Chapter 6

Differentiation

The analysis of scientific or engineering data often requires the calculation of the first (or higher) derivative of a function or of a curve defined by a table of data points. These derivative values may be needed to solve problems involving the slope of a curve, the velocity or acceleration of an object, or for other calculations.

Students in calculus courses learn mathematical expressions for the derivatives of many types of functions. But there are many other functions for which it is difficult to obtain an expression for the derivative, or indeed the function may not be differentiable. Fortunately, the derivative can always be obtained by numerical methods, which can be implemented easily on a spreadsheet. This chapter provides methods for calculation of derivatives of worksheet formulas or of tabular data.

First and Second Derivatives of Data in a Table

The simplest method to obtain the first derivative of a function represented by a table of x, y data points is to calculate Δx and Δy, the differences between adjacent data points, and use $\Delta y/\Delta x$ as an approximation to dy/dx. The first derivative or slope of the curve at a given data point x_i, y_i can be calculated using either of the following forward, backward, or central difference formulas, respectively (equations 6-1, 6-2, and 6-3).

$$\frac{dy}{dx} \approx \frac{\Delta y}{\Delta x} = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} \qquad \text{(forward difference)} \qquad (6\text{-}1)$$

$$\frac{dy}{dx} \approx \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \qquad \text{(backward difference)} \qquad (6\text{-}2)$$

$$\frac{dy}{dx} \approx \frac{y_{i+1} - y_{i-1}}{x_{i+1} - x_{i-1}} \qquad \text{(central difference)} \qquad (6\text{-}3)$$

The second derivative, d^2y/dx^2, of a data set can be calculated in a similar manner, namely by calculating $\Delta(\Delta y/\Delta x)/\Delta x$.

Calculation of the first or second derivative of a data set tends to emphasize the "noise" in the data set; that is, small errors in the measurements become relatively much more important. The central difference formula tends to reduce noise resulting from experimental error.

Points on a curve of x, y values for which the first derivative is a maximum, a minimum, or zero are often of particular importance and are termed *critical points*, that is, points where the curvature (the second derivative) changes sign are termed *inflection points*. For example, in the analysis of data from an acid-base titration, the inflection point is used to determine the equivalence point.

Calculating First and Second Derivatives

A pH titration (measured volumes of a base solution are added to a solution of an acid and the pH measured after each addition) is shown in Figure 6-1, and a portion of the spreadsheet containing the titration data in Figure 6-2. The end-point of the titration corresponds to the point on the curve with maximum slope, and this point can be estimated visually in Figure 6-1. The first and second derivatives of the data are commonly used to determine the inflection point of the curve mathematically.

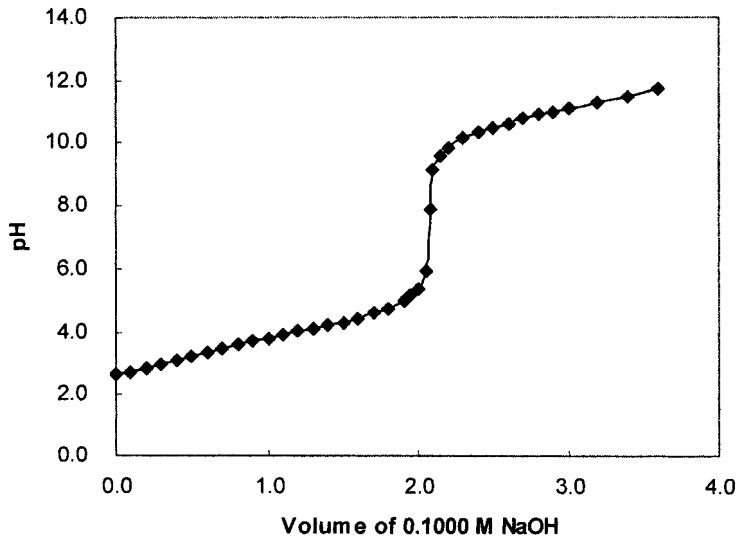

Figure 6-1. Chart of titration data.
(folder 'Chapter 06 Examples', workbook 'Derivs of Titration Data', worksheet 'Derivs')

	A	B	C	D	E	F
2	V/mL	pH	ΔV	ΔpH	V(avge)	ΔpH/ΔV
22	1.90	4.981	0.100	0.229	1.850	2.29
23	1.95	5.157	0.050	0.176	1.925	3.52
24	2.00	5.389	0.050	0.232	1.975	4.64
25	2.05	5.928	0.050	0.539	2.025	10.78
26	2.08	7.900	0.030	1.972	2.065	65.73
27	2.10	9.115	0.020	1.215	2.090	60.75
28	2.15	9.604	0.050	0.489	2.125	9.78
29	2.20	9.856	0.050	0.252	2.175	5.04
30	2.30	10.125	0.100	0.269	2.250	2.69

Figure 6-2. First derivative of titration data, near the endpoint.
(folder 'Chapter 06 Examples', workbook 'Derivs of Titration Data', worksheet 'Derivs')

Columns A through F of the spreadsheet shown in Figure 6-2 are used to calculate the first derivative, $\Delta pH/\Delta V$. Since the derivative has been calculated over the finite volume $\Delta V = V_{i+1} - V_i$, the most suitable volume to use when plotting the $\Delta pH/\Delta V$ values, as shown in column E of Figure 6-2, is

$$V_{average} = \frac{V_{i+1} + V_i}{2} \tag{6-4}$$

The maximum in ΔpH/ΔV indicates the location of the inflection point of the titration (Figure 6-3).

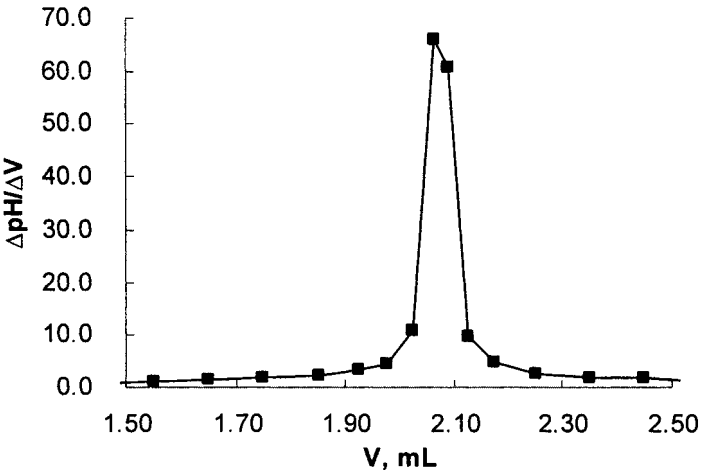

Figure 6-3. First derivative of titration data, near the endpoint.
(folder 'Chapter 06 Examples', workbook 'Derivs of Titration Data', worksheet 'Derivs')

The maximum in the first derivative curve must still be estimated visually. The second derivative, $\Delta(\Delta pH/\Delta V)/\Delta V$, calculated by means of columns E through J of the spreadsheet (shown in Figure 6-4) can be used to locate the inflection point more precisely. The second derivative, shown in Figure 6-5, passes through zero at the inflection point. Linear interpolation can be used to calculate the point at which the second derivative is zero.

