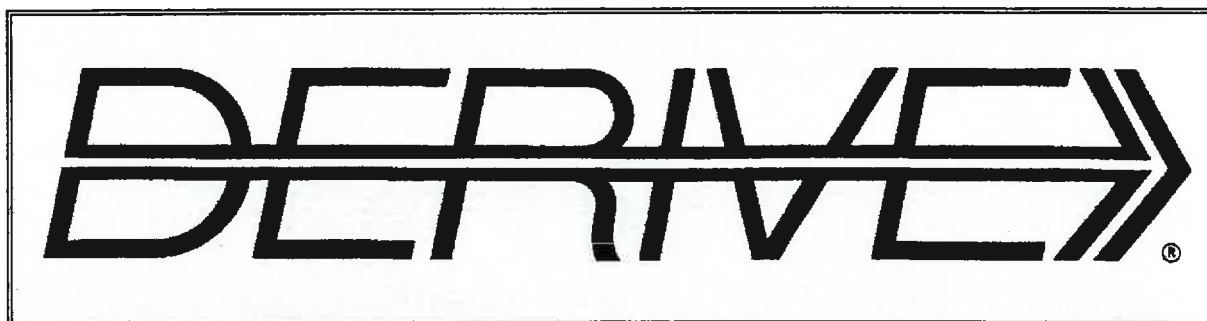


THE BULLETIN OF THE



USER GROUP

+ CAS-TI

Contents:

- | | |
|----|--|
| 1 | Letter of the Editor |
| 2 | Editorial - Preview |
| 3 | User Forum 1 |
| | Timothy Comar |
| 7 | Visualization of Hyperbolic Möbius Transformation (Part 2) |
| | Franz Schlöglhofer |
| 11 | Bézier Curves in School |
| | Josef Böhm |
| 23 | LINDENMAYER Systems |
| | MacDonald Phillips |
| 31 | Bond Price and Yield |
| 37 | User Forum 2 |
| 44 | Wonderful World of DERIVE 6 |
| | Rüdiger Baumann |
| 49 | The Josephus Problem |
| | Johann Wiesenbauer |
| 51 | Titbits from Algebra and Number Theory (26) |

- [1] *Interconnectivity: DERIVE and TI-CAS Rechner im MU*, B. Kutzler & V. Kokol-Voljc, bk teachware 2003, SR-37
- [2] *Analysis mit DERIVE*, H.-J. Kayser, bk-teachware, SR-38, to appear 2004
- [3] *Parabeln und Co. erforschen mit DERIVE*, Rainer Wonisch, bk-teachware, SR-39, to appear 2004

DERIVE Book Bargain

"Learning Differential Equations through DERIVE" develops the standard theory of first and second order differential equations, with applications to the physical and environmental sciences – supported by Derive for Windows.

We are offering the above title by Brian Lowe and John Berry to DUG readers at the special price of just 5 GBP + postage (usually 19.95 GBP). 383 Pages.

Chartwell-Yorke, Ltd

114 High Street, Belmont Village, Bolton, Lancashire, BL7 8AL

susan.yorke@cymaths.demon.co.uk

Websites

A very fine designed website presenting online-courses for working with the Voyage 200. It is in French and could be very interesting for students working with the TI and learning French.

<http://www.t3ww.org/france/fonctions.htm>

USA TODAY and Texas Instruments alliance to deliver standards-based math materials to classroom via handheld technology. Visit the following two websites



<http://education.ti.com/USAT>

<http://www.education.usatoday.com>

A linear Algebra Toolkit (recommended by <http://www.mathgate.ac.uk/>)

<http://shorl.com/fypivovitaho>

<http://shop.bk-teachware.com>

This is the address for DERIVE - & TI-related books. There is also a rich resource of additional software and inspiring maths books.

TIME-2004

Montreal Int'l Symposium on Technology and its Integration
into Mathematics Education

15-18 July 2004, Montréal, Québec, Canada

www.time-2004.etsmtl.ca

Dear DUG Members,

First of all I'd like to wish you a happy, healthy and peaceful New Year 2004. Unfortunately it was not possible to finalize and ship this *DNL* in 2003. One reason was my long – wonderful – stay in New Zealand, another one that the print shop was too busy in the last weeks of 2003. But now you have an XL *DNL* in your hands. This *DNL*#52 marks the end of a remarkable and successful 13-years history and the begin of a new era as well.

One of my eyes is weeping, because we stop mailing the printed publication of the *DNL*. I made this decision because of the enormously increasing mailing charges together with the wish of many members for receiving the *DNL* in electronical format. So my other eye is laughing announcing that we will go on publishing the *DNL* in pdf-format. This will work much faster than surface mail, you will enjoy colors and hyperlinks, will be able to produce copies of articles of your interest and surely very pleasant and welcome: you will save money! As we don't have costs for printing and mailing any longer, **DUG MEMBERSHIP IS FREE** in the future. All of you remain members for life time (at least for life time of the *DUG*). In the future you can download the *DNL* from the websites given below. I'd like to invite you to mail your email-address, then I could inform you about publication date of the future *DNL*s. If somebody has no access to internet then please contact me. We will find a way to resolve this problem.

I am very grateful for your friendship, loyalty and cooperation during the last 13 years and I hope that we can enjoy the future together for another 13?? years. It is my strong intention to go on with the *DUG* as long as there are enough contributions to be published. So I'd like to encourage you to send your articles – in particular I'd like to invite once more the TI-CAS-users to share their experiences.

Going on the web with the *DNL*s I have some ideas: I intend to revise all earlier *DNL*s beginning with #1 from 1991, adjusting them for *DERIVE* version 5

(and 6?) and converting them into pdf-format by and by. This should go hand in hand with providing an extended index for the contents and authors. So if you have some ideas for improvements for earlier *DNL*s then please let me know.

At the occasion of starting the new electronical future of the *DNL* I'd like to thank some friends for their invaluable support during the last years: this is the *DERIVE*-team, Theresa Shelby, Albert Rich and David Stoutemyer – they never let me alone

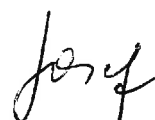
with any question, might it have been silly or not. Then I must mention Andrea Kutzler. She took care for organizing printing the *DNL*s and Walter Wegscheider and Benjamin Kainerd who very reliable put the *DERIVE*-files on the web, thanks folks. And finally vicarious for all contributors I'd like to thank Johann Wiesenbauer. His Titbits have become an essential part of the *DNL* and I hope that he will go on sharing his intimate knowledge of

DERIVE (and number theory, of course) with us.

Let me close with some words regarding *DNL*#52. This is an extraordinary large issue, as promised in *DNL*#51 and it is always the same, I have some papers on my table and putting them together I find that I would need some pages more – even with 60 pages. So I had to leave among others Rainer Geyer's wonderful contribution on fractal tetrahedrons for the next issue. We have some interesting discussion points in this User Forum, some more are waiting to be published. Simultaneously with the release of *DERIVE* 6 we show some exciting features of the new version. Please take care of the different files for versions 5 and 6. I'll come back to this important question next time.

See you in Canada?? We could arrange a *DERIVE* User Group meeting in Montréal!!

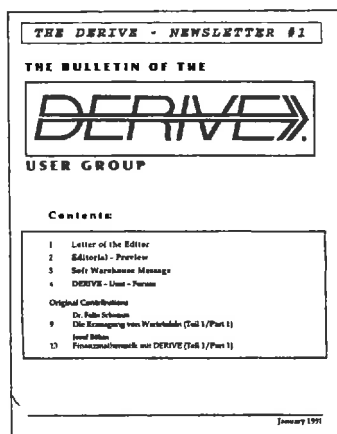
Regards as ever



Download all *DNL-DERIVE*- and TI-files from

<http://www.acdca.ac.at/t3/dergroup/index.htm>

<http://www.bk-teachware.com/main.asp?session=375059>



The *DERIVE-NEWSLETTER* is the Bulletin of the *DERIVE & CAS-TI User Group*. It is published at least four times a year. The goals of the *DNL* are to enable the exchange of experiences made with *DERIVE* and the *TI-89/92/Voyage 200* 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 *CAS-TIs* 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 & CAS-TI Newsletter* will be.

Editor: Mag. Josef Böhm
A-3042 Würmla
D'Lust 1
Austria
Phone/FAX: 43-(0)2275/8207
e-mail: nojo.boehm@pgv.at

Next issue: March 2004
Deadline: 15 February 2004

Preview: Contributions waiting to be published

Finite continued fractions St. Welke, GER
Kaprekar's "Self numbers", R. Schorn, GER
Some simulations of Random Experiments, J. Böhm, AUT
Wonderful World of Pedal Curves, J. Böhm
Another Task for End Examination, J. Lechner, AUT
Tools for 3D-Problems, P. Lüke-Rosendahl, GER
ANOVA with *DERIVE & TI*, M. R. Phillips, USA
Hill-Encryption, J. Böhm
CAD-Design with *DERIVE* and the *TI*, J. Böhm
Sierpinski-Tetrahedrons and Octahedrons, H.-R. Geyer, GER
Avoiding Convolution and Transforming Methods, M. Lesmes-Acosta, COL
Farey Sequences on the *TI*, M. Lesmes-Acosta, COL
Simulating a Graphing Calculator in *DERIVE*, J. Böhm, AUT
Henon & Co, J. Böhm
Rule 90 and other Cellular Automata, D. Sjöstrand, SWE
A Time-Value-Money Solver for *DERIVE*
and
Setif, FRA; Vermeylen, BEL; Leinbach, USA; Koller, AUT,
Keunecke, GER,and others

Impressum:

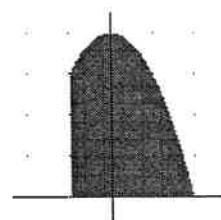
Medieninhaber: *DERIVE* User Group, A-3042 Würmla, D'Lust 1, AUSTRIA
Richtung: Fachzeitschrift
Herausgeber: Mag. Josef Böhm
Herstellung: Selbstverlag

Dirk Callens

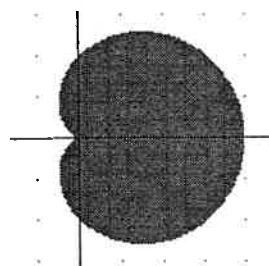
dirkcallens@hotmail.com

When we solve inequalities, using cartesian co-ordinates or polar coordinates, a certain region of the plane can be coloured.

$$-1 < x < 2 \wedge 0 < y < 4 - x^2$$



$$r < 1 + \cos(\theta)$$

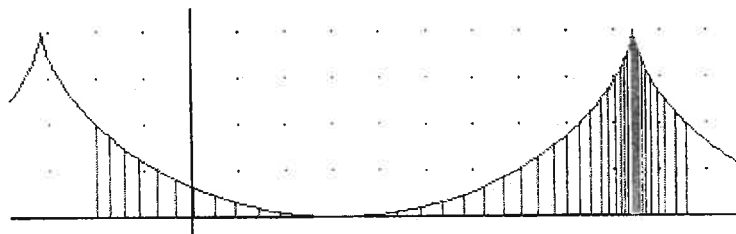


But how can we do this working with curves given in parameter form?

Can you shade the area between the x-axis and the cycloid $x = t - \cos t$, $y = 1 - \sin t$ for $0 < t < 2\pi$?

DNL: This is a demanding question. I failed using the relations and tried to fill the area by segments. It is obvious that by using parameter t for the intervals, the segments are not uniformly distributed. (Why? Fine question for students!).

```
shade_parm_x(p_curve, par, start, end, n) := UVECTOR([p_curve, [p_curve_1, 0]], par,
start, end,  $\frac{\text{end} - \text{start}}{n}$ )
shade_parm_x([t - COS(t), 1 - SIN(t)], t, 0, 2*PI, 50)
```



```
t_vals(p_curve, par, start, end, n) := UVECTOR(NSOLUTIONS(x_ = p_curve_1, t)), x_,
1
SUBST(p_curve_1, par, start), SUBST(p_curve_1, par, end),
 $\frac{\text{SUBST}(p\_curve_1, \text{par}, \text{end}) - \text{SUBST}(p\_curve_1, \text{par}, \text{start})}{n}$ 
```

```
#9: t_vals([t - COS(t), 1 - SIN(t)], t, 0, 2·π, 10)
```

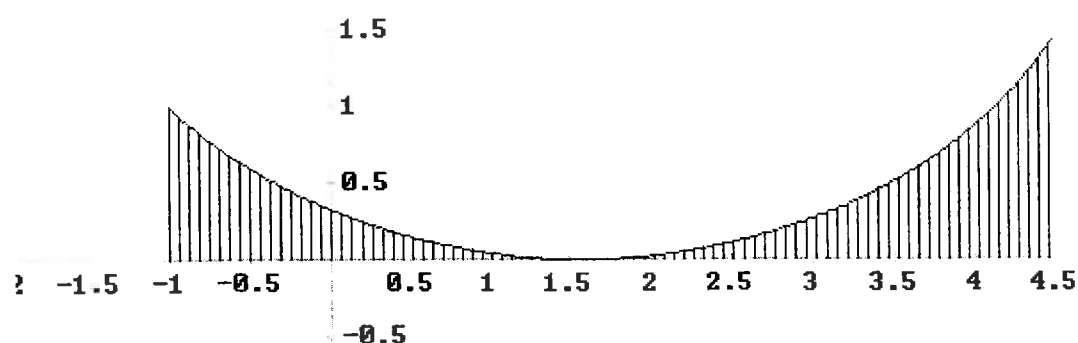
```
#10: [0, 0.503984, 0.887775, 1.22443, 1.54203, 1.85816, 2.18973, 2.56170, 3.03249,
      4.00506, 6.28318]
```

```
#11: shade2_parm_x(p_curve, par, start, end, n) := APPEND([p_curve], VECTOR([p_curve,
      [p_curve_1, 0]]], par, t_vals(p_curve, par, start, end, n)))
```

```
#12: shade2_parm_x([t - COS(t), 1 - SIN(t)], t, 0, 2·π, 100)
```

`t_vals(...)` returns uniformly distributed x-values for the area to be shaded.

`shade2_parm(...)` plots the curve together with n segments shading the area:

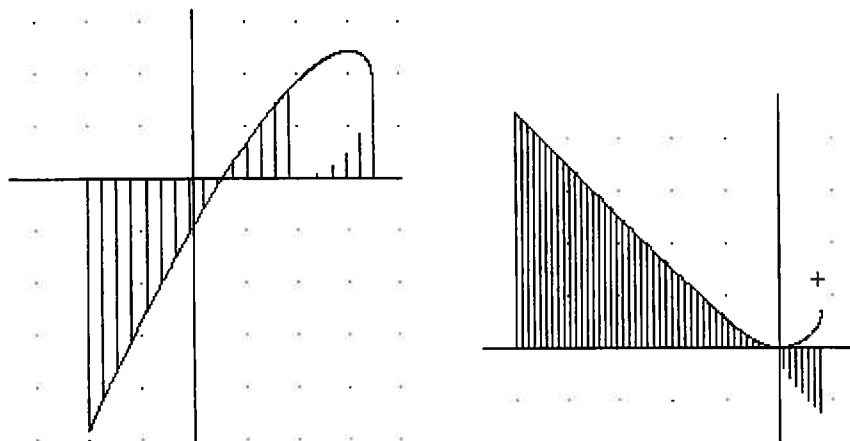


I applied my function with two other parameter representations but – not unexpected because of the problems occurring by solving the equation numerically – I was not really satisfied with the results.

```
shade2_parm_x([ [1 - t^2/4 + sqrt(t), -t^3/10 + t], t, 1, 4, 20)
```

```
folium := [ (3·t / (1 + t^3), 3·t^2 / (1 + t^3)) ]
```

```
shade2_parm_x(folium, t, -0.9, 0.9, 50)
```



I wrote to Albert Rich, who forwarded the mail to Peter Schofield, "the expert on DERIVE graphics", with the hope that "he will have some insights into the matter". Many thanks to Albert, who is really busy in improving DERIVE 6, which is a tremendous work.

And many thanks to Peter who sent an extended answer very soon and promised in a later mail an improved version.

Peter Schofield

Hello Josef, Albert and Theresa,
Thank you for your email Albert.

I know of no straightforward way of shading some parametrically defined areas. In some cases, if you can eliminate the parameter variable to obtain a closed form equation in x and y, then (using this) it is possible to make up an outline and corresponding Boolean expression to plot the interior area. However, this does not appear possible for the example of a cycloid below.

Please find attached: ParameterAreas.Zip (my efforts at solving this problem). This uses a variation of one of my bodyscanning techniques to cut up the required the area into strips - similar to the way one describes the geometrical interpretation of an integral. I suspect this is like the method tried by Josef.

ParameterAreas.zip contains two mth files:

ParameterAreas1.mth - Plots areas (integrals) in 2D between parametric curves and the x-axis.
(The ORPLOT instruction is thanks to Albert.)

ParameterAreas2.mth - Plots areas (integrals) in the Oxy-plane in the 3D-plot window between parametric curves and the x-axis. (Notice that Derive 3D-surface plots are constructed from quadrilateral panels, and so the strips are easier to define in the 3D-plot window.)

The files use rectangular coordinates and require
Options>Approximate Before Plotting ON (in both plot windows).

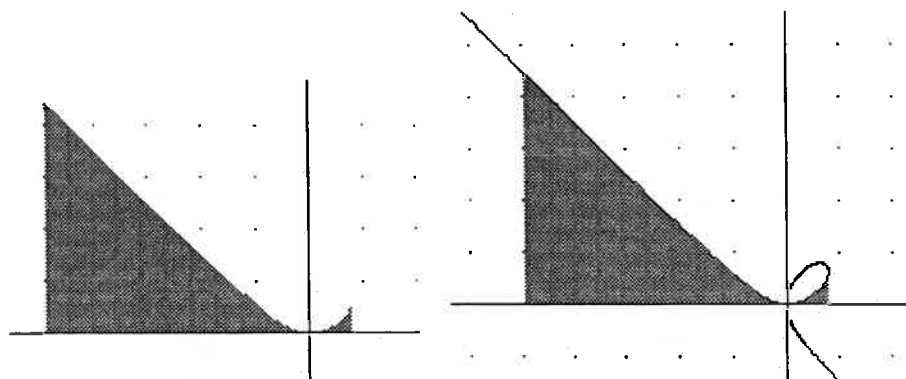
Hope you find this helpful.

All the best for Christmas and New Year.

Peter

$$\text{ORPLOT} \left(\text{XYSCAN} \left(\text{VECTOR} \left(\left[\frac{3 \cdot t}{1 + t^3}, \frac{3 \cdot t^2}{1 + t^3} \right], t, -0.9, 0.9, 0.05 \right) \right) \right)$$

$$\text{XYSCAN} \left(\text{VECTOR} \left(\left[\frac{3 \cdot t}{1 + t^3}, \frac{3 \cdot t^2}{1 + t^3} \right], t, -0.9, 0.9, 0.1 \right) \right)$$



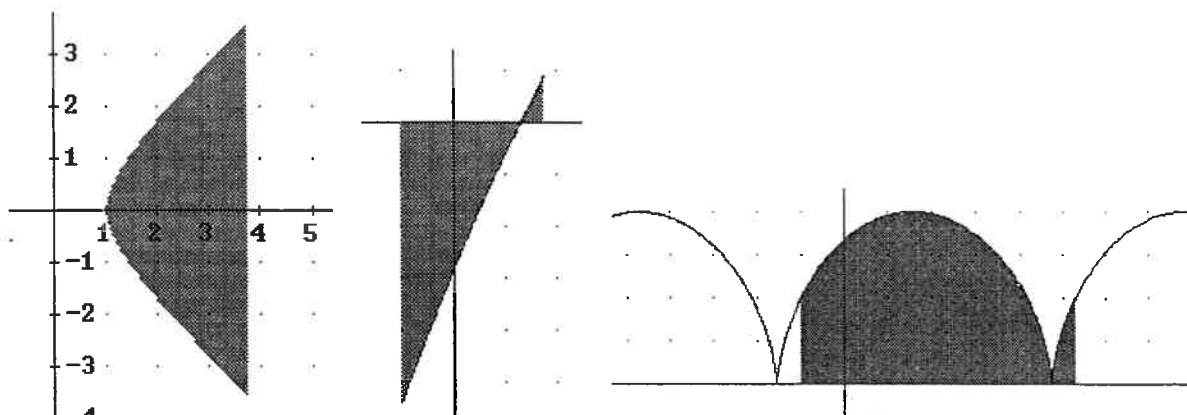
$$\text{ORPLOT}(\text{XYSCAN}(\text{VECTOR}([\cosh(t), \sinh(t)], t, -2, 2, 0.1)))$$

$$\text{ORPLOT} \left(\text{XYSCAN} \left(\text{VECTOR} \left(\left[1 - \frac{t^2}{4} + \sqrt{t}, -\frac{t^3}{10} + 1 \right], t, 1, 4, 0.05 \right) \right) \right)$$

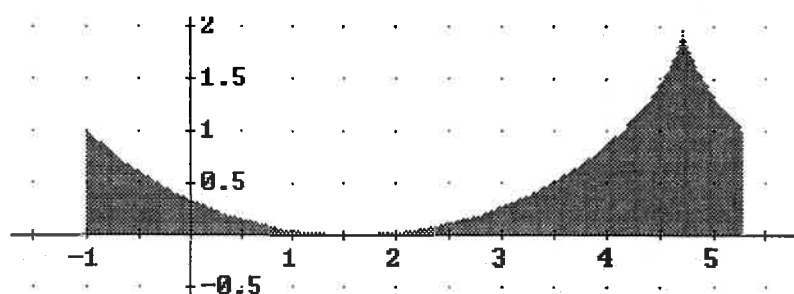
$$\text{parm_shade}(v_ , t_ , t1_ , t2_ , \text{step}) := \text{ORPLOT}(\text{XYSCAN}(\text{VECTOR}(v_ , t_ , t1_ , t2_ , \text{step})))$$

$$\text{parm_shade} \left([t - \cos(t), 2 + 2 \cdot \sin(t)], t, 0, 2 \cdot \pi, \frac{\pi}{10} \right)$$

(parm_shade is my invention to make the input easier. Josef)

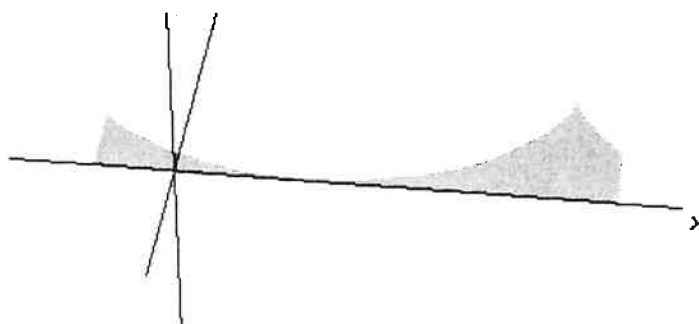


`parm_shade([t - COS(t), 1 - SIN(t)], t, 0, 2·π, $\frac{\pi}{10}$)`



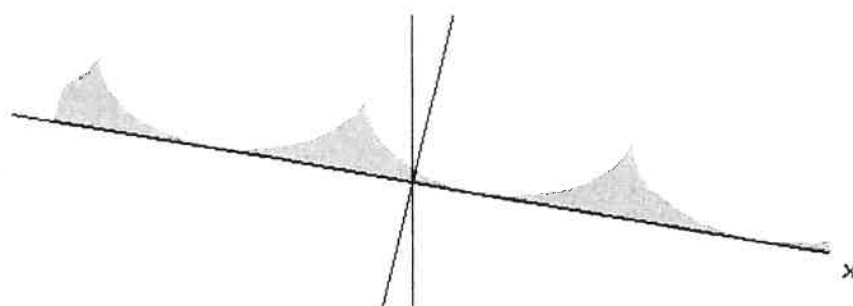
Let's have a look onto the cycloid in 3D-Plot Window:

`XYSCAN_3(VECTOR([t - COS(t), 1 - SIN(t)], t, 0, 2·π, $\frac{\pi}{10}$))`



`parm_shade_3d(v_, t_, t1_, t2_, step) := XYSCAN_3(VECTOR(v_, t_, t1_, t2_, step))`

`parm_shade_3d([t - COS(t), 1 - SIN(t)], t, -3·π, 3·π, $\frac{\pi}{10}$)`



VISUALIZATION OF HYPERBOLIC MÖBIUS TRANSFORMATIONS IN TWO AND THREE DIMENSIONS USING DERIVE (2)

Timothy D. Comar
Benedictine University
tcomar@benedu

Step 6. Plot the geodesic axis A.

