SampleSections

Here are several sample sections (at least one from each of the eleven chapters) from the second edition of my *Advanced Excel for Scientific Data Analysis*, as a downloadable pdf file. The corresponding page numbers have been added to their section headings. This is, of course, copyrighted material. Here are the sections, with their page numbers in *this* file, and an asterisk identifying sections new to the second edition.

1.5.2 Contour maps *pp. 33-36*

Contour maps are more difficult and time-consuming to generate because they involve interpolation, but they can be made in Excel. Here we will highlight Isol.xla (for ***iso-l****ivello*, Italian for iso-level or contour lines), a free, open-source custom macro written by Simonluca Santoro, which will generate an Excel contour diagram. If desired, it will even combine this with Mapper to color the resulting graph.

It can be downloaded from Leonardo Volpi's website http://digilander.libero.it/foxes/SoftwareDownloads.htm as Isol.zip, or from my website under ContourMap. Its operation and use are described in Volpi's "Plotting z = f(x,y) contour lines with Excel", which you can download from his foxes website under /documents.htm, and is included in the ContourMap download from my website. Isol.xls takes a function defined on a rectangular grid, and interpolates in this grid by triangulation. It then draws contour diagrams using the interpolated data.

As input Isol.xls takes either an Excel function, a rational function of the type $f(x,y)/g(x,y)$ where $f(x,y)$ and $g(x,y)$ are polynomials with positive integral powers of x and y of up to 4^{th} order, or a look-up table, and generates the corresponding contour diagram, color map, or their combination. Just call the Excel file Isol, specify the desired type of input and output, the data range (in terms of minimum and maximum values of x, y, and z), the number of data points per interval (i.e., the computational resolution), the number of contour lines, and the name and location of the output file. Then push Run; the macro will generate the plot as a free-standing graph in the specified output file. Afterwards you can treat that plot as any other fixed Excel plot, e.g., by changing its labels. In Fig. 1.5.3 we illustrate Isol by showing a plot of the dimensionless van der Waals curve $z = (3/x^2+y)\,(3x–1)/8$, where $x = P/P_c$, $y = V/V_c$, and $z = T/T_c$. For reproduction in this book, the subroutine Mapper(xmin, xmax, ymin, ymax, Formula, Caso, Coeff, OutputFileName) was modified to yield a gray scale. When in Isol.xls you can find this subroutine by clicking Alt◡F11, in the menu of the VBEditor module select View ⇨ Project Explorer, and then click on Modules, where you will find Mapper under Mapper_mod. You can also go there to change its color scheme.

The detail available with Isol can be increased by using a denser grid (of up to 205 data points for x and/or y) selected in the top left-hand corner of the Isol Input sheet, but this comes at the cost of slower execution, because the algorithm is fairly computer-intensive at about 100 operations per computed data point. It is therefore best to start with the contour diagram ('iso-level') and few contour lines, and to introduce additional

lines and color only after you are satisfied that the result is otherwise just the way you want it.

Exercise 1.5.2:

(1) Open Isol.xls. It shows a special spreadsheet page, in-between a regular sheet and a dialog box. The first choices to be made are in blocks C8:E11, where you use the up arrow to select "3", an Excel function, and C14:C16, where you pick "1" for iso-level, i.e., contour lines.

(2) As your exercise function take an example from Volpi's "Plotting z = f(x,y) contour lines with Excel", and use a van der Waals curve made dimensionless by scaling the pressure P, volume V and absolute temperature T of a nonideal gas by its critical pressure P_c, critical volume V_c, and critical absolute temperature T_c, in which case the equation takes the simple form $z = (3/x^2+y)\,(3x–1)\,/\,8$ where $x = P/P_c$, $y = V/V_c$, and $z = T/T_c$.

(3) Go to cell O24 on the Isol.xls sheet, where you enter the above formula as `=(3/M24^2+N24)*(3*M24-1)/8`, where M24 contains x and N24 holds y.

(4) Go to the table in B2:E5. In C3 enter the minimum value of 0.2 for x, and in D3 its maximum value, 3.2. (Letting x start at 0 would yield a run-time error as the first term in the expression for z, $3/x^2$, is computed for $x = 0$.) Likewise enter a 0 for y_{min} in C4, and the value 0.4 for y_{max} in D4. Set the corresponding resolutions in E3 and E4 to 20.

(5) In C5 place the value of 0.4 for z_{min}, and in D5 the value 2.0 for z_{max}. In E5 deposit the number of contour lines you want, in this case 9, for $z = 0.4\ (0.2)\ 2.0$.

(6) Click on the New button in cells I2:J2, and specify a spreadsheet name, such as IsoTest1 in a folder of your choice. Click on the Parameters button in J9:K11 to preserve these settings, then press Run in J6:K8. After a short delay you should see the contour diagram.

(7) Verify that you indeed get the graph you want; it should have the main features of Fig. 1.5.3, but neither its resolution nor its background color. If the formula entered in O24 is incorrect, remedy it and run again. When you are satisfied that the plot is correct in principle, refine the settings as follows.

(8) In E3 and E4 select 200 instead of 20 to increase the computational resolution (and the corresponding time taken by the computation), and in E5 place the number 17 to create contour lines at $z = 0.4\ (0.1)\ 2.0$.

(9) In block C13:C17 select "3" for contour lines plus background coloring. Press Run once more.

(10) It will now take much longer to compute the contours, because the resolution of both x and y has been increased ten-fold, and there are twice as many contour lines to calculate, resulting in a 200-fold increase in computational effort.

(11) At this point you can leave well-enough alone, or further embellish the result. If you do the latter, and especially if you want to use it in a Word file or PowerPoint presentation, I suggest that your first action is to add a worksheet. Click on the Grafico tab, right-click and select Insert, and make a new Worksheet, most likely to be called Sheet1.

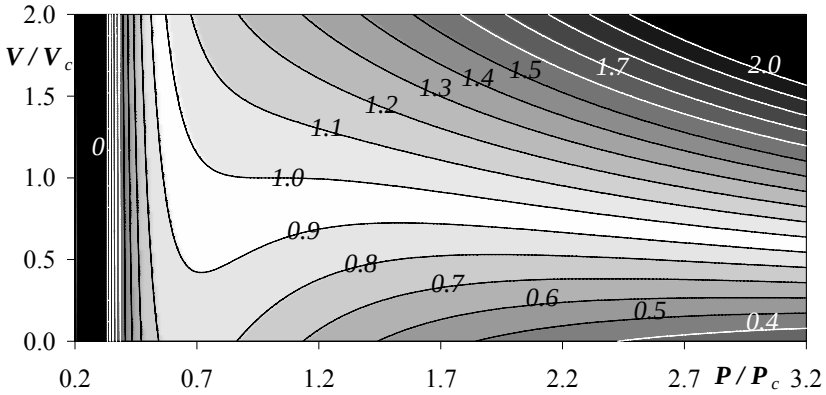

Fig. 1.5.3: A contour map drawn with Simonluca Santoro's Isol.xls with background coloring (here shown in gray-scale), showing the dimensionless van der Waals equation $T/Tc = [3/(P/Pc)^2+V/Vc]\,(3P/Pc–1)\,/\,8$ where T is absolute temperature, P is pressure, V is volume, and the subscript c denotes the critical point. Array size 200 × 200; 21 contour lines. Values of T/T_c are indicated with the curves.

(12) Click on the Grafico tab to return to the graph, then move it from its fixed format as a separate sheet to a graph floating on a spreadsheet. This will allow you to shape it in any way you want without running into pixellation problems. So, click on the chart area (e.g., on the white band above the plot) to activate it (as indicated by the nine black handles that will appear on its corners and halfway its sides), right-click, select Location, and in the resulting Chart Location dialog box select As object in Sheet1.

(13) Again activate the Chart, and align it with the spreadsheet gridlines by pulling two opposite corners to grid corners while depressing the Alt key. Now you can apply any embellishments you want, and subsequently transfer the graph to Word or PowerPoint by merely activating the *underlying* spreadsheet cells, and using Paste Special to embed (or link) the graph.

2.12 How precise is the standard deviation?

pp. 95-96

The standard deviation provides an estimate of the precision of a number, i.e., of its reproducibility under near-identical experimental conditions. We now ask the next question: how precise is the standard deviation itself? Below we will answer that question. Could we perhaps keep asking that question of the resulting answer, generating a never-ending series of questions and answers, such as "What is the imprecision of the imprecision of the imprecision, etc.?", much like a continued fraction, or the images of images of images in a hall of mirrors? Fortunately, the question asked in the heading of this section turns out to have a surprisingly simple, definitive answer.

Least squares analysis is based on the assumption that the data follow a Gaussian distribution, reflecting mutually independent, relatively small, random deviations from the average. The variance v (i.e., the square of the standard devation s) can then be shown to follow a so-called χ^2 ("chi-square") distribution. This distribution, as distinct from the Gaussian distribution, depends on only *one* parameter, in this case the number of degrees of freedom $N–P$, where N is the number of data points analyzed, and P the number of adjustable model parameters used, e.g., $P = 1$ for a proportionality, $P = 2$ for a straight line, etc. This χ^2 distribution representing the variance v has a mean $N–P$, a variance v_v of that variance of $2(N–P)$, and a standard deviation s_v of that variance of $\sqrt{[2(N–P)]}$.

However, we need the standard deviation of the standard deviation, s_s, not the standard deviation of the variance, s_v. (For comparison with the original quantities we need their standard deviations, because they have matching dimensions, which the variances do not.) How do we go from one to the other?

Let a quantity q have a variance v and a standard deviation $s = \sqrt{v}$, so that we can express the quantity together with its imprecision as $q \pm s = q \pm \sqrt{v}$. Likewise we formulate the variance v with its standard deviation s_v as $v \pm s_v$ and the standard deviation s with its standard deviation s_s as

$$\begin{aligned} s + s_s &= \sqrt{v \pm s_v} = \sqrt{v}\sqrt{1 \pm s_v/v} \approx \sqrt{v}\,(1 \pm s_v/2v) \\ &= s\,(1 \pm s_v/2v) = s\,(1 \pm s_s/s) \end{aligned} \qquad (2.12.1)$$

where we have made the usual assumption that s_v/v is very much smaller than 1, so that we can use the general expansion $\sqrt{(1 \pm \delta)} \approx 1 \pm \delta/2$ for $\delta \ll 1$. From this we see that the relative standard deviation $s_s/s = s_v/2v$ of the

standard deviation s of the quantity q is one-half of the relative standard deviation s_v/v of the variance, so that

$$s_s/s = s_v/2v = \sqrt{2(N-P)}\,/\,[2\,(N-P)] = 1\,/\,\sqrt{2(N-P)} \qquad (2.12.2)$$

A rigorous derivation of this result, for the case of the population standard deviation (rather than the sample standard deviation, i.e., with N instead of $N–P$), can be found in, e.g., J. F. Kenney and E. S. Keeping, *Mathematics of Statistics*, 2nd ed., Van Nostrand, Princeton (1951), vol. 2 pp. 170-171.

Note that all the above properties of the χ^2 distribution depend only on the number of degrees of freedom, $N–P$, and are *independent* of the actual x and y values of the data set. We can therefore estimate the imprecision of the standard deviation merely on the basis of the magnitude of $N–P$, as illustrated in Table 2.12.1.

$N–P$	s_s/s	s_s/s in %
2	0.5000	50
5	0.3162	32
10	0.2236	22
20	0.1581	16
50	0.1000	10
100	0.0707	7.1
200	0.0500	5.0
500	0.0316	3.2
1,000	0.0224	2.2
10,000	0.0071	0.7

Table 2.12.1: The relative standard deviation of the standard deviation, as given by (2.12.2), as a function of the number of degrees of freedom, $N–P$.

Clearly, the standard deviation is often over-specified, i.e., reported with far more decimals than are significant. For example, performing a simple weighing in triplicate ($N = 3$, $P = 1$, hence $N – P = 2$) yields a standard deviation with a relative precision of ±50%, so that there is no need to insist on many decimal places for such a quantity. Minor differences in standard deviations are often statistically insignificant.

