MATH 2P20
NUMERICAL ANALYSIS I
Lecture Notes

© Jan Vrbik

Contents

Chapter 1 PREVIEW

The course topics will concentrate on the following three areas:

Fitting polynomials to:

Discrete data

(either computed or empirical, and collected in a *table* of x and y values).

This may be done for several different reasons. We may want to

1. accurately *interpolate* (compute y using a value of x not found in the table itself).

2. draw a smooth picture connecting all data points,

3. fit a simple curve (linear, quadratic) to empirical (not so accurate) data. The curve be 'as close as possible' to the individual data points - we will have to agree on some overall criterion.

More complicated mathematical functions

over a specific range of x values. Similarly to the previous case. we cannot do this exactly, but have to minimize (in some well defined sense) the error of the fit.

There are several reasons for doing this:

1. Polynomials are easy to evaluate (we just add/subtract and multiply - and ultimately, all numerical computation has to be reduced to these)

2. they are also easy to integrate and differentiate - we may thus substitute our fitted polynomial for the actual function (which may be very hard or even impossible to integrate).

To facilitate the procedure (of fitting polynomials to a function), several sets of ORTHOGONAL POLYNOMIALS are introduced (e.g. Legendre, Chebyshev, Hermite, etc.).

Numerical Integration and Differentiation

Here the objective is clear; we know that many functions are impossible to integrate analytically, so we want to have an accurate way of doing this numerically. We would also like to have some idea and control over the accuracy of the results.

Integration

The way how we can numerically evaluate $\int\limits_A^B y(x)\,dx$ is to choose a set of x values (so called NODES) in the $[A,\ B]$ interval, for each (say x_i, $i = 0,\ 1,\ 2,\ ...,\ n$) of these compute the corresponding $y_i \equiv y(x_i)$. We then have to develop a formula for combining these y_i values to accurately estimate the integral (the area between the $y(x)$ function and the x axis).

A part of the problem is clearly the choice of the nodes. There are two distinct ways of approaching it:

1. A sensible (but in a sense arbitrary) choice of n equidistant values (effectively subdividing $[A,\ B]$ into n equal-length subintervals, leading to two basic 'rules' of integration, TRAPEZOIDAL and SIMPSON, to be studied in detail.
2. A choice of n points which is, in a certain sense, *optimal* (we can define 'optimal' only when we have a better understanding of the issues).

Differentiation

similarly involves estimating the value of $y'(x)$, $y''(x)$, etc. at $x = x_0$. This can be done by computing $y(x)$ at x_0 and a few extra values of x in the neighborhood of x_0 (this time, we will almost always choose them equidistant), and plugging them into the corresponding formula (which, of course, will be our task to *develop*). The major problem facing us here will the ROUND-OFF ERROR.

Related issues

The formulas for numerical differentiation can also be used (this is in fact their major application) to solve, numerically, various types of ORDINARY and partial DIFFERENTIAL EQUATIONS. We will deal with some examples of the ODE variety only (BOUNDARY-VALUE problem). In this context, we will also have to learn solving NONLINEAR (regular) equations.

Matrix Algebra

The basic problem is to solve n linear equations for n unknowns, i.e. $\mathbb{A}\mathbf{x} = \mathbf{r}$, where $\mathbb{A}$ is an n by n (square) matrix, $\mathbf{x}$ is the (column) vector of the n unknowns, and $\mathbf{r}$ is similarly a vector of the right hand side values. The simplest technique uses the so called GAUSSIAN ELIMINATION and BACKWARD SUBSTITUTION. One can reduce the round-off error by adding an extra step (row interchange) called PIVOTING.

In some (important) special cases (related to our treatment of differential equations) $\mathbb{A}$ is tridiagonal (only the elements on or adjacent to the main-diagonal are non-zero). It then becomes more efficient to use a different approach, namely a so called $\mathbb{LU}$ DECOMPOSITION of matrix $\mathbb{A}$.

We will then employ some of these techniques to learn how to solve, ITERATIVELY, n *non*-linear equations for n unknowns, by NEWTON'S METHOD (we will start with a single equation for one unknown).

Eigenvalues and eigenvectors

of square matrices are defined by

$$\mathbb{A}\mathbf{x} = \lambda\mathbf{x}$$

where $\mathbf{x}$ (non-zero) is an eigenvector and λ an eigenvalue.

To simplify the issue, we will assume that $\mathbb{A}$ is SYMMETRIC (a fairly important class of matrices), which implies that both eigenvalues and eigenvectors must be *real* (they could be *complex* in general). We will then learn how to find them, one by one (there is n of them in general), by first utilizing HOUSHOLDER'S METHOD to reduce $\mathbb{A}$ to a tridiagonal matrix, and then the applying, repeatedly, the so called $\mathbb{QL}$ algorithm to extract the smallest eigenvalue. The resulting matrix is then DEFLATED and the process repeated till all eigenvalues are found.

Remaining Topics

There is a number of important topics which we will not have time to discuss in this brief course, namely:

1. Solving ordinary differential equations (INITIAL-VALUE problem).
2. Solving partial differential equations.
3. Optimizing a function of several variables (finding its largest or smallest value).

Chapter 2 USING MAPLE

Basics

Typing an EXPRESSION (following Maple's > prompt) results in evaluating it. When the expression contains only integers (no decimal point), one gets the exact (rational) answer, as soon as at least one number in the expression is real (with a decimal point), the result is real (rounded off to 10 significant digits). The symbols $*$, / and ^ facilitate multiplication, division and exponentiation, respectively. Note that each line of your input has to end with a semicolon:

$> 4 * 5 - 3\,/\,(5 + 2) + 2\,\hat{}\,(-3)\,;$

$$\frac{1103}{56}$$

The result of any computation can be stored under a name (which you make up, rather arbitrarily), and used in any subsequent expression. Maple then remembers the value, until the end of your session, or till you deliberately replace it with a new value. Note that this (giving a name to a result) is achieved by typing the name, followed by a colon and the equal sign (a group of two symbols, representing a single operation), followed by the actual expression to be stored:

$>\ a := (3.0 + 4) * (2 - 6) + 2\,/\,3 - 4\,/\,5\,;$

$$a := -28.13333\,333$$

$>\ a\,/\,7 + 9\,;$

$$4.98095\,238$$

$>\ a := 14\,/\,6\,;$

$$a := \tfrac{7}{3}\,;$$

$>\ a\,/\,7 + 9\,;$

$$a := \tfrac{28}{3}\,;$$

(from now on, we will omit the > prompt from our examples, showing only what *we* have to type).

Maple can also handle the usual functions such as **sin, cos, tan, arcsin, arccos, arctan, exp, ln, sqrt**, etc. All angles are always measured in radians.

sin(3.) ; **sqrt**(8) ;

$$.14112\,00081$$

$$2\sqrt{2}$$

We can also **define** our own functions by:

$f := x - > x\,\hat{}\,2\,;$

$$f := x \to x^2$$

$f(3);$

$$9$$

where f is an arbitrary name.

Lists and Loops

Maple can store, under a single name, a whole LIST of values, thus:

$a := [3\,/\,2,\ 5,\ \mathbf{sqrt}\,(3),\ 7]\,;$

$$a := [\tfrac{3}{2},\ 5,\ \sqrt{3},\ 7]$$

The individual elements of a list can be referenced by indexing (and used in computing another expression):

$a[2] * 4\,;$

$$20$$

One can add elements of a list by the following COMMAND (as Maple calls them):

$\mathbf{sum}('a[i]','i'= 1..4)\,;$

$$\tfrac{27}{2} + \sqrt{3}$$

One can convert the last answer to its decimal form by:

$\mathbf{evalf}(\%)\,;$

$$15.23205\,081$$

Note that the % symbol always refers to the previous expression.

Similarly to **sum**, one can also compute **product** of elements of a list.

To subtract say 3 from each element of the list a, redefining a correspondingly, can be achieved by:

for i **from** 1 **to** 4 **do** $a[i] := a[i] - 3$ **end do**:

Note that terminating a statement by : instead of the usual ; will prevent Maple from printing the four results computed in the process (we may not need to see them individually). Also note that, upon completion of this statement, i will have the value of 5 (any information i had contained previously will have been destroyed)!

We can easily verify that the individual elements of our a list have been updated accordingly:

$a[2]\,;$

$$2$$

We may also create a list using the following approach:

$b := [\,\mathbf{seq}\,(2\ \hat{}\ i, i = 1..6)]\,;$

$$b := [2,\ 4,\ 8,\ 16,\ 32,\ 64\,]\,;$$

Variables and Polynomials

If a symbol, such as for example x, has not been assigned a specific value, Maple considers it a variable. We may then define a to be a POLYNOMIAL in x, thus:

$a := 3 - 2 * x + 4 * x\hat{}2\,;$

$$a := 3 - 2\,x + 4\,x^2$$

A polynomial can be differentiated

diff$(\,a,\,x);$

$$-2 + 8\,x$$

integrated from, say, 0 to 3

int$(\,a, x = 0\,..\,3)\,;$

$$36$$

or plotted, for a certain range of x values

plot$(\,a, x = 0..3)\,;$

We can also evaluate it, substituting a specific number for x (there are actually two ways of doing this):

subs$(x = 3, a)\,;$ **eval**$(a,\ x = 3);$

$$33$$

$$33$$

We can also multiply two polynomials (in our example, we will multiply a by itself), but to convert to a regular polynomial form, we nee to **expand** the answer:

$a * a\,;$ **expand**$(\%)\,;$

$$(3 - 2\,x + 4\,x^2)^2$$

$$9 - 12\,x + 28\,x^2 - 16\,x^3 + 16\,x^4$$

Procedures

If some specific computation (consisting, potentially, of several steps) is to be done, more than once (e.g. we would like to be able to raise each element of a list of values to a given power), we need first to design the corresponding PROCEDURE (effectively a simple computer program), for example:

$RAISETO := \mathbf{proc}(L,\, N)\,;\ \mathbf{local}\ K,\, n,\, i\,;\ K := L\,;\ n := \mathbf{nops}(L)\,;$

$\mathbf{for}\ i\ \mathbf{from}\ 1\ \mathbf{to}\ n\ \mathbf{do}\ K[i] := K[i]\,\hat{}\,N\ \mathbf{end\ do}\,;\ K\ \mathbf{end\ proc}:$

where $RAISETO$ is an arbitrary name of the procedure, L and N are arbitrary names of its ARGUMENTS (also called parameters), the first for the list and the second for the exponent, K, n and i are auxiliary names to be used in the actual computation (since they are **local**, they will not interfere with any such names used outside the procedure). First we copy L into K (Maple does not like it if we try to modify L directly) and find its length n (by the **nops** command). Then, we raise each element K to the power of N, and return (the last expression of the procedure) the modified list. We can organize the procedure into several lines by using Shift-Enter (to move to the next line).

We can then use the procedure as follows:

$SVFL([2,\, 5,\, 7,\, 1],\, 2);\ SVFL([3,\, 8,\, 4],\, -1)\,;$

$$[4,\, 25,\, 49,\, 1]$$

$$[\tfrac{1}{3},\, \tfrac{1}{8},\, \tfrac{1}{4}]$$

Matrix Algebra

We can define a matrix by:

$a := \mathbf{matrix}(2,\, 2,\, [1,\, 2,\, 3,\, 4]):$

where 2, 2 specifies its dimensions (number of rows and columns, respectively), followed by the list of its elements (row-wise).

We multiply two matrices (here, we multiply a by itself) by

$\mathbf{evalm}(a\ \&{*}\ a):$

Note that we have to replace the usual $*$ by $\&*$. Similarly, we can add and subtract (using $+$ and $-$), and raise a to any positive integer power (using ˆ).

We can also multiply a by a vector (of matching length), which can be entered as a list:

$\mathbf{evalm}(a\ \&{*}\ [2,\, 5]):$

Note that reversing the order of a and $[2, 5]$ yields a different answer.

To compute the **transpose** and **inverse** of a, we must first ask Maple to make these commands available by:

with(linalg) :

We can then perform the required operation by

transpose(a) :

etc.

Similarly, to solve a set of linear equation with a being the matrix of coefficients and [2, 3] the right hand side vector, we do:

linsolve(a, [2, 3]) :

Other useful commands:

a := **randmatrix** (5, 5) :

creates a matrix of specified dimensions with random elements,

augment(a, [6, 2, 7, 1, 0]) :

attaches the list, making it an extra (last) column of a,

submatrix(a, 2..4, 1..2) :

reduces a to a 3 by 2 submatrix, keeping only rows 2, 3 and 4, and columns 1 and 2,

swaprow(a, 2, 5) :

interchanges rows 2 and 5 of a,

addrow(a, 2, 4, 2/3) :

adds row 2 multiplied by $\frac{2}{3}$ to row 4 of a.

To recall the proper syntax of a command, one can always type:

? **addrow**

to get its whole-page description, usually with examples.

Plots

Plotting a specific function (or several functions) is easy (as we have already seen):

plot($\{\sin(x), x - x\hat{\ }3/6\}, x = 0..Pi/2)$:

One can also plot a scattergram of individual points (it is first necessary to ask Maple to make to corresponding routine available, as follows:

with(plots) :

pointplot($[[0, 2], [1, -3], [3, 0], [4, 1], [7, -2]]$);

Note that the argument was a *list* of pairs of x-y values (each pair itself enclosed in brackets).

We can combine any two such plots (usually a scattergram of points together with a fitted polynomial) by:

$pic1 := \textbf{pointplot}(\,[\textbf{seq}(\,[i/5,\,\sin(i/5)], i = 1..7)]\,):$

$pic2 := \textbf{plot}(\sin(x), x = 0..1.5):$

$\textbf{display}(pic1, pic2):$

To plot a function defined in a piecewise manner, we first define it by, say

$f := \textbf{piecewise}(x < 0, -x\hat{\ }2,\ x < 4, x\hat{\ }2,\ 4 * x):$

(note that $4 * x$ will be used 'otherwise', i.e. for all x values bigger than 4), and then use the regular

$\textbf{plot}(f, x = -5..10);$

Chapter 3 INTERPOLATING POLYNOMIALS

In this chapter, we will construct polynomials to fit (which, in this chapter, means to pass exactly through each point) a discrete set of data, such as, for example

x:	0	1	3	4	7
y:	2	–3	0	1	–2

or that generated by evaluating a specific function, say $y = \sin x$, at $x = 0$, 0.1, 0.2, ... 1.0.

There are two standard techniques for achieving this (they both result in the same polynomial), which we discuss in the next two sections.

Newton's Interpolation

The first thing to realize when fitting a polynomial (exactly) to discrete data is that the polynomial must have as many parameters (i.e. coefficients) we are free to vary as there are conditions (i.e. points) in our data set.

Thus, for example, to run a polynomial through all points of the above table, it needs to have 5 coefficients (i.e. degree 4), such that

$$c_0 + c_1 x_i + c_2 x_i^2 + c_3 x_i^3 + c_4 x_i^4 = y_i$$

where $i = 1, 2, \ldots 5$. The resulting 5 equations for 5 unknowns are *linear*, having a unique solution (unless there are two or more identical values in the x row). Later on we learn how to solve such systems of linear equations, but the general technique is quite lengthy and tedious.

Fortunately, due to the rather special from of these equations, there is an easier and faster way of getting the answers. It works as follows:

We fit the first point only by a zero-degree polynomial (a constant),

we extend this to a linear fit through the first two points,

quadratic fit through the first three points,

..., until we reach the end of the data.

Each step of the procedure can be arranged to involve only one unknown parameter, which we can easily solve for.

Example: To fit the data of the original table, we start with the constant 2 (the first of the y values). Then, we add to it $c(x-0)$, where c is a constant to yield, for the complete expression, -3 (the second y value), i.e. $2 + c = -3 \Rightarrow c = -5$. Our solution so far is thus $2 - 5x$. Now, we add to it $c(x-0)(x-1)$ so that the total expression has, at $x = 3$, the value of 0, i.e. $2 - 5 \times 3 + c \times 3 \times 2 = 0 \Rightarrow c = \frac{13}{6}$. This results in $2 - 5x + \frac{13}{6}x(x-1)$. Add $c(x-0)(x-1)(x-3)$ and make the result equal to 1 at $x = 4 \Rightarrow c = -\frac{7}{12}$.

Finally, add $c(x-0)(x-1)(x-3)(x-4)$ and make the total equal to -2 at $x = 7 \Rightarrow \frac{19}{252}$. Thus, the final answer is

$$2 - 5x + \frac{13}{6}x(x-1) - \frac{7}{12}x(x-1)(x-3) + \frac{19}{252}x(x-1)(x-3)(x-4)$$

Expanding (quite time consuming if done 'by hand') simplifies this to

$$2 - \frac{275}{28}x + \frac{1495}{252}x^2 - \frac{299}{252}x^3 + \frac{19}{252}x^4$$

One can easily verify that this polynomial passes, exactly, through all five points.

Computing the c values is made easier by utilizing the following scheme:

$$\begin{array}{cccccc}
0 & \boxed{2} & & & & \\
 & & \frac{-3-2}{1-0} = \boxed{-5} & & & \\
1 & -3 & & \frac{\frac{3}{3}-(-5)}{3-0} = \boxed{\frac{13}{6}} & & \\
 & & \frac{0-(-3)}{3-1} = \frac{3}{2} & & \frac{-\frac{1}{6}-\frac{13}{6}}{4-0} = \boxed{-\frac{7}{12}} & \\
3 & 0 & & \frac{1-\frac{3}{2}}{4-1} = -\frac{1}{6} & & \frac{-\frac{1}{18}-(-\frac{7}{12})}{7-0} = \boxed{\frac{19}{252}} \\
 & & \frac{1-0}{4-3} = 1 & & \frac{-\frac{1}{2}-(-\frac{1}{6})}{7-1} = -\frac{1}{18} & \\
4 & 1 & & \frac{-1-1}{7-3} = -\frac{1}{2} & & \\
 & & \frac{-2-1}{7-4} = -1 & & & \\
7 & -2 & & & &
\end{array}$$

Lagrange interpolation

This is a somehow different approach to the same problem (yielding, in the end, the same solution). The main idea is: Having a set of n pairs of x-y values, it is relatively easy to construct an $n-1$ degree polynomial whose value is 1 at $x = x_1$ and 0 at $x = x_2$, $x = x_3$, ... $x = x_n$, as follows

$$P_1(x) = \frac{(x-x_2)(x-x_3)....(x-x_n)}{(x_1-x_2)(x_1-x_3)....(x_1-x_n)}$$

where the denominator is simply the value of the numerator at $x = x_1$. Similarly, one can construct

$$P_2(x) = \frac{(x-x_1)(x-x_3)....(x-x_n)}{(x_2-x_1)(x_2-x_3)....(x_2-x_n)}$$

whose values are 0, 1, 0, ... 0 at $x = x_1,\ x_2,\ x_3,\ ....\ x_n$ respectively, etc.

Having these, we can then combine them into a single polynomial (still of degree $n-1$) by

$$y_1P_1(x) + y_2P_2(x) + ... + y_nP_n(x)$$

which, at $x = x_1, x_2, ...$ clearly has the value of $y_1, y_2, ...$ and thus passes through all points of our data set.

Example: Using the same data as before, we can now write the answer almost immediately

$$2\frac{(x-1)(x-3)(x-4)(x-7)}{(-1)(-3)(-4)(-7)} - 3\frac{x(x-3)(x-4)(x-7)}{1(-2)(-3)(-6)} +$$
$$\frac{x(x-1)(x-3)(x-7)}{4\times 3\times 1\times(-3)} - 2\frac{x(x-1)(x-3)(x-4)}{7\times 6\times 4\times 3}$$

which expands to the same.

Based on this fit, we can now INTERPOLATE (i.e. evaluate the polynomial) using any value of x within the bounds of the tabulated x's (taking an x outside the table is called EXTRAPOLATING). For example, at $x = 6$ the polynomial yields $y = \frac{1}{63} = 0.015873$. Since interpolation was the original reason for constructing these polynomials, they are called INTERPOLATING POLYNOMIALS. We will need them mainly for developing formulas for numerical differentiation and integration.

To conclude the section, we present another **example**, in which the y values are computed based on the $\sin x$ function, using $x = 60$, 70, 80 and 90 (in degrees), i.e.

x	$\sin x^o$
60	0.8660
70	0.9397
80	0.9848
90	1.0000

The corresponding Lagrange interpolating polynomial is

$$0.866\frac{(x-70)(x-80)(x-90)}{-6000} + 0.9397\frac{(x-60)(x-80)(x-90)}{2000} +$$
$$0.9848\frac{(x-60)(x-70)(x-90)}{-2000} + \frac{(x-60)(x-70)(x-80)}{6000} =$$
$$-0.10400 + 2.2797\times 10^{-2}x - 9.7500\times 10^{-5}x^2 - 2.1667\times 10^{-7}x^3$$

Plotting the difference between this polynomial and the $\sin x^o$ function reveals that the largest error of such and approximation throughout the 60 to 90 degree range is be about 0.00005. This would be quite sufficient when only a four digit accuracy is desired. Note that trying to extrapolate (go outside the 60-90 range) would yield increasingly inaccurate results.

This example should give us a rough idea as to how calculators (and computers) evaluate various functions - they replace it by evaluating close-fitting polynomials (with the advantage of having to add, subtract and multiply only a handful of times).

Chapter 4 CUBIC SPLINE

Suppose we would like to join n points by a *smooth* curve passing through them all, to make a nice visual presentation of our data. Why not use an interpolating polynomial? Because these, to reach all data points, tend to have many 'unnatural' peaks and dips. For example, fitting

x:	4	9	12	16	22
y:	157	41	145	92	7

we get the results of Figure 1, clearly 'overshooting' the original data.

The proper solution should be a sufficiently smooth, continuous function of x, say $f(x)$ (let us define 'smooth' as having both f' and f'' also continuous), with the minimal amount of curvature. Since curvature can be measured by $(f'')^2$, we try to minimize its total amount, integrated over the full data span, namely

$$\int_{x_1}^{x_n} \left[f''(x)\right]^2 dx$$

The solution, let us call if $S(x)$, is a collection of cubic polynomials (one of each subinterval or SEGMENT), tied to each other in a smooth and continuous manner, with the following two extra conditions, imposed at the beginning and at the end of the x range:

$$f''(x_1) = f''(x_n) = 0 \tag{4.1}$$

Proof: Let $f(x)$ be any other smooth function passing through the n points. We define $g(x) \equiv S(x) - f(x)$. Clearly, $g(x)$ is equal to zero at each x_i (let us call them NODES). We need to prove that

$$\int_{x_1}^{x_n} \left[f''(x)\right]^2 dx = \int_{x_1}^{x_n} \left[S''(x)\right]^2 dx + \int_{x_1}^{x_n} \left[g''(x)\right]^2 dx - 2\int_{x_1}^{x_n} g''(x)S''(x)\, dx \geq \int_{x_1}^{x_n} \left[S''(x)\right]^2 dx$$

Since

$$\int_{x_1}^{x_n} \left[g''(x)\right]^2 dx \geq 0$$

we will be done if we can show that

$$\int_{x_1}^{x_n} g''(x)S''(x)\, dx = 0$$

Utilizing by-part integration, we can write

$$\int_{x_1}^{x_n} g''(x)S''(x)\, dx = g'(x)S''(x)\big|_{x_1}^{x_n} - \int_{x_1}^{x_n} g'(x)S'''(x)\, dx$$

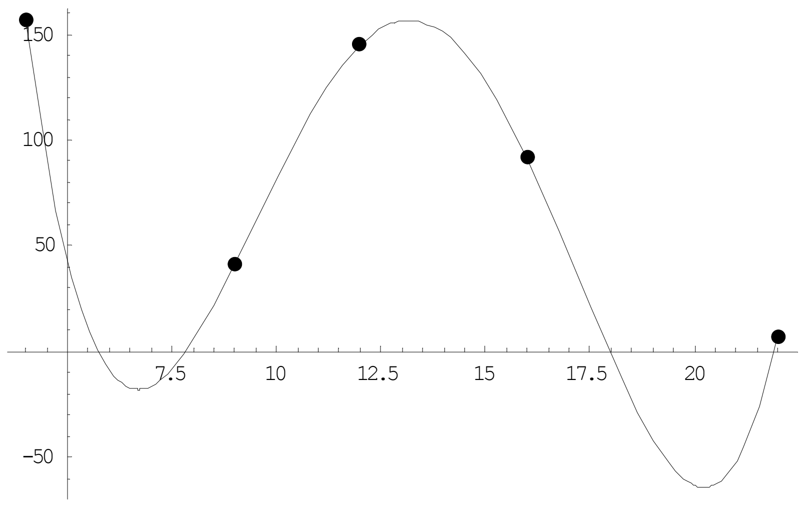

Figure 1

The first term is zero because of (4.1), the second term can be rewritten as

$$-\sum_{i=1}^{n-1}\int_{x_i}^{x_{i+1}} g'(x)S'''(x)\,dx = -\sum_{i=1}^{n-1} c_i \int_{x_i}^{x_{i+1}} g'(x)\,dx = -\sum_{i=1}^{n-1} c_i\left[g(x_{i+1})-g(x_i)\right]=0$$

since $S'''(x)$, *within each segment* is a constant (denoted c_i), because $S(x)$ is a cubic polynomial. $\square$

Step-by-step Solution

To actually construct $S(x)$, we first modify our convention concerning the number of points - we will now assume that there is $n+1$ of them (instead of the old n), resulting in n segments, and the same number of cubic polynomials. These will be constructed in the following form:

$$\begin{aligned} p_1(x) &= a_1 + b_1(x-x_1) + c_1(x-x_1)^2 + d_1(x-x_1)^3 && \text{when } x_1 \le x \le x_2 \\ p_2(x) &= a_2 + b_2(x-x_2) + c_2(x-x_2)^2 + d_2(x-x_2)^3 && \text{when } x_2 \le x \le x_3 \\ &\vdots \\ p_n(x) &= a_1 + b_n(x-x_n) + c_n(x-x_n)^2 + d_n(x-x_n)^3 && \text{when } x_n \le x \le x_{n+1} \end{aligned}$$

Clearly, substituting the left-most value of x implies that each of the a coefficients must be equal to the corresponding y (i.e. $a_1 = y_1$, $a_2 = y_2$, etc.). Similarly, substituting the right-most value, we get

$$\begin{aligned} y_1 + b_1h_1 + c_1h_1^2 + d_1h_1^3 &= y_2 \\ y_2 + b_2h_2 + c_2h_2^2 + d_2h_2^3 &= y_3 \\ &\vdots \\ y_n + b_nh_n + c_nh_n^2 + d_nh_n^3 &= y_{n+1} \end{aligned} \tag{4.2}$$

where $h_1 \equiv x_2 - x_1$, $h_2 \equiv x_3 - x_2$, etc.

To match the values of first derivatives between consecutive segments requires

$$\begin{aligned} b_1 + 2c_1 h_1 + 3d_1 h_1^2 &= b_2 \\ b_2 + 2c_2 h_2 + 3d_2 h_2^2 &= b_3 \\ &\vdots \\ b_{n-1} + 2c_{n-1} h_{n-1} + 3d_{n-1} h_{n-1}^2 &= b_n \end{aligned} \tag{4.3}$$

We also have to match the second derivatives:

$$\begin{aligned} 2c_1 + 6d_1 h_1 &= 2c_2 \\ 2c_2 + 6d_2 h_2 &= 2c_3 \\ &\vdots \\ 2c_{n-1} + 6d_{n-1} h_{n-1} &= 2c_n \end{aligned}$$

to which we may add the (last) end-point condition

$$2c_n + 6d_n h_n = 2c_{n+1} \equiv 0$$

(the $S''(x_1) = 0$ condition implies that $c_1 = 0$).

The last n equations can be solved for

$$\begin{aligned} d_1 &= \frac{c_2 - c_1}{3h_1} \\ d_2 &= \frac{c_3 - c_2}{3h_2} \\ &\vdots \\ d_n &= \frac{c_{n+1} - c_n}{3h_n} \end{aligned} \tag{4.4}$$

When substituted back in (4.2) and (4.3), this yields, respectively:

$$\begin{aligned} b_1 + \frac{c_2 + 2c_1}{3} h_1 &= s_1 \\ b_2 + \frac{c_3 + 2c_2}{3} h_2 &= s_2 \\ &\vdots \\ b_n + \frac{c_{n+1} + 2c_n}{3} h_n &= s_n \end{aligned} \tag{4.5}$$

where $s_1 \equiv \frac{y_2 - y_1}{h_1}$, $s_2 \equiv \frac{y_3 - y_2}{h_2}$, etc. (note that these are the straight-line slopes), and

$$\begin{aligned} (c_1 + c_2) h_1 &= b_2 - b_1 \\ (c_2 + c_3) h_2 &= b_3 - b_2 \\ &\vdots \\ (c_{n-1} + c_n) h_{n-1} &= b_n - b_{n-1} \end{aligned} \tag{4.6}$$

Subtracting consecutive equations of (4.5), multiplying the result by 3, and utilizing (4.6) finally results in

$$\begin{aligned} c_1 h_1 + 2c_2(h_1+h_2) + c_3 h_2 &= 3(s_2 - s_1) \\ c_2 h_2 + 2c_3(h_2+h_3) + c_4 h_3 &= 3(s_3 - s_2) \\ &\vdots \\ c_{n-1} h_{n-1} + 2c_n(h_{n-1}+h_n) + c_{n+1} h_n &= 3(s_n - s_{n-1}) \end{aligned}$$

This is now an ordinary set of $n-1$ linear equations for c_2, c_3, ...c_n. Its matrix form would look as follows:

$2(h_1+h_2)$	h_2	0	0	$\cdots$	0	$3(s_2-s_1)$
h_2	$2(h_2+h_3)$	h_3	0	$\cdots$	0	$3(s_3-s_2)$
0	h_3	$2(h_3+h_4)$	h_4	$\cdots$	0	$3(s_4-s_3)$
0	0	h_4	$2(h_4+h_5)$	$\cdots$	0	$3(s_5-s_4)$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
0	0	$\cdots$	0	h_{n-1}	$2(h_{n-1}+h_n)$	$3(s_n-s_{n-1})$

(4.7)

(to the left of the double line is the coefficient matrix, to the right is the right-hand-side vector).