We use the command

$$\text{AXIS}(M) = [0, 0, t],$$

which returns the parameterization of A , where $t > 0$.

Figure 4 shows the plot of A , which orthogonally intersects $HP(1,0,0)$ and $HP(4,0,0)$ at the points $(0,0,1)$ and $(0,0,4)$ respectively.

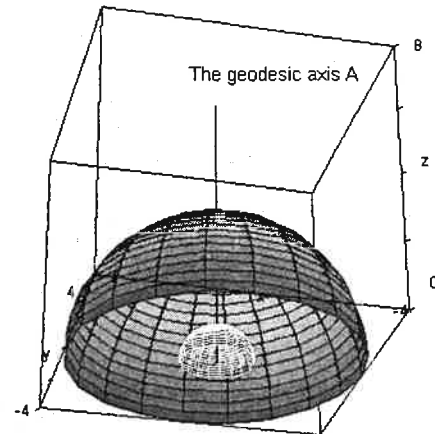


Figure 4: The geodesic axis A of \tilde{f}

Notice that \tilde{f} preserves A as a set. In the next step, we investigate how far \tilde{f} translates points along A . To do this, we recall that the hyperbolic arc length element on \mathbf{H}^3 is defined as follows. Let $\gamma(t) = (x(t), y(t), z(t))$, where $t \in [a, b]$, where $0 < a < b \leq \infty$, be the curve in \mathbf{H}^3 . The hyperbolic arc length is

$$ds = \frac{\sqrt{dx^2 + dy^2 + dz^2}}{z} = \frac{\sqrt{\frac{d^2x}{dt^2} + \frac{d^2y}{dt^2} + \frac{d^2z}{dt^2}}}{z(t)} dt$$

and the hyperbolic arc length of γ is

$$L_\gamma = \int_{t=a}^{t=b} ds = \int_a^b \frac{\sqrt{\frac{d^2x}{dt^2} + \frac{d^2y}{dt^2} + \frac{d^2z}{dt^2}}}{z(t)} dt.$$

Step 7. Compute the hyperbolic translation distance of \tilde{f} along A . This distance is the hyperbolic arc length of the geodesic segment of A from $(0, 0, a)$ to $\tilde{f}((0, 0, a)) = (0, 0, 4a)$, where $a > 0$.

Parameterize the geodesic segment of A from $(0, 0, a)$ to $(0, 0, 4a)$ by $\gamma(t) = (0, 0, t)$, where $a \leq t \leq 4a$. The hyperbolic arc length integral reduces to

$$L := \int_a^{4 \cdot a} \frac{1}{t} dt$$

which simplifies to $2 \cdot \ln(2)$, or $\ln(4)$. This means that any point on A is translated along A away from 0 by a distance of $\ln(4)$. This is the minimal translation length of f because A is a geodesic. For contrast, consider the Euclidean segment joining the points $(a, a, 0)$ and $\tilde{f}((t, 0, t)) = (4a, 0, 4a)$. This segment is not a geodesic segment and will have hyperbolic length greater than $\ln(4)$. To confirm this, we proceed to the next step.

Step 8. Compute the hyperbolic length of the Euclidean segment joining the points $(a, a, 0)$ and $\tilde{f}((t, 0, t)) = (4a, 0, 4a)$, where $a > 0$.

Parameterize the segment by $\alpha(t) = (t, t, 0)$, where $a \leq t \leq 4a$. The hyperbolic arc length integral reduces to

$$\int_a^{4 \cdot a} \frac{\sqrt{2}}{t} dt = 2 \cdot \sqrt{2} \cdot \text{LN}(2).$$

We now proceed to illustrate a regular neighborhood of the geodesic axis A . The ϵ -regular neighborhood of A is the set of points

$$N_\epsilon(A) = \{x \in \mathbf{H}^3 \mid d_h(x, A) < \epsilon\}$$

and its boundary is defined to be the set

$$\partial N_\epsilon(A) = \{x \in \mathbf{H}^3 \mid d_h(x, A) = \epsilon\}$$

where $d_h(x, A)$ is the minimum hyperbolic distance between the point $x \in \mathbf{H}^3$ and the geodesic axis A . We have created a *DERIVE* routine `RegNbdAXIS(MAT, ϵ)` which returns a parameterization of the boundary of the ϵ -regular neighborhood of the geodesic axis of the (nontrivial, non-parabolic) Möbius transformation represented by the matrix MAT . Note the creation of the routine itself provides a challenging exercise in trigonometry.

Step 9. Find a parameterization of the boundary of the 1-regular neighborhood of A and plot its surface.

First note that a Möbius transformation and its inverse have the same geodesic axis and that the matrix representation of \tilde{f}^{-1} is just M^{-1} . For sake of plotting, it will be easier to plot this regular neighborhood in terms of \tilde{f}^{-1} because its attracting fixed point is 0 rather than ∞ . We proceed as follows:

$$\text{APPROX}(\text{RegNbdAXIS}(M, 1)) = [1.175201256 \cdot \kappa \cdot \cos(\theta), 1.175201256 \cdot \kappa \cdot \sin(\theta), \kappa]$$

or

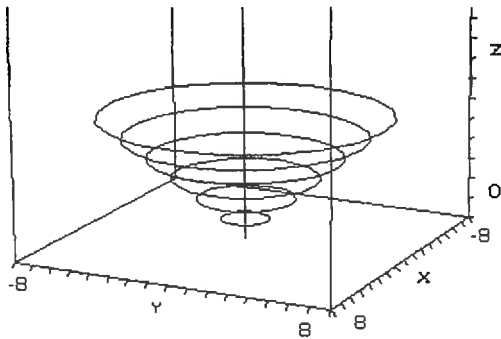
$$\text{APPROX}(\text{RegNbdAXIS}(M^{-1}, 1)) = [1.175201256 \cdot \kappa \cdot \cos(\theta), 1.175201256 \cdot \kappa \cdot \sin(\theta), \kappa]$$

where $0 \leq \theta \leq 2\pi$ and $\kappa > 0$. By looking at this parameterization, notice that $\tilde{f}(\partial N_1(A)) = \partial N_1(A)$. By substituting the values 1 to 6 for κ and plotting we observe that the level curves of this regular neighborhood are circles at Euclidean height κ centered at points on A . (See Figure 5.) See Figure 6 for the plot of the parameterized surface itself.

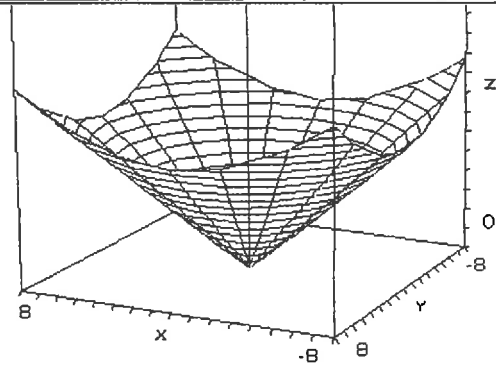
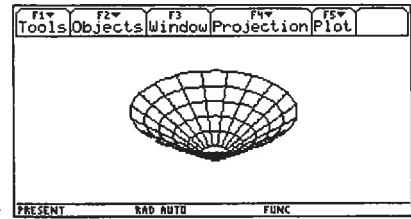
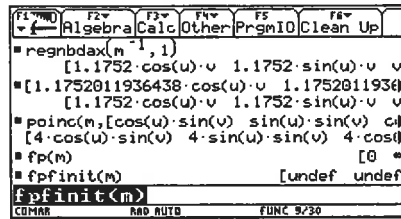
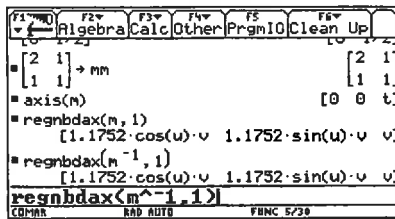
We see that this surface is an Euclidean cone with A as its axis! To develop a better sense of the geometry of a regular neighborhood of a geodesic in \mathbf{H}^3 , we conclude by briefly investigating a hyperbolic Möbius transformation which has two finite fixed points in \mathbf{C} .

Step 10. Denote by g the Möbius transformation with the matrix representation $MM = \begin{bmatrix} 2 & 1 \\ 1 & 1 \end{bmatrix}$.

Determine the fixed points, geodesic axis B , and a parameterization of the boundary of the 0.5-regular neighborhood of B . Plot B and the boundary of its 0.5-regular neighborhood.



VECTOR(RegNbdAXIS(M, 1), κ, 6)

Figure 5: Several level curves of $\partial N_1(A)$ Figure 6: The surface $\partial N_1(A)$ is an Euclidean cone

See the handheld-realization using TI-functions and a self written 3D-plot program. The cone is represented in isometric projection.

We use the command $FP(K)$ to determine the fixed points of g , which are the two points in \mathbb{C} (reported as an ordered pair)

$$FP(MM) = \left[\frac{1}{2} - \frac{\sqrt{5}}{2}, \frac{\sqrt{5}}{2} + \frac{1}{2} \right]$$

A parameterization for the geodesic axis B of g is determined using the command $AXIS(MM)$, which returns

$$AXIS(MM) = \left[\frac{\sqrt{5} \cdot \cos(t)}{2} + \frac{1}{2}, 0, \frac{\sqrt{5} \cdot \sin(t)}{2} \right]$$

where $0 \leq t \leq \pi$. Hence this geodesic axis is a semicircle that is orthogonal to \mathbb{C} at its end points. Figure 7 shows the plot of this geodesic axis.

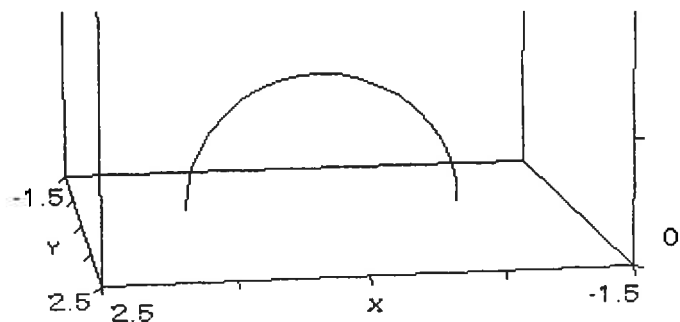
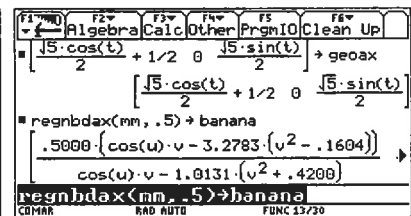
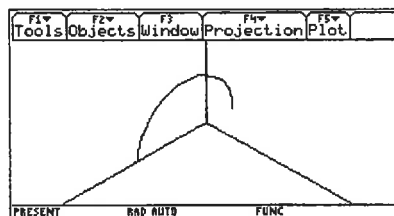
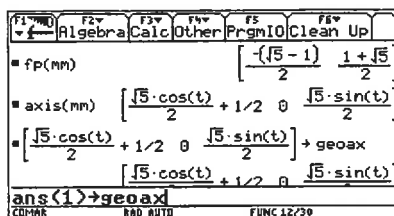


Figure 7



The TI-performance leads to the final results, too. The functions are collected in the grouped file mobmath. J.

Now the command `RegNbdAXIS(MM, 0.5)` approximates to a rather messy parameterization. By substituting the values $\kappa = 0.5, 1, 2, 3, 4, 5, 6, 7$ and plotting, we see that the level curves are circles centered along the geodesic axis B . Moreover, larger and larger values of κ give circles that become smaller and smaller in Euclidean radius and closer and closer attracting fixed point $\frac{1+\sqrt{5}}{2}$ of g .

Part of the *DERIVE*-result (below) and Figure 8:

`RegNbdAXIS(MM, 0.5)`

$$\left[\frac{4.98274 \cdot 10^{-6} \cdot (5789 \cdot \kappa^2 \cdot \cos(\theta)^2 + 6.88567 \cdot 10^{17} \cdot \kappa \cdot \cos(\theta) - 6.86191 \cdot 10^{12} \cdot \kappa \cdot \cos(\theta) - 8.37198 \cdot 10^{12})}{15 \quad -7 \quad 19} \right]$$

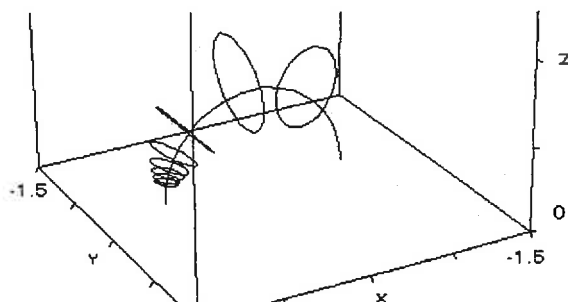


Figure 8: Several level curves of $\partial N_{0.5}(B)$

We now plot the complete surface below in Figure 9. We observe that in this case that a hyperbolic transformation has two finite fixed points, regular neighborhoods of its geodesic axis are “banana-shaped”.

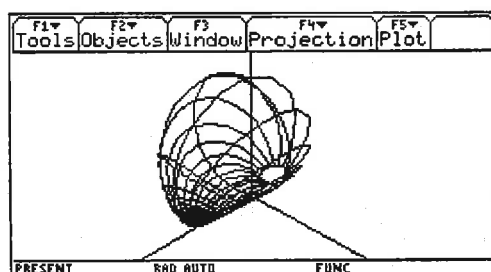
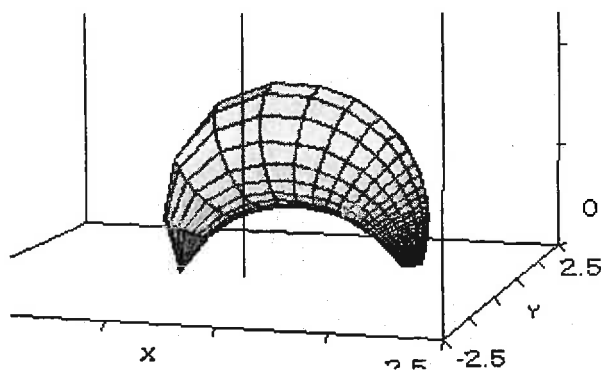


Figure 9: The surface $\partial N_{0.5}(B)$ is “banana-shaped”.

For further information

Additional activities and the latest version of **mobmath.mth** [2] can be obtained by contacting the author directly at tcomar@ben.edu.

References

- [1] A. F. Beardon, *The Geometry of Discrete Groups*, Springer-Verlag, 1983
- [2] T. D. Comar, **mobmath.mth**, Derive math files, tcomar@ben.edu, November, 2002
- [3] T. D. Comar, “Visualization of Möbius transformations in two and three dimensions using a CAS”, Proceedings of the 15th Annual International Conference of Technology in Collegiate Mathematics (to be published in September 2003 by Addison-Wesley), 2004

`mobmath.dfw`, which was used for this contribution can be downloaded (in the DNL#51-files).

Bézier Curves in School

Franz Schlöglhofer
Pädagogische Akademie Linz; University Linz

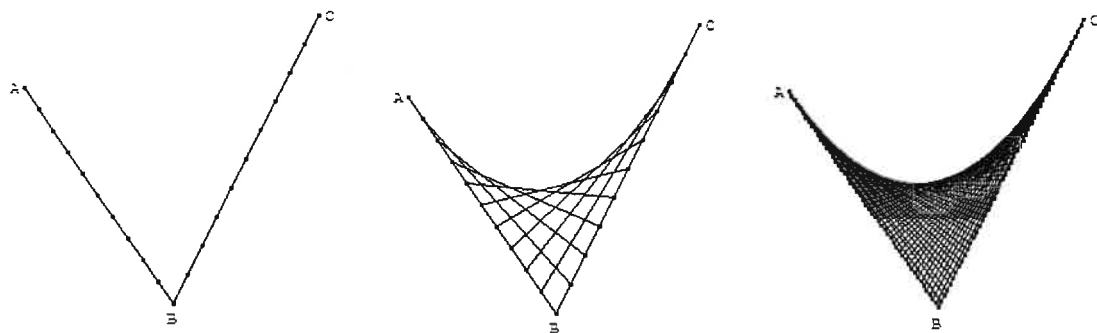
In the origin Bézier curves came into being from motor car industry especially from France (Citroen, Renault). In the 1960th there was a demand on good connections between design and production of cars. The figures produced in the design-studio should be used in a mathematic manner for the programming of the machines to produce the cars. Now Bézier curves are also used especially for computer-graphics (graphics-programs, True-Type-Fonts, ...)

The subject of Bézier curves appears to be an interesting field of mathematics for school. Students can experience modern applications of mathematics. The computer is a useful and necessary tool for calculating and plotting the curves as well. Until now there is no tradition in school-praxis for these curves. I believe that one should confront pupils with the phenomenas first and not with the mathematical background.

A Geometric Definition of Bézier Curves

Bézier curves of degree 2

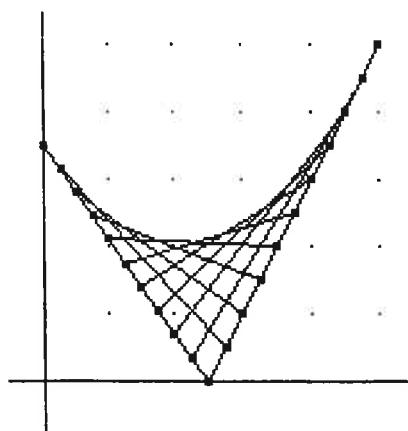
The following figures show two line segments AB and BC . The segments are divided in ten parts of equal length. In the center figure the points are connected with lines. With a little bit of phantasy you can imagine a "curve" from A to C .



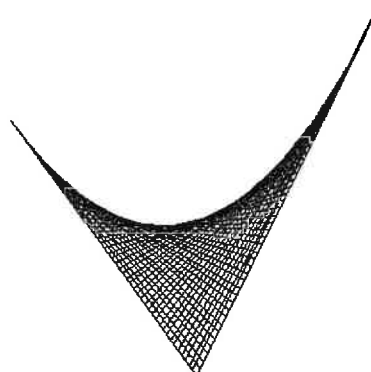
The imagination of a curve between A and C will improve if you divide the segments in 50 parts of equal length. Up to now we have only a imagination of a curve but we need a construction of points of the curve.^[1]

^[1] It is not difficult to produce a function which plots these two families of line segments created by two series of points lying on two segments AB and CD . Josef

```
threads(x1, y1, x2, y2, x3, y3, x4, y4, n) := APPEND( [ [ x1  y1 ], [ x3  y3 ],
[ x2  y2 ], [ x4  y4 ] ],
VECTOR( [ [ [x1, y1] + (i/n) * [x2 - x1, y2 - y1], [x3, y3] + (i/n) * [x4 - x3, y4 - y3] ], i,
0, n ] )
threads(0, 7, 5, 0, 5, 0, 10, 10, 10)
threads(0, 7, 5, 0, 5, 0, 10, 10, 50)
```



Students can use this function to create nice pictures.

