

A CASE STUDY IN TRAJECTORY OPTIMIZATION: RANGE MAXIMIZATION FOR A HANG GLIDER

ROBERT J. VANDERBEI

Operations Research and Financial Engineering
Princeton University

Revised October 28, 2000

ABSTRACT. In this first paper, we consider how to fly a hang glider so as to maximize or minimize the range of the glide.

1. INTRODUCTION

We express our optimization models in the AMPL modeling language [6]. 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 [9, 10, 11, 1], which implements an interior-point method for general nonlinear optimization and (b) SNOPT [7], which implements an active set strategy with a quasi-Newton method for the QP subproblem.

2. HANG GLIDING

The problem we now consider is to compute the flight inputs to a hang glider so as to provide a maximum range flight. The specific problem we shall analyze is taken from [5].

One should note that the model presented here also applies to the flight of an airplane with its engines off (only the data are different). In this case, maximizing the range could be a life-saving endeavor.

The hang glider (with pilot) is pulled down by the force of gravity associated with its mass m , has a lifting force L acting perpendicular to its velocity relative to the air, and a drag force D acting in a direction opposite to the relative velocity. Denote by x the horizontal position of the glider, by v_x the horizontal component of the absolute velocity, by y the vertical position, and by v_y the vertical component of absolute velocity.

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.

Recall that one of the lessons of the previous section is the importance of scrutinizing every model carefully looking for errors. With that in mind and with our apologies for not practicing what we preach, we ask the reader to trust us as we assert that the following description of the equations of motion for a hang glider is correct.

2.1. Stable Airmass. The equations of motion for a glider in a stable airmass are as follows:

$$\begin{aligned} v_x &= \dot{x}, & a_x &= \dot{v}_x, & a_x &= \frac{1}{m}(-L\frac{v_y}{v_r} - D\frac{v_x}{v_r}), \\ v_y &= \dot{y}, & a_y &= \dot{v}_y, & a_y &= \frac{1}{m}(L\frac{v_x}{v_r} - D\frac{v_y}{v_r}) - g \end{aligned}$$

with

$$v_r = \sqrt{v_x^2 + v_y^2}, \quad L = \frac{1}{2}c_L\rho S v_r^2, \quad \text{and} \quad D = \frac{1}{2}c_D(c_L)\rho S v_r^2.$$

In [5], it is assumed that there was an updraft 250 meters into the flight. To keep the situation simple, we start by assuming that the air is still. In the next subsection, we shall consider updrafts and more complicated situations.

The glider is controlled by the lift coefficient c_L (the pilot pushes or pulls on the control bar to change c_L). The drag coefficient c_D is assumed to depend on the lift coefficient as

$$c_D(c_L) = c_0 + kc_L^2$$

where c_0 and k are fixed parameters, $c_0 = 0.034$ and $k = 0.069662$ being realistic values (and the ones used in [5]). In addition, there are limits on the lift coefficient:

$$0 \leq c_L \leq c_{L\max} := 1.4$$

(corresponding to the control bar being pulled in all the way and pushed out all the way, respectively). The other constants in the problem have the following specific values:

| | |
|---------------|--------------------------|
| $m = 100$ | mass of glider and pilot |
| $S = 14$ | wing area |
| $\rho = 1.13$ | air density |
| $g = 9.81$ | acc due to gravity. |

The boundary conditions are:

$$\begin{aligned} x(0) &= 0, & y(T) &= 900, \\ y(0) &= 1000, & v_x(T) &= 13.23, \\ v_x(0) &= 13.23, & v_y(T) &= -1.288. \\ v_y(0) &= -1.288, & & \end{aligned}$$

The total time T for the flight is, of course, a variable. The objective is to maximize $x(T)$.

With a stable airmass, one expects that, for appropriate choice of boundary conditions, the optimal control will be static. The optimal static control is found by minimizing the ratio of drag to lift (or, equivalently, maximizing L/D):

$$D/L = \frac{c_0}{c_L} + kc_L.$$

The minimum occurs at $c_L = \sqrt{c_0/k} = 0.69862$. Then using the fact that accelerations in a static solution vanish, we deduce that

$$-\frac{v_x}{v_y} = \frac{L}{D} = 10.274$$

$$v_r = \sqrt{\frac{2mg}{\rho S \sqrt{c_D^2 + c_L^2}}} = 13.2901.$$

From this we quickly compute that

$$v_x = 13.23, \quad v_y = -1.288.$$

That is, the initial and final velocities given above in the boundary conditions for the dynamic version of the problem match the optimal values for the static version of the problem. Hence, we expect the optimal solution of the dynamic problem to be in fact static. Let's see if this is what we get.

