problem the controllable set to the origin is the region $|x_1 + x_2| < 1$. Thus in designing a feedback controller we will not be concerned with states outside of this region, but the controller should be effective throughout the controllable set.

Since the system (19) is linear except for the control bounds, we will use a positive definite quadratic form for the descent function,

$$W(\mathbf{x}) = \mathbf{x}^\top \mathbf{P} \mathbf{x},\tag{20}$$

where $\mathbf{P} > 0$ is symmetric and positive definite. Contours of constant $W(\mathbf{x})$ are ellipses surrounding the origin. In picking a \mathbf{P} matrix for steepest descent control, we note that for $u = 0$ the origin is an unstable saddle point equilibrium, with eigenvalues $\mu = \pm 1$ and corresponding eigenvectors $\boldsymbol{\xi} = [1 \ \mu]^\top$. For the stable eigenvalue $\mu = -1$, the associated eigenvector $\boldsymbol{\xi} = [1 \ -1]^\top$ is in a direction parallel to the controllability boundaries. The system trajectories that approach the origin also lie on this eigenvector. For this reason, we choose a descent function that will produce trajectories in this preferred direction of motion. In particular, we choose a real, symmetric, positive definite \mathbf{P} matrix with $\boldsymbol{\xi} = [1 \ -1]^\top$ as one of its eigenvectors, associated with the smallest eigenvalue of \mathbf{P} , in which case $\boldsymbol{\xi}$ lies on the semi-major axis of the ellipses. Since the eigenvalues of \mathbf{P} are real and positive, the eigenvectors will be orthogonal for distinct eigenvalues and will be aligned with the axes of the ellipse. The desired \mathbf{P} matrix must satisfy the eigenvector equations

$$\begin{bmatrix} p_{11} & p_{12} \\ p_{12} & p_{22} \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$

for eigenvalues $0 < a < b$. Thus we see that there is a family of such \mathbf{P} matrices, given by

$$\mathbf{P} = \alpha \begin{bmatrix} \beta + 1 & \beta - 1 \\ \beta - 1 & \beta + 1 \end{bmatrix} \quad \alpha > 0, \beta > 1.\tag{21}$$

For illustrative purposes we choose $\alpha = 1/2$ and $\beta = 4$, yielding the positive definite quadratic descent function

$$W(\mathbf{x}) = \frac{1}{2} [x_1 \ x_2] \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \frac{5}{2} x_1^2 + 3x_1 x_2 + \frac{5}{2} x_2^2.\tag{22}$$

The quickest descent control is obtained by choosing $u^*(\mathbf{x})$ to minimize

$$\begin{aligned}\dot{W}(\mathbf{x}, u) &= \frac{\partial W}{\partial x_1} f_1 + \frac{\partial W}{\partial x_2} f_2 \\ &= (5x_1 + 3x_2)x_2 + (3x_1 + 5x_2)(x_1 + u)\end{aligned}\tag{23}$$

subject to the constraints $|u| \leq 1$.

Using the trajectory following method, we would use

$$\dot{u} = -\frac{\partial W}{\partial u} = -(3x_1 + 5x_2).$$

Note that since W is linear in u

$$\sigma = 3x_1 + 5x_2$$

actually plays the role of a switching function which is nonzero everywhere, except on the **switching surface** $\sigma(\mathbf{x}) = 0$. In other words an optimal controller would be bang-bang, switching between the limits on u as the trajectory crosses the switching surface. With this in mind we increase the “gain” on \dot{u} and use

$$\dot{u} = -50(3x_1 + 5x_2).\tag{24}$$

Figure 4 illustrates use of the trajectory following algorithm to solve this problem. For clarity only four trajectories are shown. In generating this figure an initial optimization problem was **not** solved. The initial value for u is set equal to zero. Note that a trajectory either chatters to the origin (shown dark) upon reaching the switching curve (shown dashed) or it will cross this curve once and then chatter when reaching the switching curve the second time. All these trajectories start inside the controllable set and reach the origin in a chattering fashion. The solutions obtained have exactly the same behavior as those obtained with an analytical solution (Vincent & Grantham 1997).

We see that quickest descent control also produces a preferred direction of approach to the target, along the chattering switching surface corresponding to $\partial W(\mathbf{x})/\partial x_2 = 0$. However, this quickest descent control has a domain of attraction that is less than the controllable set. Trajectories starting at some points in the controllable set leave the set, never to return. This difficulty can be eliminated by changing the \mathbf{P} matrix in the descent function to yield a switching surface $\sigma(\mathbf{x}) = x_1 + x_2$ which is parallel to the controllability boundaries $x_1 + x_2 = \pm 1$.

