

A CASE STUDY IN TRAJECTORY OPTIMIZATION: DESIGNING A TOBOGGAN RUN

ROBERT J. VANDERBEI

Operations Research and Financial Engineering
Princeton University

Revised October 28, 2000

ABSTRACT. In this first paper, we consider how to design a slide to make a toboggan go from beginning to end as quickly as possible.

1. INTRODUCTION

We express our optimization models in the AMPL modeling language [2]. This language provides a common mechanism for conveying problems to codes to solve them. When solving problems we generally use two different solvers: (a) LOQO [4, 5, 6, 1], which implements an interior-point method for general nonlinear optimization and (b) SNOPT [3], which implements an active set strategy with a quasi-Newton method for the QP subproblem.

2. TOBOGGANING

It wouldn't be right to end our discussion of trajectory optimization without discussing the famous Brachistochrone problem. We do it here in this last section in the context of designing a slide which will get the rider from the beginning to the end in the shortest amount of time.

We assume in this study that the slide will be constructed out of frictionless material. We let x denote, as usual, horizontal displacement and we let y denote vertical displacement. In contrast with earlier conventions, we assume that y increases as one moves downward. The slide will connect two points having coordinates $(x_0, y_0) = (0, 0)$ and (x_f, y_f) . The toboggan starts from rest at the top of the slide. Hence, initially the toboggan has zero kinetic energy and zero potential energy (due to gravity). Since the

Date: October 28, 2000.

1991 Mathematics Subject Classification. Primary 65L10 Secondary 34B15.

Key words and phrases. trajectory optimization, optimal control, constrained optimization.

Research supported by NSF grant DMS-9870317, ONR grant N00014-98-1-0036.

<pre> param n := 512; param x {j in 0..n} := j/n; # Variables var y {j in 0..n} >= 0; # Abbreviations var dydx {j in 1..n} = (y[j]-y[j-1])/(x[j]-x[j-1]); var f {j in 1..n} = sqrt((1+dydx[j]^2)/y[j-1]); </pre>	<pre> minimize time: sum {j in 1..n} f[j]*(x[j]-x[j-1]) ; subject to y0: y[0] = 1.0e-12; subject to yn: y[n] = 1; subject to monotone {j in 1..n}: dydx[j] >= 0; let {j in 0..n} y[j] := x[j]; solve; </pre>
--	---

FIGURE 1. A working AMPL model for the Brachistochrone problem.

slide is frictionless, there is no loss of energy as heat. By conservation of energy, the total energy must always remain zero. So, when the toboggan has dropped to level y , we have

$$\frac{1}{2}mv^2 - mgy = 0.$$

Here, v denotes the speed of the toboggan. From this relation we see that

$$v = \sqrt{2gy}.$$

Now, the time to do the run can be computed as an integral of differential chunks of time

$$T = \int_0^T dt = \int_0^T \frac{ds(t)}{v(t)}.$$

This last integral can be reparametrized using any monotone function of time. The natural candidate is horizontal displacement x . With this choice, we get

$$(1) \quad T = \int_0^{x_f} \frac{\sqrt{1 + y'(x)^2}}{\sqrt{2gy(x)}} dx.$$

The problem then is to find a function $y(x)$ that minimizes this integral and satisfies the constraints $y(0) = 0$ and $y(x_f) = y_f$. Using calculus of variations, one can show that the general form of the solution to (1) is a cycloid:

$$\begin{aligned} x &= k^2(\theta - \sin \theta) \\ y &= k^2(1 - \cos \theta). \end{aligned}$$

However, to find the values of k and θ_f to satisfy the original terminal conditions involves solving a transcendental equation—not such an easy task.

The AMPL model expressing the minimization of T as given in (1) is shown in Figure 1. It took some tinkering before we were able to get to the working model shown in the Figure. The main issue is that $y(0) = 0$ appears in the denominator of the integrand when $x = 0$. Hence, the integral is a singular integral. To address this, we first changed the boundary condition to $y(0) = 10^{-12}$. Also, since $\text{dydx}[j]$

<pre> param n := 512; param y {j in 0..n} := (j/n); # Variables var x {j in 0..n}; # Abbreviations var dx dy {j in 1..n} = (x[j] - x[j-1])/(y[j] - y[j-1]); var f {j in 1..n} = sqrt((dx dy[j]^2+1)/((y[j]+y[j-1])/2)); </pre>	<pre> minimize time: sum {j in 1..n} f[j]*(y[j]-y[j-1]) ; subject to x0: x[0] = 0; subject to xn: x[n] = 1; let {j in 0..n} x[j] := y[j]; solve; </pre>
---	--

FIGURE 2. A second working AMPL model for the Brachistochrone problem.

appearing in the definition of $f[j]$ represents the value of the derivative at the midpoint of the interval $[j-1, j]$, one would expect to use the best estimate for y at this same place, i.e., $(y[j]+y[j-1])/2$. However, with this choice for denominator in the integrand, the optimal solution exhibits a very large jump discontinuity at $x = 0$. Presumably this is caused by the singularity but the details elude us. Using $y[j-1]$, as shown, works but changing it to $y[j]$ renders the problem unsolvable to both LOQO and SNOPT. Again, the reason remains a mystery. It is easy to think of lots of other things to try (and we did) but we stop here in favor of a different line of attack. The model shown in the Figure is solved by LOQO in 26 iterations. It takes 0.85 seconds on a 366 MHz PC.

Instead of using horizontal displacement as the parameterization variable in (1), we could equally well have chosen to use the vertical displacement variable. With this choice, we get

$$(2) \quad T = \int_0^{y_f} \frac{\sqrt{1 + x'(y)^2}}{\sqrt{2gy}} dy.$$

The AMPL model for this formulation of the problem is shown in Figure 2. LOQO solves this model in just 10 iterations. It takes only 0.26 seconds on a 366 MHz PC. Clearly, from a numerical perspective, this formulation is much better than the previous one.

2.1. **Lesson.** The lesson to take away from this case study is that one should consider a variety of ways to formulate a given problem. Some might be much easier to solve than others.

REFERENCES

- [1] H.Y. Benson, D.F. Shanno, and R.J. Vanderbei. Interior-Point Methods for Nonconvex Nonlinear Programming: Jamming and Comparative Numerical Testing. Technical Report ORFE-00-2, Dept. of Operations Research and Financial Engineering, Princeton University, Princeton NJ, 2000. Submitted to *Math. Prog.* 1
- [2] R. Fourer, D.M. Gay, and B.W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Scientific Press, 1993. 1

- [3] P.E. Gill, W. Murray, and M.A. Saunders. User's guide for SNOPT 5.3: A Fortran package for large-scale nonlinear programming. Technical report, Systems Optimization Laboratory, Stanford University, Stanford, CA, 1997. 1
- [4] R.J. Vanderbei. LOQO: An interior point code for quadratic programming. *Optimization Methods and Software*, 12:451–484, 1999. 1
- [5] R.J. Vanderbei. LOQO user's manual—version 3.10. *Optimization Methods and Software*, 12:485–514, 1999. 1
- [6] R.J. Vanderbei and D.F. Shanno. An interior-point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13:231–252, 1999. 1

ROBERT J. VANDERBEI, PRINCETON UNIVERSITY, PRINCETON, NJ