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

SENSITIVITY ANALYSIS

When a Little Means a Lot

By Dianne P. O'Leary

IN CONTRAST TO CLASSROOM EXERCISES, THE REAL WORLD RARELY PRESENTS US WITH A PROBLEM IN WHICH THE DATA IS KNOWN WITH ABSOLUTE CERTAINTY. SOME parameters (such as π) we can define with certainty, and others (such as $\hbar$) we know to high precision, but most data is measured and therefore contains measurement error.

So what we really solve isn't the problem we want, but some *nearby* problem, and in addition to reporting the computed solution, we really need to report a bound on either

- the difference between the true solution and the computed solution (a *forward error bound*), or
- the difference between the problem we solved and the problem we wanted to solve (a *backward error bound*).

This need occurs throughout computational science. For example,

- If we compute the resonant frequencies of a model of a building, we want to know how these frequencies change if the load within the building changes.
- If we compute the stresses on a bridge, we want to know how sensitive these values are to changes that might occur as the bridge ages.
- If we develop a model for our data and fit the parameters using least squares, we want to know how much the parameters would change if the data were wiggled within the uncertainty limits.

In this homework assignment, we use some simple problems to investigate the use of several tools for *sensitivity analysis*.

Sensitivity Is Measured by Derivatives

The best way to measure the sensitivity of a variable x to a change in a parameter t is to compute dx/dt because, by Taylor series, if δ is a small number, then

$$x(t+\delta) \approx x(t) + \delta \frac{dx}{dt}(t).$$

Thus, the change in x due to a change in t is proportional to dx/dt (whenever the second derivative d^2x/dt^2 is bounded). Sometimes we can't compute the derivative, and sometimes it doesn't exist, but this is the method of choice whenever possible.

As an example, let's use the derivative to gain insight into the sensitivity of a quadratic equation's roots to changes in the coefficients.

PROBLEM 1.

Suppose x_1 and x_2 are the roots of the quadratic equation $x^2 + bx + c = 0$.

a. Use implicit differentiation to compute dx/db.

b. We know that the roots are

$$x_{1,2} = \frac{1}{2}\left(-b \pm \sqrt{b^2 - 4c}\right).$$

Differentiate this expression with respect to b, and show that the answer is equivalent to the one you obtained in (a).

c. Find values of b and c for which the roots are very sensitive to small changes in b and values for which they aren't sensitive.

If we need to vary several parameters, the partial derivatives of the variable with respect to the parameters yield the sensitivity information.

Derivatives also give sensitivity information for constrained problems. If we want to minimize a function $f(\mathbf{x})$ subject to the constraints $\mathbf{h}(\mathbf{x}) = \mathbf{0}$, for example, we learned in calculus to introduce *Lagrange multipliers* λ, one per constraint, and look for solutions $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ for which the Lagrangian function $L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{h}(\mathbf{x})$ has a zero gradient. The "artificial" variables $\boldsymbol{\lambda}$ actually have a physical meaning: $\boldsymbol{\lambda}_i$ is the partial derivative of L with respect to the constraint h_i. Therefore, the value of the multipliers at

a point where $\mathbf{h}(\mathbf{x}) = \mathbf{0}$ measures the sensitivity of f to small changes in the constraint. For this reason, Lagrange multipliers are sometimes called *marginal values* or *reduced costs*. We'll see an example of their use in Problem 3.

Condition Numbers Give Bounds on Sensitivity

Although the derivatives of the variable with respect to the parameters provide the "gold standard" for sensitivity analysis, differentiation isn't always practical. For example, a linear system of equations with 100 variables has 10^4 coefficients in the matrix—that's a lot of partial derivatives to compute and assess! Because of this, shortcuts have been developed that give less-specific information but summarize what can happen in the worst case for perturbations of a given size. These shortcuts involve computing a *condition number* for the problem.

