1 NUMERIC VERSUS SYMBOLIC COMPUTATION

Wolfram Koepf

Konrad-Zuse-Zentrum
Takustr. 7
D-14195 Berlin
Germany

koepf@zib.de

Abstract: Twentyfive years ago when I studied Physics, only one of the students who participated in a laboratory course that I took was possessing one of the first *calculators* to process the data we were obtaining. Believe it or not, everybody else, including me, had to use a *slide-rule* for this purpose. Nowadays, the calculator is used by everybody, by far not only for academic purposes. Hence it is the responsibility of the school education, and here in particular of the Mathematics education, to take this situation into account, and to teach our children the (intelligent) use of a calculator.

In my opinion, there is no doubt that sooner or later computer algebra systems like DERIVE or Mathematica will be used by everybody in the same way as calculators are used today. Obviously this gives us a new responsibility to integrate computer algebra systems in the Math curriculum and to teach the students the use of them. When I realized this, I began to use DERIVE in my calculus courses at the Free University Berlin in particular for the Math teacher education [5]–[6].

Whereas calculators brought more numeric computation into the classroom, computer algebra systems enable the use of more symbolic computation. In this presentation, I would like to give examples how numeric and symbolic computations need each other. These examples show in particular with which

type of mathematical problems the Math education can and should be enhanced by the use of DERIVE.

Some of my examples might be considered to be too advanced or too far away from the current curriculum which on the other hand differs quite a lot in all the different countries which the participants of this meeting come from. Rather than being static material, my examples are considered to supply *ideas* to Math teachers about interesting and important *concepts* that might be incorporated in future Math education in connection with the use of computer algebra systems.

WHY DERIVE?

First I would like to justify by an example why I find DERIVE more appropriate for educational purposes than other systems like Axiom, Macsyma, Maple, Mathematica, MuPAD or REDUCE, even though in my research I mainly use the other systems.

Leibniz, one of the developers of the differential and integral calculus, still doubted the existence of an elementary antiderivative of the rational function

$$\frac{1}{1+x^4}\ .$$

The reason was that he was not able to find a proper *real factorization* of the denominator polynomial

$$1+x^4$$

in terms of quadratics. The factorization of this polynomial expression by any of the mentioned systems returns the input. This seems to approve Leibniz's opinion. DERIVE's handling is different: Applying the `Factor` menu to the input $1+x^4$ results in a submenu asking the user

```
Factor:  Amount:  Trivial Squarefree Rational raDical Complex
```

This question gives the user—e.g. the experimenting student—the necessary information about the fact that there are different algorithms that might be applied. Being satisfied with a rational factorization—this is exactly what the other systems do—no proper factors are found. However, allowing square roots using the `raDical` option generates the factors

$$1+x^4 = (x^2+\sqrt{2}\,x+1)\,(x^2-\sqrt{2}\,x+1)\ .$$

With this factorization in mind, it is easy to find the partial fraction decomposition

$$-\frac{\sqrt{2}\,x}{4\,(x^2-\sqrt{2}\,x+1)}+\frac{1}{2\,(x^2-\sqrt{2}\,x+1)}+\frac{\sqrt{2}\,x}{4\,(x^2+\sqrt{2}\,x+1)}+\frac{1}{2\,(x^2+\sqrt{2}\,x+1)}$$

for $1/(1+x^4)$ (with $\boxed{\texttt{Expand}}$), and to understand the antiderivative

$$\int \frac{1}{1+x^4}\,dx = \frac{\sqrt{2}\,\arctan(\sqrt{2}\,x-1)}{4} + \frac{\sqrt{2}\,\arctan(\sqrt{2}\,x+1)}{4} - \frac{\sqrt{2}\,\ln\left(\frac{x^2-\sqrt{2}\,x+1}{x^2+\sqrt{2}\,x+1}\right)}{8}$$

which is returned by DERIVE's integrator.

It is this type of handling that gives DERIVE an educational advantage.

HOFSTADTER'S PROBLEM

To get started I would like to consider an example from geometry that was discovered by Douglas Hofstadter. When experimenting with a geometry program in connection with the *Morley Triangle Theorem*, Hofstadter discovered new centers of triangles, and realized the collinearity of some of them. This was done by first observing these patterns by a *visual approach*, and then checking his conjectures by *numerical computations.*

Numerical computations can give one a safe feeling about what is happening, but they do not provide a mathematical proof. Hence, what's next?

Hofstadter's problem can be converted to the determinant condition

$$\begin{vmatrix} \frac{\sin(r\alpha)}{\sin((1-r)\alpha)} & \frac{\sin(2\alpha)}{\sin(-\alpha)} & \frac{\sin((2-r)\alpha)}{\sin((r-1)\alpha)} \\ \frac{\sin(r\beta)}{\sin((1-r)\beta)} & \frac{\sin(2\beta)}{\sin(-\beta)} & \frac{\sin((2-r)\beta)}{\sin((r-1)\beta)} \\ \frac{\sin(r\gamma)}{\sin((1-r)\gamma)} & \frac{\sin(2\gamma)}{\sin(-\gamma)} & \frac{\sin((2-r)\gamma)}{\sin((r-1)\gamma)} \end{vmatrix} = 0 \qquad (1.1)$$

under the assumption that α, β and γ denote the angles of the triangle, and therefore $\alpha + \beta + \gamma = \pi$.

DERIVE can *prove* that this statement is true, hence verifying Hofstadter's observation! On the other hand, without the knowledge about certain details of DERIVE's capabilities there is no chance to receive this information, i.e. to simplify the left hand side of (1.1) to zero (compare [3])!

If we enter the determinant under consideration, and substitute γ by π–α–β, then DERIVE returns a huge expression after simplification, and not zero.

But observe that it is much easier to discover that a rational expression equals zero, if it has the form numerator/denominator because then the numerator must equal zero, and one can forget the denominator. Since the setting $\boxed{\texttt{Factor Trivial}}$ brings any rational expression in this form (did you know this?), that's next. Still, DERIVE's result does not equal zero.

The reason is that trigonometric simplifications have to be carried out. To be on the safe side, one should use one of the settings $\boxed{\texttt{Manage Trigonometry}}$ `Toward` $\boxed{\texttt{Sines}}$ or $\boxed{\texttt{Cosines}}$. This guarantees that the replacements $\cos^2 x = 1 - \sin^2 x$ or $\sin^2 x = 1 - \cos^2 x$, respectively, are applied whenever possible. Otherwise $\sin^2 x + \cos^2 x$-terms might still be hidden, and zero cannot be discovered.

If we use $\boxed{\texttt{Manage Trigonometry Expand}}$ `Toward` $\boxed{\texttt{Sines}}$, Hofstadter's expression easily simplifies to zero, as announced. This finishes my DERIVE proof of Hofstadter's Theorem.

As an exercise I ask the reader to prove that (1.1) is true for arbitrary γ, not necessarily equal to $\pi - \alpha - \beta$! This identity is the reason for Hofstadter's success.

ILL-CONDITIONED PROBLEMS

Next, let's consider the problem to calculate the definite integral

$$I_n := \int_0^1 x^n \, e^{x-1} \, dx$$

for *large* n, say for $n = 1\ 000$ or $n = 1\ 000\ 000$ to as many digits as possible (see [9], [8]).

