SIAG/OPT Views-and-News

A Forum for the SIAM Activity Group on Optimization

Volume 12 Number 1 March 2001

Contents

Case Studies

Optimization and Human Movement

Elizabeth Bradley
Department of Computer Science
University of Colorado
Boulder CO 80309-0430 USA
lizb@cs.colorado.edu

Joshua M. Stuart
Stanford Medical Informatics
Stanford University Medical Center
Stanford, CA 94305-5479 USA

Human movement is an ongoing optimization process. A baseball player attempts to contact a ball with a bat so as to propel the latter as far as possible; a rower tries to impart the maximum possible force to the water through an oar, while at the same time avoiding any disturbance to the boat's forward motion. The cost functions involved are complex, implicit, nonunique, fuzzy, and often subjective. Hundreds of muscles—most of which are not under conscious control—are involved in every motion, and different body geometries, coaches, choreographers, etc., prescribe different criteria for optimal movement. At the same time, the results are unmistakeable: the difference between breakdancing and ballet is patently obvious, even to the untrained eye.

Automatic generation of motion sequences is an interesting problem that has applications in graphics and animation, in training sequence generation, in gait analysis, and even in artistic innovation[2, 3, 7]. The moving picture industry, needless to say, has devoted tremendous numbers of cycles to this problem, but the algorithms involved rely on human animators to specify "keyframes" that act as skeletons for the movement. One can use mathematical interpolation techniques like splines to move individual body parts from one keyframe to another, but these kinds of methods do not address the problem of kinesiological illegality (e.g., that the knee only bends 180 degrees, or that arms cannot pass through ribcages). Many animation packages, such as Life Forms or Poser[1], use an augmented spline approach that relies on a table of kinematic constraints to avoid illegal movements, but this type of approach is somewhat *ad hoc*. One can also generate movement sequences by modeling the physics of the body—e.g., using differential equations and solving the corresponding boundary-value problem[9]. Physics-based animation approaches are extremely interesting and highly promising, but also very difficult; deducing the control equations that humans use to recover their balance after a jump, for example, is a Ph.D. thesis-level problem[13]. *Stylistically* faithful interpolations are even harder to implement;

[1] fas.sfu.ca/lifeforms.html and www.metacreations.com/products/poser3/

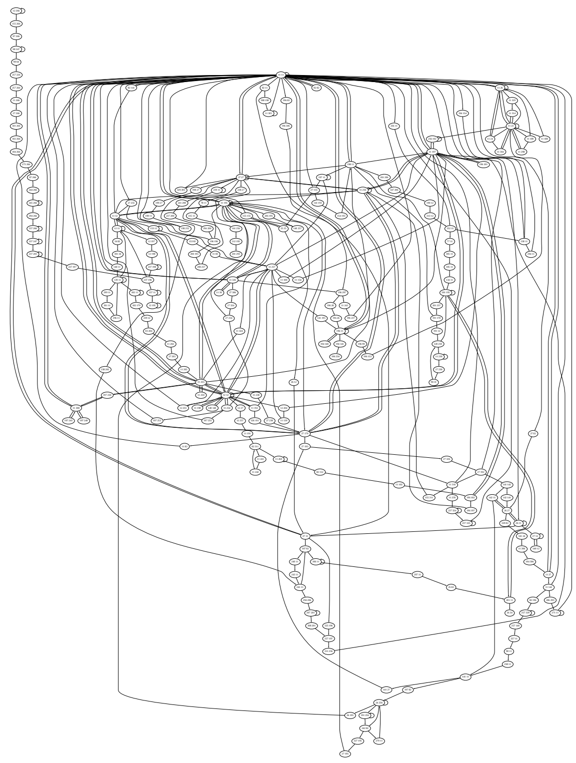

Figure 1: A *joint transition graph* that represents the movement patterns of the hips in a corpus of 38 short ballet pieces, comprising 1720 individual postures. The numbers in each state identify the discretized position of the joint. Edge weights and isolated vertices have been omitted in the interests of clarity. After [12].

neither splines nor $F = ma$ can easily capture or enforce, for instance, the requirement that classical ballet emphasizes position over motion[2], and developing a mathematics- or physics-based approach that does so would be all but impossible.

In this short paper, we describe an alternative solution to the "tweening" problem: a class of corpus-based schemes that generate physically consistent and stylistically consonant movement sequences between pairs of specified body positions. The computer program MotionMind, which instantiates these ideas, takes as input a corpus of movement sequences—e.g., ten Balanchine ballets—and a pair of body postures A and B; its output is a movement sequence that starts at A, ends at B, and fits the style of the corpus. If A and B are are "far apart," as measured by some metric that takes into account both the physics of the human body and the style of the movement genre, this can be nontrivial. MotionMind solves this problem by using statistical and graph-theoretic techniques to "learn" the grammar that is implicit in the corpus, and then applying simple heuristic search methods to the resulting graphs in order to generate movement sequences that are consistent with that grammar.

MotionMind simplifies the complex task of representing human motion by disregarding limb length. Each body posture is represented by a set of 23 quaternions—a common representational device in graphics that consists of a 3-vector and an angle of rotation around that vector[8]—each of which specifies the position of one of the body's main joints (omitting, e.g., finger and toe knuckles). To capture the movement patterns in a corpus, MotionMind examines that corpus joint by joint, building a directed, weighted graph for each one. Each vertex in these *joint transition graphs* represents a joint position (e.g., elbow bent to 10 degrees); edges repre-

[2]In ballet, body parts tend to describe piecewise-linear paths through space, emphasizing the positions at the junctions of those linear segments; in modern dance, on the other hand, the motion *between* the endpoints is often the important feature, and the choreography is crafted accordingly.

sent observed transitions between the corresponding positions, weighted using the negative log-likelihood: small values correspond to transitions that are more likely to occur. An example of such a graph is shown in figure 1. The intricate patterns of human movement are reflected by the complex topology of the graph. Note that joint angle is a continuous variable, which would imply a potentially infinite number of vertices; to avoid this problem, MotionMind discretizes the quaternion space (cf., snapping objects to a grid in graphics).

After building the set of 23 *joint transition graphs* that capture the movement grammar, MotionMind applies memory-bounded A* search[11] in order to find interpolation sequences. In general, A* finds a path from an initial state to a goal state by progressively generating successors of the current state in the search, computing a heuristic score that combines the existing path length and an estimate of the distance to the goal, and then expanding on a best-score-first basis. See [12] for more details. In this problem, the initial and goal "states" are actually 23 states in separate graphs, and MotionMind needs to search all 23 graphs in parallel (for a path from the knee angle in posture A to the knee angle in posture B, another path from the ankle angle in posture A to the ankle angle in posture B, and so on). One obvious choice of scoring function—which is based upon the assumptions that the sequence should be as short as possible and that common movements should be chosen over rare ones—is to minimize the sum of the weights of the edges in the path.

The basic idea here is fairly simple, but further consideration reveals a variety of important additional constraints. One really wants the lengths of all of those paths to be roughly equal, for example, in order that the different body parts arrive at the target posture at about the same time. Moreover, the search is complicated by the fact that joint positions cannot be interpolated in isolation: the movement patterns of the ankle, for instance, are strongly influenced by whether or not the foot is on the ground—information that is implicit in the positions of the pelvis, knees, etc. This requires that the expansion of nodes in the search be context dependent in a somewhat unusual way. MotionMind uses a Bayesian network[10], shown in figure 2, to model the constraints induced on joint motion by gravity and body topology. The pelvis is the root of this tree; three branches lead from this root to nodes corresponding to the right hip, the left hip, and the lower spine[3]. Each hip joint is the parent node to a knee, and so on. MotionMind assigns a conditional probability distribution, estimated from the corpus, to every (parent,child) pair in the tree, and models coordination by incorporating this number into the A* scoring function.

