Editor: Dianne P. O'Leary, oleary@cs.umd.edu

CLASSIFIED INFORMATION:
THE DATA CLUSTERING PROBLEM

By Nargess Memarsadeghi and Dianne P. O'Leary

MANY PROJECTS IN ENGINEERING AND SCIENCE REQUIRE DATA CLASSIFICATION BASED ON DIFFERENT HEURISTICS. DESIGNERS, FOR EXAMPLE, CLASSIFY AUTOMOBILE engine performance as acceptable or unacceptable based on a combination of efficiency, emissions, noise levels, and other criteria. Researchers routinely classify documents as "relevant to the current project" or "irrelevant." Genome decoding divides chromosomes into genes, regulatory regions, signals, and so on. Pathologists identify cells as cancerous or benign.

We can classify data into different groups by *clustering* data that are close with respect to some distance measure. In this project, we investigate the design, use, and pitfalls of a popular clustering algorithm, the *k*-means algorithm.

The Problem

For concreteness, we will cluster the pixels in the image shown in Figure 1. Suppose our original image is of size $m \times p$, with the color for each of the mp pixels recorded by b bits. Thus, the total storage requirement is mpb bits. We will choose k pixel values (colors) as *cluster centers* and map each pixel to one of these, which will form k clusters of pixels. This saves space (because we store the cluster index for each pixel instead of the pixel value) and, in addition, can filter out noise in the image.

The data for our sample problem is a 500×500 pixel image in jpeg format. For jpeg, the b bits for a pixel store $q = 3$ values (red, green, blue), each ranging between 0 and 255. We begin our investigation in Problem 1 by seeing how clustering reduces data storage.

Problem 1. Compare the number of bits required to store the original image and the clustered image.

We can state our clustering problem this way. Given n data points $\mathbf{x}_i \in \mathbb{R}^q$, $i = 1, \ldots, n$, and given a value of k, we want to find k cluster centers $\mathbf{c}_j \in \mathbb{R}^q$, $j = 1, \ldots, k$ that are in some sense optimal and then assign each data point to a cluster. We assign $\mathbf{x}_i$ to cluster $\mathbb{C}_j$ if it is closer to that cluster's center than it is to any other center. (Break ties in an arbitrary way.) The distance from data point i to its cluster's center is thus

$$d_i = \min_{j=1,\ldots,k} \left\| \mathbf{x}_i - \mathbf{c}_j \right\|,$$

and we define the radius of cluster $\mathbb{C}_j$ as

$$r_j = \max_{i:\mathbf{x}_i \in C_j} d_i .$$

For good clustering, we want each point to be close to one cluster's center. Therefore, we might want to minimize either

Figure 1. Use clustering algorithms to group the pixels of this image of Charlie. (Photograph by Timothy O'Leary.)

 Copublished by the IEEE CS and the AIP 1521-9615/03/$17.00 © 2003 IEEE

$$R = \sum_{j=1}^{k} r_j^{\ell},$$

or

$$D = \sum_{i=1}^{n} d_i^{\ell},$$

where $\ell = 1$ or 2. The variables in the minimization problem are the cluster centers.

Why the Problem Is Hard

In Problem 2, we consider some properties of the functions R and D.

Problem 2. For this problem, use the Euclidean norm with $q = 1$ and $\ell = 2$.

(a) If a function is *convex* and bounded below, then any local minimizer is a global minimizer. If not, then an algorithm for minimization might report a local minimizer (a point as good or better than any in its neighborhood) rather than a global one (a point as good or better than any other). Consider the problem with $n = 2$ points and $k = 2$ clusters. Are D and R convex functions?

(b) Are D and R differentiable functions when $n = 2$ and $k = 2$?

(c) Derive a formula for the minimizer of D when $k = 1$ and n is arbitrary.

(d) Suppose we move one of our data points $\mathbf{x}_i$ very far away from the other points, making it an *outlier*. As that point moves farther away from the others, what will happen to the cluster centers?

From Problem 2, we know that the minimization problem has some difficult properties, but let's try to compute the cluster centers using a standard optimization algorithm. To get the solution process started, we provide an array of k approximate centers. A common method for finding initial centers is to select k distinct points randomly among the data values, or perhaps to use k extreme values. When comparing algorithms, each should use the same initial centers. Problem 3 investigates our clustering criteria and the behavior of optimization algorithms.

