# THE DERIVE - NEWSLETTER #38

**THE BULLETIN OF THE**

DERIVE®

**USER GROUP** **+ TI 92**

**Contents:**

**Revised version 2017** **June 2000**

[1] **Visual Mathematics, Illustrated by the TI-92 and the TI-89,**
G.C. Dorner, J.M. Ferrard, H. Lemberg, 440 pages, Springer 2000, ISBN 2-287-59685-2.
Contents: Discrete Dynamical Systems, Differential Equations, Fourier Analysis, Interpolation and approximation, Orthogonality, Eigenvalues and eigenvectors, Calculator guide, List of programs (84). No diskette included (but the programs are not very extended).

[2] **Programmieren mit dem TI-92,** Beispielorientierte Einführung in das Programmieren,
Peter Witthinrich, bk-teachware SR-16, ISBN 3-901769-25-0.
Inhalt: Einfache Anweisungen, Schleifen und Fallentscheidungen, Unterprogramme und Funktionen, Grafiken und Tabellen.

[3] **Wachstums- und Abnahmeprozesse mit dem TI-92**, Ein Lehrgang zur Behandlung von Exponential- und Logarithmusfunktion, E Prugger, C Rauniak, E Schneider,
bk teachware SR-17, ISBN 3-901769-26-9
Ein im Unterricht praktisch erprobtes Lehr- und Arbeitsbuch, das auch zum Selbststudium geeignet ist. Viele Aufgaben zur Vertiefung und Wiederholung des Lehrstoffs machen dieses Büchlein zu einer wertvollen Hilfe für Lehrer und Schüler.

## Interesting WEB sites http://......

| | |
|---|---|
| her.nw.schule.de/SAN/WS6/althilf1.htm | Alltagshilfen in Mathematik |
| did.mat.uni-bayreuth.de/ab/jgstf.htm | Aufgaben nach Jahrgangsstufen |
| www.muenster.de/~stauff/bewmath.html | Bewegte Mathematik |
| www.stud.uni-bayreuth.de/~a3957/pythagoras/wahl.html | Rule of Pythagoras |
| sun2.mathematik.uni-freiburg.de/home/didaktik/ | Didaktik an der Uni Freiburg |
| www.suchfibel.de/index.htm | Die Suchfibel |
| www.dynageo.de/ | Euklid Homepage |
| www.bildung.hessen.de/fbereich/mathe/index.htm | Fächer Mathematik |
| kunden.swhamm.de/Geometriepage/links.htm | Geometry Page |
| did.mat.uni-bayreuth.de/geonet/index.html | Geonet |
| did.mat.uni-bayreuth.de/geonet/beispiele/beispiele.html | Geonet-Beispielseiten |
| www.mathe2.uni-bayreuth.de/~evi/math.html | Go to Mathematical Links |
| www.dpl.net/leimeier/weiterefaecher.html#Mathematik | |
| home.t-online.de/home/steidel/mathe.htm | Interesting Links to Mathematical Topics |
| www.tmg.musin.de/mathematik.htm | Internet Resources |
| did.mat.uni-bayreuth.de/ab/intro.htm | ISB Aufgabensammlung |
| www.learn-line.nrw.de/Faecher/Mathematik/Geometrie/medfoy/java.htm | JAVA-Geometry |
| www.gamelan.com/directories/pages/dir.java.educational.math.html | JAVA-Programming Resources |

Liebe DUG Mitglieder,

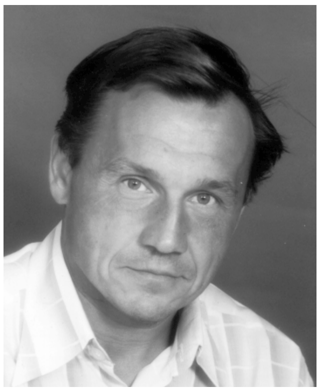
In diesem DNL finden Sie vor allem zwei Schwerpunkte gesetzt: natürlich wird in großem Maß auf das neue DERIVE 5 hingewiesen. Dabei können aber nicht alle neuen Möglichkeiten ausreichend besprochen werden. So habe ich mir gedacht, dass wir vorerst einmal auf die neuen Programmiermöglichkeiten in *DERIVE* eingehen sollten. Betrachten Sie daher die Artikel von Josef Lechner und Johann Wiesenbauer besonders unter diesem Aspekt. Johann - und sicherlich nicht nur er - wäre an "programmierten" *DERIVE*-An-wendungen besonders interessiert. Ein Terence-Etchells-Programm habe ich noch für den nächsten DNL auf Lager, aber es wäre schön, wenn auch andere `prog`, `loop`, `return`, .....-Anwender ihre Geheimnisse mit uns teilen würden.

Da wir schon beim Programmieren sind, sollte ein einfaches TI-92 Programm auch nicht fehlen. Aus Erfahrung weiß ich, dass viele eifrige und überzeugte TI-89/92 Benützer noch immer eine Scheu vor dem Programmieren haben. Thomas Himmelbauers Programm soll ein bißchen Geschmack machen.

Ich möchte mich noch besonders für Wolfram Koepfs Erlaubnis bedanken, seinen schönen Vortrag im DNL veröffentlichen zu dürfen.

Ich wünsche Ihnen allen einen schönen Sommer und freue mich auf ein Wiedersehen mit vielen von Euch bei den verschiedensten Gelegenheiten.

Josef Böhm



Dear DUG-Members,

In this DNL you will find covered mainly two fields of interest: it is obvious that we will deal with new *DERIVE* 5. Unfortunately we can not discuss all its new features, so I thought that it might make sense to write about the new programming possibilities. Please read Josef Lechner´s "Turtle" and Johann Wiesenbauer´s contribution under this special point of view. Johann - and he will not be the only one - is very interested in "programmed" *DERIVE*-applications. I have one Terence-Etchells-program preserved for the next DNL - from times, when it was not allowed to publish a DfW5-file. It would be nice if other `prog`, `loop`, `return`, ....-users would share their secrets with us.

As we are now dealing with programming, I wanted to add a not too difficult TI-92 program. I made the experience that even convinced TI-89/92 users very often are reluctant to use the programming capabilities of their machines and so don´t make use of one of the most important features. Thomas Himmelbauer´s program should make you taste blood.

I want to thank Wolfram Koepf for his permission to print his inspiring lecture in this DNL.

I wish you all a wonderful summer and hope to meet many of you at various occasions.

Josef Böhm

**Find all the DERIVE and TI-files on the following web sites**

http://www.acdca.ac.at/t3/dergroup/index.htm
http://www.bk-teachware.com/main.asp?session=375059

**now only at: http://www.austromath.at/dug/**

The *DERIVE-NEWSLETTER* is the Bulletin of the *DERIVE & TI-92 User Group*. It is published at least four times a year with a contents of 44 pages minimum. The goals of the *DNL* are to enable the exchange of experiences made with *DERIVE* and the *TI-92/89* as well as to create a group to discuss the possibilities of new methodical and didactical manners in teaching mathematics.

As many of the *DERIVE* Users are also using the *TI-92/89* the *DNL* tries to combine the applications of these modern technologies.

**Contributions:**
Please send all contributions to the Editor. Non-English speakers are encouraged to write their contributions in English to reinforce the international touch of the *DNL*. It must be said, though, that non-English articles will be warmly welcomed nonetheless. Your contributions will be edited but not assessed. By submitting articles the author gives his consent for reprinting it in the *DNL*. The more contributions you will send, the more lively and richer in contents the *DERIVE & TI-92 Newsletter* will be.

Editor: Mag. Josef Böhm
A-3042 Würmla
D´Lust 1
Austria
Phone/FAX: 43-(0)660 31 36 365
e-mail: nojo.boehm@pgv.at

Next issue: September 2000
Deadline 15 July 2000

**Preview: Contributions for the next issues**

Inverse Functions, Simultaneous Equations, Speck, NZL
A Utility file for complex dynamic systems, Lechner, AUT
Examples for Statistics, Roeloffs, NL
Quaternion Algebra, Sirota, RUS
Various Training Programs for the TI
A critical comment on the "Delayed Assignation" :==, Kümmel, GER
Sand Dunes, River Meander and Elastica, The lighter Side ....., Halprin, AUS
Type checking, Finite continued fractions, Welke, GER
Kaprekar´s "Self numbers", Schorn, GER
Some simulations of Random Experiments, Böhm, AUT
Examples for Programming with DERIVE 5, Etchells, Lechner ao
Comparing statistics tools: a pie chart with DERIVE, a stem & leaf diagram on the TI,
Winding Numbers, Welke, GER

and
Setif, FRA; Vermeylen, BEL; Leinbach, USA; Aue, GER; Koller, AUT, ......

Impressum:
Medieninhaber: DERIVE User Group, A-3042 Würmla, D´Lust 1, AUSTRIA
Richtung: Fachzeitschrift
Herausgeber: Mag.Josef Böhm
Herstellung: Selbstverlag

**Claire Phillips, Newham, UK**

Hi, I am a new user of DERIVE. I work in an East London sixth form college and am trying to develop materials for use in A level maths classes (for students who are 16-19 years old). I will be at the DERIVE Conference in July with lots of questions.

But one I have right now is: Can DERIVE plot vector equations of planes or do I have to convert to the Cartesian equations?

(at the moment we have DERIVE on 30 day trial and so have no access to an instruction book, but we have ordered the real thing and it should be on its way).

***DNL:***

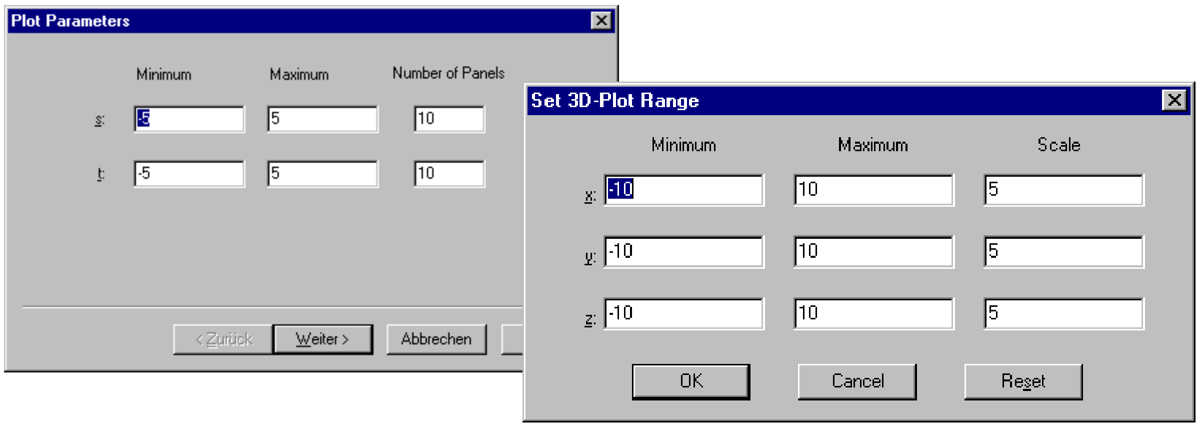
*Dear Claire,*

*I am looking forward to meeting you at the Conference and answer your questions. You can plot vector equations. Follow the instructions given below:*

(1) *Edit the equation in vector form:*

```
#1:   plane := [-3, 5, -2] + t·[-1, 2, 3] + s·[3, 4, -2]
```

(2) *Switch to the 3D-Plot window, Set Options Simplify Before Plotting On, Press F4 for Insert Plot*



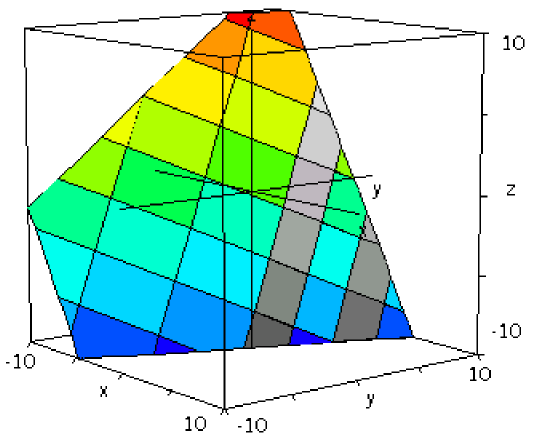
Plot Parameters

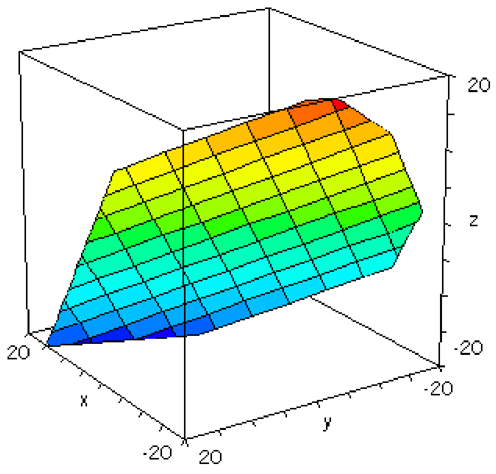
| | Minimum | Maximum | Number of Panels |
|---|---|---|---|
| s: | -5 | 5 | 10 |
| t: | -5 | 5 | 10 |

Set 3D-Plot Range

| | Minimum | Maximum | Scale |
|---|---|---|---|
| x: | -10 | 10 | 5 |
| y: | -10 | 10 | 5 |
| z: | -10 | 10 | 5 |

(3) *Set the Plot parameters for s and t according to your wishes and define the Plot Range. Then you will obtain a nice plot of your plane. See two different plots of the same plane (you can switch off the box!)*:

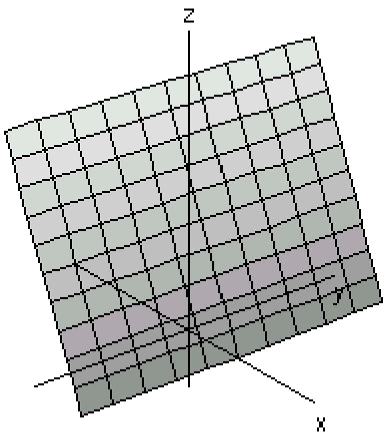




Another example plotted without the surrounding box:

$$\#2: \quad \text{plane2} := [-1, 0, 4] + t1\cdot[3, 0, -4] + t2\cdot[0, 1, 0]$$

$$\#3: \quad \text{plane2\_} := [-1, 0, 4] + \frac{t1\cdot[3, 0, -4]}{|[3, 0, -4]|} + t2\cdot[0, 1, 0]$$



**<u>Jesús Ramirez, Guadalajara, Mexico</u>**

Hi Derivers!!!!

I have a little problem but I can't find where the mistake is. I hope you can help me on this...

I'm working with a heating system closed-loop transfer function and I'm trying to find its Inverse Laplace Transform (I know Derive doesn't do that, but wait a little bit – now DERIVE 6 does, Josef). I'm writing down every step between the frequency domain and the time domain because this is part a of a Control Course homework that I have to do. The way I'm working is the following one:

- I make this substitution inside the transfer function --> Te(s) =10/s so I get this:

$$\frac{100 + \frac{10}{s}}{60\cdot s + 1.71\cdot\left(100 + \frac{10}{s}\right) + 1}$$

- I simplify the expression via Derive and I get this:

$$\frac{100\cdot(10\cdot s + 1)}{600\cdot s^2 + 1720\cdot s + 171}$$

- In order to find the Inverse Laplace Transform, first I have to find the roots on the denominator so I can name them by a symbol (let's say "a" & "b") and expand the transfer function by partial fractions ("expand" command on Derive) and finally look up at a Laplace Transforms Table (remember that this is a homework and, although I can use the computer, I must show the intermediate results)...

- Finally I use the Student Edition of MATLAB 4.0 (which has a symbolic toolbox with access to the Maple Kernel) in order to check what I got and I find the Inverse Laplace Transform with "invlaplace" (using the first expression, where I just made the substitution Te(s) = 10/s).

Well, the problem is that I came up with two completely different answers:

- Using the first expression (without simplification and all that stuff) I get the right answer (I know that's the right answer because I'm designing an automatic control closed-loop and that's the answer that less disturbs the right operation of the heating system), which is the following one:

```
T(t) = 1.6686 e^(-2.7635 t) - 0.0019 e^(-0.1031 t)
```

$$\mathrm{INVLaplace}\left(\frac{100\cdot(10\cdot s + 1)}{600\cdot s^{2} + 1720\cdot s + 171}\right) = 1.668\cdot e^{-2.763\cdot t} - 0.00196\cdot e^{-0.1031\cdot t}$$

- Using the expression with the factorized roots, that is

```
numerator = 100 + 10/s       IS THIS THE CORRECT NUMERATOR?
                                         IT DOESN'T FOLLOW.

denominator = (s - a)(s - b)

I get what I find to be a wrong answer, that is:

T(t) = 1001.17 e^(-2.7635) - 1.17 e^(-0.1031)
```

I found again the roots using MATLAB and there's no error in Derive regarding this aspect....

I'm enclosing a zip file containing the text file of the MATLAB outputs as well as a jpg image of the same computations made with Mathematica 4.0. Don't be afraid to download it, it doesn't contain virus...

I hope you can lend me a hand with this problem. Maybe I'm making a mistake over and over and the one that is in a "closed loop" is me!!

Thanks in advance!!!!!!!!!!!

Greetings from Guadalajara, Jalisco, Mexico!!!!!!!!!!!!!!

**<u>Boz Kempski, England</u>** B.Kempski@anglia.ac.uk

Jesus

Using DERIVE to expand (100(10s+1))/(600s^2+1720s+171) into partial fractions gives me (with some manipulation!)

```
1.66864/(s+2.76352)-0.00196(s+0.1031).
```

Taking inverse Laplace transforms yields the answer which you claim to be correct!
Perhaps you should check the expression that you are working with since, from what you have said, (100+10/s)/(k(s-a)(s-b)) is not equivalent to the above, initial expression.
I am using DERIVE for Windows v.4.11.

Cheers,

Boz Kempski.

**Frank Besig**

Declaring and Authoring the following in DfW 4.0 yields on simplifaction

```
#2:  a :∈ Real (0, ∞)
#1:  s :∈ Real (a, ∞)
```

$$\#3:\quad \text{LAPLACE}(\text{SIN}(a\cdot t),\ t,\ s) = \frac{\hat{e}^{-\infty\cdot \text{SIGN}(s)}\cdot(a+s)\cdot \text{SIN}(\infty)}{a^2+s^2} + \frac{a}{a^2+s^2}$$

We would like to get simply a/(a^2+s^2). Is there a way to achieve this?

**Terence Etchells, Liverpool, UK**

Declare s to be positive, that certainly does it in DfW version 4.11 (and DfW 5).

```
#4:  s :∈ Real (0, ∞)
```

$$\#5:\quad \text{LAPLACE}(\text{SIN}(a\cdot t),\ t,\ s) = \frac{a}{a^2+s^2}$$

**Johan Vegter, Netherlands** cad@hartman.nl

Hello,

I'm trying to fit several (many) datapoints to a predefined function.
Can somebody help me finding out how to do this job within Derive (v5)?

Many thanks, Johan Vegter

**Terence Etchells, Liverpool** T.A.Etchells@livjm.ac.uk

Use the FIT function.

Firstly express your data as an n x 2 matrix (data_matrix)

If you wish to fit this say to a quadratic function then use

```
FIT([x,ax^2+bx+c], data_matrix)
```

and approximate.

That's it.

Cheers, Terence

**TeM. Edward Borasky** znmeb@teleport.com http://www.teleport.com/~znmeb

That will work only if the fitted function is linear in the parameters you are fitting for, as in the example you give. Suppose, however, you need to fit the following:

$$y(a,b,c,x) = e^a\cdot\ln(b) + c\cdot\sqrt{x} + \frac{a+c+x}{\sqrt{b\cdot x}}$$

given a collection of points `[y, a, b, c, x]`.

If the partial derivatives exist wrt *a*, *b*, *c* and *x*, the easiest method is Levenberg-Marquardt. I'm pretty sure you can find this in "Numerical Recipes in C", among other places.