	E	F	G	H	I	J
2	V(avge)	ΔpH/ΔV	ΔV	Δ(ΔpH)	V(avge)	Δ(ΔpH)/ΔV
22	1.850	2.29	0.100	0.57	1.800	5.7
23	1.925	3.52	0.075	1.23	1.888	16.4
24	1.975	4.64	0.050	1.12	1.950	22.4
25	2.025	10.78	0.050	6.14	2.000	122.8
26	2.065	65.73	0.040	54.95	2.045	1373.8
27	2.090	60.75	0.025	-4.98	2.078	-199.3
28	2.125	9.78	0.035	-50.97	2.108	-1456.3
29	2.175	5.04	0.050	-4.74	2.150	-94.8
30	2.250	2.69	0.075	-2.35	2.213	-31.3

Figure 6-4. Second derivative of titration data, near the endpoint.
(folder 'Chapter 06 Examples', workbook 'Derivs of Titration Data', worksheet 'Derivs')

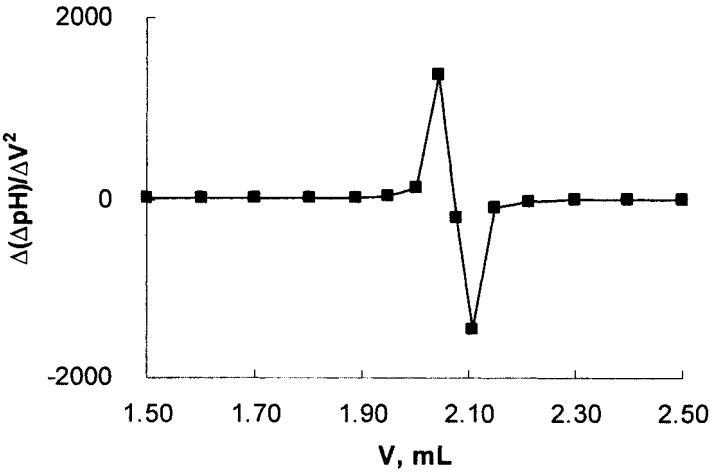

Figure 6-5. Second derivative of titration data, near the endpoint.
(folder 'Chapter 06 Examples', workbook 'Derivs of Titration Data', worksheet 'Derivs')

There are other equations for numerical differentiation that use three or more points instead of two points to calculate the derivative. Since these equations usually require equal intervals between points, they are of less generality. Again, their main advantage is that they minimize the effect of "noise." Table 6-1 lists equations for the first, second and third derivatives, for data from a table at equally spaced interval *h*.

These difference formulas can be derived from Taylor series. Recall from Chapter 4 that the first-order approximation is

$$F(x+h) \approx F(x) + hF'(x) \tag{6-5}$$

or, in the notation used in Table 6-1

$$y_{i+1} = y_i + hy'_i \tag{6-6}$$

which, upon rearranging, becomes

$$y'_i = \frac{y_{i+1} - y_i}{h} \tag{6-7}$$

admittedly, an obvious result.

The second derivative can be written as

$$y''_i = \frac{y'_{i+1} - y'_i}{h} \tag{6-8}$$

When each of the y' terms is expanded according to the preceding expression for y', the expression for the second derivative becomes

$$y''_i = \frac{(y_{i+2} - y_{i+1})/h - (y_{i+1} - y_i)/h}{h} \tag{6-9}$$

or

$$y''_i = \frac{y_{i+2} - 2y_{i+1} + y_i}{h^2} \tag{6-10}$$

The same result can be obtained from the second-order Taylor series expansion

$$F(x+h) \approx F(x) + hF'(x) + \frac{h^2}{2!}F''(x) \tag{6-11}$$

which is written in Table 6-1 as

$$y_{i+1} = y_i + hy'_i + \frac{h^2}{2!}y''_i \tag{6-12}$$

by substituting the backward-difference formula for F' from Table 6-1. Expressions for higher derivatives or for derivatives using more terms can be obtained in a similar fashion.

Table 6-1. Some Formulas for Computing Derivatives
(For tables with equally spaced entries)

First derivative, using two points:

Forward difference $$y_i' = \frac{y_{i+1} - y_i}{h}$$

Central difference $$y_i' = \frac{y_{i+1} - y_{i-1}}{2h}$$

Backward difference $$y_i' = \frac{y_i - y_{i-1}}{h}$$

First derivative, using three points:

Forward difference $$y_i' = \frac{-y_{i+2} + 4y_{i+1} - 3y_i}{2h}$$

First derivative, using four points:

Central difference $$y_i' = \frac{-y_{i+2} + 8y_{i+1} - 8y_{i-1} + y_{i-2}}{12h}$$

Second derivative, using three points:

Forward difference $$y_i'' = \frac{y_{i+2} - 2y_{i+1} + y_i}{h^2}$$

Central difference $$y_i'' = \frac{y_{i+1} - 2y_i + y_{i-1}}{h^2}$$

Backward difference $$y_i'' = \frac{y_i - 2y_{i-1} + y_{i-2}}{h^2}$$

Second derivative, using four points:

Forward difference $$y_i'' = \frac{2y_i - 5y_{i+1} + 4y_{i+2} - y_{i+3}}{h^2}$$

Second derivative, using five points:

Central difference $$y_i'' = \frac{-y_{i+2} + 16y_{i+1} - 30y_i + 16y_{i-1} - y_{i-2}}{12h^2}$$

Third derivative, using four points

Forward difference $$y_i''' = \frac{y_{i+3} - 3y_{i+2} + 3y_{i+1} - y_i}{h^3}$$

Using LINEST as a Fitting Function

Instead of calculating a derivative at an x value corresponding to a table entry, it may be necessary to obtain the derivative at an intermediate x value. This problem is related to the process of interpolation, and indeed some of the techniques from the preceding chapter can be applied here (see "Cubic Interpolation" in Chapter 5). For example, we can obtain a piecewise fitting function that applies to a localized region of the data set, and use the parameters of the fitting function to calculate the derivative. In this section and the following one, we will use a cubic equation

$$F(x) = ax^3 + bx^2 + cx + d \tag{6-13}$$

as the fitting function, using four data points to obtain the four coefficients of the cubic. (The fitted curve will pass exactly through all four points and R^2 will be exactly 1.) Once we have obtained the coefficients, the derivatives are calculated from them; the first derivative is

$$F'(x) = 3ax^2 + 2bx + c \tag{6-14}$$

and the second derivative is

$$F''(x) = 6ax + 2b \tag{6-15}$$

We can use the LINEST worksheet function (the multiple linear regression worksheet function, described in detail in Chapter 13) to obtain the coefficients a, b, c and d, then use the coefficients a, b, and c in equation 6-14 or 6-15 to calculate the first or second derivatives.