In order to get a standard deviation with no more than 10% relative imprecision, we will need at least 50 observations, while at least 5 000 measurements are required for a standard deviation with a 1% maximum imprecision. It is therefore wise to consider most standard deviations as imprecision *estimates*, and the same applies to all quantities directly derived from the standard deviation, such as "confidence" measures.

2.18 Phantom relations *pp. 110-113*

In using least squares it is tacitly assumed that the input data represent *independent* measurements. If that is not the case, quite misleading results may be obtained, as illustrated by the following problem (#9 on page 383) of K. Connors, *Chemical Kinetics, the Study of Reaction Rates in Solution* (VCH, 1990):

> "From the last four digits from the office telephone numbers of the faculty in your department, systematically construct pairs of "rate constants" as two-digit numbers times 10^{-5} s^{-1} at temperatures 300 K and 315 K (obviously the larger rate constant of each pair to be associated with the higher temperature). Make a two-point Arrhenius plot for each faculty member, evaluating $\Delta H^{\ddagger}$ and $\Delta S^{\ddagger}$. Examine the plot of $\Delta H^{\ddagger}$ against $\Delta S^{\ddagger}$ for evidence of an isokinetic relationship."

Essentially, the reader is asked to take two arbitrary two-digit y-values y_1 and y_2, assign them to pre-selected x-values x_1 and x_2 respectively, compute the resulting slope a_1 and intercept a_0, repeat this for a number of arbitrary input pairs y (for the same two x-values), and then plot the resulting a_1-values versus a_0, or vice versa. The actual procedure is somewhat less transparent, since it also involves sorting the input data, a logarithmic transformation, and giving the slopes and intercepts thermodynamic names, all steps that tend to obscure the true nature of the problem. Moreover, the above assignment uses only positive input numbers. Below we will simply take pairs of random two-digit integer values for y, associate them with two fixed x-values such as $x_1 = 300$ and $x_2 = 320$, compute the resulting slopes and intercepts, and then plot these against each other.

Exercise 2.18.1:

(1) In cells B2 and C2 place the labels y1 and y2 respectively. Do the same in cells E2:F2, and in cells H2:I2 deposit the labels a0 and a1 respectively.

(2) In cells B4 and C4 deposit the instruction `=INT(200*(RAND()-0.5))`, which will generate random two-digit integers between –100 and +100. Copy these instructions down to row 23.

(3) The numbers in B4:C23 will change every time you change something on the spreadsheet. In order to have a fixed set of random numbers, highlight B4:C23, copy it with Ctrl⌣c, highlight cell E4, and use Edit ⇨ Paste Special ⇨ Values to copy the *values* of y_1 and y_2 so obtained. After that, use the data in block E4:F23 as your random input data, while ignoring those in B4:C23 that keep changing while you work the spreadsheet.

(4) Based on the data in E4:F23, compute in column H the slope of each pair of data points (x_1, y_1), (x_2, y_2) as $(y_2 - y_1) / (x_2 - x_1)$, and in column I the corresponding intercepts as $(x_2 y_1 - x_1 y_2) / (x_2 - x_1)$.

The data in Fig. 2.18.1 seem to fall on or near a straight line, for which Trendline yields the formula $y = -311.18\ x - 0.8877$, with $R^2 =$

0.9983. Is this what you would have expected for having used random input numbers for *y*? If not, what happened?

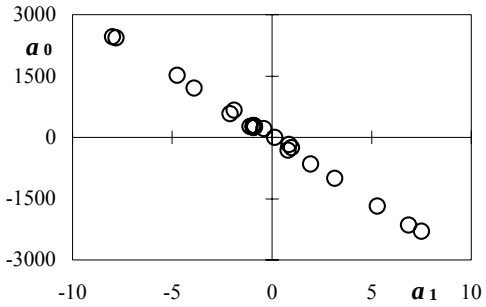

Fig. 2.18.1: An example of a phantom line you might find with $x_1 = 300$ and $x_2 = 320$.

Because each pair of input numbers *y* of this graph is completely determined by the calculated slope and intercept for given input values of *x*, the graph uses strongly *linearly correlated* pairs of input data. We already encountered the formula for that correlation, (2.10.1). The sign of (2.10.1) explains the negative correlation (causing the negative slope da_0/da_1 in Fig. 2.18.1), and the effect is the more pronounced the larger is Σx, i.e., the more eccentric are the *x*-values used. Plotting such slopes and intercepts against each other will then lead to a convincingly linear but physically meaningless near-linear relationship, approximating the proportionality $y = -x_{av}\, x$. Instead, you are merely verifying the correlation between slope and intercept, see (2.10.1), as is perhaps more evident after we rewrite $y = -x_{av}\, x$ using more appropriate symbols as $a_0 = -x_{av}\, a_1$.

This is the origin of the isokinetic relationship of J. E. Leffler, *J. Org. Chem.* 20 (1955) 1202, and illustrates what the covariance can do for you if you don't watch it. An extensive discussion of this problem, as well as a suggested solution, was given by Krug et al. in *J. Phys. Chem.* 80 (1976) 2335, 2341. For an interesting (and only seemingly alternative) explanation of this phantom relationship see G. C. McBane, *J. Chem. Educ*. 75 (1998) 919.

Exercise 2.18.1 (continued):

(6) Use the same *y*-values collected in columns H and I, but now analyze them for a pair of *x*-values *centered* around the average $x_{av} = 310$, so that $x_1 = -10$ and $x_2 = +10$. Does this support the above explanation?

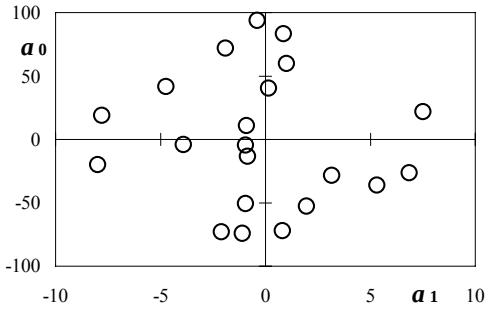

Fig. 2.18.2: The same y-values as in Fig. 2.18.1 analyzed with $x_1 = -10$ and $x_2 = +10$.

Given that the input data were random, which are the parameters that determine the 'line' in Fig. 2.18.1? There is no significant intercept, just a slope, and the latter is simply $-(\Sigma x)/N$, i.e., minus the average value of x. In the above example we have $-(\Sigma x)/N = -(300+320)/2 = -310$, so that we would expect $y = -310\ x$, which compares well with the result of Trendline, $y = -311.18\ x - 0.8877$, as illustrated in Fig. 2.18.3. Indeed, as already noticed by Leffler, in many cases the absolute values of the reported slopes of isokinetic plots were close to the average temperatures of the data sets considered. In such cases the isokinetic effect is nothing more than an artifact of incorrectly applied statistics.

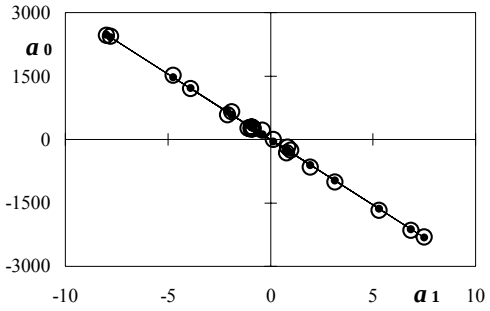

Fig. 2.18.3: The data from Fig. 2.18.1 (large open circles) and, for comparison, those computed as $a_0 = -x_{av}\, a_1$ (small filled circles connected by a thin line).

3.7 Spectral mixture analysis pp. 129-131

Figure 3.7 illustrates the absorption spectra of four fantasy species, made of one or more Gaussian peaks, and of an imaginary mixture made of these species. The peaks were simulated as $a \exp[-(x-c)^2/(2b^2)]$; instead of the exponential part you can also use the instruction `=NormDist(x,mean,stdev,false)` to generate Gaussian curves $\left(1/\sigma\sqrt{2\pi}\right) \exp\left[-(x-\bar{x})^2/2\sigma^2\right]$ where $\bar{x}$ is the mean (locating the position of the peak center), σ the standard deviation (defining its width), and where 'false' specifies the Gaussian curve rather than its integral. In exercise 3.7.1 we simulate such spectra, compute the spectrum of a mixture of these components (assuming their additivity, as in Beer's law, and the absence of other light-absorbing species), add noise to all components, then use multivariate analysis to reconstruct the composition of that mixture.

Exercise 3.7.1:

(1) In column A deposit wavelengths, and in columns B through E calculate four fantasy spectra, each with one or more Gaussian peaks. Each Gaussian peak requires three constants: an amplitude *a*, a standard deviation *b* or σ, and a center frequency *c* or mean $\bar{x}$.

(2) In columns M through Q generate random Gaussian ('normal') noise, and in columns H through K make somewhat noisy single-component spectra by adding some noise from column N to the spectrum of column B, etc., in order to create more realistic single-species spectra.

(3) Near the top of the spreadsheet enter four concentrations, and use these in column G to make a synthetic 'mixture spectrum' of the four single-component spectra, each multiplied by its assigned concentration, plus added noise from column M. (You could do without columns B through E by adding noise directly to the data in columns B through E, and then subtracting that same noise from the mixture spectrum. Noise in the single-component spectra and in the spectrum of the simulated mixture should of course be independent.)

(4) Plot the spectra of columns G through K, which might now look like those in Fig. 3.7.1. Note that the resulting curve does not show distinct features easily identifiable with any of its constituent spectra. In this particular example we have used the data of Table 3.7.1, together with noise standard deviations of 0.005 for all components as well as for the synthetic mixture. You should of course use your own data to convince yourself that this is no stacked deck.

(5) Highlight the data block in columns G through K, and call LS0 for a multivariate analysis of the mixture spectrum in terms of its four component spectra.

The results of that analysis are shown in Table 3.7.2. Despite the added noise, the absence of stark features, and considerable overlap between the various single-component spectra, the composition of the mixture is recovered quite well.

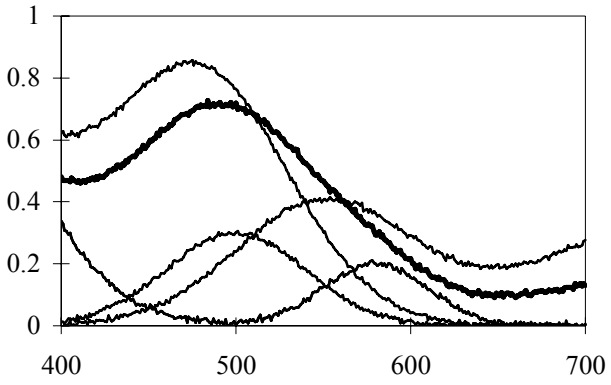

Fig. 3.7.1: The simulated single-component spectra (thin lines) and the spectrum of their mixture (heavy line). The simulation parameters used, as well as the composition of the mixture and the results of its analysis, are listed in Tables 3.7.1 and 3.7.2. Independent Gaussian noise (mean 0, st. dev. 0.005) has been added to all curves.

	ampl:	***mean:***	***st.dev.:***		***ampl:***	***mean:***	***st.dev.:***
curve 1:	300	270	80	*curve 3*:	30	500	40
	100	480	50	*curve 4*:	200	300	60
curve 2:	50	550	50		15	580	30
	70	760	80				

Table 3.7.1: The constants for the Gaussian peaks used in generating Fig. 3.7.1 with the function `NormDist()`.

	curve 1	***curve 2***	***curve 3***	***curve 4***
mixture composition:	0.650	0.500	0.300	0.200
recovered:	0.648±0.003	0.496±0.003	0.305±0.011	0.207±0.007

Table 3.7.2: The assumed and recovered composition of the synthetic mixture.

The above method is simple and quite general, as long as spectra of all mixture constituents are available. In the analysis you can include spectra of species that do not participate in the mixture: for those species, the calculation will simply yield near-zero contributions. However, a missing constituent spectrum will cause the method to fail if its contribution to the mixture spectrum is significant.

