

Computing derivatives by finite differences

Mark S. Gockenbach

In order to use an optimization code to solve a problem, the user must code the objective function (and constraint functions, if any). Coding the derivatives as well can be an imposing task for the user. For this reason, many optimization codes provide the option of approximating the necessary derivatives by finite differences.

1 Finite differences in one variable

1.1 The forward difference

To introduce the idea of a finite difference approximation, I begin with Taylor's theorem for a function $f : \mathbb{R} \rightarrow \mathbb{R}$:

$$f(x+h) = f(x) + f'(x)h + \frac{f''(x+\theta h)}{2}h^2, \quad \theta \in (0,1). \quad (1)$$

Solving the above equation for $f'(x)$ yields

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{f''(x+\theta h)}{2}h.$$

The first term on the right is an estimate of $f'(x)$ that is computable using only function values, and the second term is regarded as the error in the approximation. Assuming f is twice continuously differentiable, the following *forward difference* is an $O(h)$ approximation of $f'(x)$:

$$f'(x) \approx \frac{f(x+h) - f(x)}{h} \quad (2)$$

It would appear possible to approximate $f'(x)$ to any desired accuracy by using (2) and just taking h to be very small. However, the effect of finite precision arithmetic cannot be ignored. First of all, since x is represented using a finite number of digits, there exists a threshold $h_0 > 0$ such that, for all h with $|h| < h_0$, $x+h$ and x are indistinguishable in finite precision (this threshold, of course, depends on the particular floating point system being used). In other words, for h smaller than h_0 , the floating point operation $x_1 = x+h$ would result in $x_1 = x$. The threshold can be expressed in terms of *machine epsilon* ϵ , which can be defined as the distance from 1 to the next larger floating point number. For example, in IEEE double precision floating point arithmetic, $\epsilon = 2^{-52} \doteq 2.2 \cdot 10^{-16}$. In terms of ϵ , $h_0 = \epsilon|x|/2$. This suggests that there is an absolute limit to the accuracy with which a forward difference can approximate a derivative, namely

$$O\left(\frac{\epsilon|x|}{2}\right).$$

In fact, however, there is a more stringent restriction on h . The function f cannot be computed exactly in finite precision arithmetic and so the computed value of $f(x)$ generally differs from the exact value. I will call the computed value $\tilde{f}(x)$ and write

$$\tilde{f}(x) = f(x) + e(x).$$

If the error $e(x)$ is due only to round-off error, then it would be reasonable to assume that

$$|e(x)| \leq C\epsilon|f(x)|,$$

where C is a constant. It is important to realize, though, that e is likely to be a very nonsmooth function.

The computed forward difference has the form

$$\frac{\tilde{f}(x+h) - \tilde{f}(x)}{h},$$

and it differs from $f'(x)$ because of two types of errors: the *truncation error*, that is, the $O(h)$ error intrinsic to the forward difference approximation, and the error e in computing f .¹ I can analyze the total error as follows:

$$\begin{aligned} f'(x) - \frac{\tilde{f}(x+h) - \tilde{f}(x)}{h} &= f'(x) - \frac{f(x+h) - f(x)}{h} + \frac{f(x+h) - f(x)}{h} - \frac{\tilde{f}(x+h) - \tilde{f}(x)}{h} \\ &= -\frac{f''(x+\theta h)}{2}h + \frac{f(x+h) - \tilde{f}(x+h) - (f(x) - \tilde{f}(x))}{h} \\ &= -\frac{f''(x+\theta h)}{2}h - \frac{e(x+h) - e(x)}{h}. \end{aligned}$$

Since e is unlikely to be differentiable, the only bound available for the second term above is

$$\left| \frac{e(x+h) - e(x)}{h} \right| \leq \frac{C\epsilon(|f(x+h)| + |f(x)|)}{h} = \frac{2C\epsilon|f(x+\lambda h)|}{h}, \quad \lambda \in (0,1).$$

This suggests that the error in the forward difference due to the error in the computed f might blow up if h is taken too small, and in fact this occurs in practice (the reader can verify this with a simple example).