Figure 3 shows an example MotionMind sequence, computed using a ballet corpus. The starting and ending body postures (top left and top right in figure 3, labeled $\boxed{1}$ and $\boxed{10}$, respectively) are quite different; note the facing of the dancer and the weight distribution on the feet, for example. MotionMind's eight-move interpolation sequence moves between those positions in a very natural way. Its first move, for instance, is to lower the left leg, a natural strategy if one is going to change one's facing and end up on two feet. The following move is a simple weight shift (frames $\boxed{4}$ and $\boxed{5}$), in preparation for a lift of the right leg. This lift, which is not strictly necessary to move from the fifth frame to the tenth, is an innovation that the program inserted because of the observed patterns in the corpus; it reflects the fact that ballet dancers rarely spin with *both* feet flat on the ground. Perhaps the most interesting thing about this interpolation sequence, from a balletic standpoint, is the relévé[4] that the interpolation procedure inserted between frames $\boxed{6}$ and $\boxed{10}$. Many relévés appear in the corpus, but none of them are associated with upper body positions that resemble the one that appears in this sequence. MotionMind has invented a physically *and stylistically* appropriate way to move the dancer between the specified positions. The interpolation sequence in figure 3 includes a variety of other stylistically consistent innovations as well; consider, for example, the uplifted chest and chin in frames $\boxed{7}$ and $\boxed{9}$—posture elements that are quintessential ballet style. Recall that these postures were not simply

[3]The sacrum and the five lumbar vertebrae are lumped together. This compromise sacrifices back suppleness for lowered complexity.

[4]A relévé, which consists of lifting up on one's toes, is a stylistically required component of a direction shift in ballet.

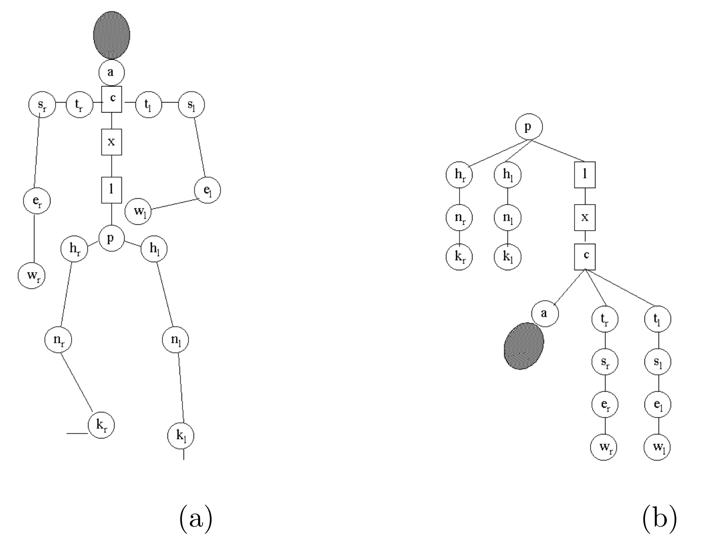

Figure 2: An influence diagram that explicitly represents the coordination of joints of the human body. Part (a) depicts the body and part (b) shows the inter-joint dependencies induced by gravity and topology: for instance, the position of the pelvis influences the positions of both hips h_r and h_l and the lumbar spine l, but the right and left ankles k_r and k_l do not directly influence one another. Without this simplifying assumption, the search space for this problem is intractable. After [12].

pasted in verbatim from the corpus; they were synthesized *joint by joint* using the transition graphs and influence-diagram directed A* search, and their fit to the genre is strong evidence of the success of the methods described in the previous section. `mpeg` movies of this sequence, along with many others, are available on the web[1].

MotionMind's algorithms have several interesting failure modes. Because of the directed nature of the graphs, the search algorithm sometimes has trouble finding interpolation subsequences between body positions that occur in inverted temporal order (e.g., reversing a baseball swing). Moreover, it often finds relatively long paths between positions that appear very similar; in one such instance, where the task was a simple 90-degree rotation of the right shoulder around the long axis of the arm, MotionMind constructed an 65-move sequence that involved much leg and trunk movement. Both of these problems are caused by limited corpus size. 1720 postures is an extremely meager sampling of human motion, so the resulting joint transition graphs are far from being connected, which means that some joint orientations are just not reachable from others. Even when the graphs *are* connected, the search may have to wander all over the graph to find a path between two given vertices. If the corpus were large and rich, the graphs would be highly connected, which would give the search algorithms more leeway. In the existing corpora, however, the paucity of edges constrains the search to very narrow (and long) paths that can translate to stilted, idiosyncratic movement sequences. This is an unavoidable problem in this application, unfortunately; the dance world has not yet embraced the notion of computer animation, so the availability of animated dances is quite limited, and motion-capture studios are expensive to set up and run. The third interesting failure mode arises from the greedy search strategy, which creates "inefficiencies" in the interpolation sequences—places where the dancer appears to be headed towards the goal state, but then moves away. For example, one of the interpolation goals in figure 3 is to change the figure's facing from left to right. By the fourth frame, the dancer has turned to the right, but in the fifth frame s/he has turned back to the left again, which is part of what necessitates the relévé sequence between frames $\boxed{6}$ and $\boxed{7}$. Finally, note that some search strategies—e.g., always taking the highest-probability branch—can be a significant source of cliché.

The primary motivation for the development of these methods was our work on a mathematical technique[4] that automatically creates variations on predefined motion sequences—an idea that was inspired by a similar scheme[5, 6] that uses a related

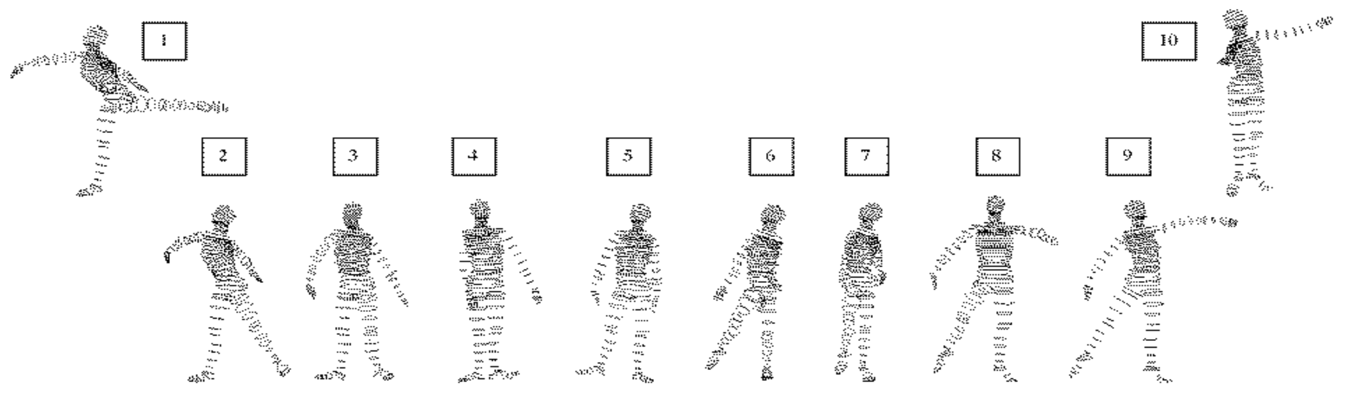

Figure 3: A "tweening" sequence generated by MotionMind. The starting and ending positions are shown at the top left and top right, respectively; the eight frames below them were computed by MotionMind. After [12].