A final note: the numbers obtained for the recovered composition are mutually dependent. When a subsequent result depends on more than one concentration, the covariance matrix should be used in its computation, rather than the individual standard deviations.

3.22 The power of simple statistics pp. 171-174

This chapter should neither leave you with the impression that least squares analysis is very complicated, nor that it is mere frosting on the cake, something you do *after* the experiment has run, when you are ready to report the final results. To the contrary, more often than not, the application of least squares analysis is quite straightforward, especially when using the ready-made tools provided by Excel and its add-ins. Moreover, when used *during* rather than after the experimental stage of a study, it can often help you identify (and remedy) experimental problems early on, thereby saving you from having to redo the experiment.

First, a caveat: the following example involves a small set of absorption measurements reported by M. S. Kim, M. Burkart & M.-H. Kim in *J. Chem. Educ.* 83 (2006) 1884, as reproduced in Table 3.22.1. In that communication, these data were not shown for their scientific implications, but only to illustrate how to visualize the process of least squares. Therefore they should not be judged as more than a demo data set, and we will use them here in that vein, merely to highlight how least squares can and should be used. Incidentally, the point listed at the origin was not really measured, but apparently was added subsequently as a mere "dummy data point", and for that reason is not included in Table 3.22.1. We therefore have just five observations, and the question is: how should they be analyzed?

concentration c (M)	***absorbance A***
0.025	0.097
0.050	0.216
0.100	0.434
0.150	0.620
0.200	0.796

Table 3.22.1: The absorbance data listed by Kim et al.

The simplest answer is to follow the theory, i.e., to use Beer's law, $A = abc$, which for a single absorbing species in a non-absorbing medium predicts a proportionality of the form $y = a_1x$, where y is the measured absorbance A, x is the known absorbate concentration c, while the slope a_1 is the product of the molar absorptivity a and the optical path length b through the light-absorbing solution. A sometimes recommended alternative is the straight line $y = a_0 + a_1x$, justified by assuming the possible presence of an absorbing background species (absent in this case, where the solutions were prepared in the lab from reagent-grade components) or

a constant instrumental offset. The latter is, of course, entirely avoidable by proper prior calibration.

Exercise 3.22.1:

(1) Enter the data for the absorbance A in one spreadsheet column, and the corresponding concentration c in the column to its immediate right. Label the columns, and make a copy of them elsewhere on the spreadsheet.

(2) Analyze one data set with LS0, the other with LS1. (You can, of course, use LinEst or Regression instead, once with and once without zero y-intercept.) For the proportionality, each of these routines will yield a slope of $4.08_4 \pm 0.06_6$, and a standard deviation s_f of the overall fit of the model function to the data of 0.018; for the straight line with arbitrary intercept, the corresponding results are a slope of $3.99 \pm 0.1_3$, an intercept of $0.01_4 \pm 0.01_6$, and $s_f = 0.019$.

Clearly, these data do not support the second model, because the intercept a_0 is smaller than its standard deviation s_0. Nor is there any other valid justification for using that model with these data: the test solutions were made by weighing and/or diluting a known, pure chemical in a high-purity, non-absorbing solvent (water), so that there was nothing to cause a constant background absorption. Arguments of instrumental drift should not be used, because such drift affects not only the origin but all measured data points. (It is, of course, most readily *observable* at the origin.) Instrumental drift is often, to a first approximation, a linear function of time, in which case one could make multiple series of measurements as a function of time, and apply a bivariate analysis with concentration and time as the independent variables. Preferably, though, once such drift is observed, it is minimized by proper instrument warm-up, using a constant-temperature lab, and by likewise controlling all its other causative factors. Prevention is always better than subsequent remediation.

Exercise 3.22.1 (continued):

(3) Now plot the data, or use the graph in Kim et al., *J. Chem. Educ.* 83 (2006) 1884, and look along its minor diagonal, i.e., from its origin at the bottom left to its top right corner. In such a foreshortened view, the data show a clear curvature. You will see the same by computing and plotting the residuals to either model curve: the first and last points are too low, the middle point too high for a straight line. Clearly, these data display some curvature not represented by a straight line.

(4) Make two more copies of the original data set, each with an extra column for c^2, and fit the parabolas $y = a_1x + a_2x^2$ and $y = a_0 + a_1x + a_2x^2$ to the data. For the two-parameter model you will find $a_1 = 4.5_3 \pm 0.1_4$, $a_2 = -2.6_7 \pm 0.8_3$, and $s_f = 0.01_0$; for the three-parameter model $a_0 = -0.024_0 \pm 0.005_9$, $a_1 = 5.0_1 \pm 0.1_3$, $a_2 = -4.5_8 \pm 0.5_7$, and $s_f = 0.004_0$.

Both parabolas yield plausible parameter values. Assuming for the sake of the exercise that these few observations really reflect a trend in the data rather than mere statistical fluctuations, we find at least three acceptable ways to represent these five data points, as $A = (4.08_4 \pm 0.06_6)\, c$

with $s_f = 0.01_8$, as $A = (4.5_3 \pm 0.1_4)\ c + (-2.6_7 \pm 0.8_3)\ c^2$ with $s_f = 0.01_0$, or as $A = -0.024_0 \pm 0.005_9 + (5.0_1 \pm 0.1_3)\ c + (-4.5_8 \pm 0.5_7)\ c^2$ with $s_f = 0.004_0$; however, the general straight line $y = a_0 + a_1 x$ is not among them. Incidentally, this latter conclusion also applies at concentrations below 0.1 M, where there are only two available measurements, because the necessarily exact fit of a two-parameter expression to just two data points has no predictive statistical value.

We see that some very simple considerations using readily available spreadsheet tools allow us to find plausible fits to these data, and to exclude an often advocated but in this case clearly inappropriate model, as indicated *by the data themselves*. However, to reach these conclusions we do need to look not only at the parameter values, but *also* at their imprecision estimates. (Incidentally, using Trendline should be discouraged in the physical sciences precisely because it does not provide such estimates.) And in order to select an appropriate model, once we look for an essentially empirical model describing the data, it is helpful to consider trends in the residuals. If this were a real experiment, it should of course be checked whether such nonlinearity is reproducible and, if it is, whether it is caused by some avoidable instrumental artifact, such as stray light or slits that are too wide, or (much less likely) by an actual deviation from Beer's law. In this way, by fully integrating least squares analysis in the measurement process rather than as an after-the-fact embellishment, we can derive maximal benefits from its use. Properly used statistics can be very helpful at the experimental stage, because they can reveal problems that may need to be fixed *at that stage*.

4.4.3 The titration of an acid salt with a strong base

pp. 203-207

We now consider a set of experimental data. As our example we use a recent report by A. L. Soli in *Chem. Educ.* 9 (2004) 42, which lists data observed for the titration of the acid salt KH_2PO_4 with NaOH. We use (4.3.3) where V_a is now the original volume of the titrated acid salt, and ${}^{H}F_a$ is the proton function

$$\begin{aligned} {}^{H}F_a &= [H^+] + [H_3PO_4] - [HPO_4^{2-}] - 2[PO_4^{3-}] - [OH^-] \\ &= [H^+] + (\alpha_3 - \alpha_1 - 2\alpha_0)\, C_a - [OH^-] \qquad (4.4.11) \\ &= [H^+] + \frac{([H^+]^3 - [H^+]K_{a1}K_{a2} - 2K_{a1}K_{a2}K_{a3})C_a}{[H^+]^3 + [H^+]^2 K_{a1} + [H^+]K_{a1}K_{a2} + K_{a1}K_{a2}K_{a3}} - \frac{K_w}{[H^+]} \end{aligned}$$

where the alphas are the concentration fractions, labeled with the number of attached protons. Meanwhile we have (4.3.9) for the strong base NaOH. Consequently, the progress of the titration is described by

$$V_b = \frac{-[H^+] - \dfrac{([H^+]^3 - [H^+]K_{a1}K_{a2} - 2K_{a1}K_{a2}K_{a3})C_a}{[H^+]^3 + [H^+]^2 K_{a1} + [H^+]K_{a1}K_{a2} + K_{a1}K_{a2}K_{a3}} + \dfrac{K_w}{[H^+]}}{[H^+] + C_b - \dfrac{K_w}{[H^+]}} V_a \qquad (4.4.12)$$

Equation (4.4.12) has two variables, $[H^+]$ and V_b, which both change continuously during the titration, at all times keeping each other in balance, and seven parameters: the two concentrations C_a and C_b, the original sample volume V_a, the three acid dissociation constants K_{a1}, K_{a2}, and K_{a3} of the triprotic acid H_3PO_4, and the ion product K_w of water. Of these, the volume V_a of the original sample is known from Soli's paper as V_a = 25 mL, and the titrant concentration as C_b = 0.1049 M.

The experimental observations are listed in Table 4.4.1, and cover the entire titration curve, from beginning to way past its equivalence point. These 54 data pairs should amply suffice to determine the five remaining unknowns: C_a, K_{a1}, K_{a2}, K_{a3}, and K_w. We will use Solver to assign numerical values to these five unknown parameters (or, more precisely, to their negative logarithms), and SolverAid to estimate the corresponding imprecisions.

V_b	pH	V_b	pH	V_b	pH
0.00	4.41	17.00	6.92	28.18	9.11
0.49	5.06	18.00	6.98	28.30	9.30
1.00	5.36	19.01	7.05	28.50	9.64
2.00	5.65	20.00	7.12	28.70	9.90
2.90	5.78	21.00	7.20	28.93	10.05
4.10	5.98	22.00	7.28	29.20	10.18
4.95	6.08	23.00	7.38	29.51	10.31
6.02	6.19	23.96	7.48	30.01	10.44
7.30	6.29	25.00	7.62	30.80	10.60
8.00	6.34	26.00	7.79	31.48	10.71
9.00	6.42	26.50	7.92	32.51	10.85
10.10	6.50	26.75	8.00	33.41	10.94
11.00	6.55	26.97	8.07	34.51	11.04
12.00	6.61	27.35	8.22	35.00	11.07
13.00	6.68	27.51	8.30	36.02	11.14
14.02	6.73	27.70	8.43	37.00	11.21
14.98	6.79	27.90	8.61	38.00	11.26
16.00	6.86	28.03	8.81	39.11	11.32

Table 4.4.1: The experimental data of Soli for the titration of 25.00 mL aqueous KH_2PO_4 with 0.1049 M NaOH. The volume V_b of added NaOH is in mL.

Exercise 4.4.4:

(1) Set up the spreadsheet with columns for pH, $[H^+]$, V_b, $V_{b,calc}$, $V_{b,guess}$, and R, or some other abbreviation for residual.

(2) Also make a column with the labels, and a column with the corresponding numerical values, for the known parameters V_a and C_b, as well as for the unknowns C_a, pK_{a1}, pK_{a2}, pK_{a3}, and pK_w, for K_{a1}, K_{a2}, K_{a3}, and K_w, and for SSR, the sum of squares of the residuals. It is convenient for subsequent use to group these as V_a and C_b, C_a through pK_w, K_{a1} through K_w, and SSR, separated by empty cells.

(3) The duplication of K's and their negative logarithms pK is intentional: the K-values are most convenient for using (4.4.12) in computing $V_{b,calc}$, but the corresponding pK's must be used in Solver, which otherwise will ignore the smaller K-values.

(4) For V_a deposit the value 25, and for C_b the value 0.1049.

(5) For C_a, pK_{a1}, pK_{a2}, pK_{a3}, and pK_w, use guess values; in Fig. 4.4.6 we have used $C_a = 0.08$, p$K_{a1} = 3$, p$K_{a2} = 6$, p$K_{a3} = 11$, and p$K_{a1} = 14$, but feel free to select others, especially ones that show a better initial fit.

(6) Compute K_{a1} as 10^{-pKa1}, and do similarly for the other K-values.

(7) Place the values of V_b and pH in their columns, compute $[H^+]$ as $=10^{-pH}$, and in the column for $V_{b,calc}$ compute V_b based on (4.4.12) and the known and guessed parameter values.

(8) Copy the resulting values (but not their formulas) to the next column, under the heading $V_{b,guess}$, using Edit ⇨ Paste Special ⇨ Values.

