Chapter 2: Finite-Difference Approximations to Derivatives

Copyright 2013, David A. Randall

2.1 Finite-difference quotients

Consider the derivative \( \frac{df}{dx} \), where \( f = f(x) \), and \( x \) is the independent variable (which could be either space or time). Finite-difference methods represent the continuous function \( f(x) \) by a set of values defined at a finite number of discrete points in a specified (spatial or temporal) region. Thus, we usually introduce a “grid” with discrete points where the variable \( f \) is defined,

using the notation \( f_j = f(x_j) = f(j\Delta x) \), we can define

the **forward difference** at the point \( j \) by \( f_{j+1} - f_j \),

\[ (1) \]
the backward difference at the point $j$ by $f_j - f_{j-1}$, and

$$ \frac{df}{dx}_j \approx \frac{f_j - f_{j-1}}{\Delta x}; $$

(2)

the centered difference quotient at the point $j + \frac{1}{2}$ by $f_{j+1} - f_j$.

$$ \frac{df}{dx}_{j+\frac{1}{2}} \approx \frac{f_{j+1} - f_j}{\Delta x}; $$

(3)

Note that $f$ itself is not defined at the point $j + \frac{1}{2}$. From (1) - (3) we can define the following “finite-difference quotients:” the forward-difference quotient at the point $j$:

$$ \frac{df}{dx}_j \equiv \frac{f_{j+1} - f_j}{\Delta x}; $$

(4)

the backward-difference quotient at the point $j$:

$$ \frac{df}{dx}_j \equiv \frac{f_j - f_{j-1}}{\Delta x}; $$

(5)

and the centered-difference quotient at the point $j + \frac{1}{2}$:

$$ \frac{df}{dx}_{j+\frac{1}{2}} \equiv \frac{f_{j+1} - f_j}{\Delta x}. $$

(6)

In addition, the centered-difference quotient at the point $j$ can be defined by

$$ \frac{df}{dx}_j \equiv \frac{f_{j+1} - f_{j-1}}{2\Delta x}. $$

(7)

Since (4) and (5) employ the values of $f$ at two points, they are sometimes referred to as two-point approximations. On the other hand, (6) and (7) are three-point approximations. When $x$ is time, the time point is frequently referred to as a “level.” In that case, (4) and (5) can be referred to as two-level approximations and (6) and (7) as three-level approximations.

How accurate are these finite-difference approximations? “Accuracy” can be measured in a variety of ways, as we shall see. One measure of accuracy is truncation error. The term refers to
the finite truncation of an infinite series expansion. As an example, consider the forward difference quotient

\[ \left( \frac{df}{dx} \right)_j \equiv \frac{f_{j+1} - f_j}{\Delta x} \equiv \frac{f((j+1)\Delta x) - f(j\Delta x)}{\Delta x}. \] 

(8)

Expand \( f \) in a Taylor series about the point \( x_j \) to obtain

\[ f_{j+1} = f_j + \Delta x \left( \frac{df}{dx} \right)_j + \frac{(\Delta x)^2}{2!} \left( \frac{d^2 f}{dx^2} \right)_j + \frac{(\Delta x)^3}{3!} \left( \frac{d^3 f}{dx^3} \right)_j + \cdots + \frac{(\Delta x)^{n-1}}{(n-1)!} \left( \frac{d^{n-1} f}{dx^{n-1}} \right)_j + \cdots. \]

(9)

This can be rearranged to

\[ \frac{f_{j+1} - f_j}{\Delta x} = \left( \frac{df}{dx} \right)_j + \epsilon, \]

(10)

where

\[ \epsilon \equiv \Delta x \left( \frac{d^2 f}{dx^2} \right)_j + \frac{(\Delta x)^2}{2!} \left( \frac{d^3 f}{dx^3} \right)_j + \cdots + \frac{(\Delta x)^{n-2}}{(n-1)!} \left( \frac{d^{n-1} f}{dx^{n-1}} \right)_j + \cdots. \]

(11)

can be called the error. The expansion (9) can be derived without any assumptions or approximations except that the indicated derivatives exist (Arfken, 1985; for a very brief review see the QuickStudy on Taylor's Series). The terms of (10) that are lumped together to make \( \epsilon \) are called the “truncation error.” If \( \Delta x \) is small enough, the leading term on the right-hand side of (11) will be the largest part of the error. The lowest power of \( \Delta x \) that appears in the truncation error is called the “order of accuracy” of the corresponding difference quotient. For example, the leading term of (11) is of order \( \Delta x \) or \( O(\Delta x) \) and so we say that (10) is a first-order approximation or an approximation of first-order accuracy. Obviously (5) is also first-order accurate.

Similar analyses of (6) and (7) show that they are of second-order accuracy. For example, we can write

\[ f_{j-1} = f_j + \left( \frac{df}{dx} \right)_j (-\Delta x) + \left( \frac{d^2 f}{dx^2} \right)_j \frac{(-\Delta x)^2}{2!} + \left( \frac{d^3 f}{dx^3} \right)_j \frac{(-\Delta x)^3}{3!} + \cdots. \]

(12)
Subtracting (12) from (9) gives

\[ f_{j+1} - f_{j-1} = 2 \left( \frac{df}{dx} \right)_j (\Delta x) + 2 \left( \frac{d^3 f}{dx^3} \right)_j \left[ (\Delta x)^3 \right] + \cdots \text{ odd powers only,} \]

which can be rearranged to

\[ \left( \frac{df}{dx} \right)_j = f_{j+1} - f_{j-1} - \frac{d^3 f}{dx^3} \left( \frac{\Delta x^2}{3!} \right) + O(\Delta x^4). \]  

(13)

Similarly,

\[ \left( \frac{df}{dx} \right)_{j+\frac{1}{2}} \equiv f_{j+1} - f_j - \frac{d^3 f}{dx^3} \left( \frac{\Delta x^2}{3!} \right) + O(\Delta x^4). \]  

(14)