Since for $x \in [0, 1]$ the relations

$$ex \leq e^x \leq e$$

are valid, s. Figure 1.1, we get for I_n the inequalities

$$\int_0^1 x^{n+1} \, dx < I_n < \int_0^1 x^n \, dx$$

or

$$\frac{1}{n+2} < I_n < \frac{1}{n+1} \ . \tag{1.2}$$

Therefore I_n is decreasing and $\lim\limits_{n\to\infty} I_n = 0$:

$$1 > I_0 > \frac{1}{2} > I_1 > \frac{1}{3} > \cdots > \frac{1}{n+1} > I_n > \frac{1}{n+2} > \cdots > \lim_{n\to\infty} I_n = 0 \ .$$

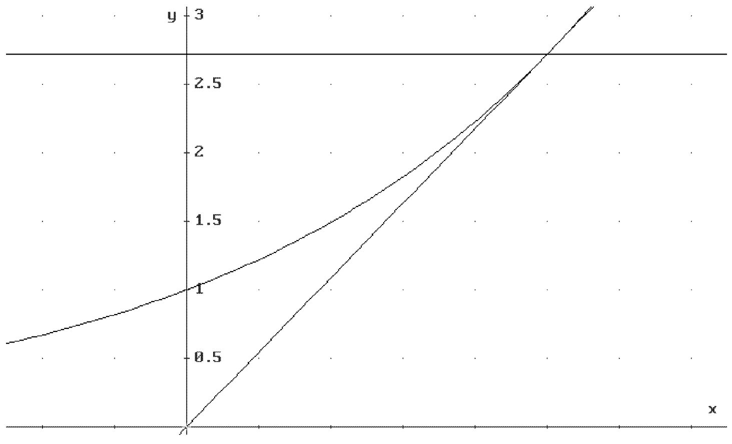


Figure 1.1 The inequalities $ex \leq e^x \leq e$

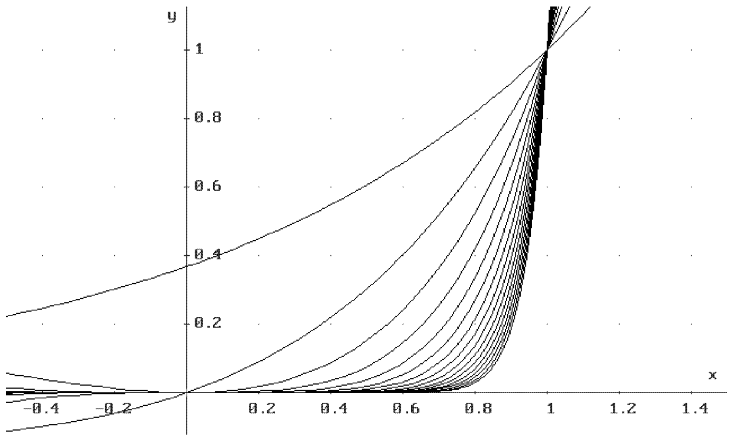


Figure 1.2 The integrands $x^n\, e^{x-1}$

This can also been seen from Figure 1.2, where the integrands are shown for $n = 0, \ldots, 15$.

Using $\boxed{\texttt{approX}}$, one can try to use DERIVE's numerical integration capabilities. On the other hand, for large n this is not practicable and rather inefficient since the integrand has large slope (namely $n + 1$) at the boundary point $x = 1$. For $n = 1\,000\,000$ this is absolutely hopeless. DERIVE generates the message `Dubious Accuracy`.

So we pin our hopes in symbolic integration! To meet the point, let's do it for $n = 20$. We get the exact value

$$\begin{aligned} I_{20} &= \int_0^1 x^{20}\, e^{x-1}\, dx = 895\ 014\ 631\ 192\ 902\ 121 - \frac{2\ 432\ 902\ 008\ 176\ 640\ 000}{e} \\ &= \quad 895\ 014\ 631\ 192\ 902\ 121 \\ &\quad - 895\ 014\ 631\ 192\ 902\ 120.95445\ 51159\ 2418\ . \end{aligned}$$

If we do this numeric computation with the default number of 6 digits, we cannot get anything of value since all digits are lost by the final *catastrophic subtraction cancellation*. If we use higher precision, 17 digits are lost. We will see that I_n is always of the form $m - k/e$ $(m, e \in \mathbb{N})$ with very large m, so the same effect occurs. Hence there is no chance to calculate I_n for large n by this method efficiently.

Our next idea is the use of a *recurrence equation* for I_n. By partial integration, one easily discovers that (for example with DERIVE ([5], Chapter 11, [8]))

$$\begin{aligned} I_n &= 1 - n\, I_{n-1} \qquad (n > 0) &(1.3) \\ I_0 &= 1 - \frac{1}{e} = 0.63212\ 05588\ 28557\ 6784...\ . \end{aligned}$$

If we iterate this recurrence equation, then by induction the formula

$$I_n = 1 - n + n(n-1) + \cdots + (-1)^n\, n! - \frac{(-1)^n\, n!}{e} = m - \frac{(-1)^n\, n!}{e} \qquad (1.4)$$

is deduced, confirming what we already mentioned: The exact value of I_n is the difference of an integer m and the ratio $\frac{(-1)^n\, n!}{e}$. Since $\lim\limits_{n\to\infty} I_n = 0$ this is always the difference of two approximately equal terms of size $n!/3$.

By the definition

```
I(n):=IF(n=0,1-1/EXP(1),1-n*I(n-1))
```

DERIVE can be used to calculate I_n recursively according to (1.3). However, if we approximate `VECTOR(I(n),n,0,20)` we see that we get nonsense, again. What's going on?

Unfortunately, the same effect as before applies to the current situation, this time, however, *in each single iteration step*! Since

$$\lim_{n\to\infty} n\, I_{n-1} = 1$$

(see (1.2)), the application of (1.3) corresponds to the subtraction of two approximately equal numbers, leading to cancellation again. If $n \approx 10$, we lose approximately one digit per iteration step, whereas if $n \approx 10^6$, then we already lose 6 digits per step! Hence, an application of (1.3) in this form gives us no chance at all to calculate I_{10^6}, besides the fact that a 10^6-times iteration anyway would be rather costly. Now we are in big trouble: Is there at all any method to calculate I_{10^6}?

The solution is the following: Similarly as the recurrence equation in the form $I_n := 1 - n\,I_{n-1}$ is especially ill-conditioned since $n\,I_{n-1} \approx 1$, the approximation being better the larger n is, the application of the recursion in *backward direction*

$$I_n := \frac{1 - I_{n+1}}{n+1} \tag{1.5}$$

is especially well-conditioned: No cancellation occurs. On the contrary decimal places are *won* in each iteration step since one divides by $n + 1$. This implies that we gain approximately one decimal place if $n \approx 10$, and approximately six decimal places if $n \approx 10^6$! This sounds great!

There is still one problem: While the recursion in forward direction has a natural initialization $I_0 = 1 - \frac{1}{e}$, such an initialization does not exist in the backward direction. Hence are we lost again? No! Since the application of the recursion in backward direction raises the precision in each iteration step, we may start with the bad (but not too bad) initial value $I_{n+k} = 0$, iterating (1.5) k-times to calculate I_n. We have only to find an appropriate number k (a detailed error analysis can be done!) to ensure that the calculated decimal places indeed are correct.