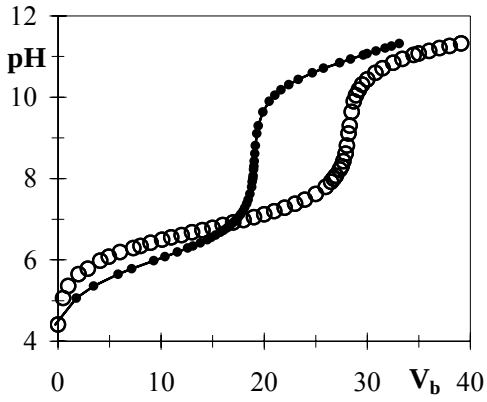

Fig. 4.4.6: The progress curve for the titration of 25.00 mL aqueous KH_2PO_4 with 0.1049 M NaOH. Open circles are the experimental data of Soli, see Table 4.4.1; the curve with small filled circles is computed with the assumed parameter values, *before* Solver is used.

(9) Plot $V_{b,calc}$ vs. pH. If you want to keep a record of the difference Solver makes, also make the corresponding plot of $V_{b,guess}$ vs. pH.

(10) Calculate SSR with the SUMXMY2 function, using the data under V_b and $V_{b,calc}$.

(11) Call Solver, and minimize SSR by letting it adjust the values of the five unknowns: C_a, pK_{a1}, pK_{a2}, pK_{a3}, and pK_{a1}. This will affect the values of $V_{b,calc}$ but not those under $V_{b,guess}$.

(12) Call SolverAid to find the standard deviations of C_a, pK_{a1}, pK_{a2}, pK_{a3}, and pK_{a1}, and their covariance matrix. Also plot the corresponding array of linear correlation coefficients.

(13) Compute the residuals $V_{b,calc} - V_b$, and plot them as a function of pH.

Now consider the so-called *equivalence volume* V_{eq}, which is defined as the volume V_b at which an equivalent amount of base has been added to the acid, i.e., the value of V_b for which $C_aV_a = C_bV_b$. Traditionally, one first determines V_{eq} from the titration curve, and then uses this to compute C_a as C_bV_{eq}/V_a, at which time one uses the standard deviations of V_a and C_b. These should therefore not be considered in the above estimate of V_{eq}. In the above example we find $V_{eq} = C_aV_a/C_b = (0.11859_2 \pm 0.00008_2) \times 25 / 0.1049 = 28.26_3 \pm 0.01_9$ mL.

For a comparison with Soli's empirical approach we restrict the analysis to 17 data points (from V_b = 26.00 to 30.01 mL) around V_{eq}. When these are analyzed in the above way we find $C_a = 0.11846_7 \pm 0.00006_5$. Again neglecting the standard deviations in V_a and C_b, this yields $V_b = 28.23_3 \pm 0.01_5$ mL, which can be compared directly with $V_{eq} = 28.22 \pm 0.026\sqrt{17} = 28.22 \pm 0.11$ mL obtained by Soli. We see that the theory-based analysis of these 17 data is some seven times more precise than Soli's strictly empirical approach. In the analysis of high-grade data, the caliber of the model used typically determines the quality of the results.

We now consider these numerical results. (1) The values obtained for both C_a and K_{a2} are quite satisfactory. This is to be expected: C_a is computed directly, without the intermediary of an equivalence point. The value of C_a is essentially independent of (in this example: neglected) activity corrections, but this does not apply to value of K_{a2}. (2) The value for K_{a1} obtained is not very precise, because the titration of KH_2PO_4 with NaOH does not provide experimental data in the region where pH ≈ pK_{a1}. To obtain a better value for K_{a1} one would have to either titrate H_3PO_4 instead of KH_2PO_4, or titrate the acid salt with a strong acid. (3) The values obtained for K_{a3} and K_w are not very precise either. As can be seen from their linear correlation coefficient, these two constants are strongly coupled, and consequently neither of them can be determined very well from this type of experiment. (4) Although these were experimental data, their analysis with a model that neglects activity effects is quite satisfactory in terms of determining the unknown concentration C_a, compare exercise 4.4.3. Of course, the values obtained for the equilibrium constants K_{a1} through K_{a3} and K_w do not agree too well with their literature values, since the latter have been corrected for activity effects by, in effect, extrapolating them to 'infinite' dilution.

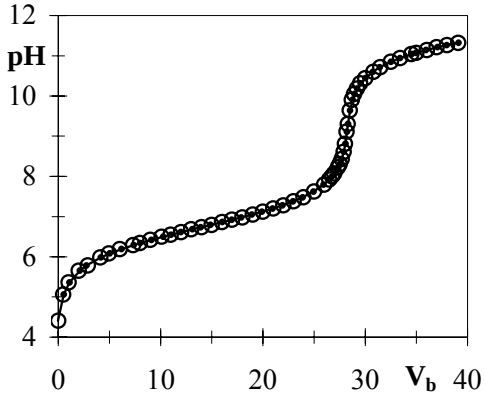

Fig. 4.4.7: The progress curve for the titration of 25.00 mL aqueous KH_2PO_4 with 0.1049 M NaOH. Open circles are the experimental data of Soli, see Table 4.4.1; the small filled circles are computed with the parameter values found by Solver. Note how the individual, computed points nest in the open circles representing the experimental data: they don't merely fit the curve, but their specific locations on it.

5.9 Data compression *pp. 292-295*

Least squares can be used to extract the essential data from, say, a noisy but otherwise linear calibration curve. Likewise we can use Fourier transformation to extract some essential features from an arbitrary signal. For example, a common chromatographic detector uses ultraviolet light to illuminate the effluent, and monitors the resulting fluorescence. This can give both qualitative and quantitative information on the eluted sample components. We will here focus on the qualitative aspect, i.e., on how to use the spectral information to identify the chemical identity of the eluting sample, assuming that we have a computer 'library' of reference spectra that can be consulted.

A fluorescence spectrum can cover a fairly wide spectral range, but often contains only a relatively small number of identifiable features. A library search can therefore be simplified considerably by compressing the data, even if such compression leads to some distortion. Fourier transform filtering can often serve this purpose, as described, e.g., by Yim et al. in *Anal. Chem*. 49 (1977) 2069. We will illustrate the method here with spectral data available on the web from the Oregon Medical Laser Center of the Oregon Graduate Institute at http :// omlc.ogi.edu/ spectra/PhotochemCAD/html/index.html.

Exercise 5.9:

(1) Go to the above website, and select a compound. At the bottom of its page, below its fluorescence spectrum (assuming it has one), click on Original Data, then highlight and copy those data. Below we will use tryptophan as an example; feel free to select instead any other fluorescence spectrum, from this or any other source. Some of these fluorescence spectra exhibit less fine-structure, some have more. Obviously, the more details one can measure and compare, the more reliable the identification can be.

(2) Leave the web page, open Notepad, and paste the spectral data there; Notepad is a convenient intermediary between external data and Excel. Save the file.

(3) Open a spreadsheet, select File ⇨ Open, and in the resulting Open dialog box specify where to look for the data file, the file name, and the file type (in this case: Text Files). In the Text Import Wizard specify Delimited, and the spectral data will appear in your spreadsheet.

(4) Graph both the fluorescence intensity FI and its logarithm. The logarithmic representation will be used here as it shows more characteristic features.

(5) Since these data were not intended for use with Fourier transformation, the number of data points, 441, is not an integer power of 2. We now have two options: reducing the data set to the nearest smaller suitable number, 256, or 'padding' the data to 512, the next-higher integer power of 2. Here we illustrate how to accomplish the latter.

(6) Perusal of the graph of log(FI) vs. wavelength shows that, at long wavelengths λ, it exhibits an essentially linear dependence on λ. We therefore extrapo-

late this linear relationship to λ = 535.5 nm by fitting the data from, e.g., 380 to 400 nm to a line, and then using the computed intercept and slope to calculate values for 400 < λ < 540 nm. Plot these to make sure that the extrapolated data are indeed continuous with the measured ones.

(7) Fourier-transform this extended data set, and plot the result. Most of the signal will be concentrated in the few lowest frequencies, see Fig. 5.9.1.

(8) In a copy, set the higher-frequency contributions to zero, inverse transform the data, and again plot the result. By repeating this while retaining, say, the 10, 20, 30, and 40 lowest frequencies, you will get a sense of how few low-frequency data are needed to represent the over-all shape of the curve, and how many more must be kept to show the minor shoulder near 300 nm.

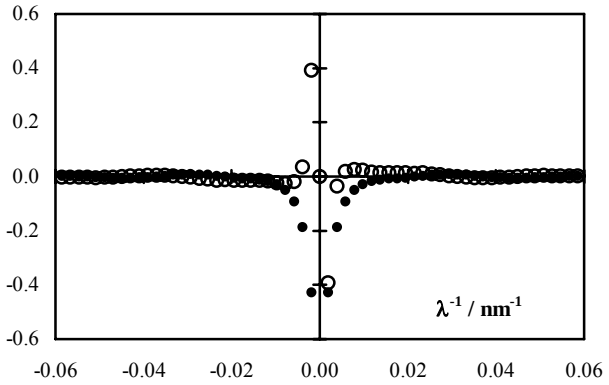

Fig. 5.9.1: The 61 low-frequency components of the Fourier transform of the tryptophan fluorescence spectrum (for excitation at 270 nm) after its extrapolation to 512 data points. The zero-frequency point is far off-scale in this plot.

(9) Figure 5.9.1 shows that retaining only 30 of the 256 frequencies is sufficient to exhibit the general shape of the fluorescence peak, without noticeable loss of information. On the other hand, the main fluorescence peak can be represented with fewer than 10 (positive and negative) frequencies.

(10) The small 'hook' at the lowest wavelengths is an artifact resulting from the requirement that the Fourier-transformed signal be a repeatable unit. In this example we were lucky; had the signal levels at the two extremes of the wavelength scale been very different, the consequent change would have led to undesirable oscillations, which can only be avoided with additional effort.

By Fourier transformation, a fluorescence spectrum can be represented by a relatively small number of frequency terms, thereby greatly facilitating library search routines for computer-based identification. In the present example, 61 data (the real and imaginary components at the lowest 30 frequencies plus the zero-frequency term) can be used to replace the original 441 data points. A further reduction to 31 points can be achieved through symmetry, because the data at negative frequencies can be reconstituted from those at the corresponding positive frequencies:

since the input function $g(t)$ is real, $G(-f)$ must be the complex conjugate of $G(f)$.

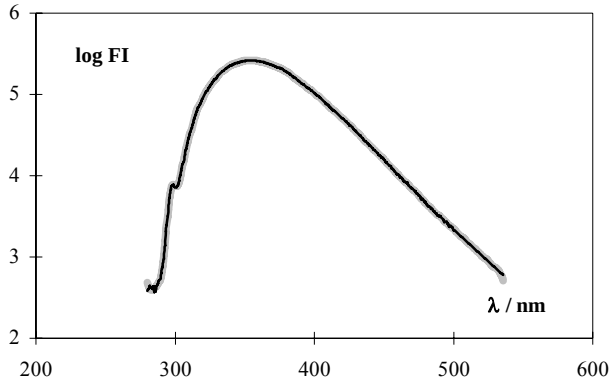

Fig. 5.9.2: The original data set extrapolated to 535.5 nm (thin black line) and its representation in terms of only 30 frequencies (broad gray band).

In section 4.6 we already encountered examples of spectral fitting, and we therefore ask here how the Fourier transform and least squares methods compare. In principle, nonlinear least-squares methods are more flexible, since they are not limited to a basis set of sines and cosines. Relatively simple spectra can often be described to the same accuracy with far fewer parameters than required for Fourier transformation. But this strongly depends on the number of features to be represented: with more peaks the balance shifts in favor of the Fourier transform method, as it does with typical nuclear magnetic resonance, infrared, and mass spectra.