If the derivatives don't exist, there are quite a few methods that will work; essentially you convert it to a minimization problem and minimize the sum of squares of the collected y data values and the computed y values from the function. The minimization is over all the meaningful values of a, b, c, and x -- for example, in this case b and x must be positive. For a problem with only two parameters, compute the sum of squares function and find its minimum graphically on a 3D plot, then do an approximate "solve" to find the exact point.

In a later DERIVE Newsletter (DNL#79) Don Phillips contributed a great article on nonlinear regression methods including the Levenberg-Marquardt method mentioned above and a second one: the Gauss-Newton method.

$$\text{dat} := \begin{bmatrix} x & y \\ 1 & 2.3 \\ 2 & 2.6 \\ 3 & 2.7 \\ 5 & 3 \\ 8 & 3.05 \\ 10 & 3.3 \end{bmatrix}$$

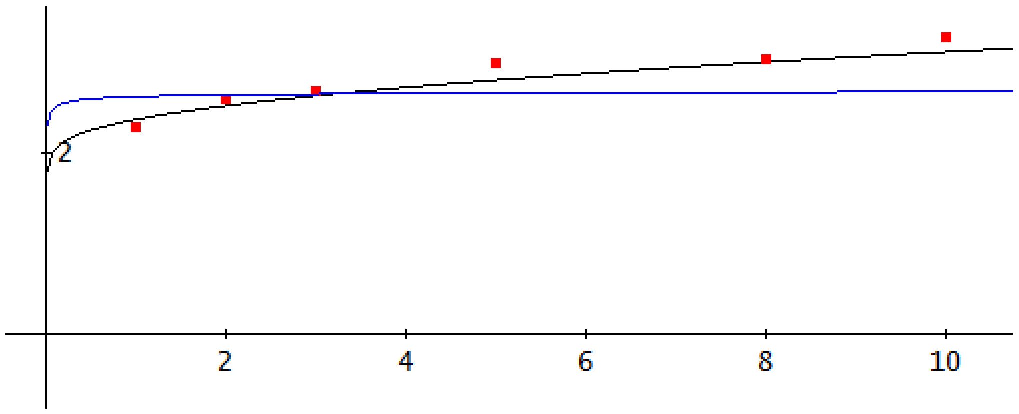
$$\text{GAUSS\_NEWTON}\left(y = e^{a}\cdot\text{LN}(b) + c\cdot\sqrt{x} + \frac{a + c + x}{\sqrt{(b\cdot x)}},\ [a,\ b,\ c],\ [2,\ 2,\ 2],\ \text{dat}\right)$$

$$y = 0.33147122\cdot\sqrt{x} - \frac{0.050354690}{\sqrt{x}} + 2.0907722$$

$$\text{MARQUARDT}\left(y = e^{a}\cdot\text{LN}(b) + c\cdot\sqrt{x} + \frac{a + c + x}{\sqrt{(b\cdot x)}},\ [a,\ b,\ c],\ [5,\ 5,\ 5],\ \text{dat}\right)$$

$$y = 0.0075770693\cdot\sqrt{x} - \frac{0.054932217}{\sqrt{x}} + 2.6738492$$

I tried applying both methods on the – strange – function given by Ramirez: the black one is the result of GAUSS_NEWTON which is much better than the regression function delivered by LEVENBERG-Marquardt. Josef



**Matija Lokar, Ljubljana, Slovenia**

Do you perhaps now how to "force" DERIVE to display a result in the form 1.3656^t when Precision := Approximate is used. All the times I get something like #e^(0.31159.. t).

$$\text{modify\_to\_a\_b\_on\_t(data\_)} := \text{VECTOR}\left(\left[\text{data\_}_{k,1},\ \text{LN}(\text{data\_}_{k,2})\right],\ k,\ 1,\ \text{DIMENSION(data\_)}\right)$$

$$\text{res(p\_)} := \text{FIT}([x,\ a\cdot x + b],\ \text{modify\_to\_a\_b\_on\_t(p\_)})$$

$$\left[\text{BB}(r) := \lim_{x\to 0} r,\ \text{AA}(r) := \left(\lim_{x\to 1} r\right) - \lim_{x\to 0} r\right]$$

$$\text{powfit(data, x)} := \text{APPROX}\left(e^{\text{BB(res(data))}} + \left(e^{\text{AA(res(data))}}\right)^{x},\ 5\right)$$

$$\text{merive} := \begin{bmatrix} 0 & 262 \\ 2 & 294 \\ 4 & 330 \\ 5 & 349 \\ 7 & 392 \\ 9 & 440 \\ 11 & 494 \\ 12 & 523 \end{bmatrix}$$

[Notation := Decimal, NotationDigits := 5]

$$\text{powfit(merive, x)} = e^{0.057628\cdot x} + 261.93$$

I would like to get `1.0593^x + 261.93` instead of the result above.

***DNL:*** I promised to put Matija´s question into the next DNL and he answered immediately:

Thank you. I tried some more ways to get around and now in most cases with

```
Notation := Decimal
APPROX(#e^(0.34),5)^x=(1.4049)^x
```

but:

```
APPROX(#e^(0.057628),5)^x=(0.94400)^(-x)
```

instead of prefered `1.0593^x`.

*Universities are places of knowledge.*
*The freshman each bring a little in with them, and the seniors take none away, so it accumulates.*

I'd recommend to rewrite the result, put the power of *e* between parentheses, highlight it and then approximate (press ≈), Josef

$$\left(e^{0.057628}\right)^{x} + 261.93$$

$$1.0593^{x} + 261.93$$

**George Dorner, USA** gdorner@harper.cc.il.us

A Springer book by Jean-Michel Ferrard and Henri Lemberg in French has been translated and is now available from Springer under our three names:

Visual Mathematics, Illustrated by the TI-92 and TI-89

See: http://www.springer.de/cgi-bin/search_book.pl?isbn=2-287-59685-2

This book is at a very high level - mostly university science and engineering students and teachers, but I´d be grateful for any advice in how to spread the word about it.

I thank Bernhard Kutzler and Josef Boehm for helping me to get in touch with Jean-Michel. That resulted in this collaboration.

(Not available, the link is not valid, Josef)

**Jean-Michel.Ferrard, Lyon, France** Jean-Michel.Ferrard@ac-lyon.fr

Dear all,

I've put some maths on the web, at

http://hpwww.ec-lyon.fr/hpserv/carip/math/prepa/

This is the first version of a big project and that's why I've been so busy and unassuming during the last year.

By now, this site has two defects:

1) It's in french.
2) You must download and install IBM TechExplorer on your computer, otherwise you'll only see the home page!

Here is the URL where you can download IBM TechExplorer.

http://www.software.ibm.com/techexplorer/

In the next century, I'll work hard on this project. I intend to present some alternative versions of the site, using other file formats: PDF, DVI, PS, MathML, and so on.

Sincerely

Jean-Michel Ferrard

The above link is not available. I got the information given below:

Welcome on this 'Ecole Centrale de Lyon' hosted server. Sorry but you are not allowed to get the requested document.

But I found another – very rich – resource provided by Michel Ferrard. Use the URL – and practise your French1

http://www.mathprepa.fr/

You can also find the IBM TechExplorer in the net.

# A Continued Fraction Expansion

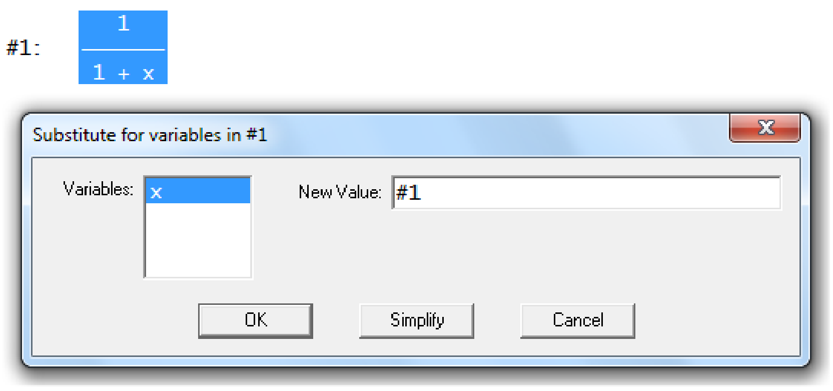
Carl Leinbach & Marvin Brubaker
Gettysburg College, PA, USA

In this brief section we will learn about using the Substitute option to build a rather complex expression. First be sure that you are starting with a clean "slate" (and memory).

We begin the investigation by editing the expression

$$\#1: \quad \frac{1}{1 + x}$$

With this expression highlighted press the SUB-key. Then you are asked for a value of x, type `#1` (the expression you have just authored). When you press Ok the following expression will appear on your screen.


#1: $\frac{1}{1 + x}$

Substitute for variables in #1

Variables: x

New Value: #1

$$\#2: \quad \frac{1}{1 + \frac{1}{1 + x}}$$

Now repeat this process three more times. Each time you are prompted for a value of x, respond with `#1`. Your last expression will be:

$$\#5: \quad \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + x}}}}}$$

At this point we are going to substitute the value 1 for x in all of these expressions. We will do it the easy way. Enter the vector:

[#1, #2, #3, #4 #5]

This gives a vector containing each step in our continued fraction expansion. Choose SUB again and replacing the value of x with 1, we have after simplifying

#7: $\left[\frac{1}{2}, \frac{2}{3}, \frac{3}{5}, \frac{5}{8}, \frac{8}{13}\right]$

Does this look familiar? What do you suppose will be the limit of the sequence?

(Stefan Welke wrote an article on Continued Fractions which together with Josef Lechner´s following contribution will accomplish this "Mini-Series on CF". Josef)

Teachers often are claiming about automatic simplification after entering an expression (in DERIVE and on the various TIs as well). Many colleagues would like to have an option to avoid even the slightest autosimplification, eg 1 + 3 should remain as 1 + 3 and fractions should remain uncancelled etc.

DfW4 and also DfW5 provides a means to avoid evaluation:

> The quote operator is useful for quoting expressions in calls on functions which you do not want to have evaluated before being passed.

```
#1:   Josef Lechner, Implementation of a Function to generate Continued Fractions, June 1998

#2:   SUMM(a, b) := '(a + b)

#3:   [SUMM(2, 3), SUMM(3, 2)] = [2 + 3, 3 + 2]
```

#4: $\text{FRAC}(a, b) := '\left(\frac{a}{b}\right)$

#5: $\text{FRAC}(15, 3) = \frac{15}{3}$

FRAC does not work in DERIVE6, but CF works pretty well, Josef

#6: $\text{FRAC}(9, \text{FRAC}(15, 3)) = \frac{9}{\frac{15}{3}}$

```
CF(m, n) :=
  If MOD(m, n) ≠ 0
     FLOOR(m, n) + FRAC(1, CF(n, MOD(m, n)))
     m/n
```

$$\text{CF}(131, 100) = \frac{1}{\frac{1}{\frac{1}{\frac{1}{3} + 2} + 4} + 3} + 1$$

```
CFO(m, n) :=
  If MOD(m, n) ≠ 0
     SUMM(FLOOR(m, n), FRAC(1, CFO(n, MOD(m, n))))
     m/n
```

$$\text{CFO}(131, 100) = 1 + \frac{1}{3 + \frac{1}{4 + \frac{1}{2 + \frac{1}{3}}}}$$

```
CF(√2, 1)
```

Compare with the output of **`CFO(√2,1)`**

$$\cfrac{1}{\cfrac{1}{\cfrac{1}{\cfrac{1}{\cfrac{1}{\cfrac{1}{\cfrac{1}{\cfrac{1}{\cfrac{1}{\cfrac{1}{\cfrac{1}{\cfrac{1}{\cfrac{1}{\cfrac{1}{\cfrac{1}{\cfrac{1}{\cfrac{1}{\cfrac{1}{\cfrac{1}{\cfrac{1}{\cfrac{1}{\cfrac{1}{\cfrac{1}{\cfrac{1}{\cfrac{1}{\sqrt{2}+1}+2}+2}+2}+2}+2}+2}+2}+2}+2}+2}+2}+2}+2}+2}+2}+2}+2}+2}+2}+2}+2}+2}+2}+2}+1$$

# Akima Splines outperform Cubic Splines

Robert Geruschkat, Ronneberg, Germany

*There was a letter to SWHH:*

With regard to: AKIMA/Interpolation of 2-dimensional sets.

Sirs,

the job of plotting accurate curves based on test data lead to computing splines according a method developed by B. AKIMA, Japan, described in the German computing magazine c`t, June 1989.

On the enclosed diskette you find some routines which do this quite nicely. The routines are of help whenever the curves are of a non-continuous character, e.g. produced by hydraulic forces reduced to a constant level by valves limiting the hydraulic pressure. Under these unusual conditions, the AKIMA-spline is the more satisfying one.

.....

I´m about 70, nowadays doing mathematics just for fun, having been in the computing business in former years.

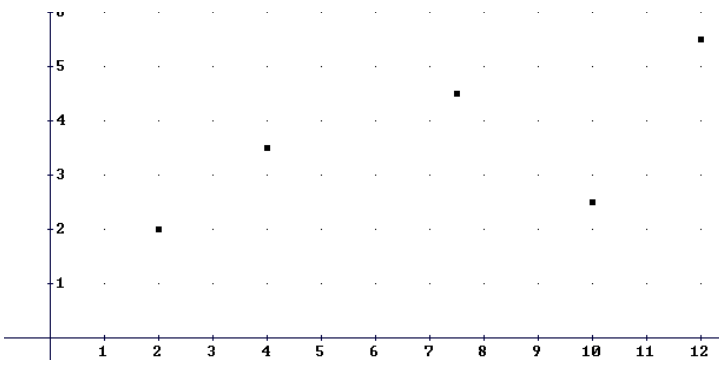
......

*And a letter to the DUG followed:*

Dear Mr Böhm,

enclosed you will find a copy of a letter which I sent to SWHH and a diskette containing MTH-files to create AKIMA-splines. Al Rich from SWHH encouraged me to send these files to you.
Best regards
Robert Geruschkat.

*As I had never heard about AKIMA-splines before I asked Mr Geruschkat for more explanations. He sent a copy of the article in c`t. After studying this together with his files I found it a very interesting contribution because of two reasons:*
*The topic is not too difficult to be presented in secondary-school and it demonstrates in an excellent way, how a CAS can help to do the boring manipulation work when the strategy about what to do is made clear.*



A set of *n* discrete points is given - in most cases data points from measurements - and we want to find an interpolating curve which passes the given points. This problem covers some chapters in most Numerical Analysis textbooks ([1], [2]).

You can use a polynomial function of grade *n*-1 - e.g. using Lagrange Interpolation - and many others.

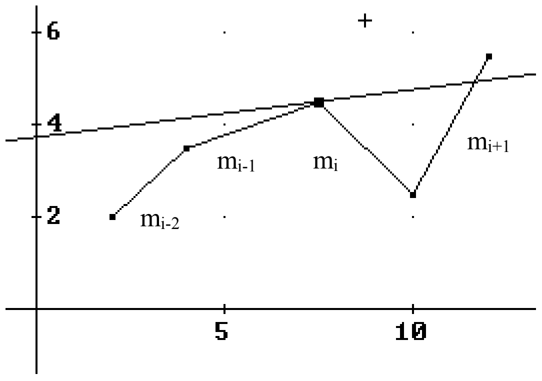
One very common method is generating a list of cubics, which then piecewise define the desired function. The cubics are connecting two following points under certain conditions. Two successive cubics meet in point ($x_i$, $y_i$) having in common the function value, the first and the second derivative. (DNL#18 and DNL#19).

In 1970 Hiroshi Akima found a new way generating a list of interpolating cubics, having in mind the way how to connect the nodes by hands using pencil and paper only. Based on the Hermite-Interpolation he doesn´t use the 2nd derivative, but demands that two successive cubics have a certain slope in common in the nodes.

Follow the explanations together with Mr Geruschkat´s MTH-file (and the following TI-script as well).

For *n* given nodes we generate *n*-1 cubics. The cubic in the *i*-th segment is defined by

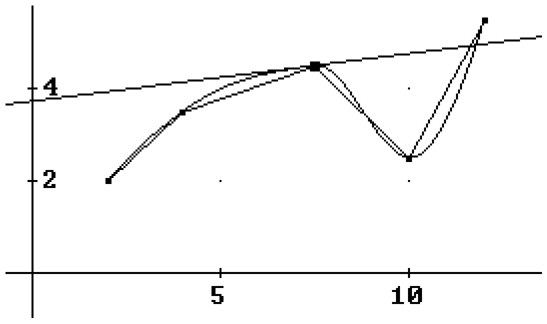
$$P_i(x) = k_0 + k_1(x - x_i) + k_2(x - x_i)^2 + k_3(x - x_i)^3$$ under the following conditions:



$$\begin{aligned} P_i(x_i) &= y_i \\ P_i(x_{i+1}) &= y_{i+1} \\ P_i'(x_i) &= t_i \\ P_i'(x_{i+1}) &= t_{i+1} \end{aligned} \qquad (*)$$

with $m_i = \dfrac{y_{i+1} - y_i}{x_{i+1} - x_i}$ the value of the $t_i$ is defined by

$$t_i = \frac{|m_{i+1} - m_i| \cdot m_{i-1} + |m_{i-1} - m_{i-2}| \cdot m_i}{|m_{i+1} - m_i| + |m_{i-1} - m_{i-2}|}$$



The picture shows the tangent in the 3rd node with $t_3$ calculated using $m_1$ through $m_4$.