The LINEST method will be illustrated using a table of absorbance data taken at 5-nm increments, part of which is shown in Figures 6-6 and 6-7; the complete range of x values is in $A$5:$A$85 and the y values in $B$5:$B$85. We wish to obtain the first derivative of this data set at 2-nm increments over the range 390–415 nm.

	A	B
3	Original Data	
4	Wavelength	Absorbance
23	390	0.552
24	395	0.582
25	400	0.598
26	405	0.600
27	410	0.586
28	415	0.559
29	420	0.521

Figure 6-6. Data used to calculate first and second derivatives.
(folder 'Chapter 06 Examples', workbook 'Derivs Using LINEST', sheet 'Using megaformula')

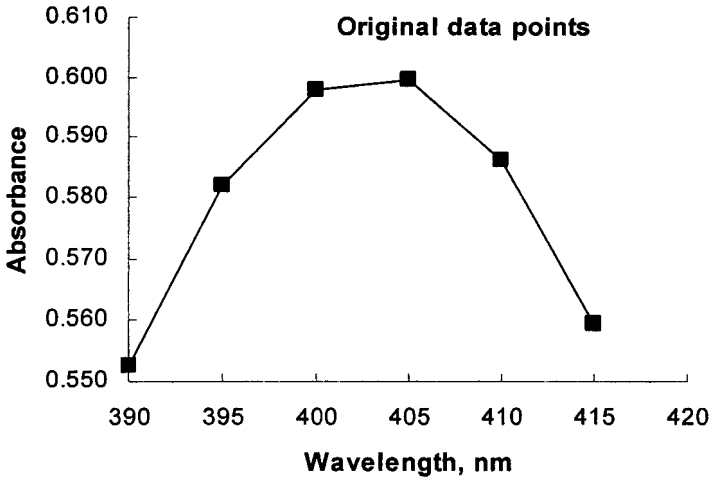

Figure 6-7. Chart of some data used to calculate first and second derivatives.
(folder 'Chapter 06 Examples', workbook 'Derivs Using LINEST', sheet 'Using megaformula')

The steps required in the calculation of the first or second derivative at a specified value of x are as follows:

(i) Use the MATCH function to find the position of the lookup value x in the table of x values. The lookup value is in cell D5 in Figure 6-8.

```
=MATCH(D5, $A$5:$A$85,1)
```

(ii) Use the OFFSET function to select the four bracketing x values:

```
=OFFSET($A$5:$A$85,D5-2,0,4,1)
```

(iii) Use a similar formula to obtain the four bracketing y values:

```
=OFFSET($B$5:$B$85,D5-2,0,4,1)
```

(iv) Use these two arrays in the LINEST formula, raising the range of x values to an array of powers; the LINEST formula must be entered in a horizontal range of three cells, and you must press CONTROL+SHIFT+ENTER:

```
=LINEST(OFFSET(known_ys,MATCH(D6,known_xs,1)-2,0,4,1),
OFFSET(known_xs,MATCH(D6,known_xs,1)-2,0,4,1)^{1,2,3},1,0)
```

(v) Use the INDEX function to obtain each of the regression coefficients a, b and c from the LINEST array. (To simplify the formula, the cells containing the preceding LINEST formula have been given the name LINEST_array.) The following equation returns the coefficient a:

```
=INDEX(LINEST_array,1)
```

(vi) Use the coefficients *a*, *b*, and *c* to calculate the first or second derivative:

If these formulas are combined into one "megaformula", the result (entered in cell E5 in Figure 6-8) is

```
=3*INDEX(LINEST(OFFSET(known_ys,MATCH(D5,x_values,1)-2,0,4,1),
OFFSET(x_values,MATCH(D5,x_values,1)-2,0,4,1)^{1,2,3},1,0),1)*x^2
+2*INDEX(LINEST(OFFSET(known_ys,MATCH(D5,x_values,1)-2,0,4,1),
OFFSET(x_values,MATCH(D5,x_values,1)-2,0,4,1)^{1,2,3},1,0),2)*x
+INDEX(LINEST(OFFSET(known_ys,MATCH(D5,x_values,1)-2,0,4,1),
OFFSET(x_values,MATCH(D5,x_values,1)-2,0,4,1)^{1,2,3},1,0),3)
```

which is rather confusing. A better approach is to use named formulas. The following table lists the named formulas and ranges used to calculate the first derivative shown in Figure 6-7.

```
x_values     =Sheet2!$A$5:$A$85
y_values     =Sheet2!$B$5:$B$85
lookup_value =Sheet2!$D$5:$D$17
pointer    =MATCH(INDIRECT(ROW()&":"&ROW()) lookup_value ,x_values,1)
known_xs     =OFFSET(x_values,pointer-2,0,4,1)
known_ys     =OFFSET(y_values,pointer-2,0,4,1)
LIN_array    =LINEST(Sheet2!known_ys,Sheet2!known_xs^{1,2,3},1,0)
aa           =INDEX(LINEST_array,1)
bb           =INDEX(LINEST_array,2)
cc           =INDEX(LINEST_array,3)
```

Using these named formulas, the formula for the first derivative becomes

```
=3*aa*x^2+2*bb*x+cc
```

Note the formula used for pointer. It incorporates an "implicit intersection" expression. Since both lookup_value and x_values are arrays, the formula

```
=MATCH(lookup_value ,x_values,1)
```

returns an array of values instead of a single value. The formula using the expression INDIRECT(ROW()&":"&ROW()) lookup_value returns a single value, the value in the array lookup_value that is in the same row as the formula.