Then there are practical constraints: fitting by nonlinear least squares may require personal judgement, and may therefore be more difficult to automate than Fourier transformation. On the other hand, Fourier transformation may need some help if the spectrum does not tend to zero at its extremes. The choice may also depend on whether the spectral information already exists in Fourier-transformed format inside a measuring instrument. For cataloguing and data searching, the Fourier transform method may be the more convenient, because it expresses all data sets in terms of the same, limited set of fixed frequencies, which greatly facilitates their intercomparison.

6.7 Iterative deconvolution using Solver

pp. 348-351

The van Cittert deconvolution method is general but quite sensitive to noise. An alternative approach was introduced by Grinvald & Steinberg, Anal. Biochem. 59 (1974) 583. It uses reverse engineering based on an assumed analytical (and therefore noise-free) function for the undistorted model signal $\boldsymbol{s}_m = f(a_i)$ in terms of one or more model parameters a_i. We convolve the model signal $\boldsymbol{s}_m$ based on guessed parameters a_i with the experimental transfer function $\boldsymbol{t}$ to obtain $\boldsymbol{r}_m = \boldsymbol{s}_m \otimes \boldsymbol{t}$. We then use Solver to adjust the a_i by minimizing the sum of squares of the residuals between $\boldsymbol{r}_m$ and the experimental (or, in our example, simulated) $\boldsymbol{r}_{exp}$. The requirement that $\boldsymbol{s}_m$ be describable as an explicit analytical function is somewhat restrictive but, when applicable, can greatly reduce noise.

Because macros do not self-update, Solver cannot respond automatically to the effect of parameter changes that involve macros. This means that, for a non-manual program, we can either rewrite Solver so that it can accommodate macros, or (much simpler) perform the convolution using a function rather than a macro. The latter approach is illustrated in exercise 6.7.1.

Exercise 6.7.1:

(1) Our example will be modeled after exercise 6.2.2 and fig. 6.2.3, i.e., based on a single exponential decay. In cells A1:D1 deposit the column labels #, s, t, and r for the rank number # (which can represent time, wavelength, etc), original (undistorted) signal ***s***, filter or transfer function ***t***, and result ***r***.

(2) In A3 place the value –100, in A4 the instruction =A3+1, and copy this down to cell A303.

(3) In B103 insert the instruction `=$D$97*EXP(-$D$98*A103)` for the transfer function ***t***, and copy this down to row 303. In cell D97 enter an amplitude value (such as 1) and in cell D98 a value for a rate constant (e.g., 0.03), with accompanying labels in column C.

(4) For the transfer function ***t***, in cell C103 place the instruction `=EXP(-1*(LN(1+(A103-$D$100)/$D$101))^2)` and copy this down to row 303.

(5) Highlight A103:C303 and call the macro Convolve, which will write the convolution ***r*** in D103:D303.

(6) Deposit Gaussian ('normal') noise (with mean 0 and standard deviation 1) in N103:O303, and supply corresponding scale values, such as 0.01 in cell N100 and 0.02 in cell O100.

(7) In cell E103 place the instruction `=C103+$N$100*N103` to simulate a noisy transfer function $\boldsymbol{t}_{exp}$, in cell F103 use `=D103+$O$100*O103` for a noisy response signal $\boldsymbol{r}_{exp}$, and copy both instructions down to row 303. In row 1 place appropriate labels. You now have a set of simulated, noisy data to try the deconvolution. In a real application these simulated values should of course to be replaced by experimental data, as anticipated by their labels.

(8) In G103 place a model function, such as `=$I$97*EXP(-$I$98*A103)`. Copy it down to row 303, and label it in G1 as s model. Place a guess value for the amplitude in I97, and an initial estimate for the rate constant in I98, with accompanying labels in column H. Do not place any numbers or text in G3:G102, but instead fill it with, e.g., yellow, to remind yourself to keep it clear.

(9) In cell H103 deposit the function `=Convol(G103:G202,$E$103:$E$202,100)` and copy it all the way to row 303. In cell H1 label the column as r model.

(10) Go to the VBA module and enter the following code for this function:

```
Function Convol(Array1, Array2, N)

Dim i As Integer
Dim Sum1 As Double, Sum2 As Double
Dim Array3 As Variant
ReDim Array3(1 To 2 * N)

Sum2 = 0
For i = 1 To N
  Sum2 = Sum2 + Array2(i)
Next i

For i = 1 To N
  Array3(i) = Array2(N + 1 - i)
Next i

Sum1 = 0
For i = 1 To N
  Sum1 = Sum1 + Array1(i - N + 1) * Array3(i)
Next i

Convol = Sum1 / Sum2

End Function
```

(11) In cell I100 deposit the function `=SUMXMY2(F103:F303,H103:H303)` and place a corresponding label such as SSR= in H100.

(12) Call Solver, and Set Target Cell to I100, Equal to Min, By Changing Cells I97:I98. Then engage SolverAid to find the corresponding uncertainties.

(13) Compare your results with those in fig. 6.7.1 which shows them before and after using Solver.

This exercise demonstrates the principle of the method. We started with amplitude $a = 1$ and rate constant $k = 0.03$, used as initial guess values $a_m = 1.5$ and $k_m = 0.02$, and then found $a_m = 1.005 \pm 0.009$ and $k_m = 0.0300 \pm 0.0004$, with a standard deviation $s_y = 0.02_0$ of the fit in $\boldsymbol{r}$.

In practice, try to keep the convolving custom function as simple as possible, and especially avoid IF statements which tend to slow it down. We have used a barebones custom function Convol() to keep it simple, even though it is wasteful of spreadsheet real estate.

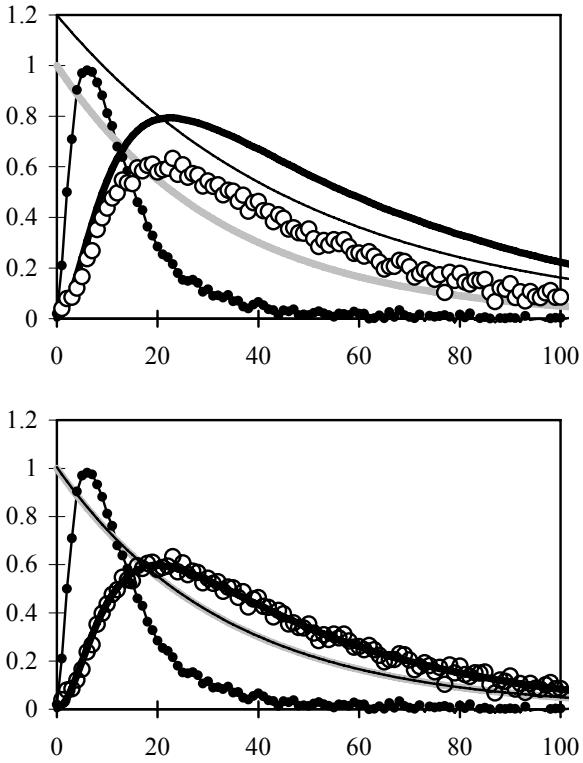

Fig. 6.7.1: The assumed signal $\boldsymbol{s}$ (gray band), the noisy transfer function $\boldsymbol{t}_{exp}$ (line with small solid points), the result $\boldsymbol{r}_{exp}$ obtained by convolving $\boldsymbol{s}$ with the (noise-free) transfer function and then adding noise (large open circles), the assumed model function $\boldsymbol{s}_m$ (line) and the resulting function $\boldsymbol{r}_m$ after convolving $\boldsymbol{s}_m$ with $\boldsymbol{t}_{exp}$ (heavy line). The top panel shows the situation just before calling Solver, the bottom panel that after Solver has been used.

When the data involve a shift in time, try to adjust its initial estimate by trial and error in the graph, using a critical region (such as where the response takes off) where it makes a significant difference, because the least squares criterion is often not very sensitive to horizontal shifts, but looks at vertical differences, and averages over the entire curve.

7.9 Stability pp.394-398

While the fourth-order Runge-Kutta method leads to higher accuracy, the semi-implicit method can be more stable. We will illustrate this by numerically integrating the differential equation

$$\frac{dy}{dx} = y^2 + 1 \tag{7.9.1}$$

with $y_{x=0} \equiv y_0 = 0$. For the semi-implicit Euler method we rewrite (7.9.1) as

$$\frac{\Delta y}{\Delta x} = (y + \Delta y/2)^2 + 1 = y^2 + y\Delta y + (\Delta y)^2/4 + 1$$
$$\approx y^2 + y\Delta y + 1 \tag{7.9.2}$$

so that

$$\Delta y \approx \frac{y^2 + 1}{1/\Delta x - y} \tag{7.9.3}$$

where we have again linearized y^2 by neglecting the term $(\Delta y)^2/4$.

Exercise 7.9.1:

(1) Start a new spreadsheet, with space for values of *n* at the top, and below this a column for *x*, and four columns for *y*. Fill the *x*-column with the numbers 0 (0.01) 3, and the top cells in the *y*-columns with zeroes for y_0.

(2) In the first *y*-column, implement the semi-implicit Euler method with the command (say in cell B6, assuming that the value of y_0 is placed in cell B5) `=B5+(B5^2+1)/(1/(A6-A5)-B5)`.

(3) In the next *y*-column use a custom function to compute y with smaller time increments, such as

```
Function siEulerY(oldX1, oldX2, n, oldY) As Double
'semi-implicit Euler method for exercise 7.7

Dim Y As Double, step As Double
Dim i As Integer

n = CInt(n)
Y = oldY
step = (oldX2 - oldX1) / n
For i = 1 To n
  Y = Y + (Y * Y + 1) / ((1 / step) - Y)
Next i
siEulerY = Y

End Function
```

(4) Place a corresponding value of *n* in cell C1, and the instruction `=siEulerY(A5,A6,$C$1,C5)` in cell C7.

(5) In the next *y*-column compute *y* with the explicit fourth-order Runge-Kutta method, using a custom function such as

```
Function e4RKY(oldX1, oldX2, n, oldY)
'explicit fourth-order Runge-Kutta for exercise 7.9

Dim X As Double, Y As Double, step As Double
Dim k1 As Double, k2 As Double
Dim k3 As Double, k4 As Double
Dim i As Integer

X = oldX1
Y = oldY
n = CInt(n)
step = (oldX2 - oldX1) / n
For i = 1 To n
  k1 = step * (Y ^ 2 + 1)
  k2 = step * (((Y + k1 / 2) ^ 2) + 1)
  k3 = step * (((Y + k2 / 2) ^ 2) + 1)
  k4 = step * (((Y + k3) ^ 2) + 1)
  Y = Y + (k1 + 2 * k2 + 2 * k3 + k4) / 6
  X = X + step
Next i
e4RKY = Y

End Function
```

(6) Place the value $n = 1$ in cell D1, and in cell D6 the instruction `=e4RKY(A5,A6,$D$1,D5)`.

(7) Use the same custom function in the last *y*-column, but this time with $n = 10$ (in cell E1).

(8) Plot the results for y as a function of x obtained with these two methods.

Figure 7.9.1 illustrates what you will find. The semi-implicit Euler method has no problem with the integration, either for a simple one-line instruction and $\Delta t = 0.01$, or with a custom function and an effective Δt of 0.01/1000 = 0.00001. The Runge-Kutta method starts out fine, but stops at x = 1.57, regardless of whether we use $\Delta t = 0.01$ or multiple steps (as with n = 10).

By now you may have recognized the function we have just integrated: it is y = tan(x) which, as you can readily verify, is indeed the solution to (7.9.1) with $y_0 = 0$. The Runge-Kutta method apparently cannot get past the discontinuity in y at $x = \pi/2 \approx 1.5708$, while this same hurdle doesn't faze the semi-implicit Euler method. We merely illustrate this here for a particular differential equation, but it reflects a rather general property: implicit methods, and even semi-implicit ones, are more stable than explicit methods.

Having the exact solution allows us to compute and plot the errors. The results so obtained are shown in Figs. 7.9.2 and 7.9.3, and show that the Runge-Kutta method has far smaller algorithmic errors when it works, but fails completely when it doesn't. When a custom function is used to reduce the effective step size, the semi-implicit Euler method can combine accuracy and reliability, as illustrated in Fig. 7.9.3.