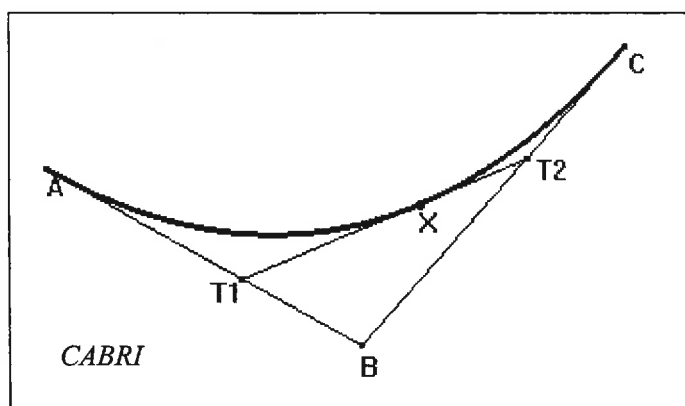


Bézier curve of degree 2

The **Bézier polygon** ABC and the **Bézier points** A, B and C define the curve.

We get point X of the curve with the following proportions of lengths of segments:

$$\overline{AT_1} : \overline{AB} = \overline{BT_2} : \overline{BC} = \overline{T_1X} : \overline{T_1T_2} = t : 1$$



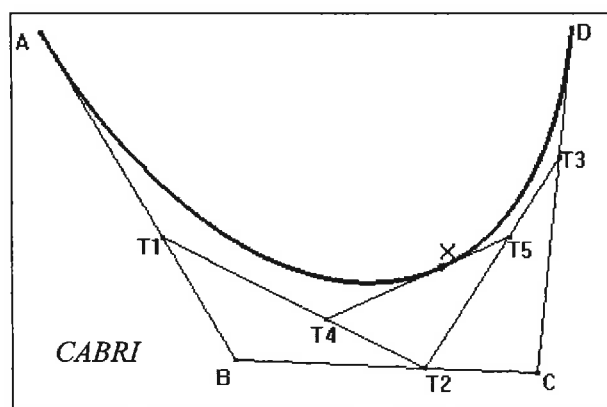
Variable t is called **parameter**. For each $t \in [0;1]$ we get one point of the Bézier curve. With $t = 0$ we get A , with $t = 1$ the point C . If t takes all numbers from $[0;1]$, we get the whole Bézier curve. If t runs from 0 to 1, X runs on the curve from A to C .

In the same way we can define a Bézier curve of degree 3.

Bézier-curve of degree 3

The **Bézier polygon** $ABCD$ and the **Bézier points** A, B, C and D define the curve.

We get the point X of the curve with the following proportions of lengths of segments:



$$\overline{AT_1} : \overline{AB} = \overline{BT_2} : \overline{BC} = \overline{CT_3} : \overline{CD} = \overline{T_1T_4} : \overline{T_1T_2} = \overline{T_2T_5} : \overline{T_2T_3} = \overline{T_4X} : \overline{T_4T_5} = t : 1$$

All what was said above about the parameter of a Bézier curve of degree 2 is also valid for a Bézier curve of degree 3.

Construction of Bézier Curves with CABRI

First we create two segments AB and CD (note the “direction” of the segments in *CABRI*).

T_1 is a Point on Object (can be moved on the segment between A and B). This point is the point of division for the Bézier curve.

The screen shot shows the transfer of T_1 to the segment CD . This gives the point T_2 .

This construction can be saved in a macro. Only with the segments AB , CD and T_1 the point T_2 can be constructed.

Now we use our macro to create a Bézier curve of degree 2.

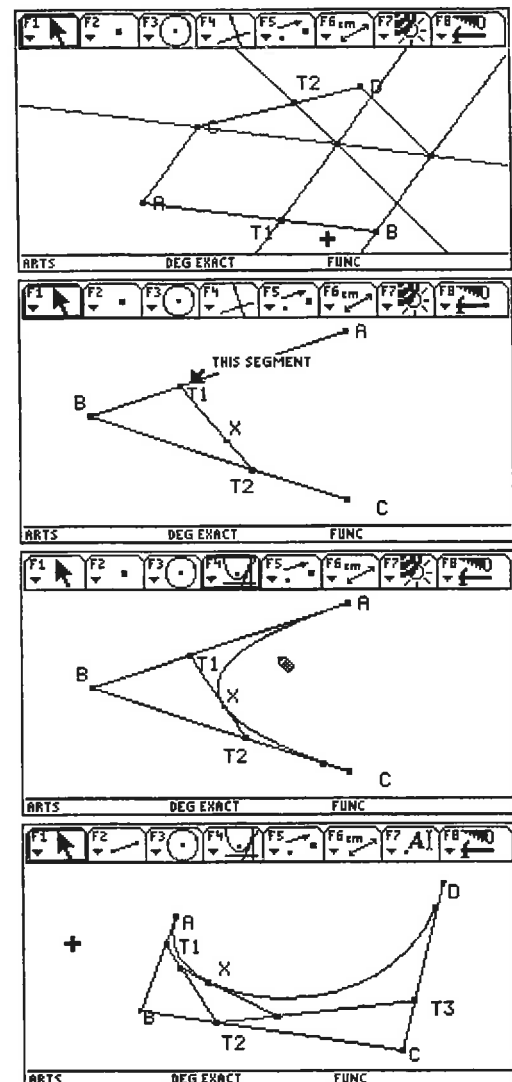
First we draw AB , BC and T_1 . Then T_1 is transferred to BC , giving T_2 and further to T_1T_2 to obtain X .

The curve is created as locus of the point X depending on the motion of T_1 .

In the same way we can build a Bézier curve of degree 3 (see the figure).

An important result of this construction:

The Bézier curve depends only on the Bézier points (the Bézier polygon). For example: These are the points A , B , C and D for the Bézier curve of degree 3. If these points are moved the curve changes its form.



For practical use it is not necessary to create Bézier curves of higher degree than 3. Complex figures can be built with more curves.

Algebraic representation of Bézier curves

We use basic knowledge to find formulae for our curves. We don't need much, the proportions of the segment lengths is sufficient.

Algebraic representation of a 2nd degree Bézier curve

$$PT : PQ = t : 1 = AT_1 : AB = BT_2 : BC = T_1X : T_1T_2$$

We want to find an expression for point X depending on the points A , B , C and the parameter t .

We use vectors and find easily T_1 from A and B and then in the same way we derive T_2 from B and C :

$$T_1 = A + t \cdot \overrightarrow{AB} = A + t \cdot (B - A) = (1 - t) \cdot A + t \cdot B \quad \text{and} \quad T_2 = (1 - t) \cdot B + t \cdot C$$

and finally we find: $X = (1 - t) \cdot T_1 + t \cdot T_2 = (1 - t) \cdot [(1 - t) \cdot A + t \cdot B] + t \cdot [(1 - t) \cdot B + t \cdot C]$

$$X = (1 - t)^2 \cdot A + 2(1 - t) \cdot t \cdot B + t^2 \cdot C.$$

Now we have found an expression for X and can define the Bézier curve of degree 3 by using points A, B, C and D .

Algebraic representation of a 3rd degree Bézier curve

According to the CABRI-figure we find the following expression:

$$X = (1 - t)^3 \cdot A + 3(1 - t)^2 \cdot t \cdot B + 3(1 - t) \cdot t^2 \cdot C + t^3 \cdot D$$

Tangents to Bézier curves

If the students are able to work with derivatives of parametric defined curves it is possible for them to find the tangents. The figures given above give the impression that the lines AB and BC are tangents of the second degree curve and AB and CD seem to be tangents of the 3rd degree curve. We can prove this using the first derivatives.

$$X(t) = (1 - t)^2 A + 2(1 - t)tB + t^2 C \rightarrow X'(t) = -2(1 - t)A + 2(1 - 2t)B + 2tC$$

$$X'(0) = -2A + 2B = 2(B - A) = \overrightarrow{AB}$$

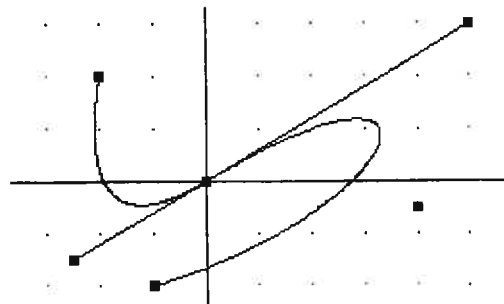
$$X'(1) = -2B + 2C = 2(C - B) = \overrightarrow{BC}$$

So we see immediately that lines AB, BC and the tangents have the same direction.

Now it is not very difficult to attach two Bézier curves having a “smooth” (= differentiable) transition.

They must have one endpoint in common and two “control points” must lie on a line.

The figure shows a curve composed of a 2nd degree - and a 3rd degree Bézier curve.



Bézier curves with DERIVE

We define the functions – including parameter t as last argument for later use - and then plot our first Bézier curves ($0 \leq t \leq 1$).

$$\text{bez2}(a_, b_, c_, t) := (1 - t)^2 \cdot a_ + 2 \cdot (1 - t) \cdot t \cdot b_ + t^2 \cdot c_$$

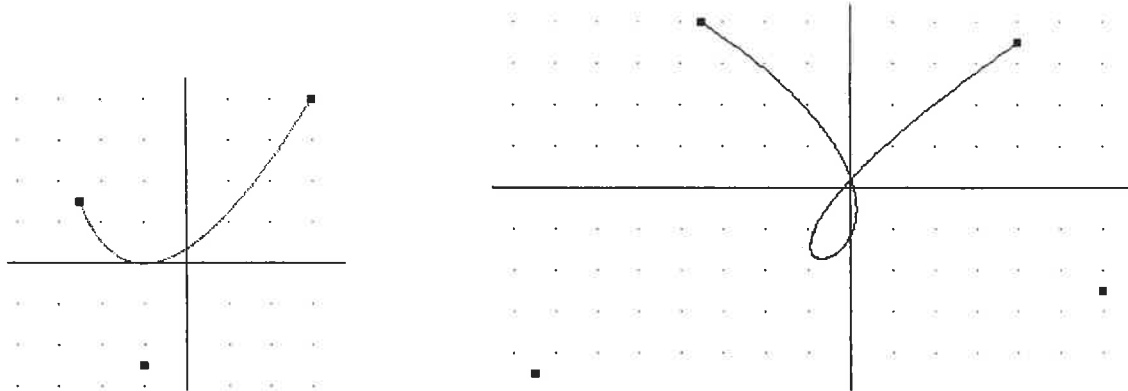
$$[p1 := [-5, 3], p2 := [-2, -5], p3 := [6, 8]]$$

$$\text{bez2}(p1, p2, p3) = [5 \cdot t^2 + 6 \cdot t - 5, 21 \cdot t^2 - 16 \cdot t + 3]$$

$$\text{bez3}(a_, b_, c_, d_, t) := (1 - t)^3 \cdot a_ + 3 \cdot (1 - t)^2 \cdot t \cdot b_ + 3 \cdot (1 - t) \cdot t^2 \cdot c_ + t^3 \cdot d_$$

$$[p1 := [8, 7], p2 := [-15, -9], p3 := [12, -5], p4 := [-7, 8]]$$

$$\text{bez3}(p1, p2, p3, p4) = [-96 \cdot t^3 + 150 \cdot t^2 - 69 \cdot t + 8, -11 \cdot t^3 + 60 \cdot t^2 - 48 \cdot t + 7]$$



What we can now with **DERIVE 6**

In DERIVE 6 we can use the Slider Bars to show in a very impressive way the influence of the "free" or "control" point B (of the triple A, B, C for 2nd degree Beziér curves) and the two variable ("control-") points B and C for the 3rd degree BC.

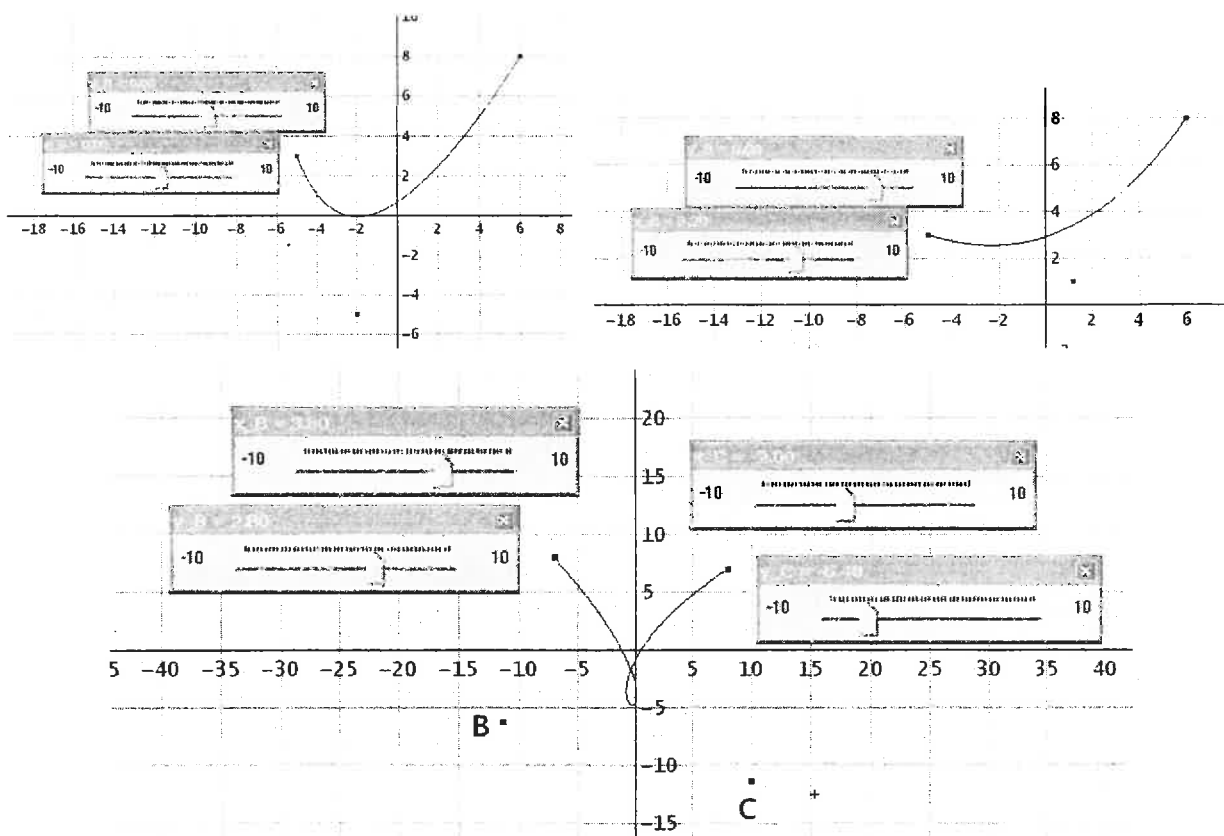
$[A := [-5, 3], B := [-2 + x_B, -5 + y_B], C := [6, 8]]$

bez2(A, B, C)

$[A := [-5, 3], B := [-2 + x_B, -5 + y_B], c := [12 + x_C, -5 + y_C], D := [-7, 8]]$

bez3(A, B, C, D)

x_B, y_B, x_C and y_C are parameters to move points B and C around using the slider bars. Here you can find some positions (initial position and another one):



(More about the exciting new features of DERIVE 6 on page 44.)

Modelling of figures with Bézier-curves

Modelling the letter "D"

It's obvious that there cannot be only one "solution" in designing a letter. Everybody can create his own font. For example it is possible to build the letter "D" with two Bézier curves. This is an important task for students to learn modelling figures realising their own ideas.

First of all we have to find a consistent way to include line segments. *Line segments are Bezier curves of degree 1.*

Two examples for "D" with *DERIVE*

d1, the "D" composed of two Bézier curves:

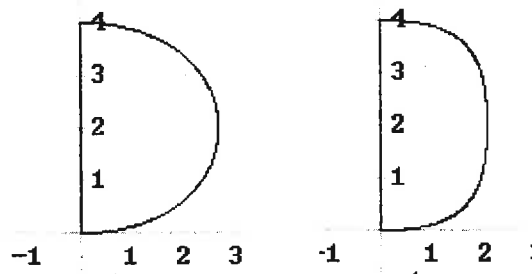
```
bez1(a_, b_, t) := (1 - t) · a_ + t · b_
d1 := [bez3([0, 0], [3.5, 0], [3.5, 4], [0, 4]), bez1([0, 0], [0, 4])]
```

and now we use three Bézier curves to create another "D", d2:

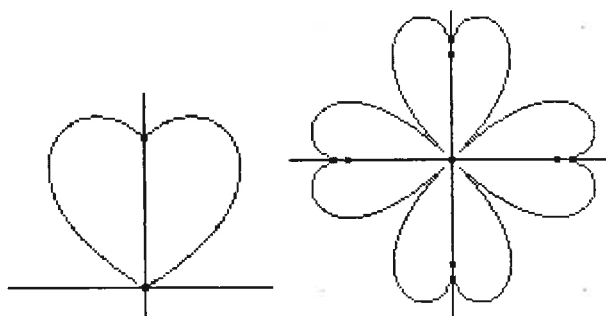
```
d2 := [bez3([0, 0], [1.5, 0], [2, 0.5], [2, 2]), bez3([0, 4], [1.5, 4], [2, 3.5],
[2, 2]), bez1([0, 0], [0, 4])]
```

... and we compare the results:

(left is d1 and right is d2).



These are wonderful open ended problems, because there are many "solutions". Everybody can find his own design.



It is an experience that students are very creative in modelling figures.

Transformations of Bézier-curves

In the first example the letter was written in the font size 8 and then enlarged with a graphic program. This method is not suitable to create letters in different sizes for use in a text processing system.



It's better to change the size of letters directly in the word processing system. The letters will be calculated for the new size and keep their form.

An advantage of Bézier curves is that these calculations are easy to perform.

D

Once more we will use the letter "D" to show how to change its size with *DERIVE*.

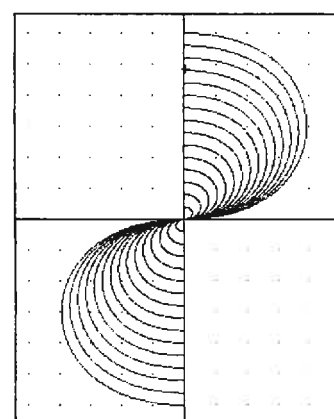
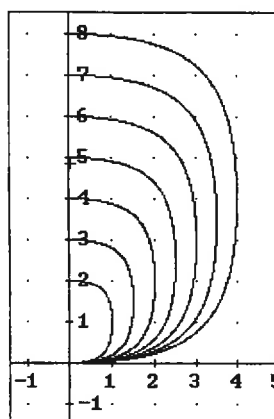
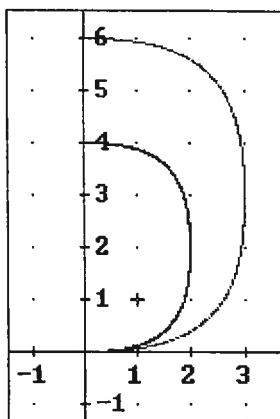
Variable k has the initial value $k = 1$. Using the VECTOR-command we can apply several k s. We multiply the coordinates in the expressions with k and as result we receive the letter in its new size but unchanged in its appearance.

As we will use bez1, bez2 and bez3 to create Bézier curves it is recommended to collect these three functions in a utility file.