	D	E	F	G
4	x	y	F'(x)	F"(x) x 10
5	390	0.552	0.00710	-4.53E-03
6	392		0.00616	-4.87E-03
7	394		0.00516	-5.20E-03
8	396		0.00405	-5.46E-03
9	398		0.00294	-5.65E-03
10	400	0.598	0.00176	-5.84E-03
11	402		0.00059	-5.85E-03
12	404		-0.00058	-5.87E-03
13	406		-0.00179	-5.80E-03
14	408		-0.00293	-5.65E-03
15	410	0.586	-0.00408	-5.49E-03
16	412		-0.00515	-5.18E-03
17	414		-0.00615	-4.88E-03

Figure 6-8. First derivative calculated using LINEST function.
The *y* values indicate the known experimental points.
(folder 'Chapter 06 Examples', workbook 'Derivs Using LINEST', sheet 'Using named formulas')

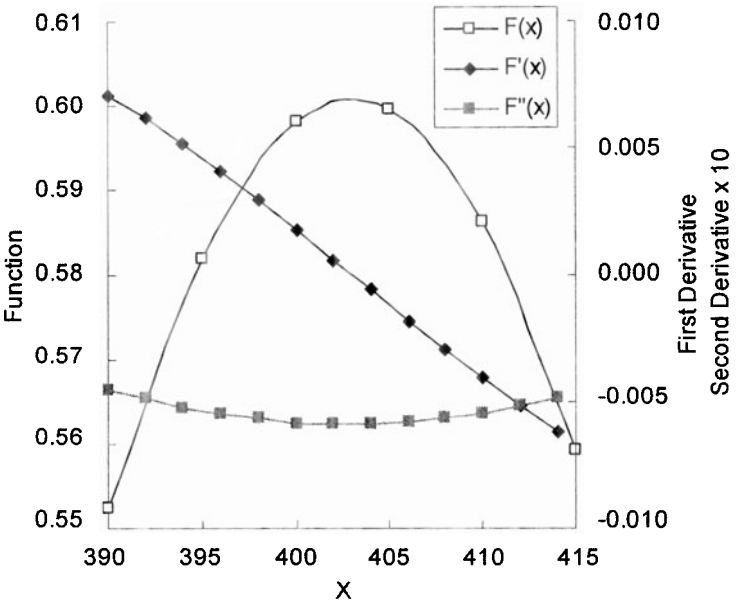

Figure 6-9. Chart of values of first and second derivative calculated using LINEST.
(folder 'Chapter 06 Examples', workbook 'Derivs Using LINEST', sheet 'Using named formulas')

Part of the table of calculated first derivative values is shown in Figure 6-8, and the values are charted in Figure 6-9. The formula used in cell F5, for example, is

```
=3*aa*x^2+2*bb*x+cc
```

One could use the x value where $F'(x) = 0$ to locate the maximum in the spectrum.

Depending on the data table being differentiated, the errors in the values returned by this method may be as great as several percent.

Derivatives of a Worksheet Formula

Instead of calculating the first or second derivative of a curve represented by data points, we may wish to find the derivative of a function (a worksheet formula). In the following, two different methods are illustrated to calculate the first or second derivative of a worksheet formula by using a user-defined function. The calculation of the first derivative of the function $y = 3x^3 + 5x^2 - 5x + 11$ is used as the example for each method

Derivatives of a Worksheet Formula Calculated by Using a VBA Function Procedure

The first example is a **Function** procedure that returns the first derivative of a specific worksheet formula. The expression for the derivative is "hard-coded" in the VBA procedure. The user must be able to provide the expression for the derivative and must modify the VBA code to apply it to a different formula. The function's only argument is the value of x, the independent variable for which the derivative is to be calculated. The main advantage of this approach is that the returned value of the derivative is exact. This approach will execute the fastest and would be suitable if the same formula is to be used many times in a worksheet.

```
Function Deriv1(x)
'User codes the expression for the derivative here.
Deriv1 = 9 * x ^ 2 + 10 * x - 5
End Function
```

Figure 6-10. Function procedure to demonstrate calculation of a first derivative. (folder 'Chapter 06 Examples', workbook 'Derivs by VBA (Part 1)', module 'Module1')

First Derivative of a Worksheet Formula Calculated by Using the Finite-Difference Method

The second example is a **Function** procedure that uses the finite-difference method. The first derivative of a formula in a worksheet cell can be obtained with a high degree of accuracy by evaluating the formula at x and at $x + \Delta x$. Since Excel carries 15 significant figures, Δx can be made very small. Under these conditions $\Delta y/\Delta x$ approximates dy/dx very well.

The user must "hard-code" the worksheet formula in VBA, in a suitable form; the derivative is calculated by numerical differentiation. Again, the function's only argument is the value of x, the independent variable. This approach would be useful if the user is unable to provide an expression for the derivative.

```
Function Deriv2(x)
OldY = fn(x)
xx = (1.00000001) * x
NewY = fn(xx)
Deriv2 = (NewY - OldY) / (xx - x)
End Function

Function fn(x)
'User codes the expression for the function here.
fn = 3 * x ^ 3 + 5 * x ^ 2 - 5 * x + 11
End Function
```

Figure 6-11. Function procedure to demonstrate calculation of first derivative. (folder 'Chapter 06 Examples', workbook 'Derivs by VBA (Part 1)', module 'Module1')

The Newton Quotient

In the previous section, the finite-difference method was shown to provide an excellent estimate of the first derivative of a function expressed as a worksheet formula. The multiplier used in the preceding user-defined function was 1.00000001. What is the optimum value of this multiplier, so that the Newton quotient $\Delta y/\Delta x$ gives the best approximation to dy/dx?

There are two sources of error in this finite-difference method of computing dy/dx: the *approximation error*, inherent in using a finite value of Δx, and the *roundoff error*, due to the limited precision of the numbers stored in the computer. We want to find the value of Δx that strikes the best balance between these two errors. If Δx is made too large, then the approximation error is large, since $dy/dx \rightarrow \Delta y/\Delta x$ only when $\Delta x \rightarrow 0$. If Δx is made too small, then the roundoff error is large, since we are obtaining Δy by subtracting two large and nearly equal numbers, $F(x)$ and $F(x + \Delta x)$.

Excel carries 15 digits in its calculations, and it turns out that multiplying x by a factor of 1.00000001 (a change in the 8th place) produces the minimum error, before round-off error begins to have an effect. Figure 6-12 illustrates this, using a quadratic equation as an example; other functions give similar results. The values in Figure 6-12 show that we can expect accuracy up to approximately the tenth digit.