Tools

In Problem 1, note that it takes $\log_2 256 = 8$ bits to store a number that ranges between 0 and 255.

For Problem 2, a real analysis text will provide information on convexity, differentiability, and minimization. A function $f(\mathbf{z})$ is convex over some convex domain if every one of its secants (line segments connecting two points on its graph) is on or above the graph.

Problem 3 relies on a program for unconstrained minimization, such as Matlab's `fminunc`. Stephen Nash and Ariela Sofer[1] discuss various algorithms for solving minimization problems.

Many sources present the *k*-means algorithm used in Problem 4 as well as other approaches to clustering.[2–5]

Ian Davidson[6] gives a nice discussion of the pitfalls we illustrate in Problems 5 and 6 as well as many others.

Many codes implement variants of the *k*-means algorithm; see, for example, Yuqian Guan's software (see www.cs.utexas.edu/users/yguan/datamining/gmeans.html/).

References

1. S.G. Nash and A. Sofer, *Linear and Nonlinear Programming*, McGraw-Hill, 1996.
2. A.K. Jain, M.N. Murty, and P.J. Flynn, "Data Clustering: A Review," *ACM Computing Surveys*, vol. 31, no. 3, 1999, pp. 264–323.
3. J.T. Tou and R.C. Gonzales, *Pattern Recognition Principles*, Addison-Wesley, 1974.
4. A. Jain and R.C. Dubes, *Algorithms for Clustering Data*, Prentice Hall, 1988.
5. K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd ed., Academic Press, 1990.
6. I. Davidson, *Understanding K-Means Non-hierarchical Clustering*, tech. report 02-2, Computer Science Dept., State Univ. of New York, Albany; www.cs.albany.edu/~davidson/courses/CSI635/UnderstandingK-MeansClustering.pdf.

Problem 3. Use your favorite optimization algorithm to minimize R with $\ell = 2$ and the Euclidean norm. Use Figure 1's data and provide a function to evaluate R. Try $k = 3, 4, 5$.

Also, minimize D with the same parameters.

(a) How does the number of variables increase with k?

(b) How does the running time increase with k?

(c) Evaluate the results of the four clusterings and justify the criteria that you choose. As one criterion, discuss how the clustered images look in comparison to the original image.

(d) How might we determine a good value of k experimentally?

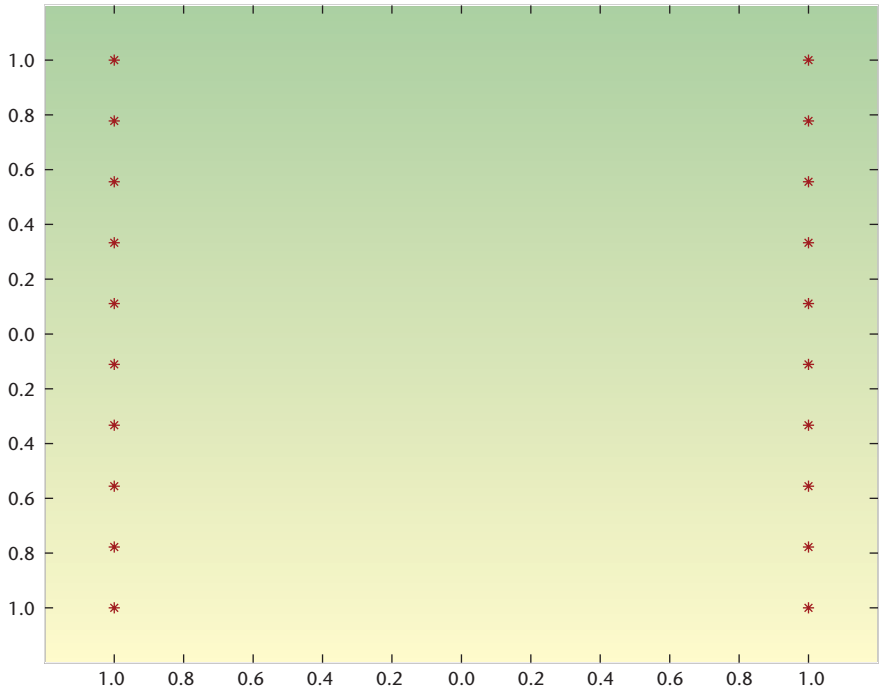

Figure 2. This data set illustrates some of the pitfalls of clustering. Compare the clusters you would form with those produced by the *k*-means algorithm.