Stretching the "D" (or any other Bézier construct, of course)

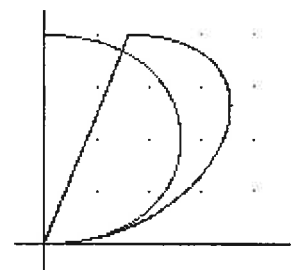
The list of vectors, which turns out to be a matrix is multiplied by the stretch- or dilatation factor.

```
stretch(m_, k_) := k_·m_
stretch(d2, 1.5)
VECTOR(stretch(d2, k), k, 0.5, 2, 0.25)
VECTOR(stretch(d1, k), k, -1.5, 1.5, 0.1)
```

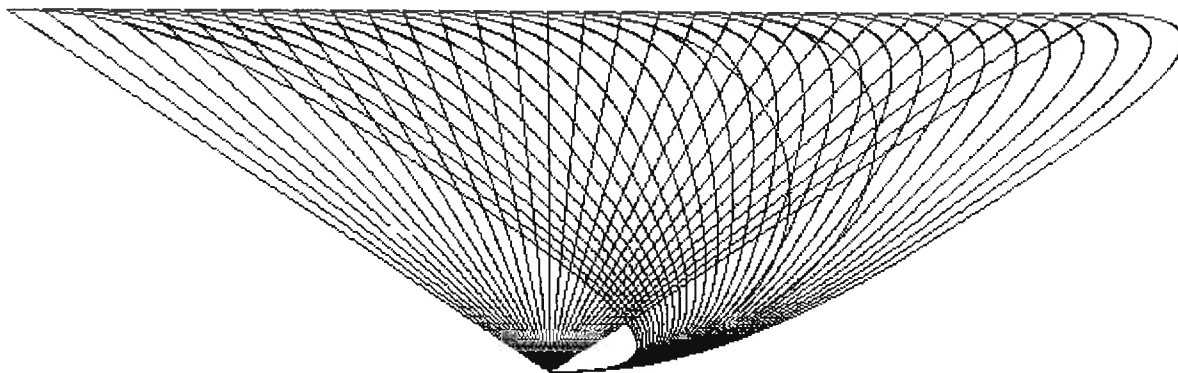


How to make "Italics" – Shearing (in German: Scherung)

```
shear(m_, k_) := VECTOR([v_1 + k_·v_2, v_2], v_, m_)
[d1, shear(d1, 0.4)]
```



VECTOR(shear(d2, k), k, -1.5, 1.5, 0.1)



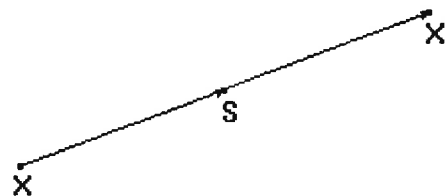
Reflection with respect to a point

In the figure the point X shall be reflected on S .
Result is the reflected point X' . This can be calculated with vectors:

$$\overrightarrow{XS} = \overrightarrow{SX'}$$

$$S - X = X' - S$$

$$X' = 2S - X$$



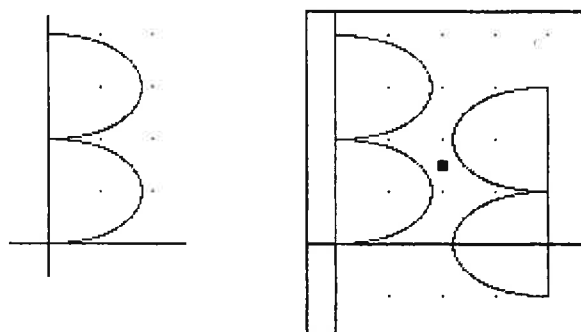
We produce a "B" and plot it with its reflection wrt [2,1.5]

(m_ = object, pt_ = reflection center)

reflp(m_, pt_) := VECTOR(2 * pt_ - v_, v_, m_)

b1 := [bez3([0, 0], [2.4, 0], [2.4, 2], [0, 2]), bez3([0, 2], [2.4, 2], [2.4, 4], [0, 4]), bez1([0, 0], [0, 4])]

reflp(b1, [2, 1.5])



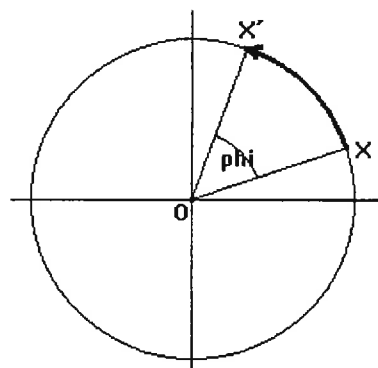
Rotation of a Bézier curve

The point $X=(x,y)$ rotates giving point $X'=(x',y')$.
Rotation center is $O=(0,0)$.

We know the following expressions for the transformation:

$$x' = \cos \varphi \cdot x - \sin \varphi \cdot y$$

$$y' = \sin \varphi \cdot x + \cos \varphi \cdot y$$



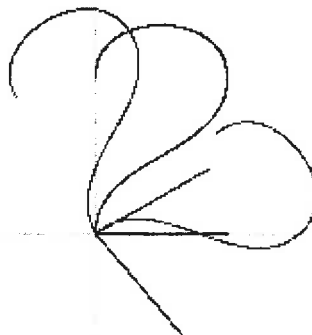
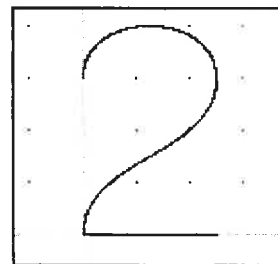
For demonstration of the rotation we use the digit "2" as example. We compose the figure of two Bézier-curves of degree 3 and one curve of degree 1.

(m_ = object, w_ = rotation angle)

```
rot(m_, w_) := VECTOR([v_1 * COS(w_) - v_2 * SIN(w_), v_1 * SIN(w_) + v_2 * COS(w_)], v_,
  m_)
```

```
two := [bez3([0, 0], [0, 1.5], [2.5, 1.5], [2.5, 3]), bez3([0, 3], [0, 13/3], [2.5,
  13/3], [2.5, 3]), bez1([0, 0], [2.5, 0])]
```

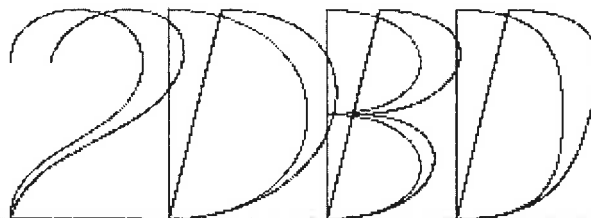
```
[rot(two, pi/6), rot(two, -5*pi/18)]
```



Shift by a vector

```
shift(m_, s_) := VECTOR(v_ + s_, v_, m_)
text := APPEND(d1, shift(b, [3, 0]), shift(d2, [5.5, 0]), shift(two, [-3, 0]))
shear(text, 0.25)
```

Using all our characters designed so far, we write a "word" (2DBD) and then transform it into Italics:



Length of Bézier curves

We show a method how to measure the length of a Bézier-curve approximately.

Let's divide the whole curve in little parts and replace these parts of the curve by line segments. So we get a polygon. Measuring the length of the polygon we obtain an approximated value of the length of the curve.

We want to find an approximative value for the length of the following Bézier curve of degree 3.

$cv(t) := bez3([-10, 9], [15, -20], [7, 20], [-5, -6], t)$

$dist(p1, p2) := \sqrt{(p2 - p1) \cdot (p2 - p1)}$

Notation := Decimal

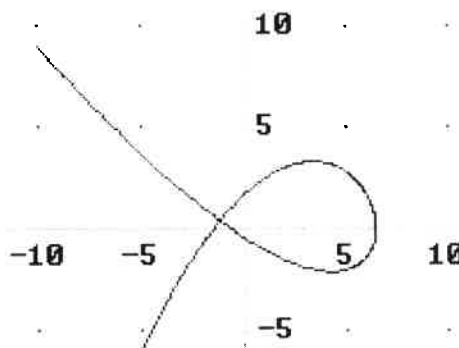
NotationDigits := 10

$\sum_{i=0}^{99} dist\left(cv\left(\frac{i}{100}\right), cv\left(\frac{i+1}{100}\right)\right)$

39.80989599

$\sum_{i=0}^{999} dist\left(cv\left(\frac{i}{1000}\right), cv\left(\frac{i+1}{1000}\right)\right)$

39.81310875



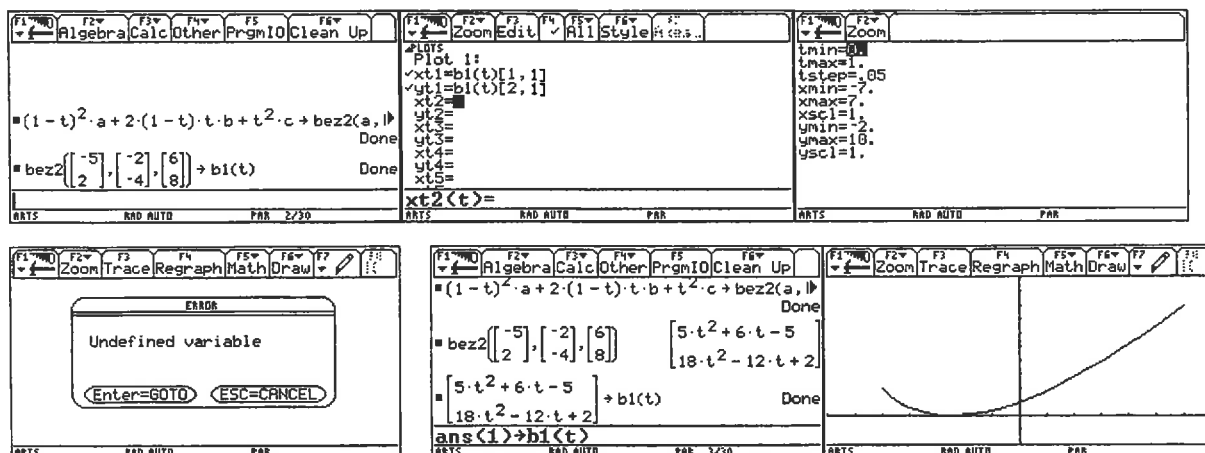
Finally we use one of the many very helpful USER-files of *DERIVE*, *INT_APPS.MTH* in order to find the analytic answer:

$$PARA_ARC_LENGTH(cv(t), t, 0, 1) = 3 \cdot \sqrt{2} \int_0^1 \sqrt{9533 \cdot t^4 - 20544 \cdot t^3 + 16340 \cdot t^2 - 5652 \cdot t + 733} \, dt$$

$$PARA_ARC_LENGTH(cv(t), t, 0, 1) = 39.81314121$$

Beziér curves on the VOYAGE 200

We show the representation of a 2nd degree Beziér-curve. First of all we define the curve in the same way as we did with *DERIVE*. $(1-t)^2 \cdot a + 2(1-t) \cdot t \cdot b + t^2 \cdot c \rightarrow bez2(a, b, c)$. We save one special curve as $b1(t)$, then in the **MODE** we choose for Graph:2:PARAMETRIC and define $xt1(t)$ and $yt1(t)$ by the components of $b1(t)$. After setting appropriate **[WINDOW]**-values we want to plot the curve – but we fail!!



You can see how to do that the procedure works: First simplify the expression and then define $b1(t)$.

Summary

Bézier curves can be interesting for mathematics education and provide a welcome extension of the subject "curves". It is an important example of applied mathematics. In this field the computer is an essential tool.

Students shall understand the foundations of these curves and also make practical applications. They shall get knowledge to work with CAS or dynamic geometric software like CABRI.

Literature

Bär G.; Geometrie, Einführung in die analytische und konstruktive Geometrie; B.G. Teubner Verlagsgesellschaft Stuttgart, Leipzig 1996

Chambers P., Rockwood A.; Interactive Curves and Surfaces; Morgan Kaufmann Publishers, Inc., 1996

Grabinger B.; Wie Microsoft Kurven biegt: Mathematik mit Paintbrush; Praxis der Mathematik 1/41; 1999

Pottmann H., Wallner J.; Computational Line Geometry; Springer-Verlag Berlin, Heidelberg New York; 2001

Scheu G., An Approach to the Bézier Curves with DERIVE, DERIVE Newsletter #19, 1995

Schlöglhofer F., Schoberleitner F., Bézier Kurven, Ideen zur Geometrie, Skriptum PI-OÖ, 2002

Bézier Surfaces (by Josef Böhm)

It is a demanding challenge to transfer this idea into the 3rd dimension. As Franz wrote in his introduction these curves were introduced for designing cars – and cars are 3D-objects. So let's try this.

We define 4 groups of Bézier points, each of them defining a Bézier curve.

$$\left[\begin{array}{l} c_0 := \begin{bmatrix} 15 & -10 & 10 \\ 15 & -5 & 0 \\ 15 & 5 & 0 \\ 15 & 10 & 10 \end{bmatrix}, c_1 := \begin{bmatrix} 10 & -8 & 7 \\ 10 & -5 & -3 \\ 10 & 5 & -3 \\ 10 & 8 & 7 \end{bmatrix}, c_2 := \begin{bmatrix} 0 & -5 & 3 \\ 0 & -3 & 2 \\ 0 & 3 & 2 \\ 0 & 12 & 6 \end{bmatrix}, c_3 := \begin{bmatrix} -10 & -12 & 6 \\ -10 & -5 & 0 \\ -10 & 5 & 0 \\ -10 & 6 & 3 \end{bmatrix} \end{array} \right]$$

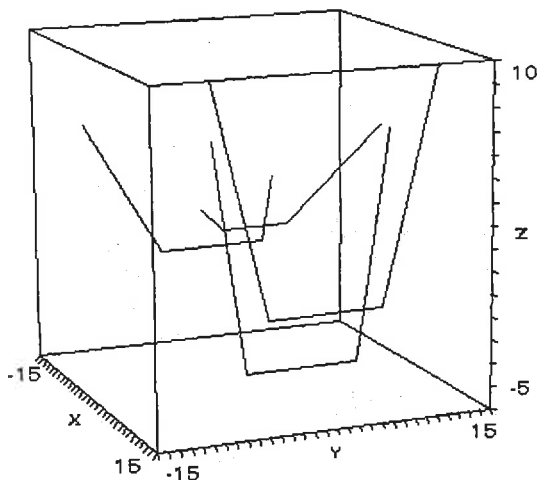
Using these points we create these 4 Bézier-curves.

$$\begin{aligned} p_0(t) &:= \sum_{i=0}^3 \text{COMB}(3, i) \cdot t^i \cdot (1-t)^{3-i} \cdot c_{0,i} \\ p_1(t) &:= \sum_{i=0}^3 \text{COMB}(3, i) \cdot t^i \cdot (1-t)^{3-i} \cdot c_{1,i} \\ p_2(t) &:= \sum_{i=0}^3 \text{COMB}(3, i) \cdot t^i \cdot (1-t)^{3-i} \cdot c_{2,i} \\ p_3(t) &:= \sum_{i=0}^3 \text{COMB}(3, i) \cdot t^i \cdot (1-t)^{3-i} \cdot c_{3,i} \end{aligned}$$

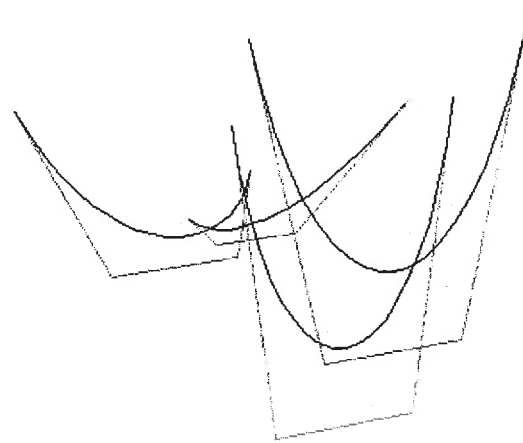
For each fixed value t we have another group of 4 points which can be used again as 4 Bézier points for Bézier curves depending on a 2nd parameter s . So we obtain finally an expression containing two parameters, t and s , both of them running from 0 to 1. And this turns out to be the parameter representation of the Bézier surface defined by 16 points. The figures on the next page should illustrate the procedure.

```
t_fam(t) := [p0(t), p1(t), p2(t), p3(t)]
```

```
surf(s, t) :=  $\sum_{k=0}^3 \text{COMB}(3, k) \cdot s^k \cdot (1-s)^{3-k} \cdot (t\_fam(t))_{k+1}$ 
```

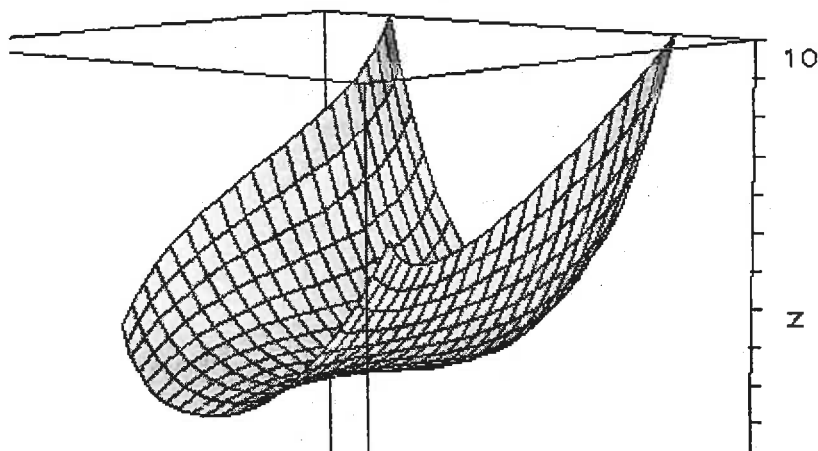


The control polygons



and the respective 3rd degree Bézier curves

Here is the final result:



One could go on and produce a function which delivers this surface in one step.

```
grid := APPEND([c0], [c1], [c2], [c3])
```

```

$$\sum_{i=0}^3 \sum_{k=0}^3 \text{COMB}(3, k) \cdot s^k \cdot (1-s)^{3-k} \cdot \text{COMB}(3, i) \cdot t^i \cdot (1-t)^{3-i} \cdot \text{grid}_{k+1, i+1}$$

```

$$\left[5 \cdot (s^3 - 3 \cdot s^2 - 3 \cdot s + 3), -s^3 \cdot (41 \cdot t^3 - 21 \cdot t^2 - 15 \cdot t + 11) + 3 \cdot s^2 \cdot (17 \cdot t^3 - 15 \cdot t^2 + 3 \cdot t + 1) - 6 \cdot s \cdot (2 \cdot t^3 - 3 \cdot t^2 + 3 \cdot t - 1) - 5 \cdot (2 \cdot t^3 - 3 \cdot t^2 - 3 \cdot t + 2), -s^3 \cdot (12 \cdot t^3 - 69 \cdot t^2 + 69 \cdot t - 8) + 3 \cdot s^2 \cdot (3 \cdot t^3 - 27 \cdot t^2 + 27 \cdot t - 1) - 9 \cdot s + 10 \cdot (3 \cdot t^2 - 3 \cdot t + 1) \right]$$

I am very grateful for a fruitful meeting with a Swiss colleague, who showed me Bézier surfaces. She wrote a wonderful script "Auch ein Teppich hat Gefühle" – "Even a Carpet has emotions" –, containing many interesting tasks and problems which are connected with Bézier-curves^[1].

^[1] Baoswan Dzung Wong, *Bézierkurven*, Orell Füssli Verlag, ISBN 3-280-04021-3

Lindenmayer Systems

Josef Böhm

In DNL#50 I posed a challenge for a string – rewriting – procedure by iterated replacing of characters in a string by other strings. Some of you solved the problem, many thanks. Aristide Lindenmayer used this technique to describe growth-processes of plants.

The first figure shows the evolution of the initial string "X" by substituting "X" by "X+Y+" and "Y" by "-X-Y". Steps 0 to 4 are created. Check it, please.

```
VECTOR([lindsys(90, [X, Y], [X+Y+, -X-Y], X, k_), k_, 0, 4)
```

```

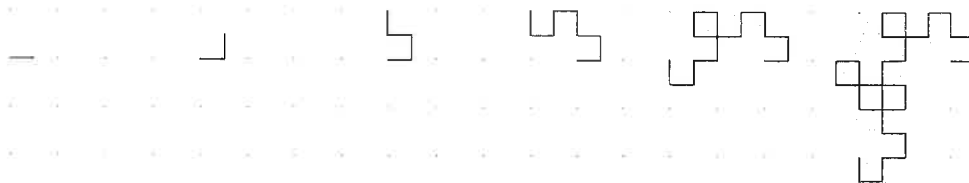
      X
    X+Y+
  X+Y+-X-Y+
X+Y+-X-Y+-X-Y+-X-Y+
X+Y+-X-Y+-X-Y+-X-Y+-X-Y+-X-Y+-X-Y+

```

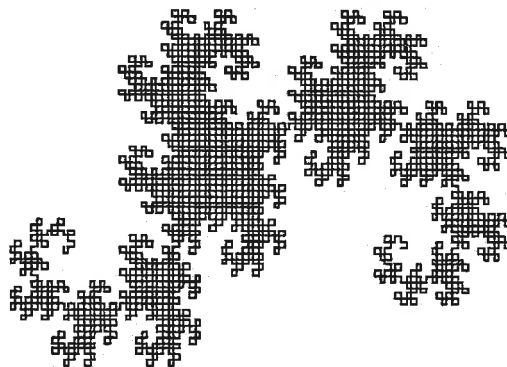
Now we will interpret the strings as commands for a "turtle". "X" and "Y" brings the turtle one step forward, "+" and "-" gives order to make a turn in positive or negative direction by a given angle (eg. 90°).

I extended function lindsys to "translate" the strings into movements and plotting the trace of the turtle (steps 0 – 5 and the result after 12 iterations give the next figures):

```
VECTOR(lindsys(90, [X, Y], [X+Y+, -X-Y], X, k, 0, [-24 + 8 * k, 0]), k, 0, 5)
```



```
lindsys(90, [X, Y], [X+Y+, -X-Y], X, 12)
```



The final version of the lindsys-program, which meets all my expectations is

```
lindsys4(angle, base, transf, startstr, iteration, startdir, startpt)
```

angle = turning angle for the turtle (in degrees), eg. 60

base = list of letters which are to be replaced, eg ["F"]

transf = list of substitutions for each member of base, eg. ["F+F--F+F"]

startstr = initial string, eg "F--F--F"

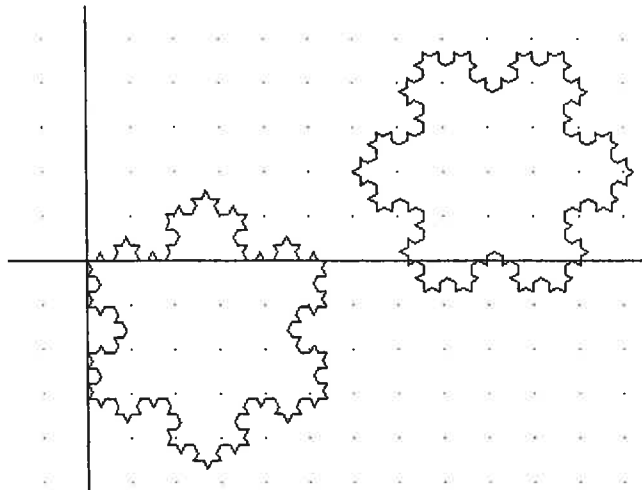
startdir = direction of the first step (degrees), by default 0

startpt = position of the starting point, by default [0, 0] Which figure are we creating now?