Given a condition number, we can make statements such as the following: if the matrix is changed to $A + E$, where

$$\frac{\|E\|}{\|A\|} \leq \delta,$$

and δ is a small number, then the difference between the solution to the linear system $A\mathbf{x} = \mathbf{b}$ and the solution $\mathbf{x}_e$ to the linear system $(A + E)\mathbf{x_e} = \mathbf{b}$ is bounded by

$$\frac{\|\mathbf{x} - \mathbf{x_e}\|}{\|\mathbf{x}\|} \leq \frac{\kappa(A)}{1 - \kappa(A)\delta}\delta, \qquad (1)$$

where $\kappa(A)$ is the condition number of A and $\delta < 1/\kappa(A)$.

By its very nature, the statement involving condition numbers is a *worst-case* statement because it must hold for all perturbations that are small enough. For some values of E, the error bound in Equation 1 can be a serious overestimate, but some particular matrix always exists for which the bound is tight.

Condition numbers let us replace the full set of partial derivatives by a single number, but even that one number can be hard to compute—for example, $\kappa(A) = \|A\| \, \|A^{-1}\|$. The norm of A is usually easy to compute—depending on our choice of norm:

- $\|A\|_1$ is the maximum of the 1-norm of the columns of A.
- $\|A\|_\infty$ is the maximum of the 1-norm of the rows of A.
- $\|A\|_2$ is the square root of the largest eigenvalue of A^TA.

A vector's 1-norm is just the sum of the absolute values of its components, so it's easy to compute $\|A\|_1$ and $\|A\|_\infty$, but the 2-norm is more problematic.

Even so, computing any of these norms for A^{-1} is quite expensive because computing the inverse of a matrix is expensive (and generally not a good idea). Therefore, we usually use an *estimated* condition number. In Matlab, we can use `cond(A,normtype)` to compute the condition number (setting `normtype` to 1, 2, or `inf`), or use the cheaper function `condest` to estimate the condition number.

Problem 2.

Consider the linear system $A\mathbf{x} = \mathbf{b}$ with

$$A = \begin{bmatrix} \delta & 1 \\ 1 & 1 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

where $\delta = 0.002$.

a. Plot the two equations defined by this system and compute the condition number of A (`cond(A)`).

b. Compute the solution $\mathbf{x}^*$ to $A\mathbf{x} = \mathbf{b}$ and also compute the solution to the nearby systems $(A + E^{(i)})\mathbf{x}^{(i)} = \mathbf{b}$ for $i = 1, \ldots, 1{,}000$, where the elements of $E^{(i)}$ are normally distributed with mean 0 and standard deviation $\tau = .0001$. (You can do this by setting each `E = tau*randn(2,2)`.) Plot $\mathbf{x}^{(i)} - \mathbf{x}^*$. This plot reveals the forward error in using $(A + E^{(i)})$ as an approximation to A. In a separate figure, plot the residuals $\mathbf{b} - A\mathbf{x}^{(i)}$ (the backward error) for each computed solution.

c. Repeat (a) and (b) with the linear system $A\mathbf{x} = \mathbf{b}$ with

$$A = \begin{bmatrix} 1+\delta & \delta-1 \\ \delta-1 & 1+\delta \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 2 \\ -2 \end{bmatrix}.$$

d. Discuss your results. Why do the forward error plots for the two problems look so different? How does the condition number relate to what you see in the forward error plots? What do the backward error plots tell you?

Monte Carlo Experiments Can Estimate Sensitivity

In Problem 2b, you did a *Monte Carlo experiment*. The idea is to take random samples of nearby problems and see how the solution changes. This is a fine way to measure sensitivity if a condition number won't give enough information or if we can't obtain derivatives, but the process can be expensive.

Let's get experience with two more applications of Monte Carlo to estimate sensitivity, one involving linear programming and one involving a differential equation.

TOOLS

Standard advanced calculus textbooks and textbooks on optimization, such as that by Stephen Nash and Ariela Sofer, discuss Lagrange multipliers.[1]

Condition numbers are commonly used to measure the sensitivity of linear systems of equations, eigenvalues, eigenvectors, and other quantities. Nicholas Higham's book is a good reference.[2]

George Fishman's book offers more information on Monte Carlo estimation.[3]

Standard statistical textbooks explain the use of confidence intervals, which can also be applied to constrained problems.[4]

One alternative to the methods we consider in the main text is *interval analysis*. In this method, we carry upper and lower bounds on each quantity along through our calculations. The result is a rigorous, although often pessimistic, set of forward error bounds on the answer. The method's most forceful advocate was Ramon E. Moore,[5] and many textbooks apply the method to scientific computing.