Using this approach, with only 20 iterations we get 50 correct digits of

$$I_{1000} = 0.00099\ 80049\ 85051\ 79787\ 28810\ 31699\ 38385\ 79102\ 38591\ 42748\ 629\ldots\,,$$

and even only 10 iterations are needed for the calculation of

$$I_{10^6} = 0.00000\ 09999\ 98000\ 00499\ 99850\ 00051\ 99979\ 70008\ 76995\ 86002\ 11468\ 8\ldots\,!$$

The first result is obtained by defining

```
IBACK(n):=IF(n=1020,0,(1-IBACK(n+1))/(n+1))
```

and calculating `IBACK(1000)`, whereas the second result follows with

```
IBACK(n):=IF(n=1000010,0,(1-IBACK(n+1))/(n+1))
```

simplifying `IBACK(1000000)` with $\boxed{\texttt{approX}}$.

ITERATIVE COMPUTATION OF π

In this section I consider another ill-conditioned problem which can be resolved by symbolic techniques ([9], [8]).

Already Archimedes calculated the number π by an approximation of the circumference of a circle by inscribed regular polygons. We take the unit circle and call the sidelength of an inscribed regular n-gon s_n.

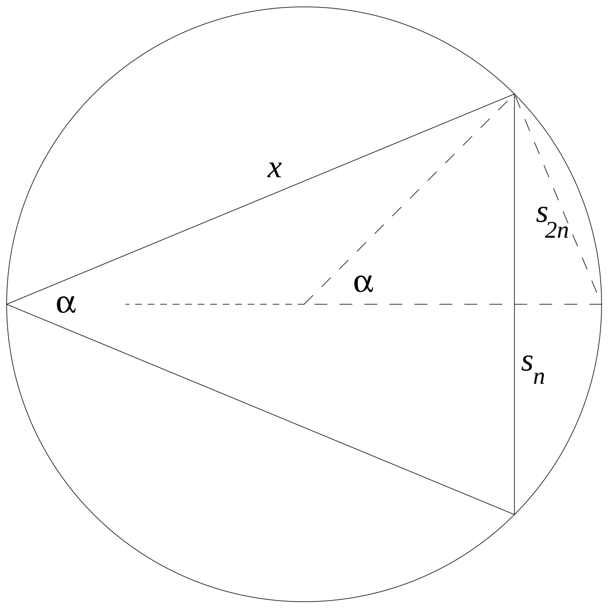


Figure 1.3 Calculation of s_n by doubling the number of vertices

In Figure 1.3 we find two similar triangles. Hence we have

$$\frac{s_{2n}}{1} = \frac{s_n}{x} \ .$$

For x, we furthermore deduce by the Theorems of Thales and Pythagoras

$$s_{2n}^2 + x^2 = 4 \ .$$

An elimination of x from these two equations (which can be easily done with DERIVE), yields

$$s_{2n} = \sqrt{2 - \sqrt{4 - s_n^2}} \ . \tag{1.6}$$

Since the n-gon has n sides of equal length s_n, for its circumference we have

$$C_n = n\, s_n \to 2\pi\ .$$

We can use the initialization with a square, for which we have $s_4 = \sqrt{2}$ and $C_4 = 4\sqrt{2}$ (or also with a 6-gon with $s_6 = 1$ and $C_6 = 6$). If $n = 2^k$ $(k \geq 2)$ is a power of two then we can use the recursive formula

$$C_n := n\, s_n\ , \quad s_n = \begin{cases} \sqrt{2} & \text{if } n = 4 \\ \sqrt{2 - \sqrt{4 - s_{n/2}^2}} & \text{if } n = 2^k\ (k > 2) \end{cases}$$

for the calculation of C_n.

These formulas lead to the DERIVE functions

```
S(n):=IF(n=4,SQRT(2),SQRT(2-SQRT(4-S(n/2)^2)))

P(n):=n*S(n)/2
```

where $p_n = C_n/2$ converges to π. If we use this DERIVE function to calculate p_n, we get unfortunately $p_{2^{11}} = 0$, far away from π.

Again, this effect is caused by subtraction cancellation! Since for $n \to \infty$ the side length s_n tends toward zero, we have moreover

$$\sqrt{4 - s_n^2} \to 2\ .$$

Therefore in the calculation of s_{2n} according to (1.6) the difference of two numbers occurs which both are approximately equal to 2. This leads to the completely worthless intermediate result $s_{2^{11}} = 0$.

How can we resolve this problem? Is there a way to avoid the dangerous subtraction? It is easy to see (for example using DERIVE) that we can rewrite the main expression in the following way

$$\sqrt{2 - \sqrt{4 - s^2}} = \frac{s}{\sqrt{2 + \sqrt{4 - s^2}}}\ .$$

This simple trick has eliminated the unwelcome subtraction. Hence we get a better approximation of π (with a working precision of 25 digits)

$$\pi \approx 3.14159\ 26535\ 89788\ 64861\ 1672...\ .$$

On the other hand, approximating π, we get

$$\pi = 3.14159\ 26535\ 89793\ 23846\ 2643...\ , \qquad (1.7)$$

hence only 14 digits are valid. This is *not* due to a bad condition of the algorithm (other systems don't fail) but to a *simplification problem* of DERIVE: The expression

$$f(s) := \frac{s}{\sqrt{2+\sqrt{4-s^2}}}$$

is not approximated accurately enough for small s. DERIVE's internal simplification mechanism seems to convert our well-conditioned formula to an ill-conditioned one, again. Substituting $s = 2^{-30}$ into $f(s)$, e.g., and applying $\boxed{\texttt{Simplify}}$ yields

$$f(2^{-30}) = \frac{\sqrt{4\ 294\ 967\ 298}}{65\ 536} - \frac{\sqrt{4\ 294\ 967\ 294}}{65\ 536} \ ,$$

so that with $\boxed{\texttt{approX}}$ subtraction cancellation occurs, again. The similar representation

$$f(2^{-30}) = \frac{\sqrt{2}}{32\ 768\sqrt{2\ 147\ 483\ 649} + 32\ 768\sqrt{2\ 147\ 483\ 647}}$$

would be much better for numerical purposes. Hence in the given case, we have even to outwit DERIVE!

To do so, we summarize that $f(s)$ is treated badly, for $0 < s < 10^{-5}$, say. For so small numbers, we may replace $f(s)$ by a polynomial approximation which is good in a neighborhood of $s = 0$, hence we calculate the Taylor approximation `TAYLOR(s/SQRT(2+SQRT(4-s^2)),s,0,5)` giving

$$f(s) \approx \frac{s}{2} + \frac{s^3}{64} + \frac{7\,s^5}{4096} \ .$$

With the implementation

```
S_AUX(s):=IF(s<10^-5,s/2+s^3/64+7*s^5/4096,s/SQRT(2+SQRT(4-s^2)))

S(n):=IF(n=4,SQRT(2),S_AUX(S(n/2)))

P(n):=n*S(n)/2
```

the approximation of $p(2^{50})$ yields all 25 digits of (1.7) correctly!

WHERE IS THE SECOND POLE?

Many people say that the use of graphic calculators and computer algebra systems in the classroom make "curve discussions" obsolete. And in most cases they certainly are right. But...