procedure to generate *musical* variations. This approach uses the mathematics of chaos to shuffle a predefined movement sequence by "wrapping" that sequence around a chaotic attractor. This establishes a symbolic dynamics that links the movement progression and the attractor geometry, which one can then use to generate variations on that original piece. Variations generated in this manner, whether musical or choreographic, are both aesthetically pleasing and strikingly reminiscent of the original sequences. The stretching and folding of the chaotic dynamics guarantee that the ordering of the pitches or movements in the variation is different from the original sequence; at the same time, the fixed geometry of the attractor ensures that a chaotic variation of Bach's Prelude in C Major or of a short Balanchine ballet sequence are related to the original piece in a sense reminiscent of the classic "variation on a theme." Broadly speaking, the chaotic variations resemble the originals with some shuffling of coherent subsequences. This is the primary source of the stylistic originality of the chaotic variation scheme — in fact, this type of subsequence shuffling is a well-established creative mechanism in modern choreography. One problem with any choreographic technique, automated or not, that involves subsequence reordering, however, is that the transitions at the subsequence boundaries can be quite jarring, and the interpolation algorithms covered in this paper can smooth these kinds of transitions in a manner that is both kinesiologically and stylistically consistent.

The "goal" of choreography is aesthetic appeal, so it is difficult to analyze the results of this work using standard scientific criteria[5]. However, there are some standard rules, procedures, and patterns in certain dance and martial arts genres; as described elsewhere[12], analyses based on these criteria suggest that MotionMind's sequences are indeed stylistically consonant. Another interesting way to evaluate these results is to construct a Turing test: say, ten sequences generated by a human choreographer and ten MotionMind sequences, in randomized order. We have put together such a test and administered it to roughly 100 people. The results are mixed; most of MotionMind's sequences are indistinguishable from human-generated ones, but a few are awkward in an artificial and recognizeable way. This, in turn, brought out another interesting variable; students who are majoring in dance found this awkwardness esthetically appealing, while computer science majors did not.

By applying techniques from statistics, graph theory, and heuristic search, the corpus-based interpolation methods described in this paper automatically construct interpolation sequences that move from one specified body posture to another in a *physically and stylistically coherent* fashion. Though our objective in doing this was to tailor generic strategies for a specific high-dimensional search problem to an unusual and demanding domain, the results could certainly be extended to other domains where

[5]The very notion of objective, quantifyable evaluation elicited much consternation and mirth—along with some offense—from our dance colleagues.

the genre of sequence is important, such as speech recognition (e.g., filling in missing parts of a signal) or text. Finally, the implementation of these algorithms allows for arbitrary body topologies, so MotionMind is by no means limited to *human* motion sequences—though one would, of course, have to adapt the quaternion-based symbol set and the influence diagram to the topology of the limbs and joints that are involved.

REFERENCES

[1] www.cs.colorado.edu/~lizb/chaotic-dance.html.

[2] J. Birringer. *Media and Performance: Along the Border.* The Johns Hopkins University Press, 1998.

[3] E. Bradley, D. Capps, and A. Rubin. Computers and choreography. In *International Conference on Dance and Technology*, Tempe, AZ, 1999.

[4] E. Bradley, and J. Stuart. Using chaos to generate variations on movement sequences. *Chaos*, 8:800-807, 1998.

[5] D. Dabby. Musical variations from a chaotic mapping. *Chaos*, 6:95-107, 1996.

[6] D. Dabby. A chaotic mapping for musical and image variation. In *Proceedings of the Fourth Experimental Chaos Conference*, 1997.

[7] J. Dunning. How to tell the computer from the dance. *New York Times*, February, 1999.

[8] W. Hamilton. On a new species of imaginary quantities connected with a theory of quaternions. *Proceedings of the Royal Irish Academy*, 2:424-434, 1843.

[9] J. K. Hodgins, W.L. Wooten, D.C. Brogan, and J.F. O'Brien. Animating human athletics. In *Proceedings of SIGGRAPH*, 1995.

[10] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, 1988.

[11] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.

[12] J. Stuart and E. Bradley. Learning the grammar of dance. In *Proceedings of the International Conference on Machine Learning (ICML)*, 1998.

[13] W.L. Wooten. *Simulation of Leaping, Tumbling, Landing, and Balancing.* PhD thesis, Georgia Institute of Technology, 1998.

A Case Study in Trajectory Optimization: Putting on an Uneven Green

Robert J. Vanderbei
Operations Research and Financial Engineering
Princeton University
Princeton, NJ
Research supported by NSF grant DMS-9870317, ONR grant N00014-98-1-0036.

1. Introduction

The problem of how to putt provides a simple framework for a discussion of trajectory optimization. The discussion in this paper is taken from a longer paper [9] on case studies in trajectory optimization. The purpose of these studies is to illustrate how recent advances in algorithms and modeling languages now make it easy to solve once difficult optimization problems using off-the-shelve software. A secondary goal is to show that it is nonetheless still possible to make subtle errors in a model which will render it (a) more difficult than it needs to be or (b) infeasible or, worse, (c) feasible but giving the wrong answer. In the past, trajectory optimization problems were thought to be difficult to solve and when failures occured it was unclear whether they were due to bad algorithms or bad models. Today, one can say that failures are most likely due simply to bad models.

We express our optimization models in the AMPL modeling language [4]. 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 [7, 8, 10, 2], which implements an interior-point method for general nonlinear optimization and (b) SNOPT [5], which implements an active set strategy with a quasi-Newton method for the QP subproblem.

One of the lessons to be learned with the putting example is how easy it is to make a wrong model. With this in mind, we advise the interested golfer to read beyond the next section because the first model, right as it may appear, is wrong.

2. The Alessandrini Model

We begin with a discussion of the problem essentially as it appears in [1].

Given a golf ball sitting at rest on a putting green, the problem is to figure out how to hit the ball so that it will go into the cup. To make sure that it does not just skim over the cup and stop at some point far beyond, we try to have the ball arrive at the cup with the smallest speed possible.

The Normal Vector. We assume that the elevation of the green is given as $(x, y, z(x, y))$ and that its shape is given by $(x/a)^2 + (y/b)^2 \leq 1$. Two tangent vectors to the surface are provided by $(1, 0, \partial z/\partial x)$ and $(0, 1, \partial z/\partial y)$. By taking the cross product of these two vectors, we obtain an upward pointing normal vector to the surface:

$$(-\partial z/\partial x, -\partial z/\partial y, 1).$$

The normal force N exerted by the surface of the green on the golf ball must point in this direction and its magnitude must be such that the total force in this direction vanishes (to keep the ball rolling on the surface).