6 Discussion

Implementing a feedback controller based on the Lyapunov optimizing method requires that a nonlinear optimization problem be solved at every point of the trajectory. We have demonstrated how one may use trajectory following to readily solve this continuous sequence of optimization problems. The basic idea behind the Lyapunov optimizing controller (LOC) is

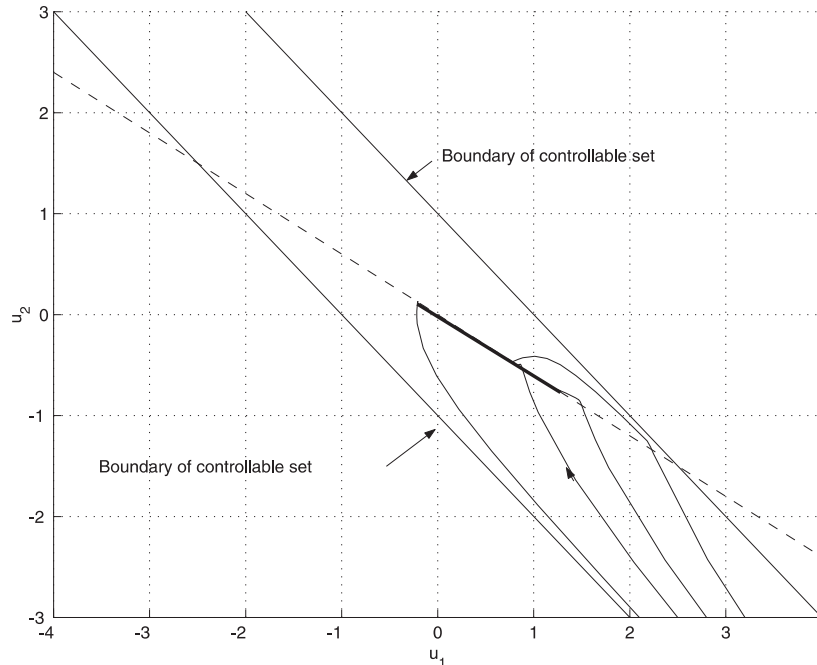


Figure 4: Use of trajectory following can result in a (near optimal) chattering solution.

to determine a control that will maximize the rate at which the system crosses lines of constant decent function. Using trajectory following to solve this problem results in a controller that is “on line.” That is the control is determined in a continuous fashion from the solution to a differential equation. Only the initial condition to this equation need be determined in order to proceed.

In the first example problem we determine the initial condition by first solving a nonlinear programming problem. So doing, the trajectory following method will track the actual solution known for this problem very closely. However as demonstrated in the second example, one need not even solve for the initial condition for u in order to successfully apply the method. In the second problem the trajectory following algorithm was set to track the gradient very quickly. After a very short period of time the algorithm is yielding the correct control. Perhaps surprisingly, we also demonstrate with the second example that the trajectory following algorithm is able to produce a chattering solution when such a solution is, indeed, the proper one under LOC.

We are interested in applying this method to much more comprehensive examples and we hope to report on this in the future.

References

- Anderson, M. J. & Grantham, W. J. (1989), ‘Lyapunov optimal feedback control of an inverted pendulum’, *J. Dynamic Systems, Measurement and Control* **111**(4), 554–558.

- Grantham, W. J. (1981), Controllability conditions and effective controls for nonlinear systems, *in* 'Proc. 20th I.E.E.E. Conf. On Decision and Control', San Diego, CA.
- Grantham, W. J. (1982), Some necessary conditions for steepest descent controllability, *in* 'Proc. American Controls Conf.', Alexandria, VA.
- Grantham, W. J. (1986), ODESSYS: Ordinary differential equations simulation system for nonlinear optimization, controls, and differential games, *in* 'Proc. 3rd I.E.E.E. Conf. On Computer-Aided Control Systems Design', Arlington, VA.
- Grantham, W. J. & Chingcuanco, A. O. (1984), 'Lyapunov steepest descent control of constrained linear systems', *I.E.E.E. Trans. on Automatic Control* **AC-29**(8), 740–743.
- Gutman, S. & Leitmann, G. (1976), Stabilizing feedback control for dynamical systems with bounded uncertainty, *in* 'Proc. I.E.E.E. Conf. On Decision and Control', pp. 94–99.
- Reklaitis, G., Ravindran, V. & Ragsdell, K. (1983), *Engineering Optimization Methods and Applications*, Wiley, New York.
- Vincent, T. & Grantham, W. (1997), *Nonlinear and Optimal Control Systems*, Wiley, New York.
- Vincent, T. L. (2000), *Progress in Optimization*, Kluwer, chapter Optimization by Way of the Trajectory Following Method, pp. 239–254. edited by X. Q. Yang, et. al.