Here, we investigate the function

$$r(x) = \frac{1000\,(x-1)}{(101x-100)(100x-99)}\ , \tag{1.8}$$

and we would like to be informed about its qualitative behavior. Let's check what DERIVE can show us about the graph of this function!

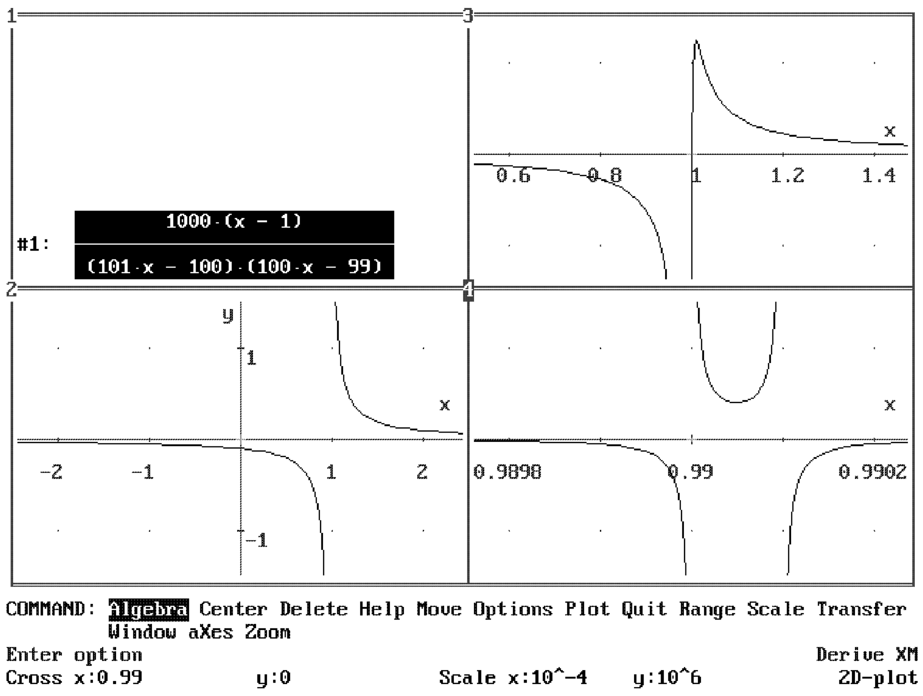

1
2
3
4
#1:
1000·(x - 1)
(101·x - 100)·(100·x - 99)
y
1
-1
x
-2
-1
1
2
0.6
0.8
1
1.2
1.4
0.9898
0.99
0.9902
COMMAND: Algebra Center Delete Help Move Options Plot Quit Range Scale Transfer
Window aXes Zoom
Enter option
Derive XM
Cross x:0.99 y:0 Scale x:10^-4 y:10^6
2D-plot

Figure 1.4 The graph of $r(x)$

On Figure 1.4, bottom left, we see what DERIVE's default answer is: The function looks like the usual hyperbola $y = 1/x$, moved to the right by one unit. But this is not correct! By a glimpse on (1.8), we observe that $r(x)$ has a zero at $x = 1$, and two poles. Hence, let's have a more careful look, and zoom in (with `<F9>`). We get Figure 1.4, top right. This shows that indeed $r(x)$ has a zero at $x = 1$, and gives us an idea about the pole. But weren't there two poles? Where is the second pole?

Next, we study the function to find out what is going on: A "curve discussion" is necessary to get the right understanding. We collect what we know:

The function $r(x)$ obviously has one zero at $x_0 = 1$ and two poles at the points

$$x_1 = \frac{99}{100} = 0.99 \qquad \text{and} \qquad x_2 = \frac{100}{101} = 0.\overline{9900}\ .$$

We calculate the zeros of the derivative

$$x_3 = 1 - \frac{\sqrt{101}}{1010} = 0.99004\,96281... \qquad \text{and} \qquad x_4 = 1 + \frac{\sqrt{101}}{1010} = 1.009950\,372...$$

with DERIVE. There is a local maximum occurring at x_4 that we see in Figure 1.4, top right.

On the other hand, a local minimum occurs between the poles, having the value

$$r(x_3) = 201\,000 + 20\,000\sqrt{101} = 401\,997.5124...\ .$$

Now we see what is going on! The values of $r(x)$ in the interval $[x_1, x_2]$ between the two poles are all larger than 400 000. Without this knowledge we cannot find the graph! An appropriate scaling yields Figure 1.4, bottom right.

MATHEMATICS BY EXPLORATION

How many terms n of the series

$$\sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{k}$$

have to be summed to approximate its limit $\ln 2$ with an error less than 10^{-k}:

$$\left| \sum_{k=1}^{n} \frac{(-1)^{k+1}}{k} - \ln 2 \right| < 10^{-k}\ ? \tag{1.9}$$

One can use DERIVE to find that

$$n(10^{-2}) = 50\ ,$$
$$n(10^{-3}) = 500\ ,$$
$$n(10^{-4}) = 5000\ .$$

This can either be done by trial and error (choose a value of n, and check whether (1.9) is true or not), or by an iteration using the `ITERATES` command. Since the latter is a little tricky, I don't go into the details. On the other hand, I would like to mention that one has to take into account that the summation is ill-conditioned (subtraction cancellation!). Combining successive positive and negative terms makes it well-conditioned. Hence, to prove that $n(10^{-4}) = 5000$ one can approximate the DERIVE statements `LN(2)-SUM(1/(2*k*(2*k-1)),k,1,2500)` and `LN(2)-SUM(1/(2*k*(2*k-1)),k,1,2500)-1/5000`.

The above series of results enables every student to ask the following *research question:* Is it true that

$$n(10^{-k}) = \frac{10^k}{2} \; ?$$

Believe it or not, the conclusion is: yes! This is not an elementary result, though. For the sake of completeness, I give the argument.

The *Euler Summation Formula*, applied to the *harmonic series*

$$H_n := \sum_{k=1}^{n} \frac{1}{k} \; ,$$

yields the asymptotics (s. [4], Equation (6.66))

$$H_n = \ln n + \gamma + \frac{1}{2n} - \frac{1}{12n^2} + \frac{1}{120n^4} - \dots \; ,$$

γ denoting the *Euler-Mascheroni constant*

$$\gamma = \lim_{n\to\infty} \Big(H_n - \ln n \Big) \; .$$

From

$$\begin{aligned} \sum_{k=1}^{2n} \frac{(-1)^{k+1}}{k} &= 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} \pm \dots - \frac{1}{2n} \\ &= 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \dots + \frac{1}{2n} - 2\left(\frac{1}{2} + \frac{1}{4} + \frac{1}{6} + \dots + \frac{1}{2n}\right) \\ &= H_{2n} - H_n \end{aligned}$$

one gets

$$\sum_{k=1}^{2n} \frac{(-1)^{k+1}}{k} = \ln 2 - \frac{1}{4n} + \frac{1}{16n^2} - \dots \; ,$$

proving the assertion.

LINEAR EQUATIONS

Whereas solving systems of linear equations with *exact rational arithmetic* (e.g. using DERIVE) by Gauß type elimination procedures in principle is trivial (nevertheless our students should study these techniques hard until they use

DERIVE for the same purpose!), the *numerical solution* is sometimes rather difficult. Instead of introducing certain pivoting techniques, I would rather like to emphasize the understanding *under which conditions* and *why* these numerical problems occur. To understand the underlying mechanisms better, one can use DERIVE to study them. It will turn out that the exact rational arithmetic can be of great help in this connection.

