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

MULTIDIMENSIONAL INTEGRATION: PARTITION AND CONQUER

By Dianne P. O'Leary

UNDERSTANDING THE BEHAVIOR OF PARTICLES SUBJECTED TO FORCES IS A BASIC THEME IN PHYSICS. THE SIMPLEST SYSTEM IS A SET OF PARTICLES CONFINED TO MOTION along a line, but even this type of system presents computational challenges. For the harmonic oscillator, a particle is subjected to a force directed toward the origin and proportional to the distance between the particle and the origin. The resulting potential is $V(x) = 1/2\alpha x^2$, where α is a constant. This system is quite thoroughly understood, and quantities of interest can be computed in closed form. Alternatively, the Ginzburg-Landau anharmonic potential, $-1/2\alpha x^2 + 1/4\beta x^4$ (α and β are constant), is related to solution of the Schrödinger equation, and quantities of interest are computed approximately. One method of obtaining such approximations is *numerical integration*, which is our focus in this assignment.

Partition Functions

A system in thermodynamic equilibrium can be characterized by its *partition function*, an expression for the expected value of e raised to a power equal to the energy of the system divided by a normalization parameter αT. From the partition function, we can compute many quantities of interest—the expected value of the energy, the *free energy*, the entropy—so it's quite an important function.

Consider a ring of particles in which each particle interacts with its two neighbors. For this set of particles, the partition function is

$$Z_d(L) = \int_{-\infty}^{\infty} \rho(a, a, L)da, \quad (1)$$

where $L = 1/(\alpha T)$, T is the temperature, α is the Boltzmann constant, d is the number of particles, and ρ is

$$\rho_d(a, b, L) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} g(a, x_1, L)g(x_1, x_2, L) \cdots g(x_{d-1}, x_d, L)g(x_d, b, L)dx_1dx_2 \cdots dx_d,$$

where

$$g(x, y, L) = \frac{1}{\sqrt{2\pi\delta}} \exp\left(-\frac{1}{2\delta}(x-y)^2 - \frac{1}{2}\delta\big(V(x)+V(y)\big)\right)$$

with $\delta = L/(d+1)$. Let's develop some algorithms for approximating $Z_d(L)$.

Numerical Integration Methods

Many methods produce excellent estimates of the value of the integral

$$I = \int_{\Omega} f(\mathbf{x})d\mathbf{x}$$

when f is a smooth function of a single variable x and Ω is a finite interval $[a, b]$. For instance, we can partition the interval into small subintervals, construct a polynomial approximation to f in each subinterval by evaluating f at several points in the subinterval, approximate the integral over the subinterval by the polynomial's integral, and then sum these approximate values. If the function slowly changes over some pieces of Ω, we can reduce the error in the approximation by taking longer subintervals there and concentrating our work on regions where the function changes more rapidly.

For multidimensional integrals, however, the situation is much less satisfactory: the approach that is so successful in

Numerical-analysis textbooks provide excellent advice on approximating integrals over low-dimensional spaces, but many problems of interest are naturally posed in high dimensions and yield integrals over regions with a large number of variables. These problems arise, for example, in evaluating the failure rate of materials, the expected return on an investment, or the expected value of the energy of a system with a large number of particles. In this issue, we study some algorithms appropriate for these problems, using a system of particles as a motivating example.

one dimension quickly becomes prohibitive. Suppose, for instance, that $\mathbf{x} \in \mathcal{R}^{10}$ and Ω is the unit hypercube $[0, 1] \times \ldots \times [0, 1]$. Then, as an example, a polynomial of degree 2 in each variable would have terms of the form

$$x_1^{\square} x_2^{\square} x_3^{\square} x_4^{\square} x_5^{\square} x_6^{\square} x_7^{\square} x_8^{\square} x_9^{\square} x_{10}^{\square},$$

where the number in each box is 0, 1, or 2. So the polynomial has $3^{10} = 59{,}049$ coefficients, which means we would need 59,049 function values to determine them. If we partition the domain by dividing each interval $[0, 1]$ into five pieces, we make $5^{10} \approx 10$ million boxes, and we need 59,049 function evaluations in each. Clearly, this method is expensive!

On the other hand, evaluating integrals this way is quite easy to *program* if we have a function `quad` that works for functions of a single variable, perhaps using a polynomial approximation method. We can use `quad` to compute

$$\int_0^1 h(x_1)dx_1,$$

where

$$h(z) = \int_0^1 \ldots \int_0^1 f(z, x_2, \ldots, x_{10})dx_2 \ldots dx_{10},$$

as long as we can evaluate $h(z)$ for any given value $z \in [0, 1]$. But $h(z)$ is just an integration, so we can evaluate it using `quad`, too: we end up with 10 nested calls to `quad`. Again, this is very expensive, but it's quite convenient.

