Numerical Differentiation

James Keesling

1 Theoretical Error in Approximating the Derivative

The most straightforward way to approximate the derivative would be to use the difference quotient used in the definition of the derivative.

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

For a small value of h, $\frac{f(x+h)-f(x)}{h}$ is approximately $f'(x)$. On the other hand, if we experiment with this formula we observe that if we take h to be too small, we get 0. Of course, it is not the case that all derivatives are 0. What causes this error is that if h is so small that $x + h = x$ in the way x is represented in the computer, then the numerator in the expression $\frac{f(x+h)-f(x)}{h}$ is zero.

So, what is the best value of h to use to get the most digits accurate in estimating $f'(x)$? The error in our formula is composed of two parts. The first and easiest to analyze is the theoretical error. The second is caused by roundoff error. Roundoff error comes from representing the numbers x and $x + h$ as floating point numbers. It can also arise through numerical inaccuracies in calculating the function f.

The theoretical error is given by a power series representation for $\frac{f(x+h)-f(x)}{h}$.

$$\begin{aligned} \frac{f(x+h) - f(x)}{h} &= \frac{f(x) + f'(x) \cdot h + f''(x) \cdot \frac{h^2}{2} + \cdots - f(x)}{h} \\ &= f'(x) + \frac{f''(x)}{2} \cdot h + \frac{f'''}{3!} \cdot h^2 + \cdots \end{aligned}$$

For small values of h the error is approximately $\frac{f''(x)}{2} \cdot h$. Obviously, the smaller h is, the more accurate the representation of $f'(x)$. However, there is also a roundoff error that needs to be taken into account.

2 Roundoff Error

Let us suppose that when we represent the value of $f(x)$ in the computer, there is an error of ϵ. The worst case in representing the difference quotient would be if the error representing

$f(x+h)$ and the error representing $f(x)$ was the same magnitude and opposite sign. Then the total error representing $f'(x)$ would be the following expression.

$$E(h) = K \cdot h + \frac{2 \cdot \epsilon}{h}$$

So, while the theoretical error goes to zero as $h \to 0$, the roundoff error goes to infinity as $h \to 0$. What is the best choice of h to give us the best estimate of the derivative? We need to minimize the error $E(h)$.

$$\frac{dE(h)}{dh} = K - \frac{2 \cdot \epsilon}{h^2}$$

If we set this equal to zero, we come up with a "best" h_0 that will give us the best accuracy in estimating $f'(x)$. This is easily seen to be

$$h_0 = \sqrt{\frac{2 \cdot \epsilon}{K}}.$$

We assume that $\sqrt{\frac{2 \cdot \epsilon}{K}} \approx 1$. Suppose that $\epsilon \approx 10^{-n}$. Then

$$h_0 \approx 10^{-\frac{n}{2}}$$

and

$$E(h_0) \approx 10^{\frac{n}{2}}.$$

In terms of accuracy, if our computer gives us n digits of accuracy and we use the difference quotient to estimate the derivative, we can expect to get about half of those digits correct by using the optimal choice of h.

On the other hand, we can use other formulas to estimate the derivative. For instance, the divided difference formula.

$$\begin{aligned} f'(x) &\approx \frac{f(x+h) - f(x-h)}{2 \cdot h} \\ &= f'(x) + \frac{f'''(x)}{3!} \cdot h^2 + \cdots \end{aligned}$$

What is the advantage of this formula? Well, the error function combining the theoretical error together with the roundoff error is now given by

$$E(h) = K \cdot h^2 + \frac{\epsilon}{h}.$$

We get a different optimal h_0.

$$h_0 = \sqrt[3]{\frac{\epsilon}{2K}}$$

Now instead of getting half of the digits correct, we can get two-thirds correct. There are other ways to approximate the derivative so that we can get more and more digits correct up to the limit of the accuracy of the computer itself.

3 Estimating Higher Derivatives Using Multiple Points

Now suppose that we want to estimate the k^{th} derivative of $f(x)$ at the point x_0 and wish to use the points $\{x_0 - n \cdot h, x_0 - (n-1) \cdot h, \ldots, x_0, x_0 + h, \ldots, x_0 + n \cdot h\}$ in making the estimate.

Consider the following formula.

$$\frac{A_{-n} \cdot f(x_0 - n \cdot h) + \cdots + A_0 \cdot f(x_0) + \cdots A_n \cdot f(x_0 + n \cdot h)}{h^k}$$

We can use a Taylor series expansion to determine what the coefficients $\{A_{-n}, \ldots, A_0, \ldots, A_n\}$ must be to estimate $f^{(k)}(x_0)$ and what the theoretical error will be for a particular value of h. To illustrate, let us use particular values of k and n, $k = 2$ and $n = 2$. We will end up with five equations and five unknowns. The unknowns will be $\{A_{-2}, A_{-1}, A_0, A_1, A_2\}$. We get the equations from the Taylor expansion. We want the Taylor expansion to be equal to $f''(x_0)$ with the smallest error possible.