As a first example [9], we consider a simple system of two equations with two variables

$$780\,x + 563\,y = 217 \tag{1.10}$$

$$913\,x + 659\,y = 254\ . \tag{1.11}$$

This corresponds to the matrix equation

$$A \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} 217 \\ 254 \end{pmatrix}$$

with

$$A = \begin{pmatrix} 780 & 563 \\ & \\ 913 & 659 \end{pmatrix}$$

The system is almost singular, hence the problem is ill-conditioned. This corresponds to the fact that the two lines representing (1.10) and (1.11) are almost collinear: For our eyes, they definitely *are* collinear, see Figure 1.5, bottom right! The system (1.10)–(1.11) has the solution

$$x = 1 \qquad \text{and} \qquad y = -1$$

whereas the slightly modified system (we change only one coefficient!)

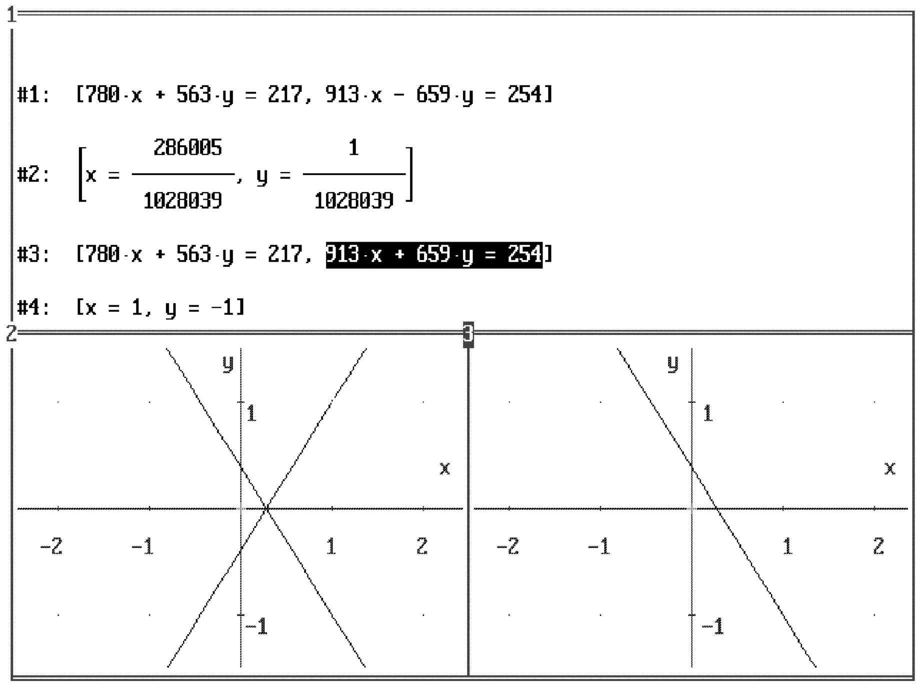
$$\begin{array}{rcrcr} 781\,x & + & 563\,y & = & 217 \\ 913\,x & + & 659\,y & = & 254 \end{array}$$

has the solution

$$\overline{x} = \frac{1}{660} \qquad \text{and} \qquad \overline{y} = \frac{23}{60}\ .$$

The two different outputs (x, y) and $(\overline{x}, \overline{y})$ have nothing to do with each other. They have a distance

$$\Big|(x, y) - (\overline{x}, \overline{y})\Big| = \frac{\sqrt{12\,541}}{55} \approx 2.03612...\ .$$



```
#1:  [780·x + 563·y = 217, 913·x - 659·y = 254]

         286005          1
#2:  [x = ———————, y = ———————]
         1028039       1028039

#3:  [780·x + 563·y = 217, 913·x + 659·y = 254]

#4:  [x = 1, y = -1]
```

Figure 1.5 Geometric representation of linear equations

Hence a relative modification of about a thousandth in the coefficients of A that we consider as the input data, yields a change in the output (x, y), i.e. the solution of the corresponding equations system, of about one: Such input faults are *amplified* by a factor of 1 000, hence the condition of A is bad! The worst case amplification factor gives the term *condition* a precise meaning.

If we solve the system (1.10)–(1.11) numerically with DERIVE with the default number of 6 digits, we still get $x = 1$, $y = -1$. Using a working precision of only four digits yields however

$$[x = @1,\ y = 0.001517\,(254 - 913\,@1)]\ .$$

Here @1 denotes an arbitrary parameter. Hence with a precision of four digits the two lines are *identical*, and all points of this line are solutions of (1.10)–(1.11)!

Note that on the other hand, the similar system

$$\begin{array}{rcrcl} 780\,x & + & 563\,y & = & 217 \\ 913\,x & - & 659\,y & = & 254 \end{array}$$

has quite different behavior, see Figure 1.5, bottom left, although only one sign is different. The new system has the solution

$$x = \frac{286\ 005}{1\ 028\ 039} \qquad \text{and} \qquad y = \frac{1}{1\ 028\ 039} \ ,$$

and the modified system (changing again one coefficient slightly)

$$\begin{array}{rcrcl} 781\,x & + & 563\,y & = & 217 \\ 913\,x & - & 659\,y & = & 254 \end{array}$$

leads to

$$\overline{x} = \frac{12\ 435}{44\ 726} \qquad \text{and} \qquad \overline{y} = -\frac{1}{4\ 066}$$

with

$$\Big|(x,y) - (\overline{x},\overline{y})\Big| = \frac{62\ 175\ \sqrt{50\ 714}}{45\ 980\ 072\ 314} \approx 3.04515 \cdot 10^{-4} \ ,$$

having the same order of magnitude as the input fault: This linear system is not ill-conditioned.

How can we measure the condition of a linear system corresponding to a matrix A? It turns out that the condition is high (bad) if either the entries of A or those of A^{-1} are of high magnitude. Obviously, if A is singular (A^{-1} does not exist) then the condition is arbitrarily bad, $\text{cond}(A) = \infty$, say. If

$$\|A\| = \|a_{jk}\| := \max_{jk} |a_{jk}|$$

is the *norm* of A, given by the maximum modulus of its entries, then for non-singular A one defines the condition by the product

$$\text{cond}(A) := \|A\| \cdot \|A^{-1}\| \ .$$

It turns out that this *matrix condition* measures the worst case amplification factor of input faults. It is invariant under the multiplication of rows by a constant, corresponding to the fact that such a modified equations system has the same solution.

The matrix norm and condition can be calculated by the DERIVE functions

```
NORM(A):=MAX(MAX(VECTOR(VECTOR(ABS(A SUB j_ SUB k_),
                         j_,1,DIMENSION(A)),
                  k_,1,DIMENSION(A))))

COND(A):=IF(DET(A)=0,inf,NORM(A)*NORM(A^(-1)))
```

With these functions, we get

$$\operatorname{cond}\begin{pmatrix} 780 & 563 \\ 913 & 659 \end{pmatrix} = 833\,569$$

and

$$\operatorname{cond}\begin{pmatrix} 780 & 563 \\ 913 & -659 \end{pmatrix} = \frac{833\,569}{1\,028\,039} = 0.810834\ .$$

The large condition number of the first matrix tells us everything about the bad condition of this mapping.