The AMPL model for the midpoint discretization is shown in Figure 1. With this discretization, we have the following variables

$$T, \quad x\{0, \dots, N\}, \quad y\{0, \dots, N\}, \quad c_L\{1, \dots, N-1\}$$

and the following equality constraints

$$\text{newt_x}\{i \text{ in } 1..N-1\}, \text{ newt_y}\{i \text{ in } 1..N-1\},$$

$$x_ic, \quad y_ic, \quad vx_ic, \quad vy_ic,$$

$$y_fc, \quad vx_fc, \quad vy_fc.$$

Hence, with this formulation the problem involves $3N+2$ variables and $2N+5$ equality constraints leaving $N-3$ degrees of freedom over which we optimize. Using $N=150$, LOQO solves this problem in 45 interior-point iterations (4.57 seconds on a 366 MHz PC). At optimality we have $x[N]=1027.383$ and $T=77.6699$. The control input as a function of time turns out to be constant as we hoped.

Now, let's consider a trapezoidal discretization. The AMPL model is shown in Figure 2. In this case, we have the following variables:

$$T, \quad x\{0, \dots, N\}, \quad y\{0, \dots, N\}, \quad vx\{0, \dots, N\}, \quad vy\{0, \dots, N\}, \quad c_L\{0, \dots, N\}$$

and the following equality constraints for the discretized problem:

$$x_eqn \{1..N\}, \quad y_eqn \{1..N\}, \quad vx_eqn \{1..N\}, \quad vy_eqn \{1..N\},$$

$$x_ic, \quad y_ic, \quad vx_ic, \quad vy_ic, \quad y_fc, \quad vx_fc1, \quad vy_fc1.$$

With this formulation, the problem involves $5N+6$ variables and $4N+7$ equality constraints leaving $N-1$ degrees of freedom over which we optimize. This is 2 more than with the previous model. Using $N=150$, LOQO solves this problem in 141 interior-point iterations (24.2 seconds on a 366 MHz PC). At

| | |
|---|---|
| <pre> param N; param x_0; param y_0; param vx_0; param vy_0; param x_n; param y_n; param vx_n; param vy_n; param cL_0; param cL_n; param cL_min; param cL_max; param c0; param k; param S; param rho; param m; param g; param W := m*g; var T >= 0; # State Variables var x {i in 0..N}; var y {i in 0..N}; # Control variables var cL {i in 1..N-1} >= cL_min, <= cL_max; # Abbreviations var vx {i in 0..N-1} = N*(x[i+1]-x[i])/T; var vy {i in 0..N-1} = N*(y[i+1]-y[i])/T; var ax {i in 1..N-1} = N*(vx[i]-vx[i-1])/T; var ay {i in 1..N-1} = N*(vy[i]-vy[i-1])/T; var cD {i in 1..N-1} = c0+k*cL[i]^2; var Vx {i in 1..N-1} = (vx[i]+vx[i-1])/2; var Vy {i in 1..N-1} = (vy[i]+vy[i-1])/2; var vr {i in 1..N-1} = sqrt(((vx[i]+vx[i-1])/2)^2 + Vy[i]^2); var D {i in 1..N-1} = .5*cD[i]*rho*S*vr[i]^2; var L {i in 1..N-1} = .5*cL[i]*rho*S*vr[i]^2; var sin_eta {i in 1..N-1} = Vy[i]/vr[i]; var cos_eta {i in 1..N-1} = Vx[i]/vr[i]; </pre> | <pre> maximize final_x: x[N]; s.t. newt_x{i in 1..N-1}: ax[i] = (-L[i]*sin_eta[i] - D[i]*cos_eta[i])/m; s.t. newt_y{i in 1..N-1}: ay[i] = (L[i]*cos_eta[i] - D[i]*sin_eta[i] - W)/m; s.t. novomit_x {i in 1..N-1}: -3 <= ax[i] <= 3; s.t. novomit_y {i in 1..N-1}: -3 <= ay[i] <= 3; # Boundary Conditions s.t. x_ic : x[0] = x_0; s.t. y_ic : y[0] = y_0; s.t. vx_ic: vx[0] = vx_0; s.t. vy_ic: vy[0] = vy_0; s.t. y_fc : y[N] = y_n; s.t. vx_fc: vx[N-1] = vx_n; s.t. vy_fc: vy[N-1] = vy_n; # Data which needs to be reinitialized data; param N := 150; param x_0 := 0; param y_0 := 1000; param vx_0 := 13.23; param vy_0 := -1.29; param y_n := 900; param vx_n := 13.23; param vy_n := -1.29; param cL_min := 0; param cL_max := 1.4; param c0 := 0.034; param k := 0.069662; param S := 14; param rho := 1.13; param m := 100; param g := 9.80665; # initial guess let {j in 0..N} x[j] := 5000*j/N; let {j in 0..N} y[j] := -100*j/N+1000; let {j in 1..N-1} cL[j] := .7; let T := 30; solve; </pre> |
|---|---|