It is sometimes suggested that the Runge-Kutta method is all you need for the numerical integration of ordinary differential equations with one-point boundary conditions. But in Fig. 7.9.2 the one-line semi-implicit Euler instruction eventually outperforms the Runge-Kutta method, and the use of smaller effective step sizes Δt / n makes the corresponding inaccuracies quite acceptable, see Fig. 7.9.3. Because of its greater stability, the semi-implicit Euler method is often a better bet, except when you already *know* the function to be well behaved over the entire range of interest.

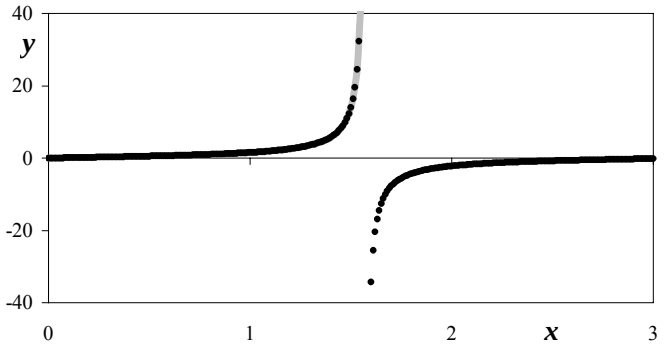

Fig. 7.9.1: The function y found by numerical integration of (7.9.1) with $y_0 = 0$. Solid dots: results from the semi-implicit Euler method for $\Delta t = 0.01$. Broad gray band: results from the explicit fourth-order Runge-Kutta method for $\Delta t = 0.01$. Note that the latter fails for $x > 1.57$.

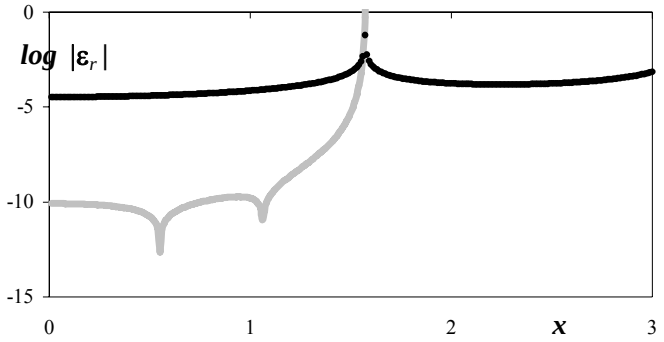

Fig. 7.9.2: The (logarithms of the absolute values of the) relative errors ε_r in the numerical simulation. Solid dots: results from the semi-implicit Euler method for $\Delta t = 0.01$. Broad gray band: results from the explicit fourth- order Runge-Kutta method for $\Delta t = 0.01$, which fails for $x > \pi/2$. The cusps in the latter curve correspond to sign changes in $\varepsilon_r(x)$.

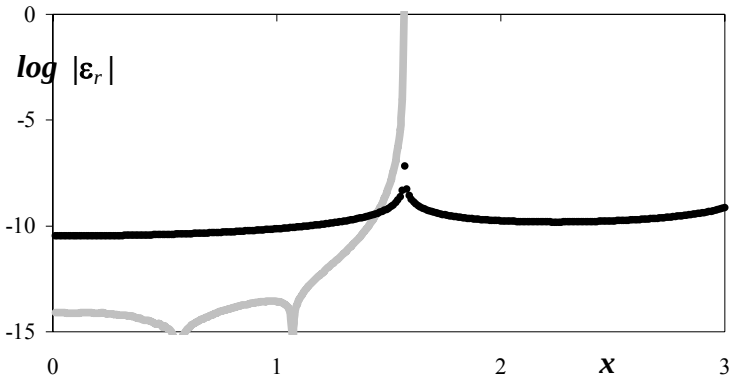

Fig. 7.9.3: The (logarithms of the absolute values of the) relative errors ε_r in the numerical simulation. Solid dots: results from the semi-implicit Euler method for $\Delta t = 0.01$ with $n = 1000$, yielding relative errors smaller than 10^{-7}. Broad gray band: results from the explicit fourth- order Runge-Kutta method for $\Delta t = 0.01$ with $n = 10$.

8.15 Attaching cell comments p. 444

When you see the result of a function, you can click on the cell that contains it, find its name, and (within its brackets) the cells that it used for its input. But a macro is much sneakier: you find the result of its action, but you will not know what input information it used, or even which macro produced it. And even when that would be clear from the context, the macro may have been updated, and you will not know which of its versions was used, or when. This, together with the possibility of subsequent, undocumented changes in one or more of its input values (which will not be reflected in the macro output) is one of the weakest spots in trying to validate an Excel spreadsheet.

While this problem can be reduced by painstaking documentation, it helps to let the macro self-document its use, automatically. This can be built in with Cell Comments, which can identify all the needed information that would not be available otherwise, such as the name of the macro used, the date and time of use (which the computer can provide), the source(s) of its input, any options exercised or special conditions used, etc. (Again, this will not prevent changes in the input parameters after the macro was last used, but is still useful when the purpose is documentation rather than fraud prevention.)

Here is a simple example. Before calling the macro, highlight a cell. The specific information given will of course depend on the application.

```
Sub AttachCellComments()

ActiveCell.Select                   ' clear prior comments
ActiveCell.ClearComments
ActiveCell.AddComment
ActiveCell.Comment.Visible = False ' if True, the comments
                                   ' will always be visible
ActiveCell.Comment.Text Text:="Whatever text and/or" _
  & Chr(10) & "information you may wish" & Chr(10) & _
  "to put here, e.g.: " & Chr(10) & _
  "Date: " & Date & Chr(10) & "Time: " & Time
ActiveCell.Comment.Shape.Height = 100      ' default 56
ActiveCell.Comment.Shape.Width = 200       ' default 96

End Sub
```

Many custom macros in the MacroBundle use an output cell to show the macro name, and display the corresponding cell comments when the pointer hovers over that cell. (Permanently visible comments can block adjacent cells from view.) You can display a cell comment permanently by right-clicking on the cell, and then on Show Comment, and undo this afterwards with Hide Comment.

9.4.3 Romberg trapezoidal integration *pp. 522-526*

Comparison of the results in exercises 9.4.1 and 9.4.2 illustrated that reducing the leading error term in (9.4.3) could be much more effective than reducing the step size δ. Here, then, is a general approach that, systematically and in order of relative importance, removes one error term after another, and thereby yields a very efficient yet still equidistant trapezoidal integration routine. We will first illustrate its principle with a fixed number $n+1 = 17$ of equidistant data points, so that the distance δ between them is $(x_n - x_0)/16$.

The error terms in (9.4.3) depend only on the derivatives at the extremes of the integration range, at $x = x_n$ and $x = x_0$, and on the step size δ. We therefore apply (9.4.3) twice: first for step size δ, then for step size 2δ. In the latter case we use only nine data points (i.e., eight double-width intervals instead of 16 single-width ones) for the same integration range. For step size δ we have

$$\begin{aligned} I_{16}^{(1)} &= \left(f_0 + 2f_1 + 2f_2 + \ldots + 2f_{15} + f_{16}\right)\delta/2 \\ &\quad + A\delta^2 + B\delta^4 + C\delta^6 + D\delta^8 + \ldots \end{aligned} \tag{9.4.4}$$

where we have abbreviated this first approximation of the integral with 16 intervals as $I_{16}^{(1)}$, and the δ-independent parts of the error terms as $A = -(f_{16}^{\mathrm{I}} - f_0^{\mathrm{I}})/12$, $B = +(f_{16}^{\mathrm{III}} - f_0^{\mathrm{III}})/720$, $C = -(f_{16}^{\mathrm{V}} - f_0^{\mathrm{V}})/30240$, etc. For step size 2δ we have, likewise,

$$\begin{aligned} I_8^{(1)} &= \left(f_0 + 2f_2 + 2f_4 + \ldots + 2f_{14} + f_{16}\right)\delta \\ &\quad + A(2\delta)^2 + B(2\delta)^4 + C(2\delta)^6 + D(2\delta)^8 + \ldots \\ &= \left(f_0 + 2f_2 + 2f_4 + \ldots + 2f_{14} + f_{16}\right)\delta \\ &\quad + 4A\delta^2 + 16B\delta^4 + 64C\delta^6 + 252D\delta^8 + \ldots \end{aligned} \tag{9.4.5}$$

Consequently we can eliminate the leading error term in A by subtracting (9.4.5) from four times (9.4.4), resulting in thrice the sought integral. We therefore find an improved formula for the integral as

$$\begin{aligned} I_{16}^{(2)} &= \left(4I_{16}^{(1)} - I_8^{(1)}\right)\!/3 \\ &= (f_0 + 4f_1 + 2f_2 + 4f_3 + 2f_4 + 4f_5 + 2f_6 + 4f_7 + 2f_8 \\ &\quad + 4f_9 + 2f_{10} + 4f_{11} + 2f_{12} + 4f_{13} + 2f_{14} + 4f_{15} + f_{16})\,\delta/3 \\ &\quad - 12\,B\,\delta^4 - 60\,C\,\delta^6 - 252\,D\,\delta^8 - \ldots \end{aligned} \tag{9.4.6}$$

This result, with its alternating inner terms (but usually without its Euler-Maclaurin error estimates) is often called Simpson's formula, even though it was described more than a century earlier by Cavalieri and,

subsequently, by Gregory and by Cotes, two contemporaries of Newton. In fact, Simpson's rule follows directly from the "barrel problem" solved by Kepler in 1612, long before the invention of calculus.

Of course we need not stop with (9.4.6), because we can repeat the process and eliminate the *B*-term to find the Boolean formula

$$\begin{aligned} I_{16}^{(3)} &= \left(16I_{16}^{(2)} - I_8^{(2)}\right)\!/15 \\ &= 16\times(f_0+4f_1+2f_2+4f_3+2f_4+4f_5+2f_6+4f_7+2f_8 \\ &\quad +4f_9+2f_{10}+4f_{11}+2f_{12}+4f_{13}+2f_{14}+4f_{15}+f_{16})\,\delta/45 \\ &\quad -16\times(12B\delta^4+60C\delta^6+252D\delta^8+\ldots)/15 \\ &\quad -2\times(f_0+4f_2+2f_4+4f_6+2f_8+4f_{10}+2f_{12}+4f_{14}+f_{16})\,\delta/45 \\ &\quad +(12B\times16\delta^4+60C\times64\delta^6+252D\times256\delta^8+\ldots)/15 \\ &= (14f_0+64f_1+24f_2+64f_3+28f_4+64f_5+24f_6+64f_7+28f_8 \\ &\quad +64f_9+24f_{10}+64f_{11}+28f_{12}+46f_{13}+24f_{14}+64f_{15}+14f_{16})\,\delta/45 \\ &\quad +192C\delta^6 + 4032D\delta^8 +\ldots \end{aligned} \tag{9.4.7}$$

The next stage eliminates the C-term, and results in

$$\begin{aligned} I_{16}^{(4)} &= \left(64I_{16}^{(3)} - I_8^{(3)}\right)\!/63 \\ &= (868f_0+4096f_1+1408f_2+4096f_3+1744f_4+4096f_5 \\ &\quad +1408f_6+4096f_7+1736f_8+4096f_9+1408f_{10}+4096f_{11} \\ &\quad +1744f_{12}+4096f_{13}+1408f_{14}+4096f_{15}+868f_{16})\,\delta/2835 \\ &\quad -967680D\delta^8 -\ldots \end{aligned} \tag{9.4.8}$$

By using 32, 64, 128 etc. rather than 16 intervals δ, this approach can be refined further, leading to the coefficients of *y* listed in Table 9.4.1. Figure 9.4.4 displays these sequences of Romberg coefficients, after scaling by their common denominators. Clearly, the higher-order Romberg coefficients are relatively minor variations on the (1, 4, 2, 4, … , 4, 1) / 3 scheme of (9.4.6), while improving the accuracy of the integration.