As an alternative, some functions $f(x)$ can be well approximated by a separable function

$$f(x) \approx f_1(x_1) f_2(x_2) \ldots f_{10}(x_{10}).$$

In that case, we can approximate our integral by

$$I \approx \int_0^1 f_1(x_1)dx_1 \ldots \int_0^1 f_{10}(x_{10})dx_{10}.$$

If this works, then great, but we aren't often that lucky.

We need another option—the methods we've discussed are either too expensive or only special-purpose. If the function is not well-approximated by a separable function, the last resort is some variant on *Monte Carlo integration*.

Monte Carlo Integration

There are two simple ways to use Monte Carlo integration. For the first, think of integration as computing the volume of some solid Ω, and embed that solid in a larger one Γ for which computing the volume is easy. As a trivial example, consider computing the area of the quarter circle Ω illustrated in Figure 1, and embed Ω in a square Γ of side 0.5. Then generate a sequence of random points in Γ, and determine the fraction ν of points that also lie in Ω. We can estimate the volume of Ω by ν times the volume of Γ. Here's a summary of our first Monte Carlo algorithm, Method 1:

- Let $\Gamma \subset \mathcal{R}^d$ be a set that contains Ω and has known volume $\mathcal{J}$.
- Generate n randomly distributed points $\{\mathbf{z}^{(i)}\} \in \Gamma$.
- Let $\hat{n}$ be the number of these points that also lie in Ω, and estimate the volume of Ω by

$$\nu_n = \frac{\hat{n}}{n}\mathcal{J}.$$

Another viewpoint gives a somewhat better algorithm, one that obtains more information from function values. Again, the idea is simple: the integral

$$I = \int_\Omega f(x)dx$$

is equal to the average value of f on Ω multiplied by the volume of Ω. This gives us a second method, Method 2:

- Generate n points $\{\mathbf{z}^{(i)}\} \subset \Omega$. For our 10-dimensional hypercube example, this requires generating $10n$ random numbers, uniformly distributed in $[0, 1]$.
- The average value of f in the region Ω is then approximated by

$$\mu_n = \frac{1}{n}\sum_{i=1}^{n} f\left(\mathbf{z}^{(i)}\right),$$

so the value of the integral is $I \approx \mu_n \int_\Omega d\mathbf{x}$.

How good is this approximation I_n? The expected value of I_n is the integral's true value—very nice! In fact, if the volume of Ω is normalized to 1, then for large n, the quantities $\sqrt{n}\ (I - I_n)/\sigma$ have an approximately normal distribution with mean 0 and variance 1, where

$$\sigma^2 = \int_\Omega (f(x) - I)^2 d\mathbf{x}\,.$$

Note that the variance is a constant that depends on the variation in f around its average value, but not on the dimension d of the integration domain Ω.

Figure 1 illustrates Methods 1 and 2.

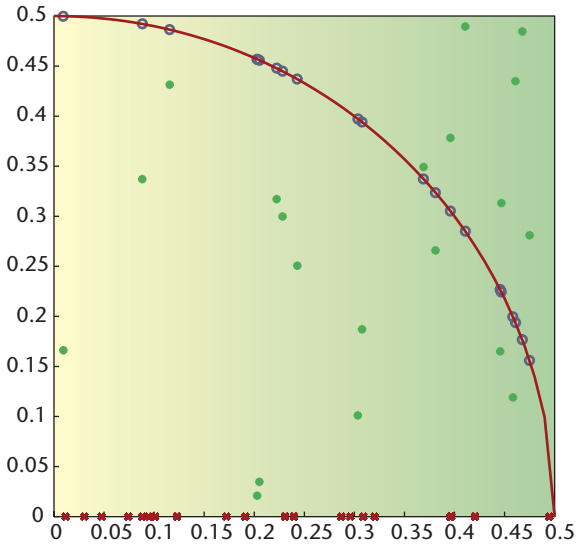

Figure 1. Estimating the area 0.19636 of a quarter circle using Monte Carlo methods with 20 function evaluations. Method 1 gives 13 of the 20 green stars inside the circle, for an estimate of 0.163. Method 2 averages the 20 blue function values to give a somewhat better estimate of 0.174. Method 3 averages function values corresponding to the red x's to yield 0.192.

Let's see how well these methods behave on a sample problem—finding the area of the quarter circle illustrated in Figure 1. But keep in mind that these methods are valuable for very high-dimensional integrals, not the trivial one that we use in Problem 1.

PROBLEM 1.

Suppose we want to estimate the area of a quarter circle with radius r:

$$I = \int_0^r \sqrt{r^2 - x^2}\, dx = \int_0^r f(x)dx.$$

Let $r = 0.5$ and use two methods:

- Method 1, testing whether randomly generated points in the rectangle $[0, r] \times [0, r]$ are inside or outside the quarter circle Ω and multiplying the fraction inside by r^2, the area of the rectangle.
- Method 2, computing the average value of $f(x)$ and multiplying by r, the length of $\Omega = [0, r]$.

Compare the quality of the two estimates for $n = 10$, 100, 1,000, 10,000, and 100,000 points by measuring the error and the convergence rate.