From (14) and (15), we see that

\[
\begin{align*}
\text{Error of } \left( \frac{df}{dx} \right)_j & \equiv \left( \frac{d^3 f}{dx^3} \right)_j \frac{\Delta x^2}{3!} = 4 \left( \frac{d^3 f}{dx^3} \right)_{j+\frac{1}{2}} \equiv 4. \\
\text{Error of } \left( \frac{df}{dx} \right)_{j+\frac{1}{2}} & \equiv \left( \frac{d^3 f}{dx^3} \right)_{j+\frac{1}{2}} \frac{\Delta x^2}{3!} = 4 \left( \frac{d^3 f}{dx^3} \right)_j \equiv 4.
\end{align*}
\]

(15)

This shows that the error of \( \left( \frac{df}{dx} \right)_j \) is about four times as large as the error of \( \left( \frac{df}{dx} \right)_{j+\frac{1}{2}} \), even though both finite-difference quotients have second-order accuracy. The point is that the “order of accuracy” tells how rapidly the error changes as the grid is refined, but it does not tell how large the error is for a given grid size. It is possible for a scheme of low-order accuracy to give a more accurate result than a scheme of higher-order accuracy, if a finer grid spacing is used with the low-order scheme.

Suppose that the leading (and dominant) term of the error has the form

\[ \varepsilon \equiv C (\Delta x)^p. \]  

(17)

From (17) we see that \( \ln(\varepsilon) \equiv p \ln(\Delta x) + \ln(C) \), and so
\[
\frac{d\left[\ln(\varepsilon)\right]}{d\left[\ln(\Delta x)\right]} \equiv p.
\]

(18)

The implication is that if we plot \(\ln(\varepsilon)\) as a function of \(\ln(\Delta x)\) (i.e., plot the error as a function of the grid spacing on “log-log” paper), we will get (approximately) a straight line whose slope is \(p\). This is a simple way to determine \textit{empirically} the order of accuracy of a finite-difference quotient. Of course, in order to carry this out in practice it is necessary to compute the error of the finite-difference approximation, and that can only be done when the exact derivative is known. For that reason, this empirical approach is usually implemented by using an analytical “test function.”

2.2 \textit{Difference quotients of higher accuracy}

Suppose that we write

\[
\frac{f_{j+2} - f_{j-2}}{4\Delta x} = \left(\frac{df}{dx}\right)_j + \frac{1}{3!}\left(\frac{d^3 f}{dx^3}\right)_j (2\Delta x)^2 + \cdots \text{ even powers only.}
\]

(19)

Here we have written a centered difference using the points \(j + 2\) and \(j - 2\) instead of \(j + 1\) and \(j - 1\), respectively. It should be clear that (19) is second-order accurate, although for any given value of \(\Delta x\) the error of (19) is expected to be larger than the error of (14). We can combine (14) and (19) with a weight, \(w\), so as to obtain a “hybrid” approximation to \(\left(\frac{df}{dx}\right)_j\):

\[
\left(\frac{df}{dx}\right)_j = w\left(\frac{f_{j+1} - f_{j-1}}{2\Delta x}\right) + (1-w)\left(\frac{f_{j+2} - f_{j-2}}{4\Delta x}\right)

- \frac{w}{3!}\left(\frac{d^3 f}{dx^3}\right)_j (\Delta x)^2 - \frac{(1-w)}{3!}\left(\frac{d^3 f}{dx^3}\right)_j (2\Delta x)^2 + O[(\Delta x)^4].
\]

(20)

Inspection of (20) shows that we can force the coefficient of \((\Delta x)^2\) to vanish by choosing

\[w + (1-w)4 = 0, \text{ or } w = 4/3.\]

(21)

With this choice, (20) reduces to
\[
\left( \frac{df}{dx} \right)_j = \frac{4}{3} \left( \frac{f_{j+1} - f_{j-1}}{2\Delta x} \right) - \frac{1}{3} \left( \frac{f_{j+2} - f_{j-2}}{4\Delta x} \right) + O\left( (\Delta x)^4 \right).
\]

This is a fourth-order accurate scheme. The derivation given above, in terms of a weighted combination of two second-order schemes, can be interpreted as a linear extrapolation of the value of the finite-difference expression to a smaller grid size, as schematically illustrated in Fig. 2.2.

Figure 2.2: Schematic illustrating an interpretation of the fourth-order difference in terms of an extrapolation from two second-order differences. The extrapolation produces approximately the same accuracy as a centered second-order scheme with a grid spacing of \( \frac{2}{3} \Delta x \).

2.2. The concepts of extrapolation and interpolation will be discussed in more detail, later in this chapter.

Is there a more systematic way to construct schemes of any desired order of accuracy? The answer is “Yes,” and one such approach is as follows. Suppose that we write a finite-difference approximation to \( \left( \frac{df}{dx} \right)_j \) in the following somewhat generalized form:
Here the \( a_k \) are coefficients or “weights,” which are undetermined at this point. *We can design a scheme by choosing suitable expressions for the \( a_k \).* In most schemes, all but a few of the \( a_k \) will be zero, so that the sum (23) in will actually involve only a few (non-zero) terms. In writing (23), we have assumed for simplicity that \( \Delta x \) is a constant; this assumption will be relaxed soon. The index \( k \) in (23) is a counter that is zero at our “home base,” grid point \( j \). For \( k < 0 \) we count to the left, and for \( k > 0 \) we count to the right. According to (23), our finite-difference approximation to \( \frac{df}{dx} \) has the form of a weighted sum of values of \( f \) at various grid points in the vicinity of point \( j \). Every finite-difference approximation that we have considered so far does indeed have this form, but you should be aware that there are (infinitely many!) schemes that do not have this form; a few of them will be discussed later.