Once we have solved for the c coefficients, we can compute d and b from (4.4) and (4.5) respectively.

Example: Our first example will apply the procedure to the following simple data set with only three segments:

x:	1	2	4	7
y:	2	–3	0	1

This implies that

h:	1	2	3
$\triangle y$:	–5	3	1
s:	–5	$\frac{3}{2}$	$\frac{1}{3}$

and

$3\,\triangle s$:	$\frac{39}{2}$	$-\frac{7}{2}$

The set of equations for c_2 and c_3 is thus

6	2	$\frac{39}{2}$
2	10	$-\frac{7}{2}$

We can solve it by constructing the coefficient-matrix' INVERSE

$$\begin{bmatrix} 10 & -2 \\ -2 & 6 \end{bmatrix} \div 56$$

where the division (by the matrix' determinant) is ELEMENTWISE (i.e. each element of the 2 by 2 is to be divided by 56), and then multiplying the inverse by the right hand side vector, thus:

$$\begin{bmatrix} 10 & -2 \\ -2 & 6 \end{bmatrix} \begin{bmatrix} \frac{39}{2} \\ -\frac{7}{2} \end{bmatrix} \div 56 = \begin{bmatrix} \frac{101}{28} \\ -\frac{15}{14} \end{bmatrix}$$

This means that

c:	0	$\frac{101}{28}$	$-\frac{15}{14}$	0

(don't forget our convention about $c_4 \equiv 0$), and

$\triangle c$:	$\frac{101}{28}$	$-\frac{131}{28}$	$\frac{15}{14}$

Based on (4.4):

d:	$\frac{101}{84}$	$-\frac{131}{168}$	$\frac{5}{42}$

and, based on (4.5)

b:	$-5 - \frac{\frac{101}{28}}{3} = \frac{-521}{84}$	$\frac{3}{2} - \frac{2 \cdot \frac{101}{28} - \frac{15}{14}}{3} \cdot 2 = -\frac{109}{42}$	$\frac{1}{3} - \frac{2 \cdot \frac{-15}{14}}{3} \cdot 3 = \frac{52}{21}$

The final solution is thus

$$\begin{array}{rcl} 2 - \frac{521}{84}(x-1) + \qquad\qquad \frac{101}{84}(x-1)^3 & 1 < & x < 2 \\ -3 - \frac{109}{42}(x-2) + \frac{101}{28}(x-2)^2 - \frac{131}{168}(x-2)^3 & 2 < & x < 4 \\ \frac{52}{21}(x-4) - \frac{15}{14}(x-4)^2 + \frac{5}{42}(x-4)^3 & 4 < & x < 7 \end{array}$$

One can verify that this solution passes through all four points and is smooth. One can also plot it by utilizing the **piecewise** feature of Maple.

To solve (4.7) when the data has more than three segments, we utilize the fact that our coefficient matrix is TRI-DIAGONAL. This makes the construction of the corresponding solution relatively easy, regardless of the matrix' size. We will now detour to look at this issue in detail.

Tri-diagonal Systems (LU Decomposition)*

First we show that any tri-diagonal matrix can be written as a product of two bi-diagonal matrices, the first having non-zero elements below the main diagonal, the second above. Furthermore, the below-diagonal elements of the fist matrix are those of the original matrix, and the main-diagonal elements of the second matrix are all equal to 1. This constitutes the so called LU DECOMPOSITION of the original matrix.

Example: Consider the following tridiagonal matrix

$$\mathbb{A} = \begin{bmatrix} 3 & -3 & 0 & 0 \\ -3 & 8 & -2 & 0 \\ 0 & 1 & 2 & 4 \\ 0 & 0 & -2 & 6 \end{bmatrix}$$

It can be written as

$$\begin{bmatrix} \boxed{1} & 0 & 0 & 0 \\ -3 & \boxed{3} & 0 & 0 \\ 0 & 1 & \boxed{5} & 0 \\ 0 & 0 & -2 & \boxed{7} \end{bmatrix} \begin{bmatrix} 1 & \boxed{2} & 0 & 0 \\ 0 & 1 & \boxed{4} & 0 \\ 0 & 0 & 1 & \boxed{6} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where $\boxed{n}$ implies that the actual value is yet to be determined (the number inside indicates the order in which one should proceed). Following this pattern, we obtain:

$$\begin{bmatrix} 3 & 0 & 0 & 0 \\ -3 & 5 & 0 & 0 \\ 0 & 1 & \frac{12}{5} & 0 \\ 0 & 0 & -2 & \frac{28}{3} \end{bmatrix} \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 1 & -\frac{2}{5} & 0 \\ 0 & 0 & 1 & \frac{5}{3} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Multiplying these two (the first is usually called $\mathbb{L}$, the second one $\mathbb{U}$), one can verify that their product yields the original matrix.

What is the advantage of having written a tridiagonal matrix $\mathbb{A}$ as a product of two bidiagonal matrices? Well, it greatly simplifies solving a linear set of equations, with $\mathbb{A}$ as the coefficient matrix.

Example: To solve

$$\mathbb{A}\mathbf{x} = \begin{bmatrix} 7 \\ 8 \\ 2 \\ -3 \end{bmatrix}$$

where $\mathbb{A}$ is the matrix of the previous example, and $\mathbf{x}$ is a vector of the four unknowns, we can rewrite it as

$$\mathbb{L}\mathbb{U}\mathbf{x} \equiv \mathbb{L}\mathbf{y} = \begin{bmatrix} 7 \\ 8 \\ 2 \\ -3 \end{bmatrix}$$

first solving (rather easily) for $\mathbf{y}$. Starting from the top equation, this yields: $y_1 = \frac{7}{3}$, $y_2 = \frac{8+7}{5} = 3$, $y_3 = \frac{2-3}{\frac{12}{5}} = -\frac{5}{12}$ and $y_4 = \frac{-3-\frac{5}{6}}{\frac{28}{3}} = -\frac{23}{56}$. Once we have solve for $\mathbf{y}$, we can similarly get the elements of $\mathbf{x}$ based on

$$\mathbb{U}\mathbf{x} = \begin{bmatrix} \frac{7}{3} \\ 3 \\ -\frac{5}{12} \\ -\frac{23}{56} \end{bmatrix}$$

but this time we have to work from bottom up: $x_4 = -\frac{23}{56}$, $x_3 = -\frac{5}{12} + \frac{5}{3} \cdot \frac{23}{56} = \frac{15}{56}$, $x_2 = 3 + \frac{2}{5} \cdot \frac{15}{56} = \frac{87}{28}$ and $x_1 = \frac{7}{3} + \frac{87}{28} = \frac{457}{84}$. One can easily verify that this is the solution of the original set of equations.

We will go (more quickly) over one more example, this time a bit larger (5 by 5). Note that applying the *general* technique (which you must have all seen before) to a problem of this size would be very tedious.

Example:

$$\begin{bmatrix} 2 & 5 & 0 & 0 & 0 \\ 4 & -3 & -2 & 0 & 0 \\ 0 & 3 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 2 & 4 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 4 & -13 & 0 & 0 & 0 \\ 0 & 3 & \frac{7}{13} & 0 & 0 \\ 0 & 0 & 1 & -\frac{13}{7} & 0 \\ 0 & 0 & 0 & 2 & \frac{80}{13} \end{bmatrix} \begin{bmatrix} 1 & \frac{5}{2} & 0 & 0 & 0 \\ 0 & 1 & \frac{2}{13} & 0 & 0 \\ 0 & 0 & 1 & \frac{13}{7} & 0 \\ 0 & 0 & 0 & 1 & -\frac{14}{13} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 4 \\ 0 \\ 2 \\ -3 \\ -1 \end{bmatrix}$$

yields $y_1 = 2$, $y_2 = \frac{8}{13}$, $y_3 = \frac{2}{7}$, $y_4 = \frac{23}{13}$, $y_5 = -\frac{59}{80}$ and $x_5 = -\frac{59}{80,}$ $x_4 = \frac{39}{40}$, $x_3 = -\frac{61}{40}$, $x_2 = \frac{17}{20}$, $x_1 = -\frac{1}{8}$.

We will now return to our discussion of **natural cubic spline**.

Example

We will now fit a cubic spline to our first example (where the regular polynomial fit did not fare too well):

$$\begin{array}{rrrrr} h: & 5 & 3 & 4 & 6 \\ \triangle y: & -116 & 104 & -53 & -85 \\ s: & -\frac{116}{5} & \frac{104}{3} & -\frac{53}{4} & -\frac{85}{6} \end{array}$$

$$\begin{array}{rrrr} 3\triangle s: & \frac{868}{5} & -\frac{575}{4} & -\frac{11}{4} \end{array}$$

$$\left[\begin{array}{rrr||r} 16 & 3 & 0 & \frac{868}{5} \\ 3 & 14 & 4 & -\frac{575}{4} \\ 0 & 4 & 20 & -\frac{11}{4} \end{array}\right]$$

The LU decomposition of the last coefficient matrix is

$$\begin{bmatrix} 16 & 0 & 0 \\ 3 & \frac{215}{16} & 0 \\ 0 & 4 & \frac{4044}{215} \end{bmatrix} \begin{bmatrix} 1 & \frac{3}{16} & 0 \\ 0 & 1 & \frac{64}{215} \\ 0 & 0 & 1 \end{bmatrix}$$

We will thus get $Y_1 = \frac{868}{5 \cdot 16} = \frac{217}{20}$, $Y_2 = (-\frac{575}{4} - 3 \cdot \frac{217}{20})\frac{16}{215} = -\frac{328}{25}$, $Y_3 = (-\frac{11}{4} + 4 \cdot \frac{328}{25})\frac{215}{4044} = \frac{213839}{80880}$ and $c_4 = \frac{213839}{80880}$, $c_3 = -\frac{328}{25} - \frac{64}{215} \cdot \frac{213839}{80880} = -\frac{14060}{1011}$, $c_2 = \frac{217}{20} + \frac{3}{16} \cdot \frac{14060}{1011} = \frac{22676}{1685}$, implying

$$\begin{array}{rrrrr} \triangle c: & \frac{22676}{1685} & -\frac{138328}{5055} & \frac{446213}{26960} & -\frac{213839}{80880} \\ \\ d: & \frac{22676}{25275} & -\frac{138328}{45495} & \frac{446213}{323520} & -\frac{213839}{1455840} \end{array}$$

and

$$\begin{array}{rrrrr} b: & -\frac{230656}{5055} & \frac{109484}{5055} & \frac{102668}{5055} & -\frac{166763}{6740} \end{array}$$

which results in a lot more sensible fit:

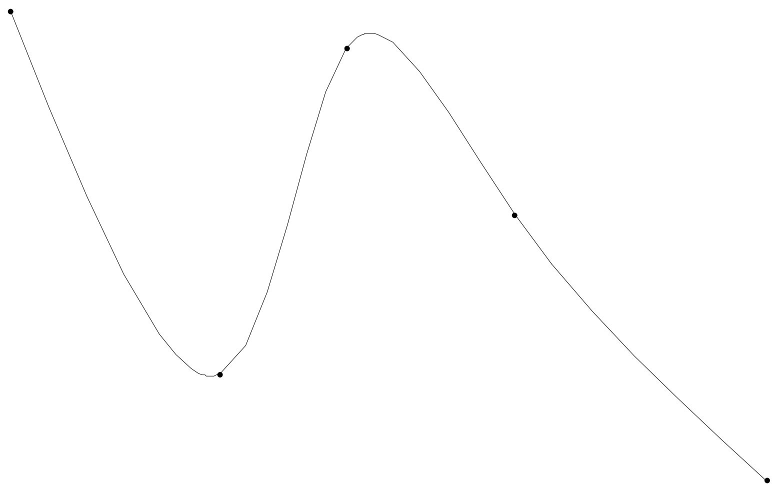

Figure 2

Chapter 5 LEAST-SQUARES FIT (MATRIX ALGEBRA*)

When fitting experimental data with a lot of random scatter, it may be more sensible to use a curve *not* passing exactly through all its points. Instead, we can let it pass 'between' points., to follow the overall general trend only. We will assume that this can be achieved by a relatively simple polynomial (straight line, parabola, etc.) - its degree is normally chosen by simply eye-balling the graph of the data.

To be able to fit the best, say, parabola to a set of points, we must first decide on some sort of criterion to define 'best'. A simple and sensible choice is to *minimize* the sum of squares of the RESIDUALS (*vertical* distances from each point the fitted curve). This of course gives the procedure its name.

The reason that we use *squares* of the distances rather than the distances themselves is rather technical - to define distances, we have to use absolute values, which would be difficult to deal with mathematically. Using squares makes the optimization a lot easier; furthermore, it imposes an extra penalty on producing even a single large residual - this is quite often what we want anyhow.

Using vertical distances (rather than , say perpendicular distances), also has the advantage of simplifying the procedure mathematically; furthermore, it makes the result *scale independent* (not effected when changing the units of x, say from miles to km).

We will now derive the relevant formulas. For simplicity, we will assume that out MODEL (the type of curve we are fitting) is a parabola - the results can be extended to a polynomial of any degree quite easily.

Normal Equations

Let us assume that we have the usual table of x-y observations with n pairs of values (they are denoted x_1, x_2, ... x_n and y_1, y_2, ... y_n). To this data, we are trying to fit a parabola, whose general equation is

$$y = a + b\,x + c\,x^2$$

where a, b and c are yet to be determined coefficients.

For each of our points, we can compute the vertical *residual* (distance from y_i to $\overline{y}_i = a + b\,x_i + c\,x_i^2$) to be

$$y_i - a - b\,x_i - c\,x_i^2$$

We want to *minimize* (by varying a, b and c) the sum of squares of these residuals, or

$$S \equiv \sum_{i=1}^{n} \left(y_i - a - b\,x_i - c\,x_i^2\right)^2$$

This can be achieved by making each of the three partial derivatives, i.e. $\frac{\partial S}{\partial a}$, $\frac{\partial S}{\partial b}$

and $\frac{\partial S}{\partial c}$, equal to zero, namely:

$$\begin{aligned}
-2\sum_{i=1}^{n}\left(y_i - a - b\,x_i - c\,x_i^2\right) &= 0\\
-2\sum_{i=1}^{n}\left(y_i - a - b\,x_i - c\,x_i^2\right)x_i &= 0\\
-2\sum_{i=1}^{n}\left(y_i - a - b\,x_i - c\,x_i^2\right)x_i^2 &= 0
\end{aligned}$$

Cancelling the factor of 2, applying the summation individually to each term, and reorganizing yields

$$\begin{aligned}
a\,n + b\sum_{i=1}^{n}x_i + c\sum_{i=1}^{n}x_i^2 &= \sum_{i=1}^{n}y_i\\
a\sum_{i=1}^{n}x_i + b\sum_{i=1}^{n}x_i^2 + c\sum_{i=1}^{n}x_i^3 &= \sum_{i=1}^{n}x_iy_i\\
a\sum_{i=1}^{n}x_i^2 + b\sum_{i=1}^{n}x_i^3 + c\sum_{i=1}^{n}x_i^4 &= \sum_{i=1}^{n}x_i^2y_i
\end{aligned}$$

This is a linear set of equations (called NORMAL EQUATIONS) for a, b and c; it can be rewritten in the following matrix form:

$$\begin{bmatrix}
n & \sum_{i=1}^{n}x_i & \sum_{i=1}^{n}x_i^2\\
\sum_{i=1}^{n}x_i & \sum_{i=1}^{n}x_i^2 & \sum_{i=1}^{n}x_i^3\\
\sum_{i=1}^{n}x_i^2 & \sum_{i=1}^{n}x_i^3 & \sum_{i=1}^{n}x_i^4
\end{bmatrix}
\begin{bmatrix} a\\ b\\ c\end{bmatrix}
=
\begin{bmatrix}
\sum_{i=1}^{n}y_i\\
\sum_{i=1}^{n}x_iy_i\\
\sum_{i=1}^{n}x_i^2y_i
\end{bmatrix}$$

(note that n can be also written as $\sum_{i=1}^{n}x_i^0$, just to complete the pattern).

It should be quite obvious that, if we repeat the whole procedure using a straight line $y = a + b\,x$ instead of a parabola, the normal equations will read:

$$\begin{bmatrix}
n & \sum_{i=1}^{n}x_i\\
\sum_{i=1}^{n}x_i & \sum_{i=1}^{n}x_i^2
\end{bmatrix}
\begin{bmatrix} a\\ b\end{bmatrix}
=
\begin{bmatrix}
\sum_{i=1}^{n}y_i\\
\sum_{i=1}^{n}x_iy_i
\end{bmatrix}$$

Similarly, fitting a cubic $y = a + b\,x + c\,x^2 + d\,x^3$ requires solving

$$\begin{bmatrix}
n & \sum_{i=1}^{n}x_i & \sum_{i=1}^{n}x_i^2 & \sum_{i=1}^{n}x_i^3\\
\sum_{i=1}^{n}x_i & \sum_{i=1}^{n}x_i^2 & \sum_{i=1}^{n}x_i^3 & \sum_{i=1}^{n}x_i^4\\
\sum_{i=1}^{n}x_i^2 & \sum_{i=1}^{n}x_i^3 & \sum_{i=1}^{n}x_i^4 & \sum_{i=1}^{n}x_i^5\\
\sum_{i=1}^{n}x_i^3 & \sum_{i=1}^{n}x_i^4 & \sum_{i=1}^{n}x_i^5 & \sum_{i=1}^{n}x_i^6
\end{bmatrix}
\begin{bmatrix} a\\ b\\ c\\ d\end{bmatrix}
=
\begin{bmatrix}
\sum_{i=1}^{n}y_i\\
\sum_{i=1}^{n}x_iy_i\\
\sum_{i=1}^{n}x_i^2y_i\\
\sum_{i=1}^{n}x_i^3y_i
\end{bmatrix}$$

etc.

One can show that the resulting smallest value of S (say $S_{\min}$) can be computed by (going back to the 'parabola' case):

$$\sum_{i=1}^{n} y_i^2 - \left[\begin{array}{ccc} a & b & c \end{array}\right] \cdot \left[\begin{array}{c} \sum\limits_{i=1}^{n} y_i \\ \sum\limits_{i=1}^{n} x_i y_i \\ \sum\limits_{i=1}^{n} x_i^2 y_i \end{array}\right]$$

where a, b and c are the optimal coefficients. This can serve to define the so called TYPICAL ERROR:

$$\sqrt{\frac{S_{\min}}{n-3}}$$

where 3 is the number of coefficients in the fitted parabola (2 for a straight line, 4 for a cubic, etc.).

Example: Fit a parabola to the following data:

x:	1	2	3	4	5
y:	2	3	2	2	1

The normal equations we need to solve have the following matrix form:

5	15	55	10
15	55	225	27
55	225	979	89

To learn how to solve these, we have to make to following detour:

Gaussian Elimination*

We will now describe a procedure for solving a linear set of equations (in principle, of any size) by GAUSSIAN ELIMINATION followed by BACKWARD SUBSTITUTION (this is hopefully just a review for most of you). The procedure works fine when performing exact (i.e. fractional) computation, it needs to be modified when working with decimal numbers (to be done in a subsequent chapter).

First we have to realize that there are the following four ELEMENTARY OPERATIONS which can be used to modify a linear set of equations *without changing its solution*. These are:

1. Adding (subtracting) a multiple of a row to (from) any other row.
2. Interchanging two rows (i.e. equations).
3. Multiplying a row (equation) by a *non-zero* number.
4. Interchanging two columns of the coefficient matrix (since this involves interchanging the corresponding unknowns, we have to keep track of these).

It should be quite obvious that, if we proceed in a systematic way, with the help of the first *two* elementary operations we can make all elements below the main diagonal of the coefficient matrix equal to zero. 'Systematic' means eliminating the below-diagonal elements in the following order (using a 4 by 4 example):

$\times$	$\times$	$\times$	$\times$
$\boxed{1}$	$\times$	$\times$	$\times$
$\boxed{2}$	$\boxed{4}$	$\times$	$\times$
$\boxed{3}$	$\boxed{5}$	$\boxed{6}$	$\times$

($\times$ denotes the elements we don't need to worry about). We may run into a small difficulty by encountering a zero element *on* the main diagonal, but that can be resolved by interchanging the corresponding row with a subsequent row having a non-zero value in that place (if we cannot find it, the set of equations does not have a unique solution).

Example: Using the normal equations of our previous example, we first subtract the first row multiplied by 3 (11) from the second (third) row, getting

5	15	55	10
0	10	60	–3
0	60	374	–21

Then, we subtract the second row multiplied by 6 from the third row:

5	15	55	10
0	10	60	–3
0	0	14	–3

Starting from the bottom, we can then compute the value of each unknown, one by one (this step constitutes the *backward substitution*). In terms of our previous example, it yields:

$$\begin{aligned} c &= -\frac{3}{14} \\ b &= \frac{-3 + 60 \cdot \frac{3}{14}}{10} = \frac{69}{70} \\ a &= \frac{10 - 15 \cdot \frac{69}{70} + 55 \cdot \frac{3}{14}}{5} = \frac{7}{5} \end{aligned}$$

One can easily verify that these solve the original set of normal equations. The typical error of this fit is

$$\sqrt{\frac{22 - 10 \cdot \frac{7}{5} - 27 \cdot \frac{69}{70} + 89 \cdot \frac{3}{14}}{5 - 3}} = 0.4781$$

We now return to our discussion of least-square polynomials (Statisticians call the procedure REGRESSION).

Symmetric Data

One can simplify the normal equations when the x-values are spaced symmetrically around a center (say $\hat{x}$, clearly equal the AVERAGE of the x_i values); this of course is the case with EQUIDISTANT SPACING. (such as in our example).

The simplification rests on the following idea: Instead of using the original $y = a + b\,x + c\,x^2$ parabola, we now express it in the following *equivalent* form:

$$y = \hat{a} + \hat{b}(x - \hat{x}) + \hat{c}(x - \hat{x})^2$$

(it should be clear how one can go from one model to the other). The normal equations for $\hat{a}$, $\hat{b}$ and $\hat{c}$ now read

$$\begin{bmatrix} n & \sum_{i=1}^{n}(x_i-\hat{x}) & \sum_{i=1}^{n}(x_i-\hat{x})^2 \\ \sum_{i=1}^{n}(x_i-\hat{x}) & \sum_{i=1}^{n}(x_i-\hat{x})^2 & \sum_{i=1}^{n}(x_i-\hat{x})^3 \\ \sum_{i=1}^{n}(x_i-\hat{x})^2 & \sum_{i=1}^{n}(x_i-\hat{x})^3 & \sum_{i=1}^{n}(x_i-\hat{x})^4 \end{bmatrix} \begin{bmatrix} \hat{a} \\ \hat{b} \\ \hat{c} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n}(x_i-\hat{x})\,y_i \\ \sum_{i=1}^{n}(x_i-\hat{x})^2 y_i \end{bmatrix}$$

But. due to the symmetry of the x_i values, the sum of *odd* powers of $x_i - \hat{x}$ must cancel out. The set will thus simplify to

$$\begin{bmatrix} n & 0 & \sum_{i=1}^{n}(x_i-\hat{x})^2 \\ 0 & \sum_{i=1}^{n}(x_i-\hat{x})^2 & 0 \\ \sum_{i=1}^{n}(x_i-\hat{x})^2 & 0 & \sum_{i=1}^{n}(x_i-\hat{x})^4 \end{bmatrix} \begin{bmatrix} \hat{a} \\ \hat{b} \\ \hat{c} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n}(x_i-\hat{x})\,y_i \\ \sum_{i=1}^{n}(x_i-\hat{x})^2 y_i \end{bmatrix}$$

which means that we can separately solve for the *odd*

$$\begin{bmatrix} n & \sum_{i=1}^{n}(x_i-\hat{x})^2 \\ \sum_{i=1}^{n}(x_i-\hat{x})^2 & \sum_{i=1}^{n}(x_i-\hat{x})^4 \end{bmatrix} \begin{bmatrix} \hat{a} \\ \hat{c} \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n}(x_i-\hat{x})^2 y_i \end{bmatrix}$$

and *even*

$$\hat{b}\sum_{i=1}^{n}(x_i-\hat{x})^2 = \sum_{i=1}^{n}(x_i-\hat{x})\,y_i$$

coefficients (by a process called DECOUPLING of equations). We than need to solve the two sets of equations, each of a smaller (half) size; this is usually a lot easier than dealing directly with the full set.

Example: Returning to our original example, we can clearly see that $\hat{x} = 3$, resulting in

$x - \hat{x}$:	–2	–1	0	1	2
y:	2	3	2	2	1

The set of equations for $\hat{a}$ and $\hat{c}$ then reads:

$$\begin{bmatrix} 5 & 10 \\ 10 & 34 \end{bmatrix} \begin{bmatrix} \hat{a} \\ \hat{c} \end{bmatrix} = \begin{bmatrix} 10 \\ 17 \end{bmatrix}$$

which can be solved by

$$\begin{bmatrix} 34 & -10 \\ -10 & 5 \end{bmatrix} \begin{bmatrix} 10 \\ 17 \end{bmatrix} \div 70 = \begin{bmatrix} \frac{17}{7} \\ -\frac{3}{14} \end{bmatrix}$$

The $\hat{b}$ equation is quite trivial, namely $10\hat{b} = -3 \Rightarrow \hat{b} = -\frac{3}{10}$. The best fitting parabola is thus

$$y = \frac{17}{7} - \frac{3}{10}(x-3) - \frac{3}{14}(x-3)^2$$

One can verify that this expands to the same answer as our previous solution. It also yields the same value of typical error:

$$\sqrt{\frac{22 - 10 \cdot \frac{17}{7} + 17 \cdot \frac{3}{14} - 3 \cdot \frac{3}{10}}{5-3}} = 0.4781$$

Weighted Fit

Suppose that some of the pairs of x-y values in our data are repeated two or more times. It is then convenient to simplify the corresponding table by listing each such pair only once, but adding a new row of the so called WEIGHTS (i.e. how many times was the pair observed), e.g.

x:	1	2	3	4	5
y:	2	3	2	2	1
w:	4	2	1	1	3

To fit the best (least-square) parabola, we now have to minimize

$$S \equiv \sum_{i=1}^{n} w_i \left(y_i - a - b\,x_i - c\,x_i^2\right)^2$$

since each squared residual must be multiplied by the corresponding weight (note that this yields the same value as the old approach would, using the original, uncompressed data).

The normal equations now read

$$\begin{bmatrix} \sum_{i=1}^{n} w_i & \sum_{i=1}^{n} w_i x_i & \sum_{i=1}^{n} w_i x_i^2 \\ \sum_{i=1}^{n} w_i x_i & \sum_{i=1}^{n} w_i x_i^2 & \sum_{i=1}^{n} w_i x_i^3 \\ \sum_{i=1}^{n} w_i x_i^2 & \sum_{i=1}^{n} w_i x_i^3 & \sum_{i=1}^{n} w_i x_i^4 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} w_i y_i \\ \sum_{i=1}^{n} w_i x_i y_i \\ \sum_{i=1}^{n} w_i x_i^2 y_i \end{bmatrix}$$

and are solved in pretty much the same manner as before. Note that the symmetry trick will work only if the w_i values possess similar symmetry (not very likely). Also note that $S_{\min}$ is now computed by

$$\sum_{i=1}^{n} w_i y_i^2 - \begin{bmatrix} a & b & c \end{bmatrix} \cdot \begin{bmatrix} \sum_{i=1}^{n} w_i y_i \\ \sum_{i=1}^{n} w_i x_i y_i \\ \sum_{i=1}^{n} w_i x_i^2 y_i \end{bmatrix}$$

resulting in the typical error of

$$\sqrt{\frac{n}{\sum_{i=1}^{n} w_i} \cdot \frac{S_{\min}}{n-3}}$$

Example: We will now fit the best straight line to the previous set of data. The normal equations are

11	30	21
30	112	49

The solution is

$$\begin{bmatrix} 112 & -30 \\ -30 & 11 \end{bmatrix} \begin{bmatrix} 21 \\ 49 \end{bmatrix} \div 332 = \begin{bmatrix} \frac{441}{166} \\ -\frac{91}{332} \end{bmatrix}$$

yielding the following 'best' straight line

$$y = \frac{441}{166} - \frac{91}{332} x$$

with the typical error of

$$\sqrt{\frac{5}{11} \cdot \frac{45 - 21 \cdot \frac{441}{166} + 49 \cdot \frac{91}{332}}{5-2}} = 0.6326$$

We could also use weights to make the procedure work harder (bigger weights) on points in a range (or ranges) considered more important by us, and be less concerned (smaller weights) with the rest of the data.