`lindsys4(60, ["F"], ["F+F--F+F"], "F--F--F", 3)` (Don't forget the quotes " !)

`lindsys4(60, [F], [F+F--F+F], F--F--F, 3)`

`lindsys4(60, [F], [F+F--F+F], F--F--F, 3, 30, [30, 10])`



Having visited well known and famous "Koch Island" we continue our journey to "Gosper Island":

`lindsys4(60, [F], [F-F+F], F-F-F-F-F-F, 5)`

The "Sierpinski Triangle" cannot be missed in a contribution treating fractals.

It appears in many shapes (here it is generated first by triangles and then by trapeziumlike polygons):

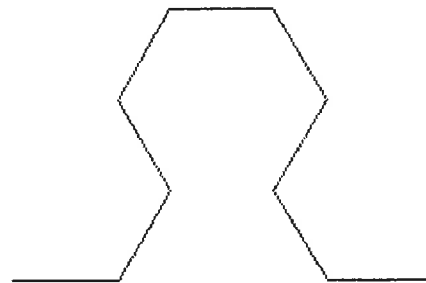
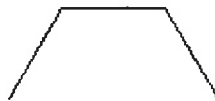
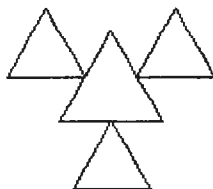
`lindsys4(120, [F, X], [FF, -FXF+FXF+FXF-], X, 2)`

`lindsys4(120, [F, X], [FF, -FXF+FXF+FXF-], X, 6)`

`lindsys4(60, [F, G], [+G-F-G+, -F+G+F-], F, 1)`

`lindsys4(60, [F, G], [+G-F-G+, -F+G+F-], F, 2)`

`lindsys4(60, [F, G], [+G-F-G+, -F+G+F-], F, 8)`



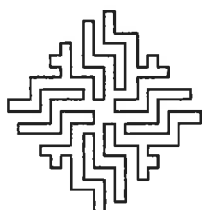
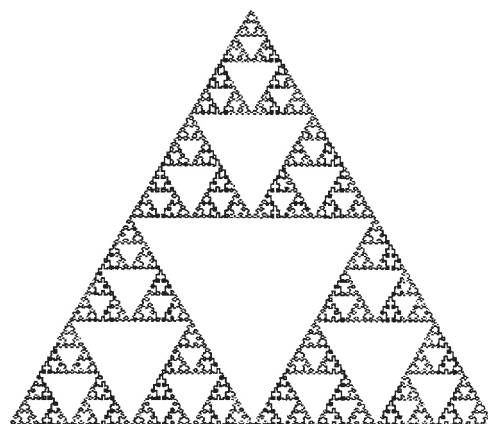
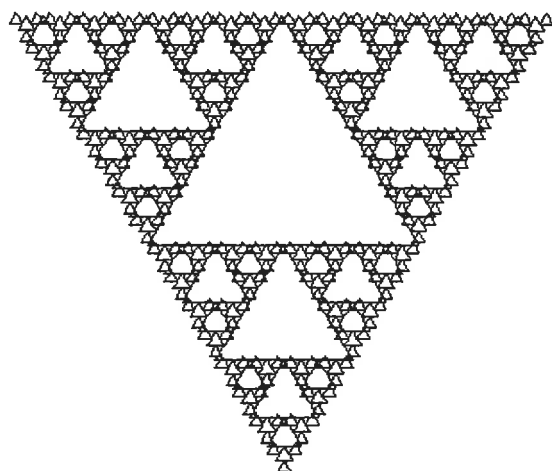
Just playing around with strings and angles gives remarkable and surprising results.

The graph of the next experiment can be found together with the promised Sierpinski Triangles on the next page.

`lindsys4(90, [F],`

`[F+F-F-FFF+FF-FF+F+FFF+FF-FFFF-F-FFF+FF-FFF+F+FFFF+FF-FFF-F-FF+FF-FFF+F+F-F],`

`F-F-F-F, 1)`



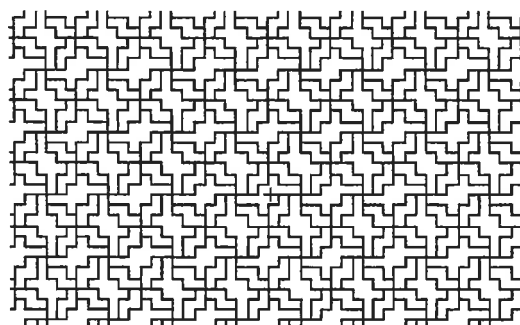
Try the 2nd stage of this pattern. You will be surprised.

The next rule gives another "KOCH Island".

```
lindsys(90, [F], [F+F-F-PF+F+F-F], F+F+F+F, 1)
```

This would be a wonderful tiling on my living room floor (quite expensive!).

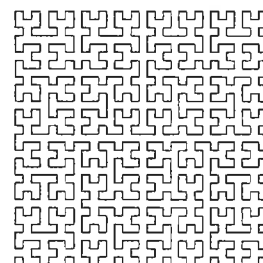
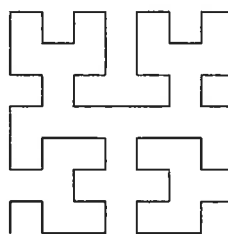
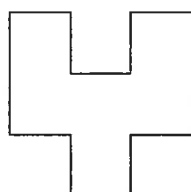
```
lindsys4(90, [X], [XF-F+F-XF+F+XF-F+F-X], F+XF+F+XF, 6)
```



Then I tried to produce the plane filling Hilbert curve

```
lindsys4(90, [L, R, F], [+RF-LFL-FR+, -LF+RFR+FL-, F], L, 1)
```

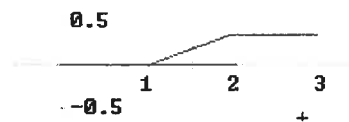
```
lindsys4(90, [L, R, F], [+RF-LFL-FR+, -LF+RFR+FL-, F], L, 2)
```



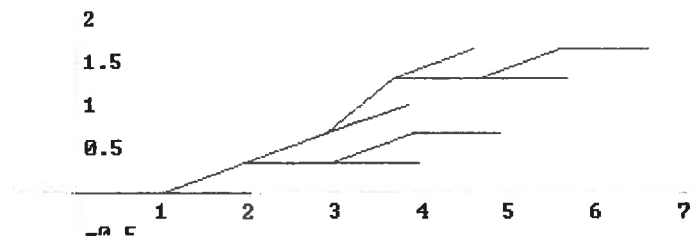
This was not so easy using my program and I had to introduce two letters for special purpose: L and R do not perform a step, they are only to be replaced by strings. So they appear as points and not as line segments.

Until now we don't have plants, because plants have branches. Lindenmayer used brackets to describe branches. "[" means that a branch starts under a constant angle and "]" marks the end of the branch, and gives order to go back to the starting point of the branch. I had to implement something like a "stack". And these are my plants:

```
lindsys3(20, [F], [F[F]+F-[F]], F, 1)
```



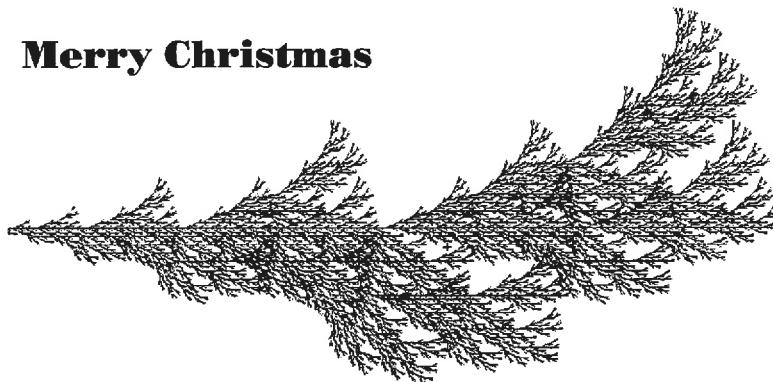
```
lindsys3(20, [F], [F[F]+F-[F]], F, 2)
```



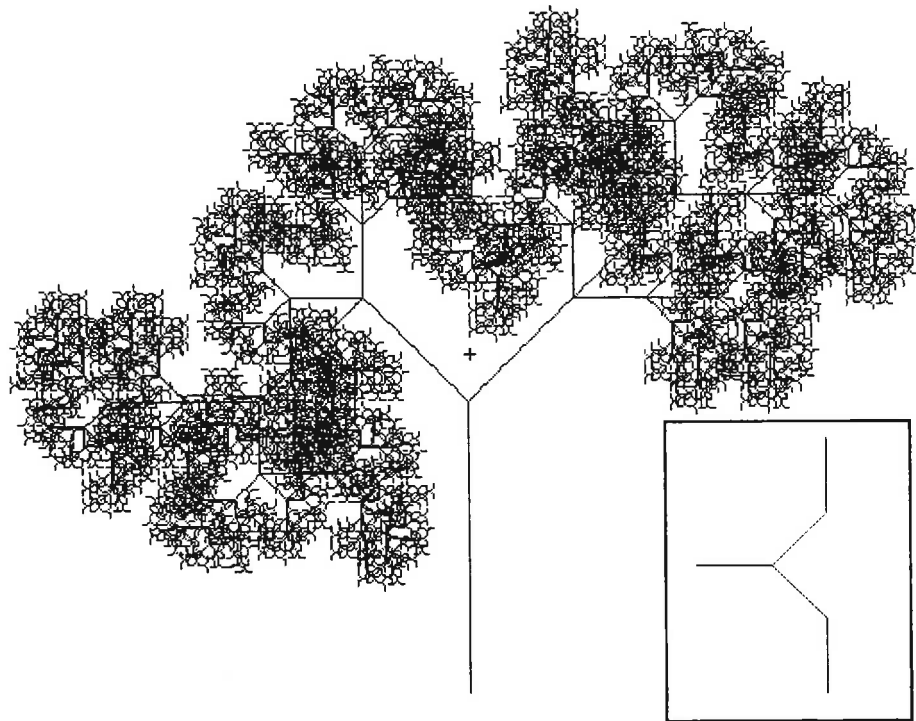
```
lindsys4(20, [F], [F[F]+F-[F]], F, 6)      lindsys4(20, [F], [F[F]+F-[F]], F, 6, 90)
```

```
lindsys4(20, [F], [F[-F]F[+F][F]], F, 6)
```

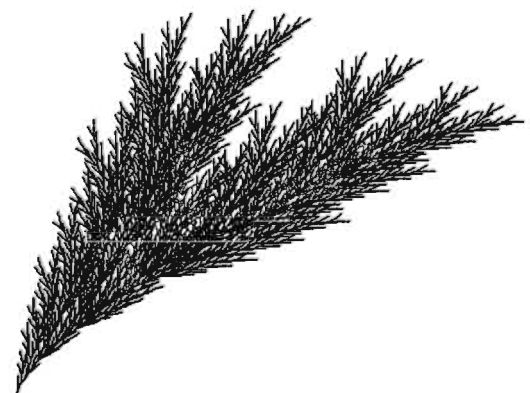
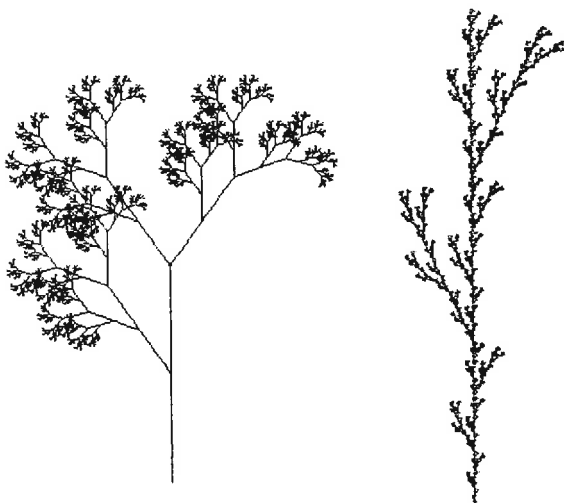
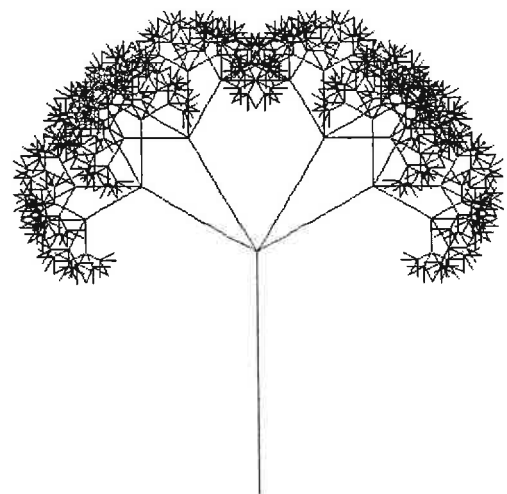
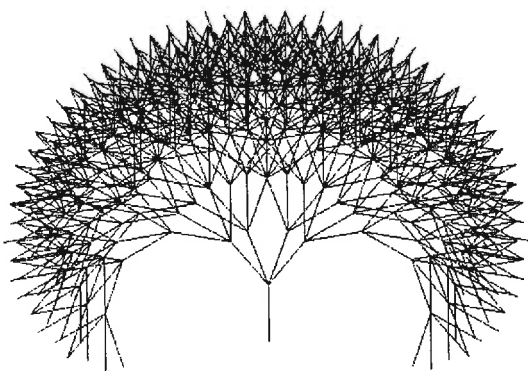
Merry Christmas



`lindsys4(45, [F, G, M], [G[+F+F][-F-F]+M, GG, M], F, 7, 90)`



A Visit in my Lindenmayer-Garden

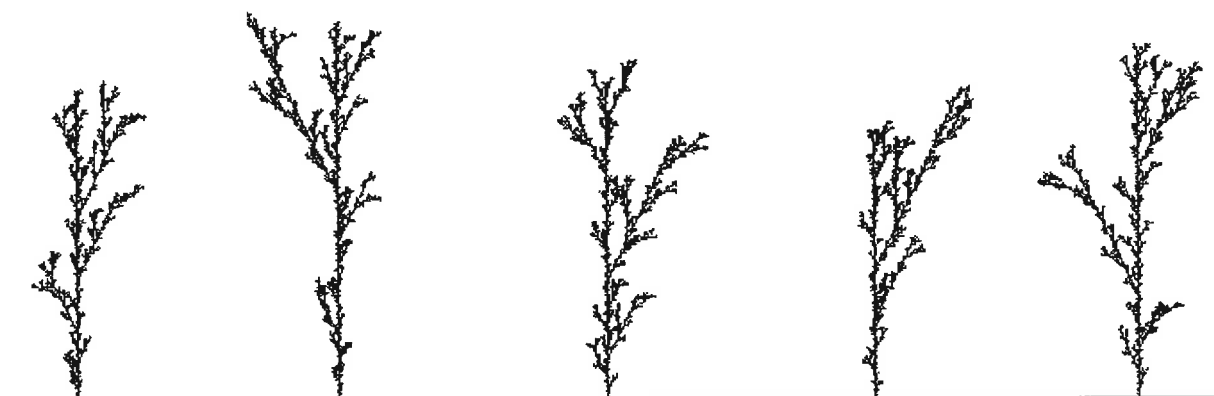


The graphs on page 27 impressed me very much and I invite you to create your own Lindenmayer Greenhouse.

All examples (and some more are part of the Lindenmayer-file on the diskette).

Then I wanted to have my plants not so deterministic and developed stochastic L-Plants. In the next function linstoch "F" is randomly replaced by one of the three strings "F[+F]F[-F]F", "F[+F]F" or "F[-F]F". You could change the function to have any special distribution. The argument 90 makes sure that my "Random Weeds" are growing vertically,

```
VECTOR(linstoch(25.7, [F], [F[+F]F[-F]F, F[+F]F, F[-F]F], F, 6, 90, [k, 0]), k,
-240, 240, 120)
```



My Lindenmayer Weed Meadow in Würmla

Another interesting investigation with the space-filling Hilbert Curve from above follows: we interpret the x-coordinates of the points (or the y-coordinates) as function of the point-numbers in the created pointlist. And then we plot the function graph.

I start with the curve of stage 6 (in black) and superimpose curve of stage 4 (red) – bottom left corner.

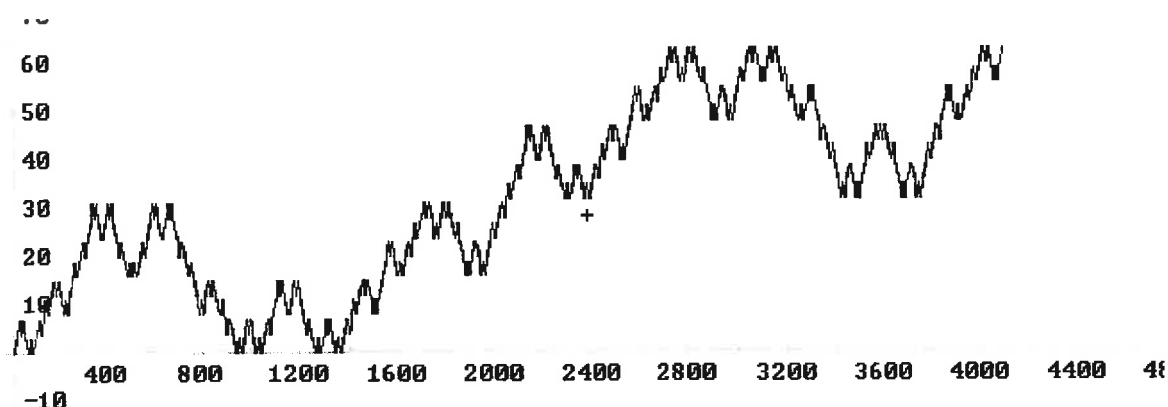
```
hilbert := lindsys4(90, [L, R, F], [+RF-LFL-FR+, -LF+RFR+FL-, F], L, 6)
```

```
DIM(hilbert) = 4096
```

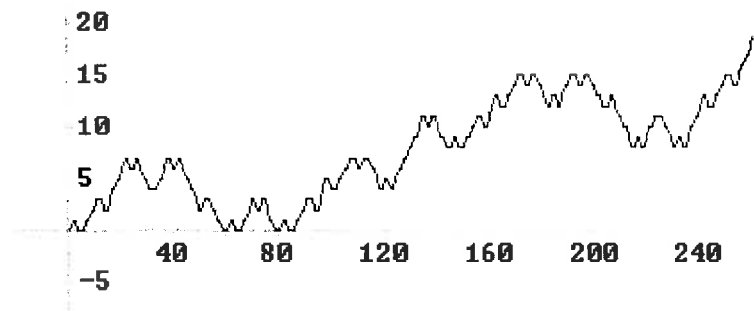
```
xhilb := VECTOR([k, hilbert_k,1], k, DIM(hilbert))
```

```
hilbert4 := lindsys2(90, [L, R, F], [+RF-LFL-FR+, -LF+RFR+FL-, F], L, 4)
```

```
xhilb4 := VECTOR([k, hilbert4_k,1], k, DIM(hilbert4))
```



The first four generations



Here again the fractal nature of this curve can be observed and we have a wonderful visualisation of self similarity. Finally I produce a "HILBERT Curtain" by transferring this space-filling curve into three dimensions.