Introducing a Taylor series expansion, we can write

\[
f(x_j + k\Delta x) = f_j^0 + f_j^1(k\Delta x) + f_j^2\frac{(k\Delta x)^2}{2!} + f_j^3\frac{(k\Delta x)^3}{3!} + \cdots
\]  

(24)

Here we introduce a new short-hand notation: \( f_j^n \) is the \( n \)th derivative of \( f \), evaluated at the point \( j \). Using (24), we can rewrite (23) as

\[
f_j^1 \equiv \frac{1}{\Delta x} \sum_{k=-\infty}^{\infty} a_k \left[ f_j^0 + f_j^1(k\Delta x) + f_j^2\frac{(k\Delta x)^2}{2!} + f_j^3\frac{(k\Delta x)^3}{3!} + \cdots \right].
\]  

(25)

By inspection of (25), we see that in order to have at least first-order accuracy, we need

\[
\sum_{k=-\infty}^{\infty} a_k = 0 \quad \text{and} \quad \sum_{k=-\infty}^{\infty} ka_k = 1.
\]  

(26)

To have at least second-order accuracy, we must impose an additional requirement:

\[
\sum_{k=-\infty}^{\infty} k^2a_k = 0.
\]  

(27)
In general, to have at least \( n \)th-order accuracy, we must require that
\[
\sum_{k=\infty}^{\infty} k^m a_k = \delta_{m,1} \quad \text{for} \quad 0 \leq m \leq n.
\] (28)

Here \( \delta_{m,1} \) is the Kronecker delta. In order to satisfy (28), we must solve a system of \( n+1 \) linear equations for the \( n+1 \) unknown coefficients \( a_k \).

According to (28), a scheme of \( n \)th-order accuracy can be constructed by satisfying \( n+1 \) equations. In particular, because (26) involves two equations, a first-order scheme has to involve at least two grid points, i.e., there must be at least two non-zero values of \( a_k \). That is pretty obvious. Note that we could make a first-order scheme that used fifty grid points if we wanted to -- but then, why would we want to? A second-order scheme must involve at least three grid points. A scheme that is parsimonious in its use of points is called “compact.”

Consider a simple example. Still assuming a uniform grid, a first order scheme for \( f^i_j \) can be constructed using the points \( j \) and \( j+1 \) as follows. From (26), we get \( a_0 + a_1 = 0 \) and \( a_1 = 1 \). Obviously we must choose \( a_0 = -1 \). Substituting into (23), we find that the scheme is given by
\[
f^i_j \equiv \frac{f(x_{j+1}) - f(x_j)}{\Delta x},
\] i.e., the one-sided difference discussed earlier. In a similar way, we can also construct a first-order scheme using the points \( j \) and \( j-1 \), with another one-sided result. If we choose the points \( j+1 \) and \( j-1 \), imposing the requirements for first-order accuracy, i.e., (27), will actually give us the centered second-order scheme, i.e.,
\[
f^i_j \equiv \frac{f(x_{j+1}) - f(x_{j-1})}{2\Delta x}
\] because (27) is satisfied “accidentally” or “automatically” -- by luck, we manage to satisfy three equations using only two unknowns. If we choose the three points \( j-1 \), \( j \), and \( j+1 \), and require second-order accuracy, we get exactly the same centered scheme, because \( a_0 \) turns out to be zero.

Next, we work out a generalization of the family of schemes given above, for the case of (possibly) non-uniform grid spacing. Eq. (23) is replaced by
\[
f^i_j \equiv \sum_{k=\infty}^{\infty} b_{j,k} f(x_{j+k}).
\] (29)

Note that, since \( \Delta x \) is no longer a constant, the factor of \( \frac{1}{\Delta x} \) that appears in (23) has been omitted in (29), and in view of this, in order to avoid notational confusion, we have replaced the...
symbol $a_k$ by $b_{j,k}$. The subscript $j$ is included on $b_{j,k}$ because the grid can be non-uniform; as will be demonstrated below, for a given value of $k$ the coefficients are different for different values of $j$, i.e., at different places on the non-uniform grid. Similarly, (24) is replaced by

$$f(x_{j+k}) = f_j^0 + f_j^1 (x_{j+k} - x_j) + f_j^2 \left( \frac{(x_{j+k} - x_j)^2}{2!} \right) + f_j^3 \left( \frac{(x_{j+k} - x_j)^3}{3!} \right) + \cdots$$

(30)

Substitution of (30) into (29) gives

$$f_j^l \equiv \sum_{k=-\infty}^{\infty} b_{j,k} \left[ f_j^0 + f_j^1 (x_{j+k} - x_j) + f_j^2 \left( \frac{(x_{j+k} - x_j)^2}{2!} \right) + f_j^3 \left( \frac{(x_{j+k} - x_j)^3}{3!} \right) + \cdots \right].$$

(31)

To have first-order accuracy with (31), we must require that

$$\sum_{k=-\infty}^{\infty} b_{j,k} = 0 \quad \text{and} \quad \sum_{k=-\infty}^{\infty} b_{j,k} (x_{j+k} - x_j) = 1, \text{ for all } j.$$

(32)

It may appear that when we require first-order accuracy by enforcing (32), the leading term of the error in (31), namely, $\sum_{k=-\infty}^{\infty} b_{j,k} f_j^2 \left( \frac{(x_{j+k} - x_j)^2}{2!} \right)$ will be of order $(\Delta x)^2$, but this is not true because, as shown below, $b_k \sim \frac{1}{\Delta x}$. To achieve second-order accuracy with (31), we must require, in addition to (32), that

$$\sum_{k=-\infty}^{\infty} b_{j,k} (x_{j+k} - x_j)^2 = 0 \text{ for all } j.$$

(33)

In general, to have at least $n$th-order accuracy, we must require that

$$\sum_{k=-\infty}^{\infty} b_k (x_{j+k} - x_j)^m = \delta_{m,1} \text{ for all } j, \text{ and for } 0 \leq m \leq n.$$

(34)

As an example, the first-order accurate scheme using the points $j$ and $j+1$ must satisfy the two equations corresponding to (32), i.e., $b_{j,0} + b_{j,1} = 0$ and $b_{j,1} = \frac{1}{x_{j+1} - x_j}$, so that
\[ b_{j,0} = \frac{-1}{x_{j+1} - x_j} \]. From (29), the scheme is
\[ f_j^1 = \frac{f(x_{j+1}) - f(x_j)}{x_{j+1} - x_j} \], which, not unexpectedly, has the same form as the result that we obtained for the case of the uniform grid. The non-uniformity of the grid is irrelevant when we consider only two points.