Linear Models

If (for whatever reasons) we decide to replace x, x^2,.... in our polynomial model by any other *specific* functions of x (e.g. $\frac{1}{x}$, e^x, $\ln x$, etc.), i.e.

$$y = a\, f_1(x) + b\, f_2(x) + c\, f_3(x)$$

in general, the procedure for finding the best (least-square) parameters requires only the following minor modification of normal equations:

$$\begin{bmatrix} \sum_{i=1}^{n} f_1(x_i)^2 & \sum_{i=1}^{n} f_1(x_i) f_2(x_i) & \sum_{i=1}^{n} f_1(x_i) f_3(x_i) \\ \sum_{i=1}^{n} f_2(x_i) f_1(x_i) & \sum_{i=1}^{n} f_2(x_i)^2 & \sum_{i=1}^{n} f_2(x_i) f_3(x_i) \\ \sum_{i=1}^{n} f_3(x_i) f_1(x_i) & \sum_{i=1}^{n} f_3(x_i) f_2(x_i) & \sum_{i=1}^{n} f_3(x_i)^2 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} \sum_{i=1}^{n} f_1(x_i)\, y_i \\ \sum_{i=1}^{n} f_2(x_i)\, y_i \\ \sum_{i=1}^{n} f_3(x_i)\, y_i \end{bmatrix}$$

This time, symmetry of the x values cannot be of any help to us.

Example: Let us fit our old data (without weights) by the following curve

$$y = \frac{a}{x} + b + c\,x^3$$

This means that $f_1(x) = \frac{1}{x}$, $f_2(x) = 1$ and $f_3(x) = x^3$. The normal equation are

$$\begin{array}{ccc|c}
\sum_{i=1}^{5} \frac{1}{x_i^2} & \sum_{i=1}^{5} \frac{1}{x_i} & \sum_{i=1}^{5} x_i^2 & \sum_{i=1}^{5} \frac{y_i}{x_i} \\
\sum_{i=1}^{5} \frac{1}{x_i} & \sum_{i=1}^{5} 1 & \sum_{i=1}^{5} x_i^3 & \sum_{i=1}^{5} y_i \\
\sum_{i=1}^{5} x_i^2 & \sum_{i=1}^{5} x_i^3 & \sum_{i=1}^{5} x_i^6 & \sum_{i=1}^{5} y_i x_i^3
\end{array}$$

or, numerically

$$\begin{array}{ccc|c}
\frac{5269}{3600} & \frac{137}{60} & 55 & \frac{73}{15} \\
\frac{137}{60} & 5 & 225 & 10 \\
55 & 225 & 20515 & 333
\end{array}$$

Gaussian (forward) elimination results in

$$\begin{array}{ccc|c}
\frac{5269}{3600} & \frac{137}{60} & 55 & \frac{73}{15} \\
0 & \frac{7576}{5269} & \frac{66675}{479} & \frac{12686}{5269} \\
0 & 0 & \frac{37673515}{7576} & -\frac{314271}{3788}
\end{array}$$

and the back substitution yields: $c = -\frac{628542}{37673515}$, $b = \frac{2253323}{684973}$ and $a = -\frac{8891100}{7534703}$ with the typical error of

$$\sqrt{\frac{22 + \frac{8891100}{7534703} \cdot \frac{73}{15} - \frac{2253323}{684973} \cdot 10 + \frac{628542}{37673515} \cdot 333}{5-3}} = 0.4483$$

(this is only a slight improvement over a simple quadratic polynomial, reflecting the fact that the choice of our new model was rather arbitrary).

We could easily modify this procedure to incorporate weights.

Our last example is also a clear indication that performing *exact* computation (using fractions) is getting rather tedious (or even impossible - consider using $\ln x$ as one of our $f(x)$ functions), and that we should consider switching to decimals. This necessitate modifying the Gaussian-elimination procedure, to keep the round off errors of the computation in check. Let us then take another detour, and look at the corresponding issues.

Pivoting*

Switching to decimals when solving a set of linear equations by Gaussian elimination makes the procedure more efficient (especially for computers), but our last algorithm then poses some problems. One of them is quite obvious: an exact zero may, due to the round-off error, 'masquerade' as a small nonzero value (e.g. -1.308×10^{-9}). If this happens on the main diagonal, the procedure will not carry out the required row interchange, and the results will immediately turn out to be totally nonsensical. We may guard for this by checking whether the element (called

PIVOT) is, in absolute value, smaller than say 3×10^{-9} (if yes, assume that it is zero), but that will not totally eliminate the problem of round-off errors. These will tend to erode the accuracy of our results, especially when a main diagonal element (which, when performing the corresponding row subtraction, appears in the denominator), is non-zero but small.

To alleviate this process (erosion of accuracy), we watch for small (not just zero) diagonal elements, and trade them (by row interchange) for the largest (in absolute value) possible element below. When done consistently (i.e. each main diagonal element, small or nor, is checked against all elements of the same column. downwards) the procedure is called PARTIAL PIVOTING).

Accuracy can be further improved (at a cost of more computation) by finding the largest (in absolute value) element below and to the right (without crossing the double line) of the diagonal element, and then swapping the two by the corresponding row and column interchange (to get the solution straight, we have to keep track of the latter). This is done at each step of the procedure, and constitutes the so called COMPLETE (maximal, full) PIVOTING.

Both partial and complete pivoting can further benefit from a procedure called SCALING, which replaces (when searching for the largest element) *absolute* magnitudes by *scaled* (i.e.e relative to the largest element of the same row) magnitudes.

To simplify the matter, *our* pivoting will always be of the complete type, without any scaling (it is one of the best pivoting strategies and it has a rather simple logic).

Example: We will re-do our previous example of Gaussian elimination, this time using decimals:

$$\begin{array}{ccc||c} a & b & c & \\ 1.4636\bar{1} & 2.28\bar{3} & 55 & 4.8\bar{6} \\ 2.28\bar{3} & 5 & 225 & 10 \\ 55 & 225 & 20515 & 333 \end{array}$$

where $4.8\bar{6}$ represents 4.866666667, etc. Note that now we need an extra (first) row, to keep track of the unknowns. The largest element is clearly 20515, which becomes our first 'pivot'. We thus need to interchange rows 1 and 3, and columns 1 and 3:

$$\begin{array}{ccc||c} c & b & a & \\ 20515 & 225 & 55 & 333 \\ 225 & 5 & 2.28\bar{3} & 10 \\ 55 & 2.28\bar{3} & 1.4636\bar{1} & 4.8\bar{6} \end{array}$$

We then eliminate the last two elements of the first column (by the usual subtraction):

$$\begin{array}{ccc||c} c & b & a & \\ 20515 & 225 & 55 & 333 \\ 0.1\times 10^{-6} & 2.532293445 & 1.68016175 & 6.347794298 \\ 0.1\times 10^{-7} & 1.680116175 & 1.316158028 & 3.973905273 \end{array}$$

Note that, due to the round-off error, we are getting 0.1×10^{-6} and 0.1×10^{-7} (relatively 'large' values) instead of true zeros. But, since we know better, we can simply take these to be zeros (which we will do, from now on) - a good computer program would not even bother to compute them.

Now, we are lucky, and the largest element of the bottom right 2 by 2 corner is 2.543261029, sitting already in its proper position (no interchanges necessary). We can thus easily complete the elimination step:

$$\begin{array}{ccc||c} c & b & a & \\ 20515 & 225 & 55 & 333 \\ 0 & 2.532293445 & 1.680116175 & 6.347794298 \\ 0 & 0 & 0.201441103 & -0.237704521 \end{array}$$

Moving on to the backward substitution, we start with the last unknown, which is a (due to our interchanges):

$$\begin{aligned} a &= -\frac{0.237704521}{0.201441103} = -1.18001\,9954 \\ b &= \frac{6.347794298 - 1.680116175\,a}{2.532293445} = 3.28965\,2282 \\ c &= \frac{333 - 55\,a - 225\,b}{20515} = -0.0166839\,2228 \end{aligned}$$

Note that, in spite of pivoting, some accuracy has been lost over the exact $a = -\frac{8891100}{7534703} = -1.18001\,9969$, $b = \frac{2253323}{684973} = 3.28965\,2293$ and $c = -\frac{628542}{37673515} = -0.0166839\,2238$.

Matrix Inverse*

Inverting square matrices follows a similar pattern, so we may as well discuss it now. Hopefully we all remember that an inverse of a SQUARE matrix $\mathbb{A}$ is a matrix (of the same size) usually denoted $\mathbb{A}^{-1}$ such that

$$\mathbb{A}\mathbb{A}^{-1} = \mathbb{A}^{-1}\mathbb{A} = \mathbb{I}$$

where $\mathbb{I}$ is the UNIT matrix (1 on the main diagonal, 0 otherwise). A square matrix which does (not) have an inverse is called REGULAR (SINGULAR) respectively.

The basic algorithm for constructing a matrix inverse works as follows:

The matrix to be inverted is appended (the corresponding Maple's command is **augment**) by a unit matrix of the same size (so now we have a matrix with twice as many columns). Using the first three elementary operations, the original matrix is converted to a unit matrix, while the appended matrix becomes the desired inverse. This can be achieved by

1. making the diagonal elements, one by one, equal to 1 (dividing the whole row by the diagonal element's original value) - if the diagonal element is equal to 0, we look for the first nonzero element directly below, and interchange the corresponding two rows first,

2. subtracting a multiple of this row (whose main diagonal element was just made equal to 1) from all other rows, to make the elements above and below this 1 equal to 0.

This will work nicely when using exact (fractional) numbers. The only difficulty we may encounter is finding 0 on the main diagonal, trying to exchange it with a nonzero element of the same column going down, and discovering that all of these are also equal to zero. This simply means that the matrix is SINGULAR (does not have an inverse), and the procedure thus terminates.

Example: To invert

$$\begin{bmatrix} 0 & 3 & -4 \\ 3 & 2 & -1 \\ 4 & -2 & 1 \end{bmatrix}$$

we first append a 3 by 3 unit matrix

$$\begin{bmatrix} 0 & 3 & -4 & 1 & 0 & 0 \\ 3 & 2 & -1 & 0 & 1 & 0 \\ 4 & -2 & 1 & 0 & 0 & 1 \end{bmatrix}$$

then interchange the first two rows

$$\begin{bmatrix} 3 & 2 & -1 & 0 & 1 & 0 \\ 0 & 3 & -4 & 1 & 0 & 0 \\ 4 & -2 & 1 & 0 & 0 & 1 \end{bmatrix}$$

multiply the first row by $\frac{1}{3}$

$$\begin{bmatrix} 1 & \frac{2}{3} & -\frac{1}{3} & 0 & \frac{1}{3} & 0 \\ 0 & 3 & -4 & 1 & 0 & 0 \\ 4 & -2 & 1 & 0 & 0 & 1 \end{bmatrix}$$

and subtract the first row, multiplied by 4, form the last row

$$\begin{bmatrix} 1 & \frac{2}{3} & -\frac{1}{3} & 0 & \frac{1}{3} & 0 \\ 0 & 3 & -4 & 1 & 0 & 0 \\ 0 & -\frac{14}{3} & \frac{7}{3} & 0 & -\frac{4}{3} & 1 \end{bmatrix}$$

Then, we multiply the second row by $\frac{1}{3}$, and subtract the resulting row, multiplied by $\frac{2}{3}$ $(-\frac{14}{3})$ from Row 1 (3):

$$\begin{bmatrix} 1 & 0 & \frac{5}{9} & -\frac{2}{9} & \frac{1}{3} & 0 \\ 0 & 1 & -\frac{4}{3} & \frac{1}{3} & 0 & 0 \\ 0 & 0 & -\frac{35}{9} & \frac{14}{9} & -\frac{4}{3} & 1 \end{bmatrix}$$

Finally, we multiply the last row by $-\frac{9}{35}$, and subtract the result, multiplied by $\frac{5}{9}$ $(-\frac{4}{3})$ from Row 1 (2):

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \frac{1}{7} & \frac{1}{7} \\ 0 & 1 & 0 & -\frac{1}{5} & \frac{16}{35} & -\frac{12}{35} \\ 0 & 0 & 1 & -\frac{2}{5} & \frac{12}{35} & -\frac{9}{35} \end{bmatrix}$$

The resulting inverse is thus

$$\begin{bmatrix} 0 & \frac{1}{7} & \frac{1}{7} \\ -\frac{1}{5} & \frac{16}{35} & -\frac{12}{35} \\ -\frac{2}{5} & \frac{12}{35} & -\frac{9}{35} \end{bmatrix}$$

which can be easily verified by direct matrix multiplication.

Incorporating pivoting

When working with decimals, to maintain good accuracy of the procedure (for matrix inversion), we have to do the same kind of pivoting as when solving sets of linear equations. To keep things simple, we will use only complete pivoting without scaling. Since this involves column interchange, we have to keep track of these and 'undo' their effect when reaching the final answer. This is best explained by an

Example: Let us now invert

$$\begin{bmatrix} -53 & 85 & 49 & 78 \\ 17 & 72 & -99 & -85 \\ -86 & 30 & 80 & 72 \\ 66 & -29 & -91 & -53 \end{bmatrix}$$

using decimal arithmetic. First we append a 4 by 4 unit matrix, then we identify -99 as our first pivot. This means that we have to interchange rows 1 and 2, and columns 1 and 3 (using Maple, by **swaprow** and **swapcolumn**):

$$\left[\begin{array}{cccccccc} \boxed{3} & \boxed{2} & \boxed{1} & \boxed{4} & & & & \\ -99 & 72 & 17 & -85 & 0 & 1 & 0 & 0 \\ 49 & 85 & -53 & 78 & 1 & 0 & 0 & 0 \\ 80 & 30 & -86 & 72 & 0 & 0 & 1 & 0 \\ -91 & -29 & 66 & -53 & 0 & 0 & 0 & 1 \end{array}\right]$$

Note how we are keeping track of column interchanges. We then go over one step of the procedure, reducing the first column to 1 and three 0's (the actual print out shows 0.999999999, 0, .1e–7 and –.1e–7, but we know better):

$$\left[\begin{array}{cccccccc} \boxed{3} & \boxed{2} & \boxed{1} & \boxed{4} & & & & \\ 1 & -0.\overline{72} & -0.\overline{17} & 85 & 0 & -0.\overline{01} & 0 & 0 \\ 0 & 120.\overline{63} & -44.\overline{58} & 35.\overline{92} & 1 & 0.\overline{49} & 0 & 0 \\ 0 & 88.\overline{18} & -72.\overline{26} & 3.\overline{31} & 0 & 0.\overline{80} & 1 & 0 \\ 0 & -95.\overline{18} & 50.\overline{37} & 25.\overline{13} & 0 & -0.\overline{91} & 0 & 1 \end{array}\right]$$

Now, we have been lucky, the largest (in absolute value) element in the remaining 3 by 3 corner is $120.\overline{63}$ itself, so no interchanges are necessary. The next reduction step yields:

$$\left[\begin{array}{cccccccc} \boxed{3} & \boxed{2} & \boxed{1} & \boxed{4} & & & & \\ 1 & 0 & -0.4405.. & 1.0751.. & 0.0060.. & -0.0071.. & 0 & 0 \\ 0 & 1 & -0.3695.. & 0.2978.. & 0.0082.. & 0.0041.. & 0 & 0 \\ 0 & 0 & -39.6716.. & -22.9501.. & -0.7309.. & 0.4462.. & 1 & 0 \\ 0 & 0 & 15.1955.. & 53.4794.. & 0.7889.. & -0.5286.. & 0 & 1 \end{array}\right]$$

Now, the largest element in the remaining 2 by 2 block is 53.4794.. This means we have to carry out one more interchange:

$$\left[\begin{array}{cccccccc} \boxed{3} & \boxed{2} & \boxed{4} & \boxed{1} & & & & \\ 1 & 0 & 1.0751.. & -0.4405.. & 0.0060.. & -0.0071.. & 0 & 0 \\ 0 & 1 & 0.2978.. & -0.3695.. & 0.0082.. & 0.0041.. & 0 & 0 \\ 0 & 0 & 53.4794.. & 15.1955.. & 0.7889.. & -0.5286.. & 0 & 1 \\ 0 & 0 & -22.9501.. & -39.6716.. & -0.7309.. & 0.4462.. & 1 & 0 \end{array}\right]$$

The next step yields:

$$\left[\begin{array}{cccccccc}
\boxed{3} & \boxed{2} & \boxed{4} & \boxed{1} & & & & \\
1 & 0 & 0 & -0.7460.. & -0.0098.. & 0.0035.. & 0 & -0.0201.. \\
0 & 1 & 0 & -0.4542.. & 0.0038.. & 0.0070.. & 0 & -0.0055.. \\
0 & 0 & 1 & 0.2841.. & 0.0147.. & -0.0098.. & 0 & 0.0186.. \\
0 & 0 & 0 & -33.1505.. & -0.3923.. & 0.2194.. & 1 & 0.4291..
\end{array}\right]$$

At this point we have reached the last main diagonal element (no more interchanges possible), so we just carry out the last reduction:

$$\left[\begin{array}{cccccccc}
\boxed{3} & \boxed{2} & \boxed{4} & \boxed{1} & & & & \\
1 & 0 & 0 & 0 & -0.0010.. & -0.0014.. & -0.0225.. & -0.0297.. \\
0 & 1 & 0 & 0 & 0.0092.. & 0.0040.. & -0.0137.. & -0.0114.. \\
0 & 0 & 1 & 0 & 0.0113.. & -0.0080.. & 0.0085.. & 0.0223.. \\
0 & 0 & 0 & 1 & 0.0118.. & -0.0066.. & -0.0301.. & -0.0129..
\end{array}\right]$$

And now comes the essential last step: We drop the leading unit matrix and attach the corresponding column labels to *rows* of the remaining matrix, thus:

$$\left[\begin{array}{ccccc}
\boxed{3} & -0.0010.. & -0.0014.. & -0.0225.. & -0.0297.. \\
\boxed{2} & 0.0092.. & 0.0040.. & -0.0137.. & -0.0114.. \\
\boxed{4} & 0.0113.. & -0.0080.. & 0.0085.. & 0.0223.. \\
\boxed{1} & 0.0118.. & -0.0066.. & -0.0301.. & -0.0129..
\end{array}\right]$$

The result needs to be rearranged (by row interchanges) to bring back the natural sequence of labels:

$$\begin{array}{ccccc}
\boxed{1} & 0.01183634283 & -0.006618577468 & -0.03016539786 & -0.01294518528 \\
\boxed{2} & 0.009271619264 & 0.004040818703 & -0.0137015577 & -0.01144897059 \\
\boxed{3} & -0.001003899605 & -0.001425749021 & -0.02250376794 & -0.02976201497 \\
\boxed{4} & 0.01139012326 & -0.008005031964 & 0.00857116598 & 0.0223770053
\end{array}$$

This is then the resulting inverse, as can be easily verified.

Chapter 6 APPROXIMATING FUNCTIONS

Suppose that we would like to find the 'best' way of approximating the $\sin x$ function by a polynomial. We will do this only for $0 < x < \frac{\pi}{2}$, since we know that, due to the $\sin x$'s periodicity, this would be sufficient to evaluate the function *everywhere*.

One way to do this would be to evaluate $\sin x$ at $x = 0$, 10, 20, ... 90 degrees, and fit the best (least-square) polynomial using techniques of the previous section. But of course this leaves the question of what would be the accuracy of our fit *between* the points thus chosen, and should not we really need to go to $x = 0$, 5, 10, 15, ...90 degrees, etc.

To avoid the problem of how finely do we have to subdivide the original interval, why don't we try to cover it all, to letting *each* of its points contribute to the 'sum' of residuals. To be able to do that, we must of course replace 'sum' by the corresponding integral, as follows:

$$S = \int_0^{\pi/2} \left(\sin x - a - b\,x - c\,x^2\right)^2 dx$$

(assuming we want to fit the best *quadratic* polynomial), and

$$S = \int_A^B \left(y(x) - a - b\,x - c\,x^2\right)^2 dx$$

in general (fitting a quadratic polynomial to a function $y(x)$, in the $[A, B]$ interval of x values).

It is a simple exercise to minimize S, getting the following set of normal equations for a, b and c

$$\begin{array}{ccc||c}
\int_A^B dx & \int_A^B x\,dx & \int_A^B x^2 dx & \int_A^B y(x) dx \\
\int_A^B x\,dx & \int_A^B x^2 dx & \int_A^B x^3 dx & \int_A^B x\,y(x)\,dx \\
\int_A^B x^2 dx & \int_A^B x^3 dx & \int_A^B x^4 dx & \int_A^B x^2 y(x)\,dx
\end{array}$$

We can substantially simplify solving this set of equations (of any size) by assuming that $A = -1$ and $B = 1$. This is not really a big restriction, since we can always go from $y(x)$ to $Y(X) \equiv y\left(\frac{A+B}{2} + \frac{B-A}{2}\,X\right)$, where $Y(X)$, when X varies from -1 to 1, has exactly the same values as $y(x)$, when x varies from A to B. Once we have found a polynomial fit to $Y(X)$, we can easily convert it to that of $y(x)$ by the following replacement: $X \to \frac{2x-(A+B)}{B-A}$ (resulting in a polynomial of the same degree).

We can thus concentrate on solving

$$\left.\begin{array}{ccc} \int_{-1}^{1} dX & \int_{-1}^{1} X\,dX & \int_{-1}^{1} X^2dX \\ \int_{-1}^{1} X\,dX & \int_{-1}^{1} X^2dX & \int_{-1}^{1} X^3dX \\ \int_{-1}^{1} X^2dX & \int_{-1}^{1} X^3dX & \int_{-1}^{1} X^4dX \end{array}\right\Vert \begin{array}{c} \int_{-1}^{1} Y(X)dX \\ \int_{-1}^{1} X\cdot Y(X)\,dX \\ \int_{-1}^{1} X^2\cdot Y(X)\,dX \end{array}$$

(or an equivalent). The question is: can we find an equivalent formulation of the problem at hand, which would be easier to solve?

Well, one thing should be obvious: $Y = a + b\,X + c\,X^2$ and

$$Y = \hat{a}\,\phi_0(X) + \hat{b}\,\phi_1(X) + \hat{c}\,\phi_2(X) \tag{6.1}$$

where $\phi_0(X)$, $\phi_1(X)$ and $\phi_2(X)$ are, at this point arbitrary, polynomials of degree zero, one and two (respectively), are *equivalent* models (they both represent a *general* quadratic polynomial). Furthermore, we can choose these $\phi_i(X)$ polynomials to be ORTHOGONAL in the $(-1, 1)$ interval, meaning that

$$\int_{-1}^{1} \phi_i(X)\cdot\phi_j(X)\,dX = 0$$

whenever $i \neq j$ (this will be verified shortly, by actually constructing them, one by one, to meet this property).

We already know (analogically with the discrete case) that, using (6.1) as our model, the normal equations are

$$\left.\begin{array}{ccc} \int_{-1}^{1} \phi_0(X)^2dX & \int_{-1}^{1} \phi_0(X)\phi_1(X)\,dX & \int_{-1}^{1} \phi_0(X)\phi_1(X)\,dX \\ \int_{-1}^{1} \phi_1(X)\phi_0(X)\,dX & \int_{-1}^{1} \phi_1(X)^2dX & \int_{-1}^{1} \phi_1(X)\phi_2(X)\,dX \\ \int_{-1}^{1} \phi_2(X)\phi_0(X)\,dX & \int_{-1}^{1} \phi_2(X)\phi_1(X)\,dX & \int_{-1}^{1} \phi_2(X)^2dX \end{array}\right\Vert \begin{array}{c} \int_{-1}^{1} \phi_0(X)Y(X)dX \\ \int_{-1}^{1} \phi_1(X)Y(X)\,dX \\ \int_{-1}^{1} \phi_2(X)Y(X)\,dX \end{array}$$

Now, choosing the $\phi_i(X)$ polynomials to be *orthogonal* makes all off-diagonal elements of the coefficient matrix equal to zero, and the corresponding set of equations trivial to solve:

$$\begin{aligned} \hat{a} &= \frac{\int_{-1}^{1} \phi_0(X)Y(X)dX}{\alpha_0} \\ \hat{b} &= \frac{\int_{-1}^{1} \phi_1(X)Y(X)dX}{\alpha_1} \\ \hat{c} &= \frac{\int_{-1}^{1} \phi_2(X)Y(X)dX}{\alpha_2} \end{aligned}$$

where $\alpha_0 \equiv \int_{-1}^{1} \phi_0(X)^2dX$, $\alpha_1 \equiv \int_{-1}^{1} \phi_1(X)^2dX$, etc.

Furthermore, if we decide to extend our fit to a cubic polynomial, we don't have to solve the normal equations from scratch, we can simply add

$$\hat{d} = \frac{\int_{-1}^{1} \phi_3(X) Y(X) dX}{\alpha_3}$$

to the previously obtained $\hat{a}$, $\hat{b}$ and $\hat{c}$ (they remain *unchanged*).

Similarly to the discrete case, typical error of the resulting fit is computed as the square root of the *average* value of $\Big(Y(X) - \hat{a}\,\phi_0(X) - \hat{b}\,\phi_1(X) - \hat{c}\,\phi_2(X)\Big)^2$, namely

$$\sqrt{\frac{S_{\min}}{2}} = \sqrt{\frac{\int_{-1}^{1} Y(X)^2 dX - \alpha_0\,\hat{a}^2 - \alpha_1\,\hat{b}^2 - \alpha_2\,\hat{c}^2}{2}}$$

Orthogonal (Legendgre) Polynomials

We now construct polynomials which are orthogonal over the $(-1, 1)$ interval.

We start with

$$\phi_0(X) = 1$$

(any non-zero constant would do, we choose the simplest one); the corresponding α_0 is equal to $\int_{-1}^{1} 1^2\, dX = 2$.

The next, linear polynomial must be orthogonal to $\phi_0(X)$; this means we have one condition to meet. Its leading coefficient can be chosen arbitrarily, and we make it equal to 1 (this will be our choice throughout the procedure - such polynomials are called MONIC, but we don't have to remember that). Its absolute term will be, at this point, arbitrary, thus: $\phi_1(X) = X + C$. Making it orthogonal to $\phi_0(X)$ requires

$$\int_{-1}^{1} 1 \cdot (X + C)\, dX = 2C = 0$$

which implies that $C = 0$. We thus have

$$\phi_1(X) = X$$

with $\alpha_1 = \int_{-1}^{1} X^2\, dX = \frac{2}{3}$.

At this point we may realize that polynomials containing only odd powers of X are automatically orthogonal (in this sense) to polynomials with only even powers. And, this will thus be the general pattern of the ϕ_0, ϕ_1, ϕ_2,... sequence (each polynomial will consist of either odd or even powers of X only, depending on its degree). We will now continue with our construction, utilizing this 'shortcut'.

The next polynomial thus have to have the following form: $\phi_2(X) = X^2 + C$ (not the same C as before). It is automatically orthogonal to $\phi_1(X)$, we must also make it orthogonal to $\phi_0(X)$ by

$$\int_{-1}^{1} 1 \cdot (X^2 + C)\, dX = \frac{2}{3} + 2C = 0 \Rightarrow C = -\frac{1}{3}$$

This results in

$$\phi_2(X) = X^2 - \frac{1}{3}$$

with $\alpha_2 = \int_{-1}^{1}(X^2 - \frac{1}{3})^2\,dX = \frac{2}{5} - \frac{4}{9} + \frac{2}{9} = \frac{8}{45}$ (the $\int_{-1}^{1} X^{2n}\,dX = \frac{2}{2n+1}$ formula comes handy).

Similarly, $\phi_3(X) = X^3 + C\,X$. Making it orthogonal to $\phi_1(X)$:

$$\int_{-1}^{1} X \cdot (X^3 + C\,X)\,dX = \frac{2}{5} + \frac{2}{3}C = 0 \Rightarrow C = -\frac{3}{5}$$

Thus

$$\phi_3(X) = X^3 - \frac{3}{5}X$$

and $\alpha_3 = \int_{-1}^{1}(X^3 - \frac{3}{5}X)^2\,dX = \frac{2}{7} - \frac{6}{5}\cdot\frac{2}{5} + \frac{9}{25}\cdot\frac{2}{3} = \frac{8}{175}$.