	A	B	C	D	E	F	G	H
1				$y = 3x^2 - 11x + 30$				
2	x	y	Δx	x+Δx	y+Δy	Δy/Δx	exact	% error
3	7.5	116.25	1.0E-05	7.5001	116.253	34.0002	34	6.6E-04
4	7.5	116.25	1.0E-06	7.50001	116.2503	34.00002	34	6.6E-05
5	7.5	116.25	1.0E-07	7.500001	116.25003	34.000002	34	6.7E-06
6	7.5	116.25	1.0E-08	7.5000001	116.250003	34.0000003	34	8.4E-07
7	7.5	116.25	1.0E-09	7.50000001	116.2500003	34.000001	34	4.2E-06
8	7.5	116.25	1.0E-10	7.500000001	116.25000003	34.00002	34	4.9E-05
9	7.5	116.25	1.0E-11	7.5000000001	116.250000003	34.0001	34	4.2E-04

Figure 6-12. Newton quotient $\Delta y/\Delta x$ as a function of the magnitude of Δx
(folder 'Chapter 06 Examples', workbook 'Derivs by VBA (Part 1)', sheet 'Newton Quotient')

Derivative of a Worksheet Formula Calculated by Using the Finite-Difference Method

The spreadsheet shown in Figure 6-13 (see folder 'Chapter 06 Examples', workbook 'Derivs by Sub Procedure') illustrates the calculation of the first derivative of a function $y = x^3 - 3x^2 - 130x + 150$ by evaluating the function at x and at $x + \Delta x$. Here a value of Δx of 1×10^{-8} was used. For comparison, the first derivative was calculated from the exact expression from differential calculus: $F'(x) = 3x^2 - 6x - 130$.

The Excel formulas in cells B11, C11, D11, E11, F11, G11 and H11 (columns C–F are hidden) are

B11	= t*x^3+u*x^2+v*x + w	$F(x)$
C11	=A11*(1+delta)	$x + \Delta x$
D11	= t*C11^3+u*C11^2+v*C11 + w	$F(x + \Delta x)$
E11	=A11*delta	Δx
F11	=D11-B11	Δy
G11	=F11/E11	$\Delta y/\Delta x$
H11	=3*t*A11^2+2*u*A11+v	dy/dx from calculus

	A	B	G	H
1	Numerical Differentiation			
2	F(x) = tx^3 + ux^2 + vx + w			
3		t	1	
4		u	-3	delta
5		v	-130	1.00E-08
6		w	150	
7			Δy/Δx	
8	x	y	By worksheet formula	From calculus
9	-10	150	230.0000006	230
10	-9	348	167.000005	167
11	-8	486	110.000002	110
12	-7	570	59.000001	59
13	-6	606	14.000002	14
14	-5	600	-24.9999994	-25
15	-4	558	-57.999998	-58
16	-3	486	-84.9999992	-85
17	-2	390	-105.999999	-106
18	-1	276	-120.9999994	-121
19	0	150	#DIV/0!	-130

Figure 6-13. First derivative calculated on a worksheet by using Δx.
(folder 'Chapter 06 Examples', workbook 'Derivs by Sub Procedure', sheet 'Deriv')

The value in cell G21 illustrates that, using this technique, an x value of zero will have to be handled differently, since multiplying zero by 1.00000001 does not produce a change in x. This problem will be dealt with in a subsequent section.

First Derivative of a Worksheet Formula Calculated by Using a VBA Sub Procedure Using the Finite-Difference Method

The approach used in the preceding section can be performed by using a VBA **Sub** procedure. The VBA code is shown in Figure 6-14. By means of an input box the user identifies the range of cells containing the formulas for which the derivative is to be calculated, with a second input box, the corresponding cells containing the independent variable x, and with a third input box, the range of cells to receive the first derivative.

```
Option Explicit
Option Base 1
'++++++++++++++++++++++++++++++++++++++++++++
Sub Derivs()
Dim z As Integer, N As Integer
Dim Old_Ys() As Double, New_Ys() As Double, Old_Xs() As Double,
Dim Derivs() As Double, increment As Double
Dim known_Xs As Object, known_Ys As Object, cel As Object

increment = 0.00000001

'Use the Set keyword to create an object variable
Set known_Ys = Application.InputBox _
("Select the range of Y values", "STEP 1 OF 3", , , , , 8)
N = known_Ys.Count
ReDim Old_Ys(N), New_Ys(N), Old_Xs(N), Derivs(N)
z = 1
For Each cel In known_Ys
  Old_Ys(z) = cel.Value
  z = z + 1
Next cel

Set known_Xs = Application.InputBox _
("Select the range of X values", "STEP 2 OF 3", , , , , 8)
z = 1
For Each cel In known_Xs
  Old_Xs(z) = cel.Value
  cel.Value = Old_Xs(z) * (1 + increment)
  z = z + 1
Next cel
z = 1
For Each cel In known_Ys
  New_Ys(z) = cel.Value
  z = z + 1
Next cel
z = 1
For Each cel In known_Xs
  cel.Value = Old_Xs(z)
  z = z + 1
Next cel

Application.InputBox("Select the destination for derivatives", _
"STEP 3 OF 3", , , , , 8).Select
For z = 1 To N
  Derivs(z) = (New_Ys(z) - Old_Ys(z)) / (increment * Old_Xs(z))
  ActiveCell.Offset(z - 1, 0).Value = Derivs(z)
Next

End Sub
```

Figure 6-14. Sub procedure to calculate first derivative.
(folder 'Chapter 06 Examples', workbook 'Derivs by Sub Procedure', module 'Derivs')

	A	B	G	H	I
1	**Numerical Differentiation**				
2	F(x) = tx^3 + ux^2 + vx + w				
3		t	1		
4		u	-3	delta	
5		v	-130	1.00E-08	
6		w	150		
7				**Δy/Δx**	
8	**x**	**y**	By worksheet formula	From calculus	By macro
9	-10	150	230.0000006	230	230.000001
10	-9	348	167.000005	167	167.000005
11	-8	486	110.000002	110	110.000002
12	-7	570	59.000001	59	59.0000013
13	-6	606	14.000002	14	14.0000016
14	-5	600	-24.9999994	-25	-24.9999994
15	-4	558	-57.999998	-58	-57.999998
16	-3	486	-84.9999992	-85	-84.9999992
17	-2	390	-105.999999	-106	-105.999999
18	-1	276	-120.9999994	-121	-121
19	0	150	#DIV/0!	-130	
20	1	18	-132.999998	-133	-133

Figure 6-15. Calculating the first derivative of a formula.
(folder 'Chapter 06 Examples', workbook 'Derivs by Sub Procedure', sheet 'Deriv')

The **Sub** procedure saves the values of x and y from the worksheet (OldX and OldY), then writes the incremented value of x (NewX) to the worksheet cell. This causes the worksheet to recalculate and display the corresponding value of $y + \Delta y$ (NewY). The derivative is calculated and written to the destination cell. Finally, the original value of x is restored. Figure 6-15 illustrates the spreadsheet of Figure 6-13 after the **Sub** procedure has been run. The errors produced by this method are much smaller than those produced by the function based on LINEST.