This is a very welcome chance to present a

**DERIVE worksheet**

- a combination of DERIVE code, comments and graphics:

First of all we will illustrate the system of equations from above (*): This interpolation is known as *Hermite*-interpolation.

```
#1:    InputMode := Word

                                          2                 3
#2:    p(x) := k0 + k1·(x - x_i) + k2·(x - x_i)  + k3·(x - x_i)

                                                     2
#3:    p1(x) := k1 + 2·k2·(x - x_i) + 3·k3·(x - x_i)

#4:    [eq1 := p(x_i) = y_i, eq2 := p(x_i_1) = y_i_1]

#5:    [eq3 := p1(x_i) = t_i, eq4 := p1(x_i_1) = t_i_1]

#6:    SOLVE([eq1, eq2, eq3, eq4], [k0, k1, k2, k3])
```

#7: $$\left[k0 = y\_i \wedge k1 = t\_i \wedge k2 = \frac{2\cdot t\_i\cdot(x\_i - x\_i\_1) + t\_i\_1\cdot(x\_i - x\_i\_1) - 3\cdot(y\_i - y\_i\_1)}{(x\_i - x\_i\_1)^2} \wedge k3 = \frac{t\_i\cdot(x\_i - x\_i\_1) + t\_i\_1\cdot(x\_i - x\_i\_1) - 2\cdot(y\_i - y\_i\_1)}{(x\_i - x\_i\_1)^3}\right]$$

We need for calculating the slope in all nodes two points before and two points after. They are fixed by extrapolation usng a quadratic at the begin of the data list and at its end.

```
#1:   [Precision := Exact, CaseMode := Sensitive, InputMode := Word, NotationDigits := 5]

#2:   [dim(n) := DIMENSION(n), na := 1]
```

## The given nodes:  (n = 5)

#3: test := $$\begin{bmatrix} 2 & 2 \\ 4 & 3.5 \\ 7.5 & 4.5 \\ 10 & 2.5 \\ 12 & 5.5 \end{bmatrix}$$

#4: $q(x) := q\_2\cdot x^2 + q\_1\cdot x + q\_0$

```
#5:   SOLVE([q(x1) = y1, q(x2) = y2, q(x3) = y3], [q_0, q_1, q_2])
```

#6: $$\left[q\_0 = \frac{x2\cdot x3\cdot(x1\cdot(y2 - y3) + x2\cdot(y3 - y1) + x3\cdot(y1 - y2))}{(x1^2 - x1\cdot(x2 + x3) + x2\cdot x3)\cdot(x3 - x2)} + \frac{x3\cdot(y3 - y2)}{x2 - x3} + y3 \wedge q\_1 = \frac{x1^2\cdot(y2 - y3) + x2^2\cdot(y3 - y1) + x3^2\cdot(y1 - y2)}{(x1^2 - x1\cdot(x2 + x3) + x2\cdot x3)\cdot(x2 - x3)} \wedge q\_2 = \frac{x1\cdot(y2 - y3) + x2\cdot(y3 - y1) + x3\cdot(y1 - y2)}{(x1^2 - x1\cdot(x2 + x3) + x2\cdot x3)\cdot(x3 - x2)}\right]$$

These are the coefficients of the extrapolation parabolas:

#7: $$\mathrm{q0}(x1, y1, x2, y2, x3, y3) := \frac{x1^2\cdot(x2\cdot y3 - x3\cdot y2) + x1\cdot(x3^2\cdot y2 - x2^2\cdot y3) + x2\cdot x3\cdot y1\cdot(x2 - x3)}{(x1 - x2)\cdot(x1 - x3)\cdot(x2 - x3)}$$

#8: $$\mathrm{q1}(x1, y1, x2, y2, x3, y3) := \frac{x1^2\cdot(y2 - y3) + x2^2\cdot(y3 - y1) + x3^2\cdot(y1 - y2)}{(x1 - x2)\cdot(x1 - x3)\cdot(x2 - x3)}$$

#9: $$\mathrm{q2}(x1, y1, x2, y2, x3, y3) := \frac{x1\cdot(y2 - y3) + x2\cdot(y3 - y1) + x3\cdot(y1 - y2)}{(x1 - x2)\cdot(x1 - x3)\cdot(x3 - x2)}$$

#10: $$\mathrm{YU}(u, x) := \mathrm{q2}(u_{1,1}, u_{1,2}, u_{2,1}, u_{2,2}, u_{3,1}, u_{3,2})\cdot x^2 + \mathrm{q1}(u_{1,1}, u_{1,2}, u_{2,1}, u_{2,2}, u_{3,1}, u_{3,2})\cdot x + \mathrm{q0}(u_{1,1}, u_{1,2}, u_{2,1}, u_{2,2}, u_{3,1}, u_{3,2})$$

#11: $$\mathrm{YO}(u, n\_, x) := \mathrm{q2}(u_{n\_ - 2,1}, u_{n\_ - 2,2}, u_{n\_ - 1,1}, u_{n\_ - 1,2}, u_{n\_,1}, u_{n\_,2})\cdot x^2 + \mathrm{q1}(u_{n\_ - 2,1}, u_{n\_ - 2,2}, u_{n\_ - 1,1}, u_{n\_ - 1,2}, u_{n\_,1}, u_{n\_,2})\cdot x + \mathrm{q0}(u_{n\_ - 2,1}, u_{n\_ - 2,2}, u_{n\_ - 1,1}, u_{n\_ - 1,2}, u_{n\_,1}, u_{n\_,2})$$

YU(u,x) is parabola determined by the first three data points of matrix u, YO(u,x,n) the parabola given by the last three data points of u (containing n points), both of them are needed to calculate two extrapolation points which are attached at the beginning and at the end of the given set of points. At the left we find $x_{-1}$ and $x_{-2}$ with $x_3 - x_1 = x_2 - x_{-1} = x_1 - x_{-2}$ and at the right we have $x_n - x_{n-2} = x_{n+1} - x_{n-1} = x_{n+2} - x_n$.

#12: $$\mathrm{YU}(test, x) = - \frac{13\cdot x^2}{154} + \frac{387\cdot x}{308} - \frac{27}{154}$$

#13: $$\mathrm{YO}(test, 5, x) = \frac{23\cdot x^2}{45} - \frac{877\cdot x}{90} + \frac{293}{6}$$

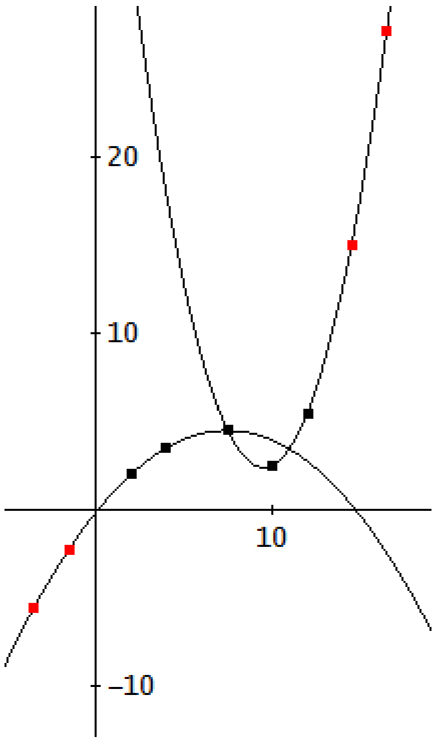
`Right(matrix)` and `left(matrix)` give the four extra points

#14: $$\mathrm{left}(wm) := \begin{bmatrix} 2\cdot wm_{1,1} - wm_{3,1} & \mathrm{YU}(wm, 2\cdot wm_{1,1} - wm_{3,1}) \\ wm_{1,1} + wm_{2,1} - wm_{3,1} & \mathrm{YU}(wm, wm_{1,1} + wm_{2,1} - wm_{3,1}) \end{bmatrix}$$

#15: right(wm, n_) :=

$$\begin{bmatrix} wm_{n\_,1} + wm_{n\_-1,1} - wm_{n\_-2,1} & YO(wm, n\_, wm_{n\_,1} + wm_{n\_-1,1} - wm_{n\_-2,1}) \\ 2\cdot wm_{n\_,1} - wm_{n\_-2,1} & YO(wm, n\_, 2\cdot wm_{n\_,1} - wm_{n\_-2,1}) \end{bmatrix}$$

You can see the given points (black) together with the extrapolated ones (red) on the two parabolas.



#16: left(test) =

$$\begin{bmatrix} -\frac{7}{2} & -\frac{157}{28} \\ -\frac{3}{2} & -\frac{9}{4} \end{bmatrix}$$

#17: right(test, 5) =

$$\begin{bmatrix} \frac{29}{2} & 15 \\ \frac{33}{2} & \frac{136}{5} \end{bmatrix}$$

epm(wm) gives the extended point list in form of a matrix.

```
#18:   epm(wm) := APPEND(APPEND(left(wm), wm), right(wm, dim(wm)))
```

#19: APPROX(epm(test)) =

$$\begin{bmatrix} -3.5 & -5.607142857 \\ -1.5 & -2.25 \\ 2 & 2 \\ 4 & 3.5 \\ 7.5 & 4.5 \\ 10 & 2.5 \\ 12 & 5.5 \\ 14.5 & 15 \\ 16.5 & 27.2 \end{bmatrix}$$

Let's refer to the formulae for the $m_i$- *and* $t_i$-values given above:

We start calculating the t-values with calculating the m-values first.

#20: $$\left[\mathrm{dx}(e,\ i) := e_{i+1,1} - e_{i,1},\ \mathrm{dy}(e,\ i) := e_{i+1,2} - e_{i,2}\right]$$

#21: $$\mathrm{MI}(wm) := \mathrm{VECTOR}\left(\frac{\mathrm{dy}(\mathrm{epm}(wm),\ i)}{\mathrm{dx}(\mathrm{epm}(wm),\ i)},\ i,\ 1,\ \mathrm{dim}(wm) + 3\right)$$

#22: $$\mathrm{APPROX}(\mathrm{MI}(test)) = [1.6785,\ 1.2142,\ 0.75,\ 0.28571,\ -0.8,\ 1.5,\ 3.8,\ 6.1]$$

#23: $$\mathrm{TI}(w) := \mathrm{VECTOR}\Bigg(\mathrm{IF}\Bigg(\Big|\left|(\mathrm{MI}(w))_{j+1} - (\mathrm{MI}(w))_j\right| + \left|(\mathrm{MI}(w))_{j-1} - (\mathrm{MI}(w))_{j-2}\right|\Big| < 0.01,$$
$$\frac{\left|(\mathrm{MI}(w))_{j+1} - (\mathrm{MI}(w))_j\right|\cdot(\mathrm{MI}(w))_{j-1} + \left|(\mathrm{MI}(w))_{j-1} - (\mathrm{MI}(w))_{j-2}\right|\cdot(\mathrm{MI}(w))_j}{0.01},$$
$$\frac{\left|(\mathrm{MI}(w))_{j+1} - (\mathrm{MI}(w))_j\right|\cdot(\mathrm{MI}(w))_{j-1} + \left|(\mathrm{MI}(w))_{j-1} - (\mathrm{MI}(w))_{j-2}\right|\cdot(\mathrm{MI}(w))_j}{\left|(\mathrm{MI}(w))_{j+1} - (\mathrm{MI}(w))_j\right| + \left|(\mathrm{MI}(w))_{j-1} - (\mathrm{MI}(w))_{j-2}\right|}\Bigg),$$
$$j,\ 3,\ \mathrm{dim}(w) + 2\Bigg)$$

The “recipe” for finding the coefficients of the splines is given on page 14.

I check the correctness of the functions by applying them on the `test`-points.

#25: $$\mathrm{k0}(wm) := \mathrm{VECTOR}(wm_{j,2},\ j,\ \mathrm{DIM}(wm) - 1)$$

#26: $$\mathrm{k0}(test) = \left[2,\ \frac{7}{2},\ \frac{9}{2},\ \frac{5}{2}\right]$$

#27: $$\mathrm{k1}(u) := \mathrm{DELETE\_ELEMENT}(\mathrm{TI}(u),\ \mathrm{dim}(u))$$

#28: $$\mathrm{k1}(test) = \left[\frac{55}{56},\ \frac{928}{1519},\ \frac{40}{387},\ -\frac{74}{1185}\right]$$

#29: $$\mathrm{k2}(wm) := \mathrm{VECTOR}\left(\frac{3\cdot\mathrm{dy}(\mathrm{epm}(wm),\ i) - \mathrm{dx}(\mathrm{epm}(wm),\ i)\cdot(2\cdot(\mathrm{TI}(wm))_{i-2} + (\mathrm{TI}(wm))_{i-1})}{\mathrm{dx}(\mathrm{epm}(wm),\ i)^2},\right.$$
$$\left.i,\ 3,\ \mathrm{dim}(wm) + 1\right)$$

#30: $$\mathrm{k2}(test) = \left[-\frac{247}{1519},\ -\frac{550316}{4114971},\ -\frac{155572}{152865},\ \frac{9361}{9480}\right]$$

#31: $$\mathrm{k3(wm)} := \mathrm{VECTOR}\left(\frac{((\mathrm{TI(wm)})_{i-1} + (\mathrm{TI(wm)})_{i-2})\cdot \mathrm{dx(epm(wm),\ i)} - 2\cdot \mathrm{dy(epm(wm),\ i)}}{\mathrm{dx(epm(wm),\ i)}^3},\ i,\ 3,\ \mathrm{dim(wm)} + 1\right)$$

#32: $$\mathrm{k3(test)} = \left[\frac{1131}{48608},\ \frac{335920}{28804797},\ \frac{1003352}{3821625},\ -\frac{391}{3792}\right]$$

We will have the following useful functions:

| | | |
|---|---|---|
| `AKMC(inputmatrix)` | --> | Matrix of the AKIMA-spline-coefficients |
| `AKMI(inputmatrix,x0)` | --> | Interpolationvalue for x0 |
| `AKMPL(inputmatrix, step)` | --> | Interpolation value table |
| `AKMSPL(inputmatrix, variable)` | --> | the splines as a piecewiese defined function |

#33: AKMC(wm) := [k0(wm), k1(wm), k2(wm), k3(wm)]`

#34: $$\mathrm{AKMC(test)} = \begin{bmatrix} 2 & \frac{55}{56} & -\frac{247}{1519} & \frac{1131}{48608} \\ \frac{7}{2} & \frac{928}{1519} & -\frac{550316}{4114971} & \frac{335920}{28804797} \\ \frac{9}{2} & \frac{40}{387} & -\frac{155572}{152865} & \frac{1003352}{3821625} \\ \frac{5}{2} & -\frac{74}{1185} & \frac{9361}{9480} & -\frac{391}{3792} \end{bmatrix}$$

#35: $$\mathrm{AKMC(test)} = \begin{bmatrix} 2 & 0.98214 & -0.16260 & 0.023267 \\ 3.5 & 0.61092 & -0.13373 & 0.011661 \\ 4.5 & 0.10335 & -1.0177 & 0.26254 \\ 2.5 & -0.062447 & 0.98744 & -0.10311 \end{bmatrix}$$

AKMI is a nice recursive function

```
       AKMI(wm, x, na) :=
         If x < wm↓na↓1 ∨ x > wm↓dim(wm)↓1
            ?
#36:        If wm↓na↓1 ≤ x ≤ wm↓(na + 1)↓1
               [x, (AKMC(wm))↓na↓1 + (x - wm↓na↓1)·(AKMC(wm))↓na↓2 + (x -
               AKMI(wm, x, na + 1)



         wm↓na↓1)^2·(AKMC(wm))↓na↓3 + (x - wm↓na↓1)^3·(AKMC(wm))↓na↓4]
```

#37: AKMI(test, 5) = [5, 3.9888]

#39: $\text{AKMPL(wm, step, na)} := \text{VECTOR}(\text{AKMI(wm, x, 1), x, } wm_{1,1}, wm_{\dim(wm),1}, \text{step})$

#40: $\text{AKMPL(test, 1)} = \begin{bmatrix} 2 & 2 \\ 3 & 2.8428 \\ 4 & 3.5 \\ 5 & 3.9888 \\ 6 & 4.2802 \\ 7 & 4.444 \\ 8 & 4.33 \\ 9 & 3.2512 \\ 10 & 2.5 \\ 11 & 3.3218 \\ 12 & 5.5 \end{bmatrix}$

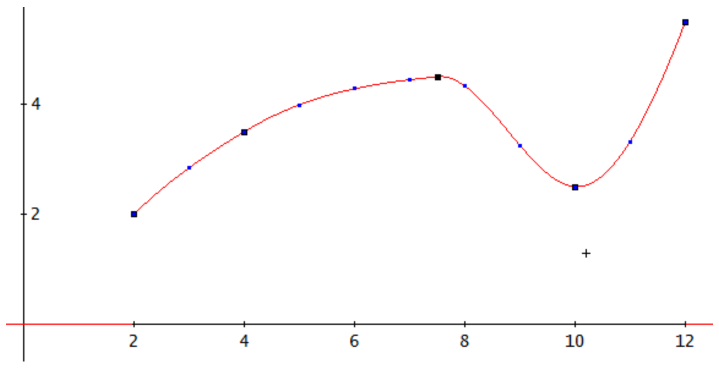
Needs 460 sec calculation time.

We can use the CHI-function to produce the piecewise defined spline function.

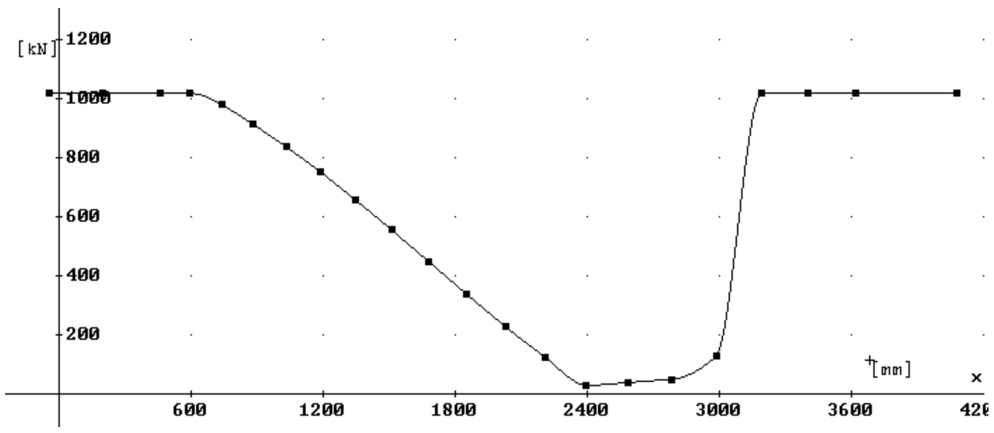
#41: $$\text{AKMSPL(wm, x)} := \sum_{i=1}^{\dim(wm)-1} \chi(wm_{i,1}, x, wm_{i+1,1}) \cdot \sum_{j=1}^{4} (\text{AKMC(wm)})_{i,j} \cdot (x - wm_{i,1})^{j-1}$$

#42: EXPAND(AKMSPL(test))

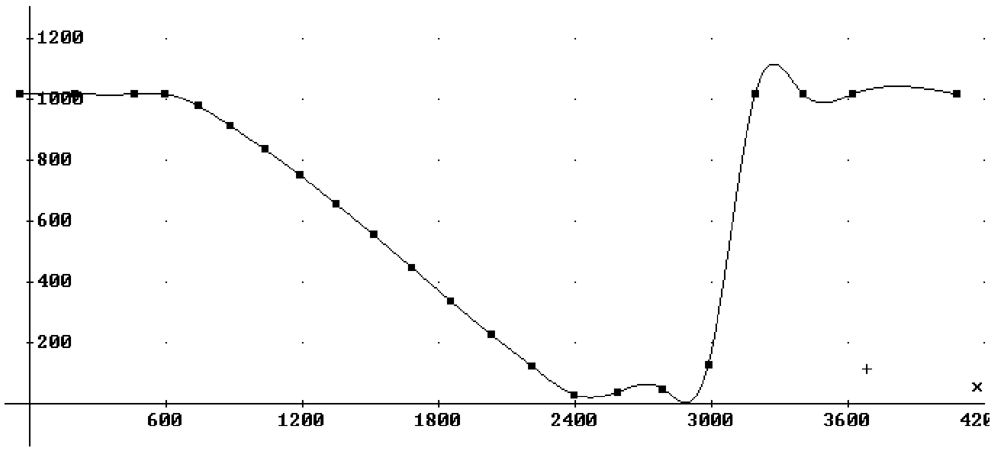
#43: $$0.12544 \cdot x^3 \cdot \text{SIGN}(2 \cdot x - 15) - 3.3256 \cdot x^2 \cdot \text{SIGN}(2 \cdot x - 15) + 28.716 \cdot x \cdot \text{SIGN}(2 \cdot x - 15) - 81.226 \cdot \text{SIGN}(2 \cdot x - 15) + 0.051555 \cdot x^3 \cdot \text{SIGN}(x - 12) - 2.0404 \cdot x^2 \cdot \text{SIGN}(x - 12) + 25.372 \cdot x \cdot \text{SIGN}(x - 12) - 102.49 \cdot \text{SIGN}(x - 12) - 0.18282 \cdot x^3 \cdot \text{SIGN}(x - 10) + 5.5028 \cdot x^2 \cdot \text{SIGN}(x - 10) - 55.209 \cdot x \cdot \text{SIGN}(x - 10) + 184.63 \cdot \text{SIGN}(x - 10) - 0.0058029 \cdot x^3 \cdot \text{SIGN}(x - 4) + 0.014267 \cdot x^2 \cdot \text{SIGN}(x - 4) + 0.16439 \cdot x \cdot \text{SIGN}(x - 4) - 0.51449 \cdot \text{SIGN}(x - 4) + 0.011633 \cdot x^3 \cdot \text{SIGN}(x - 2) - 0.1511 \cdot x^2 \cdot \text{SIGN}(x - 2) + 0.95589 \cdot x \cdot \text{SIGN}(x - 2) - 0.40042 \cdot \text{SIGN}(x - 2)$$



The plot shows the given points (large), the interpolated points (medium size and blue) and the Akima-spline. We could produce a TABLE using the spline function but we will notify that the function values in the given points are undefined because at these points the single cubics are connected by the CHI-function.

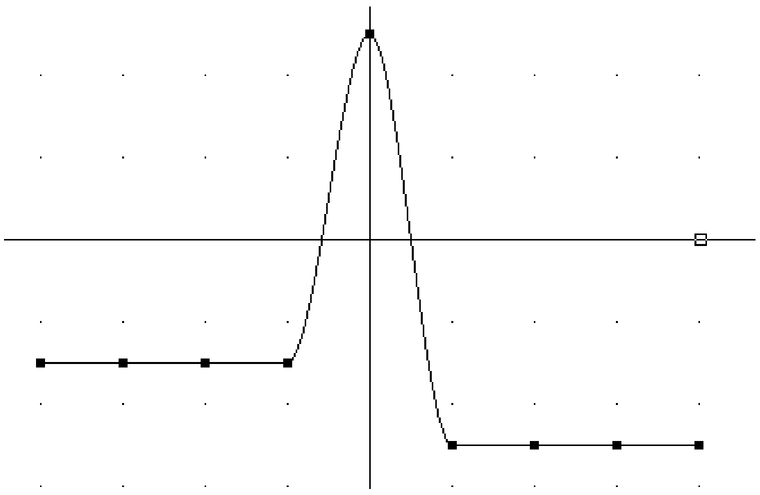


AKIMA-spline (above) versus cubic spline (below).