To construct $\phi_4(X)$ we can use $X^4 + C_2X^2 + C_0$, but it is more convenient (why?) to use $\phi_4(X) = X^4 + C_2\phi_2(X) + C_0\phi_0(X)$ instead. We have to make it orthogonal to $\phi_0(x)$:

$$\int_{-1}^{1} \phi_0(X)\cdot\left(X^4 + C_2\phi_2(X) + C_0\phi_0(X)\right)\,dX = \frac{2}{5} + 2C_0 = 0 \Rightarrow C_0 = -\frac{1}{5}$$

and to $\phi_2(X)$:

$$\int_{-1}^{1} \phi_2(X)\cdot\left(X^4 + C_2\phi_2(X) + C_0\phi_0(X)\right)\,dX = \frac{2}{7} - \frac{1}{3}\cdot\frac{2}{5} + \frac{8}{45}C_1 = 0 \Rightarrow C_1 = -\frac{6}{7}$$

implying that

$$\phi_4(X) = X^4 - \frac{6}{7}\left(X^2 - \frac{1}{3}\right) - \frac{1}{5} = X^4 - \frac{6}{7}X^2 + \frac{3}{35}$$

(note that, in the final answer, the coefficient signs always alternate), with $\alpha_4 = \int_{-1}^{1}(X^4 - \frac{6}{7}X^2 + \frac{3}{35})^2\,dX = \frac{128}{11025}$ (at this point we may as well let Maple do the dirty work).

It should be obvious, how to continue this process (called GRAM-SCHMIDT ORTHOGONALIZATION). The resulting polynomials are called LEGENDRE, and are quite useful in many other areas of Mathematics and Physics (their usual definition differs from ours - the leading coefficient is not equal to 1).

Example: We will fit the $\sin x$ function by a cubic polynomial, in the $(1, \frac{\pi}{2})$ interval. Replacing x by $\frac{A+B}{2} + \frac{B-A}{2}X = \frac{\pi}{4}(1 + X)$, this is the same as fitting

$\sin[\frac{\pi}{4}(1+X)]$ over $(-1,1)$. The previous formulas enable us to find

$$\begin{aligned}
\hat{a} &= \frac{\int_{-1}^{1}\sin[\frac{\pi}{4}(1+X)]\,dX}{2} = \frac{2}{\pi} \\
\hat{b} &= \frac{\int_{-1}^{1}X\sin[\frac{\pi}{4}(1+X)]\,dX}{\frac{2}{3}} = 6\cdot\frac{4-\pi}{\pi^2} \\
\hat{c} &= \frac{\int_{-1}^{1}(X^2-\frac{1}{3})\sin[\frac{\pi}{4}(1+X)]\,dX}{\frac{8}{45}} = -15\cdot\frac{48-12\pi-\pi^2}{\pi^3} \\
\hat{d} &= \frac{\int_{-1}^{1}(X^3-\frac{3}{5}X)\sin[\frac{\pi}{4}(1+X)]\,dX}{\frac{8}{175}} = -35\cdot\frac{960-240\pi-24\pi^2+\pi^3}{\pi^4}
\end{aligned}$$

almost immediately (with a bit of help from Maple, concerning the integration). The corresponding least-squares polynomial is thus

$$\begin{gathered}
\frac{2}{\pi}+6\cdot\frac{4-\pi}{\pi^2}X-15\cdot\frac{48-12\pi-\pi^2}{\pi^3}(X^2-\frac{1}{3}) \\
-35\cdot\frac{960-240\pi-24\pi^2+\pi^3}{\pi^4}(X^3-\frac{3}{5}X)
\end{gathered}$$

having the following typical error

$$\sqrt{\frac{\int_{-1}^{1}\sin[\frac{\pi}{4}(1+X)]^2\,dX-2\hat{a}^2-\frac{2}{3}\hat{b}^2-\frac{8}{45}\hat{c}^2-\frac{8}{175}\hat{d}^2}{2}} = 0.000833$$

(as an approximation, the polynomial will be correct to three digits). The final answer must of course be given in terms of x. This is achieved by the following replacement: $X \longrightarrow \frac{4}{\pi}\cdot x - 1$ (Maple can do this, quite efficiently, using the **subs** command). The result was also expanded and converted to decimal form.

$$-2.25846\,244\times10^{-3}+1.02716\,9553x-0.069943\,7654x^2-0.11386\,84594x^3$$

One can plot this cubic, together with the original $\sin x$, for x from 0 to $\frac{\pi}{2}$, to see that the fit is quite good (the two curves will appear identical). Note that, as soon as go beyond $\frac{\pi}{2}$, the disagreement becomes clearly visible (no extrapolation possible). Furthermore, when displaying the *difference* between the two functions, we can see (in Figure 1) that the error is clearly the largest at each end of the original range (more than double of what it would be otherwise).

Chebyshev Polynomials

In our last example, we noticed that the error of our polynomial fit increases toward the interval's ends. This is undesirable when the objective is to minimize the *largest* (rather than *typical*) error. Developing a procedure to find the best (in the new sense) solution would is quite difficult, but there exists a compromise approach which is capable of getting quite close to it. The idea is to use weights (or, in this

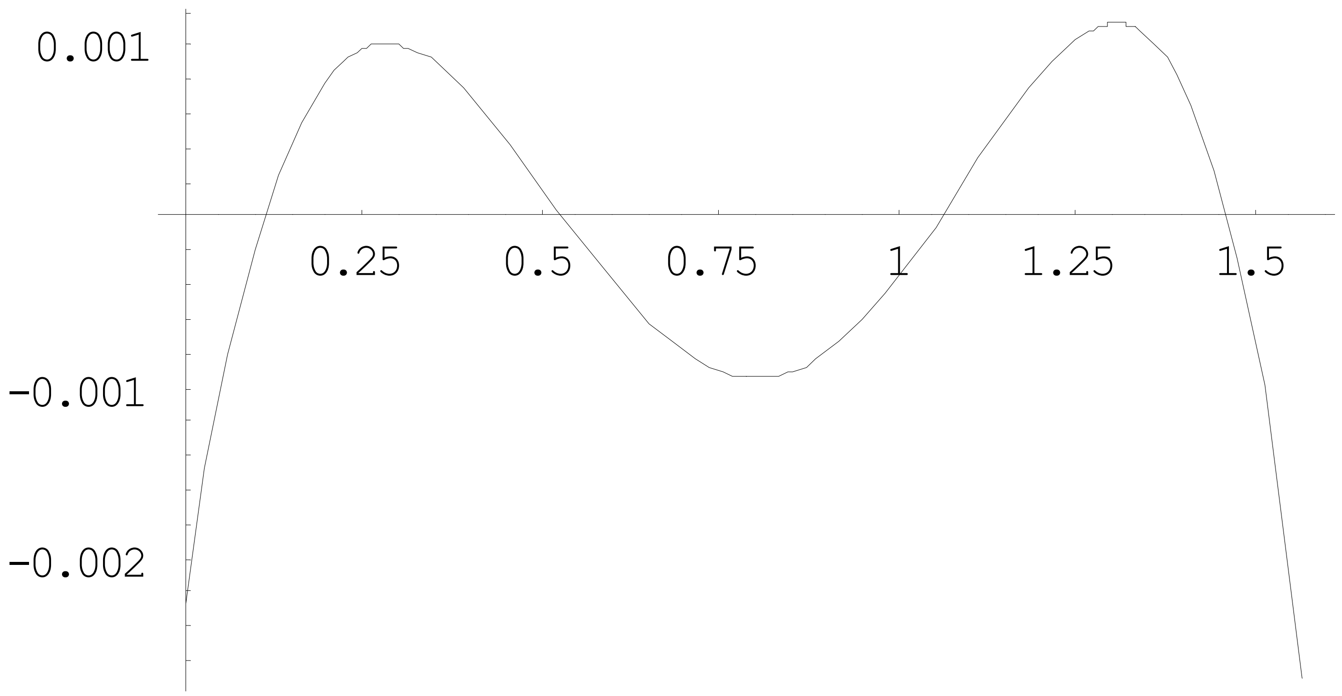

Figure 1

case, a WEIGHT FUNCTION), chosen so that the least-square procedure is forced to work harder at the interval's ends. The usual choice (which works quite well) is

$$W(X) = \frac{1}{\sqrt{1-X^2}}$$

which increases (actually, it goes to infinity) as $X \to 1$ and $X \to -1$.

We now have to modify the procedure to minimize

$$S \equiv \int_{-1}^{1} \frac{\left(Y(X) - \hat{a}\Phi_0(X) - \hat{b}\Phi_1(X) - \hat{c}\,\Phi_2(X)\right)^2}{\sqrt{1-X^2}} dX$$

where $\Phi_0(X)$, $\Phi_1(X)$ and $\Phi_2(X)$ are polynomials of degree zero, one and two. This leads to the following normal equations:

$$\left\|\begin{array}{ccc|c} \int_{-1}^{1} \frac{\Phi_0(X)^2 dX}{\sqrt{1-X^2}} & \int_{-1}^{1} \frac{\Phi_0(X)\Phi_1(X)\,dX}{\sqrt{1-X^2}} & \int_{-1}^{1} \frac{\Phi_0(X)\Phi_1(X)\,dX}{\sqrt{1-X^2}} & \int_{-1}^{1} \frac{\Phi_0(X)Y(X) dX}{\sqrt{1-X^2}} \\ \int_{-1}^{1} \frac{\Phi_1(X)\Phi_0(X)\,dX}{\sqrt{1-X^2}} & \int_{-1}^{1} \frac{\Phi_1(X)^2 dX}{\sqrt{1-X^2}} & \int_{-1}^{1} \frac{\Phi_1(X)\Phi_2(X)\,dX}{\sqrt{1-X^2}} & \int_{-1}^{1} \frac{\Phi_1(X)Y(X)\,dX}{\sqrt{1-X^2}} \\ \int_{-1}^{1} \frac{\Phi_2(X)\Phi_0(X)\,dX}{\sqrt{1-X^2}} & \int_{-1}^{1} \frac{\Phi_2(X)\Phi_1(X)\,dX}{\sqrt{1-X^2}} & \int_{-1}^{1} \frac{\Phi_2(X)^2 dX}{\sqrt{1-X^2}} & \int_{-1}^{1} \frac{\Phi_2(X)Y(X)\,dX}{\sqrt{1-X^2}} \end{array}\right\|$$

Assuming that $\Phi_0(X)$, $\Phi_1(X)$ and $\Phi_2(X)$ are orthogonal in the following new sense

$$\int_{-1}^{1} \frac{\Phi_i(X)\Phi_j(X)\,dX}{\sqrt{1-X^2}} = 0$$

whenever $i \neq j$, the best coefficients are clearly

$$\begin{aligned}\hat{a} &= \frac{\int_{-1}^{1} \frac{\phi_0(X)Y(X)dX}{\sqrt{1-X^2}}}{\alpha_0} \\ \hat{b} &= \frac{\int_{-1}^{1} \frac{\phi_1(X)Y(X)dX}{\sqrt{1-X^2}}}{\alpha_1} \\ \hat{c} &= \frac{\int_{-1}^{1} \frac{\phi_2(X)Y(X)dX}{\sqrt{1-X^2}}}{\alpha_2}\end{aligned}$$

where $\alpha_0 \equiv \int_{-1}^{1} \frac{\phi_0(X)^2dX}{\sqrt{1-X^2}}$, $\alpha_1 \equiv \int_{-1}^{1} \frac{\phi_1(X)^2dX}{\sqrt{1-X^2}}$, etc.

The typical-error formula needs to be modified to

$$\sqrt{\frac{\int_{-1}^{1} \frac{Y(X)^2dX}{\sqrt{1-X^2}} - \alpha_0\,\hat{a}^2 - \alpha_1\,\hat{b}^2 - \alpha_2\,\hat{c}^2}{\pi}}$$

where the denominator π equals to total weight, i.e. $\int_{-1}^{1} \frac{dX}{\sqrt{1-X^2}}$.

Using the Gram-Schmidt procedure, we now proceed to construct this new set of orthogonal polynomials. Since our weight function is symmetric, i.e. $W(X) = W(-X)$, the polynomials will also consist of either even or odd powers of X only. We may also need

$$\int_{-1}^{1} \frac{X^n dX}{\sqrt{1-X^2}} = \frac{(n-1)!!}{n!!}\pi$$

when n is even (equal to zero for n odd).

The first polynomial is always

$$\Phi_0(X) = 1$$

with $\alpha_0 = \int_{-1}^{1} \frac{dX}{\sqrt{1-X^2}} = \pi$.

Similarly, the next one is

$$\Phi_1(X) = X$$

with $\alpha_1 = \int_{-1}^{1} \frac{X^2dX}{\sqrt{1-X^2}} = \frac{\pi}{2}$.

To construct $\Phi_2(X) = X^2 + C$, we have to establish the value of C, based on

$$\int_{-1}^{1} \frac{(X^2+C)\,dX}{\sqrt{1-X^2}} = \pi C + \frac{\pi}{2} = 0 \Rightarrow C = -\frac{1}{2}$$

so

$$\Phi_2(X) = X^2 - \frac{1}{2}$$

with $\alpha_2 = \int_{-1}^{1} \frac{(X^2-\frac{1}{2})^2dX}{\sqrt{1-X^2}} = \frac{\pi}{8}$.

Similarly

$$\int_{-1}^{1} \frac{(X^3+CX)\cdot X\,dX}{\sqrt{1-X^2}} = \frac{3\pi}{8} + \frac{\pi}{2}C = 0 \Rightarrow C = -\frac{3}{4}$$

which yields

$$\Phi_3(X) = X^3 - \frac{3}{4}X$$

and $\alpha_3 = \int_{-1}^{1} \frac{(X^3-\frac{3}{4}X)^2 dX}{\sqrt{1-X^2}} = \frac{\pi}{32}$.

Now, $\Phi_2(X) = X^4 + C_2\Phi_2(X) + C_0\Phi_0(X)$, where

$$\begin{aligned}
\int\limits_{-1}^{1} \frac{(X^4 + C_2\Phi_2(X) + C_0\Phi_0(X)) \cdot \Phi_2(X)\, dX}{\sqrt{1-X^2}} &= \frac{\pi}{8} + \frac{\pi}{8}C_2 = 0 \Rightarrow C_2 = -1 \\
\int\limits_{-1}^{1} \frac{(X^4 + C_2\Phi_2(X) + C_0\Phi_0(X)) \cdot \Phi_0(X)\, dX}{\sqrt{1-X^2}} &= \frac{3\pi}{8} + \pi C_0 = 0 \Rightarrow C_0 = -\frac{3}{8}
\end{aligned}$$

resulting in

$$\Phi_4(X) = X^4 - (X^2 - \frac{1}{2}) - \frac{3}{8} = X^4 - X^2 + \frac{1}{8}$$

and $\alpha_4 = \int_{-1}^{1} \frac{(X^4-X^2+\frac{1}{8})^2 dX}{\sqrt{1-X^2}} = \frac{\pi}{128}$.

Etc.

The resulting polynomials are called CHEBYSHEV, they also have a variety of other applications.

Example: Repeating the previous example using Chebyshev's polynomials (rather than Legendre's) yields

$$\begin{aligned}
\hat{a} &= \int_{-1}^{1} \frac{\sin[\frac{\pi}{4}(1+X)]\, dX}{\sqrt{1-X^2}} \div \pi = 0.60219\,4701 \\
\hat{b} &= \int_{-1}^{1} \frac{X \sin[\frac{\pi}{4}(1+X)]\, dX}{\sqrt{1-X^2}} \div \frac{\pi}{2} = 0.51362\,51666 \\
\hat{c} &= \int_{-1}^{1} \frac{(X^2-\frac{1}{2}) \sin[\frac{\pi}{4}(1+X)]\, dX}{\sqrt{1-X^2}} \div \frac{\pi}{8} = -0.20709\,26885 \\
\hat{d} &= \int_{-1}^{1} \frac{(X^3-\frac{3}{4}X) \sin[\frac{\pi}{4}(1+X)]\, dX}{\sqrt{1-X^2}} \div \frac{\pi}{32} = -5.49281\,3693 \times 10^{-2}
\end{aligned}$$

resulting in

$$\begin{aligned}
&0.60219\,4701 + 0.51362\,51666\,X - 0.20709\,26885(X^2 - \tfrac{1}{2}) \\
&-5.49281\,3693 \times 10^{-2}(X^3 - \tfrac{3}{4}X)
\end{aligned}$$

having the typical error of

$$\sqrt{\frac{\int_{-1}^{1} \frac{\sin[\frac{\pi}{4}(1+X)]^2 dX}{\sqrt{1-X^2}} - \pi \cdot \hat{a}^2 - \frac{\pi}{2} \cdot \hat{b}^2 - \frac{\pi}{8} \cdot \hat{c}^2 - \frac{\pi}{32} \cdot \hat{d}^2}{\pi}} = 0.000964$$

Even though this is larger than what we got with Legendre polynomials, the overall fit (in terms of its maximum error) is much better, as we will see shortly.

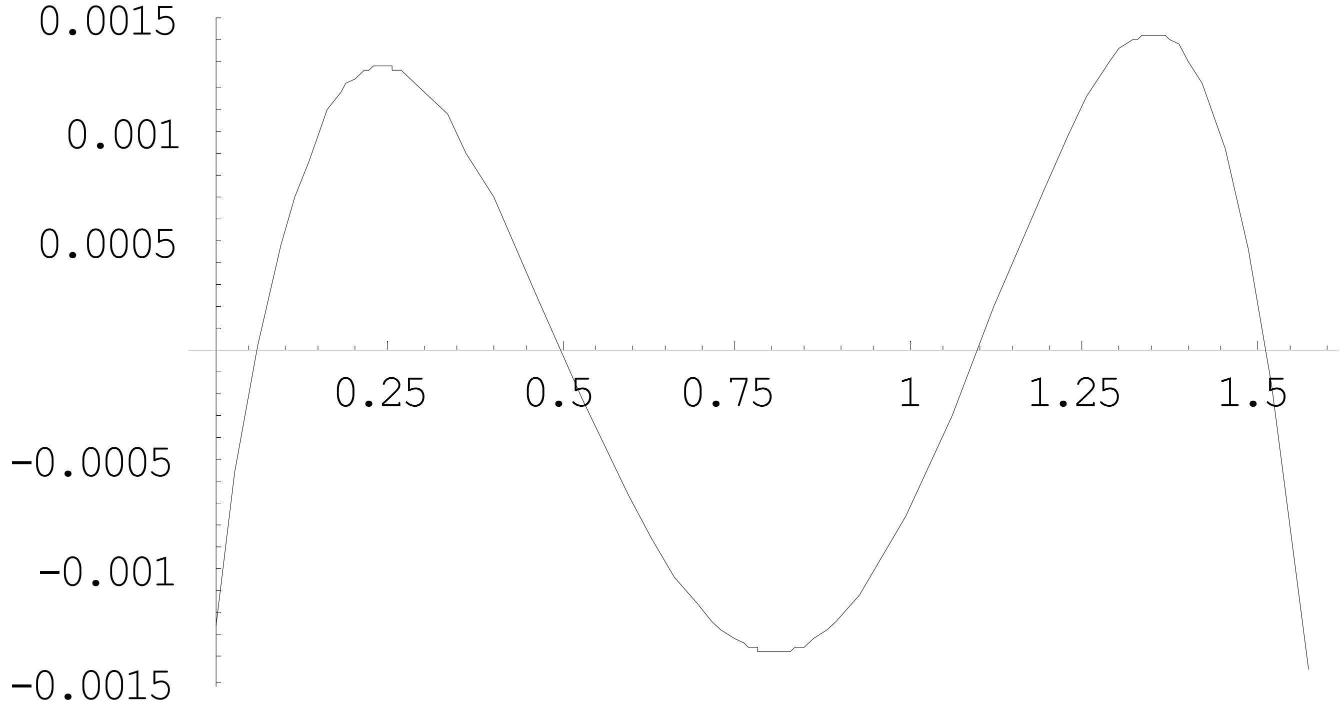

Figure 2

Replacing X by $\frac{4}{\pi} \cdot x - 1$ (this part is the same as before) and expanding the answer, yields

$$-1.24477\,557 \times 10^{-3} + 1.02396\,7553x - 0.068587\,5963x^2 - 0.11337\,70686x^3$$

Plotting the difference between this approximation and $\sin x$ clearly shows that the largest error has now been reduced to what appears to leave little room for improvement:

Laguerre and Hermite Polynomials

As long as the weight function (always non-negative) can be multiplied by any integer power of X and integrated over the chosen interval (yielding a *finite* value), one can construct the corresponding orthogonal set of polynomials. This is true even when the interval itself is infinite. We will go over two interesting and important examples:

Laguerre

The interval is $[0, \infty)$ and $W(x) = e^{-x}$. The polynomials (we will call them $L_0(x)$, $L_1(x)$, etc.), are to be orthogonal in the following sense:

$$\int_0^\infty e^{-x} L_i(x) L_j(x)\, dx = 0$$

whenever $i \neq j$. Using Gram-Schmidt (and the following helpful formula: $\int_0^\infty e^{-x} x^n dx = n!$), we get:

$L_0(x) = 1$ with $\alpha_0 = 1$, $L_1(x) = x + c$ (there is no symmetry now, we have to use all possible powers of x) where

$$\int_0^\infty e^{-x}(x+c)\,dx = 1 + c = 0 \Rightarrow c = -1 \Rightarrow L_1(x) = x - 1$$

with $\alpha_1 = \int_0^\infty e^{-x}(x-1)^2\,dx = 1$.

Similarly, $L_2(x) = x^2 + c_1 L_1(x) + c_0 L_0(x)$, so that

$$\begin{aligned}\int_0^\infty e^{-x}\left(x^2 + c_1L_1(x) + c_0L_0(x)\right)L_0(x)\;dx &= 2 + c_0 \Rightarrow c_0 = -2\\ \int_0^\infty e^{-x}\left(x^2 + c_1L_1(x) + c_0L_0(x)\right)L_1(x)\,dx &= 6 - 2 + c_1 \Rightarrow c_1 = -4\end{aligned}$$

implying that

$$L_2(x) = x^2 - 4(x-1) - 2 = x^2 - 4x + 2$$

with $\alpha_2 = \int_0^\infty e^{-x}(x^2 - 4x + 2)^2\,dx = 4$.

And, one more: $L_3(x) = x^3 + c_2L_2(x) + c_1L_1(x) + c_0L_0(x)$, where

$$\begin{aligned}\int_0^\infty e^{-x}\left(x^3 + c_2L_2(x) + c_1L_1(x) + c_0L_0(x)\right)L_0(x)\;dx &= 6 + c_0 \Rightarrow c_0 = -6\\ \int_0^\infty e^{-x}\left(x^3 + c_2L_2(x) + c_1L_1(x) + c_0L_0(x)\right)L_1(x)\;dx &= 24 - 6 + c_1 \Rightarrow c_1 = -18\\ \int_0^\infty e^{-x}\left(x^3 + c_2L_2(x) + c_1L_1(x) + c_0L_0(x)\right)L_2(x)\;dx &= 120 - 96 + 12 + 4c_2 \Rightarrow c_2 = -9\end{aligned}$$

resulting in

$$L_3(x) = x^3 - 9(x^2 - 4x + 2) - 18(x-1) - 6 = x^3 - 9x^2 + 18x - 6$$

and $\alpha_3 = \int_0^\infty e^{-x}(x^3 - 9x^2 + 18x - 6)^2\,dx = 36$.

These are the so called LAGUERRE POLYNOMIALS.

Hermite

Finally, we will consider the interval of all real numbers, with $W(x) = e^{-x^2}$. Utilizing the $W(x) = W(-x)$ symmetry, we get:

$H_0(x) = 1$ with $\alpha_0 = \int_{-\infty}^{\infty} e^{-x^2}dx = \sqrt{\pi}$, $H_1(x) = x$ and $\alpha_1 = \int_{-\infty}^{\infty} x^2 e^{-x^2}dx = \frac{\sqrt{\pi}}{2}$, $H_2(x) = x^2 + c$ where

$$\int_{-\infty}^{\infty}(x^2 + c)\,e^{-x^2}dx = \frac{\sqrt{\pi}}{2} + c\sqrt{\pi} = 0 \Rightarrow c = -\frac{1}{2} \Rightarrow H_2(x) = x^2 - \frac{1}{2}$$

and $\alpha_2 = \int_{-\infty}^{\infty} (x^2 - \frac{1}{2})^2 e^{-x^2} dx = \frac{\sqrt{\pi}}{2}$, $H_3(x) = x^3 + c\,x$ so that

$$\int_{-\infty}^{\infty} (x^3 + c\,x)\,x\,e^{-x^2} dx = \frac{3\sqrt{\pi}}{4} + c\frac{\sqrt{\pi}}{2} = 0 \Rightarrow c = -\frac{3}{2} \Rightarrow H_3(x) = x^3 - \frac{3}{2}x$$

with $\alpha_3 = \int_{-\infty}^{\infty} (x^3 - \frac{3}{2}x)^2 e^{-x^2} dx = \frac{3}{4}\sqrt{\pi}$, etc. These polynomials are called HERMITE.

For our purpose (of approximating functions), neither set of these orthogonal polynomials is as important as the previous two (Legendre and Chebyshev), but they are very useful when studying Differential Equations, etc. To us, deriving them was just an exercise in Gram-Schmidt - given an interval and a suitable weight function, we have to be able to construct the first few correspondingly orthogonal polynomials.

Chapter 7 NUMERICAL INTEGRATION

As we all know, integrating a given function may often be impossible *analytically*; one can always (assuming that the answer is finite - something we can tell in advance) carry out the corresponding integration *numerically*, to any desired accuracy. There are several techniques for doing this, in this chapter we will study most of them.

The basic idea is to replace the function to be integrated (the INTEGRAND) by a 'closely fitting' polynomial, integrating the polynomial instead (always quite trivial). The main thing to decide is: how to construct a 'closely fitting' polynomial, to a given function.

We have spent the previous chapter doing exactly that, so we may fee that we have the answer. WRONG, that approach is not going to work now. The reason is that, to fit the 'best' polynomial, we first had to compute a whole bunch of integrals, including the integral whose value we are now trying to approximate. So that would amount to circular reasoning.

We now have to go back to the simpler approach of an earlier chapter, where we would first evaluate a function at a few values of x (the so called NODES), and then do Lagrange (not Legendre) -type interpolation. So now the only choice to make is: How many nodes, and where?

Trapezoidal rule

We will start 'easy', with only two nodes, chosen (rather arbitrarily, but sensibly) at *each end* of the integration interval, where the function will be evaluated. The corresponding interpolating polynomial will be a simple straight line connecting the two points thus obtained.

Specifically, to approximate $\int_A^B y(x)\,dx$, we evaluate $y(x)$ at A and B, fit the interpolating straight line

$$y(A)\,\frac{x-B}{A-B}+y(B)\,\frac{x-A}{B-A}$$

and integrate it instead

$$\begin{aligned}
&\frac{y(A)}{A-B}\int\limits_A^B (x-B)\,dx+\frac{y(B)}{B-A}\int\limits_A^B (x-A)\,dx \\
&=\frac{y(A)}{A-B}\left(\frac{x^2}{2}-Bx\right)_{x=A}^{B}+\frac{y(B)}{B-A}\left(\frac{x^2}{2}-Ax\right)_{x=A}^{B} \\
&=\frac{y(A)}{A-B}\left(\frac{B^2-A^2}{2}-B(B-A)\right)+\frac{y(B)}{B-A}\left(\frac{B^2-A^2}{2}-A(B-A)\right) \\
&=-\,y(A)\left(\frac{B+A}{2}-B\right)+y(B)\left(\frac{B+A}{2}-A\right)=\frac{y(A)+y(B)}{2}\cdot(B-A)
\end{aligned}$$

(this can be easily verified geometrically to be the area of the resulting trapezoid).

The resulting 'trapezoidal' rule is thus

$$\int_A^B y(x)\,dx\simeq\frac{y(A)+y(B)}{2}\cdot(B-A)$$

To estimate the error of the last formula, we (Taylor) expand $y(x)$ at $x \equiv x_c = \frac{A+B}{2}$, thus

$$y(x) = y(x_c)+y'(x_c)(x-x_c)+\frac{y''(x_c)}{2}(x-x_c)^2+\frac{y'''(x_c)}{6}(x-x_c)^3+\frac{y^{iv}(x_c)}{24}(x-x_c)^4+... \tag{7.1}$$

Integrating the right hand side *exactly* yields

$$y(x_c)\cdot(B-A)+y'(x_c)\left.\frac{(x-x_c)^2}{2}\right|_{x=A}^{B}+\frac{y''(x_c)}{2}\left.\frac{(x-x_c)^3}{3}\right|_{x=A}^{B} \tag{7.2}$$

$$+\frac{y'''(x_c)}{6}\left.\frac{(x-x_c)^4}{4}\right|_{x=A}^{B}+\frac{y^{iv}(x_c)}{24}\left.\frac{(x-x_c)^5}{5}\right|_{x=A}^{B}...$$

$$= y(x_c)\,h+\frac{y''(x_c)}{24}h^3+\frac{y^{iv}(x_c)}{1920}h^5+... \tag{7.3}$$

(where $h \equiv B-A$ is the length of the integration interval), whereas, applying the *trapezoidal rule* to it results in

$$y(x_c)\,h+\frac{y''(x_c)}{8}h^3+\frac{y^{iv}(x_c)}{384}h^5+...$$

The difference between the approximate and the exact result is clearly the error of the latter, equal to

$$\frac{y''(x_c)}{12}h^3+\frac{y^{iv}(x_c)}{480}h^5+...$$

Note that the leading term is proportional to the *third* power h; also note that only odd powers of h contribute (this is due to choosing the nodes symmetrically).