A second alternative is the use of *symbolic computation*, in which we carry analytic expressions for each quantity. Systems such as Maple (www.maplesoft.com; included in Matlab) or Mathematica (www.wolfram.com) are incredibly useful, but eventually they produce a formula that must be evaluated using arithmetic. These systems have pitfalls of their own: the computation can use a tremendous amount of time and storage, and the systems can produce formulas that, when evaluated, lead to unnecessarily high relative and absolute errors.

References

1. S.G. Nash and A. Sofer, *Linear and Nonlinear Programming*, McGraw-Hill, 1996.
2. N.J. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM Press, 1996.
3. G.S. Fishman, *Monte Carlo: Concepts, Algorithms and Applications*, Springer, 2003.
4. B.W. Rust and D.P. O'Leary, "Confidence Intervals for Discrete Approximations to Ill-Posed Problems," *J. Computational and Graphical Statistics*, vol. 3, no. 1, 1994, pp. 67–96.
5. R.E. Moore, *Interval Analysis*, Prentice-Hall, 1966.

PROBLEM 3.

Investigate the sensitivity of the *linear programming problem*

$$\min_{\mathbf{x}} \mathbf{c}^T \boldsymbol{x}$$

subject to

$$\boldsymbol{A}\mathbf{x} \le b,$$

$$x_1 \ge 0,$$

$$x_2 \ge 0.$$

a. Let $\boldsymbol{A} = [1, 1]$, $b = 1$, and $\mathbf{c}^T = [-3, -1]$. Solve the linear program (using, for instance, Matlab's `linprog`) and use the Lagrange multipliers (also called *dual variables*) to evaluate the sensitivity of $\mathbf{c}^T\mathbf{x}$ to small changes in the constraints. Illustrate this sensitivity using a Monte Carlo experiment, solving 100 problems with $\boldsymbol{A}$ replaced by $\boldsymbol{A} + \boldsymbol{E}^{(i)}$, where the elements of $\boldsymbol{E}^{(i)}$ are uniformly distributed on the interval $[-\tau, \tau]$, with $\tau = 0.001$. (You can do this by setting each `E = 2*tau*(rand(1,2)-.5)`.) Plot all the solutions in one figure and all the function values $\mathbf{c}^T\mathbf{x}$ in another.

Repeat for two more examples:

b. $\boldsymbol{A} = [1, 1]$, $b = 1$, $\mathbf{c}^T = [-1.0005, -0.9995]$.

c. $\boldsymbol{A} = [0.01, 5]$, $b = 0.01$, $\mathbf{c}^T = [-1, 0]$.

Explain how the Lagrange multiplier for the constraint $\boldsymbol{A}\mathbf{x} \le b$ gives insight into the sensitivity observed in the corresponding Monte Carlo experiment.

PROBLEM 4.

Consider the very simple differential equation for $y(t)$:

$$y' = ay,$$

where $y(0) = 1$ and $a(t)$ is given. Let's investigate the equation's sensitivity to our knowledge of a.

To make the problem concrete, we can divide the US population growth rate $a(t)$ into two pieces: a rate of 0.006 due to births and deaths and a rate of 0.003 due to migration. Determine how much the population will increase over the next 50 years if this rate stays constant, and how much it will increase if we set migration to zero. Then perform Monte Carlo experiments, assuming that the birth/death rate is normally distributed with mean 0.006 and standard deviation 0.001, and the migration rate is uniformly distributed between 0 and 0.003. Also experiment with what happens if years of high growth rate come early, followed by years of low growth rate, and vice versa. Plot and discuss the results.