The code in Figure 6-14 can easily be modified to calculate the partial derivatives of a function with respect to one or several parameters of the function (e.g., *dy/da*, *dy/db*, etc.) for a cubic equation. Similar code is used in the SolvStat macro (see Chapter 14, "The Solver Statistics Add-In") and a similar approach is used in the Solver itself (see "How the Solver Works" in Chapter 14).

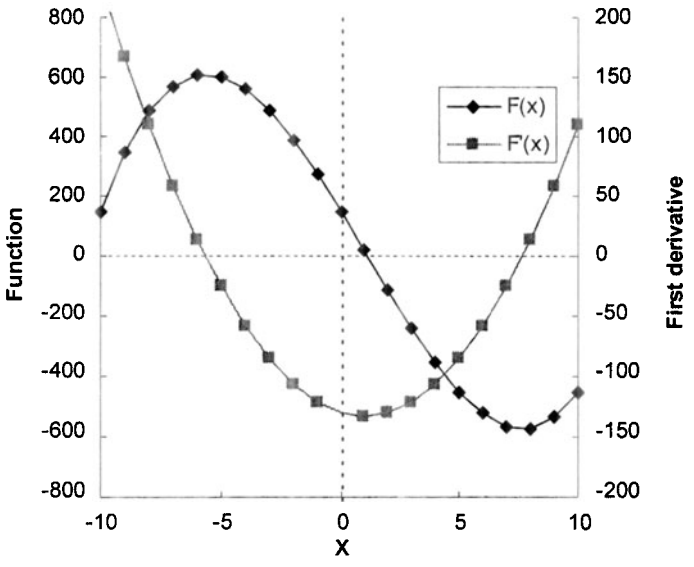

Figure 6-16. A chart of a function and its first derivative.
(folder 'Chapter 06 Examples', workbook 'Derivs by Sub Procedure', sheet 'Deriv')

The advantage of using a **Sub** procedure is that the derivative can be obtained easily, even for the most complicated worksheet formulas. All of the difficult calculations are done when the spreadsheet updates after the new value of *x* is entered in, for example, cell A9. The disadvantage of a **Sub** procedure is that if changes are made to precedent cells in the worksheet, the **Sub** procedure must be run in order to update the calculations.

First Derivative of a Worksheet Formula Calculated by Using a VBA Function Procedure Using the Finite-Difference Method

Unlike the **Sub** procedure described in the preceding section, a **Function** procedure automatically recalculates each time changes are made to precedent cells. A **Function** procedure to calculate the first derivative of a formula in a cell would be very useful. However, a function procedure can't use the approach of the preceding section (i.e., changing the value of the cell containing the *x* value), since a function procedure can't change the contents of other cells. A different approach will have to be found.

The following VBA code illustrates a simple **Function** procedure to calculate the first derivative *dy/dx* of a formula in cell, using the same approach that was used in the preceding section: the procedure calculates OldX, OldY,

NewX and NewY in order to calculate $\Delta x/\Delta y$. But in this function procedure, both the worksheet formula and the independent variable are passed to the function as arguments. The procedure is shown simply to illustrate the method; a number of modifications, to be described later, will be necessary in order to produce a "bulletproof" procedure.

The basic principle used in this **Function** procedure is the following:

(i) The two arguments of the function are references to the independent variable x and the cell containing the formula to be differentiated, $F(x)$.

(ii) Use the **Value** property to obtain the values of the arguments; these are OldX and OldY.

(iii) Use the **Formula** property of the cell to get the worksheet formula to be differentiated as the text variable FormulaText.

(iv) Use the SUBSTITUTE worksheet function to replace references to the x variable in FormulaText by the incremented x value, NewX.

(v) Use the **Evaluate** method to get the new value of the formula. This is NewY.

Since other procedures in this chapter and in subsequent chapters will use the same method for modifying and evaluating a formula, it will be worthwhile to examine the VBA code shown in Figure 6-17. The syntax of the function is FirstDerivDemo(***expression,variable***). The nine lines of code in this procedure perform the following actions:

(1) Get FormulaString, the worksheet formula (as text) by using the **Formula** property of *expression*.

(2) Get OldY, the value of the worksheet formula, by using the **Value** property of *expression*.

(3) Get XRef, the reference to the independent variable x, by using the **Address** property of *variable*. The address will be an A1-style absolute reference

(4) Get OldX, the value of the independent variable x, by using the **Value** property of *variable*.

(5) Calculate NewX, the incremented value of the independent variable, by multiplying OldX by 1.000000001.

(6) Convert all references in FormulaString to absolute by using the **ConvertFormula** method.

(7) Replace all instances of XRef in FormulaString by the value of the new variable NewX. This is done by using the SUBSTITUTE worksheet function. For example, the formula string

```
=3*$B$3^3+5*$B$3^2-5*$B$3+11
```

when cell $B$3 contains the value 2, is converted to

```
=3*2.00000002^3+5*2.00000002^2-5*x+11.
```

(8) Calculate NewY, the new value of the function, by applying the **Evaluate** method to the new formula string.
(9) Calculate and return the first derivative.

```
Option Explicit
Function FirstDerivDemo(expression, variable) As Double
'Custom function to return the first derivative of a formula in a cell.

Dim OldX As Double, OldY As Double, NewX As Double, NewY As Double
Dim FormulaString As String, XAddress As String

FormulaString = expression.Formula
OldY = expression.Value
XAddress = variable.Address  'Default is absolute reference
OldX = variable.Value
NewX = OldX * 1.00000001
FormulaString = Application.ConvertFormula(FormulaString, xlA1, xlA1, _
xlAbsolute)          'Convert all references in formula to absolute
FormulaString = Application.Substitute(FormulaString, XAddress, NewX)
NewY = Evaluate(FormulaString)
FirstDerivDemo = (NewY - OldY) / (NewX - OldX)
End Function
```

Figure 6-17. Function procedure to demonstrate calculation of first derivative.
(folder 'Chapter 06 Examples', workbook 'Derivs by VBA (Part 2)', module 'Demo')

Examples of the first derivative of some worksheet formulas calculated by the custom function are shown in Figure 6-18. The formula in cell D3 is

= FirstDerivDemo (C3,B3)

The formulas labeled "exact" in column E are the appropriate formulas from differential calculus for the first derivative of the respective functions. For example, the formula in cell E3 is

=9*B3^2+10*B3-5

	A	B	C	D	E	F
1	**Demo to Illustrate Use of Simple First Derivative Function**					
2	Function	x	F(x)	F'(x)	exact	% error
3	$y=3x^3+5x^2-5x+11$	2	45	51.0000003	51	-5.2E-07
4	$y = \sin x$	1	0.84147	0.5403023	0.5403023	-5.8E-08
5	$y=e^{-x}$	-1	0.36788	0.3678794	0.3678794	2.5E-07
6	$y = a^x$ (e.g., a = 3.5)	2.4	19.4734	24.3955256	24.3955252	-1.5E-06

Figure 6-18. Using a simple **Function** procedure to calculate some first derivatives.
(folder 'Chapter 06 Examples', workbook 'Derivs by VBA (Part 2)', sheet 'Demo Function')

Improving the VBA Function Procedure

The simple procedure shown in Figure 6-17 requires some modification.