How many function evaluations does your favorite integration routine use to get an estimate of comparable quality?

Importance Sampling

The expected value of our estimate from either of the two methods is equal to the value we're looking for, but there's a nonzero variance to our estimate: we aren't likely to get the integral's exact value. Most of the time, though, the value will be close—if n is big enough.

If we could reduce our estimate's variance, we could get by with a smaller n. This would mean less work and can be accomplished by *importance sampling*.

Suppose we want to estimate

$$I = \int_\Omega f(\mathbf{x})d\mathbf{x},$$

where Ω is a region in $\mathcal{R}^{10}$ with volume equal to one. Our Monte Carlo estimate of this integral involves taking uniformly distributed samples from Ω and taking the average value of $f(\mathbf{x})$ at these samples. We can improve on this by a good choice of a function $p(\mathbf{x})$ satisfying $p(\mathbf{x}) > 0$ for all $\mathbf{x} \in \Omega$, normalized so that

$$\int_\Omega p(\mathbf{x}) = 1.$$

Then,

$$I = \int_\Omega \frac{f(\mathbf{x})}{p(\mathbf{x})} p(\mathbf{x})d\mathbf{x}.$$

We can get a Monte Carlo estimate of this integral by taking samples from the distribution with probability density $p(\mathbf{x})$ and taking the average value of $f(x)/p(\mathbf{x})$ at these samples; we call this Method 3.

When will Method 3 be better than Method 2?

Recall that the variance of our error is proportional to

$$\sigma^2 = \int_\Omega \left(\frac{f(\mathbf{x})}{p(\mathbf{x})} - I \right)^2 p(\mathbf{x})d\mathbf{x}.$$

So, if we chose p so that $f(x)/p(\mathbf{x})$ is close to constant, σ would be close to zero! (For this to be true, we don't want $f(\mathbf{x})$ to change sign on Ω.)

Intuitively, why does importance sampling work?

- In regions where $f(x)$ is big, $p(\mathbf{x})$ will also be big, so there's a high probability that we'll sample from these regions.
- In regions where $f(x)$ is small, $p(\mathbf{x})$ will also be small, so we won't waste time sampling from regions that don't contribute much to the integral.

Figure 1 illustrates this.

The big unanswered question is how to get a good choice for $p(\mathbf{x})$. If f is nonnegative, then we can do this by sampling $f(x)$ at a few points and setting p to be piecewise constant, with value proportional to the sampled values. For our unit hypercube example, we could divide our domain into 3^{10} smaller cubes of size 1/3 on which p will be constant. If we evaluate our function f at the center of each of these, we get a grid of 3^{10} values at mesh points $\mathbf{w}_i$, with each component of $\mathbf{w}_i$ equal to 1/6, 1/2, or 5/6. We set $p(\mathbf{x})$ in each cube to be the sampled value f at the center of the cube, divided by the sum of the sampled values. For our integration, among our n random values, we could take the number of values in the ith cube to be $np(\mathbf{w}_i)$.

Let's repeat our experiment using importance sampling.

PROBLEM 2.

Compute new estimates of the integral of Problem 1 using Method 3, taking 10 samples of $f(x)$ to determine the function $p(x)$. Compare with your previous results.

Using Quasi-Random Numbers

There are many uses of random numbers; the properties we need from them depend on our ultimate intended use. For example, simulation might require the numbers to be as independent of each other as possible, but in Monte Carlo integration, we just want the proportion of points in any region to be proportional to the volume of that region. This property is actually better achieved by correlated points; many proposals for generating *quasi-random* sequences guarantee a rather even distribution of points. For example, the Van der Corput sequence generates the kth coordinate of the pth quasi-random number $\mathbf{w}_p$ in a very simple way. Write out the base-b_k representation of p, where b_k is the kth prime number:

$$p = \sum_i a_i b_k^i,$$

and let the coordinate be

$$w_p(k) = \sum_i a_i b_k^{-i-1}.$$

You might think that a regular mesh of points also has this uniform covering property, but it's easy to see (by drawing the picture) that large boxes are left unsampled if we choose a mesh. The Van der Corput sequence, however, gives a sequence that rather uniformly covers the unit hypercube with samples, as Problem 3 demonstrates.

PROBLEM 3.

Generate 500 pseudo-random points $\mathbf{v}_p$ in $\mathcal{R}^2$ and 500 quasi-random points $\mathbf{w}_p$ in $\mathcal{R}^6$ (p = 1, …, 500). Plot the pseudo-random points. Then for the quasi-random points, make a plot of the first two coordinates, a separate plot of the third and fourth coordinates, and a final plot of the fifth and sixth coordinates. Discuss the desirability of using each of these four choices for "random" points.

How effective are quasi-random points in approximating integrals? For quasi-random points, the absolute value of the error $I - I_n$ is bounded by $V[f]\,(\log n)^d n^{-1}$, where $V[f]$ is a measure of the variation of f, evaluated by integrating the absolute value of the dth partial derivative of f with respect to each of its variables, and adding on a boundary term. If d isn't too big and f isn't too wild, the result of Monte Carlo integration using quasi-random points will probably have smaller error than using pseudo-random points.