In its current form, the trapezoidal rule is clearly too primitive to yield accurate results. For example, applying it to $\int_0^{\pi/2}\sin x\,dx$ (which has the exact value of 1) we get $\frac{1}{2}\cdot\frac{\pi}{2} = 0.785$, off by 21.5%. How could we improve the accuracy of the trapezoidal rule? Since its error is proportional to h^3, we clearly need to reduce that value of h. This can be done by subdividing the original (A, B) interval into n (equal-length) subintervals, and apply the trapezoidal rule individually to each subinterval, adding the results. This leads to the so called

Composite rule

The new value of h is now clearly equal to $\frac{B-A}{n}$, with the nodes at $x_0 = A$, $x_1 = A+h$, $x_2 = A+2h$, ..., $x_n = B$ (we will call the corresponding values of the integrand y_0, y_1, ..., instead of the full $y(x_0)$, $y(x_1)$, ..., just to simplify our notation). The complete formula will now read

$$\frac{y_0+y_1}{2}h+\frac{y_1+y_2}{2}h+....+\frac{y_{n-1}+y_n}{2}h =$$
$$\frac{y_0+2y_1+2y_2+...+2y_{n-1}+y_n}{2n}\cdot(B-A)$$

where the first term is a weighted average of the y_i values, the endpoints taken only half as 'seriously' as the rest. This is called the COMPOSITE TRAPEZOIDAL RULE. The overall error of the result will be

$$\frac{h^3}{12}\sum_{i=1}^{n}y''\left(\tfrac{x_i+x_{i-1}}{2}\right)$$

which will tend (as n increases) to

$$\frac{h^3}{12} \cdot n\, y''_{av} + ... = h^2 \frac{B-A}{12}\, y''_{av} + ...$$

where y''_{av} is the average value of $y''(x)$ over the (A,B) interval. Since both y''_{av} and $\frac{B-A}{12}$ are fixed, we can see that the error will be proportional to h^2 (the higher order terms proportional to h^4, h^6, etc.). Later on this will give us an idea of how to further improve the results.

Example: When we apply this composite rule to $\int_0^{\pi/2} \sin x\, dx$ using $n = 1$, 2, 4, 8, 16, 32 and 64 (it is convenient to double the value of n each step; this means we can still utilize the previously computed values of the function). we get:

n	$\frac{\frac{\pi}{2}\left(1+2\sum_{i=1}^{n-1}\sin(\frac{\pi}{2}\frac{i}{n})\right)}{2n}$	error
1	0.78539 81635	0.2146
2	0.94805 9449	0.0519
4	0.98711 5801	0.0129
8	0.99678 5172	0.0032
16	0.99919 66805	0.0008
32	0.99979 91945	0.0002
64	0.99994 98	0.00005

Note that, with $n = 64$, we have achieved a 4 digit accuracy. Also note that the error is reduced, in each step, roughly by a factor of four.

Romberg integration

Realizing that the error of the composite rule follows a regular pattern (i.e. is reduced by 4 when doubling n), we can write $I_0 = I + \frac{c}{4^0}$, $I_1 = I + \frac{c}{4^1}$, $I_2 = \frac{c}{4^2}$, $I_3 = \frac{c}{4^3}$, ..., where I_i is the result of the composite rule with $n = 2^i$ subintervals and I is the exact answer. We can now eliminate the error term from any two such consecutive results, i.e.

$$\begin{aligned} I_i &= I + \frac{c}{4^i} \\ I_{i+1} &= I + \frac{c}{4^{i+1}} \end{aligned}$$

and solve for I, by

$$I = \frac{4I_{i+1} - I_i}{3}$$

This will result in a substantially more accurate answer for I (still not exact though, we know that there are additional error terms proportional to n^4, n^6, etc.).

Example: Using our previous example, we get

i	$J_i \equiv \frac{4I_{i+1}-I_i}{3}$
1	1.00227 9878
2	1.00013 4585
3	1.00000 8296
4	1.00000 0517
5	1.00000 0033
6	1.00000 0002

The errors are now a lot smaller (we have reached nearly a 10 digit accuracy), furthermore, they decrease, in each step, by roughly a factor of 16.

Getting the new column of more accurate answers (J_i), and realizing that their errors are now reduced by a factor of 16 in each step, we can continue the idea of eliminating them, by computing

$$K_i \equiv \frac{16J_{i+1} - J_i}{15}$$

The K's errors will decrease by a factor of 64, so we can improve them further, by

$$L_i \equiv \frac{64K_{i+1} - K_i}{63}$$

etc., until we end up with only one number (or the values no longer change, within the computational accuracy).

Example: Continuing the previous example:

i	J_i	$K_i = \frac{16J_{i+1}-J_i}{15}$	$L_i = \frac{64K_{i+1}-K_i}{63}$
1	1.00227 9878	0.99999 15654	1.00000 0009
2	1.00013 4585	0.99999 98774	0.99999 99997
3	1.00000 8296	0.99999 9998	
4	1.00000 0517		

at which point we have clearly reached the 10-digit limit imposed by Maple (which, by default, evaluates all results to this accuracy only - this can be increased by setting **Digits** := 20 ;). Note that we have reached a nearly 10 digit accuracy with 16 evaluations of the function gone through two stages of Romberg (K_3), or 8 evaluations with three Romberg stages (L_1).

Once we reach the limit of Maple's accuracy, the results start to deteriorate (whether we increase i, or go to the next stage of Romberg), so we should not overdo it (ultimately, monitoring convergence, and imposing some sensible 'stopping' rule, should be incorporated into the procedure - we will not go into this).

Simpson Rule

Another idea to improve the basic trapezoidal rule is simply to increase the number of nodes (from 2 to 3). The most logical choice for the extra (third) node is to put it at the center of the (A, B) interval. The corresponding interpolating polynomial is the following quadratic

$$\frac{(x - \frac{A+B}{2})(x - B)}{\frac{A-B}{2}(A - B)}\, y(A) + \frac{(x - A)(x - B)}{\frac{B-A}{2} \cdot \frac{A-B}{2}}\, y(\tfrac{A+B}{2}) + \frac{(x - A)(x - \frac{A+B}{2})}{(B - A)\frac{B-A}{2}}\, y(B)$$

To integrate over x from A to B, we first evaluate (using by-part integration)

$$\begin{aligned}
\int_A^B (x-\tfrac{A+B}{2})(x-B)\,dx &= (x-\tfrac{A+B}{2})\,\tfrac{(x-B)^2}{2}\Big|_{x=A}^{B} - \tfrac{1}{2}\int_A^B (x-B)^2\,dx \\
&= \tfrac{(B-A)^3}{4} + \tfrac{(A-B)^3}{6} = \tfrac{(B-A)^3}{12} \\
\int_A^B (x-A)(x-B)\,dx &= (x-A)\tfrac{(x-B)^2}{2}\Big|_{x=A}^{B} - \tfrac{1}{2}\int_A^B (x-B)^2\,dx \\
&= -\tfrac{(B-A)^3}{6} \\
\int_A^B (x-A)(x-\tfrac{A+B}{2})\,dx &= \tfrac{(x-A)^2}{2}\,(x-\tfrac{A+B}{2})\Big|_{x=A}^{B} - \tfrac{1}{2}\int_A^B (x-A)^2\,dx \\
&= \tfrac{(B-A)^3}{4} - \tfrac{(B-A)^3}{6} = \tfrac{(B-A)^3}{12}
\end{aligned}$$

Putting it together, the interpolating quadratic integrates to

$$\frac{y(A)+4y(\frac{A+B}{2})+y(B)}{6}\cdot(B-A)$$

which is our new approximation to $\int_A^B y(x)\,dx$ called the SIMPSON RULE. Note that the first term represents a weighted average of the three computed values of $y(x)$, the mid point being 4 times as 'heavy' as the two end points.

One can easily verify that the formula yields the exact answer whenever $y(x)$ is a polynomial of degree 2 or less (as ti must), it is actually also correct for all cubics (due to its symmetry), but fails (resulting in a small error) as soon as quartic terms are encountered (we will do this using $A=0$ and $B=1$, which proves to be sufficient):

$y(x)$:	Simpson:	Exact:
1	$\frac{1+4+1}{6}=1$	$\int_0^1 dx=1$
x	$\frac{0+4\cdot\frac{1}{2}+1}{6}=\frac{1}{2}$	$\int_0^1 x\,dx=\frac{1}{2}$
x^2	$\frac{0+4\cdot\frac{1}{4}+1}{6}=\frac{1}{3}$	$\int_0^1 x^2dx=\frac{1}{3}$
x^3	$\frac{0+4\cdot\frac{1}{8}+1}{6}=\frac{1}{4}$	$\int_0^1 x^3dx=\frac{1}{4}$
x^4	$\frac{0+4\cdot\frac{1}{16}+1}{6}=\frac{5}{24}$	$\int_0^1 x^4dx=\frac{1}{5}$

Note that the first four answers are correct, but the last one is off by about 4%.

This may actually give us an idea as to how to derive the Simpson-rule coefficients more efficiently: Knowing that the formula has to be exact for all quadratics, and knowing that it will have to have the following form:

$$\left[c_l\,y(A)+c_c\,y(\tfrac{A+B}{2})+c_r\,y(B)\right](B-A)$$

we apply it to $y(x)=1$, x and x^2, using $A=-1$ and $B=1$ (the most convenient choice now), getting

$y(x)$:	Formula:	Exact:
1	$2(c_l+c_c+c_r)$	2
x	$2(c_r-c_l)$	0
x^2	$2(c_r+c_l)$	$\frac{2}{3}$

This implies that

$$\begin{aligned} c_l + c_c + c_r &= 1 \\ c_r - c_l &= 0 \\ c_r + c_l &= \tfrac{1}{3} \end{aligned}$$

which is solved by $c_l = c_r = \frac{1}{6}$, $c_c = \frac{2}{3}$, resulting in the same rule as before (if one remembers that a symmetric choice of nodes must result in symmetric c coefficients, one can proceed even faster).

Error analysis

Applying the Simpson rule to (7.1) yields

$$y(x_c)\,h + \frac{y''(x_c)}{24}\,h^3 + \frac{y^{iv}(x_c)}{1152}h^5 + ...$$

which means that the error of the Simpson rule is

$$\frac{y^{iv}(x_c)}{2880}h^5 + ...$$

(only odd powers of h contributing). This is a *substantial* improvement over the trapezoidal rule. And, true enough, applying Simpson to the old $\int_0^{\pi/2} \sin x\, dx$ yields 1.002279877, a fairly respectable answer (compare this to trapezoidal's 0.785 - even though we are not quite fair, since we now need an extra evaluation).

Composite rule

We can improve the accuracy of the Simpson rule by dividing (A, B) into $\frac{n}{2}$ subintervals (where n is even - here we want to be fair to trapezoidal rule in terms of the total number of function evaluations, which should equal to $n+1$). Applying the Simpson rule individually to each subinterval, and adding the answers, yields

$$\frac{y_0 + 4y_1 + y_2}{6}h + \frac{y_2 + 4y_3 + y_4}{6}h + ... + \frac{y_{n-2} + 4y_{n-1} + y_n}{6}h =$$
$$\frac{y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + ... + 2y_{n-2} + 4y_{n-1} + y_n}{3n} \cdot (B - A)$$

since now $h = \frac{B-A}{\frac{n}{2}} = \frac{2}{n}(B - A)$. The error of this composite rule is qual to

$$\frac{h^5}{2800}\sum_{i=1}^{n/2} y^{iv}(x_{2i-1}) + ... \simeq \frac{h^5}{2800} \cdot \frac{n}{2}\, y^{iv}_{av} + ... = \frac{h^4}{2800}\,(B - A)\, y^{iv}_{av} + ...$$

where the next term would be proportional to h^6, h^8, etc. This should give us a clear idea as to how to apply Romberg algorithm to improve consecutive results of the composite rule.

Example: Applying the Simpson composite rule to $\int_0^1 \sin x\, dx$ yields

n	$\frac{\frac{\pi}{2}\left(1+4\sum_{i=1}^{n/2} \sin\frac{\pi\,(2i-1)}{2\,n}+2\sum_{i=1}^{n/2-1}\sin\frac{\pi\,(2i)}{2\,n}\right)}{3n}$	$K_i = \frac{16I_{i+1}-I_i}{15}$	$L_i = \frac{64K_{i+1}-K_i}{63}$
2	1.00227 9878	0.99999 15654	1.00000 0009
4	1.00013 4585	0.99999 98774	0.99999 99997
8	1.00000 8296	0.99999 9998	
16	1.00000 0517		

Note that:

1. the error of the composite rule decreased, in each step, roughly by a factor of 16,
2. we skipped (as we must) the J_i stage of Romberg.

Other Rules

Going beyond trapezoidal (two-node) and Simpson (three-node) rule, we now learn to design our own rules, using any number of nodes.

Example: The most 'natural' choice of 4 nodes is at A, $A+\frac{B-A}{3}$, $B-\frac{B-A}{3}$, B (equidistant). Incorporating the symmetry of this choice, the corresponding rule will have the form of

$$\left[c_e\, y(A) + c_i\, y(A+\tfrac{B-A}{3}) + c_i\, y(B-\tfrac{B-A}{3}) \;+ c_e\, y(B)\right](B-A)$$

So, it now boils down to finding the value of c_e and c_i. We can do this most efficiently by taking $A=-1$ and $B=1$ ($A+\frac{B-A}{3}=-\frac{1}{3}$, $B-\frac{B-A}{3}=\frac{1}{3}$), and making the rule correct with $y(x)=$ and $y(x)=x^2$ (odd powers integrate to zero automatically), namely:

$$\begin{aligned}(2c_e+2c_i)\cdot 2 &= 2\\ (2c_e+2\tfrac{c_i}{9})\cdot 2 &= \tfrac{2}{3}\end{aligned}$$

which yields $c_e=\frac{1}{2}-c_i$, $1-2c_i+\frac{2}{9}c_i=\frac{1}{3}\Rightarrow c_i=\frac{3}{8}$, $c_e=\frac{1}{8}$. The resulting rule is thus:

$$\int_A^B y(x)\,dx \simeq \frac{y(A)+3\,y(A+\frac{B-A}{3})+3\,y(B-\frac{B-A}{3})\;+\;y(B)}{8}(B-A)$$

Dividing (A,B) into $\frac{n}{4}$ subintervals, we can now construct the corresponding composite rule. ....

In our next example, we try a different (yet sensible) choice of nodes.

Example: Going back to 3 nodes, we divide (A,B) into three equal subintervals and place a node in the middle of each, thus: $x_1=A+\frac{B-A}{6}$, $x_2=\frac{A+B}{2}$, $x_3=B-\frac{B-A}{6}$. The resulting formula has to read

$$(c_s\, y_1 + c_c y_2 + c_s y_3)(B-A)$$

where $y_1\equiv y(x_1)$, $y_2\equiv y(x_2)$, ... Using the same approach as before (now $x_1=-\frac{2}{3}$, $x_2=0$, $x_3=\frac{2}{3}$) we get

$$\begin{aligned}(2c_s+c_c)\cdot 2 &= 2\\ 2\cdot\tfrac{4}{9}\,c_s\cdot 2 &= \tfrac{2}{3}\end{aligned}$$

implying that $c_s = \frac{3}{8}$ and $c_c = \frac{2}{8}$, i.e.

$$\int_A^B y(x)\,dx \simeq \frac{3\,y_1 + 2\,y_2 + 3\,y_3}{8}\,(B-A)$$

Based on (**??**) and (7.2), the error of this rule is

$$\left(y(x_c)\,h + \frac{y''(x_c)}{24}h^3 + \frac{y^{iv}(x_c)}{2592}h^5 + ...\right) - \left(y(x_c)\,h + \frac{y''(x_c)}{24}h^3 + \frac{y^{iv}(x_c)}{1920}h^5 + ...\right)$$
$$= \ -\frac{7\,y^{iv}(x_c)}{51840}\,h^5 + ...$$

i.e. *smaller* (in terms of its leading term) than that of the Simpson rule, and of the opposite sign. This makes us think: Is there a way of eliminating the leading term entirely, by yet another choice of nodes (which seem very likely now), and how would we do it? We will take up this issue in a subsequent section.

We will go over one more example of constructing our own rule, based on a choice of nodes which is no longer symmetrical. Under the circumstances, such a choice would be rather unusual and inconvenient, but we need to learn how to deal with that possibility as well (what we lose is the symmetry of the corresponding weights, i.e. the c coefficients).

Example: Rather than deriving a general rule applicable to any A to B integral, we derive a formula to approximate

$$\int_0^3 y(x)\,dx$$

assuming that only three values of $y(x)$ are known, $y(0)$, $y(1)$ and $y(3)$. The formula will have the form of

$$c_0\,y(0) + c_1\,y(1) + c_3\,y(3)$$

where the values of the c coefficients will be computed by making the formula exact for $y(x) = 1$, $y(x) = x$ and $y(x) = x^2$ (now, we cannot skip odd powers!), namely:

$$c_0 + c_1 + c_3 = \int_0^3 dx = 3$$
$$c_1 + 3\,c_3 = \int_0^3 x\,dx = \frac{9}{2}$$
$$c_1 + 9\,c_3 = \int_0^3 x^2\,dx = 9$$

The last two equations can be solved for c_1 and c_3 by

$$\begin{bmatrix} 9 & -3 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \frac{9}{2} \\ 9 \end{bmatrix} \div 6 = \begin{bmatrix} \frac{9}{4} \\ \frac{3}{4} \end{bmatrix}$$

implying (based on the first equation) that $c_0 = 0$. The final rule is thus

$$\int_0^3 y(x)\,dx \simeq \frac{9}{4}\,y(1) + \frac{3}{4}\,y(3)$$

Incidentally, this example also demonstrates that is possible to get a three-point formula for the price of two function evaluations!

Being able to design our own formulas is important for the following reason:

Singular and improper integrals

All the formulas we derived so far will fail miserably when evaluating an integral like

$$\int_0^1 \frac{\exp(x^2)}{\sqrt{x}}\,dx$$

which exists, having a finite value. The reason is that the integrand, when evaluated at $x = 0$, has an infinite value, so most of our approximate formulas would simply 'blow up'. Even those formulas which do not use the left end point would perform rather badly, their errors would be quite large and would not follow the usual pattern (Romberg integration would not work). The old formulas implicitly assumed that the integrand is not just continuous (and therefore finite) throughout the $[A, B]$ interval, but also that all of its derivative are (this means, we would have difficulty not only when dividing by $\sqrt{x}$ at $x = 0$, but also when multiplying by $\sqrt{x}$, since its *derivatives* become singular).

There are two ways of resolving this, one is analytical (not always available, and not of the main interest to us in any case), which removes the singularity by some smart change of variable (such as $z = \sqrt{x}$ in the above case). The other one would 'remove' the singularity by redesigning our formulas of numerical integration along the following lines:

The actual singularity (in the original example) is due to the $\frac{1}{\sqrt{x}}$ part of the integrand. Even though creating such a havoc numerically, the $\frac{1}{\sqrt{x}}$ function is quite trivial to integrate analytically (right?), so we will separate it from the rest of the integrand, thus:

$$\int_0^1 \frac{y(x)}{\sqrt{x}}\,dx$$

where $y(x)$ can now be arbitrary. We can now derive an approximate formula for this kind of integral in the usual fashion: select a few nodes in $(0, 1)$, fit the corresponding interpolating polynomial to $y(x)$ - leaving the $\frac{1}{\sqrt{x}}$ part out of it, and the integrate$\simeq$

$$\int_0^1 \frac{p(x)}{\sqrt{x}}\,dx$$

instead (quite trivial now), where $p(x)$ is the interpolating polynomial.

Example: Using $x = 0$, $\frac{1}{2}$ and 1 as our nodes, we get

$$p(x) = \frac{(x-\frac{1}{2})(x-1)}{\frac{1}{2}}\,y(0) + \frac{x\,(x-1)}{-\frac{1}{4}}\,y(\tfrac{1}{2}) + \frac{x\,(x-\frac{1}{2})}{\frac{1}{2}}\,y(1)$$

Since

$$\begin{aligned}
\int_0^1 \frac{(x-\frac{1}{2})(x-1)}{\sqrt{x}}\,dx &= \frac{2}{5} \\
\int_0^1 \frac{x\,(x-1)}{\sqrt{x}}\,dx &= -\frac{4}{15} \\
\int_0^1 \frac{x\,(x-\frac{1}{2})}{\sqrt{x}}\,dx &= \frac{1}{15}
\end{aligned}$$

our integration rule reads:

$$\int_0^1 \frac{y(x)}{\sqrt{x}}\,dx \simeq \frac{12}{15}\,y(0) + \frac{16}{15}\,y(\tfrac{1}{2}) + \frac{2}{15}\,y(1)$$

The rule is exact whenever $y(x)$ is a polynomial of degree two or less.

This gives us yet another (more expedient) way of finding the coefficients: Writing the rule as

$$\int_0^1 \frac{y(x)}{\sqrt{x}}\,dx \simeq c_0\,y(0) + c_{\frac{1}{2}}\,y(\tfrac{1}{2}) + c_1\,y(1)$$

(note that c_0 no longer equals c_1, $\frac{1}{\sqrt{x}}$ 'breaks' the symmetry), we make it correct using $y(x) = 1$, x and x^2:

$$\begin{aligned}
c_0 + c_{\frac{1}{2}} + c_1 &= \int_0^1 \frac{1}{\sqrt{x}}\,dx = 2 \\
\frac{1}{2}c_{\frac{1}{2}} + c_1 &= \int_0^1 \frac{x}{\sqrt{x}}\,dx = \frac{2}{3} \\
\frac{1}{4}c_{\frac{1}{2}} + c_1 &= \int_0^1 \frac{x^2}{\sqrt{x}}\,dx = \frac{2}{5}
\end{aligned}$$

resulting in $c_{\frac{1}{2}} = 4(\frac{2}{3} - \frac{2}{5}) = \frac{16}{15}$, $c_1 = \frac{2}{3} - \frac{8}{15} = \frac{2}{15}$, and $c_0 = 2 - \frac{16}{15} - \frac{2}{15} = \frac{12}{15}$ (check). Applying this rule to the original $\int_0^1 \frac{\exp(x^2)}{\sqrt{x}}\,dx$ results in $\frac{12}{15} + \frac{16}{15}\,e^{\frac{1}{4}} + \frac{2}{15}\,e = 2.532$. This compares favorably (0.4% error) with the exact answer of 2.543.

To extend the formula to $\int_0^A \frac{y(x)}{\sqrt{x}}\,dx$, we introduce $z = A\,x$ and write

$$\int\limits_0^A \frac{y(x)}{\sqrt{x}}\,dx = \int\limits_0^1 \frac{y(\frac{z}{A})}{\sqrt{\frac{z}{A}}}\,\frac{dz}{A} \simeq \left[\frac{12}{15}\,y(0) + \frac{16}{15}\,y(\tfrac{A}{2}) + \frac{2}{15}\,y(A)\right]\cdot\sqrt{A}$$

This can be used to develop the corresponding composite rule: Divide the original interval into several subintervals, apply the last formula to the first of these subintervals, the usual Simpson rule to the rest (no singularity there). We don't have time to go into details.

Similarly, our standard rule would not work with an integral like $\int_0^\infty \frac{\exp(-x)}{x+2}\,dx$, where the length of the interval is infinite. Again, we can fix this either analytically (change of variable), or by selecting a few nodes in the $(0,\infty)$ interval and approximating

$$\int\limits_0^\infty y(x)\exp(-x)\,dx$$

by $\int_0^\infty p(x)\exp(-x)\,dx$, where $p(x)$ is the interpolating polynomial to the $y(x)$ function (note that, similarly to the previous case, we separated $\exp(-x)$ from the rest of the integrand, so that the actual integration becomes possible).

Example: We choose $x = 0$, 1, 2 and 3 as our four nodes. The interpolating polynomial is thus

$$\frac{(x-1)(x-2)(x-3)}{-6}\,y(0) + \frac{x\,(x-2)(x-3)}{2}\,y(1) +$$
$$+\frac{x\,(x-1)(x-3)}{-2}\,y(2) + \frac{x\,(x-1)(x-2)}{6}\,y(3) =$$
$$\frac{x^3-6x^2+11x-6}{-6}\,y(0) + \frac{x^3-5x^2+6x}{2}\,y(1) +$$
$$+\frac{x^3-4x^2+3x}{-2}\,y(2) + \frac{x^3-3x^2+2x}{6}\,y(3)$$

Multiplying by e^{-x} and integrating from 0 to ∞ yields (remember that $\int_0^\infty x^k e^{-x}dx = k!$):

$$\frac{6-12+11-6}{-6}\,y(0) + \frac{6-10+6}{2}\,y(1) + \frac{6-8+3}{-2}\,y(2) + \frac{6-6+2}{6}\,y(3) =$$
$$\tfrac{1}{6}\,y(0) + y(1) - \tfrac{1}{2}\,y(2) + \tfrac{1}{3}\,y(3)$$

which is our final rule for approximating $\int_0^\infty y(x)\exp(-x)\,dx$. Note that, this time, one of our coefficients is negative. This can happen, but it is usually an indication of badly chosen nodes (we would like the coefficients to be all positive, and of similar size). Applied to the original $\int_0^\infty \frac{\exp(-x)}{x+2}\,dx$. our formula yields: $\frac{1}{6}\cdot\frac{1}{2}+\frac{1}{3}-\frac{1}{2}\cdot\frac{1}{4}+\frac{1}{3}\cdot\frac{1}{5} = 0.358\bar{3}$, reasonably close (0.8% error) to the exact answer of 0.3613.

The obvious questions to ask are: Is there a better way of selecting our four nodes? Is there a *best* way of selecting them? 'Best' in what sense?

Gaussian Integration

We will first try to answer these questions in the context of approximating the basic

$$\int_{-1}^{1} y(x)\,dx$$

We know that any three (four, five)-node formula will be exact for all quadratic (cubic, quartic, ...) polynomials, but if we are lucky (e.g. Simpson's rule), we can go higher than this. How high can we go?

The answer is this: If we choose n nodes to be the roots of the n degree Legendre polynomial (one can show that they all must be in the -1 to 1 range), the corresponding rule (called GAUSSIAN) will be exact for all polynomials of degree $\leq 2n-1$ (instead of the usual $n-1$). This also pushes the order of the leading error term from h^{n+1} to h^{2n+1} - a very substantial improvement!