The Normal Force. Since the only forces that are not tangential to the green are the force of gravity and the normal force itself, we must have the projection of the force of gravity on the normal direction be exactly opposite to the magnitude of the normal force:

$$-mg(e_z \cdot N)/\|N\| = -\|N\|,$$

where m is mass of the ball, g is acceleration due to gravity, e_z is the unit vector pointing in the vertical direction, and of course N is proportional to the normal vector given above. From this relation, we get that

$$N_z = \frac{mg}{(\partial z/\partial x)^2 + (\partial z/\partial y)^2 + 1}$$

and that

$$N_x = -\partial z/\partial x N_z \qquad N_y = -\partial z/\partial y N_z.$$

Friction. There is friction between the ball and the green. It is assumed to be proportional to the normal force and to point in a direction opposite to the velocity:

$$F = -\mu\|N\|\frac{v}{\|v\|}.$$

Equations of Motion. If we denote the trajectory by $u(t) = (x(t), y(t), z(t))$, then the equations of motion are

$$\begin{aligned} v &= \dot{u} \\ a &= \dot{v} \\ ma &= N + F - mge_z. \end{aligned} \tag{1}$$

Boundary Conditions. The initial and final positions are known,

$$u(0) = u_0 \qquad \text{and} \qquad u(T) = u_f,$$

but the time T at which the final position is reached is a variable.

This problem can be cast as a (nonconvex) nonlinear optimization problem by discretizing the time interval $[0, T]$ into N small time segments and writing discrete approximations for the derivatives that appear in the model. There are many ways to do this. In this paper, we discuss two popular discretizations: midpoint discretization and trapezoidal discretization. We begin with the midpoint method. Letting `x[j]`, `y[j]`, and `z[j]` denote the positional coordinates at time jT/N, `j=0,1,...,N`, we define discrete approximations to the three components of velocity at the midpoint of each time interval as follows:

```
vx[j+0.5]=(x[j+1]-x[j])/(T/N),
vy[j+0.5]=(y[j+1]-y[j])/(T/N),
vz[j+0.5]=(z[j+1]-z[j])/(T/N),
```

`j=0,1,...,N-1`. Discrete approximations for acceleration are defined similarly:

```
ax[j] = (vx[j+0.5]-vx[j-0.5])/(T/N),
ay[j] = (vy[j+0.5]-vy[j-0.5])/(T/N),
az[j] = (vz[j+0.5]-vz[j-0.5])/(T/N),
```

`j=1,...,N-1`. The equations of motion given by (1), together with the boundary conditions, complete the constraints defining the model:

```
ax[j] = (Nx[j] + Fr_x[j])/m,
ay[j] = (Ny[j] + Fr_y[j])/m,
az[j] = (Nz[j] + Fr_z[j])/m - g.
```

Here, `Nx[j]`, `Ny[j]`, and `Nz[j]` are shorthand for

```
    Nz[j] = m*g/(dzdx[j]^2 + dzdy[j]^2 + 1),
    Nx[j] = -dzdx[j]*Nz[j],
    Ny[j] = -dzdy[j]*Nz[j]
```

and `Fr_x[j]`, `Fr_y[j]`, and `Fr_z[j]` are shorthand for the three components of friction along the trajectory. Our first ampl model for this problem is shown in Figure 1. In this particular instance the shape of the green involves two rather flat, but slightly sloped, sections with a smooth ramp between them. The ball is initially on the lower section and the cup is on the higher section, a difficult putt similar to the one Tiger Woods faced on the 18th hole in the final round of the 2000 PGA Championship. The function $z(x, y)$ we use to define this ramp is

$$z(x, y) = -0.3 \arctan(y) + 0.05(x + y).$$

Neither LOQO nor SNOPT was able to solve the model shown in Figure 1. When this happens, it is natural to suspect that the problem is infeasible. Why should the model in Figure 1 be infeasible? Alessandrini was able to solve supposedly the same model (using a different elevation function for the green). We tried several different surfaces and they all fail with all codes *except* when the surface is planar (including, of course, tilted planar surfaces). Every optimizer we tried is able to solve such planar problems easily. This proved to be a good hint that something is wrong with the model.

After much pondering, it occured to us that z is being specified in two ways—once as an explicit function of x and y and a second time as the solution to a differential equation. Since the differential equation is computed by a somewhat crude discretization, it is entirely possible that the two specifications are enough different from each other to render the model infeasible. So, we tried two things:

1. Removing from the model the explicit statement of how `z` depends on `x` and `y`. That is, we changed

   ```
   var z{i in 0..n} = -0.3*atan(y[i])
                      + 0.05*(x[i]+y[i]);
   ```

   to just

   ```
   var z{i in 0..n};
   ```

   (This, we later learned, is how Alessandrini formulated the problem.)

2. Removing from the model the part of the differential equation that relates to the `z` component of the trajectory. That is, we removed the constraints `newt_z`, `zinit`, and `zfinal`.

The first of these changes produces a model that solves easily while the second one appears still to be infeasible. Hence, we seem to be on to something but more errors may be lurking. The trajectory found with the elevation constraint removed is shown in Figure 2. This trajectory looks almost right except that it seems to go airborne in the early part of the trajectory and then tunnel into the grass in the final stages. The ball is clearly not staying on the green but instead is flying through the air to the cup. This indicates that our differential equation for z is wrong. And, if it is wrong, then the equations for x and y ought to be wrong as well.

But what is wrong? The derivation was straightforward—how could it possibly be wrong?

3. The Correct Putting Model

The key to understanding what is wrong with our implementation of the Alessandrini model is contained in the observation that the model in Figure 1 is solvable when and only when the surface of the green is planar. This suggests that the derivation is only valid for that case. What is different when the surface is not planar? Well, if you drive a car over the crest of a hill you feel lighter than normal (pun intended), whereas if you speed through a valley you feel heavier. The weight that one feels is the magnitude of the normal force. Hence, the magnitude of the normal force is not constant when the surface has hills and valleys. As you go through a valley, the magnitude of the normal force must be greater than nominal in order to accelerate you along the arc defining the upward bending curve.

From this discussion, it is easy now to see that the magnitude of the normal force must be such that it compensates both for the pull of gravity and for the out-of-tangent-plane acceleration along the path:

$$\|N\| = mg\frac{e_z \cdot N}{\|N\|} + m\frac{a(t) \cdot N}{\|N\|}.$$

```
param g := 9.8; # acc due to gravity
param m := 0.01; # mass of a golf ball
param x0 :=  1; # coords of start pt
param y0 :=  2;
param xn :=  1; # coords of ending pt
param yn := -2;
param n := 50; # num of time points
param mu;

var T >= 0; # total time for the putt
var x{0..n};  # coords of the traj
var y{0..n};

var z   {i in 0..n}
   = -0.3*atan(y[i])+0.05*(x[i]+y[i]);
var dzdx{i in 0..n}
   = 0.05;
var dzdy{i in 0..n}
   = -0.3/(1+y[i]^2) + 0.05;

# v[i] denotes the deriv at midpt of
# the interval i(T/n) to (i+1)(T/n).
var vx{i in 0..n-1} = (x[i+1]-x[i])*n/T;
var vy{i in 0..n-1} = (y[i+1]-y[i])*n/T;
var vz{i in 0..n-1} = (z[i+1]-z[i])*n/T;

# a[i] denotes the accel at midpt of
# the interval (i-0.5)(T/n)
# to (i+0.5)(T/n), i.e. at i(T/n).
var ax{i in 1..n-1} = (vx[i]-vx[i-1])*n/T;
var ay{i in 1..n-1} = (vy[i]-vy[i-1])*n/T;
var az{i in 1..n-1} = (vz[i]-vz[i-1])*n/T;

var Nz{i in 1..n-1}
   = m*g/(dzdx[i]^2 + dzdy[i]^2 + 1);
var Nx{i in 1..n-1} = -dzdx[i]*Nz[i];
var Ny{i in 1..n-1} = -dzdy[i]*Nz[i];
var Nmag{i in 1..n-1}
   = m*g/sqrt(dzdx[i]^2 + dzdy[i]^2 + 1);

var vx_avg{i in 1..n-1}
   = (vx[i]+vx[i-1])/2;
var vy_avg{i in 1..n-1}
   = (vy[i]+vy[i-1])/2;
var vz_avg{i in 1..n-1}
   = (vz[i]+vz[i-1])/2;
```

```
var speed{i in 1..n-1}
   = sqrt(vx_avg[i]^2 + vy_avg[i]^2
                      + vz_avg[i]^2);

var Frx{i in 1..n-1}
   = -mu*Nmag[i]*vx_avg[i]/speed[i];
var Fry{i in 1..n-1}
   = -mu*Nmag[i]*vy_avg[i]/speed[i];
var Frz{i in 1..n-1}
   = -mu*Nmag[i]*vz_avg[i]/speed[i];

minimize finalspeed:
   vx[n-1]^2 + vy[n-1]^2;

s.t. newt_x {i in 1..n-1}:
   ax[i] = (Nx[i] + Frx[i])/m;
s.t. newt_y {i in 1..n-1}:
   ay[i] = (Ny[i] + Fry[i])/m;
s.t. newt_z {i in 1..n-1}:
   az[i] = (Nz[i] + Frz[i] - m*g)/m;

s.t. xinit: x[0] = x0;
s.t. yinit: y[0] = y0;
s.t. zinit: z[0]
   = -0.3*atan(y[0])+0.05*(x[0]+y[0]);

s.t. xfinal: x[n] = xn;
s.t. yfinal: y[n] = yn;
s.t. zfinal: z[n]
   = -0.3*atan(y[n])+0.05*(x[n]+y[n]);

s.t. onthegreen {i in 0..n}:
   x[i]^2 + y[i]^2 <= 16;

let T := 1.5;

let mu := 0.07;
let {i in 0..n}
   y[i] := (i/n)*yn + (1-i/n)*y0;
let {i in 0..n}
   x[i] := y[i]^2/2;

solve;
```

Figure 1: A first AMPL model for the putting problem. Note that the variable `v[i]` in the model is the same as `v[i+0.5]` in the text.

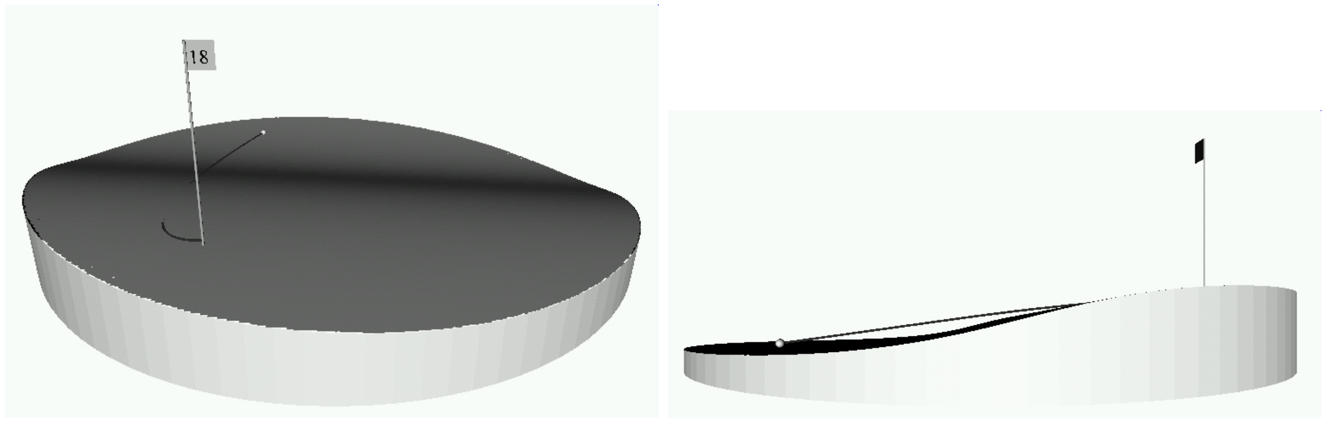

Figure 2: Two views of the trajectory obtained from the model in Figure 1 with the elevation constraint removed. *Note: For the online version of this paper, you can click on the figure to start a 3-D animation. In the animation, click on the flag to start the ball rolling.*

From this relation we can deduce that

$$N_z = m\frac{g - a_x(t)\frac{\partial z}{\partial x} - a_y(t)\frac{\partial z}{\partial y} + a_z(t)}{(\partial z/\partial x)^2 + (\partial z/\partial y)^2 + 1}.$$

Everything else in the previous derivation remains the same.

The complete correct model is shown in Figure 3. As shown in Figure 4, this trajectory does indeed follow the surface correctly (as it must given the model).

4. Trapezoidal Discretization

The second common discretization technique is called the *trapezoidal method*. With this method, values for velocity and acceleration are defined at the same discrete times as for position; that is, at `jT/N`, `j=0,1,...,N`. Instead of giving a formula defining each component of velocity in terms of a difference of the corresponding component of position, we give constraints that say that the average value at two adjacent times is equal to the appropriate difference:

```
(vx[i]+vx[i-1])/2 = (x[i]-x[i-1])/(T/n);
(vy[i]+vy[i-1])/2 = (y[i]-y[i-1])/(T/n);
(vz[i]+vz[i-1])/2 = (z[i]-z[i-1])/(T/n);
```

Constraints that must be satisfied by the components of acceleration are similar:

```
(ax[i]+ax[i-1])/2 = (vx[i]-vx[i-1])/(T/n);
(ay[i]+ay[i-1])/2 = (vy[i]-vy[i-1])/(T/n);
(az[i]+az[i-1])/2 = (vz[i]-vz[i-1])/(T/n);
```

The AMPL model for the trapezoidal discretization using the correct formulation of the putting problem is shown in its entirety in Figure 5. Both SNOPT and LOQO solve this formulation of the problem but each takes about twice as long as when solving the corresponding midpoint discretization formulation. Furthermore, LOQO requires a slight relaxation in the stopping criteria (the infeasibility tolerance needs to be increased from its default of 10^{-6} to 2×10^{-5}).