PROBLEM 4.

Compute new estimates of the integral of Problem 1 using quasi-random numbers in Method 2 instead of pseudo-random numbers. Compare the results.

Back to Our Partition Function

Let's get back to computing a partition function.

PROBLEM 5.

a. Let L = 1 and α = 1.38 Angstroms^2 g/sec^2 K and consider the harmonic oscillator potential $V(x)$. Determine finite integration limits for a, x_1, …, x_d so that the partition function $Z_d(L)$ in Equation 1 can be approximated to three-digit accuracy. (Do this by bounding the neglected part of the integral.)

b. Use your favorite one-dimensional integration routine to estimate the partition function $Z_d(L)$. When you need a function value $\rho(a, a, L)$, use Monte Carlo integration to obtain it. Try n = 100, 1,000, 10,000, 100,000, 1,000,000 (if possible), and d = 1, 2, 4, 8, 16.

Tools

The integration techniques we use here, as well as algorithms for generating quasi-random numbers, are discussed in a good review article.[1] Stratified sampling, also discussed there, is another tool for improving Monte Carlo estimates.

Isabel Beichl and Francis Sullivan give a good introduction to importance sampling.[2]

In an article on the partition function,[3] the authors also present the divide-and-conquer approach to computing it.

Monte Carlo integration is also used to compute integrals in rarefied gas dynamics,[1] the failure rate of materials,[4] and quantum chromodynamics (QCD).[5]

For information on faster algorithms for approximating integrals in one-dimensional space, such as that used in Matlab's `quad`, see a standard numerical methods textbook.[6]

References

1. R.E. Caflisch, "Monte Carlo and Quasi-Monte Carlo Methods," *Acta Numerica*, vol. 7, 1998, pp. 1–49.
2. I. Beichl and F. Sullivan, "The Importance of Importance Sampling," *Computing in Science & Eng.*, vol. 1, no. 2, 1999, pp. 71–73.
3. M. Nauenberg, F. Kuttner, and M. Furman, "Method for Evaluating One-Dimensional Path Integrals," *Physical Rev. A*, vol. 13, no. 3, 1976, pp. 1185–1189.
4. G.I. Schuëller, H.J. Pradlwarter, and P.S. Koutsourelakis, "A Comparative Study of Reliability Estimation Procedures for High Dimensions," *Proc. 16th ASCE Eng. Mechanics Conf.*, ASCE, 2003; www.ce.washington.edu/em03/proceedings/papers/777.pdf.
5. G.C. Fox, R.D. Williams, and P.C. Messina, *Parallel Computing Works*, Morgan Kaufmann, 1994, www.netlib.org/utk/lsi/pcwLSI/text/node34.html.
6. C.F. van Loan, *Introduction to Scientific Computing*, Prentice Hall, 2000.

c. Repeat the experiment using quasi-random points in the Monte Carlo integration. What can you say about the accuracy and convergence rate of your estimates?

Partition and Conquer

We noted earlier that if the function to be integrated is separable, we can reduce our problem to a product of one-dimensional integrations. Although our partition function can't be separated into $d + 1$ factors, it can be partially separated by noticing that each variable x_i appears only in a pair of functions g, so it makes sense to accumulate

$$\hat{g}(x_{i-1}, x_{i+1}) = \int_{-\infty}^{\infty} g(x_{i-1}, x_i, L)\, g(x_i, x_{i+1}, L)\, dx$$

for even values of i. If we repeat this trick on $\hat{g}$ for values of i that are multiples of 4, we reduce our problem further, and continuing, we can actually reduce the problem to a two-dimensional integration when d is a power of 2. Although the partition function provides a good test problem for multidimensional integration, we should really compute it using this divide-and-conquer formulation.

Partial Solution to Last Issue's Homework Assignment

Achieving a Common Viewpoint: Yaw, Pitch, and Roll

By Dianne P. O'Leary and David A. Schug

As you might recall from the last issue, there are many ways to define 3D rotations; we specified yaw ϕ, pitch θ, and roll ψ, as is common in flight control. In this coordinate system, the angles ϕ, θ, and ψ are called the *Euler angles*.

Define a rotation $\boldsymbol{Q}$ as the product of three matrices $\boldsymbol{Q}(\phi, \theta, \psi) = \boldsymbol{Q}_{roll}\,\boldsymbol{Q}_{pitch}\,\boldsymbol{Q}_{yaw}$, where

$$Q_{roll} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & \sin\psi \\ 0 & -\sin\psi & \cos\psi \end{bmatrix},$$

$$Q_{pitch} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix},$$

$$Q_{yaw} = \begin{bmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

with the Euler angles satisfying $-\pi < \phi \le \pi$, $-\pi/2 < \theta \le \pi/2$, and $-\pi < \psi \le \pi$.