Proof: We know that $\phi_n(x)$ is orthogonal to $\phi_0(x)$, $\phi_1(x)$, ...$\phi_{n-1}(x)$. Since any polynomial, say $q(x)$, of degree $n-1$ (or less) can be written as a linear combination of these, $q(x)$ itself must be orthogonal to $\phi_n(x)$, i.e.

$$\int_{-1}^{1} q(x)\cdot\phi_n(x)\,dx = 0$$

Let $p(x)$ be an arbitrary polynomial of degree smaller than $2n$. Dividing it by $\phi_n(x)$ (synthetic division!) will yield a quotient $q(x)$ and a remainder $r(x)$, both of degree smaller than n (even though for different reasons). One can thus write

$$p(x) = q(x)\cdot\phi_n(x) + r(x)$$

The exact integration of $p(x)$ is thus equal to

$$\int_{-1}^{1} r(x)\,dx$$

since the first term integrates to 0, as we already know. Applying the corresponding Gaussian rule to $\int_{-1}^{1} p(x)\,dx = \int_{-1}^{1} q(x)\cdot\phi_n(x)\,dx + \int_{-1}^{1} r(x)\,dx$ will also yield the exact answer of 0 for the first term (since $\phi_n(x)$ evaluates to 0 at all nodes!), and the exact answer for the second term (since the degree of $r(x)$ is less than n, and *any* n-node rule is exact in that case). □

Example: To derive a three-node formula we first find the roots of $\phi_3(x) = x^3 - \frac{3}{5}x$. These are clearly: $x_1 = -\sqrt{\frac{3}{5}}$, $x_2 = 0$ and $x_3 = \sqrt{\frac{3}{5}}$. The actual real will read

$$2\cdot[c_1\,y(x_1) + c_2\,y(x_2) + c_1\,y(x_3)]$$

($c_3 = c_1$ due to symmetry). We make the rule exact with $y = 1$ and $y = x^2$:

$$\begin{aligned} 4c_1 + 2c_2 &= 2 \\ 4c_1\cdot\frac{3}{5} &= \frac{2}{3} \end{aligned}$$

implying that $c_1 = \frac{5}{18}$ and $c_2 = \frac{8}{18}$. The final rule is thus

$$2 \cdot \frac{5\,y(x_1) + 8\,y(x_2) + 5\,y(x_3)}{18}$$

One can verify that it is also exact when $y = x^4$: $2 \cdot (5 \cdot \frac{9}{25} + 5 \cdot \frac{9}{25}) \div 18 = \frac{2}{5}$ (it is automatically correct for all odd powers of x). It can be extended to approximate

$$\int\limits_A^B y(x)\,dx \simeq (B-A) \cdot \frac{5\,y(\frac{A+B}{2} - \sqrt{\frac{3}{5}}\frac{B-A}{2}) + 8\,y(\frac{A+B}{2}) + 5\,y(\frac{A+B}{2} + \sqrt{\frac{3}{5}}\frac{B-A}{2})}{18}$$

Applying it to our usual benchmark of $\int_0^{\pi/2} \sin x\,dx$ yields:

$$\frac{\pi}{36} \cdot \left(5\sin[\tfrac{\pi}{4}(1 - \sqrt{\tfrac{3}{5}})] + 8\sin(\tfrac{\pi}{4}) + 5\sin[\tfrac{\pi}{4}(1 + \sqrt{\tfrac{3}{5}})]\right) = 1.00000\,8122$$

a spectacular improvement over Simpson's 1.0022. Furthermore, applying it separately to the $(0, \frac{\pi}{4})$ and $(\frac{\pi}{4}, \frac{\pi}{2})$ subintervals (the composite idea), results in

$$\frac{\pi}{72} \cdot \left(5\sin[\tfrac{\pi}{8}(1 - \sqrt{\tfrac{3}{5}})] + 8\sin(\tfrac{\pi}{8}) + 5\sin[\tfrac{\pi}{8}(1 + \sqrt{\tfrac{3}{5}})]\right) +$$
$$\frac{\pi}{72} \cdot \left(5\sin[\tfrac{\pi}{8}(3 - \sqrt{\tfrac{3}{5}})] + 8\sin(\tfrac{3\pi}{8}) + 5\sin[\tfrac{\pi}{8}(3 + \sqrt{\tfrac{3}{5}})]\right) = 1.00000\,0119$$

a 68 fold increase in accuracy (theory predicts 64). Romberg algorithm further improves the two results to $(64 \times 1.00000\,0119 - 1.00000\,8122) \div 63 = 0.99999\,9992$.

Similarly, we can now find the best (Gaussian) n-node formula to approximate the $\int_0^\infty y(x)\,e^{-x}dx$ integration. An almost identical approach leads to the following prescription:

1. Using Gram-Schmidt, construct a sequence of polynomials (up to degree n) orthogonal in the following sense

$$\int_0^\infty L_i(x)L_j(x)\,e^{-x}dx = 0$$

   when $i \neq j$ (we have already done this, these are the Laguerre polynomials).

2. Find the roots of $L_n(x) = 0$ - they are to be used as nodes.

3. Replace $y(x)$ in the original integral by the corresponding interpolating polynomial and carry out the integration (which is now quite trivial). This yields the desired formula, which is exact when $y(x)$ is a polynomial of order less than $2n$ (rather than the usual n).

Example: When $n = 3$, we first solve

$$x^3 - 9x^2 + 18x - 6 = 0$$

with the help of Maple's **fsolve** command (we will learn how to solve non-linear equations later on), getting

$$\begin{aligned} x_1 &= 0.4157745568 \\ x_2 &= 2.294280360 \\ x_3 &= 6.289945083 \end{aligned}$$

The corresponding interpolating polynomial is

$$p(x) = \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)}\,y(x_1) + \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)}\,y(x_2) + \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)}\,y(x_3)$$

which, when multiplied by e^{-x}, integrates to

$$\frac{2-x_2-x_3+x_2\,x_3}{(x_1-x_2)(x_1-x_3)}\,y(x_1) + \frac{2-x_1-x_3+x_1\,x_3}{(x_2-x_1)(x_2-x_3)}\,y(x_2) + \frac{2-x_1-x_2+x_1\,x_2}{(x_3-x_1)(x_3-x_2)}\,y(x_3) =$$

$$0.71109\,30101\,y(x_1) + 0.27851\,77336\,y(x_2) + 0.010389\,2565\,y(x_3)$$

We can now check that the rule is exact not only for $y(x) = 1$, x and x^2:

$$0.71109\,30101 \;+ 0.27851\,77336 \;+ 0.010389\,2565 \;= 1 \equiv \int_0^\infty e^{-x}dx$$

$$0.71109\,30101\,x_1 + 0.27851\,77336\,x_2 + 0.010389\,2565\,x_3 = 1 \equiv \int_0^\infty x\,e^{-x}dx$$

$$0.71109\,30101\,x_1^2 + 0.27851\,77336\,x_2^2 + 0.010389\,2565\,x_3^2 = 2 \equiv \int_0^\infty x^2e^{-x}dx$$

but also for $y(x) = x^3$, x^4 and x^5:

$$0.71109\,30101\,x_1^3 + 0.27851\,77336\,x_2^3 + 0.010389\,2565\,x_3^3 = 6 \equiv \int_0^\infty x^3e^{-x}dx$$

$$0.71109\,30101\,x_1^4 + 0.27851\,77336\,x_2^4 + 0.010389\,2565\,x_3^4 = 24 \equiv \int_0^\infty x^4\,e^{-x}dx$$

$$0.71109\,30101\,x_1^5 + 0.27851\,77336\,x_2^5 + 0.010389\,2565\,x_3^5 = 120 \equiv \int_0^\infty x^5e^{-x}dx$$

whereas with $y(x) = x^6$ we get

$$0.71109\,30101\,x_1^6 + 0.27851\,77336\,x_2^6 + 0.010389\,2565\,x_3^6 = 684$$

which is 5% off the correct answer of 720.

Applying the formula to $\int_0^\infty \frac{\exp(-x)}{x+2}\,dx$ of one of our previous examples yields

$$\frac{0.71109\,30101}{x_1+2} + \frac{0.27851\,77336}{x_2+2} + \frac{0.010389\,2565}{x_3+2} = 0.3605$$

which is a lot closer (0.2% error) to the correct answer of 0.3613 than the four-node rule derived earlier.

To extend this three-node formula to

$$\int_0^\infty y(x)\, e^{-\beta\, x} dx$$

we introduce $z \equiv \beta\, x$, which enables us to write the integral as

$$\int_0^\infty y(\tfrac{z}{\beta})\, e^{-z} \tfrac{dz}{\beta} \simeq$$
$$\tfrac{1}{\beta} \left[0.71109\,30101\, y(\tfrac{x_1}{\beta}) + 0.27851\,77336\, y(\tfrac{x_{21}}{\beta}) + 0.010389\,2565\, y(\tfrac{x_3}{\beta}) \right]$$

Similarly, we could deal with

$$\int_A^\infty y(x)\, e^{-x} dx$$

where the lower limit is A instead of 0 (try it).

It should now be quite obvious how to construct a Gaussian formula with a given number of nodes to approximate

$$\int_{-1}^{1} \frac{y(x)}{\sqrt{1-x^2}}\, dx$$

using the roots of the corresponding Hermite polynomial for nodes.

The situation is slightly more difficult in a case like

$$\int_0^1 \frac{y(x)}{\sqrt{x}}\, dx$$

where we don't have the corresponding set of orthogonal polynomials as yet. This means that the task of designing a Gaussian formula for approximating the above integral will first require constructing these, using Gram-Schmidt.

Example: To design a modest two-node Gaussian formula to approximate the previous integral, we need to find the first three polynomials orthogonal in the following sense

$$\int_0^1 \frac{\phi_i(x)\phi_j(x)}{\sqrt{x}}\, dx = 0$$

when $i \neq j$. We have practiced this enough, so we skip the details, quoting the results only:

$$\begin{aligned}
\phi_0(x) &= 1 \\
\phi_1(x) &= x - \frac{1}{3} \\
\phi_2(x) &= x^2 - \frac{6}{7}x + \frac{3}{35}
\end{aligned}$$

But, to be on the save side, we do verify that they are correct:

$$\begin{aligned}
\int_0^1 \frac{\phi_0(x)\phi_1(x)}{\sqrt{x}}\,dx &= \frac{1}{\frac{3}{2}} - \frac{\frac{1}{3}}{\frac{1}{2}} = 0 \\
\int_0^1 \frac{\phi_0(x)\phi_2(x)}{\sqrt{x}}\,dx &= \frac{1}{\frac{5}{2}} - \frac{\frac{6}{7}}{\frac{3}{2}} + \frac{\frac{3}{35}}{\frac{1}{2}} = 0 \\
\int_0^1 \frac{\phi_1(x)\phi_2(x)}{\sqrt{x}}\,dx &= \frac{1}{\frac{7}{2}} - \frac{\frac{6}{7}}{\frac{5}{2}} + \frac{\frac{3}{35}}{\frac{3}{2}} - \frac{1}{3}\left(\frac{1}{\frac{5}{2}} - \frac{\frac{6}{7}}{\frac{3}{2}} + \frac{\frac{3}{35}}{\frac{1}{2}}\right) = 0
\end{aligned}$$

Our two nodes will thus be

$$\begin{aligned}
x_1 &= \frac{3}{7} - \sqrt{\frac{9}{49} - \frac{3}{35}} = \frac{3}{7} - \frac{2}{35}\sqrt{30} \\
x_2 &= \frac{3}{7} + \sqrt{\frac{9}{49} - \frac{3}{35}} = \frac{3}{7} + \frac{2}{35}\sqrt{30}
\end{aligned}$$

the interpolating polynomial is $p(x) = \frac{x - x_2}{x_1 - x_2}\, y(x_1) + \frac{x - x_1}{x_2 - x_1}\, y(x_2)$ which, when divided by $\sqrt{x}$, integrates to

$$\frac{\frac{2}{3} - 2x_2}{x_1 - x_2}\, y(x_1) + \frac{\frac{2}{3} - 2x_1}{x_2 - x_1}\, y(x_2) = \left(1 + \tfrac{\sqrt{30}}{18}\right) y(x_1) + \left(1 - \tfrac{\sqrt{30}}{18}\right) y(x_2)$$

This rule can be verified to be correct for $y(x) = 1$, x, x^2 and x^3:

$$\begin{aligned}
\left(1 + \tfrac{\sqrt{30}}{18}\right) + \left(1 - \tfrac{\sqrt{30}}{18}\right) &= 2 \\
\left(1 + \tfrac{\sqrt{30}}{18}\right) x_1 + \left(1 - \tfrac{\sqrt{30}}{18}\right) x_2 &= \frac{2}{3} \\
\left(1 + \tfrac{\sqrt{30}}{18}\right) x_1^2 + \left(1 - \tfrac{\sqrt{30}}{18}\right) x_2^2 &= \frac{2}{5} \\
\left(1 + \tfrac{\sqrt{30}}{18}\right) x_1^3 + \left(1 - \tfrac{\sqrt{30}}{18}\right) x_2^3 &= \frac{2}{7}
\end{aligned}$$

but

$$\left(1 + \tfrac{\sqrt{30}}{18}\right) x_1^4 + \left(1 - \tfrac{\sqrt{30}}{18}\right) x_2^4 = \frac{258}{1225}$$

no longer equal the correct answer of $\frac{2}{9}$ (off by about 5%).

To approximate

$$\int_0^1 \frac{\exp(x^2)}{\sqrt{x}}\,dx \simeq \left(1 + \tfrac{\sqrt{30}}{18}\right) \exp(x_1^2) + \left(1 - \tfrac{\sqrt{30}}{18}\right) \exp(x_2^2) = 2.528$$

which is reasonably close (0.6% error) to the exact answer of 2.543.

We now modify the formula to approximate

$$\int_0^A \frac{y(x)}{\sqrt{x}}\,dx = A\int_0^1 \frac{y(A\,z)}{\sqrt{A\,z}}\,dz \simeq \sqrt{A}\left[\left(1+\tfrac{\sqrt{30}}{18}\right)\,y(A\,x_1)+\left(1-\tfrac{\sqrt{30}}{18}\right)y(A\,x_2)\right]$$

This, applied to

$$\int_0^{1/2} \frac{\exp(x^2)}{\sqrt{x}}\,dx$$

yields

$$\frac{1}{\sqrt{2}}\left[\left(1+\tfrac{\sqrt{30}}{18}\right)\exp(\tfrac{x_1^2}{4})+\left(1-\tfrac{\sqrt{30}}{18}\right)\exp(\tfrac{x_2^2}{4})\right] = 1.4898$$

a fairly decent (0.02% error) approximation to the exact answer of 1.4901.

Chapter 8 NUMERICAL DIFFERENTIATION

In this chapter we learn how to design formulas to numerically approximate the value of the first (second, third) derivative of a specific function at a given point. There are some similarities with numerical integration, but also some major differences, namely:

1. The motivation: Unlike integration, differentiation is an easy, routine procedure when done analytically, why bother with numerical estimation? True enough, the main application of numerical differentiation is not to compute derivatives, but to solve differential equations, both ordinary and partial.

2. Now, we will not fuss much about selecting nodes; we usual use simple, equidistant spacing (no 'Gaussian' differentiating).

3. In addition to reducing the so called TRUNCATION ERROR any such formula (the one expanded in powers of h), we will also have to fight the ROUND-OFF ERROR which, for numerical differentiation (unlike integration), becomes a major issue (that is why they sometimes say that numerically, integration is easy and differentiation hard - the exact opposite of the analytic situation).

The main idea remains the same though, to approximate say $y'(x_0)$ we select a few nodes (at and near x_0), evaluate $y(x)$ at these nodes, fit the corresponding interpolating polynomial, and differentiate it, instead of the original function.

Example: We choose, as our nodes, x_0 itself and then $x_0 - h$ and $x_0 + h$, where h is 'small'. The corresponding interpolating polynomial is

$$\frac{(x-x_0)(x-x_0-h)}{(-h)\cdot(-2h)}\,y(x_0-h)+\frac{(x-x_0+h)(x-x_0-h)}{h\cdot(-h)}\,y(x_0)+ \quad (8.1)$$
$$\frac{(x-x_0+h)(x-x_0)}{2h\cdot h}\,y(x_0+h)$$

Differentiating with respect to x, and then substituting $x = x_0$ yields

$$y'(x_0)\simeq\frac{y(x_0-h)}{-2h}+\frac{y(x_0+h)}{2h}=\frac{y(x_0+h)-y(x_0-h)}{2h}$$

which is clearly the slope computed based on the two end points. Using (7.1), the right hand side reduces to

$$y'(x_0)+\frac{y'''(x_0)}{6}h^2+\frac{y^v(x_0)}{120}h^4+....$$

The truncation error of the formula is thus

$$\frac{y'''(x_0)}{6}h^2+\frac{y^v(x_0)}{120}h^4+....$$

This formula may give us the impression that the error can be reduced, fast and easily, by making h very small. Unfortunately, when h is very small, the round-off error of the expression becomes huge, wiping out any accuracy gain achieved by reducing the truncation error.

Example: Using the formula of the last example, we approximate $y'(x_0)$ where $y(x) = \exp(x^2)$ and $x_0 = 1$. The exact value is easily computed to be $2e = 5.43656\,3656$.

h	$\frac{\exp[(1+h)^2]-\exp[(1-h)^2]}{2h}$	Error:
0.1	5.52788 333	0.0913197
0.01	5.43746 980	0.0009061
0.001	5.43657 250	0.0000088
0.0001	5.43656000	0.0000037
0.00001	5.43655000	0.0000136
0.000001	5.43650000	0.0000636

According our formula, the error should be reduced by a factor of 100 in each step. This is clearly so up to $h = 0.001$, but the next improvement is only by a factor of 2.4, and after that the results actually deteriorate! This means that the formula should never be used with h much smaller than 0.001 (assuming a 10 digit computational accuracy).

Richardson Extrapolation

We can reduce the truncation error of the formula by a technique practically identical to the old Romberg algorithm which, in this context, is called RICHARDSON EXTRAPOLATION. We just have to be careful not to reduce h beyond the value of 0.001 .

Example: Re-doing the previous example (this time, we will keep on reducing h by a factor of 2):

h	$R_i = \frac{\exp[(1+h)^2]-\exp[(1-h)^2]}{2h}$	$S_i = \frac{4R_{i+1}-R_i}{3}$	$T_i = \frac{16S_{i+1}-S_i}{15}$	$\frac{64T_{i+1}-T_i}{63}$
$\frac{1}{4}$	$6.03135\,7050$	$\mathbf{5}.\mathbf{4}2938\,7349$	$\mathbf{5}.\mathbf{43657}\,7469$	$\mathbf{5}.\mathbf{43656\,36}69$
$\frac{1}{8}$	$\mathbf{5}.57987\,9776$	$\mathbf{5}.\mathbf{436}12\,8086$	$\mathbf{5}.\mathbf{43656\,3}886$	$\mathbf{5}.\mathbf{43656\,3}704$
$\frac{1}{16}$	$\mathbf{5}.\mathbf{4}7206\,6010$	$\mathbf{5}.\mathbf{4365}3\,6649$	$\mathbf{5}.\mathbf{43656\,3}708$	
$\frac{1}{32}$	$\mathbf{5}.\mathbf{4}4541\,8990$	$\mathbf{5}.\mathbf{43656}\,2016$		
$\frac{1}{64}$	$\mathbf{5}.\mathbf{43}877\,6260$			

at which point we have clearly reached the limit of how much accuracy we can squeeze out of this table (the correct digits are in bold). Note that now it is short of impossible to get the full 10 digit accuracy (we will be lucky to get 8 correct digits), due to the round off error (always looming in the background).

Higher-Degree Formulas

To obtain a more accurate and faster converging formula, one has to increase the number of nodes. The most logical next step is to use $x = x_0$, $x_0 \pm h$ *and* $x_0 \pm 2h$. We may have noticed already (in our previous example) that the resulting coefficients are ANTISYMMETRIC (i.e. symmetric nodes have coefficients of *opposite* sign), which effectively eliminated the $y(x_0)$ value (it had 0 coefficient). One can show that this is the case in general, when approximating a derivative of an *odd* order (*even*-order derivatives will still require *symmetric* formulas).

There are two ways of further simplifying the formula derivation:

1. We can derive the formula assuming that $x_0 = 0$ and $h = 1$, and then transform it to allow arbitrary values (similar to introducing X when fitting functions - this time $X \equiv \frac{x-x_0}{h}$). In this new scale, the interpolating polynomial (using only $x_0 \pm h$ and $x_0 \pm 2h$ as nodes, not expecting x_0 to contribute) becomes

$$\frac{(X+1)(X-1)(X-2)}{(-1)\times(-3)\times(-4)}Y(-2) + \frac{(X+2)(X-1)(X-2)}{1\times(-2)\times(-3)}Y(-1) +$$
$$\frac{(X+2)(X+1)(X-2)}{3\times 2\times(-1)}Y(1) + \frac{(X+2)(X+1)(X-1)}{4\times 3\times 1}Y(2)$$

   Differentiating this expression with respect to X, then setting $X = 0$ yields

$$\frac{(-1)\times(-2)+1\times(-2)+1\times(-1)}{-12}Y(-2) + \frac{(-1)\times(-2)+2\times(-2)+2\times(-1)}{6}Y(-1) +$$
$$\frac{1\times(-2)+2\times(-2)+2\times 1}{-6}Y(1) + \frac{1\times(-1)+2\times(-1)+2\times 1}{12}Y(2) =$$
$$\frac{1}{12}Y(-2) - \frac{2}{3}Y(-1) + \frac{2}{3}Y(1) - \frac{1}{12}Y(2)$$

   Since $\frac{dy(x)}{dx} = \frac{dY(X)}{dX} \cdot \frac{dX}{dx} = \frac{dY(X)}{dX} \cdot \frac{1}{h}$, our final approximate formula reads:

$$y'(x_0) \simeq \frac{y(x_0 - 2h) - 8\,y(x_0 - h) + 8\,y(x_0 + h) - y(x_0 + 2h)}{12} \tag{8.2}$$

2. Faster yet, knowing that the formula (due to antisymmetry) must have the form of

$$-c_2\,Y(-2) - c_1 Y(-1) + c_1 Y(1) + c_2 Y(2)$$

   and be exact when $Y(X) = 1$, X, X^2, X^3 and X^4 (since it is, effectively, a *five* node formula) we make it correct with $Y(X) = X$ and X^3 (it already is correct with 1, X^2 and X^4, why?):

$$\begin{aligned} 2c_2 + c_1 + c_1 + 2c_2 &= 1 \\ 8c_2 + c_1 + 8c_1 + c_2 &= 0 \end{aligned}$$

   implying that $c_1 = -8c_2$ and $c_2 = -\frac{1}{12}$. We thus get, much more quickly, the same answer as before.

With the help of (7.1), we get

$$y(x_0 - 2h) = \tag{8.3}$$
$$y(x_0) - 2h\,y'(x_0) + \frac{4h^2}{2}y''(x_0) - \frac{8h^3}{6}y'''(x_0) + \frac{16h^4}{24}y^{iv}(x_0) - \frac{32h^5}{120}y^{v}(x_0) + ...$$
$$y(x_0 - h) = \tag{8.4}$$
$$y(x_0) - h\,y'(x_0) + \frac{h^2}{2}y''(x_0) - \frac{h^3}{6}y'''(x_0) + \frac{h^4}{24}y^{iv}(x_0) - \frac{h^5}{120}y^{v}(x_0) + ...$$
$$y(x_0 + h) = \tag{8.5}$$
$$y(x_0) + h\,y'(x_0) + \frac{h^2}{2}y''(x_0) + \frac{h^3}{6}y'''(x_0) + \frac{h^4}{24}y^{iv}(x_0) + \frac{h^5}{120}y^{v}(x_0) + ...$$
$$y(x_0 + 2h) = \tag{8.6}$$
$$y(x_0) + 2h\,y'(x_0) + \frac{4h^2}{2}y''(x_0) + \frac{8h^3}{6}y'''(x_0) + \frac{16h^4}{24}y^{iv}(x_0) + \frac{32h^5}{120}y^{v}(x_0) + ...$$

The right hand side of (8.2) is thus equal to

$$y'(x_0) - \frac{h^4}{30}y^{v}(x_0) + ...$$

where the second term represents the error of the formula.

Example: Estimating $\frac{d\exp(x^2)}{dx}$ at $x = 1$ yet one more time (using our last formula), we get:

h	$R_i = \frac{-\exp[(1+2h)^2]+8\exp[(1+h)^2]-8\exp[(1-h)^2]+\exp[(1-2h)^2]}{12h}$	$T_i = \frac{16R_{i+1}-R_i}{15}$	$\frac{64T_{i+1}-T_i}{63}$
$\frac{1}{4}$	**5.** 30723 9257	**5. 43**753 0565	**5. 43656** 2332
$\frac{1}{8}$	**5. 4**2938 7358	**5. 43657** 7463	**5. 43656 36**67
$\frac{1}{16}$	**5. 436**12 8081	**5. 43656 3**885	
$\frac{1}{32}$	**5. 4365**3 6647		

with a bit of Richardson extrapolation.

Nonsymmetric spacing

Sometimes we are restricted in our choice of nodes. For example, we may be required to evaluate $y(x)$ only at and to the right of x_0.

Example: We would like to approximate $y'(x_0)$ by a formula with three nodes, x_0, $x_0 + h$ and $x_0 + 2h$. We know that, in terms of X, the formula will have the form of

$$c_0 Y(0) + c_1 Y(1) + c_2 Y(2)$$

(the coefficients will no longer have any symmetry). Making it exact for $Y(X) = 1$, X and X^2 yields:

$$\begin{aligned} c_0 + c_1 + c_2 &= 0 \\ c_1 + 2c_2 &= 1 \\ c_1 + 4c_2 &= 0 \end{aligned}$$

which yields $c_2 = -\frac{1}{2}$, $c_1 = 2$, $c_0 = -\frac{3}{2}$. The transformed formula is thus:

$$y'(x_0) \simeq \frac{-3y(x_0) + 4y(x_0 + h) - y(x_0 + 2h)}{2h}$$

Its right hand side, with the help of (8.5) and (8.6) evaluates to

$$y'(x_0) - \frac{h^2}{3} y'''(x_0) - \frac{h^3}{4} y^{iv}(x_0) - ...$$

Note that the main error term is now proportional to h^2 (since we have three nodes), after which both odd and even powers of h contribute.

Applied to our benchmark problem, this yields:

h	$R_i = \frac{-3\exp(1)+4\exp[(1+h)^2]-\exp[(1+2h)^2]}{2h}$	$S_i = \frac{4R_{i+1}-R_i}{3}$	$\frac{8S_{i+1}-S_i}{7}$
$\frac{1}{16}$	**5.** 35149 250	**5. 43**909 0859	**5. 4365**3 1527
$\frac{1}{32}$	**5. 4**1719 127	**5. 436**85 1443	
$\frac{1}{64}$	**5. 43**193 640		

Note the formula's poor performance (due to being nonsymmetric). Due to all orders of h now contributing to the error term, Richardson extrapolation now follows a different pattern (for the same reason, the resulting improvement is not as impressive as before).

Higher derivatives

To approximate $y''(x_0)$, we first derive a basic three-node (symmetric) formula. The interpolating polynomial is of course the same (8.1), differentiating it *twice* yields the following result:

$$\frac{y(x_0 - h) - 2y(x_0) + y(x_0 + h)}{h^2}$$

Utilizing (8.5) and (8.4), this equals to

$$y''(x_0) + \frac{y^{iv}(x_0)}{12} h^2 + ...$$

where the second term represents the truncation error. The round-off error is now substantially bigger than in the $y'(x_0)$ case (a reflection of having h^2 in the denominator, instead of h).

Example: We will use the previous formula to approximate the second derivative of $\exp(x^2)$ at $x = 1$ (analytically, the second derivative equals to $(4x^2 + 2)\exp(x^2)$, and evaluates, at $x = 1$, to $6e = 16.30969\,097$):

h	$R_i = \frac{\exp[(1-h)^2]-2\exp(1)+\exp[(1+h)^2]}{h^2}$	$S_i = \frac{4R_{i+1}-R_i}{3}$	$\frac{16S_{i+1}-S_i}{15}$
$\frac{1}{16}$	**16. 3**77100	**16. 309**653	**16. 3097**12
$\frac{1}{32}$	**16. 3**26515	**16. 3097**08	
$\frac{1}{64}$	**16. 3**13910		

Clearly, the second stage of Richardson extrapolation can no longer improve the accuracy (due to the extra random error). Things would look different if the computation is carried out using a 15-digit accuracy:

h	$R_i = \frac{\exp[(1-h)^2]-2\exp(1)+\exp[(1+h)^2]}{h^2}$	$S_i = \frac{4R_{i+1}-R_i}{3}$	$\frac{16S_{i+1}-S_i}{15}$
$\frac{1}{16}$	**16.3**7709 985	**16.3096**5 102	**16.30969 09**8
$\frac{1}{32}$	**16.3**2651 323	**16.30968** 848	
$\frac{1}{64}$	**16.3**1389 467		

Finally, we derive a formula to approximate $y'''(x_0)$. Here, the minimum of four nodes is needed (always the order of the derivative plus one), we will choose them at $x_0 \pm h$ and $x_0 \pm 2h$. This time, we use the X trick; in the 'capital' scale, the formula must read (utilizing its antisymmetry):

$$-c_2\,Y(-2) - c_1\,Y(-1) + c_1\,Y(1) + c_2\,Y(2)$$

Furthermore, it has to yield the exact answer with $Y(X) = X$ and X^3:

$$\begin{aligned} 2c_2 + c_1 + c_1 + 2c_2 &= 0 \\ 8c_2 + c_1 + c_1 + 8c_2 &= 6 \end{aligned}$$

which implies $c_1 = -2c_2$ and $c_2 = \frac{1}{2}$. We thus obtain

$$Y'''(0) \simeq \frac{Y(2) - 2Y(1) + 2Y(-1) - Y(-2)}{2}$$

or, going back to $y(x)$:

$$y'''(x_0) \simeq \frac{y(x_0+2h) - 2y(x_0+h) + 2y(x_0-h) - y(x_0-2h)}{2h^3}$$

With the help of (8.3) to (8.6), the right hand side reduces to

$$y'''(x_0) + \frac{y^v(x_0)}{4}\,h^2 + ...$$

Example: We use this formula to approximate the third derivative of $\exp(x^2)$ at $x = 1$ (the exact answer is $20e = 54.36563\,657$). Based on our previous experience, we carry out the computation using 20-digit accuracy (to eliminate the round-off error, which would otherwise 'steal' from us another two digits).

h	$R_i = \frac{\exp[(1+2h)^2]-2\exp[(1+h)^2]+2\exp[(1-h)^2]-\exp[(1-2h)^2]}{2h^3}$	$S_i = \frac{4R_{i+1}-R_i}{3}$	$\frac{16S_{i+1}-S_i}{15}$
$\frac{1}{32}$	**54.** 57311 583	**54.365**53 087	**54.36563 65**9
$\frac{1}{64}$	**54.** 41742 711	**54.36562** 998	
$\frac{1}{128}$	**54.37**857 926		

As mentioned earlier, the main application of these formulas is to solve, numerically, several types of ordinary and partial differential equations. Before we can do that, we have to first study yet another important issue, that of solving

Chapter 9 NONLINEAR EQUATIONS

We will start with the case of one equation with one unknown, which can be always written as

$$f(x) = 0$$

Finding all solutions graphically is easy: we plot $f(x)$ against x (using a wide enough range of x values, to make sure we have not missed anything); then each time the graph crosses (or touches) the x axis, the corresponding x coordinate provides a solution to the equation. The only difficulty with this approach is the accuracy - we will be lucky to get the first two digits of the correct answer. But, this will be enough to give a good estimate of the solution, which can ten be refined (to any accuracy) by the so called

Newton's Method

Expanding the graph of $f(x)$ around the point of intersect and observing only a small segment of it, the result will look almost like a straight line (with only slight curvature). We will also assume that the graph crosses the x axis at a non-zero angle (rather then just touching it). It is then obvious (draw the corresponding picture) that our initial estimate of the root (say x_0) can be improved by fitting a straight line with a slope of $f'(x_0)$ trough the point $[x_0,\ f(x_0)]$, and finding *its* intercept with the x axis. Since

$$y - f(x_0) = f'(x_0) \cdot (x - x_0)$$

is the equation of this straight line, solving

$$-f(x_0) = f'(x_0) \cdot (x - x_0)$$

for x yields the desired improved solution, namely:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

This yields a better, but not necessarily 10-digit accurate solution. But clearly, we can now apply the same idea again, using x_1 in place of x_0. This will result in

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}$$

And again:

$$x_3 = x_2 - \frac{f(x_2)}{f'(x_2)}$$

etc., until the answers no longer change (within our usual, 10-digit accuracy).