It is not necessary to display several lines of function code, if you are not interested in this bulky expression. Switch to the plot window and plot the result having activated the option Approximate Before Plotting. But be patient, give some minutes calculation time.

$$\text{last} := \begin{bmatrix} -4 & -1.5 \\ -3 & -1.5 \\ -2 & -1.5 \\ -1 & -1.5 \\ 0 & 2.5 \\ 1 & -2.5 \\ 2 & -2.5 \\ 3 & -2.5 \\ 4 & -2.5 \end{bmatrix}$$



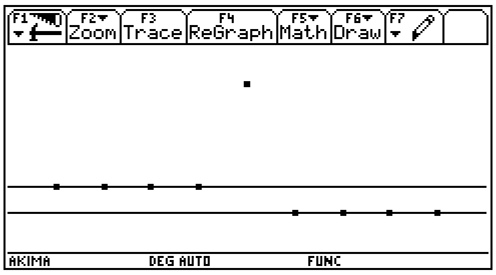
```
:Akimaspline
 :Enter the matrix of the given points and store as pts
 :
C:[⁻4,⁻1.5;⁻3,⁻1.5;⁻2,⁻1.5;⁻1,⁻1.5;Ø,2.5;1,⁻2.5;2,⁻2.5;3,⁻2.5;4,⁻2.5]→pts
 :
C:⁻5→xmin:5→xmax:1→xscl
C:⁻4→ymin:4→ymax:1→yscl
 :1. Quadratic Extrapolation in general
 :
C:dim(pts)[1]→pn
C:seq(pts[i,1],i,1,3)→l1
C:seq(pts[i,2],i,1,3)→l2
C:quadreg l1,l2
C:regeq(x)→yu(x)
C:yu(x)
C:ans(1)→y1(x)
 :
C:seq(pts[i,1],i,pn-2,pn)→l1
C:seq(pts[i,2],i,pn-2,pn)→l2
C:quadreg l1,l2
C:regeq(x)→yo(x)
C:yo(x)
C:ans(1)→y2(x)
C:seq(pts[i,1],i,1,pn)→l1
C:seq(pts[i,2],i,1,pn)→l2
C:newplot 1,1,l1,l2,,,,4
```

It might be nice to follow the AKIMA-procedure reading a TI-92 script. Compare the steps and you will recognize many similarities with the DERIVE - file.

Step by step one can transfer both the derivation and the realisation on the TI and make clear the algorithm to students.
Josef

I used some other data points (being also of a nature which makes "AKIMA"-application useful).

```
 :
 :switch to Graph-Window and plot the data points
 :
 :two additional points before:
 :
C:[2*pts[1,1]-pts[3,1],y1(2*pts[1,1]-pts[3,1]);pts[1,1]+pts[2,1]-
pts[3,1],y1(pts[1,1]+pts[2,1]-pts[3,1])]→pv
 :
 :two additional points after
 :
C:[pts[pn,1]+pts[pn-1,1]-pts[pn-2,1],y2(pts[pn,1]+pts[pn-1,1]-pts[pn-
2,1]);2*pts[pn,1]-pts[pn-2,1],y2(2*pts[pn,1]-pts[pn-2,1])]→ph
 :
C:augment(augment(pvᵀ,ptsᵀ),phᵀ)ᵀ→epts
 :
C:seq((epts[i+1,2]-epts[i,2])/(epts[i+1,1]-epts[i,1]),i,1,dim(epts)[1]-
1)→mi
 :
C:seq(when(abs(abs(mi[j+1]-mi[j])+abs(mi[j-1]-mi[j-2]))<Ø.Ø1,(abs(mi[j+1]-
mi[j])*mi[j-1]+abs(mi[j-1]-mi[j-2])*mi[j])/Ø.Ø1,(abs(mi[j+1]-mi[j])*mi[j-
1]+abs(mi[j-1]-mi[j-2])*mi[j])/(abs(mi[j+1]-mi[j])+abs(mi[j-1]-mi[j-
2]))),j,3,dim(mi)-1)→t
 :
C:a+b*(x-xi)+c*(x-xi)^2+d*(x-xi)^3→spli(x)
 :
C:spli(xi)=yi
 :a[i] = pts[i,2]
 :
C:seq(pts[i,2],i,1,dim(pts)[1]-1)→ai
 :

C:spli(xii)=yii
C:d(spli(x),x)=ti|x=xi
 :
```

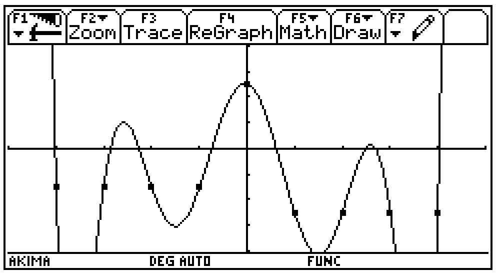


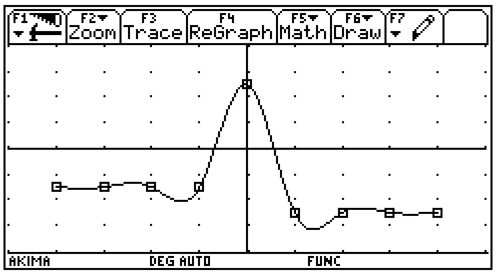
The two quadratic extrapolation functions which in this case appear as horizontal lines.

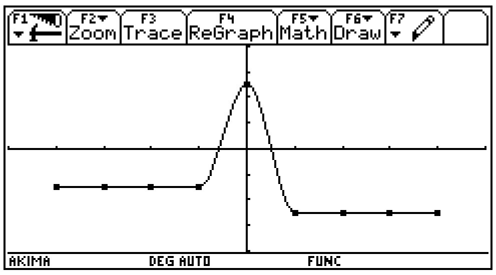
```
C:d(spli(x),x)=tii|x=xii
 :b[i] = t[i]
 :
C:seq(t[i],i,1,dim(t)-1)→bi
:subst for a = yi and b = ti
 :
C:expand(spli(xii)=yii)| a=yi and b=ti
C:factor(ans(1))
 :
C:expand(d(spli(x),x)=tii)| a=yi and b=ti and x = xii
C:factor(ans(1))
 :
C:rref([[(xi-xii)^2,⁻(xi-xii)^3,ti*(xi-xii)+yii-yi][⁻2*(xi-xii),3*(xi-
xii)^2,tii-ti]])→aux
 :
C:aux[1,3]
 :
 :here are the coefficients c[i]:
 :
C:seq(⁻(2*t[i]+t[i+1]-3*mi[i+2])/(pts[i+1,1]-pts[i,1]),i,1,dim(pts)[1]-
1)→ci
 :
C:seq((t[i]+t[i+1]-2*mi[i+2])/(pts[i+1,1]-pts[i,1])^2,i,1,dim(pts)[1]-1)→di
 :
 :AKIMA - Spline list
C:seq(di[i]*(x-pts[i,1])^3+ci[i]*(x-pts[i,1])^2+bi[i]*(x-
pts[i,1])+ai[i],i,1,pn-1)
C:1→s
C:when(x<pts[s,1] or x>pts[dim(pts)[1],1],"?",when(x≥pts[s,1] and
x≤pts[s+1,1],[x,ai[s]+bi[s]*(x-pts[s,1])+ci[s]*(x-pts[s,1])^2+di[s]*(x-
pts[s,1])^3],splint(x,s+1)))→splint(x,s)
 :
C:when(x<pts[s,1] or x>pts[dim(pts)[1],1],undef,splint(x,s)[1,2])→spligr(x)
 :
 :Plot spligr(x) in the full Graph-Window
```

*Unfortunately the when() - construct to plot the function as a whole doesn´t work on the TI-92PLUS in this form (the PLUS seems to handle the functions in another way than the "ordinary" TI-92). I didn´t find another solution as to add the functions piece by piece.*

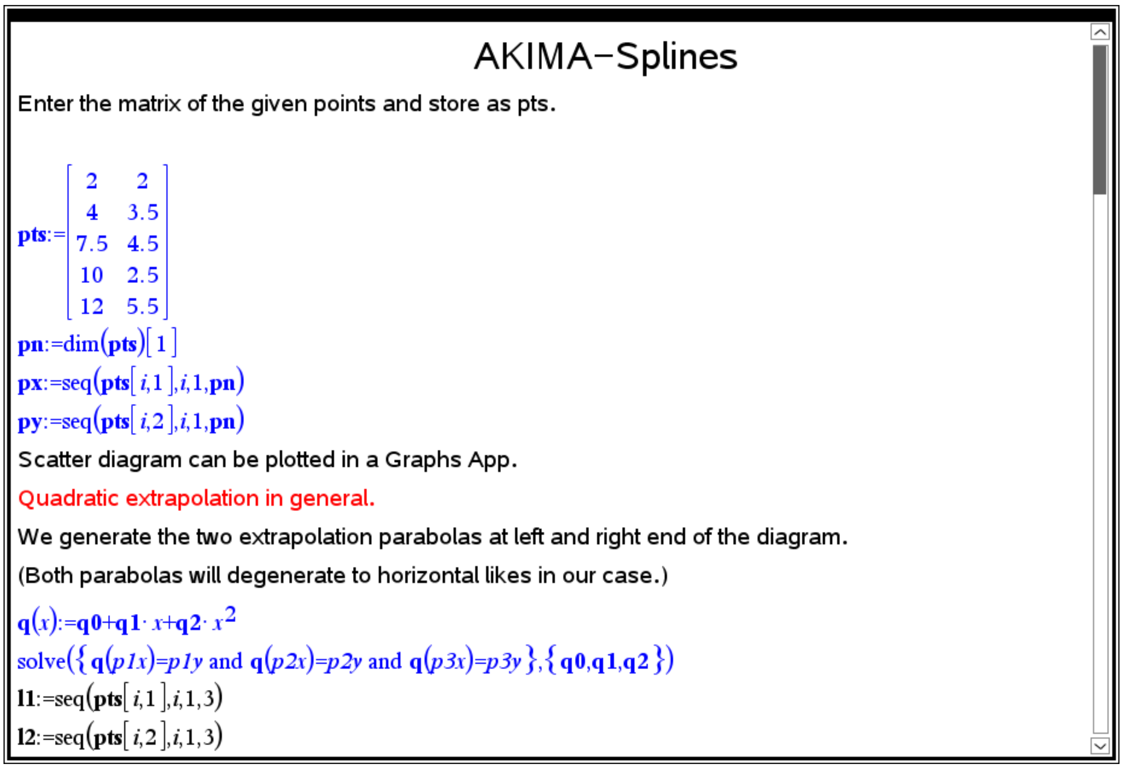
The screen shots below show the big difference between various interpolations: (1) the Lagrange Polynomial [1], (2) the Cubic Spline [2], [3] and (3) the AKIMA-Spline.







We don't have the wonderful script-tool – highly appreciated by many of us – on TI-NspireCAS but we can perform the demonstration in a Notes-Application (Josef):



AKIMA–Splines

Enter the matrix of the given points and store as pts.

$$\mathbf{pts}:=\begin{bmatrix}2 & 2\\ 4 & 3.5\\ 7.5 & 4.5\\ 10 & 2.5\\ 12 & 5.5\end{bmatrix}$$

$\mathbf{pn}:=\dim(\mathbf{pts})[1]$

$\mathbf{px}:=\mathrm{seq}(\mathbf{pts}[i,1],i,1,\mathbf{pn})$

$\mathbf{py}:=\mathrm{seq}(\mathbf{pts}[i,2],i,1,\mathbf{pn})$

Scatter diagram can be plotted in a Graphs App.

Quadratic extrapolation in general.

We generate the two extrapolation parabolas at left and right end of the diagram.

(Both parabolas will degenerate to horizontal likes in our case.)

$\mathbf{q}(x):=\mathbf{q0}+\mathbf{q1}\cdot x+\mathbf{q2}\cdot x^2$

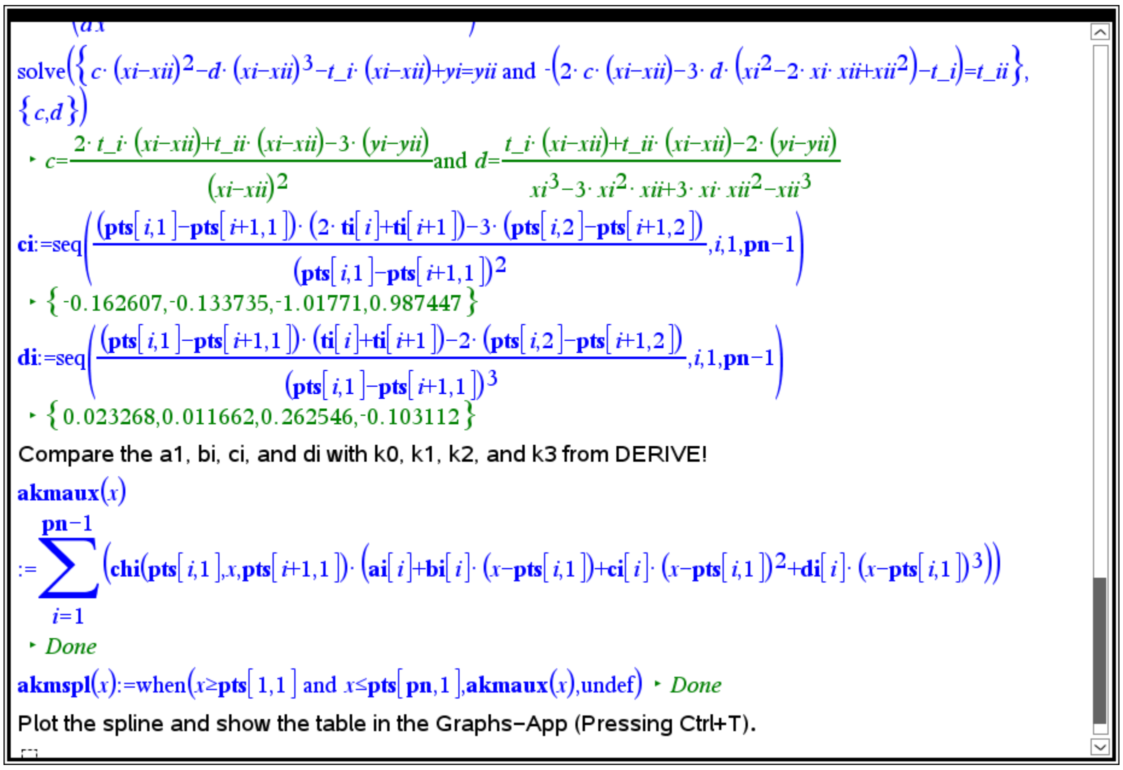
$\mathrm{solve}(\{\mathbf{q}(p1x)=p1y \text{ and } \mathbf{q}(p2x)=p2y \text{ and } \mathbf{q}(p3x)=p3y\},\{\mathbf{q0},\mathbf{q1},\mathbf{q2}\})$

$\mathbf{l1}:=\mathrm{seq}(\mathbf{pts}[i,1],i,1,3)$

$\mathbf{l2}:=\mathrm{seq}(\mathbf{pts}[i,2],i,1,3)$

TI-92, V200 and TI-Nspire, they all don't provide DERIVE's CHI-function. Michel Beaudin does: he offers chi(a,x,b) in his toolbox. It can easily be defined in the calculator:

$$chi(a,x,b):=\frac{\mathrm{sign}(x-a)-\mathrm{sign}(x-b)}{2} \qquad \textit{Done}$$



$\mathrm{solve}(\{c\cdot(xi-xii)^2-d\cdot(xi-xii)^3-t\_i\cdot(xi-xii)+yi=yii \text{ and } -(2\cdot c\cdot(xi-xii)-3\cdot d\cdot(xi^2-2\cdot xi\cdot xii+xii^2)-t\_i)=t\_ii\},$
$\{c,d\})$

$\blacktriangleright\ c=\dfrac{2\cdot t\_i\cdot(xi-xii)+t\_ii\cdot(xi-xii)-3\cdot(yi-yii)}{(xi-xii)^2} \text{ and } d=\dfrac{t\_i\cdot(xi-xii)+t\_ii\cdot(xi-xii)-2\cdot(yi-yii)}{xi^3-3\cdot xi^2\cdot xii+3\cdot xi\cdot xii^2-xii^3}$

$\mathbf{ci}:=\mathrm{seq}\left(\dfrac{(\mathbf{pts}[i,1]-\mathbf{pts}[i+1,1])\cdot(2\cdot\mathbf{ti}[i]+\mathbf{ti}[i+1])-3\cdot(\mathbf{pts}[i,2]-\mathbf{pts}[i+1,2])}{(\mathbf{pts}[i,1]-\mathbf{pts}[i+1,1])^2},i,1,\mathbf{pn}-1\right)$

$\blacktriangleright\ \{^-0.162607,^-0.133735,^-1.01771,0.987447\}$

$\mathbf{di}:=\mathrm{seq}\left(\dfrac{(\mathbf{pts}[i,1]-\mathbf{pts}[i+1,1])\cdot(\mathbf{ti}[i]+\mathbf{ti}[i+1])-2\cdot(\mathbf{pts}[i,2]-\mathbf{pts}[i+1,2])}{(\mathbf{pts}[i,1]-\mathbf{pts}[i+1,1])^3},i,1,\mathbf{pn}-1\right)$

$\blacktriangleright\ \{0.023268,0.011662,0.262546,^-0.103112\}$

Compare the a1, bi, ci, and di with k0, k1, k2, and k3 from DERIVE!

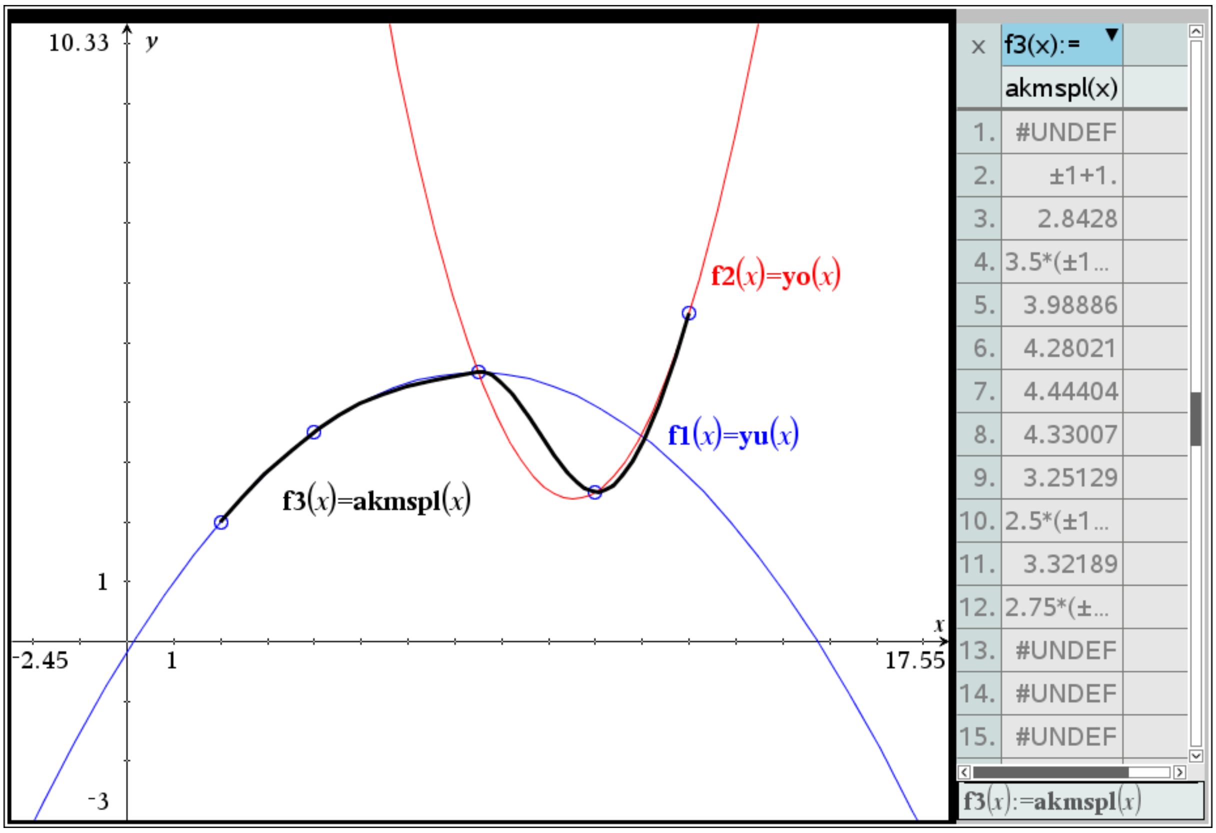
$$\mathbf{akmaux}(x):=\sum_{i=1}^{\mathbf{pn}-1}\left(\mathbf{chi}(\mathbf{pts}[i,1],x,\mathbf{pts}[i+1,1])\cdot(\mathbf{ai}[i]+\mathbf{bi}[i]\cdot(x-\mathbf{pts}[i,1])+\mathbf{ci}[i]\cdot(x-\mathbf{pts}[i,1])^2+\mathbf{di}[i]\cdot(x-\mathbf{pts}[i,1])^3)\right)$$

$\blacktriangleright$ *Done*

$\mathbf{akmspl}(x):=\mathrm{when}(x\geq\mathbf{pts}[1,1] \text{ and } x\leq\mathbf{pts}[\mathbf{pn},1],\mathbf{akmaux}(x),\mathrm{undef})$ $\blacktriangleright$ *Done*

Plot the spline and show the table in the Graphs–App (Pressing Ctrl+T).