Problem 1.

a. Explain geometrically the effect of applying a rotation $\boldsymbol{Q}$ to a vector $[x, y, z]^T$ to create the vector $\boldsymbol{Q}\,[x, y, z]^T$.
b. Show that if $\boldsymbol{Q}$ is any 3×3 orthogonal matrix (that is, $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$, the identity matrix), then $\boldsymbol{Q}$ can be expressed as $\boldsymbol{Q}_{roll}\,\boldsymbol{Q}_{pitch}\,\boldsymbol{Q}_{yaw}$ for some choice of angles ψ, θ, and ϕ.

Answer: a. First we rotate the object by an angle ϕ in the xy-plane. Then we rotate by an angle $-\theta$ in the new xz-plane, and finish with a rotation of ψ in the resulting yz-plane.

b. (Actually, this is only true if $\det(Q) > 0$; otherwise, we need to multiply by –1.) We will use the QR decomposition of a matrix; any nonsingular matrix can be expressed as the product of an orthogonal matrix times an upper triangular one. One way to compute this is to use plane rotations to reduce elements below the diagonal of our matrix to zero. Let's apply this to the matrix $\boldsymbol{Q}^T$. By choosing ϕ appropriately, we can make $\boldsymbol{Q}_{yaw}\,\boldsymbol{Q}^T$ have a zero in row 2, column 1. Similarly, by choosing θ, we can force a zero in row 3, column 1 of $\boldsymbol{Q}_{pitch}\,\boldsymbol{Q}_{yaw}\,\boldsymbol{Q}^T$ (without ruining our zero in row 2, column 1). Finally, we can choose ψ to force $\boldsymbol{Q}_{roll}\,\boldsymbol{Q}_{pitch}\,\boldsymbol{Q}_{yaw}\,\boldsymbol{Q}^T$ to be upper triangular. Now the product of orthogonal matrices is orthogonal, and the only upper triangular orthogonal matrices are diagonal. We can force the product to be the identity by choosing the signs as follows: $\cos\phi < 0$ when $q_{11} < 0$; $\cos\psi < 0$ when $q_{33} < 0$; $\sin\theta < 0$ when $q_{31} > 0$. We conclude that $\boldsymbol{Q}_{roll}\,\boldsymbol{Q}_{\text{pitch}}\,\boldsymbol{Q}_{yaw} = (\boldsymbol{Q}^T)^{-1} = \boldsymbol{Q}$. This method for proving this property is particularly nice because it leads to a fast algorithm that we can use in Problem 4 to recover the Euler angles given an orthogonal matrix $\boldsymbol{Q}$.

Let $\boldsymbol{A}$ be the $3 \times n$ matrix ($n = 7$) whose columns are the coordinates of the first set of points: $\boldsymbol{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_7]$. Define $\boldsymbol{B}$ similarly from the second set of points. Then we want to determine the three Euler angles so that

$$\boldsymbol{B} = \boldsymbol{Q}(\phi, \theta, \psi)\boldsymbol{A}.$$

by minimizing

$$f(\phi, \theta, \psi) = \|\boldsymbol{B} - \boldsymbol{Q}(\phi, \theta, \psi)\boldsymbol{A}\|_F^2 \equiv \sum_{i=1}^{n} \|\mathbf{b}_i - \boldsymbol{Q}(\phi, \theta, \psi)\mathbf{a}_i\|_2^2$$

Problem 2.

Use a nonlinear least-squares solver to find the Euler angles for the data sets generated by taking the yaw $\phi = \pi/4$, roll $\psi = \pi/9$, and

$$\boldsymbol{A} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & -1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 2 \\ 0 & 1 & 2 & 3 & 4 & 4 & 4 \end{bmatrix}.$$

Let the pitch θ vary between $-\pi/2$ and $\pi/2$ in steps of $\pi/120$. Plot the computed Euler angles, and, in a separate plot, the Frobenius norm of the error in $\boldsymbol{Q}$ and the root-mean-square distance (RMSD) in the computed positions.