The most efficient approach, at each successive step, is to eliminate the leading error term by doubling the number *n* of segments used, i.e., by halving δ. This algorithm starts with $n = 2$, i.e., with just two segments of width $\delta = f_n - f_0$, and consequently with only $n+1$ or three data points, f_0 through f_2. At each iteration stage, the macro doubles *n* and halves δ. While the coefficients listed in Table 9.4.1 can be used as such, one more often only follows the logic that led to them, in terms of the successive integrals $I_n^{(k)}$, which can be written in a very compact code.

Romberg integration is usually the best available general-purpose algorithm for the integration of equidistant data points with model functions that are integratable and smooth (i.e., continuous, with continuous

derivatives, and without singularities) over their entire integration range, including their edges. Trapezoidal integration is only more efficient when the derivatives of the function at both integration limits are either zero (as in exercise 9.4.1) or equal (as in integrating a repetitive function over exactly one or more repeat cycles).

n	**scheme**	**coefficients**					**denominator**
2	*a* (***b***) *a*	*a* = 1	*b* = 2				2
2^2	*a b* (***c*** *b*) *a*	*a* = 1	*b* = 4	*c* = 2			3
2^3	*a b c b* (***d*** *b c b*) *a*	*a* = 14	*b* = 64	*c* = 24	*d* = 28		45
2^4	*a b c b d b c b* (***e*** *b c b d b c b*) *a*						
		a = 868	*b* = 4096	*c* = 1408	*d* = 1744	*e* = 1736	2835
2^5	*a b c b d b c b e b c b d b c b* (***f*** *b c b d b c b e b c b d b c b*) *a*						
		a = 220472	*b* = 1048576	*c* = 352256			
		d = 443648	*e* = 440928	*f* = 440944			722925
2^6	*a b c b d b c b e b c b d b c b f b c b d b c b e b c b d b c b* (***g*** *b c b d b c b e b c b d b c b f b c b d b c b e b c b d b c b*) *a*						
		a = 225322384	*b* = 1073741824	*c* = 358612992			
		d = 453591040	*e* = 450622976	*f* = 450644800			
		g = 450644768					739552275
2^7	*a b c b d b c b e b c b d b c b f b c b d b c b e b c b d b c b g b c b d b c b e b c b d b c b f b c b d b c b e b c b d b c b* (***h*** *b c b d b c b e b c b d b c b f b c b d b c b e b c b d b c b g b c b d b c b e b c b d b c b f b c b d b c b e b c b d b c b*) *a*						
		a = 922469840096	*b* = 4398046511104	*c* = 1466731331584			
		d = 1857191673856	*e* = 1844844527616	*f* = 1844939854848			
		g = 1844939680128	*h* = 1844939680192				3028466566125

Table 9.4.1: The coefficients of the Romberg trapezoid scheme for its first seven iterations. Each subsequent iteration doubles *n* and uses one additional coefficient, which is highlighted in boldface. Brackets indicate potential repeat units.

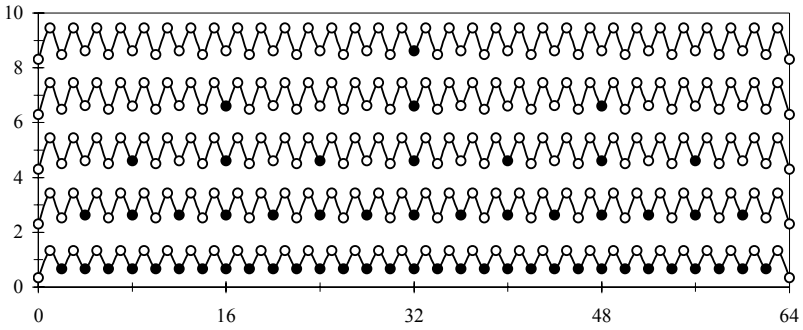

Fig. 9.4.4: The sequence of Romberg coefficients for (from bottom to top) $n = 4$, 8, 16, 32, and 64, after division by the corresponding denominator, for a data set of 65 points. For the sake of clarity, successive curves are each shifted upwards by 2. The filled circles correspond to the boldface letters in Table 9.4.1. The next-higher-order Romberg sequence, for $n = 2^7$, would require at least 129 data points.

Romberg integration is automated in the custom macro Romberg which, like Trapez, comes in two flavors: RombergAuto and RombergSpecify. In fact, the Romberg approach involves such a small modification of the Trapez macro that they are both implemented by the subroutine Integrate.

Exercise 9.4.5:

(1) We will once more visit the problem of the exercises 9.4.2 through 9.4.4, namely the integration of $\exp[-x^2]$ between $x = -1$ and $x = +1$.

(2) Somewhere on a spreadsheet place an appropriate input data block for this problem, two columns wide and at least three rows high, and highlight it.

(3) Call RombergAuto, and specify 14 significant digits in no more than 30 iterations. The answer will appear almost instantaneously, using only nine iterations.

(4) Make sure that the function Equation(x) carries the correct expression. Again highlight the input data block, and call RombergSpecify. You get the same answer, but now even faster, although the difference between a blink of an eye, and a tenth of one, will hardly matter.

(5) Verify that Romberg integration of this function between zero and infinity (in practice, the range from 0 to 8 will do) likewise yields a fast response with either RombergAuto or RombergSpecify.

The fast convergence of Romberg integration, to a higher accuracy than could be obtained with Trapez, says it all: when given a choice such as you now have with the availability of these two custom macros, Romberg integration is almost always superior to trapezoidal integration, both in terms of accuracy and speed. The only time that trapezoidal integration holds an advantage is when the function has equal derivatives at its integration limits.

The Romberg method is clearly an equidistant trapezoidal method with successively refined weights. Why does it often work where the trapezoidal method fails? Because, by using non-uniform weights, it manages to reduce a number of error sources, and therefore converges at a lower number of iterations, which don't require such a huge number of data points and therefore cause smaller round-off errors. Eventually, when an integral is sufficiently difficult, we will again run into the same problem but, in practice, Romberg integration makes a large number of integration problems amenable to machine solution that is accurate (to within 14 significant decimals, which is about all one can expect in software that follows the IEEE 754 protocol, see section 9.1) and often fast. Moreover, if ever needed, you can modify an open-source macro such as Romberg to obtain still higher accuracy by using the tools discussed in chapter 11.

When, at a particular x-value, a function cannot be evaluated because it is *indeterminate*, such as 0/0 or ∞/∞, the spreadsheet can often provide an easy numerical way to find its limiting behavior. In computing, e.g., the value of $f = (x^3-8)/(x-2)$ at $x = 2$, one obtains the error message `#DIV/0!` But calculating and plotting f as a function of x *around* $x = 2$ (at, say, $x = 1.999$, 1.999999, $2-10^{-9}$, $2-10^{-12}$, $2+10^{-12}$, $2+10^{-9}$, 2.000001, and 2.001) strongly suggests that $f = 12$ at $x = 2$, the same result that one would obtain by using the rule of de l'Hôpital. Similarly, for $f = \sin(x)/x$ at $x = 0$ we find $f = 1$, for $f = [\cos(2x)/(x-\pi/4)]^2$ at $x = \pi/4$ we get 4, etc. When you encounter such indeterminate points at the edge of the integration range, simply replace them by their correct limiting value, and proceed with Romberg. If they happen to lie inside the integration range, divide the range into subranges such that all indeterminate points are at their edges, integrate the individual subranges, and add their integrals.

While most functions that a scientist or engineer would need to integrate numerically are rather well behaved, that need not always be the case. Even a simple function such as $f(x) = 1/x$ approaches $\pm\infty$ when x approaches 0, and will give difficulties at this *singularity*. In such a case, again split the integration range into subranges so that all singularities occur only at their edges, and then use either Romberg midpoint integration (see the next section) or, when there is no need to use equidistant data, Romberg-Kahan integration (described in section 9.4.5) or Gauss-Legendre integration (not discussed here at all).

10.9.2 General information on Matrix.xla pp. 582-586

The following information was abstracted from the December 2006 version of the *Tutorial of Numerical Analysis for Matrix.xla*, and the associated *Reference Guide for Matrix.xla*. These documents (or those that came with your version) should be consulted for further details. In contrast to all Excel-provided functions and macros, the specific codes used in Volpi's matrix tools are directly accessible to the user.

Except where indicated otherwise, all instructions listed here require the user to highlight the target area first, before calling the function, and to enter the instruction with Ctrl◡Shift◡Enter. Do so even if you use the corresponding dialog box, i.e., do *not* use its OK button. When you are uncertain of the correct dimension of the result of a matrix instruction, first try it out using an ample test target area in an unused part of the spreadsheet. Any superfluous cells will exhibit the error message #N/A, except when one or both of the dimensions is 1. Note the correct block size, erase the entire test block, and redo.

For the sake of notational compactness, we will denote a square *diagonal* matrix by **D** with elements d_{ii}, a square *tridiagonal* matrix by **T** with elements t_{ij} where $| j - i | \leq 1$, most other *square* matrices by **S**, *rectangular* matrices by **R**, and all matrix elements by m_{ij}. A *vector* will be shown as **v**, with elements v_i, and a *scalar* as *s*. Particular values are denoted by *x* when real, and by *z* when complex. All optional parameters are shown in straight brackets, [ ]. All matrices, vectors, and scalars are assumed to be real, except when specified otherwise. All matrices are restricted to two dimensions, and vectors to one dimension. For manipulating matrices and vectors of higher dimensions, the spreadsheet will seldom be a suitable computational environment anyway.

With some functions, the user is given the option *Integer* of applying integer arithmetic. When a matrix only contains integer elements, selecting integer arithmetic may avoid most round-off problems. On the other hand, the range of integer arithmetic is limited, so that overflow errors may result if the matrix is large and/or contains large numbers.

Another common option is *Tiny*, which defines the absolute value of quantities that can be regarded as most likely resulting from round-off errors, and are therefore set to zero. When not activated, the routine will use its default value of 10^{-15}.

The various instructions listed in appendix B are grouped by functionality, a necessarily somewhat arbitrary arrangement. We therefore show here, in Table 10.9.1, an alphabetical listing of the 115 functions

available in Matrix.xla 2.3.2, the latest version as of this writing (March 2007), with a one-line function description, so that the reader can get a rough idea of the richness of this instruction set.

function	**brief description**
GJstep	Step-by-step tracing of Gauss-Jordan algorithm
MAbs	absolute value of vector **v**
MAbsC	absolute value of complex vector **v**
MAdd	add two matrices: $\mathbf{R}_1 + \mathbf{R}_2$
MAddC	add two complex matrices: $\mathbf{C}_1 + \mathbf{C}_2$
MAdm	Create an admittance matrix from a branch matrix
MBAB	similarity transformation $\mathbf{S}_1^{-1}\,\mathbf{S}_2\,\mathbf{S}_1$
MBlock	transform reducible sparse **S** into block-partitioned form
MBlockPerm	permute matrix for MBlock
MChar	compute characteristic matrix of **S** at x
MCharC	compute characteristic matrix of **C** at z
MCharPoly	compute characteristic polynomial of **S**
MCharPolyC	compute characteristic polynomial of **C**
MCholesky	Cholesky decomposition
MCmp	companion matrix of monic polynomial
MCond	condition number κ of **R**
MCorr	correlation matrix
MCovar	covariance matrix
MCplx	convert two real matrices into one complex matrix
MDet	determinant of **S**
MDet3	determinant of **T** in $n\times3$ format
MDetC	determinant of a complex square matrix
MDetPar	determinant of **S** containing symbolic parameter k
MDiag	convert vector **v** into diagonal matrix **D**
MDiagExtr	extract the diagonal of **D**
MEigenSortJacobi	sort eigenvectors by absolute value of eigenvalues
MEigenvalJacobi	Jacobi sequence of orthogonality transforms
MEigenvalMax	find maximum absolute value of **S**
MEigenvalPow	eigenvalues of **S** by power method
MEigenvalQL	eigenvalues of **T** by QL algorithm
MEigenvalQR	eigenvalues of **S** by QR decomposition
MEigenvalQRC	eigenvalues of **C** by QR decomposition
MEigenvalTTpz	eigenvalues of tridiagonal Toeplitz matrix
MEigenvec	eigenvectors of **S** for given eigenvalues
MEigenvecC	eigenvectors of **C** for given eigenvalues
MEigenvecInv	eigenvectors of eigenvalue by inverse iteration
MEigenvecInvC	complex eigenvectors of eigenvalue by inverse iteration
MEigenvecJacobi	orthogonal similarity transform of symmetric matrix **S**
MEigenvecMax	eigenvector for dominant eigenvalue
MEigenvecPow	eigenvector of **S** by power method
MEigenvecT	eigenvectors for given eigenvalues of **T**
MExp	matrix exponential, $e^{\mathbf{S}}$
MExpErr	error term in $e^{\mathbf{S}}$