FIGURE 1. The hang-glider range-maximization model using a midpoint discretization.

optimality we have $x[N] = 1027.488$ and $T = 77.7241$. After flying more than a kilometer, this optimal solution is better than the previous one by 10 centimeters. Perhaps this just reflects a difference in the discretization or maybe it is really a different answer. To see which it is, let's look at the control

| | |
|--|---|
| <pre> param N; param x_0; param y_0; param vx_0; param vy_0; param x_n; param y_n; param vx_n; param vy_n; param cL_0; param cL_n; param cL_min; param cL_max; param c0; param k; param S; param rho; param m; param g; param W := m*g; var T >= 10, <= 200; # State Variables var x {i in 0..N}; var y {i in 0..N}; var vx {i in 0..N} <= 30, >= -30; var vy {i in 0..N} <= 30, >= -30; # Control variables var cL {i in 0..N} >= cL_min, <= cL_max; # Abbreviations var cD {i in 0..N} = c0+k*cL[i]^2; var vr {i in 0..N} = sqrt(vx[i]^2 + vy[i]^2); var D {i in 0..N} = .5*cD[i]*rho*S*vr[i]^2; var L {i in 0..N} = .5*cL[i]*rho*S*vr[i]^2; var sin_eta {i in 0..N} = vy[i]/vr[i]; var cos_eta {i in 0..N} = vx[i]/vr[i]; var ax {i in 0..N} = (-L[i]*sin_eta[i] - D[i]*cos_eta[i])/m; var ay {i in 0..N} = (L[i]*cos_eta[i] - D[i]*sin_eta[i] - W)/m; maximize final_x: x[N]; # Trapezoidal Discretization s.t. x_eqn {j in 1..N}: (x[j]-x[j-1])/(T/N) = (vx[j]+vx[j-1])/2; s.t. y_eqn {j in 1..N}: (y[j]-y[j-1])/(T/N) = (vy[j]+vy[j-1])/2; s.t. vx_eqn {j in 1..N}: (vx[j]-vx[j-1])/(T/N) = (ax[j]+ax[j-1])/2; </pre> | <pre> s.t. vy_eqn {j in 1..N}: (vy[j]-vy[j-1])/(T/N) = (ay[j]+ay[j-1])/2; # Boundary Conditions s.t. x_ic : x[0] = x_0; s.t. y_ic : y[0] = y_0; s.t. vx_ic : vx[0] = vx_0; s.t. vy_ic : vy[0] = vy_0; s.t. y_fc : y[N] = y_n; s.t. vx_fcl: vx[N] = vx_n; s.t. vy_fcl: vy[N] = vy_n; s.t. monotone_x {i in 1..N}: x[i] >= x[i-1]; s.t. novomit_x {i in 0..N}: -3 <= ax[i] <= 3; s.t. novomit_y {i in 0..N}: -3 <= ay[i] <= 3; # Data which needs to be reinitialized data; param N := 150; param x_0 := 0; param y_0 := 1000; param vx_0 := 13.23; param vy_0 := -1.29; param y_n := 900; param vx_n := 13.23; param vy_n := -1.29; param cL_min := 0; param cL_max := 1.4; param c0 := 0.034; param k := 0.069662; param S := 14; param rho := 1.13; param m := 100; param g := 9.80665; # initial guess let {j in 0..N} x[j] := 5000*j/N; let {j in 0..N} y[j] := -100*j/N+1000; let {j in 0..N} vx[j] := 13.23; let {j in 0..N} vy[j] := -1.29; let {j in 0..N} cL[j] := 0.7; let T := 30; solve; </pre> |
|--|---|

FIGURE 2. The hang-glider range-maximization model using a trapezoidal discretization.