To obtain a second-order accurate approximation to \( f_j^1 \) on an arbitrary grid, using the three points, \( j-1, j, \) and \( j+1 \), we must require, from (32) that
\[ b_{j,-1} + b_{j,0} + b_{j,1} = 0 \text{ and } b_{j,-1}(x_{j-1} - x_j) + b_{j,1}(x_{j+1} - x_j) = 1, \] which suffice for first-order accuracy, and additionally from (34) that
\[ b_{j,-1}(x_{j-1} - x_j)^2 + b_{j,1}(x_{j+1} - x_j)^2 = 0. \] The solution of this system of three equations is
\[ b_{j,-1} = \left( \frac{-1}{x_{j+1} - x_{j-1}} \right) \left( \frac{x_{j+1} - x_j}{x_j - x_{j-1}} \right), \] \[ b_{j,0} = \left( \frac{1}{x_{j+1} - x_{j-1}} \right) \left[ \left( \frac{x_{j+1} - x_j}{x_j - x_{j-1}} \right) \left( \frac{x_j - x_{j-1}}{x_{j+1} - x_j} \right) \right], \] \[ b_{j,1} = \left( \frac{1}{x_{j+1} - x_{j-1}} \right) \left( \frac{x_j - x_{j-1}}{x_{j+1} - x_j} \right). \] For the case of uniform grid-spacing this reduces to the familiar centered second-order scheme.

Here is a simple and very practical question: Suppose that we use a scheme that has second-order accuracy on a uniform grid, but we go ahead and apply it on a non-uniform grid. What happens? As a concrete example, we use the scheme
\[ f_j^1 = \frac{f(x_{j+1}) - f(x_{j-1})}{x_{j+1} - x_{j-1}}. \] By inspection, we have
\begin{align}
  b_{j,-1} &= \frac{-1}{x_{j+1} - x_{j-1}}, \\
  b_{j,0} &= 0, \\
  b_{j,1} &= \frac{1}{x_{j+1} - x_{j-1}}.
\end{align}

(Eq. 41) - (43) do satisfy both of the conditions in (35), so that the scheme does have first-order accuracy, even on the non-uniform grid. Eq. (37) - (39) are not satisfied, however, so it appears that second-order accuracy is lost.

This argument is a bit too hasty, however. Intuition suggests that, if the grid-spacing varies slowly enough, the scheme given by (40) should be nearly as accurate as if the grid-spacing were strictly constant. Intuition can never prove anything, but it can suggest ideas. Let's pursue this idea to see if it has merit. Define \( \Delta x_j \equiv x_{j+1} - x_j \) for all \( j \), and let \( \delta \) be the grid spacing at some reference grid point. We write the centered second-order scheme appropriate to a uniform grid, but apply it on a non-uniform grid:

\[
  f_j^1 \equiv \frac{f_{j+1} - f_{j-1}}{x_{j+1} - x_{j-1}} = \left[ f_j + f_j^1(\Delta x)_{j+1/2} + \frac{1}{2!} f_j^2(\Delta x)^2_{j+1/2} + O(\delta^3) \right] - \left[ f_j - f_j^1(\Delta x)_{j-1/2} + \frac{1}{2!} f_j^2(\Delta x)^2_{j-1/2} + O(\delta^3) \right] \frac{(\Delta x)_{j+1/2} + (\Delta x)_{j-1/2}}{(\Delta x)_{j+1/2} - (\Delta x)_{j-1/2}}.
\]

This can be simplified to

\[
  f_j^1 \equiv f_j^1 + \frac{1}{2!} f_j^2 \left[ (\Delta x)_{j+1/2} - (\Delta x)_{j-1/2} \right] + O(\delta^2).
\]

There is indeed a “first-order term” in the error, as expected, but notice that it is proportional to \( (\Delta x)_{j+1/2} - (\Delta x)_{j-1/2} \), which is the difference in the grid spacing between neighboring points, i.e., it is a “difference of a difference.” If the grid spacing varies slowly enough, this term will be proportional to \( \delta^3 \), i.e., it will be second-order. For example, if \( (\Delta x)_{j+1/2} = \delta(1 + \alpha x_{j+1/2}) \), then
\[
\left[ (\Delta x)_{j+1/2} - (\Delta x)_{j-1/2} \right] = \delta(1 + \alpha x_{j+1/2}) - \delta(1 + \alpha x_{j-1/2}) = \delta\alpha(x_{j+1/2} - x_{j-1/2}) \sim O(\delta^2),
\]

provided that \( \alpha \sim O(1) \) or smaller.

Finally, we observe that a very similar approach can be used to derive approximations to higher-order derivatives of \( f \). For example, to derive approximations to the second derivative, \( f_j^2 \), on a (possibly) non-uniform grid, we write

\[
f_j^2 \equiv \sum_{k=-\infty}^{\infty} c_{j,k} f(x_{j+k}).
\]

(47)

Obviously, it is going to turn out that \( c_{j,k} \sim \frac{1}{(\Delta x)^2} \). Substitution of (30) into (47) gives

\[
f_j^2 \equiv \sum_{k=-\infty}^{\infty} c_{j,k} \left[ f_j^0 + f_j^1 (x_{j+k} - x_j) + f_j^2 \frac{(x_{j+k} - x_j)^2}{2!} + f_j^3 \frac{(x_{j+k} - x_j)^3}{3!} + \ldots \right],
\]

(48)