Confidence Intervals Can Give Insight

Another way to assess sensitivity is to make a statement such as the following: if we repeat the data measurement many times, we expect that, 95 percent of the time, the solution's computed value will lie in the interval $[x^{lo}, x^{up}]$. Such an interval is called a *confidence interval*, and $\alpha = .95$ is the *confidence level*, determined from statistical estimation, assuming that the errors in the measurements are random.

As an example, consider a linear system $\boldsymbol{A}\mathbf{x} = \mathbf{b}$ and assume that the error $\mathbf{b} - \mathbf{b}_{\text{true}}$ is normally distributed with mean $\mathbf{0}$ and variance S^2. Suppose we want to estimate the value of $\mathbf{w}^T\mathbf{x}$; taking $\mathbf{w}^T$ to be the first unit vector, for example, gives us an estimate of x_1. We proceed as follows:

- Determine a value κ from the cumulative normal distribution, so that
$$\frac{1}{\sqrt{2\pi}}\int_{-\kappa}^{+\kappa} e^{-z^2/2}dz = \alpha.$$
For 95 percent confidence intervals, $\kappa \approx 1.960$.
- Given the value κ, compute
$$\phi^{lo} = \mathbf{w}^T\hat{\mathbf{x}} - \kappa\sqrt{\mathbf{w}^T(\boldsymbol{A}^T\boldsymbol{S}^{-2}\boldsymbol{A})^{-1}\mathbf{w}}$$
$$\phi^{up} = \mathbf{w}^T\hat{\mathbf{x}} + \kappa\sqrt{\mathbf{w}^T(\boldsymbol{A}^T\boldsymbol{S}^{-2}\boldsymbol{A})^{-1}\mathbf{w}},$$
where $\hat{\mathbf{x}}$ solves $\boldsymbol{A}\mathbf{x} = \mathbf{b}$.

Then $[\phi^{lo}, \phi^{up}]$ is a 100 α percent confidence interval for $\mathbf{w}^T\mathbf{x}$.

The procedure has more general forms. It's possible to construct (wider) confidence intervals that have *joint* probability α so that, for example, we can bound all the components of $\mathbf{x}$ simultaneously. There's also a nonparametric form of the result that lets us compute confidence intervals when the error isn't normally distributed. See the "Tools" sidebar for some references.

Let's apply our procedure to the examples in Problem 2.

Problem 5.

Using the two linear systems from Problem 2, perform a Monte Carlo experiment that computes the solution to the nearby systems

$$\boldsymbol{A}\mathbf{x}^{(i)} = \mathbf{b} + \mathbf{e}$$

for $i = 1, \ldots, 1{,}000$, where the elements of $\mathbf{e}$ are normally distributed with mean 0 and standard deviation $\tau = 0.0001$. Compute 95 percent confidence intervals on each component of the solution to the two linear systems, and see how many of the components of the Monte Carlo samples lie within the confidence limits.

We've experimented here with several ways to measure sensitivity in our problems; the "Tools" sidebar gives additional alternatives. A good computational scientist computes not just an answer to a problem but an assessment of how good it is, and sensitivity analysis is an important piece of this assessment.

Partial Solution to Last Issue's Homework Assignment

Multigrid Methods: Managing Massive Meshes

By Dianne P. O'Leary

In the last issue's installment of your homework assignment, we investigated multigrid methods for solving linear systems. (Please see p. 10 of this issue of *CiSE* for more information about multigrid methods, or visit our Web site at www.computer.org/cise/homework/.)

Problem 1.

Work through the V-Cycle algorithm to see exactly what computations it performs on our simple example for a sequence of grids defined by $h = 1/2$, $1/4$, $1/8$, and $1/16$. Estimate the amount of work, measured by the number of floating-point multiplications and divisions that it does.

Answer:
The V-Cycle performs the operations shown in Figure 1.