Copublished by the IEEE CS and the AIP 1521-9615/04/$20.00 © 2004 IEEE

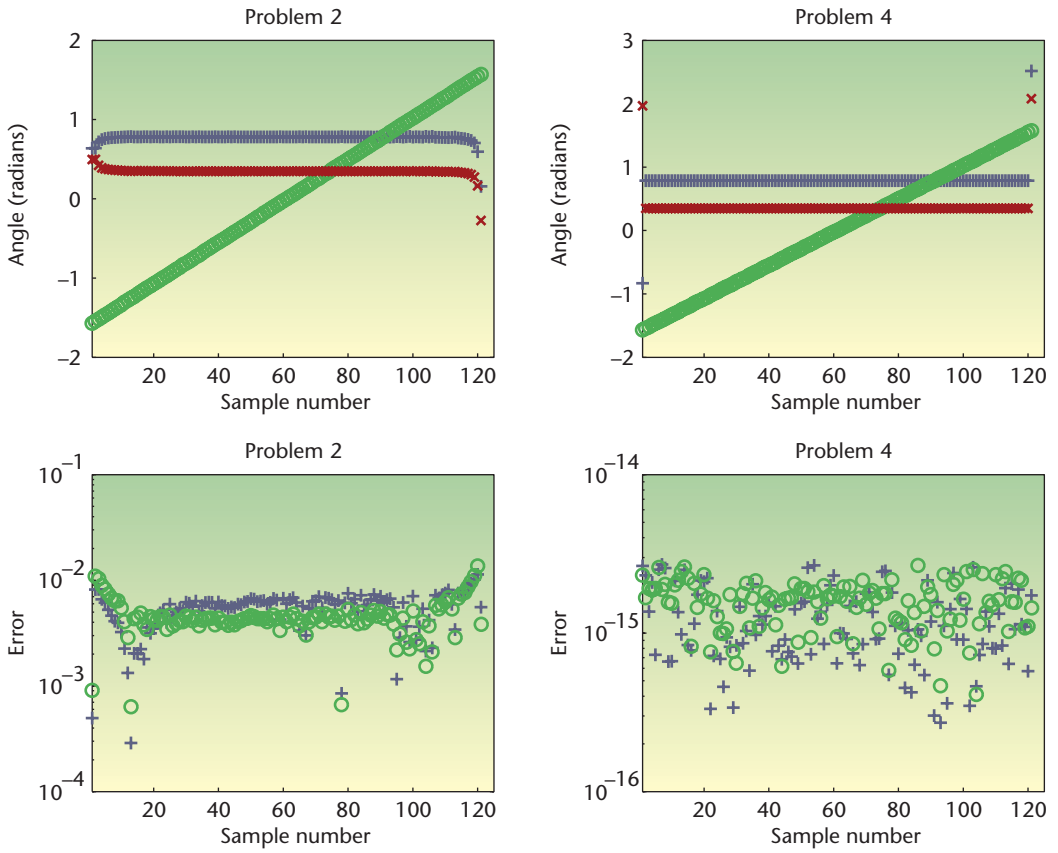

Figure A. Results of Problems 2 (left) and 4 (right). The top graphs show the computed yaw (blue plusses), pitch (green circles), and roll (red x's), and the bottom graphs show the error in Q (blue plusses) and the error in the rotated positions (green circles).

Discuss the time needed for solution and the accuracy obtained.

Answer: A sample Matlab program to solve this problem is available at www.computer.org/cise/homework/; Figure A shows the results. In most cases, two-to-four-digit accuracy is achieved for the angles and positions, but trouble is observed when the pitch is close to vertical ($\pm \pi/2$).

PROBLEM 3.

a. Recall that the trace *tr* of a square matrix is the sum of its main diagonal entries. We need two facts about traces in order to derive our algorithm. Prove that for any matrix $\boldsymbol{C}$, $tr(\boldsymbol{C}^T\boldsymbol{C}) = \|\boldsymbol{C}\|_F^2$, and that for any matrix $\boldsymbol{D}$ for which the product $\boldsymbol{CD}$ is defined, $tr(\boldsymbol{CD}) = tr(\boldsymbol{DC})$.

b. Use the first fact to show that the $\boldsymbol{Q}$ that minimizes $\|\boldsymbol{B} - \boldsymbol{QA}\|_F^2$ over all choices of orthogonal $\boldsymbol{Q}$ also maximizes $tr(\boldsymbol{A}^T\boldsymbol{Q}^T\boldsymbol{B})$.

c. Suppose that the singular value decomposition (SVD) of the $m \times m$ matrix $\boldsymbol{BA}^T$ is $\boldsymbol{U\Sigma V}^T$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are $m \times m$ and orthogonal, and Σ is diagonal with diagonal entries $\sigma_1 \geq \ldots \geq \sigma_m \geq 0$. Define $\boldsymbol{Z} = \boldsymbol{V}^T\boldsymbol{Q}^T\boldsymbol{U}$. Use these definitions and the second fact to show that

$$tr(\boldsymbol{A}^T\boldsymbol{Q}^T\boldsymbol{B}) = tr(\boldsymbol{Q}^T\boldsymbol{BA}^T) = tr(\boldsymbol{Z\Sigma}) \leq \sum_{i=1}^{m} \sigma_i .$$

d. If $\boldsymbol{Z} = \boldsymbol{I}$, then

$$tr(\boldsymbol{Q}^T\boldsymbol{BA}^T) = \sum_{i=1}^{m} \sigma_i .$$

What choice of $\boldsymbol{Q}$ ensures this?