One can show that this procedure is QUADRATICALLY CONVERGENT, which means that the number of correct digits roughly doubles in each ITERATION (one step of the procedure).

Example: To solve

$$e^x = 2 - x$$

we plot the two functions (e^x and $2-x$) and note that they intersect in a point whose x coordinate is roughly equal to 1 (this is an alternate way of doing it - one does not even need Maple). Our x_0 is thus set to 1, $f(x) = e^x - 2 + x$ and $f'(x) = e^x + 1$. And we are ready to iterate:

$$\begin{aligned}
x_1 &= 1 - \frac{e-2+1}{e+1} = 0.53788\,28428 \\
x_2 &= x_1 - \frac{e^{x_1}-2+x_1}{e^{x_1}+1} = 0.\mathbf{44}561\,67486 \\
x_3 &= x_2 - \frac{e^{x_2}-2+x_2}{e^{x_2}+1} = 0.\mathbf{44285}\,67246 \\
x_4 &= x_3 - \frac{e^{x_3}-2+x_3}{e^{x_3}+1} = 0.\mathbf{44285\,44011} \\
x_5 &= x_4 - \frac{e^{x_4}-2+x_4}{e^{x_4}+1} = 0.\mathbf{44285\,44011}
\end{aligned}$$

at which point the value no longer changes. Quadratic convergence is be clearly observed.

The method fails (convergence becomes extremely slow) when $f(x)$ only touches the x axis, without crossing it (this is an indication that both $f(x)$ *and* $f'(x)$ have a root at that point), or crosses it at 0 angle (an indication that $f''(x)$ is also equal to zero at that point). The best way to overcome this problem is: As soon as we notice it (from graph, or slow convergence of the standard technique), we switch to solving $f'(x) = 0$ instead. If that is still giving trouble, solve $f''(x) = 0$, etc. In the end (to be on the save side), we should verify that the final answer does meet $f(x) = 0$.

Example: We know that $f(x) = 1 + \sin x$ has a root at $x = \frac{3}{2}\pi = 4.71238\,8981$ ($\equiv 270^o$). If we try to find it by the regular technique (starting at $x_0 = 5$), we get

$$\begin{aligned}
x_1 &= x_0 - \frac{1+\sin x_0}{\cos x_0} = \mathbf{4}.85519\,4921 \\
x_2 &= x_1 - \frac{1+\sin x_1}{\cos x_1} = \mathbf{4}.\mathbf{7}8367\,0356 \\
x_3 &= x_2 - \frac{1+\sin x_2}{\cos x_2} = \mathbf{4}.\mathbf{7}4801\,457 \\
x_4 &= x_3 - \frac{1+\sin x_3}{\cos x_3} = \mathbf{4}.\mathbf{7}3019\,9891 \\
x_5 &= x_4 - \frac{1+\sin x_4}{\cos x_4} = \mathbf{4}.\mathbf{7}2129\,4199 \\
x_6 &= x_5 - \frac{1+\sin x_5}{\cos x_5} = \mathbf{4}.\mathbf{71}684\,156 \\
x_7 &= x_6 - \frac{1+\sin x_6}{\cos x_6} = \mathbf{4}.\mathbf{71}461\,527 \\
&\vdots
\end{aligned}$$

Even though the procedure seems to converge to the correct answer, the process would obviously take forever.

If instead we solve $(1+\sin x)' = \cos x = 0$, we get

$$\begin{aligned} x_1 &= x_0 + \frac{\cos x_0}{\sin x_0} = \mathbf{4.7}0418\,7084 \\ x_2 &= x_1 + \frac{\cos x_1}{\sin x_1} = \mathbf{4.71238}\,9164 \\ x_3 &= x_2 + \frac{\cos x_2}{\sin x_2} = \mathbf{4.71238\,898} \\ x_4 &= x_3 + \frac{\cos x_3}{\sin x_3} = \mathbf{4.71238\,898} \end{aligned}$$

we get the exact answer in 3 iterations (even though an extra iteration is needed to confirm that). One can now easily verify that

$$1 + \sin 4.71238\,898 = 0$$

We now have a technique for finding roots of our orthogonal polynomials which, luckily, all must have SIMPLE *real* roots spreading over the original (A, B) interval (otherwise, polynomial roots require special consideration - we will not go into that).

Example: When dealing with the third-degree Laguerre polynomial, we had to rely on Maple to get its three roots. Now, we can do this on our own. Plotting

$$x^3 - 9x^2 + 18x - 6 = 0$$

indicates that there is a root near $x_0 = 6$. We thus get

$$\begin{aligned} x_1 &= x_0 - \frac{x_0^3 - 9x_0^2 + 18x_0 - 6}{3x_0^2 - 18x_0 + 18} = \mathbf{6}.33333\,3333 \\ x_2 &= x_1 - \frac{x_1^3 - 9x_1^2 + 18x_1 - 6}{3x_1^2 - 18x_1 + 18} = \mathbf{6.29}071\,5373 \\ x_3 &= x_2 - \frac{x_2^3 - 9x_2^2 + 18x_2 - 6}{3x_2^2 - 18x_2 + 18} = \mathbf{6.28994\,5}332 \\ x_4 &= x_3 - \frac{x_3^3 - 9x_3^2 + 18x_3 - 6}{3x_3^2 - 18x_3 + 18} = \mathbf{6.28994\,5083} \end{aligned}$$

Based on our experience with quadratic convergence, we don't have to verify that the last answer is correct.

Once we have a root of a cubic equation, we can DEFLATE the polynomial by carrying out the following SYNTHETIC DIVISION:

$$\begin{aligned} &(x^3 - 9x^2 + 18x - 6) \div (x - 6.28994\,5083) = \\ &x^2 - 2.710054917\,x + 0.9539034002 \end{aligned}$$

The remaining two roots can then be found easily to be:

$$\begin{aligned} \tfrac{2.710054917}{2} + \sqrt{(\tfrac{2.710054917}{2})^2 - 0.9539034002} &= 2.29428\,0362 \\ \tfrac{2.710054917}{2} - \sqrt{(\tfrac{2.710054917}{2})^2 - 0.9539034002} &= 0.41577\,45565 \end{aligned}$$

in agreement with our previous example.

We will now extend this technique to solve the case of

Several Unknowns

We will start by trying to solve two nonlinear equations with two unknowns, which we will call x_1 and x_2 (collectively $\mathbf{x}$, using a vector notation). The two equations can be always written in the following form:

$$\begin{aligned} F_1(x_1, x_2) &= 0 \\ F_2(x_1, x_2) &= 0 \end{aligned}$$

where each F is now a function of two variables (our unknowns). The issue of finding a reasonably accurate starting (INITIAL) solution is now a lot more difficult - plotting the two functions is still possible (even though now we need two 3D plots), but extracting the information we need is a lot more difficult (and even this approach will ultimately fail, when we have 3 or more unknowns). There are various techniques capable of a GLOBAL SEARCH (in n dimensional space, where n is the number of unknowns) for a good staring point; due to a lack of time, we have to bypass this step and simply assume that a reasonably good estimate is provided to us.

All we need to do then is to adapt the Newton's technique to two and more variables. Assuming that we have a pair of initial values x_{1_0} and x_{2_0}, each of the F functions can be (Taylor) expanded, around this point, as follows:

$$\begin{aligned} F_1(x_1, x_2) &= F_1(x_{1_0}, x_{2_0}) + \frac{\partial F_1(x_{1_0}, x_{2_0})}{\partial x_1}(x_1 - x_{1_0}) + \frac{\partial F_1(x_{1_0}, x_{2_0})}{\partial x_2}(x_2 - x_{2_0}) + ... \\ F_2(x_1, x_2) &= F_2(x_{1_0}, x_{2_0}) + \frac{\partial F_2(x_{1_0}, x_{2_0})}{\partial x_1}(x_1 - x_{1_0}) + \frac{\partial F_2(x_{1_0}, x_{2_0})}{\partial x_2}(x_2 - x_{2_0}) + ... \end{aligned}$$

or, using a matrix notation,

$$\mathbf{F}(\mathbf{x}) = \mathbf{F}(\mathbf{x}_{(0)}) + \left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right|_{\mathbf{x}_{(0)}} (\mathbf{x} - \mathbf{x}_{(0)}) + ... \tag{9.1}$$

where $\mathbf{x}_{(0)}$ is a column vector with components x_{1_0} and x_{2_0}, and $\frac{\partial \mathbf{F}}{\partial \mathbf{x}}$ denotes, rather symbolically, the following matrix of partial derivatives (called the JACOBIAN)

$$\frac{\partial \mathbf{F}}{\partial \mathbf{x}} \equiv \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} \end{bmatrix}$$

of each equation (one per row) differentiated with respect of each variable (one per column). Making the left hand side of (9.1) equal to a zero vector, and solving for $\mathbf{x}$ yields:

$$\mathbf{x}_{(1)} = \mathbf{x}_{(0)} - \left[\left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right|_{\mathbf{x}_{(0)}} \right]^{-1} \mathbf{F}(\mathbf{x}_{(0)})$$

which constitutes one iteration. In this spirit we can continue:

$$\mathbf{x}_{(2)} = \mathbf{x}_{(1)} - \left[\left.\frac{\partial \mathbf{F}}{\partial \mathbf{x}}\right|_{\mathbf{x}_{(1)}} \right]^{-1} \mathbf{F}(\mathbf{x}_{(1)})$$

etc., until convergence is reached.

Example: Solve

$$\begin{aligned} x_1 \cos x_2 + 0.716 &= 0 \\ x_2 \sin x_1 - x_1^2 - 1.305 &= 0 \end{aligned}$$

starting with $x_{1_0} = 1$ and $x_{2_0} = 3$. The Jacobian is clearly

$$\begin{bmatrix} \cos x_2 & -x_1 \sin x_2 \\ x_2 \cos x_1 - 2x_1 & \sin x_1 \end{bmatrix}$$

Evaluating the left hand side of each of our equations (using the initial values of x_1 and x_2) yields

$$\mathbf{F}_{(0)} = \begin{bmatrix} -0.27399\,24966 \\ 0.21941\,29540 \end{bmatrix}$$

similarly evaluating the Jacobian results in

$$\mathbb{J}_{(0)} = \begin{bmatrix} -0.98999\,24966 & -0.14112\,00081 \\ -0.37909\,3082 & 0.84147\,09848 \end{bmatrix}$$

We can now compute the values of x_{1_1} and x_{2_1} as the two components of

$$\mathbf{x}_{(1)} = \mathbf{x}_{(0)} - [\mathbb{J}_{(0)}]^{-1}\mathbf{F}_{(0)} = \begin{bmatrix} 0.77486\,46744 \\ 2.\,63782\,4472 \end{bmatrix}$$

where $\mathbf{x}_{(0)}$ is the initial vector (with components 1 and 3). This completes the first iteration. We can now repeat the process, until the two values no longer change.

Even though we can still manage to do this with a calculator, we have clearly reached the point at which it may be better to delegate the routine but tedious computation to Maple. This can be done as follows:

$F := [\ x[1]*\mathbf{cos}(x[2]) + 0.716,\ x[2]*\mathbf{sin}(x[1]) - x[1]\hat{}2 - 1.305\ ];$

$J := \mathbf{matrix}(2,2):$

for i **to** 2 **do** **for** j **to** 2 **do** $J[i,\ j] := \mathbf{diff}(F[i],\ x[j])$ **end do** **end do:**

$x := [1.,\ 3.]:$

with(linalg):

$x := \mathbf{evalm}(x - \ \mathbf{linsolve}(J,\ F)\ );$

The last line computes the $\mathbf{x}_{(1)}$ vector (instead of $\mathbb{J}^{-1}\mathbf{F}$, we solve the equivalent set of equations). We find it convenient not to introduce a new name, but call both $\mathbf{x}_{(0)}$ and $\mathbf{x}_{(1)}$ simply x. This has the advantage that, by re-executing the last line, we will automatically get $\mathbf{x}_{(2)}$ etc. In our case, executing the last line five times yields:

[0.**7**748646744, **2**.637824472]

[0.**78**25429930, **2**, **71**9825617]

[0.**7854**682773, **2.7178**42406]

[0.**785460622**0, **2.717875728**]

[0.**7854606228**, **2.717875728**]

at which point the procedure has clearly converged (note that, due to the round off error, the last digit of either component may keep on changing, sometimes indefinitely).

It should be clear how the formulas extend to the case of three or more unknowns. Let us do an example which was in the last year's exam.

Example: We need to solve the following three equations (the algebraic, geometric and harmonic mean of three numbers is given, respectively, as):

$$\begin{aligned} \frac{x_1+x_2+x_3}{3} &= 7 \\ \sqrt[3]{x_1x_2x_3} &= 4 \\ \frac{3}{\frac{1}{x_1}+\frac{1}{x_2}+\frac{1}{x_3}} &= \frac{16}{7} \end{aligned}$$

For the initial values we will take $x_1 = 1.5$, $x_2 = 5$ and $x_3 = 10$. All we need to modify in our previous computer 'program' is the definition of F, thus (it does not hurt to simplify the equations):

$F :=$ [$x[1]+x[2]+x[3]-21$, $x[1]*x[2]*x[3]-64$, $1/x[1]+1/x[2]+1/x[3]-21/16$];

the dimensions (i.e. 3 instead of 2 in the next two lines), and the initial values. We will also need to rewrite the last line (to fix one of Maple's many quirks) as follows:

$x :=$ **evalm**($x -$ **inverse**(J) &* F);

As result, we get:

[.6989495801, 3.572619045, 16.72843138]

[.8746180084, 4.895879953, 15.22950204]

[.9791494539, 3.948814285, 16.07203626]

[.9992737556, 4.004281052, 15.99644519]

[.9999992176, 3.999999364, 16.00000142]

[.9999999991, 4.000000005, 16.00000000]

The exact answer thus seems to be $x_1 = 1$, $x_2 = 4$ and $x_3 = 16$ (this can be easily verified against the original equations).

If we used the equations in their original form, it would have taken us 9 iterations to reach the same conclusion.

The choice of the initial values is quite critical, see what would happen if we change x_{1_0} to 2.0 .

Chapter 10 **ODE**, BOUNDARY-VALUE PROBLEM

We will consider only the case of 2^{nd} order differential equations, which can usually be written as

$$y'' = f(x, y, y') \tag{10.1}$$

As we all know, there is normally an infinite set of solutions (i.e. $y(x)$ functions which meet the equation) - to narrow it down to a single, unique solution, we have to impose two extra conditions. These are of two basic types:

1. Initial values, where both $y(x)$ and $y'(x)$ are specified at the same, 'initial' value of x_0 (often equal to 0). To solve this kind of problem requires techniques which we don't have time to discuss in this course.

2. Boundary values of $y(x)$ are given at $x = A$ and $x = B$. This is the situation we will study in detail now.

We will make our task easier by first considering a special case of (10.1), namely that of a

Linear Differential Equation

which can be written as follows

$$y''(x) + p(x)\,y'(x) + q(x)\,y(x) = r(x) \tag{10.2}$$

where $p(x)$, $q(x)$ and $r(x)$ are specific (given) functions of x. The equation is linear in $y(x)$ and its derivatives (but not in x). It is now more convenient to combine all y related terms on the left hand side of the equation. The two boundary conditions will be written as $y(A) = \alpha$ and $y(B) = \beta$.

The equation is quite often impossible to solve analytically, so we need a numerical technique for doing the job. The idea is quite simple: we subdivide the (A, B) interval into n equal-length subintervals (the nodes will be denoted x_0, x_1, x_2, ... x_{n-1}, x_n, where $x_0 = A$ and $x_n = B$), and try to solve for the corresponding y_0, y_1, y_2, ... y_{n-1} and y_n (the first and last of these are simply equal to A and B respectively, but how about the rest?). This is done by replacing y'' and y' in (10.2) by a suitable numerical approximation (we know how to compute these), at each of the inside nodes, thus getting $n-1$ ordinary, linear equations for y_1, y_2, ... y_{n-1}. Solving these will give us a good approximation of the desired solution (we of course get only a set of x-y points, instead of a function, but we know how to extend this to a smooth curve). This is sometimes referred to as the finite-difference technique.

The simplest (and reasonably adequate) way of approximating y'' at x_i is to take

$$\frac{y_{i-1} - 2y_i + y_{i+1}}{h^2}$$

Similarly, we can approximate y' at x_i by

$$\frac{y_{i+1} - y_{i-1}}{2h}$$

where $h = \frac{B-A}{n}$. Note that both formulas have error terms proportional to h^2 (in terms of the leading term). The same can then be expected of our final solution.

When substituting these into (10.2), point by point, we get

$$\begin{aligned}
\frac{y_0 - 2y_1 + y_2}{h^2} + \frac{y_2 - y_0}{2h} p_1 + y_1 q_1 &= r_1 \\
\frac{y_1 - 2y_2 + y_3}{h^2} + \frac{y_3 - y_2}{2h} p_2 + y_2 q_2 &= r_2 \\
&\vdots \\
\frac{y_{n-2} - 2y_{n-1} + y_n}{h^2} + \frac{y_n - y_{n-2}}{2h} p_{n-1} + y_{n-1} q_{n-1} &= r_{n-1}
\end{aligned}$$

where $p_1 \equiv p(x_1)$, $p_2 \equiv p(x_2)$, ... (similarly for q_1, q_2, ... and r_1, r_2, ...).

This can be expressed in the following matrix form:

$$\begin{bmatrix}
-2 + h^2 q_1 & 1 + \frac{hp_1}{2} & 0 & \cdots & 0 \\
1 - \frac{hp_2}{2} & -2 + h^2 q_2 & 1 + \frac{hp_2}{2} & \cdots & \vdots \\
0 & 1 - \frac{hp_3}{2} & -2 + h^2 q_3 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & 1 - \frac{hp_{n-2}}{2} \\
0 & \cdots & 0 & 1 - \frac{hp_{n-1}}{2} & -2 + h^2 q_{n-1}
\end{bmatrix}
\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_{n-1} \end{bmatrix} =
\begin{bmatrix}
r_1 h^2 - y_0 \left(1 - \frac{hp_1}{2}\right) \\
r_2 h^2 \\
r_3 h^2 \\
\vdots \\
r_{n-1} h^2 - y_n \left(1 + \frac{hp_{n-1}}{2}\right)
\end{bmatrix}$$

The fact that the resulting matrix of coefficients is tri-diagonal greatly simplifies the task of solving the equations.

Example: Using the technique, we will solve the following differential equation

$$y'' = \frac{3}{1+x} y' + \frac{4}{(1+x)^2} y = 1 + x$$

with boundary conditions: $y(1) = 0$ and $y(2) = 9$. Coincidentally (the equation is of a very special form), this problem has the following analytic solution:

$$y(x) = x^3 + x^2 - x - 1$$

which will enable us to compare our results with the exact answer.

First we choose $n = 4$ ($h = \frac{1}{4}$), and fill out the following table:

x_i	$\frac{hp_i}{2}$	$h^2 q_i$	$h^2 r_i$
1.25	$-\frac{3}{18}$	$\frac{4}{81}$	$\frac{9}{64}$
1.50	$-\frac{3}{20}$	$\frac{4}{100}$	$\frac{10}{64}$
1.75	$-\frac{3}{22}$	$\frac{4}{121}$	$\frac{11}{64}$

This means, to get the corresponding set of y_i values, we have to solve the following linear (tri-diagonal) set of equations:

$-\frac{158}{81}$	$\frac{15}{18}$	0	$\frac{9}{64} - 0 \times \frac{21}{18}$
$\frac{23}{20}$	$-\frac{196}{100}$	$\frac{17}{20}$	$\frac{10}{64}$
0	$\frac{25}{22}$	$-\frac{238}{121}$	$\frac{11}{64} - 9 \times \frac{19}{22}$

Bypassing the details (we have nothing new to learn here), the answers are:

$$\begin{aligned} y_1 &\equiv y(1.25) = \frac{1935711}{1537792} = 1.25876 \ (1.265625) \\ y_2 &\equiv y(1.50) = \frac{1197625}{384448} = 3.11518 \ (3.125) \\ y_3 &\equiv y(1.75) = \frac{148071847}{26142464} = 5.66404 \ (5.671875) \end{aligned}$$

Comparing them with the exact values (in parentheses), we can see that their relative errors are about 0.5, 0.3 and 0.1% respectively.

We can improve the value of y_2 by Richardson extrapolation by redoing the problem using $n = 2$ ($h = \frac{1}{2}$). the new table of $\frac{hp_i}{2}$, $h^2 q_i$ and $h^2 r_i$ values now has only one row:

x_i	$\frac{hp_i}{2}$	$h^2 q_i$	$h^2 r_i$
1.50	$-\frac{3}{10}$	$\frac{4}{25}$	$\frac{10}{16}$

and the corresponding set of equations is reduced to only one:

$-\frac{46}{25}$	$\frac{10}{16} - 0 \times \frac{13}{10} - 9 \times \frac{7}{10}$

Note that now we are using a different system of indexing (the new y_1 is the old y_2, etc.); also note that the right hand side has been computed based on

$$r_1 h^2 - y_0(1 - \tfrac{hp_1}{2}) - y_2(1 + \tfrac{hp_1}{2})$$

(this obviously happens only with $n = 2$). The solution is $y_1 \equiv y(1.5) = 3.08424$, having a 1.3% error. Richardson extrapolation can now improve the value of $y(1.5)$ by computing

$$\frac{4 \times 3.11518 - 3.08424}{3} = 3.12549$$

having the error of only 0.015% (a twenty-fold improvement over the $n = 4$ answer). We get a substantially improved accuracy, but at *fewer* points.

In our next example, we like to achieve a better accuracy by simply increasing the value of n. That of course makes the computation rather tedious, so we abandon the 'by-hand' approach and write the corresponding Maple 'program' instead.

Example: We now solve

$$y'' - \exp(\tfrac{x}{2})\, y' + \ln(1+x)\, y = \sin x$$

with $y(0) = 2$ and $y(3) = 5$, by typing:

```
n := 6:  h := 3./n:
x := [seq(h*i, i = 1..n-1)]:
MC := matrix(n-1, n-1, 0):
for i to n-1 do MC[i,i] := -2 + h^2* ln(1 + x[i]) end do:
for i to n-2 do MC[i,i+1] := 1 - h/2* exp(x[i]/2);
    MC[i+1,i] := 1 + h/2* exp(x[i+1]/2) end do:
r :=  [seq(h^2* sin(x[i]), i = 1..n-1)]:
r[1] := r[1] - 2*(1 + h/2* exp(x[1]/2)):
r[n-1] := r[n-1] - 5*(1 - h/2* exp(x[n-1]/2)):
with(linalg):
y := linsolve(MC, r);
        y := [2.284400610, 2.673196903, 3.177073356, 3.797670488, 4.504905834]
with(plots):
poinplot( [ [0,2], seq( [ [x[i], y[i] ], i = 1..n-1), [3,5] ] );
```

the last line plotting the results. We can now re-run the program with $n = 12$, getting a better approximation to the solution. We can then also substantially improve the values of $y(0.5)$, $y(1.0)$, $y(1.5)$, ... $y(2.5)$ by Richardson extrapolation.

Nonlinear Case

We will now go back to the general (nonlinear) case of

$$y'' - f(x, y, y') = 0$$

where $f(x, y, y')$ can be any expression involving x, y and y'. We can discretize this equation (subject to some boundary conditions) in the same manner as before, this time getting a nonlinear set of *ordinary* equations for y_1, y_2, ... y_{n-1}.

Example: To solve

$$y'' + \frac{y' \cdot y}{1 + x^2} - x = 0$$

subject to $y(0) = 2$ and $y(3) = -1$, we use $n = 3$ ($h = 1$). At $x_1 = 1$ and $x_2 = 2$, our approximation yields:

$$\begin{aligned} y_0 - 2y_1 + y_2 + \frac{y_2 - y_0}{2} \cdot \frac{y_1}{2} - 1 &= 0 \\ y_1 - 2y_2 + y_3 + \frac{y_3 - y_1}{2} \cdot \frac{y_2}{5} - 2 &= 0 \end{aligned}$$

where $y_0 = 1$ and $y_3 = 0$. These are then two nonlinear equations for y_1 and y_2. They will be solved by finding the Jacobian

$$\begin{bmatrix} -2 + \frac{y_2 - 1}{4} & 1 + \frac{y_1}{4} \\ 1 - \frac{y_2}{10} & -2 - \frac{y_1}{10} \end{bmatrix}$$

and a sensible set of initial values, which in these circumstances can be obtained from a simple straight line connecting the boundary values ($y_1 = \frac{2}{3}$ and $y_2 = \frac{1}{3}$ in our case). And we are ready to iterate, i.e. replace y_1 and y_2 by the two components of

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} - \begin{bmatrix} -2 + \frac{y_2 - 1}{4} & 1 + \frac{y_1}{4} \\ 1 - \frac{y_2}{10} & -2 - \frac{y_1}{10} \end{bmatrix}^{-1} \begin{bmatrix} -2y_1 + y_2 + \frac{y_2 - 1}{2} \cdot \frac{y_1}{2} \\ y_1 - 2y_2 - \frac{y_1}{2} \cdot \frac{y_2}{5} - 2 \end{bmatrix}$$

evaluated using the current (starting with the initial) values of y_1 and y_2. The iterations yield:

y_1:	y_2:
-0.7230514093	$-\mathbf{1.2}95190714$
$-0.\mathbf{498}7596596$	$-\mathbf{1.2811}72478$
$-0.\mathbf{4984471907}$	$-\mathbf{1.281152950}$
$-0.\mathbf{49844719}01$	$-\mathbf{1.281152950}$

The Newton's technique thus converged in three iterations. This of course does not mean that we have a 10-digit accurate solution to the original differential equation!