You can hide the output (first screenshot) and then step by step perform the procedure (second screenshot). All intermediate results are presented (compare with DERIVE!).



| | f3(x):= akmspl(x) |
|---|---|
| 1. | #UNDEF |
| 2. | ±1+1. |
| 3. | 2.8428 |
| 4. | 3.5*(±1… |
| 5. | 3.98886 |
| 6. | 4.28021 |
| 7. | 4.44404 |
| 8. | 4.33007 |
| 9. | 3.25129 |
| 10. | 2.5*(±1… |
| 11. | 3.32189 |
| 12. | 2.75*(±… |
| 13. | #UNDEF |
| 14. | #UNDEF |
| 15. | #UNDEF |

f3(x):=akmspl(x)

Just enter another pointlist ptn, run through the procedure and you have another example:

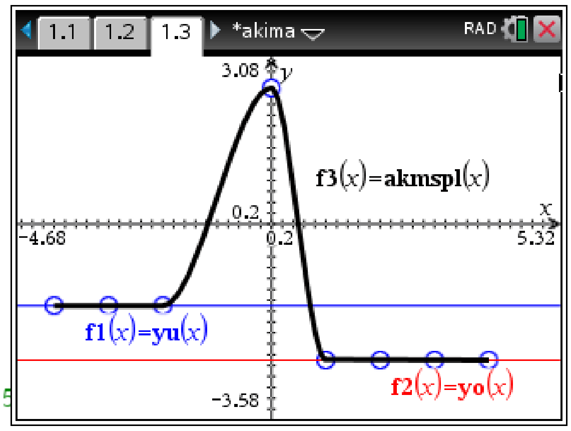
Enter the matrix of the given points and store as pts.

$$\mathbf{pts}:=\begin{bmatrix}-4 & -1.5\\ -3 & -1.5\\ -2 & -1.5\\ 0 & 2.5\\ 1 & -2.5\\ 2 & -2.5\\ 3 & -2.5\\ 4 & -2.5\end{bmatrix} \blacktriangleright \begin{bmatrix}-4 & -1.5\\ -3 & -1.5\\ -2 & -1.5\\ 0 & 2.5\\ 1 & -2.5\\ 2 & -2.5\\ 3 & -2.5\\ 4 & -2.5\end{bmatrix}$$

$\mathbf{pn}:=\dim(\mathbf{pts})[1] \blacktriangleright 8$

$\mathbf{px}:=\text{seq}(\mathbf{pts}[i,1],i,1,\mathbf{pn}) \blacktriangleright \{-4,-3,-2,0,1,2,3,4\}$

$\mathbf{py}:=\text{seq}(\mathbf{pts}[i,2],i,1,\mathbf{pn}) \blacktriangleright \{-1.5,-1.5,-1.5,2.5,-2.5$



[1] Akima-Interpolation, Dirk Hilberg, c't 1989, Heft 6.
[2] Visual Mathematics Illustrated by the TI-92 and the TI-89, G.C.Dorner ao, Springer 2000.
[3] Numerical Analysis via DERIVE, Steven Schonefeld, Mathware 1994
[4] DERIVE Newsletter #18 & #19

DERIVE 5 provides a lot of interesting tools for programming. Josef Lechner (together with Eugenio Roanes and Johann Wiesenbauer) produced a DERIVE-Logo-implementation some times ago. Working as a Beta-Tester Josef improved his DERIVE-Turtle applying the new features. As there is not so much written about programming with DERIVE in the official manual this file might give some insight in this very new DERIVE feature - also presented as a DERIVE worksheet. Josef.
(The file runs in DERIVE 6 without any problems. MTH38.zip contains a version for DERIVE 5 and DERIVE 6 as well.)

## Logo - Implementation for DfW 5.0
## (c) Josef Lechner

Using ideas from DNL#25, p 15 - 20

Implementation of basic commands:

```
new     -->  sets the Logo-basics
home    -->  sets turtle in origin
fd(l)   -->  forward by length l
bk(l)   -->  back by length l
lt(a)   -->  turn left by angle a
rt(a)   -->  turn rigth by angle a
The turtle's trace is given in the plotable list trace_
```

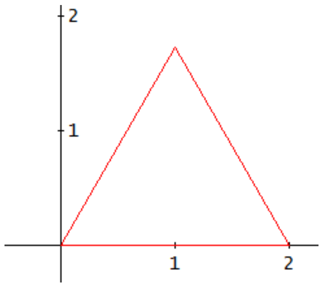
```
#1:   [InputMode := Word, Angle := Degree, Precision := Approximate]

#2:   new := PROG(xcor_ := 0, ycor_ := 0, heading_ := 0, trace_ := [[0, 0]], RETURN trace_)

#3:   home := PROG(x_cor := 0, ycor_ := 0, heading_ := 0, trace_ := APPEND(trace_, [[xcor_,
        ycor_]]), RETURN trace_)

      fd(l) :=
        Prog
          xcor_ := xcor_ + l·COS(heading_°)
#4:       ycor_ := ycor_ + l·SIN(heading_°)
          trace_ := APPEND(trace_, [[xcor_, ycor_]])
          RETURN trace_

#5:   bk(l) := fd(-l)

      lt(a) :=
        Prog
#6:       heading_ := MOD(heading_ + a, 360)
          RETURN trace_

#7:   rt(a) := lt(-a)
```

Before plotting one has to switch on the "Approximate Before Plotting"-Option in the Plot Window.

```
      Block commands := commands
#8:                                 DIM(commands)
```

**Example 1: A linear program**

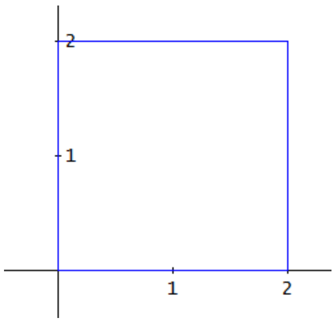
```
#9:                                new = [[0, 0]]

#10:  Block(fd(2), lt(120), fd(2), lt(120), fd(2))
```



Don't forget to simplify new= to set the turtle into the origin

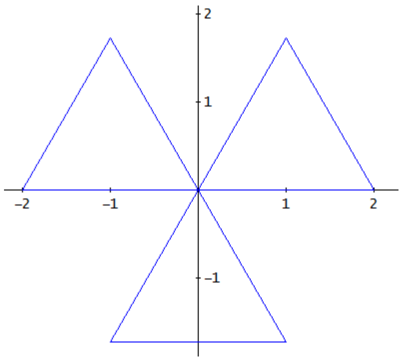
**Example 2: A loop**

```
       last(list) := list
#11:                     DIM(list)

#12:   repeat(n, c) := last(VECTOR(c, i, 1, n))


#13:                                          new = [[0, 0]]

#14:   repeat(4, 'Block(fd(2), lt(90)))
```



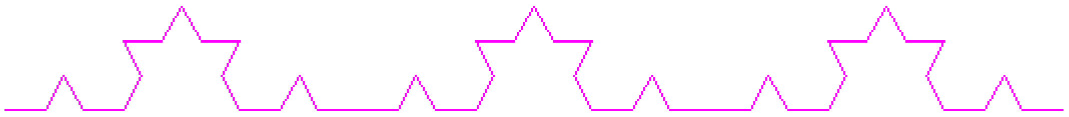
**Example 3: A nested loop**

```
#15:                             new = [[0, 0]]

#16:  repeat(3, 'Block(repeat(3, 'Block(fd(2), lt(120))), rt(120)))
```



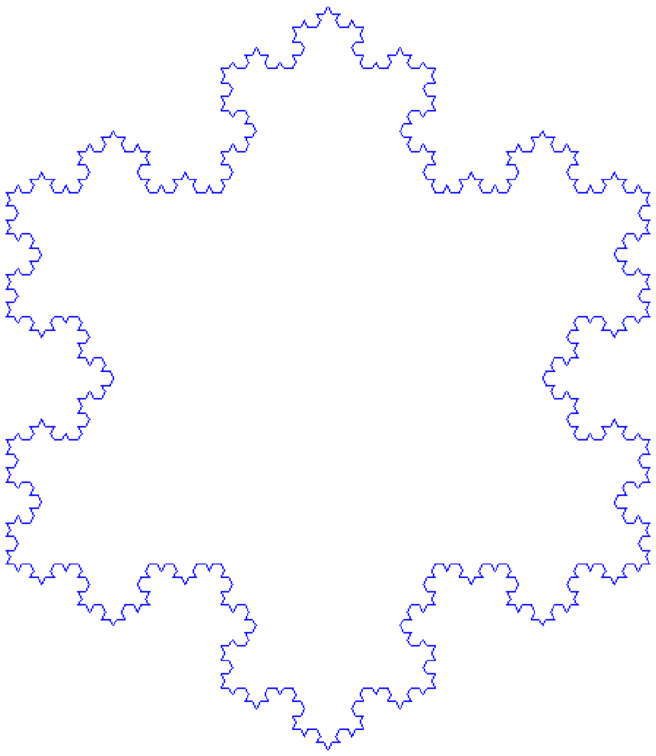
**Example 4: Recursion**

```
          koch(s, t) :=
            Prog
              If t = 0
                 RETURN fd(s)
              koch(s/3, t - 1)
#17:          lt(60)
              koch(s/3, t - 1)
              rt(120)
              koch(s/3, t - 1)
              lt(60)
              koch(s/3, t - 1)

#18:                              new = [[0, 0]]

#19:      koch(1, 2)
```



The result of three times plotting #19 (without simplifying new).

```
#20:   kochstar(s, t) := repeat(3, 'Block(koch(s, t), rt(120)))

#21:                              new = [[0, 0]]

#22:   kochstar(1, 3)
```



# Cross Coordinates - Undocumented?

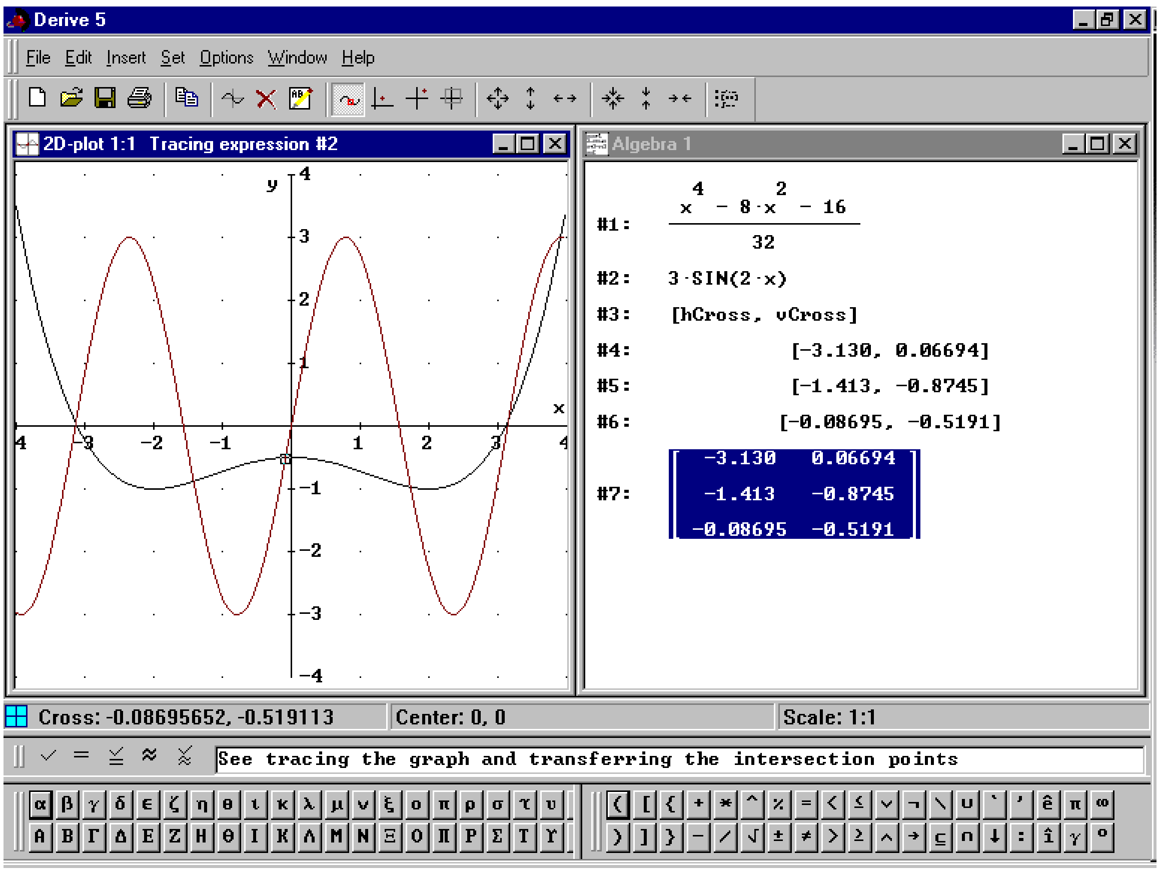
Mr Wickmann from Aachen asked for the possibility to transfer the cross coordinates from the Plot Window to the Algebra Window (eg to create a matrix). I was sure that DfW4 does not offer this feature and hoped that DfW5 would have a hidden undocumented function to do that. So I wrote to Theresa Shelby.

This is her answer:

Yes, in the 2D-plot window the variables hCross and vCross are automatically assigned to the last horizontal (x) and vertical (y) position. Go to the Algebra Window and simplify or approximate them to see their values.

Additionally in the 3D-Plot Window, there are variables xCross, yCross, and zCross which are assigned to the mesh lines and z value when tracing a 3D-plot.

I´m not sure if these variables made it into any documentation.
Best regards, Theresa.


Derive 5
File Edit Insert Set Options Window Help
2D-plot 1:1 Tracing expression #2
Algebra 1

#1: $\dfrac{x^4 - 8\cdot x^2 - 16}{32}$

#2: 3·SIN(2·x)

#3: [hCross, vCross]

#4: [-3.130, 0.06694]

#5: [-1.413, -0.8745]

#6: [-0.08695, -0.5191]

#7: $\begin{bmatrix} -3.130 & 0.06694 \\ -1.413 & -0.8745 \\ -0.08695 & -0.5191 \end{bmatrix}$

Cross: -0.08695652, -0.519113 | Center: 0, 0 | Scale: 1:1

See tracing the graph and transferring the intersection points

*On 20 May I was invited to give a talk at the 1st Belgian Congress on Information Technology in Maths Education. This Conference was organized in Brussels by Jaqueline Sengier and Dirk Janssens. They brought together the French and the Dutch speaking Maths teachers within the frame of T^3 (Teachers Teaching with Technology). The meeting became a success thanks the efforts of Jaqueline and Dirk and their staff members, many thanks to you all.*
*At this occasion Wolfram Koepf gave a plenary talk, which was quite interesting. I asked Wolfram and the organizers for publishing his lecture in the DNL. They all agreed and here we "proudly present":*

# Mathematics with DERIVE as Didactical Tool
# Plenary Talk

**Wolfram Koepf**
Univerity Kassel
koepf@mathematik.uni-kasseö.de

## 1 Introduction

In this talk, I would like to show you how DERIVE can be used as a didactical tool in secondary high school education. For my demonstration I use the newest release DERIVE for Windows 5 which supports the use of worksheets. I will concentrate on some examples describing general topics that I find most appropriate to enhance our educational opportunities. In this introduction, I would like to embed the sooner or later expected replacement of numerical calculators in mathematics education by symbolic ones into a historical context.

In the history of mathematics from time to time new calculation tools were invented replacing old ones. Obviously the known mathematical calculation techniques were quite different at the time before logarithm tables made the computation of products and powers easier, and the traditional techniques changed when these methods were generally accessible. Similarly, when the slide-rule came up, the used—and taught—calculation techniques changed again.

More dramatically, when numerical calculators came into the scene, they made both the logarithm tables and the slide-rule dispensable, and, indeed in our days students do no longer have the knowledge about these former mighty tools,[1] let alone their use.

As an example, I have experienced that since the numerical calculator was introduced into the classroom, students tend to avoid working with rational arithmetic. They prefer to use decimal numbers instead. Even if these computations might give inaccurate results in certain situations (e.g. 0.999999999 rather than 1), this is one of the expected results of the use of the new tool, whether we like it or not. The students don't find this an essential disadvantage. Whenever mathematical tools are replaced by new ones, teaching and exercising changes. Obviously we should be aware of this fact, and we should try to make sure that important knowledge and techniques do not get lost.

[1] Ask your students whether they can tell you what a slide-rule is!

In my opinion, computer algebra tools are more adequate for mathematics education than numerical calculators are. Whereas a numerical calculator gives some new possibilities like the fast computation of trigonometric functions etc., it cannot really replace the former computations as the simple example of rational arithmetic shows. Hence, one can expect positive side effects when computer algebra comes into the game.

The examples of this article are mainly taken from my book [3] which offers material for the use of DERIVE in the secondary education at German high schools. I hope I can convince you that save the large personal involvement of each teacher, there are great chances with the use of computer algebra in the classroom.

## 2 Probability

Computer algebra systems should be used to work with large integers. As already mentioned, since the numerical calculator was introduced into the classroom, students tend to avoid working with rational arithmetic. Rational arithmetic comes back into the picture when using computer algebra.

Here I give an example from probability. The point is that such computations are out of reach for hand computations although they form interesting and by no means pathological questions. It is a typical situation that realistic "real world problems" are not generally accessible for hand computation.

One can use numeric computation for such problems, however the student has no chance to do the necessary error analysis. With some calculators, the result may be even wrong since large factorials are involved.[2] Using exact rational arithmetic, errors do not occur at all. Nevertheless all results presented in this lecture can be computed in a second on today's PC's.