We get the following equations.

$$\begin{aligned}
A_{-2} + A_{-1} + A_0 + A_1 + A_2 &= 0 \\
-2 \cdot A_{-2} - A_{-1} + 0 \cdot A_0 + A_1 + 2 \cdot A_2 &= 0 \\
-4 \cdot A_{-2} - A_{-1} + 0 \cdot A_0 + A_1 + 4 \cdot A_2 &= \frac{h^2}{2} \\
-8 \cdot A_{-2} - A_{-1} + 0 \cdot A_0 + A_1 + 8 \cdot A_2 &= 0 \\
-16 \cdot A_{-2} - A_{-1} + 0 \cdot A_0 + A_1 + 16 \cdot A_2 &= 0
\end{aligned}$$

We define M to be the matrix below, A be the vector of coefficients, and B vector whose entries are the right hand side of the equations above. We then can convert the above set of equations into a vector calculation.

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 1 & 2 \\ 4 & 1 & 0 & 1 & 4 \\ -8 & -1 & 0 & 1 & 8 \\ 16 & 1 & 0 & 1 & 16 \end{bmatrix} A = \begin{bmatrix} A_{-2} \\ A_{-1} \\ A_0 \\ A_1 \\ A_2 \end{bmatrix} B = \begin{bmatrix} 0 \\ 0 \\ \frac{2}{h^2} \\ 0 \\ 0 \end{bmatrix}$$

$$A = M^{-1} \cdot B$$

When we determine the coefficients, we get the following method of approximating the second derivative.

$$f''(x_0) = \frac{-f(x_0 - 2h) + 16 \cdot f(x_0 - h) - 30 \cdot f(x_0) + 16 \cdot f(x_0 + h) - f(x_0 + 2h)}{12h^2} + O(h^3)$$

We can estimate the error by the following formula.

$$E(h) = K \cdot h^3 + \frac{\epsilon}{h^2}$$

We get the following approximate value for the optimal h.

$$h_0 \approx \sqrt[5]{\epsilon}$$

We should be able to get approximately three-fifths of the digits correct using this h_0.

4 TI-89 Program Calculating the Required Coefficients to Estimate $f^{(k)}(x)$

Let us now consider a very general case. We want to estimate $f^{(k)}(x)$ using the m points

$$\{x + n_1 \cdot h, x + n_2 \cdot h, \ldots, x + n_m \cdot h\}.$$

We assume that $m > k$. We also assume that the n_i's are distinct. We want to determine a formula of the form

$$f^{(k)}(x) = \sum_{i=1}^{m} A_i \cdot f(x + n_i \cdot h) + O(h^{m-k}).$$

We also want to know what would be the optimal h and with what accuracy we can estimate $f^{(k)}(x)$ using the formula.

Reflecting on the example given in §3, it is easy to see how to handle the general case. Let M and B be the following matrices.

$$M = \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ n_1 & n_2 & n_3 & \cdots & n_m \\ n_1^2 & n_2^2 & n_3^2 & \cdots & n_m^2 \\ \vdots & \vdots & & & \vdots \\ n_1^{m-1} & n_2^{m-1} & n_3^{m-1} & \cdots & n_m^{m-1} \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \frac{k!}{h^k} \\ \vdots \\ 0 \end{bmatrix}$$

Then the matrix of coefficients is given by

$$A = M^{-1} \cdot B.$$

Note M is the transpose of the Vandermonde matrix for the vector $[n_1, n_2, \ldots, n_m]$.

Below is the TI-89 program. The input variable a is the vector $[n_1, n_2, \ldots, n_m]$. The input variable n is $m-1$. The input variable k is the level of the derivative $f^{(k)}(x)$ to be approximated. The program can be tested using the example worked out in §3.

```
:numerdif(a,k)
:Prgm
:dim(a)[2]-1 → n
:newMat(n+1,n+1) → vander
:newMat(n+1,1) → coef
:newMat(n+1,1) → temp
:vanderm(a)
:k!/h∧k → temp[k+1,1]
:(vander^T)∧(-1)*temp → coef
:EndPrgm
```

Note that the numerator of the formula will be accurate to $O(h^m)$. Since the denominator will have h^k as a factor, the overall accuracy will be $O(h^{m-k})$.