Your implementation for this problem should be rather general: write a function `mycluster` that takes as input the n data values, an initial guess for the k cluster centers, a convergence tolerance, and a maximum number of iterations. The output will be assignments of each data point to a cluster, the set of k cluster centers, the number of data values in each cluster, and each cluster's radius. Use another function to evaluate R or D, given the k current cluster centers.

Write a function `map_to_cluster` that takes the data values and cluster centers as input and returns the cluster number for each data value and the counts of the number assigned to each cluster. Use this function to generate the clustered image.

To keep the computation time reasonable, determine the cluster centers based on a sample of points in the image rather than using all 250,000 pixels. Choose the 1,000 points in columns 210 and 211 in the sample image, then experiment with other choices of points to study the algorithm's sensitivity to this choice.

The *k*-Means Algorithm

A general-purpose minimization routine is a good tool to have, because it's useful for a wide variety of problems. But sometimes we can develop a better algorithm by taking advantage of a problem's special structure. The *k*-means algorithm minimizes neither D nor R, but it iterates by clustering based on the current centers and then moving each center to the centroid of the points in the cluster.

Choose initial centers $\mathbf{c}_1, \ldots, \mathbf{c}_k$. Repeat until the centers stop changing:

- For $i = 1, \ldots, n$, assign each data point $\mathbf{x}_i$ to the cluster $\mathbb{C}_j$ whose center $\mathbf{c}_j$ is closest to it, breaking ties in an arbitrary way.
- For $j = 1, \ldots, k$, recompute the center $\mathbf{c}_j$ to be the mean (centroid) of the points in the cluster:

$$\mathbf{c}_j = \frac{1}{n_j} \sum_{i:\mathbf{x}_i \in \mathbb{C}_j} \mathbf{x}_i ,$$

where n_j is the number of data points in $\mathbb{C}_j$.
In Problem 4, we implement this algorithm.

Problem 4. Implement the *k*-means algorithm and run it with the same data and values of k as Problem 3. Compare its performance to that of the algorithm in Problem 3.

Pitfalls in Data Clustering

This form of data clustering is quite useful, and the *k*-means algorithm is very successful in practice. Nevertheless, many pitfalls are associated with its use. We investigate two of these—dependence of the answer on the starting data and the number of clusters—in Problem 5.

Problem 5. Consider the data set of $n = 20$ data points with $q = 2$, shown in Figure 2:

$$(1, -1 + 2j/9),\ (-1,\ -1 + 2j/9),$$

for $j = 0, \ldots, 9$. Run the k-means algorithm with $k = 2$, 3, 4. Initialize the centers to the first k points in the list

$$(-1, -1),\ (1, 1),\ (-1, 1),\ (1, -1).$$

Display the clustered data. Discuss the effects of choosing the "wrong" value for k. Then, repeat the experiment, initializing the centers to $(0,\ -1 + 2j/(k-1))$, $j = 0, \ldots, k-1$. Note that although the answer is different, it is also a local minimizer of the (nonconvex) function R. Compare with the first set of answers and discuss the difficulty it illustrates with this kind of clustering.

Sensitivities of the clustering to the initial choice of centers and the number of clusters are serious pitfalls. As we see in Problem 6, another serious pitfall arises from the sensitivity of the clustering to variable transformations.

Problem 6. Consider the data set from Problem 5, but multiply the second component of each data point by 100. Repeat the clustering experiments, applying the same transformation to the initial centers. Discuss why coordinate scaling is important in clustering algorithms.

Through our investigations, we see that despite its pitfalls, clustering is an important tool for data classification, noise reduction, and storage savings. Check the Web site for data for the problems and, later, for sample solutions (http://computer.org/cise/homework/v5n5.htm).

Acknowledgments

Simon P. Schurr provided helpful comments on this project.