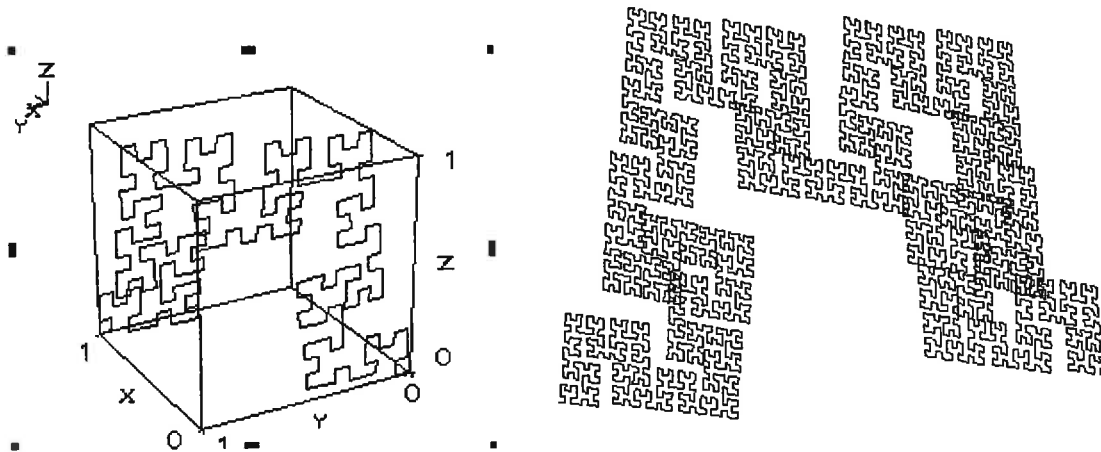
An appropriate scaling squeezes all points into the unit cube. x -coordinate is the number of the point, y - and z -coordinates in space are the x - and y -coordinates of the plane curve.

```
h4 := lindsys4(90, [L, R, F], [+RF-LFL-FR+, -LF+RFR+FL-, F], L, 4)
```

```
VECTOR( APPEND( [ [ -k / 256 ], [ 1 / 15 * h4_k ] ], k, 1, 256 )
```

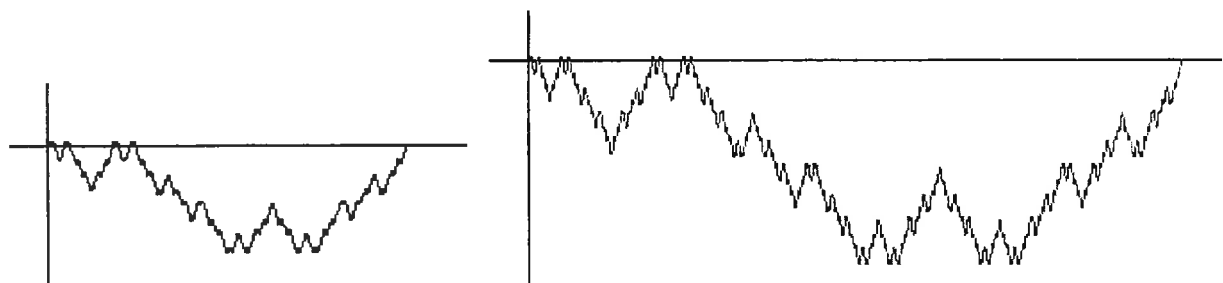
```
h6 := lindsys4(90, [L, R, F], [+RF-LFL-FR+, -LF+RFR+FL-, F], L, 6)
```

```
VECTOR( APPEND( [ [ -k / 4096 ], [ 1 / 63 * h6_k ] ], k, 1, 4096 )
```



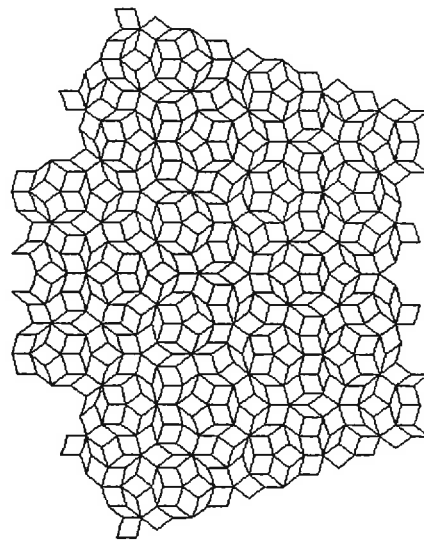
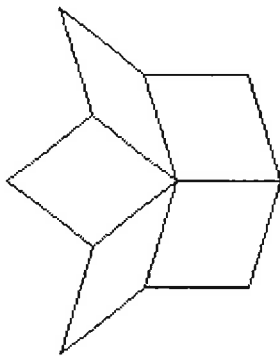
But take care! Calculating and plotting takes some time!!

Results from another plane filling curve (stage 3 and stage 4).

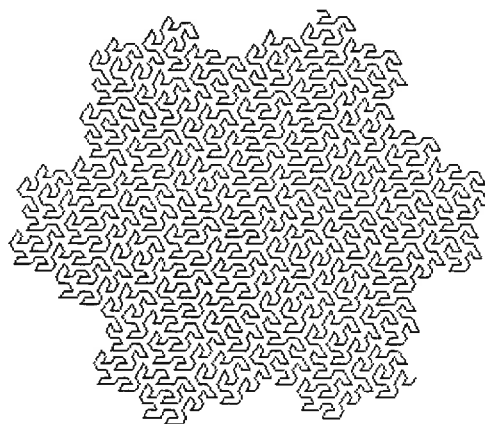
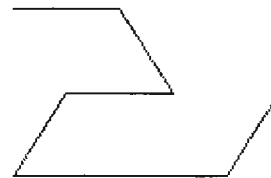
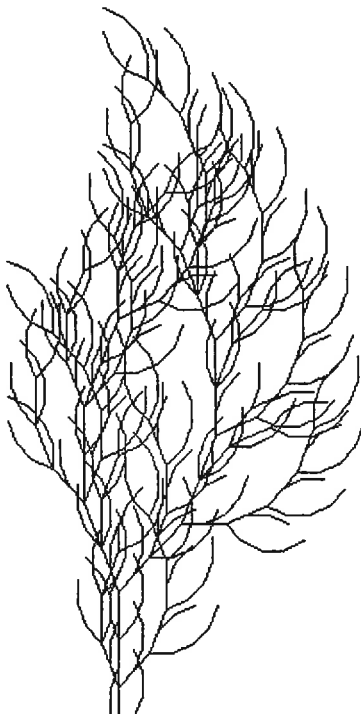


A rich resource for L-Systems (and for numerous other kinds of fractals) is the freeware program FRACTINT, version 20, which can be downloaded from the web. My tries are performed with my 3rd version of lindsys, lindsys4 works, too.

Penrose Tiling



```
lindsys3(36, [W, X, Y, Z], [YF++ZF---XF[-YF---WF]++, +YF---ZF[---WF---XF]+,
-WF++XF[+++YF++ZF]-, --YF+++WF[+ZF+++XF]---XF], +WF---XF---YF---ZF, 5)
```



```
lindsys3(360, [F], [FF-[-F+F+F+F]+[+F-F-F]], ++++F, 3)
```

```
lindsys3(60, [L, R], [FL-FR--FR+FL++PLFL+FR-, +FL-FRFR--FR-FL++FL+FR], FL, 4)
```

- [1] H. Lauwerier, *Fraktale verstehen und selbst programmieren*, Bd 2, Wittig 1992
- [2] J. Peitgens et al., *Fractals for the Classroom, Part II*, Springer 1992
- [3] T. Wegener et al., *Fraktale Welten (Fractal Creations)*, te-wi, 1992, (Waite Group, 1991)

Financial Mathematics I

Bond Price and Yield for Derive 5

MacDonald Phillips
phillipsm@gao.gov
donphillips@starpower.net
October 2003

The `Bond(set, mat, cpn, call, yld, price, cf, cal)` program will calculate the price or yield (as well as the coupon or redemption/call value) of just about any bond. The variables are:

- set = settlement date (entered as mm.ddyyyy)
- mat = maturity date (entered as mm.ddyyyy)
- cpn = coupon (annual coupon as a percent)
- call = redemption/call value (amount per \$100 of face value)
- yld = yield (annual yield as a percent)
- price = price (amount per \$100 of face value)
- cf = coupon frequency (default is 2 for semi-annual coupons, other values may be 1, 3, 4, 6, 12)
- cal = calendar type (default is 1 for Actual/Actual calendar; enter 2 for 30/360 day calendar)

The bond program, as well as the calendar programs, may be seen by clicking on Declare/Function Definition and selecting the appropriate program. The program to calculate the last and next coupon dates was adopted from UBONDS-Universal Bond Solver for the HP LX Palmtops by Tony Hutchins. This program uses the last-day-of-the-month convention for coupons, i.e., if the maturity date is on the last day of the month, the last and next coupon dates fall on the last day of the month also.

Most bonds have semi-annual coupons, thus the default of 2 for `cf`. And, in general, most government bonds use the Actual/Actual calendar for computing accrued interest while corporate, municipal, and U.S. Agency bonds generally use the 30/360 day calendar.

This program works well for typical bond values. It also works for "extreme" values of yield and price, including negative yields. By extreme yields, I mean yields as low as `ABS(.000001%)`. All computed values are rounded to 6 decimal places and therefore any computed yield less than `.0000005%` is rounded to zero. If you find any cases that the program cannot compute, please let me know so I can try and adjust the program. And please include the variable values you used. And, as usual, any suggestions for improvement are always welcome.

Note: When solving for any of the variables always Simplify, never Approximate. The calendar programs need to run in Exact mode.

p32	MacDonald Phillips: Bond Price and Yield	D-N-L#52
-----	--	----------

Example 1:

What price should you pay on August 10, 2003, for a 6.75% U.S. Treasury Bond that matures on May 1, 2018, if you want a yield of 8.375% compounded semi-annually? The redemption value at maturity is 100, the calendar basis is Actual/Actual and coupons are paid semi-annually. There is no need to enter anything for *cf* or *cal* since semi-annual and Actual/Actual are the default values.

Bond(8.102003, 5.012018, 6.75, 100, 8.375, price)

Calendar	Actual/Actual
Settlement Date	Sunday 8.102003
Maturity Date	Tuesday 5.012018
Coupon (%)	6.75
Redemption/Call Value	100
Yield (%)	8.375
Price	86.376543
Accrued Interest	1.852582
Last Coupon Date	5.012003
Next Coupon Date	11.012003
Number of Coupons	30
Days from LCPN to NCPN	184
Days from LCPN to SET	101
Accrual Fraction	0.54891304347826

The output matrix gives you a lot of information. Most of it is self-explanatory. The calculated price is \$86.38 per \$100 of face value. The accrued interest is \$1.85 per \$100 of face value. The number of days from the last coupon date to the next coupon date is 184. The number of days from the last coupon date to the settlement date is 101. Thus the accrual fraction, for calculating accrued interest is $101/184 = .548913...$ It should be noted that in practice a settlement date would never fall on a weekend; it would always fall on a business day.

Example 2:

What is the yield of the above bond if the price is 88.25?

Bond(8.102003, 5.012018, 6.75, 100, yield, 88.25)

Coupon (%)	6.75
Redemption/Call Value	100
Yield (%)	8.131156
Price	88.25
Accrued Interest	1.852582

The calculated yield is 8.13%.

Example 3:

What is the price of a 6% corporated bond purchased on May 2, 2003, and maturing on March 3, 2022, if the yield is 5.7%? If the bond is callabl on March 3, 2006, for a call call value of 102.75, what is the yield-to-call? Use the 30/360 day calendar and semi-annual coupons.

First, calculate the price.

Bond(5.022003, 3.032022, 6, 100, 5.7, price, 2, 2)

Calendar	30/360
Settlement Date	Friday 5.022003
Maturity Date	Thursday 3.032022
Coupon (%)	6
Redemption/Call Value	100
Yield (%)	5.7
Price	103.427958
Accrued Interest	0.983333
Last Coupon Date	3.032003
Next Coupon Date	9.032003
Number of Coupons	38
Days from LCPN to NCPN	180
Days from LCPN to SET	59
Accrual Fraction	0.327777777777777

The calculated price is 103.43.

Now calculate the yield-to-call. Don't forget to change the call date and call value.

Bond(5.022003, 3.032006, 6, 102.75, yield, 103.427958, 2, 2)

Calendar	30/360
Settlement Date	Friday 5.022003
Maturity Date	Friday 3.032006
Coupon (%)	6
Redemption/Call Value	102.75
Yield (%)	5.581233
Price	103.427958
Accrued Interest	0.983333
Last Coupon Date	3.032003
Next Coupon Date	9.032003
Number of Coupons	6
Days from LCPN to NCPN	180
Days from LCPN to SET	59
Accrual Fraction	0.327777777777777

The yield-to-call is 5.58%.

Example 4:

Calculate the price of a zero coupon bond purchased on May 19, 2003, maturing on June 30, 2007, with a yield of 10%. Coupons are semi-annual and the calendar basis is 30/360.

Bond(5.192003, 6.302017, 0, 100, 10, price, 2, 2)

Calendar	30/360
Settlement Date	Monday 5.192003
Maturity Date	Friday 6.302017
Coupon (%)	0
Redemption/Call Value	100
Yield (%)	10
Price	25.22744
Accrued Interest	0
Last Coupon Date	12.312002

And solving for the coupon using the computed price.

Bond(5.192003, 6.302017, cpn, 100, 10, 25.22744, 2, 2)

Calendar	30/360
Settlement Date	Monday 5.192003
Maturity Date	Friday 6.302017
Coupon (%)	0
Redemption/Call Value	100
Yield (%)	10
Price	25.22744

Now for some yields very close to zero and negative yields.

Bond(5.192003, 6.302017, 6, 100, 0.000001, price)

Calendar	Actual/Actual
Settlement Date	Monday 5.192003
Maturity Date	Friday 6.302017
Coupon (%)	6
Redemption/Call Value	100
Yield (%)	0.000001
Price	104.696112
Accrued Interest	2.303867
Last Coupon Date	12.312002
Next Coupon Date	6.302003
Number of Coupons	29
Days from LCPN to NCPN	181
Days from LCPN to SET	139
Accrual Fraction	0.767955801104972

Solving the the yield using the computed price.

Bond(5.192003, 6.302017, 6, 100, yield, 184.696112)

Calendar	Actual/Actual
Settlement Date	Monday 5.192003
Maturity Date	Friday 6.302017
Coupon (%)	6
Redemption/Call Value	100
Yield (%)	0.000001
Price	184.696112

And using a negative yield.

Bond(5.192003, 6.302017, 6, 100, -0.000001, price)

Calendar	Actual/Actual
Settlement Date	Monday 5.192003
Maturity Date	Friday 6.302017
Coupon (%)	6
Redemption/Call Value	100
Yield (%)	-0.000001
Price	184.696153

And solving for the yield from the price.

Bond(5.192003, 6.302017, 6, 100, yield, 184.696153)

Calendar	Actual/Actual
Settlement Date	Monday 5.192003
Maturity Date	Friday 6.302017
Coupon (%)	6
Redemption/Call Value	100
Yield (%)	-0.000001
Price	184.696153
Accrued Interest	2.303867

And using a larger negative yield.

Bond(5.022003, 3.032006, 6, 100, -5, price)

Calendar	Actual/Actual
Settlement Date	Friday 5.022003
Maturity Date	Friday 3.032006
Coupon (%)	6
Redemption/Call Value	100
Yield (%)	-5
Price	133.994215

In computing bond prices and yields, I had to develop a set of calendar programs. The main programs compute the Julian day number of a date and the date from a Julian day number. $JDN(date)$ computes the Julian day number for dates from Jan. 1, 4713 B.C. Dates B.C. are entered as negative dates, e.g. -1.014713. There is no year 0. The years go from -1 B.C. to 1 A.D. Always remember to enter date as mm.ddyyyy. $JDNtoDate(j)$ computes the date from the Julian day number. $DBD(d1,d2,t)$ computes the number of days between two dates. t is the type of calendar used. The default is 1 for Actual/Actual. Setting t to 2 uses a 30/360 day calendar. $\Delta_{360}(d1,d2)$ computes the days between two dates based on a 30 day month/360 day year calendar. $Date(d,days)$ will compute the date that is a number of days from the date d . And, $DOW(date)$ computes the day of the week for any date.

In Financial Mathematics II I develop routines for general annuities (TVM) and amortization schedules. Future articles will cover NPV and IRR for irregular cash flows, the equation of value, options, stocks, life insurance and life annuities, stochastic interest rates, etc.

(Don provided a DERIVE 5 and a DERIVE 6 version of his file, thanks a million.)

As you might know I was a teacher at a College for Business Administration and our curriculum covers a wide range of financial mathematics. So I was very happy to see Don's DERIVE-application. Unfortunately the TI-Application "Advanced Finance Mathematics" containing a worksheet "Bond" was removed from TI's homepage. I heard about a bug. It is promised that this application will be available again in the future.

I wrote a TVM-Solver for the CAS-TIs which made use of their CAS-capabilities. The original TI-TVM-Solver does not, which I cannot understand. It was not too difficult to transfer my Solver to DERIVE. Don and I will present our Solvers together with a bundle of examples in one of the next DNLs. Josef

The price for a house is 144 720 €. Seller and Buyer agree on a first installment of 30 000 € followed by monthly payments of 5439 € due at the end of the months. What is the effective interest rate?

$$tvm(24, x, -114720, 5439, 0, 12, 1) = \begin{bmatrix} N = & 24 \\ I\% = & 13.49003483 \\ PV = & -114720 \\ PMT = & 5439 \\ FV = & 0 \\ PpY = & 12 \\ CpY = & 1 \\ pmt : & END \\ interest : & INTEREST \end{bmatrix}$$

or another one: (What was the problem?)

$$tvm(\infty, 8.5, x, -500, 0, 12, 4, b, d) = \begin{bmatrix} N = & \infty \\ I\% = & 8.5 \\ PV = & 70085.84875 \\ PMT = & -500 \\ FV = & 0 \\ PpY = & 12 \\ CpY = & 4 \\ pmt : & BEGIN \\ interest : & DISCOUNT \end{bmatrix}$$

On Rational denominators

Fritz Mohr

fritz@mohr-web.de

Simplifying $\frac{1}{\sqrt[3]{5}}$ gives the fraction with rational denominator.

$$\frac{1}{\sqrt[3]{5}} = \frac{5^{2/3}}{5}$$

Simplifying $\frac{1}{1-\sqrt[3]{5}}$ does not simplify anyway.

Is it not possible for *DERIVE* to simplify such more complex expressions to a fraction with rational denominator?

Thanks for any help.

Terence Etchells

t.a.etchells@LIVJM.AC.UK

Hi Fritz,

Does this expression simplify to an expression with a rational denominator? I don't believe it does, I think I have a proof but I need to check it.

I believe there are no flaws in the proof (let me know if there are any) below that $\frac{1}{1-\sqrt[3]{5}}$ cannot be transformed into an expression with a rational denominator. Does anyone have a simpler proof? I am sure Johann Wiesenbauer will have one?

If $\frac{1}{1-\sqrt[3]{5}}$ were to have a rational denominator then it would simplify to an expression of the form $\frac{a+b\cdot Q}{c}$, where b cannot equal 0 (as this would make our expression rational) and c cannot = 0 for the obvious reason. a , b and c are rational numbers and Q is some irrational number.

If we rearrange $\frac{a+b\cdot Q}{c} = \frac{1}{1-\sqrt[3]{5}}$ to make c the subject, then $c = a + b\cdot Q - a\cdot\sqrt[3]{5} - b\cdot Q\cdot\sqrt[3]{5}$.

In order that c be rational, the expression $b\cdot Q - a\cdot\sqrt[3]{5} - b\cdot Q\cdot\sqrt[3]{5}$ must be rational.

In order that this expression be rational we must find a Q that makes the term $-b\cdot Q\cdot\sqrt[3]{5}$ rational (such a Q is $\sqrt[3]{5^2}$). So imagine we have found such a Q then the expression $b\cdot Q - a\cdot\sqrt[3]{5} - b\cdot Q\cdot\sqrt[3]{5}$ becomes $b\cdot Q - a\cdot\sqrt[3]{5} + R$ where R is some rational number.

Hence $b\cdot Q - a\cdot\sqrt[3]{5}$ must also be rational, this can only be the case if $Q = \frac{a}{b}\cdot\sqrt[3]{5}$. But this causes a

contradiction as R would then be $-b\cdot Q\cdot\sqrt[3]{5} = -b\cdot\left(\frac{a}{b}\right)\cdot\sqrt[3]{5}\cdot\sqrt[3]{5} = -a\cdot\sqrt[3]{5^2}$ which is irrational.

Hence there does not exist a Q such that c is rational.

Hence $\frac{1}{1-\sqrt[3]{5}}$ cannot be written in the form $\frac{a+b\cdot Q}{c}$ where a , b and c are rational and Q is some irrational number.

This is why *DERIVE* can't transform it into this form.

Ignacio Larrosa Cañestro

ilarrosa@mundo-r.com

We have that $a^3 - b^3 = (a - b)(a^2 + ab + b^2)$. Then let $a = 1$ and $b = 5^{\frac{1}{3}}$, and multiply and divide for $(a^2 + ab + b^2)$.

You get

$$\frac{1}{1 - \sqrt[3]{5}} = -\frac{1 + \sqrt[3]{5} + \sqrt[3]{5^2}}{4}$$

$$\begin{aligned} a^3 - b^3 &= (a - b) \cdot (a^2 + a \cdot b + b^2) \\ 1 - (5^{\frac{1}{3}})^3 &= (1 - 5^{\frac{1}{3}}) \cdot (1^2 + 1 \cdot 5^{\frac{1}{3}} + (5^{\frac{1}{3}})^2) \\ -4 &= (1 - 5^{\frac{1}{3}}) \cdot (1 + 5^{\frac{1}{3}} + 5^{\frac{2}{3}}) \\ -\frac{4}{1 - 5^{\frac{1}{3}}} &= 1 + 5^{\frac{1}{3}} + 5^{\frac{2}{3}} \\ \frac{1}{1 - 5^{\frac{1}{3}}} &= \frac{1 + 5^{\frac{1}{3}} + 5^{\frac{2}{3}}}{-4} \\ \frac{1}{1 - 5^{\frac{1}{3}}} &= -\frac{5^{\frac{2}{3}}}{4} - \frac{5^{\frac{1}{3}}}{4} - \frac{1}{4} \end{aligned}$$

Terence Etchells

Hi All,

Well done Ignacio, I made assumptions about Q that I ought not to have,

Cheers

Terence

Stefan Welke

Hallo Fritz,

We find $(1 - 5^{1/3}) * (1 + 5^{1/3} + 5^{2/3}) = 1 - 5$, and as a consequence:

$$1/(1 - 5^{1/3}) = (1 + 5^{1/3} + 5^{2/3})/(1 - 5)$$

This is an expression with rational denominator.

Remember: $(1 - q)(1 + q + q^2 + q^3 + \dots + q^n) = 1 - q^{n+1}$.