A first-order accurate approximation is ensured if we enforce the three conditions

\[
\sum_{k=-\infty}^{\infty} c_{j,k} = 0, \text{ and } \sum_{k=-\infty}^{\infty} c_{j,k} (x_{j+k} - x_j)^2 = 2!, \text{ and } \sum_{k=-\infty}^{\infty} c_{j,k} (x_{j+k} - x_j) = 0, \text{ for all } j.
\]

(49)

To achieve a second-order accurate approximation to the second derivative, we must additionally require that

\[
\sum_{k=-\infty}^{\infty} c_{j,k} (x_{j+k} - x_j)^3 = 0, \text{ for all } j.
\]

(50)

In general, to have an \( n \)th-order accurate approximation to the second derivative, we must require that

\[
\sum_{k=-\infty}^{\infty} c_{j,k} (x_{j+k} - x_j)^m = (2!)^m \delta_m^2 \text{ for all } j, \text{ and for } 0 \leq m \leq n+1.
\]

(51)

Earlier we showed that, in general, a second-order approximation to the first derivative must involve a minimum of three grid points, because three conditions must be satisfied [i.e.,...
Now we see that, in general, a second-order approximation to the second derivative must involve four grid points, because four conditions must be satisfied, i.e., (46) and (47). Five points may be preferable to four, from the point of view of symmetry. In the special case of a uniform grid, three points suffice.

At this point, you should be able to see (“by induction”) that on a (possibly) non-uniform grid, an $n$th-order accurate approximation to the $l$th derivative of $f$ takes the form

$$f^l_j \equiv \sum_{k=-\infty}^{\infty} d_k f(x_{j+k}),$$

(52)

where

$$\sum_{k=-\infty}^{\infty} (x_{j+k} - x_j)^m d_k = (l!) \delta_{m,j} \text{ for } 0 \leq m \leq n+l-1.$$  

(53)

This is a total of $n+l$ requirements, so in general a minimum of $n+l$ points will be needed. It is straightforward to write a computer program that will automatically generate the coefficients for a compact $n$th-order-accurate approximation to the $l$th derivative of $f$.

What happens for $l = 0$?

### 2.3 Extension to two dimensions

The approach presented above can be generalized to multi-dimensional problems. We will illustrate this using the two-dimensional Laplacian operator. The Laplacian is used, for example, in the diffusion equation, which is

$$\frac{\partial f}{\partial t} = K \nabla^2 f,$$  

(54)

where $t$ is time and $K$ is a positive diffusion coefficient, assumed constant in the example above. A later chapter is devoted entirely to the diffusion equation.

Earlier in this chapter, the one-dimensional differences could represent either space or time differences, but our discussion of the Laplacian is unambiguously about space differencing.

Consider a fairly general finite-difference approximation to the Laplacian, of the form

$$\left(\nabla^2 f \right)_j \equiv \sum_{k=-\infty}^{\infty} e_{j,k} f(x_{j+k}, y_{j+k}).$$  

(55)
Here we use one-dimensional indices even though we are on a two-dimensional grid. The grid is not necessarily rectangular, and can be non-uniform. The subscript $j$ denotes a particular grid point (“home base” for this calculation), whose coordinates are $(x_j, y_j)$. Similarly, the subscript $j+k$ denotes a grid point in the neighborhood of point $j$, whose coordinates are $(x_{j+k}, y_{j+k})$. In practice, a method is needed to compute (and perhaps tabulate for later use) the appropriate values of $k$ for each $j$; for purposes of the present discussion this is an irrelevant detail.

The two-dimensional Taylor series is

$$f(x_{j+k}, y_{j+k}) = f(x_j, y_j) + \left[ (\delta x)_k f_x + (\delta y)_k f_y \right]$$

$$+ \frac{1}{2!} \left[ (\delta x)^2_k f_{xx} + 2(\delta x)_k (\delta y)_k f_{xy} + (\delta y)^2_k f_{yy} \right]$$

$$+ \frac{1}{3!} \left[ (\delta x)^3_k f_{xxx} + 3(\delta x)_k (\delta y)_k f_{xxy} + 3(\delta x)_k (\delta y)^2_k f_{xyy} + (\delta y)^3_k f_{yyy} \right]$$

$$+ \frac{1}{4!} \left[ (\delta x)^4_k f_{xxxx} + 4(\delta x)_k (\delta y)_k f_{xxyy} + 6(\delta x)_k (\delta y)^2_k f_{xxyy} + 4(\delta x)_k (\delta y)_k f_{xyyy} + (\delta y)^4_k f_{yyyy} \right] + \cdots$$

which can be written out in gruesome detail as

$$f(x_{j+k}, y_{j+k}) = f(x_j, y_j)$$

$$+ \left[ (\delta x)_k f_x + (\delta y)_k f_y \right]$$

$$+ \frac{1}{2!} \left[ (\delta x)_k^2 f_{xx} + 2(\delta x)_k (\delta y)_k f_{xy} + (\delta y)_k^2 f_{yy} \right]$$

$$+ \frac{1}{3!} \left[ (\delta x)_k^3 f_{xxx} + 3(\delta x)_k (\delta y)_k f_{xxy} + 3(\delta x)_k (\delta y)_k^2 f_{xxy} + (\delta y)_k^3 f_{yyy} \right]$$

$$+ \frac{1}{4!} \left[ (\delta x)_k^4 f_{xxxx} + 4(\delta x)_k (\delta y)_k f_{xxyy} + 6(\delta x)_k (\delta y)_k^2 f_{xxyy} + 4(\delta x)_k (\delta y)_k^3 f_{xyyy} + (\delta y)_k^4 f_{yyyy} \right] + \cdots$$

where

$$(\delta x)_k \equiv x_{j+k} - x_j \quad \text{and} \quad (\delta y)_k \equiv y_{j+k} - y_j,$$

and it is understood that all of the derivatives are evaluated at the point $(x_j, y_j)$. Note the “cross terms,” that involve products of $(\delta x)_k$ and $(\delta y)_k$, and the corresponding cross-derivatives. A more general form is

$$f(r + a) = \sum_{n=0}^{\infty} \frac{1}{n!} (a \cdot \nabla)^n f(r),$$

(58)
where \( \mathbf{r} \) is a position vector, and \( \mathbf{a} \) is a displacement vector (Arfken, 1985, p. 309). You should confirm for yourself that the general form (58) is consistent with (56). Eq. (58) does not make use of any particular coordinate system, but it can be used to work out the series expansion using any coordinate system, e.g., spherical coordinates or polar coordinates, by using the appropriate form of \( \nabla \).