Partial Solution to "Robot Control: Swinging Like a Pendulum"

By Dianne P. O'Leary and Yalin E. Sagduyu

Problem 1. Consider the *undriven damped pendulum* modeled by

$$m\ell \frac{d^2\theta(t)}{dt^2} + c\frac{d\theta(t)}{dt} + mg\sin(\theta(t)) = u(t), \qquad (1)$$

when $u(t) = 0$ and $c > 0$. Linearize the second-order nonlinear differential equation using the approximation $\sin(\theta(t)) \approx \theta(t)$. Transform this equation into a first-order system of ODEs of the form $\mathbf{y}' = \boldsymbol{A}\mathbf{y}$, where $\boldsymbol{A}$ is a 2×2 matrix, and the two components of the vector $\mathbf{y}(t)$ represent $y_1(t) = \theta(t)$ and $y_2(t) = d\theta(t)/dt$. Determine the eigenvalues of $\boldsymbol{A}$. Show that the damped system is *stable*—that the real part of each eigenvalue is negative—and that the undamped system is not. Use the eigenvalue information to show how the solutions behave in the damped and undamped systems.

Answer: Under the transformation, Equation 1 becomes

$$\begin{bmatrix} 1 & 0 \\ c & m\ell \end{bmatrix} \begin{bmatrix} y'_1(t) \\ y'_2(t) \end{bmatrix} = \begin{bmatrix} y_2(t) \\ -mg\sin(y_1(t)) \end{bmatrix}$$

or

$$\begin{bmatrix} y'_1(t) \\ y'_2(t) \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ -c/(m\ell) & 1/(m\ell) \end{bmatrix} \begin{bmatrix} y_2(t) \\ -mg\sin(y_1(t)) \end{bmatrix}.$$

Replacing $\sin(y_1(t))$ with $y_1(t)$ gives the system

$$\mathbf{y}' = \begin{bmatrix} y'_1(t) \\ y'_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -g/\ell & -c/(m\ell) \end{bmatrix} \begin{bmatrix} y_1(t) \\ y_2(t) \end{bmatrix} = \mathbf{A}\mathbf{y}.$$

The eigenvalues of the matrix $\boldsymbol{A}$ are the roots of $\det(\boldsymbol{A} - \lambda\boldsymbol{I}) = 0$, or the roots of $\lambda^2 + \lambda c/(m\ell) + g/\ell = 0$:

$$\lambda_{1,2} = -\frac{c}{2m\ell} \pm \sqrt{\frac{c^2}{4m^2\ell^2} - \frac{g}{\ell}}.$$

For the undamped case, $c = 0$, so the real part of each eigenvalue is zero and the system is unstable. The real part of each eigenvalue is negative if $c > 0$, so in the damped case, the system is stable.

If $\lambda_1 \neq \lambda_2$, the eigenvectors of the matrix are

$$\begin{bmatrix} 1 \\ \lambda_1 \end{bmatrix}, \begin{bmatrix} 1 \\ \lambda_2 \end{bmatrix}$$

so the solution to the differential equation is

$$\mathbf{y}(t) = \alpha_1 \begin{bmatrix} 1 \\ \lambda_1 \end{bmatrix} e^{\lambda_1 t} + \alpha_2 \begin{bmatrix} 1 \\ \lambda_2 \end{bmatrix} e^{\lambda_2 t},$$

where α_1 and α_2 are constants determined by two additional conditions. If the discriminant satisfies $c^2/(4m^2\ell^2) - g/\ell > 0$, then the solution decays; otherwise it can have an oscillatory component in addition to a decaying one.

Problem 2. Consider the function

$$v(\theta, d\theta/dt) = \frac{(1-\cos\theta)g}{\ell} + \frac{1}{2}\left(\frac{d\theta}{dt}\right)^2$$

for the pendulum Equation 1 describes. Show that it is a valid positive definite Liapunov function for the undriven model. Investigate the stability of the solution $\theta(t) = 0$, $d\theta(t)/dt = 0$ for undamped and damped systems.

Answer: We note that $v(0, 0) = 0$, and it is easy to see that $v > 0$ for all other values of its arguments. Therefore, the point $\theta = 0$, $d\theta/dt = 0$ is stable for both the damped ($c > 0$) and undamped ($c = 0$) cases.

To investigate asymptotic stability, we differentiate

$$\begin{aligned}\frac{d}{dt}v(\mathbf{y}(t)) &= \frac{g\sin\theta(t)}{\ell}\frac{d\theta(t)}{dt} + \frac{d\theta(t)}{dt}\frac{d^2\theta(t)}{dt^2} \\ &= \frac{g\sin\theta(t)}{\ell}\frac{d\theta(t)}{dt} - \frac{d\theta(t)}{dt}\frac{1}{m\ell}\left(c\frac{d\theta(t)}{dt} + mg\sin(\theta(t))\right) \\ &= \frac{-c}{m\ell}\left(\frac{d\theta(t)}{dt}\right)^2 \leq 0.\end{aligned}$$

For the undamped case, this is identically zero, and we cannot conclude that we have asymptotic stability. For the damped case, we note that the set defined by $dv(\mathbf{y}(t))/dt = 0$ contains all points (θ, $d\theta/dt = 0$). The only invariant set is the one containing the single point (0,0) so this point is asymptotically stable.