Now we are on safe grounds to have a detailed look at the *Hilbert Matrices*

$$\mathcal{H}_n := \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+1} \\ \vdots & \vdots & \vdots & \frac{1}{j+k-1} & \vdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n-1} \end{pmatrix} .$$

They can be defined by the DERIVE function

```
HILBERT(n):=VECTOR(VECTOR(1/(j_+k_-1),j_,1,n),k_,1,n)
```

They give well-known examples of ill-conditioned matrices. The larger we choose n, the worse is the condition of $\mathcal{H}_n$. Let's check this with DERIVE. Using $\boxed{\texttt{approX}}$, we approximate the call `VECTOR(COND(HILBERT(n)),n,1,20)`. This yields

$$\big[1, 12, 192, 6480, 1.792\cdot10^{5}, 4.41\cdot10^{6}, 1.33402\cdot10^{8}, 4.25180\cdot10^{9}, 1.27470\cdot10^{10}$$
$$6.19417\cdot10^{10}, 6.58486\cdot10^{10}, 7.34160\cdot10^{10}, 2.22089\cdot10^{11}, 5.23931\cdot10^{11}, 1.20211\cdot10^{10},$$
$$2.80244\cdot10^{10}, 2.43921\cdot10^{10}, 1.95063\cdot10^{10}, 2.40912\cdot10^{10}, 2.54206\cdot10^{10}\big]\ .$$

The result looks as if the condition is never worse than about 10^{11}. This is neither plausible nor true. But what the hell... Oh, yeah, the calculation of the condition needs the calculation of the inverse matrix, and the calculation of the inverse is ill-conditioned and produces nonsense results!

An exact calculation shows e.g. that

$$\operatorname{cond}\mathcal{H}_{20} = 3\,613\,560\,329\,006\,048\,768\,624\,640\,000 = 3.61356 \cdot 10^{27} .$$

Here DERIVE's rationally exact arithmetic is very helpful. If we $\boxed{\texttt{Simplify}}$ the call `VECTOR(COND(HILBERT(n)),n,1,20)` (generating rationally exact output), and use $\boxed{\texttt{approX}}$ afterwards, then we receive the correct condition numbers

$$\big[1, 12, 192, 6480, 179200, 4.41000 \cdot 10^{6}, 1.33402 \cdot 10^{8}, 4.24994 \cdot 10^{9}, 1.22367 \cdot 10^{11}, 3.48067 \cdot 10^{12}, 1.17643 \cdot 10^{14}, 3.65944 \cdot 10^{15}, 1.06518 \cdot 10^{17}, 3.52176 \cdot 10^{18}, 1.14708 \cdot 10^{20}, 3.52527 \cdot 10^{21}, 1.10552 \cdot 10^{23}, 3.71252 \cdot 10^{24}, 1.18439 \cdot 10^{26}, 3.61356 \cdot 10^{27}\big] .$$

Just for fun, the following is the output of the matrix product `HILBERT(10) . HILBERT(10)^(-1)`:[1]

$$\begin{pmatrix}
0.995822 & -0.0111111 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 2.6 & 1 & 1 & -1 & -1 & 2 & 0 & -2 & 1 \\
0.00201612 & 1.33333 & 3 & 3 & -2 & -1 & 1 & 0 & -3 & -2 \\
-0.00322061 & 0.128205 & 2 & 3 & -4 & 0 & 1 & 0 & -2 & -1 \\
0.00208986 & 0.412087 & 2 & 3 & 0 & 0 & 1 & -2 & -3 & -2 \\
-0.00133689 & -0.647619 & 2 & 1 & -2 & 1 & 3 & -2 & 0 & -1 \\
-0.00497347 & 0.533333 & 0 & 3 & -1 & 1 & 2 & 0 & -1 & -2 \\
0 & 0.691176 & 1 & 3 & -2 & 1 & 0 & 0 & -1 & -2 \\
-2.60264 \cdot 10^{-4} & 1.24369 & 0 & 0 & -3 & 3 & -3 & 1 & -1 & -1 \\
-0.00121580 & 2.12865 & 1 & 1 & -1 & 2 & -2 & -3 & -1 & 0
\end{pmatrix}$$

calculated by 6-digit precision! This should be the unit matrix!

COMPUTATION OF CHEBYSHEV POLYNOMIALS

Loading `ORTH_POL.MTH`, one can calculate classical orthogonal polynomials. But strange things happen. . .[2] Let's assume we would like to know $T_{100}(1/4)$, the value of the hundredth Chebyshev polynomial of the first kind at the point $x = 1/4$. Let's do it! Simplifying `CHEBYCHEV_T(100,1/4)` yields

$$T_{100}(1/4) = \frac{2\,512\,136\,227\,142\,750\,476\,878\,317\,151\,377}{2\,535\,301\,200\,456\,458\,802\,993\,406\,410\,752} .$$

This, indeed, is the correct value of $T_{100}(1/4)$. On the other hand, if we $\boxed{\texttt{approX}}$ the expression `CHEBYCHEV_T(100,1/4)`, we get the stupid result

$$T_{100}(1/4) \approx 38.8363\ !$$

Looking at $T_{100}(x)$ (it is an alternating series!) gives us the clue that, again, subtraction cancellation occurs. What a pity!

Here is a method to resolve this question. Using the identities ([1], (22.7.24))

$$T_{2n}(x) = 2\,T_n(x)^2 - 1$$

for even n, and

$$T_{2n-1}(x) = 2\,T_n(x)\,T_{n-1}(x) - x$$

for odd n gives an efficient method to compute $T_n(x)$ which moreover is numerically stable! Hence (for numerical purposes) we may define

```
CHEBYSHEVT(n,x):=IF(n=0,1,IF(n=1,x,
            IF(FLOOR(n/2)=n/2,2*CHEBYSHEVT(n/2,x)^2-1,
            2*CHEBYSHEVT((n-1)/2,x)*CHEBYSHEVT((n+1)/2,x)-x)))
```

If we $\boxed{\texttt{approX}}$ the expression `CHEBYSHEVT(100,1/4)` we get immediately $T_{100}(1/4) \approx 0.990863$. There is no problem to approximate $T_n(x_0)$ for much larger $n \approx 10^6$ using this code.

The underlying divide-and-conquer approach ($T_n(x)$ is calculated with the aid of predecessors $T_m(x)$ whose index m has only half the size of n) is not the fastest method to calculate the expanded polynomial $T_n(x)$ though. For this purpose, in the DERIVE file `ORTH_POL.MTH`, the definition

$$T_n(x) = \frac{n}{2} \sum_{k=0}^{\lfloor n/2 \rfloor} \frac{(-1)^k\,(n-k-1)!}{k!\,(n-2k)!}\,(2x)^{n-2k} = \sum_{k=0}^{\lfloor n/2 \rfloor} a_k$$

is given. Due to DERIVE's fast calculation of the occurring factorial terms this is not a bad choice. But one can do better avoiding the calculation of factorials!