**Problem 1:**
*What is the probability $P_1$ to throw exactly 50 times Head when throwing a fair coin 100 times?*

Clearly, the teacher should develop the theory behind such questions, and the binomial distribution will show up. Hence the result is given by the formula

$$P_1 = \binom{100}{50} \cdot \left(\frac{1}{2}\right)^{100} ,$$

$\binom{n}{k}$ denoting the binomial coefficient. In DERIVE, this can be entered as

```
COMB(100, 50) * (1/2)^100
```

[2] Indeed, large factorials are only involved if a numerical calculator does not support products. In a computer algebra system binomial coefficients are computed differently and more efficiently.

and simplification yields

$$P_1 = \frac{12611418068195524166851562157}{158456325028528675187087900672},$$

a rational number with huge numerator and denominator. Such a result is not accessible with a numerical calculator. On the other hand, in DERIVE we can now approximate the previous exact result and get

$$P_1 = 0.07958923738 ,$$

clearly accurate in all its decimal places, and no error analysis is necessary. To interpret the result, it might be unexpected that in almost 8 % of all repetitions of the experiment to throw a coin 100 times *exactly* 50 times Head occurs. Clearly this provokes an even more complicated question like

**Problem 2:**
*What is the probability $P_2$ to throw between 45 and 55 times Head when throwing a fair coin 100 times?*

The binomial distribution gives the sum

$$P_2 = \sum_{k=45}^{55} \binom{100}{k} \cdot \left(\frac{1}{2}\right)^{100} ,$$

which is in DERIVE

```
SUM(COMB(100, k) * (1/2)^100, k, 45, 55) .
```

DERIVE computes very fast

$$P_2 = \frac{28868641920228451421269389993}{39614081257132168796771975168} .$$

Clearly, such a computation is absolutely intractable by hand computation! The decimal analogue of the result is

$$P_2 = 0.7287469759 .$$

Hence in more than 2/3 of all repetitions of the experiment to throw a coin 100 times we get between 45 and 55 times Head.

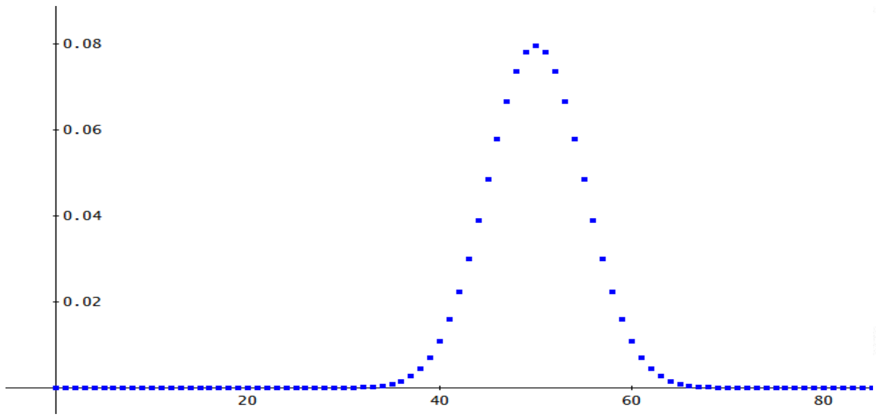
This gives a hint how "wide" the underlying binomial distribution is. To measure its width, we compute the standard deviation $\sigma = \sqrt{np(1-p)}$,

```
σ := SQRT(100 * (1/2) * (1/2)) ,
```

to obtain $\sigma = 5$. This corresponds to the well-known fact that in about 2/3 of all cases the result of a normally-distributed random variable lies in the 1-$\sigma$ neighborhood $[\mu - \sigma, \mu + \sigma]$ of the expected value $\mu$, which in our case is $\mu = 50$.

One could also easily check the 2-$\sigma$ and 3-$\sigma$ neighborhoods of the expected value which should contain about 95 % and 99 % of the results, respectively, therefore proving that the binomial distribution is already "almost normal" if $n = 100$.

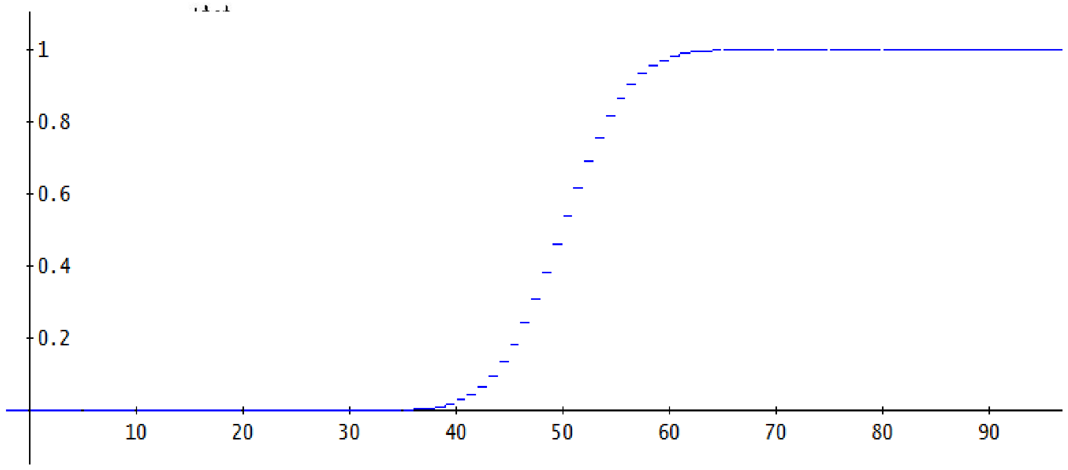
Another hint for the width of the binomial distribution as well as for its similarity to the normal distribution is given by its graph



which we receive after simplifying

```
VECTOR([k, COMB(100, k) * (1/2)^100], k, 0, 100)
```

Next, we generate a picture of the distribution function of the binomial distribution



```
SUM(COMB(100, k) * (1/2)^100 * CHI(k, x, ∞), k, 0, 100)
```

to be compared with the distribution function of the corresponding Gaussian normal distribution

$$N(\mu, \sigma, x) := \int_{-\infty}^{x} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \ .$$

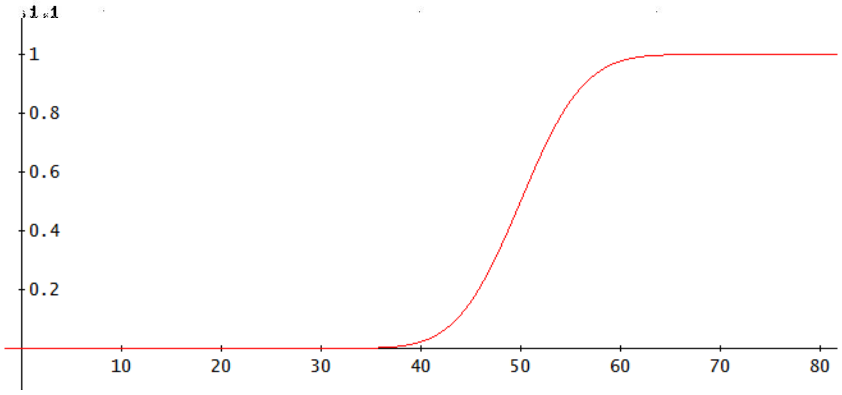
A plot of the simplified expression

$\mathtt{INT}(1/(\sigma * \mathtt{SQRT}(2 * \pi)) * \mathtt{EXP}(-(x-50)\hat{}2/(2*\sigma\hat{}2)), -\infty, x)$

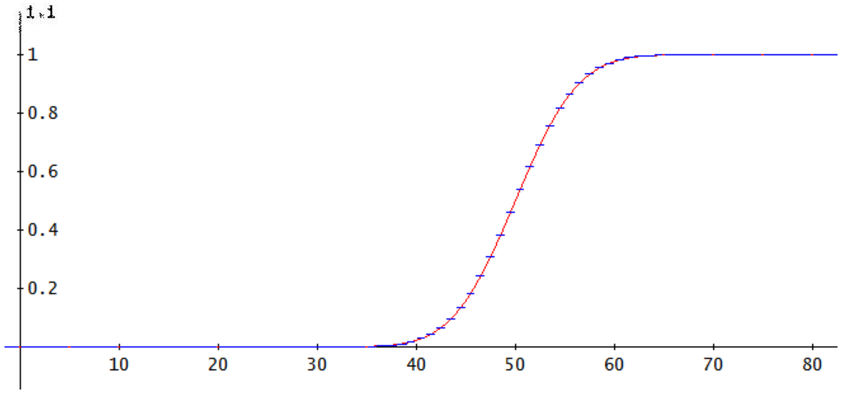
involving the error function

$$\frac{\operatorname{erf}\left(\frac{\sqrt{2}x}{10} - 5\sqrt{2}\right) + 1}{2}$$

yields



The superposition of both functions is most impressive:



Obviously we are free – and DERIVE offers the capability – to study, e.g., also the situation for $n = 1000$.

## 3 Graphics

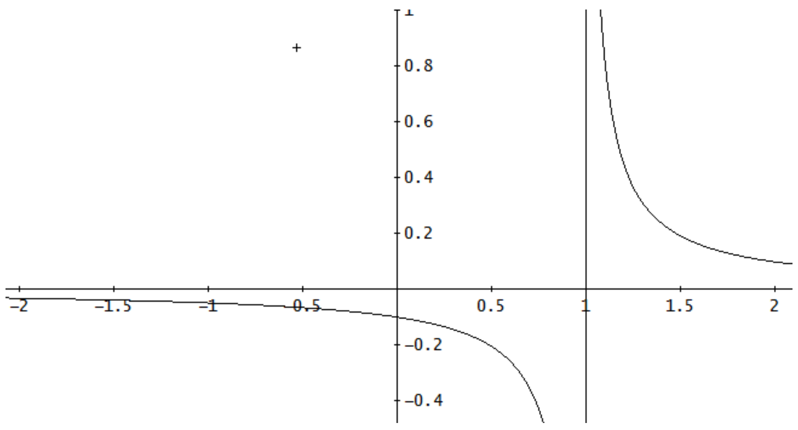
We already saw in the above section, that graphs can give very interesting insights into mathematical situations. On the other hand, graphing is not that easy in all situations as the following example shows [4]:

**Problem 3:**
*Graph the rational function* $f(x) = \frac{1000\,(x-1)}{(101x-100)\,(100x-99)}$. *Where is the second pole?*

We enter

$f := 1000 * (x - 1)/((101 * x - 100) * (100 * x - 99))$ .

DERIVE immediately graphs $f$ as



It looks as if there is a maximum just right of 1, a zero at 1, and a pole just left of 1. Actually, this picture is not bad compared to the results of other computer algebra software. But nevertheless: A glimpse at the formula shows that there should be two poles. Where is the second pole? Even zooming out does not give any hint, so what is going on?

Clearly, directly from the formula we see that the two poles are at $x = \frac{100}{101}$ and $x = \frac{99}{100}$, both slightly smaller than 1. To find out how the graph between the two poles looks, we search for the critical points by

SOLVE(DIF($f, x$), $x$)

and get the result

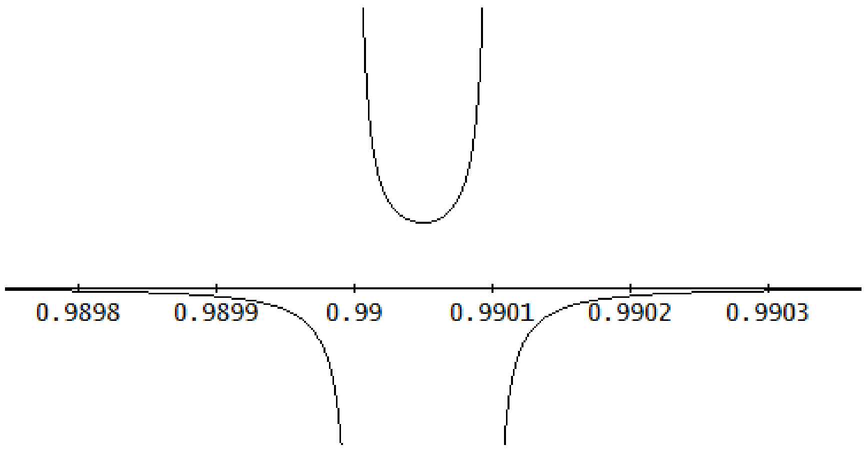
$$x = \pm\infty \quad \text{or} \quad x = 1 - \frac{\sqrt{101}}{1010} \quad \text{or} \quad x = \frac{\sqrt{101}}{1010} + 1 .$$

The last solution is obviously larger than 1, hence yields the maximum, so the middle solution is the one we are interested in. Substituting this number into $f$

SUBST($f, x, 1 -$ SQRT(101)/1010)

yields the value $20000\sqrt{101} + 201000 \approx 4.019975124 \cdot 10^5$. Now we see what's happening: The function has a local minimum at $x = 1 - \frac{\sqrt{101}}{1010}$ and all its values are larger than $4 \cdot 10^5$ between the two poles. That's why we didn't see anything! Using a reasonable plot range[3] gives the following graph:

[3]using Set: Plot Range in the 2D-Plot window



This example shows that the use of computer algebra enhanced graphing is much superior to the use of a graphics calculator with which such information is not available.

## 4 Factorization

In this section we deal with a question which is out of reach for hand computation as well as for graphics calculators. Here, symbolic computation is the essential ingredient.

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. Let's check this with DERIVE. We enter

$g := 1/(1+x^4)$

and use DERIVE to factorize $g$ by

FACTOR($g$)

to get

$$1+x^4$$

again. This seems to approve Leibniz's opinion. However, we get more detailed information when using DERIVE's `Simplify` menu. The `Simplify: Factor` submenu asks the user for the degree of factorization: Trivial, Rational, Radical or Complex, Rational being the default choice. This gives us a clue why a proper factorization was not found in the first step: A factorization of $g$ over the rationals $\mathbb{Q}$ does not exist![4] However, since we search for a factorization over $\mathbb{R}$, we choose the Radical option to get

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

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

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

with `Simplify: Expand`, and to understand the antiderivative

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

which is returned by DERIVE's integrator.

## 5 Parametric Representation of Conic Sections

Students of Physics need the parametric representation of conic sections when planetary motion is studied. University teachers in Germany observe and criticize that a typical student does not have this knowledge. The reason is simple: With hand computations the parametric representation of conic sections is rather difficult to derive. Here, we show how a rational factorization gives this result very easily. Note that although rational factorizations are difficult to find, they are very easy to check.

We start with the rectangular equation of an ellipse whose left focus lies in the origin:

$$(x-e)^2/a^2+y^2/b^2-1=0\ .$$

We replace $b^2$ by $a^2-e^2$ using the substitution

$$(x-e)^2/a^2+y^2/\mathtt{SQRT}(a^2-e^2)^2-1=0\ .$$

Simplification yields

$$\frac{x^2\,(a^2-e^2)+2\,e\,x\,(e^2-a^2)+a^2\,y^2-(a^2-e^2)^2}{a^2\,(a^2-e^2)}=0\ .$$

[4] Actually this was *proved* by DERIVE in the first calculation!

Next, we introduce polar coordinates $x = r\cos\phi$ and $y = r\sin\phi$ by another substitution. Simplification gives

$$\texttt{equation} := \frac{e^2\,r^2\,\mathrm{COS}(\phi)^2 + 2\,e\,r\,(a^2 - e^2)\,\mathrm{COS}(\phi) + a^4 - a^2\,(2\,e^2 + r^2) + e^4}{a^2\,(e^2 - a^2)} = 0\,.$$

You see that the simplification procedure has removed the sine expression using the trigonometric identity $\sin^2\phi = 1 - \cos^2\phi$. But what's next?

What we want is a representation of $r$ in terms of $\phi$. It turns out that students who don't have any idea how to proceed, but are accustomed with DERIVE, nevertheless have the idea to let DERIVE solve the equation for $r$, and this is a successful approach! On the other hand the main reason for this success is hidden. Let's rather try to factorize the given expression by

```
FACTOR(equation)
```

with the result

$$\frac{(e\,r\,\mathrm{COS}(\phi) + a^2 + a\,r - e^2)\,(e\,r\,\mathrm{COS}(\phi) + a^2 - a\,r - e^2)}{a^2\,(a + e)\,(e - a)} = 0\,.$$

It is now obvious why the equation easily can be solved for $r$: By the factorization, either of the two numerator factors must be zero. Using

```
SOLVE(equation, r)
```

returns these two solutions

$$r = \frac{e^2 - a^2}{e\,\mathrm{COS}(\phi) - a} \quad \text{or} \quad r = \frac{e^2 - a^2}{e\,\mathrm{COS}(\phi) + a}\,.$$

It turns out that only one of these is positive, hence we get

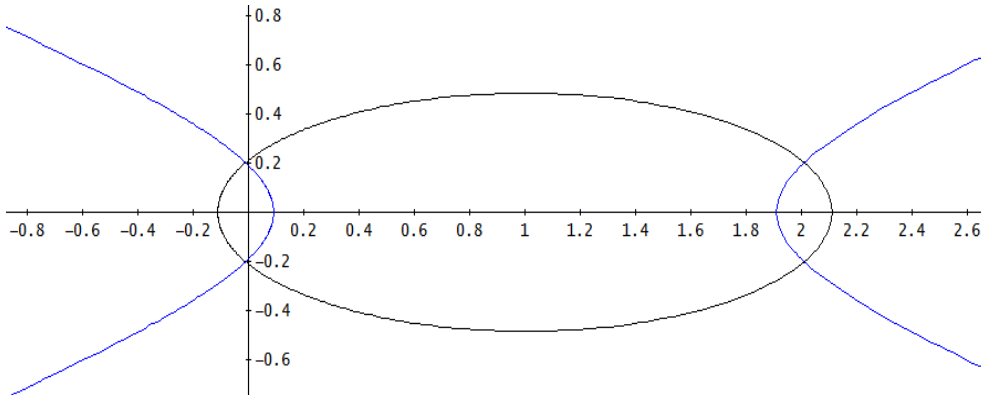
$$r = \frac{e^2 - a^2}{e\,\mathrm{COS}(\phi) - a}\,.$$

Finally, we introduce the eccentricity $\varepsilon = e/a$ which is smaller than 1 for an ellipse, larger than 1 for a hyperbola and equal to 1 for a parabola, and obtain

$$r = \frac{e\,(\varepsilon^2 - 1)}{\varepsilon\,(\varepsilon\,\mathrm{COS}(\phi) - 1)}\,.$$

It turns out ([3], Chapter 2) that a similar computation yields the same result for a hyperbola, for which, however, $\varepsilon > 1$.

To get a picture, we produce a graph for $e = 1$, $\varepsilon = 0.9$ and $\varepsilon = 1.1$, respectively.[5] This yields



## 6 Cubic Newton Iteration

Let's start to derive the usual Newton iteration by linearization. Declare $f$ by

$f(x) :=$

to be an arbitrary function of a single variable. Linearizing $f$ at the point $x_n$ gives the equation

$y = f(\mathrm{xn}) + (x - \mathrm{xn}) * f'(\mathrm{xn})$ .

If $x_n$ is an approximation of a zero, we therefore can set $y = 0$, and solve for $x$ to get the next iteration value:

SOLVE$(0 = f(\mathrm{xn}) + (x - \mathrm{xn}) * f'(\mathrm{xn}), x)$

with the solution

$$x = x_n - \frac{f(x_n)}{f'(x_n)} .$$

This gives the well-known Newton iteration formula for $x_{n+1}$ which usually converges quadratic to a simple zero of $f$ (see e.g. [2] for details).

Now we are ready to develop a cubic interpolation scheme (see e.g. [3], Chapter 3) by starting with a second order parabolic approximation of $f$:[6]

[5] To generate a polar plot, one uses `Set: Coordinate System: Polar` in the 2D-Plot window.
[6] Instead of using the TAYLOR function, one could also enter the expression by hand.

$y = \texttt{TAYLOR}(f(x), x, \texttt{xn}, 2)$

which simplifies to

$$y = \frac{(x - x_n)^2 \, f''(x_n)}{2} + (x - x_n) \, f'(x_n) + f(x_n) \; .$$

Since $x - x_n \approx -\frac{f(x_n)}{f'(x_n)}$ by the Newton iteration, one can linearize the quadratic factor. Setting $y = 0$ and solving for $x$, again, we get

$\texttt{equation} := \texttt{SOLVE}(0 = (x - \texttt{xn}) * (-f'(\texttt{xn})/f(\texttt{xn}))/2 + (x - \texttt{xn}) * f'(\texttt{xn}) + f(\texttt{xn}), x)$

with the solution

$$x = \frac{2 \, f(x_n) \, f'(x_n)}{f(x_n) \, f''(x_n) - 2 \, f'(x_n)^2} + x_n \; .$$

This is a cubic iteration scheme as we shall see. Let's test it to calculate square roots. To compute $\sqrt{a}$, we use

$f(x) := x^2 - a \; .$

With this definition, one can simplify `equation` again to obtain

$$x = 3 \, x_n - \frac{8 \, x_n^3}{a + 3 \, x_n^2} \; .$$

Let's check that this gives a cubic iteration scheme: Simplifying $x_{n+1} - a$ by

$\texttt{FACTOR}((3 * \texttt{xn} - 8 * \texttt{xn}^3/(a + 3 * \texttt{xn}^2))^2 - a)$

gives

$$\frac{(x_n^2 - a)^3}{(a + 3 \, x_n^2)^2}$$

proving the cubic convergence. The same method shows that Newton's iteration is quadratic convergent.