Answer: a. Suppose that $\boldsymbol{C}$ is $m \times n$. The first fact follows from

$$tr(\boldsymbol{C}^T\boldsymbol{C}) = \sum_{k=1}^{n}\left(\sum_{i=1}^{m} c_{ik}^2\right) = \sum_{k=1}^{n}\sum_{i=1}^{m} c_{ik}^2 = \|\boldsymbol{C}\|_F^2 .$$

To prove the second fact, note that

$$tr(\boldsymbol{CD}) = \sum_{k=1}^{m}\left(\sum_{i=1}^{n}(c_{ki}d_{ik})\right),$$

while

$$tr(\boldsymbol{DC}) = \sum_{i=1}^{n}\left(\sum_{k=1}^{m}(d_{ik}c_{ki})\right),$$

which is the same.

b. Note that

$$\|\boldsymbol{B}-\boldsymbol{QA}\|_F^2 = tr((\boldsymbol{B}-\boldsymbol{QA})^T(\boldsymbol{B}-\boldsymbol{QA})) = tr(\boldsymbol{B}^T\boldsymbol{B}+\boldsymbol{A}^T\boldsymbol{A}) - 2\,tr(\boldsymbol{A}^T\boldsymbol{Q}^T\boldsymbol{B}),$$

so we can minimize the left-hand side by maximizing $tr(\boldsymbol{A}^T\boldsymbol{Q}^T\boldsymbol{B})$.

c. We compute

$$tr(\boldsymbol{Q}^T\boldsymbol{B}\boldsymbol{A}^T) = tr(\boldsymbol{Q}^T\boldsymbol{U}\Sigma\boldsymbol{V}^T) = tr(\boldsymbol{V}^T\boldsymbol{Q}^T\boldsymbol{U}\Sigma)$$
$$= tr(\boldsymbol{Z}\Sigma) = \sum_{i=1}^{m}\sigma_i z_{ii} \le \sum_{i=1}^{m}\sigma_i, \quad (1)$$

where the inequality follows from the fact that elements of an orthogonal matrix lie between –1 and 1.

d. Since $\boldsymbol{Z} = \boldsymbol{V}^T\boldsymbol{Q}^T\boldsymbol{U}$, we have $\boldsymbol{Z} = \boldsymbol{I}$ if $\boldsymbol{Q} = \boldsymbol{U}\boldsymbol{V}^T$.

Problem 4.

Use the SVD to find the Euler angles for the data in Problem 2. Compare with your previous results.

Answer: Figure A shows the results. The computed results are much better than those of Problem 2, with errors at most 10^{-14} and no trouble when the pitch is close to vertical.

Problem 5.

Given a fixed rotation matrix $\boldsymbol{Q}$, show that the minimizer of $\|\boldsymbol{B}-\boldsymbol{QA}-\mathbf{t}\mathbf{e}^T\|_F$, where $\mathbf{e}$ is a column vector of ones, satisfies $\mathbf{t} = \mathbf{c}_A - \boldsymbol{Q}\mathbf{c}_B$.

Answer: We compute

$$\|\boldsymbol{B}-\boldsymbol{QA}-\mathbf{t}\mathbf{e}^T\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n}(\boldsymbol{B}-\boldsymbol{QA})_{ij}^2 - 2t_i(\boldsymbol{B}-QA)_{ij} + nt_i^2,$$

and setting the partial derivative with respect to t_i to zero yields

$$t_i = \frac{1}{n}\sum_{j=1}^{n}(\boldsymbol{B}-\boldsymbol{QA})_{ij}.$$

Therefore,

$$t = \frac{1}{n}\sum_{j=1}^{n}\mathbf{b}_j - \frac{1}{n}\boldsymbol{Q}\mathbf{a}_j$$
$$= \mathbf{c}_B - \boldsymbol{Q}\mathbf{c}_A.$$

This nice observation was made by Richard Hanson and Michael Norris.[1]

Problem 6.

Implement this algorithm and try it on the data from Problem 2 using $\theta = \pi/4$ and 20 randomly generated translations $\mathbf{t}$. Repeat the experiment with 20 more translations, adding perturbations to the $\boldsymbol{A}$ data that are uniformly distributed between -10^{-3} and 10^{-3}, to see how sensitive the computation is to uncertainty in the measurements.

Answer: Figure B shows the results. With no perturbation, the errors in the angles, the error in the matrix $\boldsymbol{Q}$, and the RMSD are all less than 10^{-15}. With perturbation in each element uniformly distributed between -10^{-3} and 10^{-3}, the errors rise to about 10^{-4}.

Others have compared the SVD method with other methods,[2,3] although none of these authors knew that the method was due to Hanson and Norris.[1]

Problem 7.

a. Suppose that all of our points in $\boldsymbol{A}$ lie on a line. Is there more than one choice of $\boldsymbol{Q}$ that minimizes $\|\boldsymbol{B}-\boldsymbol{QA}\|$? Illustrate this with a numerical example.

b. Use this insight to characterize the degenerate cases for which $\boldsymbol{Q}$ is not well determined.

c. Suppose that our data produces the angles $(\phi, \theta = \pi/2, \psi)$, but a small perturbation causes a small increase in the angle θ so that it is greater than $\pi/2$. Generate such an example: you'll see that the computed angles are quite different. This jump in angle is called *gimbal lock*, a term borrowed from the locking of the mechanism that moves a stabilizing gyroscope in cases in which the angle goes out of the device's range of motion.