Substituting from (56) into (55), we find that

\[
(f_{xx} + f_{yy})_j = \sum_{k=-\infty}^{\infty} e_{j,k} \left[ f \left( x_j, y_j \right) + \left( \delta x \right)_k f_x + \left( \delta y \right)_k f_y \right]
\]

\[
+ \frac{1}{2!} \left[ (\delta x)^2 f_{xx} + 2 (\delta x) (\delta y)_k f_{xy} + (\delta y)^2 f_{yy} \right]
\]

\[
+ \frac{1}{3!} \left[ (\delta x)^3 f_{xxx} + 3 (\delta x)^2 (\delta y)_k f_{xxy} + 3 (\delta x) (\delta y)^2 f_{xyy} + (\delta y)^3 f_{yyy} \right]
\]

\[
+ \frac{1}{4!} \left[ (\delta x)^4 f_{xxxx} + 4 (\delta x)^3 (\delta y)_k f_{xxyy} + 6 (\delta x)^2 (\delta y)^2 f_{xxyy} + 4 (\delta x) (\delta y)^3 f_{xxyy} + (\delta y)^4 f_{yyyy} \right] + \ldots \right] .
\]

(59)

Notice that we have expressed the Laplacian on the left-hand side of (59) in terms of Cartesian coordinates. The motivation is that we are going to use the special case of Cartesian coordinates \((x, y)\) as an example, and in the process we are going to to “match up terms” on the left and right sides of (59). Note, however, that the use of Cartesian coordinates in (59) does not limit its applicability to Cartesian grids. Eq. (59) can be used to analyze the truncation errors of a finite-difference Laplacian on any planar grid, regardless of how the grid points are distributed.

To have first-order accuracy, we need

\[
\sum_{k=-\infty}^{\infty} e_{j,k} = 0 , \text{ for all } j ,
\]

(60)

\[
\sum_{k=-\infty}^{\infty} e_{j,k} (\delta x)_k = 0 , \text{ for all } j ,
\]

(61)

\[
\sum_{k=-\infty}^{\infty} e_{j,k} (\delta y)_k = 0 , \text{ for all } j , \text{ and}
\]

(62)

An Introduction to Numerical Modeling of the Atmosphere
\[
\sum_{k=-\infty}^{\infty} e_{j,k} (\delta x)_k^2 = 2!, \text{ for all } j,
\]
\[
(63)
\]
\[
\sum_{k=-\infty}^{\infty} e_{j,k} (\delta x)_k (\delta y)_k = 0, \text{ for all } j,
\]
\[
(64)
\]
\[
\sum_{k=-\infty}^{\infty} e_{j,k} (\delta y)_k^2 = 2!, \text{ for all } j.
\]
\[
(65)
\]
From (55) and (59), it is clear that \( e_{j,k} \) is of order \( \delta^{-2} \), where \( \delta \) denotes \( \delta x \) or \( \delta y \). Therefore, the quantities inside the sums in (61-62) are of order \( \delta^{-1} \), and those inside the sums in (63-65) are of order one. This is why (63-65) are required, in addition to (60-62), to obtain first-order accuracy.

Eq. (60) implies [with (55)] that a constant field has a Laplacian of zero, as it should. That’s nice.

So far, (60-65) involve six equations. This means that to ensure first-order accuracy for a general grid, six grid points are needed for the general case. There are exceptions to this. If we are fortunate enough to be working on a highly symmetrical grid, it is possible that the conditions for second-order accuracy can be satisfied with just six points, or even fewer than that. For example, if we satisfy (60-65) on a square grid, we will get second-order accuracy “for free,” and, as you will show when you do the homework, it can be done with only five points. More generally, with a non-uniform grid, we need the following four additional conditions to achieve second-order accuracy:

\[
\sum_{k=-\infty}^{\infty} e_{j,k} (\delta x)_k^3 = 0, \text{ for all } j,
\]
\[
(66)
\]
\[
\sum_{k=-\infty}^{\infty} e_{j,k} (\delta x)_k^2 (\delta y)_k = 0, \text{ for all } j,
\]
\[
(67)
\]
\[
\sum_{k=-\infty}^{\infty} e_{j,k} (\delta x)_k (\delta y)_k^2 = 0, \text{ for all } j,
\]
\[
(68)
\]
\[ \sum_{k=-\infty}^{\infty} e_{j,k} (\delta y)^3_k = 0, \text{ for all } j. \]  

(69)

In general, a total of ten conditions, i.e., Eqs. (60-69), must be satisfied to ensure second-order accuracy on a non-uniform two-dimensional grid.

If we were working in more than two dimensions, we would simply replace (56) by the appropriate multi-dimensional Taylor series expansion. The rest of the argument would be parallel to that given above, although of course the requirements for second-order accuracy would be more numerous.

For the continuous Laplacian on a closed or periodic domain, we can prove the following two important properties:

\[ \int_{A} (\nabla^2 f) dA = 0, \]  

(70)

\[ \int_{A} f (\nabla^2 f) dA \leq 0. \]  

(71)

Here the integrals are with respect to area, over the entire domain. These results hold for any sufficiently differentiable function \( f \). For the diffusion equation, (54), Eq. (70) implies that diffusion does not change the area-averaged value of \( f \), and the inequality (71) implies that diffusion reduces the area average of the square of \( f \).