Now, we test the iteration for

$a := 10 \; .$

The iteration can be executed by the command

$$\texttt{ITERATES}(3 * \mathrm{xn} - 8 * \mathrm{xn}^3/(a + 3 * \mathrm{xn}^2), \mathrm{xn}, 1, 5)$$

which we approximate to get

$$[1, 2.384615384, 3.144897306, 3.162277527, 3.16227766, 3.16227766]\,.$$

Hence $\sqrt{10}$ is computed from the (bad) initial guess 1 in 4 steps, whereas Newton's iteration

$$\texttt{ITERATES}(\mathrm{xn} - f(\mathrm{xn})/f'(\mathrm{xn}), \mathrm{xn}, 1, 5)$$

gives

$$[1, 5.5, 3.659090909, 3.196005081, 3.162455622, 3.162277665]$$

convergence in five steps. From the practical point of view this might be not very essential, but from a theoretical point of view the higher order Newton method is rather interesting.

Note that by a similar computation iteration schemes with convergence of arbitrary order can be obtained which however--as a negative computational repercussion get more and more complicated. Note further that the cubic iteration scheme for the square root discussed in this section was discovered by Dedekind (see e.g. [5], pp. 99).

## 7 Final Remarks

I hope that I could convince you that the smart use of computer algebra in the classroom can be a very convenient didactical tool to teach and develop mathematical insights. On the other hand, one should not disregard the fact that nevertheless the students should obtain advanced mathematical skills before using computer algebra since otherwise they might neither be able to handle such a program nor to understand its results. A similar remark applies to hand-held computer algebra tools like the TI-89. Therefore, it is very important that the essential mathematical skills are still practiced.

Especially if you have weak students, it may happen that this scheme cannot be put into practice. One essential problem are time constraints. Furthermore, my experience is that weaker students tend to be overwhelmed by the handling of a computer algebra system[7] since they need to acquire even more skills to use such a program which they try to avoid.

Note that in the meantime many different teachers all over the world use computer algebra in Math education and curriculum discussions are on their way. A collection of interesting projects on this topic can be found in Section 3.8 on "Computer Algebra in Education" which I compiled for [1].

[7]even though the students might enjoy the calculation assistance

## References

[1] Grabmeier, J., Kaltofen, E. and Weispfenning, V. (Eds): *New Reference Book on Computer Algebra*, Springer, Berlin, 2000.

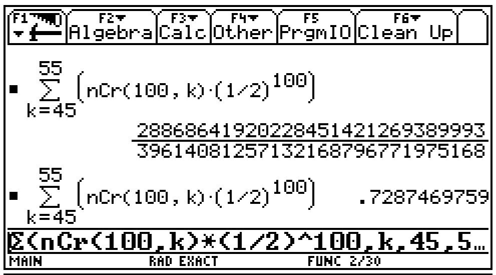
[2] Koepf, W., Ben-Israel, A. and Gilbert, R. P.: *Mathematik mit Derive*, Vieweg, Braunschweig/Wiesbaden, 1993.

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

[4] Koepf, W.: Numeric versus symbolic computation. International Journal of Computer Algebra in Mathematics Education **5** (1), 1998, 29–54.

[5] Schmersau, D. and Koepf, W.: *Die reellen Zahlen als Fundament und Baustein der Analysis.* Oldenbourg, Munich, 2000.

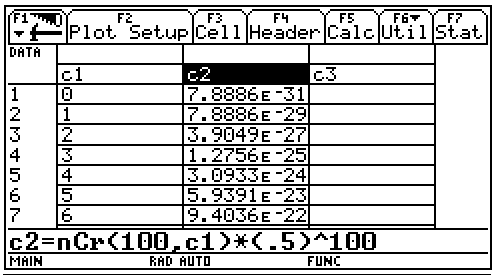
See some TI-92 screen shots to underline the fact, that Wolfram´s ideas do not depend on the chosen CAS and are not bound on the use of the PC, Josef.


F1 | F2 Algebra | F3 Calc | F4 Other | F5 PrgmIO | F6 Clean Up

$$\sum_{k=45}^{55}\left(nCr(100,k)\cdot(1/2)^{100}\right) \quad \frac{28868641920228451421269389993}{39614081257132168796771975168}$$

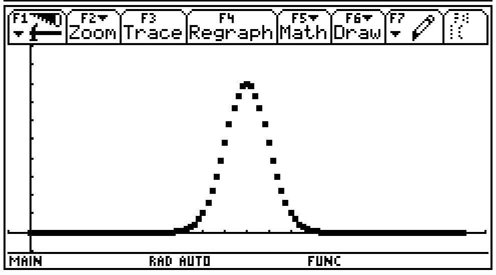
$$\sum_{k=45}^{55}\left(nCr(100,k)\cdot(1/2)^{100}\right) \quad .7287469759$$

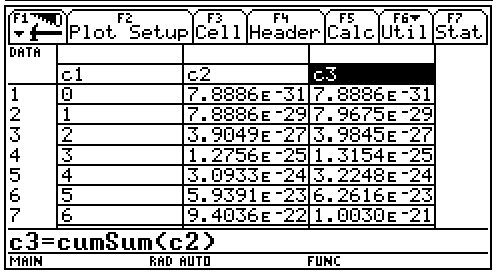
```
Σ(nCr(100,k)*(1/2)^100,k,45,5…
MAIN          RAD EXACT          FUNC 2/30
```


F1 | F2 Plot Setup | F3 Cell | F4 Header | F5 Calc | F6 Util | F7 Stat

| DATA | c1 | c2 | c3 |
|---|---|---|---|
| 1 | 0 | 7.8886E-31 | |
| 2 | 1 | 7.8886E-29 | |
| 3 | 2 | 3.9049E-27 | |
| 4 | 3 | 1.2756E-25 | |
| 5 | 4 | 3.0933E-24 | |
| 6 | 5 | 5.9391E-23 | |
| 7 | 6 | 9.4036E-22 | |

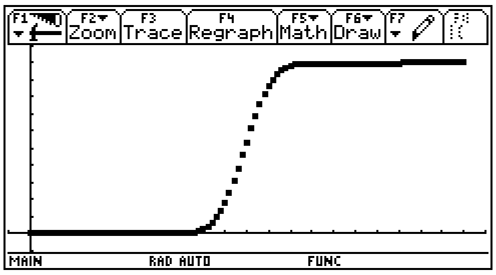
```
c2=nCr(100,c1)*(.5)^100
MAIN          RAD AUTO          FUNC
```

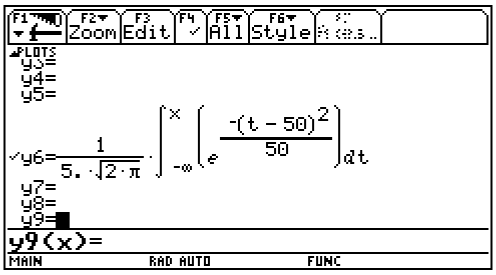



F1 | F2 Plot Setup | F3 Cell | F4 Header | F5 Calc | F6 Util | F7 Stat

| DATA | c1 | c2 | c3 |
|---|---|---|---|
| 1 | 0 | 7.8886E-31 | 7.8886E-31 |
| 2 | 1 | 7.8886E-29 | 7.9675E-29 |
| 3 | 2 | 3.9049E-27 | 3.9845E-27 |
| 4 | 3 | 1.2756E-25 | 1.3154E-25 |
| 5 | 4 | 3.0933E-24 | 3.2248E-24 |
| 6 | 5 | 5.9391E-23 | 6.2616E-23 |
| 7 | 6 | 9.4036E-22 | 1.0030E-21 |

```
c3=cumSum(c2)
MAIN          RAD AUTO          FUNC
```



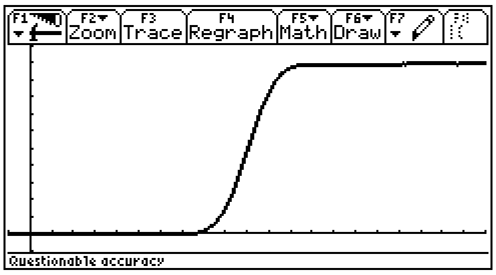


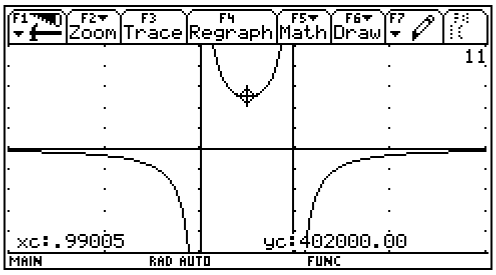
PLOTS
y3=
y4=
y5=

$$\checkmark y6=\frac{1}{5.\cdot\sqrt{2\cdot\pi}}\cdot\int_{-\infty}^{x}\left(e^{\frac{-(t-50)^2}{50}}\right)dt$$

y7=
y8=
y9=

```
y9(x)=
MAIN          RAD AUTO          FUNC
```





# Experiments Using CBL/CBR and the TI-92 – New Conceptions for Teaching Natural Science

Heinz-Dieter Hinkelmann, Korneuburg, Austria

## 1. Introduction

With this title a series had been started to enlarge the possibilities of application of the TI-92 under the aspect of natuaral science. For that means experiments based on CBL™ (Computer Based Laboratory) and CBR™ (Computer Based Ranger), both products of *Texas Instruments*, are described as well as the measuring sensors of CBL-basic equipment and the sensors and probes of the *Vernier* company.

We will show some experiments for the natural science lessons and their mathematical analysis. If anybody has more ideas, suggestions or questions, just please contact me by using the following adress:

heinz-dieter.hinkelmann@telecom.at

The measuring instruments will be shortly described for those, who don´t know them. Today´s edition describes the two of the three probes of basic equipment TI provides you to get started with the CBL and the following editions of DNL will explain the third one and other sensors and probes.

## 2. How do the sensors work?

The mode of sensor working (*Texas Instrument* or *Vernier*) together with the CBL is rather similar: Data of an electric size (voltage, current, or resistance) are measured and transferred to CBL. The so called „AutoIDENT“-feature allows the CBL unit to automatically identify specific probes when you connect them to the CBL channels. It determines what kind of data is going to be measured, and loads an equation for converting the data into the appropriate measurement unit.

The light probe´s output is a voltage that is linearly proportional to the amount of irradiance it senses. The AutoIDENT resistor contained in the probe causes the CBL software to automatically convert the measured voltage to mW/cm$^2$ units. The range of light over which the probe is sensitive is 10 µW/cm$^2$ to 1 mW/cm$^2$. The light probe can be used in the visible and near infrared light range. It is designed to work in air only; it is not waterproof.

The temperature probe measures the resistance of a thermistor. A thermistor is a variable resistor whose resistance decreases nonlinearly with increasing temperature. The best-fit approximation to this nonlinear characteristic is the Steinhart-Hart equation. The AutoIDENT resistor contained in the probe causes the CBL software to automatically convert the measured resistance to °C by the aid of the equation. The temperature probe is sensitive between –20°C to +125°C. It is water resistant (maximal for 24 hours). You can use it with some chemicals which are listed in the handbook.

Of course the voltage probe does not need any transformation equations. The AutoIDENT resistor contained in the probe causes the CBL software to automatically measure voltage. The black hook should be connected to ground and the red hook to the signal voltage.

To perform experiments with the CBL, you will need TI-GRAPH LINK™ cable and software and CBL programs onto your computer. You will use the TI-GRAPH LINK™ cable and software to transfer the CBL programs from the computer to the calculator. We recommend the PHYSICS program set from *Vernier*. You can download it from the *Vernier* web site (www.vernier.com).

## 3. Experiments with the basic probes

### 3.1 Discharging of a capacitor

*When a capacitor discharges, its voltage follows an exponential phenomenon:* $U(t) = U_o.e^{-kt}$

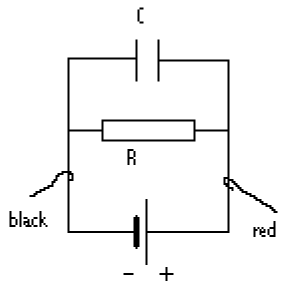
*($U(t)$ ... voltage across the capacitor at any time, $t$ … time, $U_o$ ... capacitor´s initial voltage, $k$ ... constant that depends on the physical characteristics of the capacitor)*

**Equipment required**

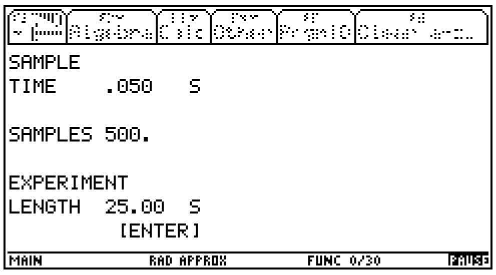
- CBL unit, TI-92 calculator with unit to unit link cable
- TI voltage probe
- Capacitor (220 µF), resistor (100 kΩ), battery (9V)

#

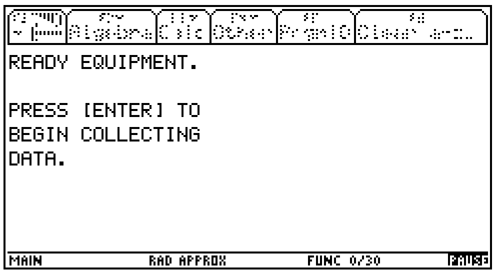
**Instructions**



- ❑ Connect the circuit as shown on the right.
- ❑ Connect CBL and the TI-92 and connect the voltage probe to the CH1 input.
- ❑ Run the PHYSICS program.
- ❑ Select "Set up probes" from the "Main menu"; select "One" as the number of probes; select "Voltage" from the "Select probe menu".
- ❑ Confirm that the voltage probe is connected and press <Enter>.
- ❑ Select "Collect Data" from the "Main menu" and "Time graph" from the "Data collection menu". Sample time should be 0.05 s and number of samples 500. This makes 25 s for discharging the capacitor.
- ❑ Press <Enter> and select "Use time setup" to continue.

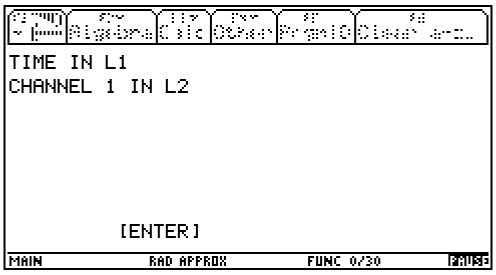


```
SAMPLE
TIME    .050   S

SAMPLES 500.

EXPERIMENT
LENGTH  25.00  S
         [ENTER]
```

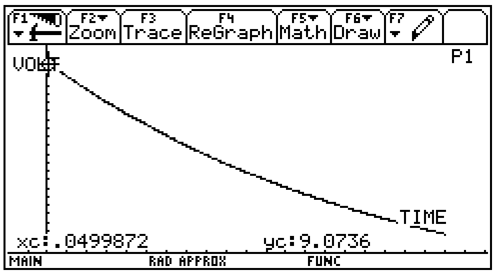


```
READY EQUIPMENT.

PRESS [ENTER] TO
BEGIN COLLECTING
DATA.
```

- ❑ Disconnect the contact to the battery and at the same time press <Enter> to begin data collection.
- ❑ Press <Enter> and note the shape of your graph of potential versus time.

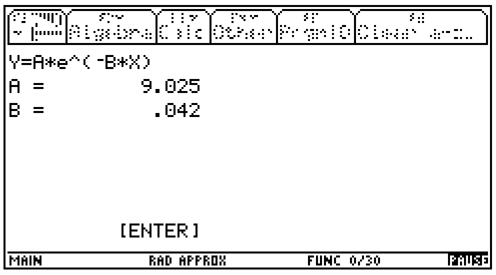


```
TIME IN L1
CHANNEL 1 IN L2

         [ENTER]
```

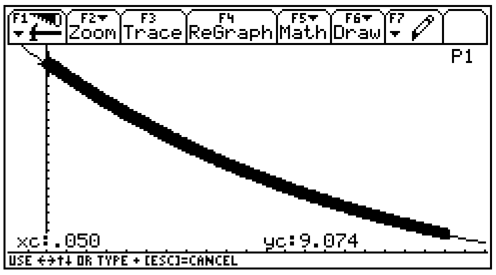


Next fit the exponential function to your data.

- ❑ Press <Enter> and select "No" to return to the "Main menu".
- ❑ Select "Analyze" from the "Main menu"; select "Curve fit" from the "Analyze menu"; select "Exponent L1, L2" from the "Curve fit menu". Press <Enter> to view the fitted equation along with your data.



```
Y=A*e^(-B*X)
A =        9.025
B =         .042

         [ENTER]
```

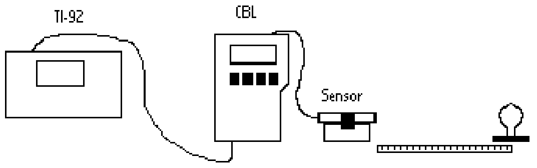


## 3. 2 Irradiance of light

*Irradiance of light is defined as the rate at which energy is transferred per unit area, measured in watts per square meter. As light propagates away from its source, the energy associated with it tends out. In this experiment you will measure irradiance at a variety of distances from a smaller source of light, and see, how the irradiance varies with distance.*

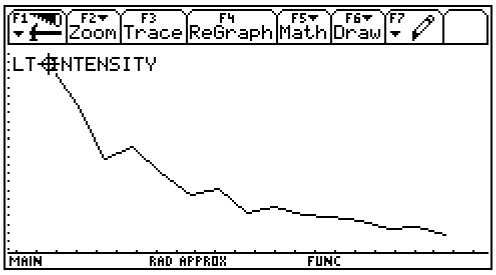
**Equipment required**

- CBL unit, TI-92 calculator with unit to unit link cable
- TI light probe
- Light bulb (25 W)
- Wooden block, tape
- Meter stick



**Instructions**

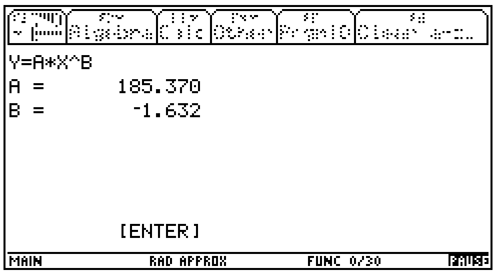
- ❑ Secure the light probe to a wooden block using a tape. The structure of the experiment is shown above.
- ❑ Connect CBL and the TI-92 and connect the light probe to the CH1 input.
- ❑ Run the PHYSICS program.
- ❑ Select "Set up probes" from the "Main menu"; select "One" as the number of probes; select "Light" from the "Select probe menu".
- ❑ Confirm that the light probe is connected and press <Enter>.
- ❑ Select "Collect Data" from the "Main menu" and "Trigger/Prompt" from the "Data collection menu".
- ❑ Place the light sensor 30 cm from the light bulb filament and press <Trigger> on the CBL. Then enter the distance between the light sensor and the light source on the calculator (here: "30"). Press <Enter> to conclude your entry.
- ❑ Select "More data" to collect additional data. Move the light sensor 5 cm farther away from the light source, press <Trigger> and enter "35".
- ❑ Continue in this fashion until 15 data points have been collected. For the final data point select "Stop and graph" instead of "More data".
- ❑ Examine the graph of light intensity (in mW/cm$^2$) versus distance.