Since one has the rational term ratio

$$\frac{a_k}{a_{k-1}} = -\frac{(n-2\,k+2)\,(n-2\,k+1)}{4\,k\,x^2\,(n-k)} \tag{1.12}$$

and the initial value

$$a_0 = 2^{n-1}\,x^n \tag{1.13}$$

one can implement the Chebyshev polynomials (for symbolic purposes) by

```
CHEBYSHEVT_LIST(n,x):=ITERATES([a_ SUB 1+1,
   -(n-2*a_ SUB 1+2)*(n-2*a_ SUB 1+1)/
   (4*a_ SUB 1*x^2*(n-a_ SUB 1))*a_ SUB 2],
   a_,[1,2^(n-1)*x^n],FLOOR(n/2))
```

```
CHEBYSHEVT(n,x):=SUM((CHEBYSHEVT_LIST(n,x)) SUB k_ SUB 2,
                     k_,1,FLOOR(n/2)+1)
```

which is an almost direct translation of (1.12)–(1.13). Here `a_` is an abbreviation for the vector (k, a_k), hence `a_ SUB 1` denotes k, and `a_ SUB 2` denotes a_k.

Do not use this implementation for numerical purposes since it has the same cancellation defect as above! But with this code one can calculate $T_{10000}(x)$, e.g., in reasonable time.[3] DERIVE is as fast (or even faster) as the other systems Axiom, Macsyma, Maple, Mathematica, MuPAD or REDUCE [7]!

AUTOMATIC DIFFERENTIATION

As a final topic I would like to introduce automatic differentiation. A first simple example is given by the function

$$F(x) := \begin{cases} \sin x & \text{if } x > 0 \\ \arctan x & \text{otherwise} \end{cases} ,$$

and the problem is to differentiate $F(x)$. DERIVE's builtin solution is to define $F(x)$ in terms of characteristic functions

```
F(x):=CHI(0,x,infinity)*SIN(x)+CHI(-infinity,x,0)*ATAN(x)
```

Then differentiation of $F(x)$ is possible and yields a representation in terms of the sign function. The representation used

$$f := \chi_{(0,\infty)} \sin x + \chi_{(-\infty,0]} \arctan x$$

is not really a functional one but is one in terms of *expressions*, namely as a sum of two products of certain subexpressions. Note that we normally represent functions by expressions and do then differentiation using the differentiation rules, in our case

$$f' = \chi'_{(0,\infty)} \sin x + \chi_{(0,\infty)} \cos x + \chi'_{(-\infty,0]} \arctan x + \chi_{(-\infty,0]} \frac{1}{1+x^2} .$$

DERIVE knows how to differentiate the characteristic function, and hence finds f'. On the other hand, we can interpret $F(x)$ as a *program*, given with the aid of the `IF` programming construct:

```
F(x):=IF(x>0,SIN(x),ATAN(x))
```

DERIVE does *not* differentiate such a program. *Automatic differentiation* is a method to differentiate programs rather than functions. The result is then another program which calculates the derivative. We will see that in a certain way this is more direct than the expression approach. Obviously it is more general since all expressions can be understood as (rather short) programs. The main technique is to differentiate the occurring local variables. This works for many programming constructs including loops etc., but since DERIVE's programming capabilities are rather limited, we will only consider the `IF` construct. In our example case, local variables are implicitly given, `a:=SIN(x)`, `b:=ATAN(x)`. Differentiating these, we get e.g.

$$F'(x) = \begin{cases} \frac{d}{dx}\sin x & \text{if } x > 0 \\ \\ \frac{d}{dx}\arctan x & \text{otherwise} \end{cases} . \tag{1.14}$$

To underline the fact that the result is a program that calculates the derivative function, we give the derivative function a separate name, and call it `FP(x)`. By (1.14) we can define this derivative function by the definition

```
FP(x):=IF(x>0,DIF(SIN(x),x),DIF(ATAN(x),x))
```

Applying this technique iteratively, we can obtain representations for all higher derivatives of $F(x)$.

The advantage of automatic differentiation is its general applicability. In addition, the method can reduce computing time and memory requirements significantly.

To prove the usefulness of this approach, I give a more complicated recursive example. Let

$$G(x,n) := \begin{cases} 0 & n = 0 \\ x & n = 1 \\ G(x,n-1) + \sin(G(x,n-2)) & n > 1 \end{cases} .$$

Figure 1.6 shows the graphs of $G(x,n)$ for $n = 0, \ldots, 10$. We would like to find the values $G'(0,n)$ for arbitrary $n \in \mathbb{N}_0$, and $G'''(0,100)$, e.g. Try to calculate these!

We define

```
G(x,n):=IF(n=0,0,IF(n=1,x,G(x,n-1)+SIN(G(x,n-2))))
```

Using the classical approach, we cannot calculate `DIF(G(x,n),x)`, but we can do the same for specific n. This yields very complicated expressions. Calculating `VECTOR(DIF(G(x,n),x),n,0,10)`, and substituting $x = 0$ into the result, we receive

$$[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55] .$$

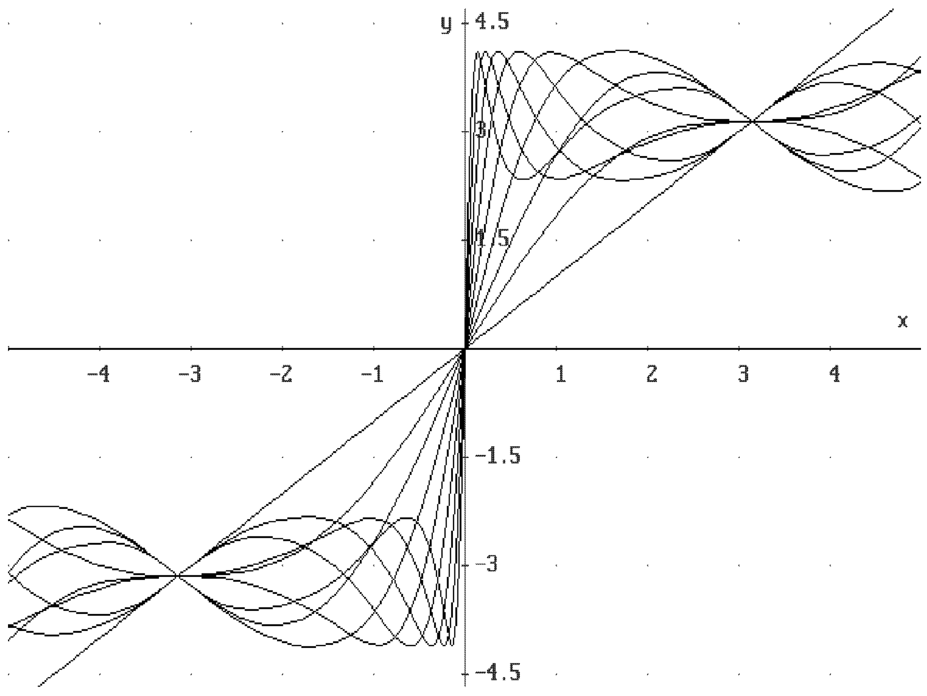


Figure 1.6 The functions $G(x, n)$ for $n = 0, \ldots, 10$

Could it be that these numbers are the Fibonacci numbers F_n, given by the recurrence equation

$$F_n = F_{n-1} + F_{n-2} \ , \quad F_0 = 0 \ , F_1 = 1 \ ? \tag{1.15}$$

It seems to be so but how can we prove it?