To build a more accurate solution, we use $n = 6$ ($h = \frac{1}{2}$). This can be done, with the help of Maple, as follows:

```
n := 6:   h := 3/n:   y[0] := 2:   y[n] := -1:

x := [seq(h * i, i = 1..n - 1)]:

F := [seq( (y[i + 1] - 2 * y[i] + y[i - 1])/h^2+

      (y[i + 1] - y[i - 1])/2/h * y[i]/(1 + x[i]^2) - x[i],  i = 1..n - 1)]:

J := matrix(n - 1, n - 1):

for i to n - 1 do  for j to n - 1 do  J[i, j] :=diff(F[i], y[j]) end do  end do:

y := [seq(2. - 3 * h * i,  i = 1..n - 1)]:

with(linalg):
```

for i **to** 5 **do** y :=**evalm**(y−**inverse**(J) &∗ F) **end do**;

y := [0.5386935342, −0.5221789195, −**1**.395663039, −**1**.905281418, −**1.6**51949253]

y := [0.**62**83620082, −0.**32**27896441, −**1.08**7744374, −**1.579**900666, −**1.618**444037]

y := [0.**63297**74788, −0.**31586**10669, −**1.082427**720, −**1.579185**984, −**1.618252**806]

y := [0.**632980769**5, −0.**315859549**1, −**1.082428414**, −**1.579186594**, −**1.61825313**3]

y := [0.**632980769**7, −0.**315859549**0, −**1.082428414**, −**1.579186594**, −**1.61825313**4]

Comparing the second and fourth value with y_1 and y_2 of our previous ($n = 3$) solution, we can see that the solution is not yet very accurate. We can improve the accuracy of the $y(1)$ and $y(2)$ values by Richardson extrapolation, but now (having a general program) it is a lot easier to simply keep on doubling the value of n, until a sufficiently close agreement between consecutive solutions is reached. Note that to re-run the program using a new value of n, we first have to type:

restart:

Chapter 11 MATRIX' EIGENVALUES

Here we assume a rudimentary knowledge of matrix algebra (adding, subtracting and multiplying two matrices; we have already learned how to invert a square matrix). Let us go over the other basic matrix properties and related definitions:

Making rows of a matrix $\mathbb{A}$ to be columns of a new matrix $\mathbb{A}^{\mathrm{T}}$ (for a square matrix, this means flipping the matrix around its main diagonal) create the so called matrix TRANSPOSE. On can easily show that

$$(\mathbb{A}\mathbb{B})^{\mathrm{T}} = \mathbb{B}^{\mathrm{T}}\mathbb{A}^{\mathrm{T}}$$

A matrix which equals its own transpose (it must be square) is called SYMMETRIC. Note that a product of two symmetric matrices is not necessarily symmetric: $(\mathbb{A}\mathbb{B})^{\mathrm{T}} = \mathbb{B}^{\mathrm{T}}\mathbb{A}^{\mathrm{T}} = \mathbb{B}\mathbb{A}$ which is not equal to $\mathbb{A}\mathbb{B}$ in general (matrix multiplication is usually not COMMUTATIVE).

For an important class of matrices, $\mathbb{A}^{\mathrm{T}}$ is equal to $\mathbb{A}^{-1}$ (matrix inversion then becomes trivial). Such matrices are called ORTHOGONAL. One can show that a product of two orthogonal matrices remains orthogonal:

$$(\mathbb{P}_1\mathbb{P}_2)^{-1} = \mathbb{P}_2^{-1}\mathbb{P}_1^{-1} = \mathbb{P}_2^{\mathrm{T}}\mathbb{P}_1^{\mathrm{T}} = (\mathbb{P}_1\mathbb{P}_2)^{\mathrm{T}}$$

For a square matrix $\mathbb{A}$, a vector $\mathbf{x}$ such that

$$\mathbb{A}\mathbf{x} = \lambda\,\mathbf{x}$$

(i.e. when pre-multiplied by $\mathbb{A}$, $\mathbf{x}$ changes its length by a factor of λ, but not its direction) is called the matrix' EIGENVECTOR (λ is the corresponding EIGENVALUE). Eigenvectors and eigenvalues of real matrices may be complex in general. But, for symmetric matrices, they must be all real. To simplify our task (and avoid dealing with complex numbers), we will learn how to construct eigenvalues and eigenvectors of symmetric matrices only.

It is very easy to find eigenvalues (with their eigenvectors) of DIAGONAL matrices, right?

One can show that, when $\mathbb{S}$ is regular (and, of matching size), the new matrix

$$\mathbb{S}^{-1}\mathbb{A}\mathbb{S}$$

has the same eigenvalues (not eigenvectors) as $\mathbb{A}$ (modifying $\mathbb{A}$ in this manner is called SIMILARITY TRANSFORMATION).

Proof: $\mathbb{S}^{-1}\mathbb{A}\mathbb{S}\,\mathbf{y} = \lambda\,\mathbf{y}$ implies $\mathbb{A}\mathbb{S}\,\mathbf{y} = \lambda\,\mathbb{S}\,\mathbf{y}$. Thus, λ is and eigenvalue of $\mathbb{A}$, with the corresponding eigenvector $\mathbf{x} = \mathbb{S}\,\mathbf{y}$. □

Thus, if we can find a similarity transformation to result in a *diagonal* matrix $\mathbb{D}$ ($= \mathbb{S}^{-1}\mathbb{A}\mathbb{S}$), we have effectively found the eigenvalues (and eigenvectors - the columns of $\mathbb{S}$) of the original matrix (and this is how it is usually done). Working with symmetric matrices only actually gives us a further advantage: one can show that, when $\mathbb{A}$ is symmetric, it will be diagonalized by a similarity transformation with $\mathbb{S}$ not just regular, but also *orthogonal.*

Constructing the corresponding orthogonal matrix (let us call it $\mathbb{P}$ instead of $\mathbb{S}$, to emphasize the fact) will be a long and tedious process, consisting of a whole sequence of similarity transformations (each trying to get us a bit closer to a fully diagonal matrix), thus:

$$...\mathbb{P}_2^{\mathrm{T}}\mathbb{P}_2^{\mathrm{T}}\mathbb{P}_1^{\mathrm{T}}\mathbb{A}\mathbb{P}_1\mathbb{P}_2\mathbb{P}_3...$$

As we already know, a product of orthogonal matrices will remain orthogonal throughout this process.

First, we describe a procedre for making $\mathbb{A}$ (by an orthogonal similarity transformation) tri-diagonal (clearly a major step towards full diagonalization).

Householder's Method

Let us assume that $\mathbf{w}$ is a (column) vector whose NORM (defined as $\sqrt{\mathbf{w}^{\mathrm{T}}\mathbf{w}}$, i.e. the square root of the sum of squares of all its elements) is equal to 1. Then

$$\mathbb{P} \equiv \mathbb{I} - 2\,\mathbf{w}\,\mathbf{w}^{\mathrm{T}}$$

is clearly both symmetric and *orthogonal*, since $\mathbb{P}^2 = (\mathbb{I} - 2\,\mathbf{w}\,\mathbf{w}^{\mathrm{T}})^2 = \mathbb{I} - 4\,\mathbf{w}\,\mathbf{w}^{\mathrm{T}} + 4\,\mathbf{w}\,\mathbf{w}^{\mathrm{T}}\mathbf{w}\,\mathbf{w}^{\mathrm{T}} = -4\,\mathbf{w}\,\mathbf{w}^{\mathrm{T}} + 4\,\mathbf{w}\,\mathbf{w}^{\mathrm{T}} = \mathbb{I}$.

Suppose now that we have a symmetric matrix $\mathbb{A}$,and would like to make it tri-diagonal by an orthogonal similarity transformation. We first construct a vector $\mathbf{u}_1$ by::

1. Take the first column of $\mathbb{A}$ and replace its first element by zero.
2. Compute the norm (say h_1) of the remaining elements, and subtract it from the second element, when this element is positive (add otherwise). Note that the norm of the new vector is $H_1 = 2h_1(h_1 - |a_{21}|)$.

The matrix

$$\mathbb{P}_1 \equiv \mathbb{I} - \frac{2\,\mathbf{u}_1\,\mathbf{u}_1^{\mathrm{T}}}{H_1}$$

is clearly symmetric and orthogonal (by the previous argument), and

$$\mathbb{P}_1\mathbb{A}\mathbb{P}_1$$

is a similarity transformation of $\mathbb{A}$, which makes all but the first *two* elements in the first column (and row, due to symmetry) equal to 0 (we will skip the proof, but we will clearly see this 'at work' in our subsequent example).

We will then construct $\mathbf{u}_2$ by relacing the first two elements of the second row (of the resulting matrix) by 0, computing the (remaining element) norm h_2 and subtracting (adding) it from (to) the third element. Based on this, we compute

$$\mathbb{P}_2 \equiv \mathbb{I} - \frac{2\,\mathbf{u}_2\,\mathbf{u}_2^{\mathrm{T}}}{H_2}$$

and carry out the corresponding similarity transformation.

Continuing in this manner, we will transform the orginal matrix to its similar, tri-diagonal form in $n-2$ steps (n is the matrix dimension).

Example: To tri-diagonalize

$$\begin{bmatrix} 4 & 1 & -2 & 2 \\ 1 & 2 & 0 & 1 \\ -2 & 0 & 3 & -2 \\ 2 & 1 & -2 & -1 \end{bmatrix}$$

we have $h_1 = \sqrt{1^2 = (-2)^2 + 2^2} = 3$,

$$\mathbf{u}_1 = \begin{bmatrix} 0 \\ 1-3 \\ -2 \\ 2 \end{bmatrix}$$

$H_1 = 2 \cdot 3 \cdot (3-1) = 12$, and

$$\mathbb{P}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & \frac{2}{3} & \frac{2}{3} & -\frac{2}{3} \\ 0 & \frac{2}{3} & \frac{2}{3} & -\frac{2}{3} \\ 0 & -\frac{2}{3} & -\frac{2}{3} & \frac{2}{3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & -\frac{2}{3} & \frac{2}{3} \\ 0 & -\frac{2}{3} & \frac{1}{3} & \frac{2}{3} \\ 0 & \frac{2}{3} & \frac{2}{3} & \frac{1}{3} \end{bmatrix}$$

There are various 'shortcuts' for computing $\mathbb{P}_1\mathbb{A}\mathbb{P}_1$, but we will concentrate on the logic of the algorithm, rather than its efficient implementation. So we will let Maple do the evaluation, efficient or not:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & -\frac{2}{3} & \frac{2}{3} \\ 0 & -\frac{2}{3} & \frac{1}{3} & \frac{2}{3} \\ 0 & \frac{2}{3} & \frac{2}{3} & \frac{1}{3} \end{bmatrix} \begin{bmatrix} 4 & 1 & -2 & 2 \\ 1 & 2 & 0 & 1 \\ -2 & 0 & 3 & -2 \\ 2 & 1 & -2 & -1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{1}{3} & -\frac{2}{3} & \frac{2}{3} \\ 0 & -\frac{2}{3} & \frac{1}{3} & \frac{2}{3} \\ 0 & \frac{2}{3} & \frac{2}{3} & \frac{1}{3} \end{bmatrix}$$
$$= \begin{bmatrix} 4 & 3 & 0 & 0 \\ 3 & \frac{10}{3} & -\frac{4}{3} & -1 \\ 0 & -\frac{4}{3} & -1 & -\frac{4}{3} \\ 0 & -1 & -\frac{4}{3} & \frac{5}{3} \end{bmatrix}$$

Continuing to similarly reduce the second column (row), we get $h_2 = \sqrt{(-\frac{4}{3})^2 + (-1)^2} = \frac{5}{3}$

$$\mathbf{u}_1 = \begin{bmatrix} 0 \\ 0 \\ -\frac{4}{3} + \frac{5}{3} \\ -1 \end{bmatrix}$$

$H_2 = 2 \cdot \frac{5}{3} \cdot (\frac{5}{3} - \frac{4}{3}) = \frac{10}{9}$, and

$$\mathbb{P}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{5} & -\frac{3}{5} \\ 0 & 0 & -\frac{3}{5} & \frac{9}{5} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{4}{5} & \frac{3}{5} \\ 0 & 0 & \frac{3}{5} & -\frac{4}{5} \end{bmatrix}$$

So, finally, $\mathbb{P}_2\mathbb{P}_1\mathbb{A}\mathbb{P}_1\mathbb{P}_2$ equals to

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{4}{5} & \frac{3}{5} \\ 0 & 0 & \frac{3}{5} & -\frac{4}{5} \end{bmatrix} \begin{bmatrix} 4 & 3 & 0 & 0 \\ 3 & \frac{10}{3} & -\frac{4}{3} & -1 \\ 0 & -\frac{4}{3} & -1 & -\frac{4}{3} \\ 0 & -1 & -\frac{4}{3} & \frac{5}{3} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{4}{5} & \frac{3}{5} \\ 0 & 0 & \frac{3}{5} & -\frac{4}{5} \end{bmatrix}$$

$$= \begin{bmatrix} 4 & 3 & 0 & 0 \\ 3 & \frac{10}{3} & -\frac{5}{3} & 0 \\ 0 & -\frac{5}{3} & -\frac{33}{25} & -\frac{68}{75} \\ 0 & 0 & -\frac{68}{75} & \frac{149}{75} \end{bmatrix}$$

which has the desired tri-diagonal form. We know that this matrix (let us call it $\mathbb{A}_0$) has the same eigenvalues as the original one.

Making a symmetric matrix tri-diagonal is the easy part. Making a tri-diagonal matrix fully diagonal is a lot more difficult, and can be achieved only approximately (even though, to an arbitrary accuracy) by iterating (until the off diagonal elements 'nearly' disappear). This is what we will discuss now.

QR Decomposition

Any matrix can be written as a product of an orthogonal matrix, say $\mathbb{Q}$, and an upper triangular matrix $\mathbb{R}$. We need to learn how to do this with our tri-diagonal matrices only. Once we have achieved that, i.e. found $\mathbb{Q}$ and $\mathbb{R}$ such that

$$\mathbb{A}_0 = \mathbb{Q}\,\mathbb{R}$$

it should be clear that

$$\mathbb{A}_1 \equiv \mathbb{R}\,\mathbb{Q} = \mathbb{Q}^{\mathrm{T}}\mathbb{A}_0\mathbb{Q}$$

is a similarity transformation of $\mathbb{A}_0$. Not only that, $\mathbb{A}_1$ remains symmetrical (quite obvious) and tri-diagonal (not so obvious, but we will see it in our examples). Finally (and this is the most important part), its off diagonal elements *decrease* in magnitude (which means that we can do this *repeatedly* until they practically disappear). At that point, we are done: we have found the eigenvalues of the original matrix to be the main diagonal elements of the resulting one.

So the only missing part of the algorithm is carrying out the QR decomposion. For this purpose we must define a special kind of *orthogonal* 2 by 2 matrix called ROTATION, which has the follwing general form

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix}$$

where $s^2 + c^2 = 1$. One can easily verify that the matrix is orthogonal, thus:

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \begin{bmatrix} c & -s \\ s & c \end{bmatrix} = \begin{bmatrix} c^2+s^2 & -c\,s+s\,c \\ -s\,c+c\,s & s^2+c^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Note that is we replace, in a unit matrix of *any size*, one of the main-diagonal 2 by 2 submatrices by the previous matrix, e.g.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c & s & 0 \\ 0 & -s & c & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

the resulting matrix is still orthogonal.

The QR decomposition is performed as follows: we premultiply $\mathbb{A}_0$ by a rotation constructed to make the first below-diagonal element equal to zero. This is achieved by using the following two numbers for c and s, respectively:

1. take the first two elements of column 1,
2. change the sign of the second one,
3. and 'normalize' them (divide each by the sqaure root of their sum of squares).

Then, similarly, construct and apply another rotation to eliminate the second below-diagonal element, etc., until you eliminate all below-diagonal elements.

Example: Suppose we want to construct the QR decomposition of the following tridiagonal matrix:

$$\mathbb{A}_0 \equiv \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 3 & 2 & 0 \\ 0 & 2 & -3 & 4 \\ 0 & 0 & 4 & 1 \end{bmatrix}$$

First we premultiply by

$$\mathbb{P}_1 = \begin{bmatrix} \frac{2}{\sqrt{5}} & -\frac{1}{\sqrt{5}} & 0 & 0 \\ \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

to obtain

$$\begin{bmatrix} \sqrt{5} & -\sqrt{5} & -\frac{2}{5}\sqrt{5} & 0 \\ 0 & \sqrt{5} & \frac{4}{5}\sqrt{5} & 0 \\ 0 & 2 & -3 & 4 \\ 0 & 0 & 4 & 1 \end{bmatrix}$$

then by

$$\mathbb{P}_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \frac{\sqrt{5}}{3} & \frac{2}{3} & 0 \\ 0 & -\frac{2}{3} & \frac{\sqrt{5}}{3} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

(note that now c and s have been computed based on the second and third elements of row 2), to get

$$\begin{bmatrix} \sqrt{5} & -\sqrt{5} & -\frac{2}{5}\sqrt{5} & 0 \\ 0 & 3 & -\frac{2}{3} & \frac{8}{3} \\ 0 & 0 & -\frac{23}{15}\sqrt{5} & \frac{4}{3}\sqrt{5} \\ 0 & 0 & 4 & 1 \end{bmatrix}$$

and finally by

$$\mathbb{P}_3 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -\frac{23}{\sqrt{1249}} & \frac{60}{\sqrt{6245}} \\ 0 & 0 & -\frac{60}{\sqrt{6245}} & -\frac{23}{\sqrt{1249}} \end{bmatrix}$$

(based on the last two elements of column 3), to get the resulting $\mathbb{R}$ matrix

$$\begin{bmatrix} \sqrt{5} & -\sqrt{5} & -\frac{2}{5}\sqrt{5} & 0 \\ 0 & 3 & -\frac{2}{3} & \frac{8}{3} \\ 0 & 0 & \frac{1}{15}\sqrt{6245} & -\frac{56}{3747}\sqrt{6245} \\ 0 & 0 & 0 & -\frac{103}{1249}\sqrt{1249} \end{bmatrix}$$

Note that that the corresponding $\mathbb{Q} = \mathbb{P}_1^T\mathbb{P}_2^T\mathbb{P}_3^T$ (this follows from $\mathbb{P}_3\mathbb{P}_2\mathbb{P}_1\mathbb{A}_0 = \mathbb{R}$).

So we have just constructed the QR decomposition of $\mathbb{A}_0$. What we really want in the end is $\mathbb{A}_1 = \mathbb{R}\,\mathbb{P}_1^T\mathbb{P}_2^T\mathbb{P}_3^T$, which should turn out to be symmetric and tri-diagonal. This takes only some extra matrix multiplication (note that in each step, only two columns of $\mathbb{R}$ change - an efficient program would take that into consideration, since full-matrix multiplication is very 'expensive'), yielding

$$\mathbb{A}_1 = \begin{bmatrix} 3 & -\frac{3}{5}\sqrt{5} & 0 & 0 \\ -\frac{3}{5}\sqrt{5} & \frac{14}{9} & \frac{2}{45}\sqrt{6245} & 0 \\ 0 & \frac{2}{45}\sqrt{6245} & -\frac{38807}{11241} & -\frac{1236}{1249}\sqrt{5} \\ 0 & 0 & -\frac{1236}{1249}\sqrt{5} & \frac{2369}{1249} \end{bmatrix} =$$

$$\begin{bmatrix} 3.0 & -1.341640787 & 0 & 0 \\ -1.341640787 & 1.555555556 & 3.512236106 & 0 \\ 0 & 3.512236106 & -3.452272929 & -2.212794252 \\ 0 & 0 & -2.212794252 & 1.896717374 \end{bmatrix}$$

This procedure can then be repeated to build $\mathbb{A}_2$, $\mathbb{A}_3$, etc. To make it easier, we will use the following Maple procedure called **step,** which takes $\mathbb{A}_i$ as its argument, and returns $\mathbb{A}_{i+1}$:

```
step := proc(A) local n, a, i, j, p;    n := rowdim(A):
for i to n − 1 do   a := submatrix(A, i..i + 1, 1..n):
p[i] := matrix(2, 2, [a[1, i], a[2, i], −a[2, i], a[1, i]]) / sqrt(a[1, i]ˆ2 + a[2, i]ˆ2):
a := evalm(p[i] &∗ a):
for j to n do A[i, j] := a[1, j]:  A[i + 1, j] := a[2, j]  end do:    end do:
for i to n − 1 do   a := submatrix(A, 1..n, i..i + 1):
a := evalm(a &∗ transpose(p[i]) ):
for j to n do A[j, i] := a[j, 1]:  A[j, i + 1] := a[j, 2]  end do:    end do:
A  end:
```

To get $\mathbb{A}_2$ (assuming that we have already computed $\mathbb{A}_1$), all we have to do is type:

$A_2 := step(A_1);$

and getting

$$\begin{bmatrix} 3.759259259 & -1.477425003 & 0 & 0 \\ -1.477425003 & -1.705370807 & 4.388847474 & 0 \\ 0 & 4.388847474 & -0.7240742723 & 0.9371079120 \\ 0 & 0 & 0.9371079120 & 1.670185820 \end{bmatrix}$$

In this form, the procedure converges rather slowly (eventually, we will have to do something about that!), so we quote the result of every thent iteration only:

$$\mathbb{A}_{10} = \begin{bmatrix} -0.0778492650 & -5.192390056 & 0 & 0 \\ -5.192390056 & -1.106701279 & 0.0320364388 & 0 \\ 0 & 0.0320364388 & 2.830526381 & 0.0021280394 \\ 0 & 0 & 0.0021280394 & 1.354024164 \end{bmatrix}$$

$$\mathbb{A}_{20} = \begin{bmatrix} -5.726428187 & -.9309776898 & 0 & 0 \\ -.9309776898 & 4.542070063 & 0.00009717882 & 0 \\ 0 & 0.00009717882 & 2.830337028 & -0.0000006391 \\ 0 & 0 & -0.0000006391 & 1.354021096 \end{bmatrix}$$

$$\mathbb{A}_{30} = \begin{bmatrix} -5.809267571 & -0.09602590019 & 0 & 0 \\ -0.09602590019 & 4.624909454 & 0.0000007118 & 0 \\ 0 & 0.0000007118 & 2.830337023 & -0.0000000004 \\ 0 & 0 & -0.0000000004 & 1.354021096 \end{bmatrix}$$

$$\mathbb{A}_{40} = \begin{bmatrix} -5.810141972 & -0.0098268078 & 0 & 0 \\ -0.0098268078 & 4.625783856 & 0.0000000052 & 0 \\ 0 & 0.0000000052 & 2.830337023 & 0 \\ 0 & 0 & 0 & 1.354021096 \end{bmatrix}$$

$$\mathbb{A}_{50} = \begin{bmatrix} -5.810151128 & -0.0010055427 & 0 & 0 \\ -0.0010055427 & 4.625793015 & 0 & 0 \\ 0 & 0 & 2.830337025 & 0 \\ 0 & 0 & 0 & 1.354021096 \end{bmatrix}$$

$$\mathbb{A}_{60} = \begin{bmatrix} -5.810151219 & -0.00010289358 & 0 & 0 \\ -0.00010289358 & 4.625793111 & 0 & 0 \\ 0 & 0 & 2.830337017 & 0 \\ 0 & 0 & 0 & 1.354021096 \end{bmatrix}$$

$$\mathbb{A}_{70} = \begin{bmatrix} -5.810151211 & -0.0000105287 & 0 & 0 \\ -0.0000105287 & 4.625793102 & 0 & 0 \\ 0 & 0 & 2.830337011 & 0 \\ 0 & 0 & 0 & 1.354021096 \end{bmatrix}$$

$$\mathbb{A}_{80} = \begin{bmatrix} -5.810151211 & -0.00000107738 & 0 & 0 \\ -0.00000107738 & 4.625793102 & 0 & 0 \\ 0 & 0 & 2.830337011 & 0 \\ 0 & 0 & 0 & 1.354021096 \end{bmatrix}$$

at which point the main diagonal elements no longer change (even though we still have not reached a full convergence - the off-diagonal elements have not quite disappeared yet).

Based on this example, we can make a few general observations:

1. The procedure converges much too slowly!
2. The convergence is the fastest at the lower right corner, and slowly drifts up the main diagonal.
3. The eigenvalues are extracted from the smallest (in magnitude) - the lower right corner, to the largest (upper left corner).

The main issue is now: how to speed up the convergence? This is achieved by the so called SHIFT, which works as follows: If, at any stage of the process, we subtract from $\mathbb{A}_i$ a multiple of the unit matrix, say $s\,\mathbb{I}$, all its eigenvalues are reduced by s (the eigenvectors are not affected). The closer we can bring the smallest (in magnitude) eignevalue to zero, the faster the procedure will converge. So it all hinges on how to estimate the smallest eigenvalue. This we do as folows:

We compute the two eignavalues of the lower right 2 by 2 'corner' (submatrix) of $\mathbb{A}_i$, say

$$\begin{bmatrix} a & b \\ b & c \end{bmatrix}$$

by using the following formula

$$\frac{a+c}{2} \pm \sqrt{\left(\frac{a-c}{2}\right)^2 + b^2}$$

and take the one smaller in magnitude to be our s. Then we subtract $s\,\mathbb{I}$ from $\mathbb{A}_i$, before we carry out the next *RQ* step. Then, we repeat the whole procedure (find the eigenvalues of the lower right corner, subtract the new $s\,\mathbb{I}$, etc.) until convergence is reached (this will now happen a lot faster: each iteration will *triple* the number of correct digits in the last eignenvalue - the so called CUBIC convergence). We of course have to keep track of all the s values, and make the corresponding adjustment in the resulting eigenvalues.

There is one more trick which will speed up the computation: once we reduce the off-diagonal element in the last row (and column) to zero, the corresponding main diagonal element *is* one of the eigenvalues. We may then delete the last row and column (the so called DEFLATION) and continue the process with a matrix of a smaller size. We continue this till we reach a 2 by 2 matrix, for which the eigenvales can be computed analytically.

Example: Starting with the same

$$\mathbb{A}_0 \equiv \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 3 & 2 & 0 \\ 0 & 2 & -3 & 4 \\ 0 & 0 & 4 & 1 \end{bmatrix}$$

as before, we first find the eigenvalues of the lower right corner:

$$-1 \pm \sqrt{20}$$

so $s = -1 + \sqrt{20}$. Our **step** procedure is then applied to

$$\begin{bmatrix} 3-\sqrt{20} & -1 & 0 & 0 \\ -1 & 4-\sqrt{20} & 2 & 0 \\ 0 & 2 & -2-\sqrt{20} & 4 \\ 0 & 0 & 4 & 2-\sqrt{20} \end{bmatrix}$$

resulting in

$$\begin{bmatrix} -2.086016217 & -1.127928288 & 0 & 0 \\ -1.127928288 & -6.142517389 & 4.548831156 & 0 \\ 0 & 4.548831156 & -2.229740719 & -0.7820049124 \\ 0 & 0 & -0.7820049124 & -0.4302694951 \end{bmatrix}$$

Similarly, the next s is -0.1379246338, which needs to be subtracted from the main diagonal. Used as the argument of **step**, we obtain:

$$\begin{bmatrix} -3.944739819 & -3.252698362 & 0 & 0 \\ -3.252698362 & -7.003053686 & 1.042796325 & 0 \\ 0 & 1.042796325 & 1.071996644 & 0.2847589388 \\ 0 & 0 & 0.2847589388 & -0.4610484219 \end{bmatrix}$$

And, two more times: $s = -0.5122327109$

$$\begin{bmatrix} -8.127461129 & -1.846776024 & 0 & 0 \\ -1.846776024 & -1.827792430 & .6901018070 & 0 \\ 0 & .6901018070 & 1.658979780 & 0.0013572467 \\ 0 & 0 & 0.0013572467 & 0.0083593393 \end{bmatrix}$$

$s = 0.008358223295$

$$\begin{bmatrix} -8.624456904 & -0.3419015412 & 0 & 0 \\ -0.3419015412 & -1.287002968 & 0.7941414393 & 0 \\ 0 & 0.7941414393 & 1.590112347 & 0.0000000003 \\ 0 & 0 & 0.0000000003 & 0.0000001888 \end{bmatrix}$$

So, after four iterations, we found the first eigenvalue, equal to

$$0.0000001888 + 0.008358223295 - 0.5122327109 - 0.1379246338 - 1 + \sqrt{20}$$
$$= 2.83033\,7023$$

Note that, without shift, we needed someting like twenty iterations to get this value. Also note that, when we use shift, we don't necesarily extract the smallest eigenvalue first.

Deflating the matrix to

$$\begin{bmatrix} -8.624456904 & -0.3419015412 & 0 \\ -0.3419015412 & -1.287002968 & 0.7941414393 \\ 0 & 0.7941414393 & 1.590112347 \end{bmatrix}$$

we find that the next s equals to -1.491646077. Subtracting it from the main diagonal elements of the previous matrix, and applying **step,** results in:

$$\begin{bmatrix} -7.148692727 & -0.03946443807 & 0 \\ -0.03946443807 & 3.285889993 & -0.0589673574 \\ 0 & -0.0589673574 & 0.01639343608 \end{bmatrix}$$

Repeating one more time: $s = 0.0153302697$

$$\begin{bmatrix} -7.164141138 & -0.01802010583 & 0 \\ -0.01802010583 & 3.271740963 & 0.00000000127 \\ 0 & 0.00000000127 & 0.00000007064 \end{bmatrix}$$

which is sufficient to get the next eigenvalue, thus:

$$\begin{aligned} & 0.00000007064 + 0.0153302697 - 1.491646077 + \\ & +0.008358223295 - 0.5122327109 - 0.1379246338 - 1 + \sqrt{20} \\ & = 1.35402\,1097 \end{aligned}$$

Note that we had to adjust the eigenvalue by adding cumulative sum of *all* the s values so far.

Deflating the last matrix yields

$$\begin{bmatrix} -7.164141138 & -0.01802010583 \\ -0.01802010583 & 3.271740963 \end{bmatrix}$$

whose eigenvalues are -7.164172254 and 3.271772079. Adjusted by the same cumulative sum, these yield $-5.81015\,1227$ and $4.62579\,3106$ for the last two eigenvalues of the original matrix. Using shift and deflation, we have thus been able to extract all four eigenvalues in six iterations. This is a substancial improvement over 80 iterations of the previous example.