The fact that LOQO requires a relaxation in the stopping rule suggests that something might be wrong with the model. John Betts [3] seems to have identified the issue. He points out that the speed of the ball as it arrives at the cup is zero and hence there is a singularity in the differential equation at the final time. Of course, a numerical approximation might never experience the singularity exactly but it still can feel the effect. For the problem at hand, at the optimal solution LOQO has `speed[n] = 2.6e-6` and SNOPT has `speed[n] = 1.7e-6`. These values are not zero but they are getting close and one could imagine that numerical issues related to the singularity of the differential equation are beginning to enter in here. To test this, we changed the optimization objective from minimizing the final speed to minimizing the deviation of the final speed from some small prescribed value. In particular, we tried `(vx[n]^2 + vy[n]^2 - 0.25)^2`. With this objec-

```
param g := 9.8; # acc due to gravity
param m := 0.01; # mass of a golf ball
param x0 :=  1; # coords of start pt
param y0 :=  2;
param xn :=  1; # coords of ending pt
param yn := -2;
param n := 50; # num of time points
param mu;

var T >= 0; # total time for the putt
var x{0..n};  # coords of the traj
var y{0..n};

var z   {i in 0..n}
   = -0.3*atan(y[i])+0.05*(x[i]+y[i]);
var dzdx{i in 0..n}
   = 0.05;
var dzdy{i in 0..n}
   = -0.3/(1+y[i]^2) + 0.05;

# v[i] denotes the deriv at midpt of
# the interval i(T/n) to (i+1)(T/n).
var vx{i in 0..n-1} = (x[i+1]-x[i])*n/T;
var vy{i in 0..n-1} = (y[i+1]-y[i])*n/T;
var vz{i in 0..n-1} = (z[i+1]-z[i])*n/T;

# a[i] denotes the accel at the midpt of
# the interval (i-0.5)(T/n)
# to (i+0.5)(T/n), i.e. at i(T/n).
var ax{i in 1..n-1} = (vx[i]-vx[i-1])*n/T;
var ay{i in 1..n-1} = (vy[i]-vy[i-1])*n/T;
var az{i in 1..n-1} = (vz[i]-vz[i-1])*n/T;

var Nz{i in 1..n-1}
   = m*
     (g-ax[i]*dzdx[i]-ay[i]*dzdy[i]+az[i])
     /(dzdx[i]^2 + dzdy[i]^2 + 1);
var Nx{i in 1..n-1} = -dzdx[i]*Nz[i];
var Ny{i in 1..n-1} = -dzdy[i]*Nz[i];
var Nmag{i in 1..n-1}
   = m*
     (g-ax[i]*dzdx[i]-ay[i]*dzdy[i]+az[i])
     /sqrt(dzdx[i]^2 + dzdy[i]^2 + 1);
```

```
var vx_avg{i in 1..n-1} = (vx[i]+vx[i-1])/2;
var vy_avg{i in 1..n-1} = (vy[i]+vy[i-1])/2;
var vz_avg{i in 1..n-1} = (vz[i]+vz[i-1])/2;

var speed{i in 1..n-1}
   = sqrt(vx_avg[i]^2 + vy_avg[i]^2
                      + vz_avg[i]^2);

var Frx{i in 1..n-1}
   = -mu*Nmag[i]*vx_avg[i]/speed[i];
var Fry{i in 1..n-1}
   = -mu*Nmag[i]*vy_avg[i]/speed[i];
var Frz{i in 1..n-1}
   = -mu*Nmag[i]*vz_avg[i]/speed[i];

minimize finalspeed:
   vx[n-1]^2 + vy[n-1]^2;

s.t. newt_x {i in 1..n-1}:
   ax[i] = (Nx[i] + Frx[i])/m;
s.t. newt_y {i in 1..n-1}:
   ay[i] = (Ny[i] + Fry[i])/m;

s.t. xinit: x[0] = x0;
s.t. yinit: y[0] = y0;

s.t. xfinal: x[n] = xn;
s.t. yfinal: y[n] = yn;

s.t. onthegreen {i in 0..n}:
   x[i]^2 + y[i]^2 <= 16;

let T := 1.5;

let mu := 0.07;
let {i in 0..n} y[i] := (i/n)*yn + (1-i/n)*y0;
let {i in 0..n} x[i] := y[i]^2/2;

solve;
```

Figure 3: A second, and this time correct, AMPL model for the putting problem. This version is very similar to before—the main difference is in the definitions of `Nz` and `Nmag`.

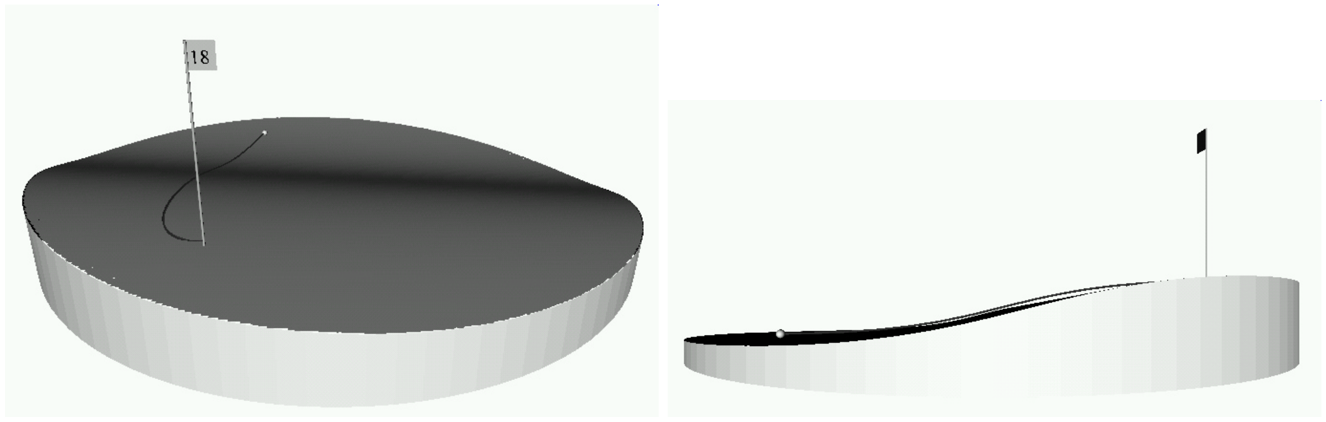

Figure 4: Two views of the trajectory from the correct model shown in Figure 3. Note how the trajectory follows the contour of the green.

tive function, both solvers are able to find a solution in a much more robust fashion (i.e., using fewer iterations and being successful over a wider range of choice of some of the other parameters in the problem). Our local golf expert (aka John Mulvey) indicates that this is the objective function used by real golfers anyway. He says that a real golfer does not want the ball to arrive at the cup with too little speed because then small imperfections in the green can have rather large unpredictable effects in those last few inches near the cup.

It is interesting to note that the midpoint rule is "less" bothered by the singularity issue. The reason is that the final speed in that model is the average final speed over the last time interval. This number is small but not as small as the final speed in the trapezoidal rule. For example, LOQO gets a final speed of `7e-3` with this discretization, which is a few orders of magnitude larger than it got with the trapezoidal rule.