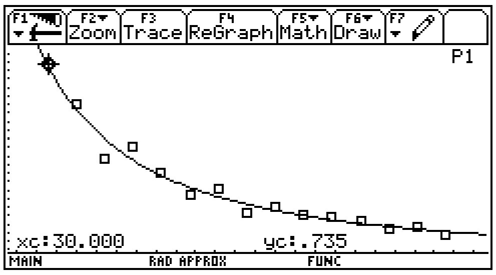
Next fit the power function to your data.

- ❑ Press <Enter> and select "No" to return to the "Main menu".
- ❑ Select "Analyze" from the "Main menu"; select "Curve fit" from the "Analyze menu"; select "Power L1, L2" from the "Curve fit menu". Press <Enter> to view the fitted equation along with your data.

**Note:** The measurement of low light levels is difficult and will often result in a power of between –1.5 and –1.8 rather then the expected –2.



```
Y=A*X^B
A =      185.370
B =       -1.632

        [ENTER]
```



# Titbits from Algebra and Number Theory(17)

by Johann Wiesenbauer, Vienna

Last time I started the discussion of the new programming features of DfW5 by revisiting the Euclidean algorithm, which is certainly one of the most important algorithm of all mathematics. Whenever a and b are two elements of a Euclidean domain R, for example the integral domain **Z** or the polynomial ring K[x] over any field K, the Euclidean algorithm is a very efficient way to compute d=gcd(a,b) in R. Here gcd(a,b) stands for the **g**reatest **c**ommon **d**ivisor of and b, which is any d that has the following two properties:

1. $d|a$ and $d|b$, i.e. d is a common divisor of and b.
2. $t|a$ and $t|b$ implies $t|d$, i.e. all common divisors of a and b are also divisors of d.

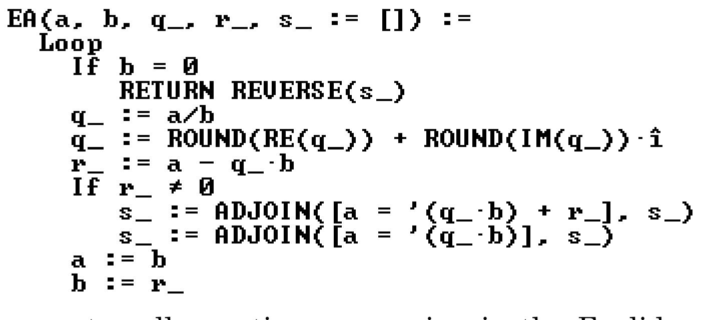
If d is a greatest common divisor of a and b, then it is easy to see that $ed$ for any unit e $\in$ R, i.e. divisor of 1, also has this property. In fact, apart from this ambiguity the gcd(a,b) is uniquely determined. Moreover, if R is a Euclidean domain, elements x,y $\in$ R can be found such that d=xa+yb and it comes as no surprise that again the Euclidean algorithm (or more precisely a slightly modified version of it) can be used to find x,y.

For a change let's consider the integral domain **Z**[i] of Gaussian integers, i.e. the subring of the field **C** consisting of all complex numbers a+bi, where a and b are integers. Suppose that u+vi, x+yi $\in$ **Z**[i] with x+yi $\neq 0$ are chosen arbitrarily. To get the "quotient" q and "remainder" r of these elements, first consider the Quotient $\tilde{q}$ of u+vi and x+yi in **C** and then let q be the "nearest" Gaussian integer to $\tilde{q}$. (Here "nearest" means with respect to the Euclidean distance, if both q and $\tilde{q}$ are viewed as points in the Gaussian plane.) From an arithmetic point of view, you get q from $\tilde{q}$ by simply rounding its coordinates to the next integer. If we define r by r:= u+vi –q(x+yi), then because of

$$|r|^2 = |u+vi-q(x+yi)|^2 = |\tilde{q}-q|^2\,|x+yi|^2 \le \frac{1}{2}|x+yi|^2 < |x+yi|^2$$

**Z**[i] becomes an Euclidean domain, if the "degree" of a Gaussian integer z is defined by $|z|^2$.

Let's use Derive to illustrate this by means of an example. The following routine



```
EA(a, b, q_, r_, s_ := []) :=
  Loop
    If b = 0
       RETURN REVERSE(s_)
    q_ := a/b
    q_ := ROUND(RE(q_)) + ROUND(IM(q_))·î
    r_ := a - q_·b
    If r_ ≠ 0
       s_ := ADJOIN([a = '(q_·b) + r_], s_)
       s_ := ADJOIN([a = '(q_·b)], s_)
    a := b
    b := r_
```

computes all equations appearing in the Euclidean algorithm applied to the Gaussian integers a and b. For example, calling ea(a,b) with a= 17+13i and b=14+11i yields the table

```
EA(17 + 13·î, 14 + 11·î)
```

$$\begin{bmatrix} 17 + 13\cdot\hat{\imath} = 1\cdot(14 + 11\cdot\hat{\imath}) + 3 + 2\cdot\hat{\imath} \\ 14 + 11\cdot\hat{\imath} = 5\cdot(3 + 2\cdot\hat{\imath}) - 1 + \hat{\imath} \\ 3 + 2\cdot\hat{\imath} = (-2\cdot\hat{\imath})\cdot(-1 + \hat{\imath}) + 1 \\ -1 + \hat{\imath} = (-1 + \hat{\imath})\cdot 1 \end{bmatrix}$$

The last nonvanishing remainder, which is 1 in this example, is then the gcd(a,b).

Now every remainder (including a and b) can be written in the form xa+yb for some Gaussian integers x,y. This is trivial for a and b, as one can choose x=1,y=0 and x=0,y=1, respectively. Assuming that we already have representations

$$r_i = x_i a + y_i b \text{ for all } i < n$$

we immediately get from

$$r_{n-2} = q_{n-2} r_{n-1} + r_n$$

a representation

$$r_n = x_n a + y_n b\ ,$$

for $r_n$, where $x_n$ and $y_n$ are given inductively by

$$x_n = x_{n-2} - q_{n-2} x_{n-1}\ ,\quad y_n = y_{n-2} - q_{n-2} y_{n-1}\ .$$

In the following program we take advantage of the fact that the numbers $x_i$ and $y_i$ are defined recursively in exactly the same way as the remainders $r_i$.

```
EXTENDED_GCD(a, b, q_, r_) :=
  Prog
    a := [a, [1, 0]]
    b := [b, [0, 1]]
    Loop
      If FIRST(b) = 0 exit
      q_ := FIRST(a)/FIRST(b)
      q_ := ROUND(RE(q_)) + ROUND(IM(q_))·î
      r_ := a - q_·b
      a := b
      b := r_
    If FIRST(a) = 0
       RETURN [0, [0, 0]]
    Loop
      If 0 ≤ PHASE(FIRST(a)) < π/2
         RETURN a
      a :* î
```

Since the units in Z[i] are exactly the four powers of i, unless the trivial case d=gcd(0,0)=0, we always have four gcd's, namely d, di, -d,-di. In the last loop of the program the one in the first quadrant is chosen. Here is again our example from above:

```
EXTENDED_GCD(17 + 13·î, 14 + 11·î) = [1, [1 - 10·î, -1 + 12·î]]

     (1 - 10·î)·(17 + 13·î) + (-1 + 12·î)·(14 + 11·î) = 1
```

Now let's consider the polynomial ring **Q**[x]over the field **Q** of rational numbers. As is well known, this is again an Euklidean domain, where the division is the "ordinary" division of polynomials in x over **Q** and the "degree" of polynomials is the usual degree of polynomials. The following question could be a nice exercise in a Derive-course: What changes have to made to the program above when dealing with this case? Here is a way to solve this problem.

```
POLY_XGCD(a, b, x, r_) :=
  Prog
    a := [a, [1, 0]]
    b := [b, [0, 1]]
    Loop
      If FIRST(b) = 0 exit
      r_ := a - QUOTIENT(FIRST(a), FIRST(b))·b
      a := b
      b := r_
    If FIRST(a) = 0
       RETURN [0, [0, 0]]
    If NUMBER?(FIRST(a))
       RETURN a/FIRST(a)
    a·DENOMINATOR(FACTOR(FIRST(a), Trivial))
```

The last line of the program makes sure that the resulting polynomial has integer coefficients. (Remember that in **Q**[x] the units are exactly the nonzero elements of **Q** and we may multiply the gcd with units to our heart's content!) Furthermore, if the gcd is a nonzero constant, the outcome will always be 1. What follows is a very small example.

```
POLY_XGCD(2·(x + 1)·(x + 2)·x, 3·(x + 1)·(x + 3), x)
```

$$\left[6\cdot(x+1),\ \left[1,\ \frac{2\cdot(1-x)}{3}\right]\right]$$

$$\left[1,\ \frac{2\cdot(1-x)}{3}\right]\cdot[2\cdot(x+1)\cdot(x+2)\cdot x,\ 3\cdot(x+1)\cdot(x+3)] = 6\cdot(x+1)$$

Another important class of Euclidean domains are the integral domains $\mathbf{Z}_p[x]$, where p is any prime. If you feel like it, you could try to modify our POLY_GCD( ) in order to cover this case as well. I for my part would however like to turn to a different topic now.

Whenever Derive-programs dealt with matrices in the past you were bound to see a lot of SUB i's and SUB i SUB j 's. This may change in the future, as Derive provides some very powerful and fast new programming constructs like FIRST(v), REST(v) and ADJOIN(v), which should help to avoid indices and double indices. Let's take a look at the following example from MISC.MTH

$$\text{POLY\_INTERPOLATE\_AUX}(a,\ x,\ k,\ n) := \left(\prod_{j\_=1}^{k-1}\frac{x-a_{j\_,1}}{a_{k,1}-a_{j\_,1}}\right)\cdot\prod_{j\_=k+1}^{n}\frac{x-a_{j\_,1}}{a_{k,1}-a_{j\_,1}}$$

$$\text{POLY\_INTERPOLATE}(a,\ x) := \sum_{k\_=1}^{\text{DIMENSION}(a)} a_{k\_,2}\cdot\text{POLY\_INTERPOLATE\_AUX}(a,\ x,\ k\_,\ \text{DIMENSION}(a))$$

and then at the following version using the constructs above

```
IP_LAGRANGE(a, x, a_, p_, q_, s_ := []) :=
  Prog
    a_ := a COL 1
    p_ := Π(VECTOR(x, i_, DIM(a)) - a_)
    Loop
      If a_ = []
         RETURN REVERSE(a COL 2)·s_
      q_ := QUOTIENT(p_, x - FIRST(a_))
      s_ := ADJOIN(q_/LIM(q_, x, FIRST(a_)), s_)
      a_ := REST(a_)
```

What a difference! Although the variables have the same meaning – in particular, a is a matrix whose rows are the points which are to be interpolated – there are no double indices and almost no single indices. Furthermore, there was absolutely no need to introduce an auxiliary function. Is there a price to pay for these goodies in terms of computation time? Not at all! The following calculation

```
IP_LAGRANGE([[1, ..., 100], [1, ..., 100]]`, x) = x
```

took 26.7 s versus 40.9 s using POLY_INTERPOLATE( ) above!

This example is a little bit embarrassing though. Every child who knows the concept of a linear function will immediately guess that the line y =x interpolates the 100 points (1,1),(2,2),…,(100,100). Is it inevitable that it takes that long? I'm afraid the answer is yes, if you insist on using Lagrange's interpolation formula, which is not exactly a good idea. On the other hand, everybody who is in the know when it comes to the efficiency of algorithms knows that from an algorithmic point of view Newton's interpolation formula is far superior to Lagrange's. Here the interpolating polynomial is represented in the form

$$f(x) = c_0 + c_1(x - x_1) + c_2(x - x_1)(x - x_2) + ... + c_{n-1}(x - x_1)...(x - x_{n-1})$$

where $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ are the given points and the coefficients $c_0, c_1, ..., c_{n-1}$ can be computed in a very efficient way by means of so-called "divided differences" (cf. [1], p504). One advantage, which can be easily seen from this formula, is that unlike Lagrange's interpolation formula after adding new points there is no need to throw away the calculations so far: The new terms which come into play on the right side will not affect the interpolation property of f(x) for the "old" points.

What follows is an implementation of Newton's interpolation formula making full use of the new powerful constructs of DfW5.

```
IP_NEWTON(p, x, a_ := 0, f_ := 1, n_, u_, v_, x_, y_) :=
  Prog
    u_ := p COL 1
    v_ := u_
    y_ := p COL 2
    n_ := DIM(y_)
    Loop
      a_ :+ FIRST(y_)·f_
      If n_ = 1 ∨ y_^2 = 0
         RETURN a_
      f_ :* x - FIRST(v_)
      n_ :- 1
      u_ := DELETE(u_, -1)
      v_ := REST(v_)
      x_ := u_ - v_
      y_ := DELETE(y_, -1) - REST(y_)
      y_ := VECTOR(y_↓i_/x_↓i_, i_, 1, n_)
```

What about our example from above? Is this routine notably faster? You bet! The following computation

```
IP_NEWTON([[1, ..., 100], [1, ..., 100]]`, x) = x
```

took only 0.02s, which is faster by a factor of more than 1 000 (!). It must be said though that this choice of points is the "best case" (more generally if the resulting polynomial is of low degree!). On average, it is for 100 points "only" 3 times faster than my

IP_LAGRANGE( ) above and only about 5 times faster than the “official” POLY_INTERPOLATE( ) from MISC.MTH.

By the way, my implementation of the Chinese Remainder Theorem (CRT) from NUMBER.MTH uses similar ideas than Lagrange’s interpolation formula:

```
CRT(a, m, p_) :=
  Prog
    p_ := Π(m)
    MOD(a·VECTOR(p_/m_·INVERSE_MOD(p_/m_, m_), m_, m), p_)
```

Here $a=(a_1,...,a_r)$ and $\mathrm{m}=(m_1,...,m_r)$ are two vectors of integers and of the same length r. Moreover, the numbers of m must be pairwise coprime. Then CRT(a,m) yields the uniquely determined integer x with $0 \le x < m_1...m_r$ such that $x \equiv a_i \bmod m_i$, i=1,2,..,r. As in Lagrange’s interpolation formula the “building blocks” for the solution x are terms, which are $\equiv \mathrm{a_i}$ mod $\mathrm{m_i}$ for exactly one i and 0 otherwise. (In a way, the $\mathrm{x_i}$ correspond to the $m_i$ and the $y_i$ to the $\mathrm{a_i}$!) If you are guessing now that there might be possible an improvement by switching from Lagrange’s idea to Newton’s, you certainly have a point there! Here is the program

```
crt(a, m, m_ := 1, x_ := 0) :=
  Loop
    If m = []
       RETURN x_
    x_ := x_ + (FIRST(a) - x_)·m_·INVERSE_MOD(m_, FIRST(m))
    m_ :* FIRST(m)
    x_ := MOD(x_, m_)
    a := REST(a)
    m := REST(m)
```

In order to see the difference we select 10 integers $a_i$ and $m_i$ each with about 100 digits by simplifying the following two statements

```
                                       100
m := ITERATES(NEXT_PRIME(p), p, 10       + 267, 9)

                        100
a := VECTOR(RANDOM(10     ), i, 1, 10)
```

It takes CRT(a,m) only 0.09s to compute the solution on my Pentium 450 PC, which is incredibly fast! Even though, crt(a,m) surpasses this performance with 0.01s, i.e. by a factor of almost 10! (Check it!)

Let me conclude by pointing out that Derive has learnt to deal with strings at last! I am sure that we’ll see a lot of intriguing new programs that make use of this new feature. One that comes to mind at once is the use in cryptography for source coding. Unlike other CAS source coding and decoding, i.e. the translation of a message consisting of alphanumerical characters into a number and back, had not been possible before the advent of DfW5.

But they’re a lot of other applications where you could use it. Take the routine ODD_PART(n) from the last Titbits for example. The implementation I offered then was

```
ODD_PART(n) :=
  Loop
    If ODD?(n)
       RETURN n
    n := n/2
```

Would you think that there is still a much shorter implementation? Believe it or not, there is one! Look!

```
ODD_PART(n) := REVERSE(REVERSE(n))
```

This immediately leads to the shortest program for the Rabin-Miller-test that is conceivable! (If there were a section “short and sweet programs” in Guiness’ book of records, this program would surely be a contender for victory!)

```
RABIN_MILLER(n, a, s_) :=
  Prog
    s_ := REVERSE(REVERSE(n - 1))
    a := - ABS(MODS(a^s_, n))
    Loop
      If a = -1 exit
      s_ :* 2
      If s_ = n - 1
         RETURN false
      a := MODS(a^2, n)
```

On average, it is a little bit slower than the #1 in terms of speed, namely

```
RABIN_MILLER(n, a, s_) :=
  Prog
    s_ := n - 1
    Loop
      s_ :/ 2
      If ODD?(s_) exit
    a := - ABS(MODS(a^s_, n))
    Loop
      If a = -1 exit
      s_ :* 2
      If s_ = n - 1
         RETURN false
      a := MODS(a^2, n)
```

but there is an interesting exception. As is well-known, all Fermat numbers $F_n$, i.e.numbers of the form

$$F_n = 2^{2^n} + 1 \quad (n \ge 0)$$

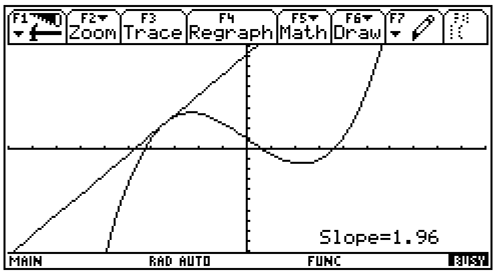
fulfil the Rabin-Miller-test for the base a=2. The corresponding test is for n>10 about 4 times faster with the “short” implementation, though in the region of n=17 you’ll get a memory overflow! More about these peculiar numbers another time! (j.wiesenbauer@tuwien.ac.at)

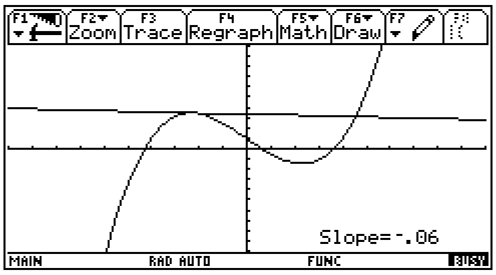
[1] D.E.Knuth, The Art of Computer Programming, Vol 2: Seminumerical Algorithms, 3rd ed., Addison-Wesley, Reading, Mass.,1998

## Let´s move a tangent along a graph and note its slope

Thomas Himmelbauer, Vienna, Austria

```
tangent()
Prgm
DelVar xa,ya,key,ka,df,da,pic1
Text "Function defined as y1(x)?"
getMode("ALL")→state
setMode("Graph","FUNCTION")
setMode("Display Digits","FIX 2")
setGraph("axes","ON")
setGraph("grid","OFF")
Define df(x)=d(y1(x),x)
FnOff :PlotsOff :FnOn 1:ZoomStd
StoPic pic1
Trace
xc→xa:10→xres
df(xa)→ka:y1(xa)-ka*xa→da
FnOff 1
Define y2(x)=ka*x+da
RclPic pic1
PtText "Slope="&string(ka),3,⁻8
Loop
getKey()→key
While key=0
getKey()→key
EndWhile
If key=4360 Then
  Goto end
EndIf

If key=340:xa+20/238→xa

If key=337:  xa-20/238→xa

df(xa)→ka
y1(xa)-ka*xa→da
FnOff 1
Define y2(x)=ka*x+da
RclPic pic1
PtText "Slope="&string(ka),3,⁻8

EndLoop
Lbl end
DelVar xa,ya,key,ka,df,da,pic1
setMode(state)
EndPrgm
```





Each keypress ► or ◄ moves the point on the graph together with its tangent. The current slope of the tangent is displayed on the screen.

Programming of the cursor´s coordinates, controlling the cursor using the keyboard, working with system variables xc and yc, storing and loading of pictures.

Used commands are:
`delvar, text, getmode, setmode, setgraph, define, fnoff, plotsoff, fnon, zoomstd, stopic, trace, rclpic, pttext, loop - endloop, if - then - endif, getkey(), while - endwhile`.

Recommended exercise: trace the graph stepwise by simultaneously generating and calculating the area between graph and x-axis.