The finite-difference requirements corresponding to (70) and (71) are

\[ \sum_{\text{all } j} (\nabla^2 f) A_j \equiv \sum_{\text{all } j} \left[ \sum_{k=-\infty}^{\infty} e_{j,k} f(x_{j+k}, y_{j+k}) \right] A_j = 0, \]  

(72)

\[ \sum_{\text{all } j} f_j (\nabla^2 f) A_j \equiv \sum_{\text{all } j} f_j \left[ \sum_{k=-\infty}^{\infty} e_{j,k} f(x_{j+k}, y_{j+k}) \right] A_j \leq 0, \]  

(73)

where \( A_j \) is the area of grid-cell \( j \). Suppose that we want to satisfy (72-73) regardless of the numerical values assigned to \( f(x_j, y_j) \). This may sound impossible, but it can be achieved by suitable choice of the \( e_{j,k} \). As an example, consider what is needed to ensure that (72) will be satisfied for an arbitrary distribution of \( f(x_j, y_j) \). Each value of \( f(x_j, y_j) \) will appear more than
once in the sum on the right-hand side of (72). We can “collect the coefficients” of each value of \( f(x_j, y_j) \), and require the sum of the coefficients to be zero. To see how this works, define \( j' = j + k \), and write

\[
\sum_{j, k = -\infty}^{\infty} e_{j, k} f(x_{j+k}, y_{j+k}) A_j = \sum_{j, j' = -\infty}^{\infty} e_{j, j'-j} f(x_{j'}, y_{j'}) A_j
\]

\[
= \sum_{j, j' = -\infty}^{\infty} e_{j, j'-j} f(x_{j'}, y_{j'}) A_j
\]

\[
= \sum_{j, j' = -\infty}^{\infty} f(x_{j'}, y_{j'}) \left( \sum_{j} e_{j, j'-j} A_j \right)
\]

\[
= 0 .
\]  

(74)

In the second line above, we simply change the order of summation. The last equality above is a re-statement of the requirement (72). The only way to satisfy it for an arbitrary distribution of \( f(x_j, y_j) \) is to write

\[
\sum_{j} e_{j, j'-j} A_j = 0 \text{ for each } j',
\]

(75)

which is equivalent to

\[
\sum_{j} e_{j, k} A_j = 0 \text{ for each } k .
\]

(76)

If the \( e_{j, k} \) satisfy (76), then (72) will also be satisfied.

Similar (but more complicated) ideas were used by Arakawa (1966), in the context of energy and enstrophy conservation with a finite-difference vorticity equation. This will be discussed in Chapter 11.

2.4 Laplacians on rectangular grids

Consider a nine-points on a rectangular grid, as shown in Fig. 2.3. Because this grid has a high degree of symmetry, is possible to obtain second-order accuracy with just five points, and in fact this can be done in two different ways, corresponding to the two five-point stencils shown
Based on their shapes, one of the stencils can be called “+”, and the other one “x”. We assume a grid spacing $d$ in both the $x$ and $y$ directions, and use a two-dimensional indexing system, with counters $i$ and $j$ in the $x$ and $y$ directions, respectively. Using the methods explained above, it can be shown that the second-order finite-difference Laplacians are given by

$$\nabla^2 f(i,j) \equiv \frac{f_{i+1,j} + f_{i-1,j} + f_{i,j+1} + f_{i,j-1} - 4f_{i,j}}{d^2} \text{ with the + stencil},$$

(77)

and

$$\nabla^2 f(i,j) \equiv \frac{f_{i+1,j+1} + f_{i-1,j+1} + f_{i+1,j-1} + f_{i-1,j-1} - 4f_{i,j}}{(\sqrt{2}d)^2} \text{ with the x-stencil}.$$  

(78)

The Laplacian based on the x stencil cannot “see” a checkerboard pattern in the function represented on the grid, as shown by the plus and minus symbols in the top two panels. What I mean by this is that the scheme returns a zero for the Laplacian even when a checkerboard is present. In contrast, the + stencil feels the checkerboard. The checkerboard solution can be called a “computational mode” because it is a solution of a finite-difference approximation to $\nabla^2 f = 0$.

Figure 2.3: The grey shading shows two five-point stencils that can be used to create second-order Laplacians on a rectangular grid. In the upper two panels, the plus and minus symbols represent an input function that has the form of a checkerboard. In the lower two panels the plus and minus symbols represent an input function that has the form of a “rotated checkerboard,” which can also be viewed as a set of stripes. (The checkerboard in the top two panels can be viewed as “rotated stripes.”)
but it does not correspond to any true solution of the differential equation. Computational modes are bad. This is our first encounter with them, but they will be discussed in much more detail later.

The + stencil is not necessarily the clear winner here. It under-estimates the strength of the “rotated checkerboard” (which can also be called “stripes”) shown in the bottom two panels of Fig. 2.3, while the x stencil feels it more strongly. The best second-order Laplacian uses all nine points, and can be obtained by writing a weighted sum of the two Laplacians given by (77) and 78). What principle would you suggest for choosing the values of the weights? The nine-point stencil will involve more arithmetic than either of the five-point stencils, but the benefit may justify the cost in this case.

2.5 Why be square?

As is well known, only three regular polygons tile the plane: equilateral triangles, squares, and hexagons. Fig. 2.4 shows planar grids made from each of these three possible polygonal elements.

On the triangular grid and the square grid, some of the neighbors of a given cell lie directly across cell walls, while others lie across cell vertices. As a result, finite-difference operators constructed on these grids tend to use “wall neighbors” and “vertex neighbors” in different ways. For example, the simplest second-order finite-difference approximation to the gradient, on a square grid, uses only “wall neighbors;” vertex neighbors are ignored. Although it is certainly possible to construct finite-difference operators on square grids (and triangular grids) in which information from all neighboring cells is used [e.g., the Arakawa Jacobian, as discussed by Arakawa (1966) and in Chapter 11], the essential anisotropies of these grids remain, and are unavoidably manifested in the forms of the finite-difference operators.