Albert Rich

As other responders to your question have said, *DERIVE* could rationalize the expression $\frac{1}{1 - 5^{\frac{1}{3}}}$ using

$$\text{the transformation } \frac{1}{1 - b^{\frac{1}{n}}} \Rightarrow \frac{\left(1 + b^{\frac{1}{n}} + b^{\frac{2}{n}} + \dots + b^{\frac{n-1}{n}}\right)}{1 - b}$$

However, *DERIVE* does not do this rationalization because it introduces additional fractional powers.

Hope this helps.

Aloha,

Albert D. Rich, Co-author of *DERIVE*

On Factorization Problems

Ignacio Larrosa Cañestro

ilarrosa@mundo-r.com

I get the polynomial correctly

$$x^8 - 40x^6 + 352x^4 - 960x^2 + 576$$

by squaring from three times, isolating the roots

$$x = \sqrt{2} + \sqrt{3} + \sqrt{5}.$$

$$x = \sqrt{2} + \sqrt{3} + \sqrt{5}$$

$$\text{EXPAND}((x - \sqrt{2})^2 = (\sqrt{3} + \sqrt{5})^2)$$

$$x^2 - 2\sqrt{2}x + 2 = 2\sqrt{15} + 8$$

$$\text{EXPAND}((x^2 - 6)^2 = (2\sqrt{15} + 2\sqrt{2}x)^2)$$

$$x^4 - 12x^2 + 36 = 8x^2 + 8\sqrt{30}x + 60$$

$$\text{EXPAND}((x^4 - 20x^2 - 24)^2 = 64 \cdot 30x^2)$$

$$x^8 - 40x^6 + 352x^4 + 960x^2 + 576 = 1920x^2$$

$$x^8 - 40x^6 + 352x^4 - 960x^2 + 576 = 0$$

But when I factorize the polynomial in radicals with *DERIVE*, it gives:

$$\text{FACTOR}(x^8 - 40x^6 + 352x^4 - 960x^2 + 576, \text{radical}, x)$$

$$(x + \sqrt{(20\sqrt{6} + 4\sqrt{114})}) \cdot (x - \sqrt{(20\sqrt{6} + 4\sqrt{114})}) \cdot (x + \sqrt{(20\sqrt{6} - 4\sqrt{114})}) \cdot (x - \sqrt{(20\sqrt{6} - 4\sqrt{114})}) \cdot (x^2 + x\sqrt{(40\sqrt{6} - 48)} + 24) \cdot (x^2 - x\sqrt{(40\sqrt{6} - 48)} + 24)$$

Try to expand this expression!!

And in complex,

$$\text{FACTOR}(x^8 - 40x^6 + 352x^4 - 960x^2 + 576, \text{Complex}, x)$$

$$(x + \sqrt{(20\sqrt{6} + 4\sqrt{114})}) \cdot (x - \sqrt{(20\sqrt{6} + 4\sqrt{114})}) \cdot (x + \sqrt{(20\sqrt{6} - 4\sqrt{114})}) \cdot (x - \sqrt{(20\sqrt{6} - 4\sqrt{114})}) \cdot (x + \sqrt{(10\sqrt{6} - 12)} + i\sqrt{(36 - 10\sqrt{6})}) \cdot (x + \sqrt{(10\sqrt{6} - 12)} - i\sqrt{(36 - 10\sqrt{6})}) \cdot (x - \sqrt{(10\sqrt{6} - 12)} + i\sqrt{(36 - 10\sqrt{6})}) \cdot (x - \sqrt{(10\sqrt{6} - 12)} - i\sqrt{(36 - 10\sqrt{6})})$$

getting four real roots, none of them $\sqrt{2} + \sqrt{3} + \sqrt{5}$, and four complex conjugate ones.But substituting $x \rightarrow \sqrt{t}$ we get

$$t^4 - 40t^3 + 352t^2 - 960t + 576$$

$$(t + 2\sqrt{15} + 2\sqrt{10} - 2\sqrt{6} - 10) \cdot (t + 2\sqrt{15} - 2\sqrt{10} + 2\sqrt{6} - 10) \cdot (t - 2\sqrt{15} + 2\sqrt{10} + 2\sqrt{6} - 10) \cdot (t - 2\sqrt{15} - 2\sqrt{10} - 2\sqrt{6} - 10)$$

$$\text{EXPAND}((x - \sqrt{2} - \sqrt{3} - \sqrt{5}) \cdot (x + \sqrt{2} + \sqrt{3} + \sqrt{5})) = x^2 - 2\sqrt{15} - 2\sqrt{10} - 2\sqrt{6} - 10$$

which *DERIVE* factorizes correctly (we can find our root in the 4th factor!).

Or approximating

$$(t - 0.828457) \cdot (t - 3.67960) \cdot (t - 6.52243) \cdot (t - 28.9695)$$

getting four linear real factors as expected. My version of *DERIVE* is 5.06 in Spanish.

Saludos, Ignacio Larrosa Cañestro

Ng Tze Beng

matngtb@NUS.EDU.SG

Dear All,

this bug seems to be in the previous version of 4.xx, too. I can definitely reproduce the same error with my DfW 5.06 English version. I hope version 6 would correct this.

DERIVE seems to have this problem. may we know, from the inventors of *DERIVE*, the algorithm used for the factoring program? Following Ignacio's suggestion we should get the following 8th degree polynomial whose roots are all eight possible sums of $\pm\sqrt{a}, \pm\sqrt{b}, \pm\sqrt{c}$ with a, b, c non negative and real. But if you set $a = 2$ and any b and c of form $b = p$ and $c = p + 2$ which are consecutive primes then we shall consistently getting errors, first only 4 real roots, the 8 incorrect real roots. The algorithm might have become instable.

$$\begin{aligned} & x^8 - 4x^6 \cdot (a + b + c) + 2x^4 \cdot (3a^2 + 2a \cdot (b + c) + 3b^2 + 2b \cdot c + 3c^2) - 4x^2 \cdot (a^3 - a^2 \cdot (b + c) - \\ & a \cdot (b^2 - 10b \cdot c + c^2) + b^3 - b^2 \cdot c - b \cdot c^2 + c^3) + a^4 - 4a^3 \cdot (b + c) + 2a^2 \cdot (3b^2 + 2b \cdot c + 3c^2) - \\ & 4a \cdot (b^3 - b^2 \cdot c - b \cdot c^2 + c^3) + b^4 - 4b^3 \cdot c + 6b^2 \cdot c^2 - 4b \cdot c^3 + c^4 \end{aligned}$$

I have tried some values with $a = > 2$ and prime, b and c any odd integers > 2 , *DERIVE* seems to have no problem. But with $a =$ even square free integer, b and c odd natural numbers, the same problem occurs. However, with any two of a, b and c even, the third number may be even and odd, factorization proceeds correctly.

Can Albert Rich give us some insight into this problem? Thank you.

Ng Tze Beng

Valeriu Anisiu

vanisiu@PERSONAL.RO

Hello Derivers,

Concerning the polynomial $p0 := x^8 - 40x^6 + 352x^4 - 960x^2 + 576$. It indeed satisfies the very unpleasant relation $\text{expand}(\text{factor}(p0, \text{Complex}, x)) \neq p0$.

I found a very curious fact in *DERIVE* 6 with this polynomial. The minimal polynomial in $Z[X]$ having a root $\sqrt{a} + \sqrt{b} + \sqrt{c}$ can be obtained as

$$p3(a, b, c, x) := \text{FIRST}(\text{GROEBNER_BASIS}([u^2 - a, v^2 - b, w^2 - c, x - u - v - w], [u, v, w, x]))$$

$$p3(2, 3, 5, x) = x^8 - 40x^6 + 352x^4 - 960x^2 + 576$$

Our polynomial is $p3(2, 3, 5, x)$. Defining also

$\text{testzero}(p, x) := \text{EXPAND}(\text{FACTOR}(p, \text{Complex}, x)) - p$

$\text{shouldbe0}(a, b, c, x) := \text{testzero}(p3(a, b, c, x))$

$\text{shouldbe0}(1377, 1112, 573, x) = 0$

$\text{shouldbe0}(2, 3, 7, x) = 0$

then shouldbe(2, 3, 5, x) is simplified to a nonzero poly-nomial!

But for several other values I have tested $\text{shouldbe0}(a, b, c, x)$ simplifies to 0.

My question is: what is so special with $p0$?

(or maybe with 2-3-5). Does some philosophical problem hide here?

Cheers, Vanisiu

MacDonald Phillips

phillipsm@gao.gov

Hello Derivers,

I too get the same results with *DERIVE* 6 Beta as Valeriu. However I also have an HP95LX with *DERIVE* 2.5 on it. When I factor p0 I get entirely different factors than with *DERIVE* 6. The factors are:

$$\begin{array}{cccc} -\sqrt{5} - \sqrt{3} + \sqrt{2} & \sqrt{5} + \sqrt{3} - \sqrt{2} & -\sqrt{5} + \sqrt{3} - \sqrt{2} & \sqrt{5} - \sqrt{3} + \sqrt{2} \\ -\sqrt{5} + \sqrt{3} + \sqrt{2} & \sqrt{5} - \sqrt{3} - \sqrt{2} & -\sqrt{5} - \sqrt{3} - \sqrt{2} & \sqrt{5} + \sqrt{3} + \sqrt{2} \end{array}$$

These are the factors as determined by *DERIVE* 2.5 but simplified by *DERIVE* 6 Beta. For instance, the first factor above was $-\sqrt{-2 \cdot \sqrt{6} - 2 \cdot \sqrt{10} + 2 \cdot \sqrt{15} + 10}$. *DERIVE* 2.5 was not able to simplify the factor any further.

$$- \sqrt{(-2 \cdot \sqrt{6} - 2 \cdot \sqrt{10} + 2 \cdot \sqrt{15} + 10)} = -\sqrt{5} - \sqrt{3} + \sqrt{2} \quad (\text{DERIVE 6})$$

These of course are the correct factors. And, when expanded, give the original equation. Something, I don't know what, seems to be amiss in the factoring routine in *DERIVE* 5 and 6. Perhaps Albert will clear this up for us.

Cheers, Don

On Plotting FunctionsAdam Marlewski

amarlew@math.put.poznan.pl

How to plot correctly the graphs of the relations $u = t^2 - 3t + 3$ and $t = u^2 - 3u + 3$?

Although I set the order variables to t, u, x, y, z the graphs of both relations are identical, while they have to differ!

In case $u = t^2 - 3t + 3$ the parabola is laying along the vertical axis, and it is correct.

In case $t = u^2 - 3u + 3$ the parabola has to lay along the vertical axis, but *DERIVE* shows it laying along the vertical axis (and it is errorous).

Ignacio Larrosa Cañestro

ilarrosa@mundo-r.com

Substitute $t \rightarrow x$ and $u \rightarrow y$. Then *DERIVE* plots a parabola with horizontal axis.

I don't exactly know how *DERIVE* assigns axes to variables in 2D-plot graphics, but if you use x and y , x always will be assigned to the horizontal axis and y to the vertical one.

Terence Etchells

A quick work around is to ensure that your dependent variable is y and the independent is x .

$$\text{i.e. } x = y^2 - 3y + 3.$$

This plots properly.

Cheers, Terence

Adam Marlewski

Hello Derivers, and – in particular – Ignacio and Terence,
once again where is my problem in plotting in *DERIVE* 5.04.

Case 1

Under standard variable order, i.e. $\text{VariableOrder} := [x, y, z]$
and if this order is changed to $\text{VariableOrder} := [y, x, z]$
the graph of the relation $x^2 - 3xy - y^2 + 5 = 0$ remains the same.
In particular it means that the horizontal axis is still the x-axis.

Case 2

Now the relation $t^2 - 3tu - u^2 + 5 = 0$ is plotted differently:

- if there is set $\text{VariableOrder} := [t, u, x, y, z]$, then the graph passes the point $(-1, 3/2 - \sqrt{33}/2)$ and it means that the horizontal axis is t-variable axis.
- if there is set $\text{VariableOrder} := [u, t, x, y, z]$, then the graph does not pass this point and it means that the horizontal axis is u-variable axis.

The question is:

when does the variable order (stated by the value of state-variable VariableOrder) affect plotting?

In examples above we see it does not effect (case 1) and it obviously does in case 2.

Lottery with *DERIVE*

Lester

I would like to have a simple expression which will allow me to extract 6 unique numbers from a total of 49, with no duplicates or zero entries. I tried the following, which gives a sequence of 6 numbers, but occasionally it will give two of the same number abd also include zero sometimes. How do I resolve this point?

`VECTOR (RANDOM (50) , x, 1, 6)`

Terence Etchells

This should do it

```

LOTTERY(n, list, t_, newlist, counter) := PROG(newlist := [], counter := 1,
  LOOP(IF(counter > n, RETURN newlist), t_ := RANDOM(DIM(list)) + 1, newlist
    := ADJOIN(list↓t_, newlist), list := DELETE(list, t_), counter := 1))

```

To perform the task you require then simplify

`LOTTERY(6, [1, ..., 49]) = [48, 46, 26, 11, 30, 17]`

Additional advice: If you want to avoid receiving the same "Random Jackpot" at each new start of *DERIVE* first simplify `RANDOM(0)` or even better include this into the program. See below the first lines of the program. (Josef)

```

LOTTERI(n, list, t_, newlist, counter, dummy) :=
  Prog
    dummy := RANDOM(0)
    newlist := []
    counter := 1

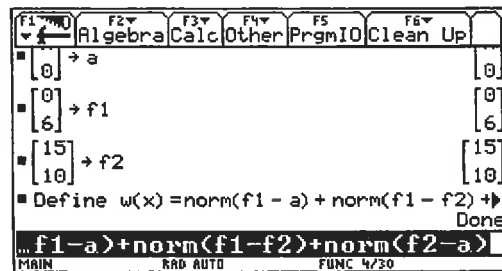
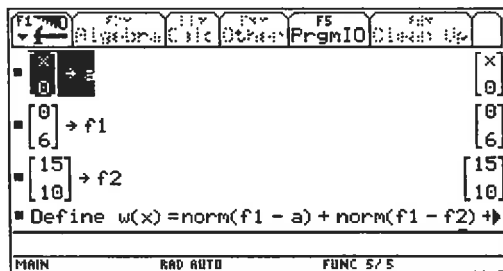
```

Misbehaving Voyage 200??

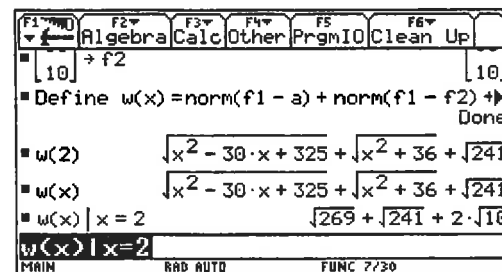
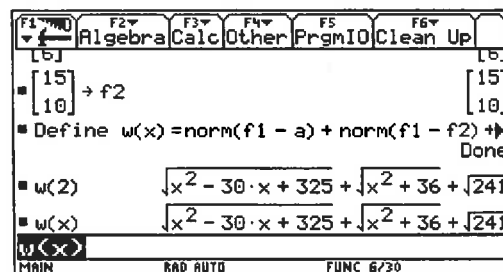
Thomas Himmelbauer

I want to minimize the circumference of a triangle with two fixed vertices and one being variable on the x -axis.

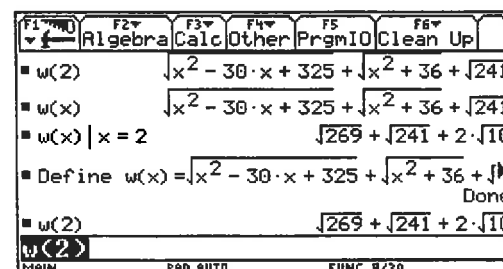
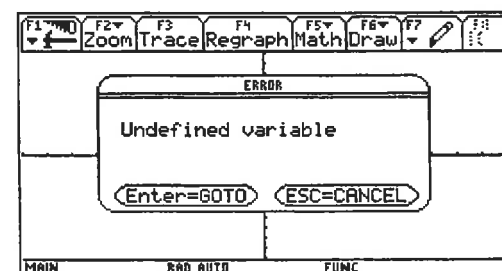
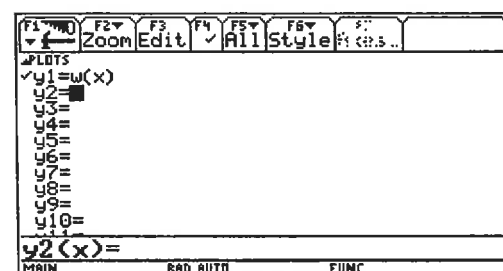
My students defined a circumference-function using the norm-function and we came across a bad surprise:



The function cannot be evaluated. It is possible using the with-operator. There are no problems doing in the same way on the TI-92 PLUS!



And we don't have any graph, of course.



If we first simplify the expression and then define the function it works.

Can you explain this?

Dear Thomas,

Your problem can be resolved very easily: Define a as $a(x)$ – first screen shot. Then it works as expected.

Josef

The Wonderful World of *DERIVE* 6

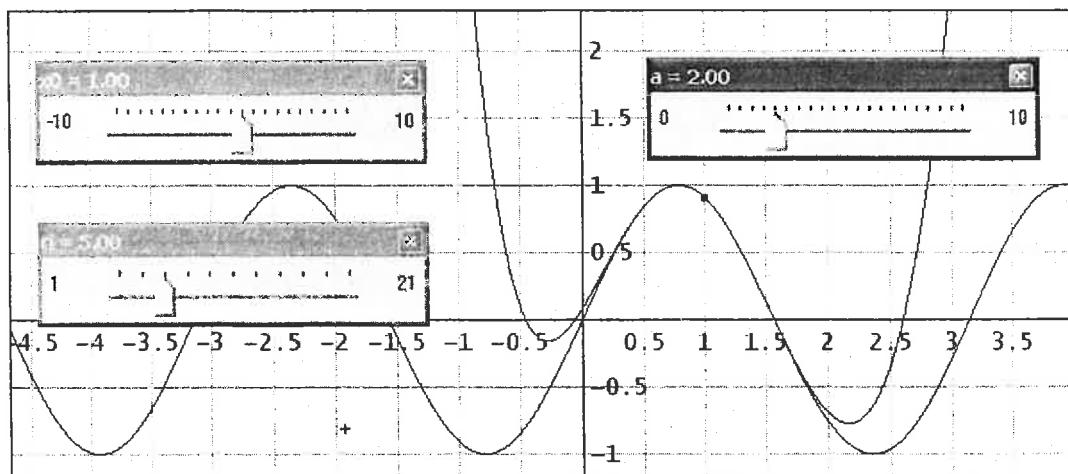
DERIVE 6 offers a bundle of new and exciting features. I'd like to present a selection of them – in particular those which are very impressive and helpful for CAS-supported math education. I am sure that Johann Wiesenbauer will focus on "*DERIVE* Interl" subject matters. See also the **Groebner Bases** on page 40.

I'll start with the "Slider Bars", which have been introduced in the Bézier curve contribution.

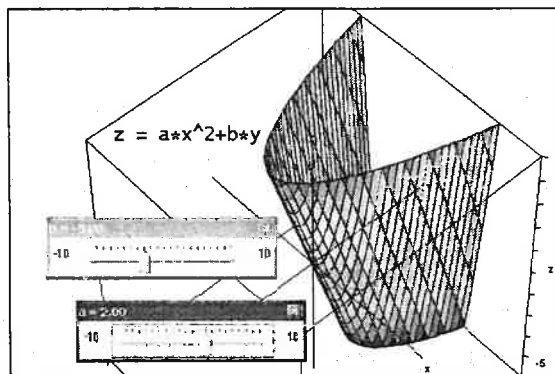
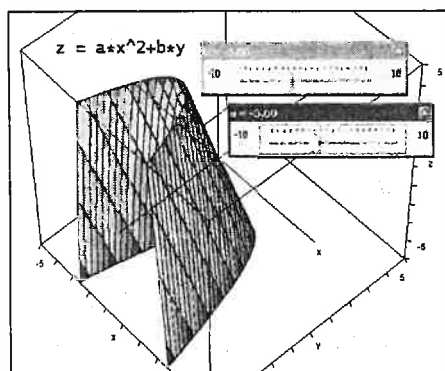
Assume that you want to demonstrate the Taylor expansion for the sine function. I want to present the graph of the sine function (with variable period length) together with the graph of the Taylor polynomial (of variable degree) for variable locations. I edit a vector (list) containing the function, the Taylor polynomial and the point on the graph – three parameters, a , n , and x_0 .

```
[SIN(a·x), TAYLOR(SIN(a·x), x, x0, n), [x0, SIN(a·x0)]]
```

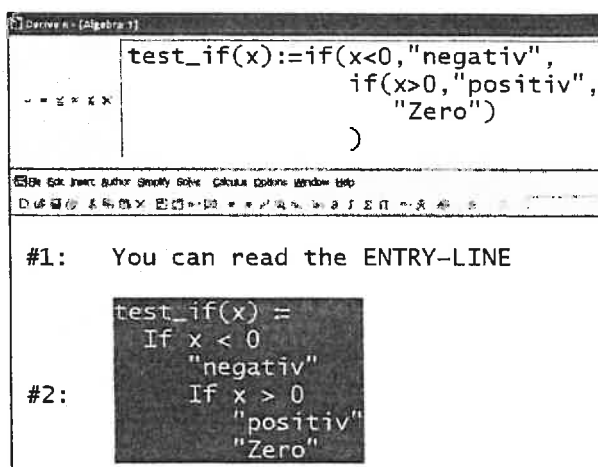
I switch to the 2D-Plot Window and first Insert three Slider Bars for a , n and x_0 (specify Minimum, Maximum and number of intervals for the variables). Then I plot the list and move the bars



As you can see in the next figures, slider bars can be introduced for 3D-Plots, too. Moreover it is now much easier to rotate the objects using the mouse.