5. Lessons

1. It is deceptively easy to formulate a problem incorrectly.

2. Incorrect formulations are surprisingly likely to be infeasible.

3. Infeasibility is especially hard for nonlinear solvers to detect reliably.

4. In the early days of optimization, a nonconvex problem with 10 or more variables was considered exceedingly hard to solve. In its most compact form, the problem here only really has 2 decision variables: the x and y components of the initial velocity vector that the putter imparts to the golf ball. After giving the ball its initial kick, the rest is determined by physics. One could formulate the problem this way. There would be just two decision variables and there would be a fairly complicated integrator function that would determine if the trajectory actually arrives at the hole and, if it does, the speed at which it arrives there. Using this integrator function as a "black box", one could make an optimization problem with just two variables. However, with modern optimization technology it is easy to incorporate the physics into the optimization model as we have done here and get a much larger model but one that is not any more difficult to solve. In fact, by expressing both the optimization part of the model and the physics in the same place and using the same "language" provides a level of model control that was totally lacking before. For example, if the physics is wrong, as it was in our first attempt, then the optimization problem is likely to be infeasible. If the physics and the optimization are separated from each other it is especially hard to identify what (or who!) is at fault. By having them

```
param g := 9.8; # acc due to gravity
param m := 0.01; # mass of a golf ball
param x0 :=  1; # coords of start pt
param y0 :=  2;
param xn :=  1; # coords of ending pt
param yn := -2;
param n := 50; # num of time points
param mu;

var T >= 0; # total time for the putt
var x{0..n};  # coords of the traj
var y{0..n};

var z   {i in 0..n}
   = -0.3*atan(y[i])+0.05*(x[i]+y[i]);
var dzdx{i in 0..n}
   = 0.05;
var dzdy{i in 0..n}
   = -0.3/(1+y[i]^2) + 0.05;

var vx{i in 0..n};
var vy{i in 0..n};
var vz{i in 0..n};

var ax{i in 0..n};
var ay{i in 0..n};
var az{i in 0..n};

var Nz{i in 0..n}
   = m*
     (g-ax[i]*dzdx[i]-ay[i]*dzdy[i]+az[i])
     /(dzdx[i]^2 + dzdy[i]^2 + 1);
var Nx{i in 0..n} = -dzdx[i]*Nz[i];
var Ny{i in 0..n} = -dzdy[i]*Nz[i];
var Nmag{i in 0..n}
   = m*
     (g-ax[i]*dzdx[i]-ay[i]*dzdy[i]+az[i])
     /sqrt(dzdx[i]^2 + dzdy[i]^2 + 1);

var speed{i in 0..n}
   = sqrt(vx[i]^2 + vy[i]^2 + vz[i]^2);
```

```
var Frx{i in 0..n}
   = -mu*Nmag[i]*vx[i]/speed[i];
var Fry{i in 0..n}
   = -mu*Nmag[i]*vy[i]/speed[i];
var Frz{i in 0..n}
   = -mu*Nmag[i]*vz[i]/speed[i];

minimize finalspeed: vx[n]^2 + vy[n]^2;

s.t. vx_def {i in 1..n}:
   (vx[i]+vx[i-1])/2=(x[i]-x[i-1])/(T/n);
s.t. vy_def {i in 1..n}:
   (vy[i]+vy[i-1])/2=(y[i]-y[i-1])/(T/n);
s.t. vz_def {i in 1..n}:
   (vz[i]+vz[i-1])/2=(z[i]-z[i-1])/(T/n);

s.t. ax_def {i in 1..n}:
   (ax[i]+ax[i-1])/2=(vx[i]-vx[i-1])/(T/n);
s.t. ay_def {i in 1..n}:
   (ay[i]+ay[i-1])/2=(vy[i]-vy[i-1])/(T/n);
s.t. az_def {i in 1..n}:
   (az[i]+az[i-1])/2=(vz[i]-vz[i-1])/(T/n);

s.t. newt_x {i in 0..n}: ax[i] = (Nx[i] + Frx[i])/m;
s.t. newt_y {i in 0..n}: ay[i] = (Ny[i] + Fry[i])/m;

s.t. xinit: x[0] = x0;
s.t. yinit: y[0] = y0;
s.t. xfinal: x[n] = xn;
s.t. yfinal: y[n] = yn;

s.t. onthegreen {i in 0..n}:
   x[i]^2 + y[i]^2 <= 16;

let T := 1.5;

let {i in 0..n} x[i] := (i/n)*xn + (1-i/n)*x0;
let {i in 0..n} y[i] := (i/n)*yn + (1-i/n)*y0;
let {i in 0..n} vx[i] := (xn-x0)/T;
let {i in 0..n} vy[i] := (yn-y0)/T;

let mu := 0.07;

solve;
```

Figure 5: The correct putting model with a trapezoidal discretization. Note how positions, velocities, and accelerations are all defined over the same index set.

together, it is easy to print out variables, trajectories, dual variables, etc. and all of this information can be useful in figuring out what is wrong with a model.

5. It wasn't mentioned in the discussion above, but one of the lessons in this example is how important it is to give an initial solution that is close to the optimal solution. For example, the optimal value of T is close to 2 in the examples above. We initialized T to be 1.5. Both LOQO and SNOPT find the right solution for any value of T between 1 and 3 but outside this range the solvers start to get into trouble. For example, neither of the solvers was able to solve the problem when initialized with $T = 5$.

6. Final Remarks

We have considered just one trajectory optimization problem—the putting problem. With this problem a number of issues came up that needed to be resolved. It turns out that these same issues are common in trajectory optimization problems. Hence, this example serves as a good prototype for trajectory optimization in general.

Finally, note that in preparing this case study we contacted Stephen Alessandrini to ask him about the fact that his model was incorrect. It turns out that when he derived the equations for his model his interest was in the planar case. It was only at the final stages of writing that he added a nonplanar example and he didn't realize the equations didn't apply. Interestingly, it was this last example that caught the eye of others, see for example [3], and for a time the incorrect model propogated unchecked.

This problem is not purely an academic exercise. See [6] for a description of a system in which putting trajectories were used for real-time animation during television coverage.

REFERENCES

[1] S.M. Alessandrini. A motivational example for the numerical solution of two-point boundary-value problems. *SIAM Review*, 37(3):423–427, 1995.

[2] 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.

[3] J.T. Betts. *Practical Methods for Optimal Control using Nonlinear Programming.* SIAM, Philadelphia, PA, 2000.

[4] R. Fourer, D.M. Gay, and B.W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming.* Scientific Press, 1993.

[5] 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.

[6] W.E. Lorensen and B. Yamrom. Golf green visualization. *IEEE Computer Graphics Appl.*, 12:35–44, 1992.

[7] R.J. Vanderbei. LOQO: An interior point code for quadratic programming. *Optimization Methods and Software*, 12:451–484, 1999.

[8] R.J. Vanderbei. LOQO user's manual—version 3.10. *Optimization Methods and Software*, 12:485–514, 1999.

[9] R.J. Vanderbei. Interior-Point Methods for Nonconvex Nonlinear Programming: Jamming and Comparative Numerical Testing. Technical Report ORFE-00-3, Dept. of Operations Research and Financial Engineering, Princeton University, Princeton NJ, 2000.

[10] R.J. Vanderbei and D.F. Shanno. An interior-point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13:231–252, 1999.

Bulletin

1st Annual McMaster Optimization Conference:
Theory and Applications
(MOPTA 01)
August 2-4, 2001, McMaster University
Hamilton, Ontario, Canada
http://www.cas.mcmaster.ca/~oplab/confs/mopta01/

The 1st annual McMaster Optimization Conference (MOPTA 01) will be held at the campus of McMaster University. It will be hosted by the Advanced Optimization Lab at the Department of Computing and Software and is co-sponsored by the Fields Institute.

SCOPE: The conference aims to bring together a diverse group of people from both discrete and continuous optimization, working on both theoretical and applied aspects. We aim to bring together researchers from both the theoretical and applied communities who do not usually get the chance to interact in the framework of a medium-scale event.