MExtract	extract submatrix of **R**
MHessenberg	convert **S** into Hessenberg form
MHilbert	create Hilbert matrix
MHilbertInv	create inverse Hilbert matrix
MHouseholder	create Householder matrix
MIde	generate an identity matrix
MInv	invert **S**
MInvC	invert a complex square matrix
MLeontInv	invert the Leontief matrix
MLU	LU decomposition
MMopUp	eliminate round-off errors in **R**
MMult	Excel's matrix multiplication, $\mathbf{R}_1\ \mathbf{R}_2$
MMult3	multiply **T** and **R**
MMultC	Multiply two complex matrices: $\mathbf{C}_1\ \mathbf{C}_2$
MMultS	multiply a matrix and a scalar
MMultSC	multiply a complex matrix and a scalar
MMultTpz	multiply a Toeplitz matrix and a vector
MNorm	find matrix or vector norm
MNormalize	normalize **R**
MNormalizeC	normalize **C**
MOrthoGS	modified Gram-Schmidt orthogonalization
MpCond	$-\log_{10}$ of the condition number κ
MPerm	generate permutation matrix from permutation vector
MPow	integer power of square matrix: $\mathbf{S}^n$
MPowC	integer power of a complex square matrix
MProd	product of two or more matrices: $\mathbf{R}_1\ \mathbf{R}_2\ \mathbf{R}_3$ ...
MPseudoinv	pseudo-inverse of **R**
MQH	decomposition of **S** with vector **v**
MQR	QR decomposition
MQRIter	eigenvalues of **S** by iterative QR diagonalization
MRank	rank of **R**
MRnd	generate a random matrix **R**
MRndEig	generate a random matrix with given eigenvalues
MRndEigSym	generate a symmetrical matrix with given eigenvalues
MRndRank	generate a random matrix of given rank or determinant
MRndSym	generate a symmetrical matrix of given rank or determinant
MRot	orthogonal rotation matrix
MRotJacobi	Jacobi orthogonal rotation of symmetrical matrix **S**
MSub	subtract two matrices, $\mathbf{R}_1 - \mathbf{R}_2$
MSubC	subtract two complex matrices: $\mathbf{C}_1 - \mathbf{C}_2$
MT	transpose, $\mathbf{R}^T$
MTartaglia	create Tartaglia (or Pascal) matrix
MTC	transpose a complex matrix
MTH	form complex conjugate **C*** of a complex matrix
MTrace	trace of **S**
MVandermonde	create Vandermonde matrix
PathFloyd	Floyd sequential algorithm for shortest-path pairs
PathMin	vectors of shortest-path pairs between given sites
PolyRoots	find roots of a polynomial

PolyRootsQR	find roots of a polynomial using QR algorithm
PolyRootsQRC	find roots of a complex vector using QR algorithm
ProdScal	scalar product of two vectors, $\mathbf{v}_1 \bullet \mathbf{v}_2$
ProdScalC	scalar product of complex vectors
ProdVect	vector product of two vectors, $\mathbf{v}_1 \bullet \mathbf{v}_2$
RegrCir	least squares fit to a circle
RegrL	linear least squares fit based on singular value decomposition
RegrP	RegrL for a polynomial fit
Simplex	simplex optimization
SVDD	yields **D** of singular value decomposition
SVDU	yields **U** of singular value decomposition
SVDV	yields **V** of singular value decomposition
SysLin	Gauss-Jordan linear system solver
SysLin3	tridiagonal linear system solver
SysLinC	Gauss-Jordan linear solver of complex system
SysLinIterG	iterative Gauss-Seidel linear system solver
SysLinIterJ	iterative Jacobi linear system solver
SysLinSing	singular linear system solver
SysLinT	triangular linear system solver
SysLinTpz	solve Toeplitz system by Levinson's method
TraLin	linear transformation
VarimaxIndex	varimax index for given factor loading matrix
VarimaxRot	orthogonal rotation of factor loading matrix
VectAngle	angle between two vectors

Table 10.9.1: List of functions in Matrix.xla version 2.3. For a more detailed description see appendix B and, especially, Volpi's Reference Guide for Matrix.xla and his Tutorial on Numerical Analysis with Matrix.xla.

11.7 The error function *pp. 619-620*

The error function

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \tag{11.7.1}$$

is another example of sloppy Excel programming. The error function is basic to both engineering and statistics, and appears in many problems of heat and mass transport that describe 'random walk' processes, such as heat conduction and molecular diffusion. As its name indicates, the error function is also related to the cumulative Gaussian ('normal') error distribution curve

$$\frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{x} e^{-(t-\mu)^2/2\sigma^2} dt = \frac{1}{2}\left[1 + \mathrm{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right)\right] \tag{11.7.2}$$

with mean μ and standard deviation σ.

Excel's error function takes two forms, one following the common definition (11.7.1), the other the rather unusual, two-parameter form

$$\mathrm{erf}(a,b) = \frac{2}{\sqrt{\pi}} \int_a^b e^{-t^2} dt = \mathrm{erf}(b) - \mathrm{erf}(a) \tag{11.7.3}$$

which we will not consider here.

For negative values of x, we have the simple symmetry relation

$$\mathrm{erf}(-x) = -\mathrm{erf}(x) \tag{11.7.4}$$

while for positive values of x there is a straightforward, nonalternating power series

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} e^{-x^2} \sum_{n=0}^{\infty} \frac{2^n x^{2n+1}}{1 \cdot 3 \cdot \ldots \cdot (2n+1)} \tag{11.7.5}$$

For $x > 6$, we have erf(x) = 1 to at least 15 figures, so that the function need not be computed. A VBA code reflecting these properties is shown below. Figure 11.7.1 shows the errors of the Excel function Erf(x) and of our cErf(x), and speaks for itself. Again, the results were checked against the same function computed with much longer numberlength.

```
Function cErf(X)

' Based on Abramowitz & Stegun eq.(7.1.6)

Dim n As Integer
Dim Factor, Term, Y

If X < -6 Then
  Y = -1
  GoTo A
ElseIf X > 6 Then
```

```
  Y = 1
  GoTo A
Else
  Term = X * Exp(-X * X)
  Y = Term
  n = 1
  Do
    Factor = 2 * X * X / (2 * n + 1)
    Term = Term * Factor
    Y = Y + Term
    n = n + 1
  Loop Until Abs(Term / Y) < 1E-50
  Y = 2 * Y / Sqr([Pi()])
End If

A:
cErf = Y

End Function
```

Additional cFunctions are available from my website http://www.bowdoin.edu/~rdelevie/excellaneous, where they can be found in the file cFunctions. The basic ingredient to verify such corrected functions is the availability of high-accuracy reference data as can be found in tables, obtained with other software or, most conveniently, computed with higher-accuracy routines that can be grafted onto Excel. Much of the remainder of this chapter will be devoted to such software, and the file cFunctions includes the codes used for validating these functions.

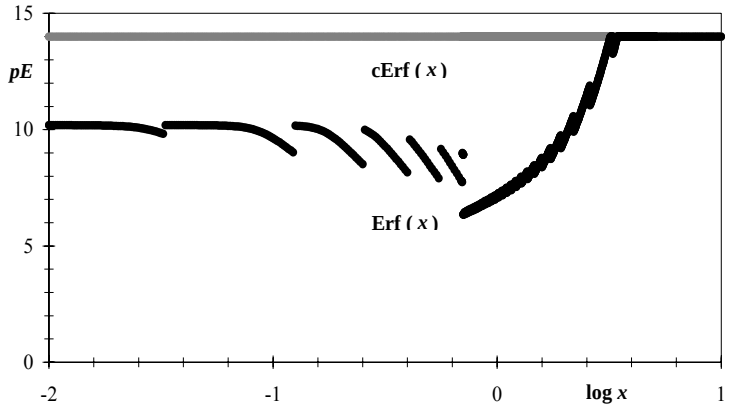

Fig. 11.7.1: Error plots of Erf(x) (black) and its corrected version cErf(x) (gray). For $x < 0$, Erf(x) gives no result, for $0 < x \lesssim 10^{-2}$ it is good to only 10 decimal figures (apparently because of an error in the value of $2/\sqrt{\pi}$ used), around $x = 0.707$ its accuracy dips to pE = 6.3, and for $x > 1.57\times10^{162}$ it again fails to give a result. Moreover, as shown here, the curve of Erf(x) vs. x shows a number of spurious discontinuities. By contrast, the function cErf is smooth and accurate to at least 14 significant decimal digits over the entire range of possible input values, $2.2\times10^{-308} \le |x| \le 1.8\times10^{308}$.

11.15 Overview of Xnumber rules pp. 644-645

Finally, at the risk of being repetitive, here is a summary of the rules you should follow when modifying old macros to use Xnumbers. If you create a new macro, I suggest that you still write it first in regular VBA, debug it, and only then convert it into Xnumbers.

(1) Make a copy of the macro, and give it a distinct name, such as one starting with an x. Save the unmodified original. High-precision macros should not be used as routine updates of general-purpose macros, since they are too slow, even when set to the standard double precision of 15 decimals. This is so because much of the extra execution time is up front, regardless of the precision specified.

(2) Before the dimensions are listed, insert the lines

```
Dim MP As Xnumbers
Set MP = New Xnumbers
With MP
```

where MP stands for multi-precision, and DgtMax is the default (quadruple) precision 30. If you want a precision different from its default value 30, specify that just below the `With MP` statement, as with

```
.DigitsMax = 50
```

or, if you so prefer, with an input box that lets you specify the value of DigitsMax from the spreadsheet.

(3) Just before the end of the subroutine, insert the lines

```
End With
Set MP = Nothing
```

(4) If the original code uses the With … End With construction, remove it or, if that is simpler, specify MP with every extended numberlength instruction. There can be only one With … End With statement operating at any place in the program, but you can use With MP in the computation section, and With something else in the output part, as long as the first With statement has been terminated before the second is activated.

(5) In the dimension statements, remove the dimension specifiers `As Single` and `As Double` from any parameter you want to use in extended numberlength, because these dimension statements will otherwise overrule Xnumbers and keep the parameter in their dimension-specified precisions. By leaving their dimensions unspecified (but of course included in the list, otherwise Option Explicit would complain), these parameters will be assumed to be dimensioned `As Variant`, which is how Xnumbers can use them.

(6) Modify all precision-critical instructions to fit the Xnumber format. All x-modified instructions should start with `MP.x`, or with `.x` when you use the With … End With construction. (But first make sure that there are no other With … End With loops in the program.) For complicated expressions, either break up the expression into several simpler parts, or use the `.x_Eval` function.

By comparing the macros in the xMacroBundle with their parents in the MacroBundle you will see that their conversion to extended number-length usually does not involve very much work, because you need not rethink the structure of your code, but merely modify some of its instructions. Moreover, all input and output statements and their formatting remain unchanged. For more specifics, including additional functions and more details, see the Xnumbers_DLL_ Ref manual.

*****

In summary, with Xnumbers.dll and a few relatively minor modifications of your macros and subroutines, can have high-precision computations while the structure of these routines, as well as your data input and output formats, remain unchanged. This can provide a very useful back-up for any custom macros, as it allows you to check the accuracy of your code, and can help you identify areas of code where it is worthwhile to pay extra attention to algorithmic accuracy. Some of the macros of the MacroBundle have already spawned x-versions which you can find in the xMacroBundle. You are of course welcome to try them out. Please let me know by e-mail, at rdelevie@bowdoin.edu, if you encounter any problems with them.