Problem 3. Consider the linearized version of the driven (or forced), damped pendulum system with constant force term u. Transform the corresponding differential equation to a first-order ODE system of the form $\mathbf{y}' = \boldsymbol{A}\mathbf{y} + \boldsymbol{B}\mathbf{u}$. Specify the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ and show that the system is controllable for both the damped and undamped cases.

Answer: From Problem 1, we see that

$$\boldsymbol{A} = \begin{bmatrix} 0 & 1 \\ -g/\ell & -c/(m\ell) \end{bmatrix}, \qquad \boldsymbol{B} = \begin{bmatrix} 0 \\ 1/(m\ell) \end{bmatrix}.$$

Our dimensions are $n = 2$, $m = 1$, so the controllability matrix is

$$[\boldsymbol{B} \quad \boldsymbol{AB}] = \begin{bmatrix} 0 & 1/(m\ell) \\ 1/(m\ell) & -c/(m\ell)^2 \end{bmatrix}.$$

This matrix has rank 2, independent of c, so the system is controllable.

Problem 4. For the initial conditions $\theta(0) = \pi/4$ and $d\theta(0)/dt = 0$, use an ODE solver to find the numerical solutions on the interval $t = [0, 30]$ for the nonlinear model in Equation 1 for

1. an undamped ($c = 0$), undriven ($u = 0$) pendulum,
2. a damped ($c > 0$), undriven ($u = 0$) pendulum, and
3. a damped ($c > 0$), driven pendulum with the applied forces $u = mg\sin(\theta_f)$, where $\theta_f = \pi/8$, $\pi/4$, and $\pi/3$.

Repeat the same experiments for the pendulum's linearized model and discuss the difference in behavior of the solutions. It will help if you plot the $\theta(t)$ results for the corresponding linear and nonlinear models in the same figure.

Answer: See the program `problem4.m` on the Web page (http://computer.org/cise/homework/v5n5.htm). Figures A and B show the results. The models for the undamped

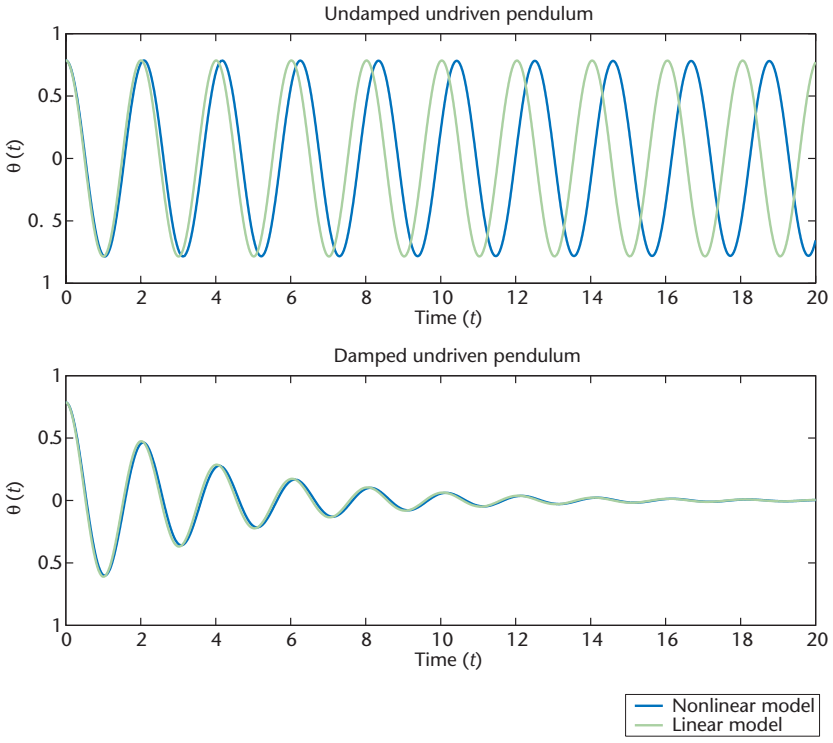

Figure A. The linear and nonlinear models for damped and undamped driven models.