Answer: a. Yes. Since the rank of matrix $\boldsymbol{A}$ is 1 in this case, we have two singular values $\sigma_2 = \sigma_3 = 0$. Therefore, we only

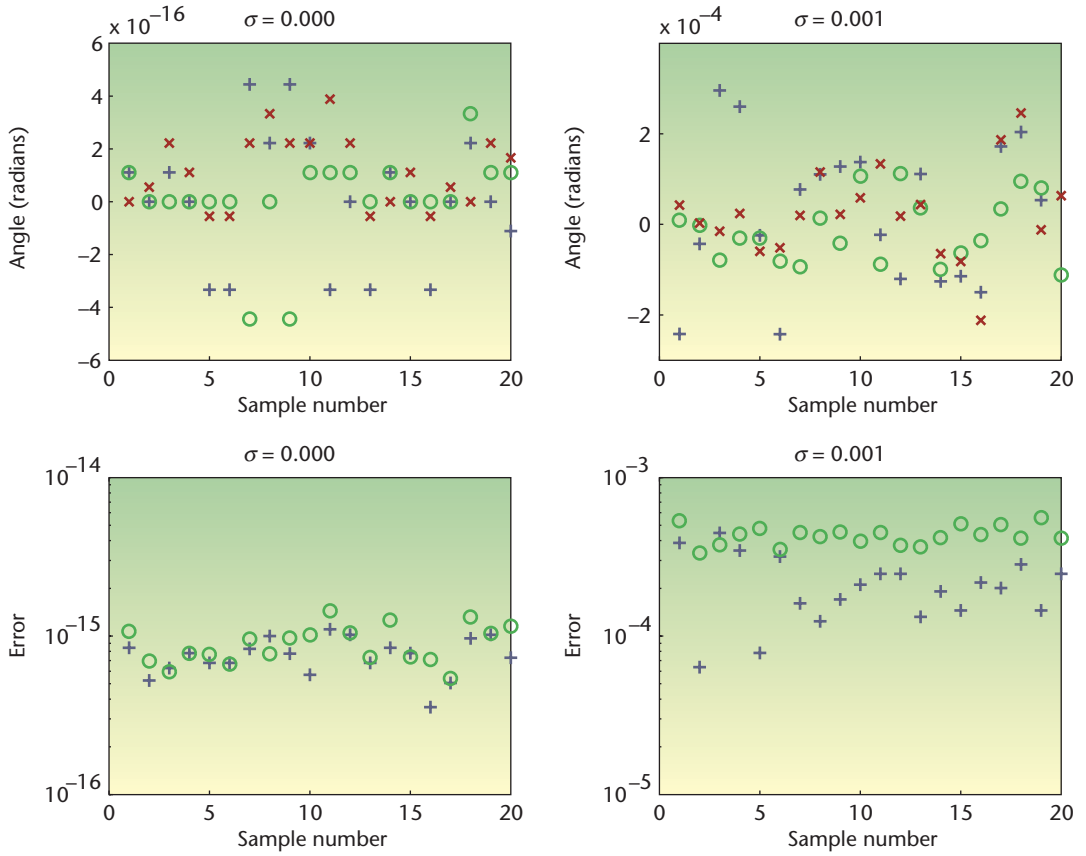

Figure B. Results of Problem 6. The left side shows the result with no perturbation and the right side with perturbation of order 10^{-3}. The top graphs show the computed yaw (blue plusses), pitch (green circles), and roll (red x's), and the bottom graphs show the error in Q (blue plusses) and the error in the rotated positions (green circles).

need $z_{11} = 1$ in Equation 1 and we don't care about the values of z_{22} or z_{33}.

b. Degenerate cases result from unfortunate choices of the points in $\boldsymbol{A}$ and $\boldsymbol{B}$. If all the points in $\boldsymbol{A}$ lie on a line or a plane, then multiple solution matrices $\boldsymbol{Q}$ exist. Additionally, if two singular values of the matrix $\boldsymbol{B}^T\boldsymbol{A}$ are nonzero but equal, then small perturbations in the data can create large changes in the matrix $\boldsymbol{Q}$.[1]

c. A degenerate case and a case of gymbal lock are illustrated on the Web page. CISE

References

1. R.J. Hanson and M.J. Norris, "Analysis of Measurements Based on the Singular Value Decomposition," *SIAM J. Scientific and Statistical Computing*, vol. 2, no. 3, 1981, pp. 363–373.
2. K. Kanatani, "Analysis of 3D Rotation Fitting," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 5, 1994, pp. 543–549.
3. D.W. Eggert, A. Lorusso, and R.B. Fisher, "Estimating 3D Rigid Body Transformations: A Comparison of Four Major Algorithms," *Machine Learning and Applications*, vol. 9, 1997, pp. 272–290.

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 has 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/.

David A. Schug is an advanced special student at the University of Maryland, where he received a BS in mathematics. Contact him at david.schug@navy.mil.