Let's continue with automatic differentiation. We can define the derivative `GP(x,n)` and the second and third derivatives `GPP(x,n)` and `GPPP(x,n)` of $G(x, n)$, respectively, by

```
GP(x,n):=IF(n=0,0,IF(n=1,1,GP(x,n-1)+GP(x,n-2)*COS(G(x,n-2))))

GPP(x,n):=IF(n=0,0,IF(n=1,0,GPP(x,n-1)
             +GPP(x,n-2)*COS(G(x,n-2))-SIN(G(x,n-2))*GP(x,n-2)^2))

GPPP(x,n):=IF(n=0,0,IF(n=1,0,GPPP(x,n-1)
     +GPPP(x,n-2)*COS(G(x,n-2))-GPP(x,n-2)*SIN(G(x,n-2))*GP(x,n-2)
```

```
    -COS(G(x,n-2))*GP(x,n-2)^3
    -2*SIN(G(x,n-2))*GP(x,n-2)*GPP(x,n-2)))
```

Here we applied iteratively the chain and product rules. I ask the reader to check these results! Note that now we have *programs* to calculate $G'(x,n)$, $G''(x,n)$, and $G'''(x,n)$. In particular, for $x = 0$ the program to calculate $G'(x,n)$ reads as

$$G'(0,n) := \begin{cases} 0 & n = 0 \\ 1 & n = 1 \\ G'(0,n-1) + G'(0,n-2)\,\cos(G(0,n-2)) & n > 1 \end{cases} .$$

Since by definition $G(0,n) = 0$, this is exactly the definition (1.15) of the Fibonacci numbers! Hence our first question is completely solved by this approach.

The above program cannot solve our second question to find $G'''(0,100)$, though. The reason is that the program `GPPP(x,n)` is recursive, which calls itself twice, moreover calling `G(x,n)`, `GP(x,n)` as well as `GPP(x,n)`. Those programs are highly inefficient, see e.g. the DERIVE User Manual [2], § 10.6. Other systems like Maple have a `remember` option for this purpose that store computed results in the memory. Such an option would help a lot in this situation.

Calling `GPPP(0,13)`, we can calculate $G'''(0,13) = -995424$ in 70 seconds on an INTEL 486 CPU, 100 MHz. On the other hand, if one rewrites all recursive computations by iterative ones, then one gets in a second

$$G'''(0,100) =$$

$$-3\,497\,461\,101\,660\,385\,688\,954\,109\,997\,243\,219\,765\,648\,878\,232\,202\,900\,028\,526\,850.$$

Note that there is no chance at all to find this value by the calculation of the hundredth derivative of $G(x,n)$, substituting $x = 0$!

CONCLUSION AND FUTURE DIRECTIONS OF DERIVE

In the Abstract, I announced to give examples how numeric and symbolic computations need each other. Some of the examples given were about symbolic identities, hence mathematical theorems (e.g. Hofstadter's Theorem) that needed numerical (and graphical) evidence to be discovered, but then could be proved by sophisticated symbolic computations. Other examples showed ill-conditioned numerical computations which could be replaced by well-conditioned ones using symbolic techniques. Finally the "curve discussion" example showed that graphical techniques—despite of their importance and chances—are not always capable to show the qualitative behavior of the functions we are dealing with directly. Sometimes the combination with symbolic computations cannot be avoided.

It is the combination of numeric and symbolic capabilities, enriched by graphical ones, that gives a program like DERIVE an essential advantage over a calculator (or graphics calculator).

Whereas for most problems occurring in school education DERIVE does a gorgeous job, there are some well-known algorithms that in my opinion are missing in DERIVE and should be integrated in the future.

- **Polynomial zeros** Questions like: Where are the zeros of the *Wilkinson polynomial*

  $$\begin{aligned} P(x) &:= (x-1)(x-2)\cdots(x-20) - \frac{x^{19}}{10^7} \\ &= x^{20} - 219.00000\,01\,x^{19} + 20615\,x^{18} + \ldots \end{aligned}$$

  cannot be solved by DERIVE although there are efficient methods implemented and available in other computer algebra systems. DERIVE can be utilized to give graphical evidence to the interesting fact that $P(x)$ has only 14 real zeros, and no real zero in the interval $[10, 20]$. Note that one should use the left hand representation to plot the graph since the expanded form again is ill-conditioned!

- **Solution of Polynomial Systems** Even a simple system like

  $$x\,y - 8 = 0 \qquad \text{and} \qquad x^2 - 5\,x + y + 2 = 0$$

  corresponding to the intersection of a hyperbola and a parabola, cannot be solved by DERIVE's `SOLVE` command since it is *nonlinear*. Mighty methods based on the calculation of Gröbner bases are available in other computer algebra systems that are capable to find *all solutions* of a polynomial system.

  On the other hand, not all other systems find the solutions for questions like: For which of the parameters a, b and m are the following true:

  $$\frac{2\sqrt{b}}{\sqrt{a}+\sqrt{b}} + \frac{\frac{\sqrt{a}^3+\sqrt{b}^3}{\sqrt{a}+\sqrt{b}} - \sqrt{a}\,\sqrt{b}}{a+b} = 1\ ,$$

  and

  $$\frac{\sqrt{1+m}}{\sqrt{1+m}-\sqrt{1-m}} + \frac{1-m}{m-1+\sqrt{1+m}\,\sqrt{1-m}} = \frac{m}{1-\sqrt{1+m}\,\sqrt{1-m}}\ ?$$

  DERIVE is successful!

Notes

1. If you use $\boxed{\texttt{approX}}$ for the factors first, then the result is even worse. Why?

2. The results of the `ORTH_POL.MTH` package can be reproduced in older versions (up to 3. . .) of DERIVE. In the newer releases the code of this section is already adopted.

3. To avoid the generation of the immensely complicated output on the screen—which takes much longer than the computation itself—use the dummy function `DUMMY(term):="done"`.

References

[1] Abramowitz, M. and Stegun, I.A.: *Handbook of Mathematical Functions.* Dover Publ., New York, 1964.

[2] DERIVE User Manual, Version 3. Seventh Edition, 1994.

[3] Engel, A.: Geometrische Beweise mit dem PC. In: Tagungsband der *DERIVE Days Düsseldorf*, 19.–21. April 1995. Ed. by Bärbel Barzel, Landesmedienzentrum Rheinland-Pfalz, Düsseldorf, 1995, 27–36.

[4] Graham, R.L., Knuth, D.E. and Patashnik, O.: *Concrete Mathematics. A Foundation for Computer Science.* Addison-Wesley, Reading, Massachussets. Second Edition, 1994.

[5] Koepf, W., Ben-Israel, A., Gilbert, R.P.: *Mathematik mit DERIVE.* Vieweg, Braunschweig/Wiesbaden, 1993.

[6] Koepf, W.: *Höhere Analysis mit DERIVE*, Vieweg, Braunschweig/Wiesbaden, 1994.

[7] Koepf, W.: Efficient computation of orthogonal polynomials in computer algebra. Konrad-Zuse-Zentrum Berlin (ZIB), Preprint SC 95-42; updated version available at `http://www.zib.de/koepf`.

[8] Koepf, W.: *DERIVE für den Mathematikunterricht.* Vieweg, Braunschweig/Wiesbaden, 1996.

[9] Neundorf, W.: Kondition eines Problems und angepaßte Lösungsmethoden. Lecture given at the DMV-Tagung, Ulm, September 1995.