INVITED TALKS: Distinguished invited speakers include: Dimitri Bertsekas (MIT) John Dennis (Rice University) Ignacio Grossmann (Carnegie Mellon University) Don Jones (General Motors) Michael Todd (Cornell University) Lieven Vandenberghe (UCLA) David Williamson (IBM Almaden)

CONTRIBUTED TALKS: Each accepted paper will be alloted a 25 minute talk. Authors wishing to speak should submit an abstract in ASCII or LaTex source, to terlaky@mcmaster.ca by April 30, 2001. Please use "MOPTA" in the email subject line. Notification of acceptance / Program available: May 31, 2001. Deadline for early registration: June 30, 2001.

ORGANIZING COMMITTEE: Tams Terlaky, terlaky@mcmaster.ca (Chair)
Stavros Kolliopoulos (McMaster University)
Tom Luo (McMaster University)
Jiming Peng (Delft University of Technology)
Henry Wolkowicz (University of Waterloo)

Registration information is available at
http://www.cas.mcmaster.ca/∼oplab/confs/mopta01/

The Mathematical Programming Society and the Optimization Technology Center (Argonne / Northwestern) are pleased to announce "Optimization Online": a new web-based repository of e-prints on optimization and related topics available at

http://www.optimization-online.org

We encourage everyone in the optimization community to post technical reports on the site and to check out the latest reports by others. After many months of preparation, we believe that our site is ready to handle a tidal wave of submissions, so give us your best shot!

You can also sign up for a digest of submissions, to be emailed at the end of each month.

Submissions are moderated by members of a Coordinators Board, who check for correctness and completeness of the author-title-URL information but do not usually referee the content of each report. Authors may change their report listing and upload new versions of their reports as necessary. Authors are responsible for following applicable copyright laws, removing their report when required to do so by its publisher.

The scope of technical reports to be posted on Optimization Online roughly corresponds to that of the major journals in optimization. Reports are organized by a subject classification scheme; see site for details.

Please send comments and questions about Optimization Online to optonlin@mcs.anl.gov

Karen Aardal, Jean-Pierre Goux, Sanjay Mehrotra, Steve Wright Principal Coordinators, Optimization Online.

Comments from the Chair and Editor

I am honoured to have been chosen as the new Chair of the SIAM Activity Group on Optimization. I have been in touch with the rest of the board and we are all enthusiastic to continue making this an exciting group. I am looking forward to working with our new board:

Vice Chair: Philippe L. Toint
Program Director: Anders Forsgren
Secretary/Treasurer: Natalia M. Alexandrov
Newsletter: Juan Meza (leaving unfortunately)

We owe a debt of thanks to Tom Coleman for his excellent work as Chair. SIAG/OPT has thrived under his guidance. I hope to continue his good work. *A historical note: Tom was the fifth chair of SIAG/OPT* *following Jorge Moré. Tom got his doctorate in my department at the University of Waterloo under the supervision of Andy Conn, the third chair of SIAG/OPT. And Andy was preceded by John Dennis, an adjunct professor at University of Waterloo. So, other than Jorge and the first Chair Paul Boggs, I am following a Waterloo tradition in SIAG/OPT.*

We also need to thank Juan Meza for his excellent job as editor of our Newsletter. Unfortunately, Juan is not continuing on. We will announce his replacement as soon as possible.

Status

To get an idea of recent events, I looked at previous issues of the SIAG Optimization Newsletter at: http://csmr.ca.sandia.gov/~meza/siagopt/news.html. In particular, the following is from the Chairman's Column in No 9, of the Fall/97 issue:

> "The SIAG/OPT is alive and healthy. We continue growing; in 1995 the membership was 592, in 1996 we grew to 630, and now we are at 674. We are the fastest-growing SIAG by far, and the second largest SIAG."

Following are my comments stimulated by Jorge Moré's Chairman's column of 1997.

The SIAM Optimization meetings have been successful over the years. The next one will be held in Toronto, Canada, May 20-23, 2002, http://iris.gmu.edu/~asofer/opt2002.html. This is being held together with the thematic year *Numerical and Computational Challenges in Science and Engineering August 2001 to August 2002*, at the Fields Institute, URL: www.fields.utoronto.ca/programs/scientific/01-02/numerical/. As there will be many visitors in Toronto for the thematic year, the meeting promises to be doubly interesting. As is our custom, the SIAG/OPT Prize will be announced at the meeting and the winner will present a plenary talk.

The next two issues of SIAG/OPT newsletter, Views & News, are in the pipeline. We continue to solicit articles and suggestions from the membership.

The SIAG/OPT Web site http://www.siam.org/siags/siagopt.htm points to our own SIAG/OPT webpage. This was handled by Ariela Sofer and will be taken over by Natalia Alexandrov. We continue to encourage members with home pages to register their home pages with SIAM. Still, only a few members (about 160) have taken advantage of this offer. If you wish to be listed, send a message to Laura Helfrich at helfrich@siam.org with your name and the URL for your Web page. We will continue to make this web page useful and interesting.

The e-mail forum continues. You can use this forum for technical questions, announcements of papers, conferences, books, and software. In particular, technical questions are encouraged. Perhaps, this will lead to interesting discussions. You can use this forum by sending a message to opt@siam.org.

Multi-Media

My emphasis during my tenure as chair will be on providing information related to multi-media and mathematics, and extending the abilities of SIAG/OPT in disseminating this type information.

Several interesting related web sites are:

1. The Optimization Technology Center and the NEOS guide/server at http://www.ece.nwu.edu/OTC/ has to be the first choice in this list. Most people have now heard of this server, which solves optimization problems online.

2. The North American OpenMath Initiative at http://www.naomi.math.ca/, with information on a single standard for storing, transmitting, and manipulating mathematical information in online scientific documents and systems.

3. For those with access to ELECTRONIC TRANSACTIONS ON NUMERICAL ANALYSIS: etna.mcs.kent.edu/, see Volume 8, 1999, the article "On Gershgorin-type problems and ovals of Cassini", by Richard S. Varga and Alan Krautstengl. This article contains an interactive supplement that allows for graphic illustrations of Gershgorin disks and Cassini ovals. *It is hoped that our online newsletter will have such supplements.*

4. Optimization Online has started recently: http://www.optimization-online.org. This is a repository for eprints (see the bulletin section for more information).

5. The e-OPTIMIZATION.COMMUNITY at: http://e-optimization.com, provides information on algorithms, conferences, forum, etc... In particular, they provide live online conferences/presentations. We should participate, provide speakers, for this.

Thanks to this issue's Views-and-News authors - your contributions are very much appreciated! And to all those SIAG/OPT members who have not contributed to the Views-and-News in recent years, please consider writing a short expository article on your favorite topic. Controversy is welcome. Moreover, if you would like a thematic issue, then please make a suggestion!

Indeed, we are pleased to say that the next Views-and-News is underway and we will have another special edition with Urmila Diwaker who will be acting as our guest editor for an issue devoted to optimization under uncertainty.

Finally, if you haven't already done so, check out the Newsletter on the web. We're trying something new for this issue. All of the references to the web that appear within this document can now be accessed directly by clicking on the reference. Enjoy!

Henry Wolkowicz, SIAG/OPT Chair
University of Waterloo
Department of Combinatorics and Optimizatino
Waterloo, Ontario Canada **hwolkowicz@uwaterloo.ca**

Juan C. Meza, Editor
Sandia National Laboratories
P.O. Box 969, MS 9217
Livermore, CA 94551
meza@ca.sandia.gov