First, the simple procedure replaces all instances of XRef, the reference to the independent variable *x*, in FormulaString with a number value. For example, a cell reference such as A2 will be replaced with a number value such as 0.05. But there are cases where the substring A2 should not be replaced. Our procedure needs to handle the following possibilities, all of which contain the substring A2 within FormulaString:

(i) the reference XRef and references in FormulaString may be relative, absolute or mixed,

(ii) FormulaString contains a name such as BETA2,

(iii) FormulaString contains a reference such as AA2, or

(iv) FormulaString contains a reference such as A25.

By using the **Address** property to obtain an absolute reference (e.g., $A$2) and using the **ConvertFormula** method to convert all references in FormulaString to absolute, we have already eliminated problems arising from cases (i), (ii), and (iii). Only case (iv) poses a problem: the substring $A$2 in $A$25 will be substituted by 0.05, yielding 0.055. And so, as is often the case with computer programming, a project that initially appeared to be simple requires some additional programming.

We could write a formula parser that would break FormulaString into its component parts and inspect each one. Not impossible, but that would require extensive programming. A much simpler solution turns out to be the following: by means of a loop, we replace each instance of, for example, A2 individually, and, instead of replacing the reference with a number (e.g., 0.05), we replace the reference with the number concatenated with the space character (e.g., 0.05 0). We then evaluate the resulting string after each substitution. The reference $A$25 yields the string 0.05 5. When evaluated, this gives rise to an error, and an **On Error GoTo** statement is used so that the faulty substitution is not incorporated into the FormulaString to be evaluated. Inspection of the code in the latter half of the procedure in Figure 6-21 should make the process clear.

A second problem with the simple procedure of Figure 6-17 is that when x = 0, NewX = OldX, NewY = OldY and the procedure returns a #VALUE! error. The error produced by a zero value for the independent variable x is handled by adding an additional optional argument *scale_factor*. The syntax of the function is dydx(*expression, reference*, **Optional** *scale_factor*). If x is zero, a value for *scale_factor* must be entered by the user. *Scale_factor* is used to calculate the Δx for numerical differentiation. *Scale_factor* should be the same order of magnitude as typical x values used in the formula.

The **Function** procedure is shown in Figure 6-19.

```
Option Explicit
Function dydx(expression, variable, Optional scale_factor) As Double
'Custom function to return the first derivative of a formula in a cell.
'expression is F(x), variable is x.
'scale_factor is used to handle case where x = 0.
'Workbook can be set to either R1C1- or A1-style.

Dim OldX As Double, NewX As Double, OldY As Double, NewY As Double
Dim delta As Double
Dim NRepl As Integer, J As Integer
Dim FormulaString As String, XRef As String, dummy as String
Dim T As String, temp As String

delta = 0.00000001

'Get formula and value of cell formula (y).
FormulaString = expression.Formula    'Returns A1-style formula; default is
absolute.
OldY = expression.Value
'Get reference and value of argument (x).
OldX = variable.Value
XRef = variable.Address    'Default is A1-style absolute reference.

'Handle the case where x = 0.
'Use optional scale_factor to provide magnitude of x.
'If not provided, returns #DIV0!
If OldX <> 0 Then
  NewX = OldX * (1 + delta)
Else
  If IsMissing(scale_factor) Or scale_factor = 0 Then _
  dydx = CVErr(xlErrDiv0): Exit Function
  NewX = scale_factor * delta
End If

'Convert all references to absolute
'so that only text that is a reference will be replaced.
T = Application.ConvertFormula(FormulaString, xlA1, xlA1, xlAbsolute)

'Do substitution of all instances of x reference with value.
'Substitute reference, e.g., $A$2,
'with a number value, e.g., 0.2, followed by a space
'so that $A$25 becomes 0.2 5, which results in an error.
'Must replace from last to first.
NRepl = (Len(T) - Len(Application.Substitute(T, XRef, ""))) / Len(XRef)
For J = NRepl To 1 Step -1
   temp = Application.Substitute(T, XRef, NewX & " ", J)
   If IsError(Evaluate(temp)) Then GoTo pt1
   T = temp
pt1: Next J
NewY = Evaluate(T)
dydx = (NewY - OldY) / (NewX - OldX)
End Function
```

Figure 6-19. Improved **Function** procedure to calculate first derivative.
(folder 'Chapter 06 Examples', workbook 'Derivs by VBA (Part 2)', module 'FirstDeriv')

	A	B	C	D	E	F
1	**Demo to Illustrate Use of Advanced First Derivative Function**					
2	Reference in formula or in argument can be absolute, relative, mixed or a name.					
3	Function	x	F(x)	F'(x)	exact	% error
4	$y=3x^3+5x^2-5x+11$	2	45	51.00000027	51	-5.2E-07
5	y = sin x	1	0.84147	0.54030231	0.5403023	-5.8E-08
6	$y=e^{-x}$	-1	0.36788	0.36787944	0.3678794	2.5E-07
7	$y = a^x$ (e.g., a = 3.5)	2.4	19.4734	24.39552511	24.395525	3.8E-07
8	$y=3x^3+5x^2-5x+11$	0	11	#VALUE!	-5	#VALUE!
9	$y=3x^3+5x^2-5x+11$	0	11	-4.99999988	-5	2.4E-06

Figure 6-20. Using the improved function procedure to calculate some first derivatives. The optional argument *scale_factor* is used in row 9 to eliminate the #VALUE! error seen in row 8. (folder 'Chapter 06 Examples', workbook 'Derivs by VBA (Part 2)', sheet Better Function')

The examples in Table 6-20 illustrate the values of the first derivative calculated by using the function dydx, compared with the "exact" values.