In contrast, hexagonal grids have the property that all neighbors of a given cell lie across cell walls; there are no “vertex neighbors.” In this sense, hexagonal grids are quasi-isotropic. As a result, the most natural finite-difference Laplacians on hexagonal grids treat all neighboring cells in the same way; they are as symmetrical and isotropic as possible.

Further discussion of these three polygonal grids will be given later in the course.
2.12 Summary

It is straightforward to design finite-difference schemes to approximate a derivative of any order, with any desired order of accuracy, on irregular grids of any shape, and in multiple dimensions. The schemes can also be designed to satisfy rules based on properties of the differential operators that they approximate; for example, we can force the area-average of a two-dimensional finite-difference Laplacian to vanish on a periodic domain. Some finite-difference schemes suffer from an inability to recognize small-scale, “noisy” computational modes on the grid, such as checkerboard patterns.
Problems

1. Prove that a finite-difference scheme with errors of order \( n \) gives exact derivatives for polynomial functions of degree \( n \) or less. For example, a first-order scheme gives exact derivatives for linear functions.

2. Choose a simple differentiable function \( f(x) \) that is not a polynomial. Find the exact numerical value of \( \frac{df}{dx} \) at a particular value of \( x \), say \( x_1 \). Then choose
   a) a first-order scheme, and
   b) a second-order scheme to approximate \( \frac{df}{dx} \) at \( x_1 \). Plot the log of the absolute value of the error of these approximations as a function of the log of \( \Delta x \). By inspection of the plot, verify that the errors of the schemes decrease, with the expected slopes (which you should estimate from the plots), as \( \Delta x \) decreases.

3. Repeat the analysis of Section 2.3, up to Eq. (64), using cylindrical (polar) coordinates.

4. Prove that the only regular convex polygons that tile the plane are the equilateral triangle, the square, and the equilateral hexagon.

5. For this problem, use equations (60)-(69), which state the requirements for second-order accuracy of the Laplacian.
   a) Consider a plane tiled with perfectly hexagonal cells. The dependent variable is defined at the center of each hexagon. Find a second-order accurate scheme for the Laplacian that uses just the central cell and its six closest neighbors, i.e., just seven cells in total. Make a sketch to show explain your notation for distances in the \( x \) and \( y \) directions. *Hint:* You can drastically simplify the problem by taking advantage of the high degree of symmetry.
   b) Repeat for a plane tiled with equilateral triangles. You will have to figure out which and how many cells to use, in addition to the central cell. Use the centroids of the triangles to assign their positions. Make a sketch to explain your notation for distances in the \( x \) and \( y \) directions. *Hint:* You can drastically simplify the problem by taking advantage of the high degree of symmetry.
   c) Repeat for a plane tiled with squares, using the central cell and the neighboring cells across the cell walls. Make a sketch to show explain your notation for distances in the \( x \) and \( y \) directions. *Hint:* You can drastically simplify the problem by taking advantage of the high degree of symmetry.
d) Repeat for a plane tiled with squares, using the central cell and the neighboring cells that lie diagonally across cell corners. Make a sketch to show explain your notation for distances in the $x$ and $y$ directions. *Hint:* You can drastically simplify the problem by taking advantage of the high degree of symmetry.

e) For each of the four cases discussed above, show that a constant input field gives a Laplacian of zero everywhere.

f) For each of the four cases discussed above, and *assuming a doubly periodic domain on a plane*, discuss whether it is possible to have Laplacian = 0 throughout the entire domain even when the input field is not constant. Fig. 2.5 can help you to understand how to define the computational domain and how to implement the periodic boundary conditions.
6. a) Invent a way to “index” the points on a hexagonal grid with periodic boundary conditions. As a starting point, I suggest that you make an integer array like this: NEIGHBORS(I, J). The first subscript would designate which point on the grid is “home base,” i.e., it would be a one-dimensional counter that would cover the entire grid. The second subscript would range from 0 to 6 (or from 1 to 7). The smallest value (0 or 1) would designate the central point, and the remaining 6 would designate its six neighbors. The indexing problem then reduces to generating the array NEIGHBORS(I, J), which need only be done once for a given grid.

Figure 2.5: Sketch showing four copies of a periodic, nearly square domain on a (low-resolution) hexagonal grid. All four corners of each domain are located in the centers of hexagonal cells. The vertical edges of each domain run through vertical stacks of cells. Obviously the picture could be rotated by 90° (or any other angle) without changing the idea.
b) Set up a hexagonal grid to represent a square domain with periodic boundary conditions. It is not possible to create an exactly square domain using a hexagonal grid. Therefore you have to set up a domain that is approximately square, with 100 points in one direction and the appropriate number of points (you get to figure it out) in the other direction. The total number of points in the approximately square domain will be very roughly 8000. This (approximately) square domain has periodic boundary conditions, so it actually represents one “patch” of an infinite domain. The period in the \( x \)-direction cannot be exactly the same as the period in the \( y \)-direction, but it can be pretty close. Make sure that the boundary conditions are truly periodic on your grid, so that no discontinuities occur.

c) Using the tools created under parts a) and b) above, write a program to compute the Laplacian of a given doubly periodic function.

d) Empirically demonstrate the order of accuracy of your program using an analytical test function, for which you can compute the Laplacian exactly. To do this, you have to invent a measure of the overall error; justify your definition.

7. Consider a scheme for the Laplacian on a square grid that is given by a weighted sum of (77) and (78). Propose a basis for choosing the weight. Explain the motivation for your proposal. There is more than one good answer to this question.

8. The Laplacian can be written using Cartesian coordinates \((x,y)\) as \(\nabla^2 f = f_{xx} + f_{yy} \). Consider a second Cartesian coordinate system, \((x',y')\), that is rotated, with respect to the first, through an angle \(\theta\). Prove by direct calculation that \(f_{xx} + f_{yy} = f_{x'x'} + f_{y'y'}\). This demonstrates that the Laplacian of a function is invariant with respect to rotations of the function.