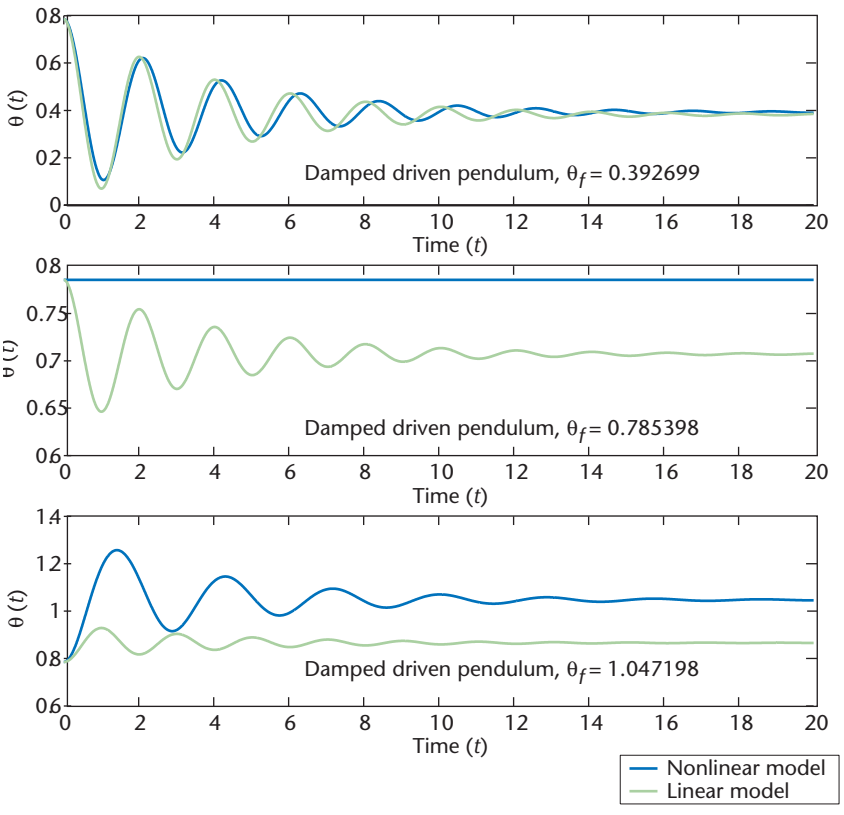

Figure B. The linear and nonlinear driven models.

undriven pendulum quickly show a phase difference in their results, while the damped undriven pendulum results are quite similar. For the driven pendulum, the linear and nonlinear results differ more as the angle θ_f gets bigger, and the linear models do not converge to θ_f.

Problem 5. Consider the linearized model of Equation 1 with constant applied force $u(t) = mg \sin(\pi/8)$ and damping constant $c = 0.5$. Suppose that we have the boundary conditions $\theta(0) = \pi/32$ and $\theta(10) = \theta_B$, where θ_B is the value of the solution when $d\theta(0)/dt = 0$. Apply the shooting method to find the solutions to the damped, driven, linearized pendulum equation on the time interval $t = [0,10]$. For the shooting method, use a nonlinear equation solver and an ODE solver for initial value problems. Try different initial guesses for $d\theta(0)/dt$ and compare the results.

Now use the finite-difference method to solve this boundary value problem with $h = 0.01$. Use your favorite linear system solver to solve the resulting linear system of equations.

Compare the results of the shooting and finite-difference methods with the solution to the original initial value problem.

Answer: See the program `problem5.m` on the Web page (http://computer.org/cise/homework/v5n5.htm). The $\theta(t)$ results for the original solution, shooting method, and finite-difference method differ by at most 0.004.