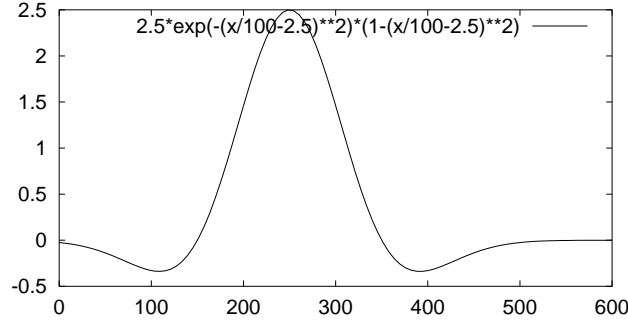


FIGURE 3. Updraft profile

input—Figure 4. From the control input we see that this solution is definitely not static. But it is also not implementable as it is discontinuous at $t = 0$ (followed by nontrivial control inputs for approximately the first 9 seconds of the flight).

The discontinuity of the control input suggests that the model formulation, i.e. the discretization, has too many degrees of freedom. To check this hypothesis, we tried introducing *continuity* constraints. First we added just one such constraint:

$$cL[0] = cL[1]$$

With this constraint, we got a solution that was closer to the static solution but was still not itself static. So, we added a second continuity constraint:

$$cL[1] = cL[2]$$

With these two constraints, the model solves in 222 interior-point iterations (41.1 seconds on a 366 MHz PC). At optimality we get $x[N] = 1027.383$ and $T = 77.6699$ in exact agreement with the static solution. Furthermore, the optimal control is again static.

It is noteworthy that the “correct” number of degrees of freedom for the adjusted trapezoidal rule matches the number of degrees of freedom from the midpoint rule. Clearly, something fundamental is going on here.

2.2. Unstable Airmass. The original problem studied in [5] involved an updraft 250 meters into the flight. The vertical velocity profile for this updraft is given by

$$u_a(x) = u_m e^{-\left(\frac{x}{R}-2.5\right)^2} \left(1 - \left(\frac{x}{R} - 2.5\right)^2\right)$$

and is shown in Figure 3. To change the model to account for this unstable airmass profile, we simply replace every occurrence of v_y in the model with $v_y - u_a(x)$. Both the midpoint discretization and the trapezoidal discretization (with the two c_L continuity constraints) are easy to solve. See Figures 5–10.

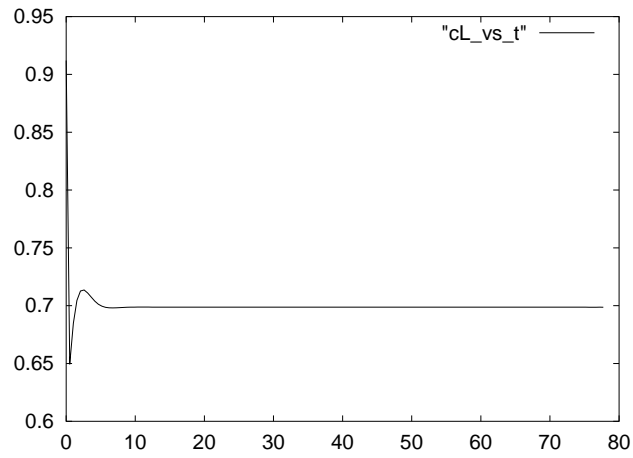
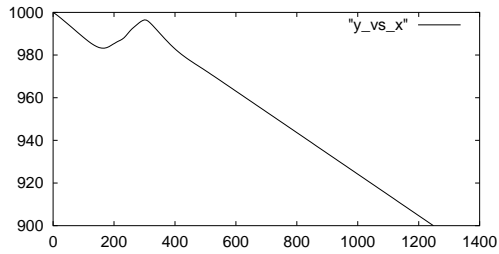
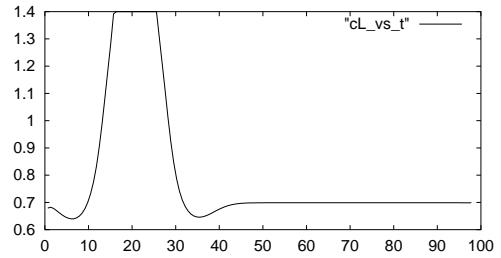
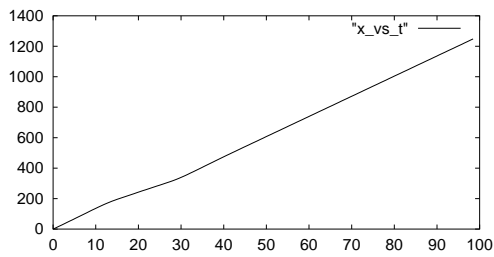
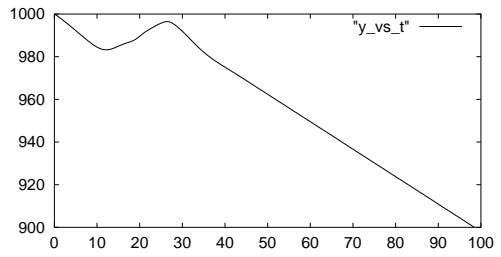
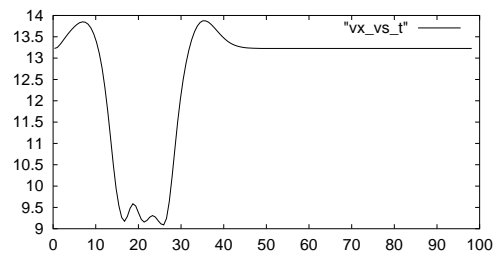
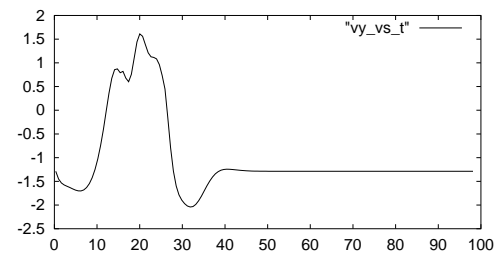