η_1 G-S iterations for $h = 1/16$:	15 η_1 multiplications (by $h^2/2$)
$h = 1/16$ residual evaluation	16 multiplications
Multiplication by $R_{1/8}$	14 multiplications
η_1 G-S iterations for $h = 1/8$:	7 η_1 multiplications
$h = 1/8$ residual evaluation	8 multiplications
Multiplication by $R_{1/4}$	6 multiplications
η_1 G-S iterations for $h = 1/4$:	3 η_1 multiplications
$h = 1/4$ residual evaluation	4 multiplications
Multiplication by $R_{1/2}$	2 multiplications
Direct solution of the system for $h = 1/2$:	1 multiplication
Multiplication by $P_{1/2}$	2 multiplications
η_2 G-S iterations for $h = 1/4$:	3 η_2 multiplications
Multiplication by $P_{1/4}$	6 multiplications
η_2 G-S iterations for $h = 1/8$:	7 η_2 multiplications
Multiplication by $P_{1/8}$	14 multiplications
η_2 G-S iterations for $h = 1/16$:	15 η_2 multiplications

Figure 1. Answer to Problem 1. The V-Cycle algorithm performs these operations for a sequence of grids defined by h = 1/2, 1/4, 1/8, and 1/16.

The total cost is less than the cost of $2(\eta_1 + \eta_2)$ iterations, plus two residual calculations, all done on the $h = 1/16$ grid, plus four multiplications by $R_{1/8}$.

PROBLEM 2.

Convince yourself that the storage necessary for all the matrices and vectors is also a modest multiple of the storage necessary for the finest grid.

Answer:
The right-hand sides have

$$15 + 7 + 3 + 1 = 16(1 + 1/2 + 1/4 + 1/8) - 4$$

elements, which is less than twice the storage necessary for the right-hand side for the finest grid. The same is true for the solution vectors. Similarly, each matrix h has at most half the number of nonzeros as the one for the next finer grid, so the total matrix storage is less than two times that for $A_{1/16}$.

We can store the matrices P_h as sparse matrices or, because we need only to form their products with vectors, we can just write a function to perform multiplication without explicitly storing them.

PROBLEM 3.

Write a program that applies the multigrid V-Cycle iteration to the two-dimensional problems used in the homework assignment given in the last issue of *CiSE*. The Matlab program `generateproblem.m` produces a structure called `mesh`, which contains, in addition to the matrices and right-hand side information, the operators `P` and the coordinates `p` of the mesh points. The differential equation is

$$-u_{xx}(x, y) - u_{yy}(x, y) + \kappa u(x, y) = f(x, y)$$

for $(x, y) \in [-1, 1] \times [-1, 1]$ (`myproblem=1`) or for this domain with a hole cut out (`myproblem=2`). The boundary conditions are that u is zero on the square's boundary and, for the second case, that the normal derivative is zero at the hole's boundary.

Set $\kappa = 0$ and compare the time for solving the problem using multigrid to the methods defined in the last issue.

Answer:
In the solution to the July/August 2006 homework assignment on iterative methods (see the last issue), we saw that the fastest algorithm for the finest grid in `myproblem=1` was the AMD-Cholesky algorithm, which, on my computer, took

roughly 0.2 seconds; the storage was roughly five times that for the matrix. The fastest iterative method—conjugate gradients with an incomplete Cholesky preconditioner—took 0.9 seconds. My implementation of multigrid for this problem took four iterations and 8.2 seconds. Multigrid's virtue, though, is that if we want a finer mesh, we'll still probably get convergence in roughly four iterations; the number of iterations for the other algorithms increased with h, so eventually multigrid wins.

PROBLEM 4.

Repeat your experiment from Problem 3, but use $\kappa = 10$ and 100 and then $\kappa = -10$ and -100. (When $\kappa \neq 0$, the differential equation is called the *Helmholtz equation*.) The differential equation remains elliptic for positive κ but not for negative. How was multigrid convergence affected?

Answer:

The number of iterations remained four for $\kappa = 10$, 100, and -10, but for $\kappa = -100$, multigrid failed to converge. As noted in the iterative methods solution, we need a more complicated algorithm.

Note that the function `smooth.m` is a much faster implementation of Gauss-Seidel than that given in the iterative method's solution.

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. O'Leary is a member of SIAM, the ACM, and AWM. Contact her at oleary@cs.umd.edu; www.cs.umd.edu/users/oleary/.