If $|f| \leq M_1$ and $|f''| \leq M_2$, then the error in the computed estimate of $f'(x)$ is then bounded by

$$\frac{M_2 h}{2} + \frac{2C\epsilon M_1}{h}. \quad (3)$$

Ideally, the step size h would be chosen to minimize (3). A simple calculation shows that the optimal value of h is

$$h = 2\sqrt{\frac{CM_1}{M_2}}\sqrt{\epsilon}. \quad (4)$$

Since the bounds M_1 , M_2 , and C (particularly the last two) are unlikely to be known, formula (4) contains only one piece of useful information: the step size h should be chosen to be proportional to $\sqrt{\epsilon}$. In the absence of further information, the usual practice is to choose h to be $|x|\sqrt{\epsilon}$, suitably safeguarded in case x is close to zero. The following formula is recommended:²

$$h = \sqrt{\epsilon} \max\{|x|, \text{typx}\} \text{sign}(x) \quad (5)$$

($\text{sign}(x)$ is 1 if $x \geq 0$ and -1 otherwise). Therefore, in practice, the forward difference formula can produce estimates with errors on the order of $\sqrt{\epsilon}|x|$.

In many realistic applications, there are significant errors in the computation of $f(x)$ that do not result simply from round-off. In this case, the machine epsilon ϵ can be replaced by an estimate η of the error in computing $f(x)$.

¹There is actually a third type of error, namely, the additional round-off error in computing the difference and the quotient in the forward difference; however, this error is negligible compared to the other errors.

²By Dennis and Schnabel [1].

1.2 The central difference

In an optimization problem, the fundamental optimality condition is $f'(x) = 0$, and algorithms are designed to satisfy this condition, at least approximately. For this reason, a highly accurate solution requires an accurate computation of the derivative. Sometimes a more accurate finite difference formula is preferred over the forward difference; the most common is the *central difference*

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2). \quad (6)$$

This formula and its error can be derived by expanding $f(x \pm h)$ in Taylor series, adding the two series, and solving for $f'(x)$. An analysis of the error in the *computed* central difference,

$$\frac{\tilde{f}(x+h) - \tilde{f}(x-h)}{2h}$$

shows that h should be proportional to $\epsilon^{1/3}$. This larger value of h is the reason that the central difference formula is more accurate in practice—a larger h reduces the errors propagated from errors in computing f . On the other hand, (6) has the disadvantage that it is twice as expensive as the forward difference formula, requiring two function values instead of one. (Of course, (2) also needs two function values, but in the context of an optimization algorithm, $f(x)$ must be computed anyway, so it is available for “free.”) Most optimization codes use the forward difference because of efficiency concerns, but some have an option to use central differences in case the user wants a more accurate final solution.

2 Finite differences for the gradient

If $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is smooth, then $\nabla f(x)$ can be estimated using finite differences to approximate each partial derivative in turn. For example, using forward differences,

$$\nabla f(x)_i = \frac{\partial f}{\partial x_i}(x) \doteq \frac{f(x + h_i e_i) - f(x)}{h_i},$$

where e_i is the i th standard basis vector. The vector $x + h_i e_i$ results from perturbing only the i th component of x ; it is a good idea to compute a different step size h_i for each partial derivative in case the components of x vary in size. Each h_i should be computed via (5).

It is important to notice that estimating the gradient using finite differences costs n times as much as evaluating the function ($2n$ times as much if central differences are used). This is a significant expense that may be unbearable if function evaluations are expensive.

3 Finite differences for the Jacobian

In an algorithm for solving nonlinear equations, the Jacobian of a vector-valued function is needed. The Jacobian is most conveniently estimated one column at a time:

$$\frac{\partial F}{\partial x_j}(x) \doteq \frac{F(x + h_j e_j) - F(x)}{h_j},$$

where $\partial F / \partial x_j$ represents the j th column of the Jacobian matrix. As in the case of the gradient, each component x_j of x should have its own step size h_j .

4 Finite differences for the Hessian

If the user provides a routine to compute the gradient of $f : \mathbf{R}^n \rightarrow \mathbf{R}$, then the Hessian can be viewed as the Jacobian of the gradient and estimated as in the previous section. The result H is unlikely to be exactly symmetric, so it is important to replace the H obtained by finite differences by

$$\frac{1}{2} (H + H^T),$$

which is necessarily symmetric.

In case the analytic gradient is not available, then the Hessian can be estimated using second-order finite differences, although doing so requires an additional

$$\frac{n^2 + 3n}{2}$$

function evaluations. The appropriate formula, which can be derived from Taylor's theorem, is

$$\frac{\partial^2 f}{\partial x_i \partial x_j}(x) = \frac{f(x + h_i e_i + h_j e_j) - f(x + h_i e_i) - f(x + h_j e_j) + f(x)}{h_i h_j} + O(\max\{|h_i|, |h_j|\}).$$

The main drawback in using finite differences to approximate the Hessian is the expense; in contrast to the case with the gradient, accuracy is not of primary importance.

References

- [1] J.E. Dennis, Jr. and R.B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice-Hall, Englewood Cliffs, 1983.