FIGURE 4. Control input as a function of time. Trapezoidal discretization method.

The trapezoidal discretization without the two extra continuity constraints exhibits the same anomalous behaviour as was illustrated by the stable air example. This version of the model appears in the COPS suite of problems [2]. The fact that it causes trouble for some solvers is documented in [3].

FIGURE 5. y vs x FIGURE 8. cL vs t FIGURE 6. x vs t FIGURE 9. y vs t FIGURE 7. v_x vs t FIGURE 10. v_y vs t

For more on optimal control of flight paths, we refer the reader to Stengel’s classic text [8] and to the recent book by Bryson [4].

2.3. **Lessons.** In this section there are two lessons:

- (1) Before solving a problem of interest, always test a model by first solving a problem whose solution is mathematically tractable.
- (2) The two discretization methods that we’ve discussed throughout this paper are commonly used for simple numerical integration of ODEs. In this case, the problem is well-posed if the number of equations matches the number of variables; i.e., there are no degrees of freedom. One can show that a problem is well-posed with respect to midpoint discretization if and only if it is well-posed with respect to trapezoidal discretization. However, as we saw in this example, for control problems, i.e. problems where there are degrees of freedom, the midpoint discretization will sometimes have fewer degrees of freedom than the trapezoidal discretization. It seems that the midpoint discretization has the “correct” number and that the trapezoidal discretization has too many.

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] A.S. Bondarenko, D.M. Bortz, and J.J. Moré. COPS: Constrained optimization problems. <http://www-unix.mcs.anl.gov/more/cops/>. 7
- [3] A.S. Bondarenko, D.M. Bortz, and J.J. Moré. COPS: Large-scale nonlinearly constrained optimization problems. Technical report, Technical Report ANL/MCS-TM-237, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne IL, 1998. Revised Oct 1999. 7
- [4] A.E. Bryson. *Dynamic Optimization*. Addison Wesley Longman, Inc., Menlo Park, CA, 1999. 9
- [5] R. Bulirsch, E. Nerz, H.J. Pesch, and O. von Stryk. Combining direct and indirect methods in optimal control: Range maximization of a hang glider. In R. Bulirsch, A. Miele, J. Stoer, and K.H. Well, editors, *Optimal Control: Calculus of Variations, Optimal Control Theory and Numerical Methods*, pages 273–288. Birkhauser Verlag, Basel, Boston, Berlin, 1993. 1, 2, 6
- [6] R. Fourer, D.M. Gay, and B.W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Scientific Press, 1993. 1
- [7] 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
- [8] R.F. Stengel. *Optimal Control and Estimation*. Dover, Mineola, NY, 1994. 9
- [9] R.J. Vanderbei. LOQO: An interior point code for quadratic programming. *Optimization Methods and Software*, 12:451–484, 1999. 1
- [10] R.J. Vanderbei. LOQO user’s manual—version 3.10. *Optimization Methods and Software*, 12:485–514, 1999. 1
- [11] 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