5 Another Method of Numerical Differentiation

In this section we give an alternative approach to numerically estimating the derivative. Consider the general case in the previous section. We want to estimate $f^{(k)}(x)$ using the m points

$$\{x + n_1 \cdot h, x + n_2 \cdot h, \ldots, x + n_m \cdot h\}.$$

However, the way we will do it is to fit a Lagrange polynomial to the function values at these points. We then take the k^{th} derivative of this polynomial at the point x. It turns out that this method of estimating the derivative will be the same as the one that we have already described. As a result, this approach gives us another way to determine the coefficients given in §4. Let $P(x)$ be the Lagrange polynomial through the points

$$\{(x + n_1 \cdot h, f(x + n_1 \cdot h)), (x + n_2 \cdot h, f(x + n_2 \cdot h)), \ldots, (x + n_m \cdot h, f(x + n_m \cdot h))\}.$$

Let $L_i(x)$ be the Lagrange polynomial through the points

$$\{(x + n_1 \cdot h, 0), (x + n_2 \cdot h, 0), \ldots, (x + n_i \cdot h, 1), \ldots, (x + n_m \cdot h, 0)\}.$$

Then

$$f(x) = \sum_{i=1}^{m} f(x + n_i \cdot h) \cdot L_i(x).$$

Taking the k^{th} derivative of both sides we get the following formula.

$$f^{(k)}(x_0) = \sum_{i=1}^{m} L_i^{(k)}(x_0) \cdot f(x_0 + n_i \cdot h)$$

So, it is clear that $A_i = L_i^{(k)}(x_0)$.

Note that $L_i^{(k)}(x_0)$ will depend on h as well as x_0. Let us apply this method to determine A_{-1} in the example in §3 approximating $f''(x_0)$ using the points $\{x_0 - 2h, x_0 - h, x_0, x_0 + h, x_0 + 2h\}$. We get

$$L_{-1}(x) = \frac{2}{3} \cdot \frac{(x - x_0)}{h} + \frac{2}{3} \cdot \frac{(x - x_0)^2}{h^2} + \frac{1}{6} \cdot \frac{(x - x_0)^3}{h^3} - \frac{1}{6} \cdot \frac{(x - x_0)^4}{h^4}.$$

Thus we get

$$L''_{-1}(x_0) = \frac{4}{3 \cdot h^2} = \frac{16}{12 \cdot h^2}.$$

This was the coefficient that we got using the previous approach.

This approach also allows us to estimate the k^{th} derivative of $f(x)$ at x_0 without computing the coefficients in our first method. We can determine the points $\{x_0 + n_1 \cdot h, x_0 + n_2 \cdot h, \ldots, x_0 + n_m \cdot h\}$ for a particular value of h. We then compute the values of $f(x)$ at each of these points. Then we determine the Lagrange polynomial $P(x)$ which takes on the values of $f(x)$ at the interpolation points. Then compute the k^{th} derivative of $P(x)$ and evaluate that derivative at the point x_0.

6 Two Useful Programs

We have developed programs that do all of the above steps except one. We need to produce the vector $[f(x_0+n_1 \cdot h), f(x_0+n_2 \cdot h, \ldots, f(x_0+n_m \cdot h)]$. Once we have that vector of values, we could either compute the Lagrange polynomial as in §5 or we could use it together with the coefficients developed in §4 to estimate the derivative by that method. The program below will produce this vector of function values. The variable f in the program is the function with assumed variable x. The variable a is the vector of values at which we want the function f evaluated. The variable n is such that $n + 1$ is the length of the vector a. The output *vectev* is the vector of function values at the vector of values.

```
:vecteval(f,a)
:Prgm
:dim(a)[2]-1 → n
:f → g
:newMat(n+1,1) → vectev
:For i,0,n
:g|x=a[i+1,1] → vectev[i+1,1]
:EndFor
:Disp vectev
:EndPrgm
```

Here is a program that numerically evaluates the derivative. The variable f is the function that is to be differentiated at the point a. The variable k is the derivative to be estimated, $f^{(k)}(a)$. The function variable is assumed to be x. The variable b is the vector $\{n_1, n_2, \ldots, n_{m+1}\}$. The output variable p gives a formula for estimating the derivative that depends on h. One can then determine estimates by computing $p|h = c$ in the command line of the home screen for various values of c.

```
:numdiff(f,a,b,k)
:Prgm
:dim(b)[2]-1 → m
:numerdif(b,k)
:0 → p
:For i,0,m
:p+coef[i+1,1]*f|x=(a+b[1,i+1]*h) → p
:EndFor
:Disp p
:EndPrgm
```