The worksheet formulas in column C and the corresponding functions in column D are:

C4 =3*B4^3+5*B4^2-5*B4+11	D4 =dydx($C$4,$B$4)
C5 =SIN($B5)	D5 =dydx(C5,B5)
C6 =EXP($B$6)	D6 =dydx(C6,B6)
C7 =a^B7	D7 =dydx(C7,B7)
C8 =3*B8^3+5*B8^2-5*B8+11	D8 =dydx(C8,B8)
C9 =3*B9^3+5*B9^2-5*B9+11	D9 =dydx(C9,B9,1)

Rows 4–6 illustrate that relative, absolute or mixed references can be used in the worksheet formula or in the arguments of the custom function. Row 9 illustrates the use of the optional argument *scale_factor* when the x value is zero.

Second Derivative of a Worksheet Formula

The VBA code for the **Function** procedure shown in Figure 6-21 requires only slight modification to provide a function that returns the second derivative of a function as a cell formula. The syntax of the d2xdy2 function is identical to that of the function dydx.

The code is shown in Figure 6-21. The function calculates the central derivative uing three points (see the formula in Table 6-1). Note that the multiplier used to calculate Δx is 1E-4 instead of 1E-8.

```
Option Explicit
Function d2ydx2(expression, variable, Optional scale_factor) As Double
'Custom function to return the second derivative of a formula in a cell.
'expression is F(x), variable is x.
'Uses central difference formula.
'scale_factor is used to handle case where x = 0.
'Workbook can be set to either R1C1- or A1-style.

Dim OldX As Double, OldY As Double
Dim NewX1 As Double, NewX2 As Double
Dim NewY1 As Double, NewY2 As Double
Dim XRef As String
Dim delta As Double
Dim FormulaString As String, T As String
Dim temp As String
Dim NRepl As Integer, J As Integer

delta = 0.0001

'Get formula and value of cell formula (y).
FormulaString = expression.Formula    'Returns A1-style formula
OldY = expression.Value
'Get reference and value of argument (x).
OldX = variable.Value
XRef = variable.Address    'Default is A1-style absolute reference

'Handle the case where x = 0.
'Use optional scale_factor to provide magnitude of x.
'If not provided, returns #DIV0!
If OldX <> 0 Then
  NewX1 = OldX * (1 + delta)
  NewX2 = OldX * (1 - delta)
Else
  If IsMissing(scale_factor) Or scale_factor = 0 Then _
  d2ydx2 = CVErr(xlErrDiv0): Exit Function
  NewX1 = scale_factor * delta
  NewX2 = -scale_factor * delta
End If

'Convert all references to absolute
'so that only text that is a reference will be replaced.
FormulaString = Application.ConvertFormula(FormulaString, xlA1, xlA1, _
xlAbsolute)

T = FormulaString
NRepl = (Len(T) - Len(Application.Substitute(T, XRef, ""))) / Len(XRef)
'Do substitution of all instances of x reference with incremented x value
For J = NRepl To 1 Step -1
   temp = Application.Substitute(T, XRef, NewX1 & " ", J)
   If IsError(Evaluate(temp)) Then GoTo pt1
   T = temp
pt1: Next J
'Evaluate the expression.
NewY1 = Evaluate(T)
```

```
T = FormulaString
'Now do substitution of all instances of x reference with decremented x value
For J = NRepl To 1 Step -1
   temp = Application.Substitute(T, XRef, NewX2 & " ", J)
   If IsError(Evaluate(temp)) Then GoTo pt2
   T = temp
pt2: Next J
NewY2 = Evaluate(T)
d2ydx2 = (NewY1 + NewY2 - 2 * OldY) / Abs((NewX1 - OldX) * (NewX2 - OldX))
End Function
```

Figure 6-21. **Function** procedure to calculate second derivative.
(folder 'Chapter 06 Examples', workbook 'Derivs by VBA (Part 2)', module 'SecondDeriv')

Figure 6-22 illustrates the use of the dydx and d2ydx2 custom functions. The formula in cell B4 is

```
=aa*A4^3+bb*A4^2+cc*A4+dd
```

(aa, bb, cc, dd are named ranges. The formula in cell C4 is

```
=dydx(B4,A4,1)
```

	A	B	C	D	E	F	G	H
1	First and Second Derivative Functions							
2	$y = 2x^3 - 20x^2 + 11x + 30$							
3	x	F(x)	F'(x)	exact	% error	F"(x)	exact	% error
4	-5	-775	361.0000021	361	5.8E-07	-100.0000002	-100	-2.0E-07
5	-4	-462	267.0000003	267	1.0E-07	-88.0000002	-88	-2.0E-07
6	-3	-237	185.0000020	185	1.1E-06	-75.9999997	-76	-4.5E-07
7	-2	-88	114.9999996	115	3.9E-07	-64.0000003	-64	-5.0E-07
8	-1	-3	57.0000001	57	1.6E-07	-52.0000000	-52	-7.5E-08
9	0	30	10.9999998	11	1.4E-06	-40.0000001	-40	-2.8E-07
10	1	23	-22.9999999	-23	-3.9E-07	-28.0000002	-28	-6.6E-07
11	2	-12	-45.0000001	-45	-2.0E-07	-15.9999999	-16	-6.1E-07
12	3	-63	-55.0000004	-55	-7.0E-07	-4.0000003	-4	-8.3E-06
13	4	-118	-52.9999997	-53	-5.0E-07	7.9999998	8	2.8E-06
14	5	-165	-38.9999993	-39	-1.7E-06	19.9999999	20	2.5E-07

Figure 6-22. Using **Function** procedures to calculate
first and second derivatives of a function.
(folder 'Chapter 06 Examples', workbook 'Derivs by VBA (Part 2)', sheet 'First and Second Derivs')

Note the use of the optional argument *scale_factor* that prevents an error in cells C9 and F9 when the value of the independent variable in cell A9 is zero.

Concerning the Choice of Δx for the Finite-Difference Method

In preceding sections, the $x + \Delta x$ used for the calculation of the derivatives was calculated by multiplying x by 1.00000001. Thus Δx is a "scaled" increment. An alternative approach would have been to use a constant Δx of, e.g., 0.00000001. Either approach has its advantages and disadvantages.

The constant-increment method eliminates the need to handle the case of $x = 0$ separately. However, the method fails when x is very large, e.g., 10^8. The scaled-increment method handles a wide range of x values, but fails in some special cases, such as for $\sin x$ when $x = 1000$.

You should be aware of these limitations when using the dydx and d2ydx2 custom functions.