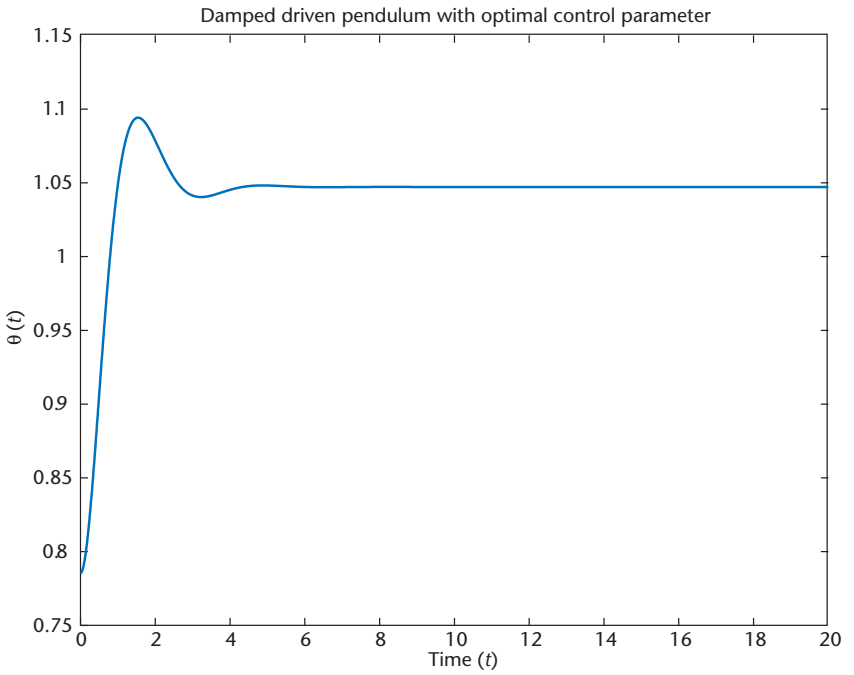

Figure C. The path of the robot arm with optimal control.

Problem 6. Consider the damped, driven pendulum with applied force

$$u(t) = mg\sin(\theta_f) + mlb\, d\theta(t)/dt,$$

where $\theta_f = \pi/3$. This force is a particular *closed-loop control* with control parameter b, and it drives the pendulum position to θ_f. The initial conditions are given as $\theta(0) = \pi/4$ and $d\theta(0)/dt = 0$. Assume $c = 0.5$ as the damping constant, $t_c = 5$ seconds as the time limit for achieving the position θ_f, and $h = 0.01$ as the time increment for numerical solutions.

We will call a parameter b successful if the pendulum position satisfies $|\theta(t) - \theta_f| < 10^{-3}$ for $5 \le t \le 10$. Approximate the total energy by

$$\hat{e}_f \approx \sum_{k=1}^{5/h} |u(kh)|h\,.$$

Write a function that evaluates $\hat{e}_f$. The input to the function should be the control parameter b and the output should be the approximate total consumed energy $\hat{e}_f$.

For stability of the closed-loop control system, we impose the constraint $b < c/(ml)$, which makes the real parts of the eigenvalues (of the linearized version) of the system strictly negative. Now use your favorite constrained minimization solver to select the control parameter b to minimize the energy function $\hat{e}_f(b)$. Display the optimal parameter and graph the resulting $\theta(t)$.

Answer: See the program `problem6.m` on the Web page (http://computer.org/cise/homework/v5n5.htm). The energy function returns the energy as specified earlier plus a large positive *penalty term* in case the parameter is unsuccessful; the penalty keeps the minimizer from choosing an unsuccessful parameter. For $b = -1.7859$, the total energy consumed is about 43.14 Joules. Figure C shows the pendulum's motion. Note that it is always a good idea to sketch the function to be minimized to see if the reported solution is reasonable.

Nargess Memarsadeghi is a graduate student in the Computer Science Department at the University of Maryland; she is also an employee of NASA Goddard Space Flight Center. She received her BS in computer science from the University of Maryland at College Park. Contact her at nargess@cs.umd.edu; www.cs.umd.edu/nargess.

Dianne P. O'Leary is a professor of computer science and a faculty member in the Institute for Advanced Computer Studies and the Applied Mathematics Program at the University of Maryland. She received a BS in mathematics from Purdue University and a PhD in computer science from Stanford. She is a member of SIAM, ACM, and AWM. Contact her at oleary@cs.umd.edu; www.cs.umd.edu/users/oleary/.

Yalin E. Sagduyu is a PhD student in electrical engineering and a research assistant in the Institute for Systems Research at the University of Maryland. He received a BSc in electrical engineering from Bogazici University, Istanbul, Turkey, and an MSc in electrical and computer engineering from the University of Maryland, College Park. Contact him at sagduyuy@eng.umd.edu; www.ece.umd.edu/~sagduyuy.