Let's go back to the Algebra Window. I don't know how often I had wished to enlarge the font size in the Entry line – remembering numerous workshops and classes in large PC-labs. The audience couldn't follow my typed commands and I had to write them down on the blackboard or even worse, lacking this "most important technology" I had to spell them letter for letter. But now we can scale not only the expressions in the Algebra Window but also simultaneously the type font in the entry line. By the way the Unicode-font is much prettier than the DfW5-font. (But take care mixing up working with version 5 and version 6 – I'll come back to this in the next *DNL*.)

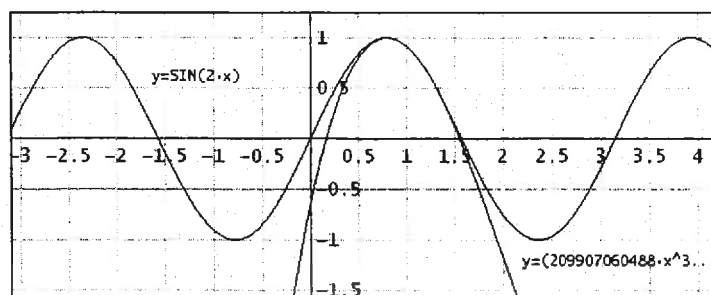
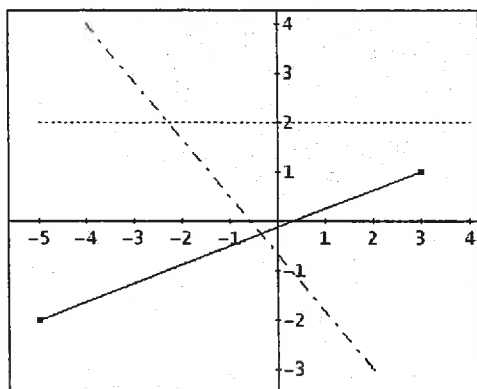


And I also don't know how often the audience had problems to catch a long IF-construction or any other expression (program, ...) which turned out to be longer than one line.

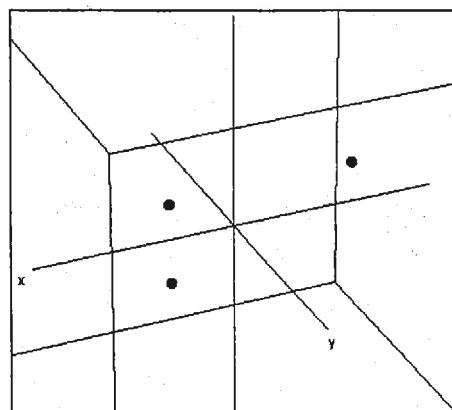
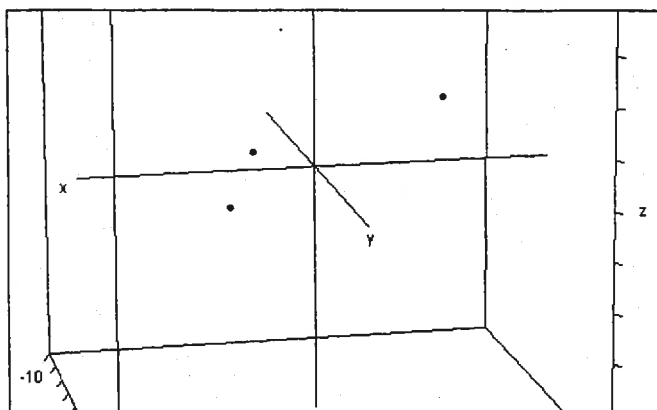
Now we have a **Multi-Line-Editor**. It is not very large, because restricted for only some lines, but it is a big step into the right direction.

Concerning editing complex expressions, the **Parentheses Matching** feature is very helpful.

The **2D-Plot** Window offers **different styles** for plotting point connecting segments (from solid to dash-dot-dot) and we can let the system **automatically present the expressions** of the plotted graphs. The colour of this annotation corresponds with the plot colour of the respective graph.



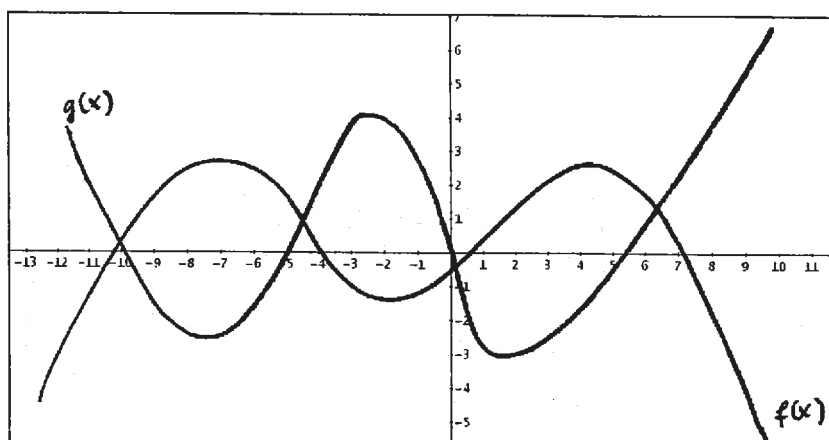
In the **3D-Plot** Window we have three **different sizes** for representing points.



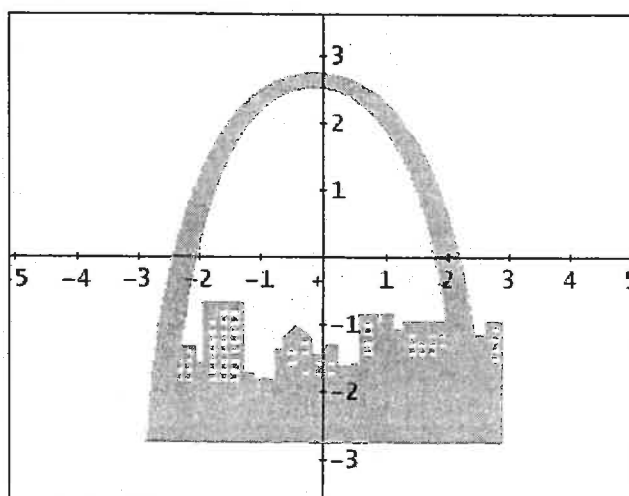
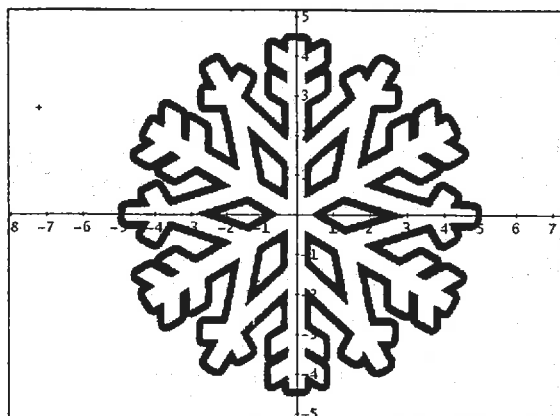
A feature which has – in my personal opinion – an enormous educational potential is the possibility to **load external graphics** (in bmp-format) as background on the 2D-Plot Window.

I can imagine that many problems for modelling will achieve a new dimension and quality. You can load handmade sketches of graphs (provided by a scanner), prepared *DERIVE*-graphs, graphics from numerous clip arts collections, from the Internet,

Which functions describe the given ones in the best possible way?



(Second question: Which is the **best possible way**?)



Are you able to reproduce the Snow Flake on your *DERIVE* screen?

This is the famous Gateway Arch from St. Louis. What is its mathematical representation?

Finally two main improvements:

Display Steps in the simplification with display of the transformation rules or without.

I will not show any differentiation- or integration process, but a transformation with trigonometric functions. Please follow the screen shots.

#1: Trigonometry := Expand

#2: $\text{SIN}(3 \cdot x) + \text{COS}(3 \cdot x)$

$$\text{COS}\left(z + \frac{\pi}{2}\right) \rightarrow -\text{SIN}(z)$$

$$\#3: 4 \cdot \text{SIN}(x) \cdot \text{COS}(x)^2 - \text{SIN}(x) + \text{COS}(x) \cdot (1 - 4 \cdot (-\text{SIN}(x))^2)$$

$$\#4: 4 \cdot \text{SIN}(x) \cdot \text{COS}(x)^2 + \text{COS}(x) \cdot (1 - 4 \cdot \text{SIN}(x)^2) - \text{SIN}(x)$$

Now we transform in the "other direction":

#5: Trigonometry := Collect

#6: $\text{SIN}(3 \cdot x) + \text{COS}(3 \cdot x)$

If $x > 0$,

$$\text{ATAN}(y, x) \Rightarrow \text{ATAN}\left(\frac{y}{x}\right)$$

$$a \cdot \text{SIN}(z + c) + b \cdot \text{COS}(z + d) \Rightarrow \sqrt{(2 \cdot a \cdot b \cdot \text{SIN}(c - d) + a^2 + b^2)} \cdot \text{SIN}(z + \text{ATAN}(a \cdot \text{SIN}(c) + b \cdot \text{COS}(d), a \cdot \text{COS}(c) - b \cdot \text{SIN}(d)))$$

$$\#7: \text{SIN}\left(3 \cdot x + \text{ATAN}\left(\text{COS}(0) + \text{SIN}(0), \text{COS}\left(\frac{\pi}{2}\right) + \text{COS}(0)\right)\right) \cdot \sqrt{(2 \cdot \text{SIN}(0) + 2)}$$

$$\text{COS}(0) \Rightarrow 1$$

$$\#8: \text{SIN}\left(3 \cdot x + \text{ATAN}\left(1 + \text{SIN}(0), \text{COS}\left(\frac{\pi}{2}\right) + \text{COS}(0)\right)\right) \cdot \sqrt{(2 \cdot \text{SIN}(0) + 2)}$$

Now I switch off displaying the rules and set the problem for the students to find the rules which were applied by *DERIVE* (or better by Albert Rich, of course.) This will open lots of discussions because often different rules can be applied in a meaningful way. We might get a glimpse into the programmer's secrets!

Displaying the rules is switched off!!

$$\#9: \text{SIN}\left(3 \cdot x + \text{ATAN}\left(1, \text{COS}\left(\frac{\pi}{2}\right) + \text{COS}(0)\right)\right) \cdot \sqrt{(2 \cdot \text{SIN}(0) + 2)}$$

$$\#10: \text{SIN}(3 \cdot x + \text{ATAN}(1, \text{COS}(0))) \cdot \sqrt{(2 \cdot \text{SIN}(0) + 2)}$$

$$\#11: \text{SIN}(3 \cdot x + \text{ATAN}(1, 1)) \cdot \sqrt{(2 \cdot \text{SIN}(0) + 2)}$$

$$\#12: \text{SIN}(3 \cdot x + \text{ATAN}(1)) \cdot \sqrt{(2 \cdot \text{SIN}(0) + 2)}$$

$$\#13: \text{SIN}\left(3 \cdot x + \frac{\pi}{4}\right) \cdot \sqrt{(2 \cdot \text{SIN}(0) + 2)}$$

$$\#14: \text{SIN}\left(3 \cdot x + \frac{\pi}{4}\right) \cdot \sqrt{2}$$

$$\#15: \sqrt{2} \cdot \text{SIN}\left(3 \cdot x + \frac{\pi}{4}\right)$$

And the really big thing, the **compatibility between PC and handheld technology**.

On the next page I show my *DERIVE* 6 worksheet.

My first attempt to transfer a file to the Voyage 200

```
#1:  2
      2
      x  + x + 4 = 0

#2:  SOLVE(2·x  + x + 4 = 0, x, Real)

#3:                                     false

#4:  2
      2
      x  + x + 4 = 0, x)

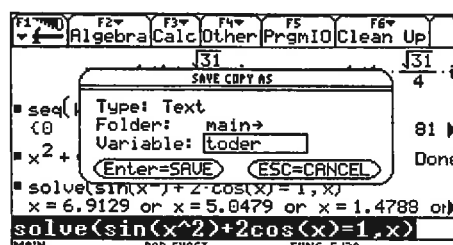
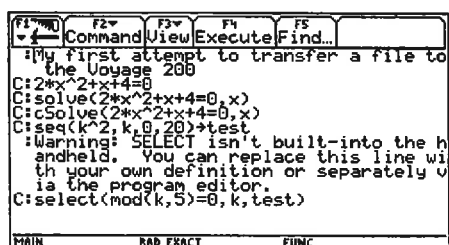
#5:                                     1      √31·i      1      √31·i
                                     4      4      4      4

#6:  test := VECTOR(k , k, 0, 20)

#7:  SELECT(MOD(k, 5) = 0, k, test)

#8:  [0, 25, 100, 225, 400]
```

... and transfer this file to my *Voyage 200* – as a textfile (eg. named toti). This text file can be opened on the *TI* and executed step by step (or as a whole, of course). One has the choice to transmit with or without simplification.



As you can see, the *TI* "understands" the *DERIVE*-syntax (**VECTOR**($k^2, k, 0, 20$) changes to **seq**($k^2, k, 0, 20$), but as its "brain" is much smaller, it does not know all *DERIVE*-commands. The **SELECT**-command is unknown, one has to write his own *TI*-program to perform a selection.

I proceed on the *TI* with defining a function of two variables and solving an equation, save the whole as another text-file (eg. as **toder**) and send it back to the PC. The program *TI-Connect* is necessary for the exchange of "messages". It is on the *DERIVE* 6 CD.

```
#6:  test := [0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196, 225, 256, 289,
      324, 361, 400]

#7:  f(x, y) := x  + y  2  2

#8:                                     2  2
                                     f(x, y) := x  + y

#9:  2
      2
      SIN(x ) + 2·COS(x) = 1, x, Real)

#10:                                     2
      SIN(x ) + 2·COS(x) = 1

#11:  NSOLVE(SIN(x ) + 2·COS(x) = 1, x, Real)

#12:                                     x = 1.478772610
```

Working with *DERIVE* will become more comfortable and personal by customizing the toolbars including creating own short cuts.

Das Josefspiel – The Josephus Problem

Rüdeger Baumann, Celle
baumann-celle@t-online.de

This algorithm is named for a historian of the first century, Flavius Josephus, who survived the Jewish-Roman war due to his mathematical talents. Legend has it that he was one out of 41 Jewish rebels trapped by the Romans. His companions preferred suicide to escape, so they decided to form a cycle and to kill every third person and to proceed around the circle until no one was left. Josephus wasn't excited by the idea of killing himself, so he calculated where he has to stand to survive the vicious circle. The following implementation demands the input of how many people will stand around a circle and how many shall be passed over before the next one is killed.

Auf einem Schiff befinden sich zwei Gruppen von Passagieren. Um uns nicht dem Vorwurf der Rassen diskriminierung auszusetzen – in den alten Fassungen der Geschichte handelt es sich nämlich um Schwarze und Weiße, bald um Türken und Christen – nehmen wir einfach an, es seien 15 Wiener und 15 Klagenfurter. Nun gerät das Schiff bei einem Unwetter in Seenot; Rettung scheint nur möglich, wenn die Hälfte der Passagiere über Bord geht. Der Kapitän schlägt folgende Abzählmethode vor: die Passagiere stellen sich im Kreis auf, jeder neunte muss sich opfern. Da der Kapitän Klagenfurter ist, möchte er die Passagiere so stellen, dass nur die Wiener ausballotiert werden. Wie macht er das?

Ihren Namen hat die Geschichte von der folgenden Begebenheit: Im Jahre 67 n.Chr. wurde die galiläische Stadt Jotapata nach längerer Belagerung vom römischen General (später Kaiser) Vespasian eingenommen. Dem Historiker Flavius Josephus (37 – 95), welcher sich an der Verteidigung an führender Position beteiligt hatte, gelang es, unerkannt durch die feindlichen Linien hindurch in eine tiefe Zisterne zu springen, welche – von oben unsichtbar – mit einer geräumigen Höhle verbunden war. Er fand dort vierzig Honoratioren vor, die sich verborgen hielten, um der Sklaverei zu entgehen. Nach zwei Tagen wurde das Versteck von einer Frau verraten, die mit in der Höhle gewesen und bei einem nächtlichen Erkundungsgang von den Römern aufgegriffen worden war. Die Römer forderten Josephus auf, aus der Höhle herauszukommen und sicherten ihm freies Geleit zu. Da zogen die anderen jedoch das Schwert und drohten, ihn umzubringen, falls er dem Verlangen der Römer Folge leisten sollte; sie wollten sich lieber selbst entleiben als in die Hände der Römer fallen.

In dieser kritischen Situation verfiel Josephus auf einen Ausweg, indem er folgende Prozedur vorschlug: Alle 41 Personen sollten sich in einer Reihe aufstellen. An einem Ende beginnend, sollte dann jeder dritte durch den rechten Nebenmann getötet werden. Am Ende der Reihe sollte die Zählung mit dem Anfang der Reihe fortgesetzt werden, bis nur noch einer übrig bliebe, der sich dann selbst umbringen sollte. Nach diesem Vorschlag wurde verfahren. Josephus plazierte sich an 16. und einen anderen schwächeren Mann an 31. Stelle, so dass er als Vorletzter und der andere (den er überwältigen konnte) an letzter Stelle ausschied. Auf diese Weise rettete Josephus sein Leben.

Das allgemeine Problem lautet: Es werden n nummerierte Objekte im Kreis angeordnet; dann wird beginnend mit der Nummer k , jedes k -te Objekt ausgeschieden, wobei sich der Kreis sofort wieder schließt. Gesucht ist die Reihenfolge der ausgeschiedenen Objekte; diese Reihenfolge heißt *Josephus-Permutation*. Ist beispielsweise $n = 10$ und $k = 2$, so ergibt sich (bei Unterstreichung ausgeschiedener Elemente):

1	<u>2</u>	3	<u>4</u>	5	<u>6</u>	7	<u>8</u>	9	<u>10</u>
1	<u>3</u>	5	<u>7</u>	9					
<u>1</u>	5	<u>9</u>							

Die zugehörige Josephus Permutation lautet somit (2, 4, 6, 8, 10, 3, 7, 1, 9, 5).

Rüdeger started with an example having 15 Person from Vienna and 15 from Klagenfurt on a boat, which is in distress on sea (on the Woerther Lake??). The captain gives order that 15 people have to jump from the boat. All passengers form a circle and every ninth has to jump. How does the captain – himself being from Klagenfurt – arrange the people to save the Carinthians?

For $n = 10$ people with every second person killed one obtains the *Josephus permutation* shown above. (The underlined "elements" are eliminated.)

Rüdeger provided a *DERIVE*-program which in his own words "terribly slow". So his challenge for the *DERIVIANS* (and the TI-ers, too, of course) to write an efficient program. The results for the problems from above are given below.

```

Josef(n, s, Liste := []) :=
  Prog
  v := APPEND([1, ..., n])
  z := n
  i := 0
  Loop
  i := i + 1
  If i > n
    RETURN REVERSE(Liste)
  j := 0
  Loop
  j := j + 1
  If j > s exit
  r := IP(z < n, z + 1, 1)
  Loop
  If v[r] ≠ 0 exit
  r := IP(r < n, r + 1, 1)
  z := r
  Liste := ADJOIN(v[r], Liste)
  v := REPLACE(0, v, z)

```

Nach seinen eigenen Worten ist das nebenstehende Programm "fürchterlich" langsam und Rüdegers Herausforderung ist es, ein effizientes Programm zu schreiben!

Josef(10, 2) = [2, 4, 6, 8, 10, 3, 7, 1, 9, 5]

Josef(41, 3) = [3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 1, 5, 10, 14, 19, 23, 28, 32, 37, 41, 7, 13, 20, 26, 34, 40, 8, 17, 29, 38, 11, 25, 2, 22, 4, 35, 16, 31]

Josef(30, 9) = [9, 18, 27, 6, 16, 26, 7, 19, 30, 12, 24, 8, 22, 5, 23, 11, 29, 17, 10, 2, 28, 25, 1, 4, 15, 13, 14, 3, 20, 21]

The captain will place the 15 citizens of Klagenfurt on positions 1, 2, 3, 4, 10, 11, 13, 14, 15, 17, 20, 21, 29.

References:

Interesting homepages:

<http://mathworld.wolfram.com/JosephusProblem.html>

<http://www.auto.tuwien.ac.at/~blieb/woop/josephus.html>

Titbits from Algebra and Number Theory (26)

by Johann Wiesenbauer, Vienna

The time since my last column has been a particularly eventful one. The top event for Derivers is, of course, the advent of the new version 6 of Derive with a number of highly interesting new features. What I find most exciting is the new extremely versatile unicode, which is freely scalable on the screen. At last I can adjust the size of the writing such that the students in my lectures can follow the Derive demos even if they sit in the last row. Great! As for programming in Derive, this version has finally put an end to the one-line editor (just when I got used to it!) and for the first time allows the output of intermediate results on the screen while the program is still running. I won't carry coals to Newcastle though by raving about the new features of DfW 6, as you certainly have tried them out yourself in the meantime and also will more read about them in the rest of this issue of the DNL. Let me only point out once more that I always tacitly assume that you use the most recent version of Derive for the programs introduced here, which is Derive 6 at the moment.

Well, another thing (though by far not equally earthshaking - life isn't fair, is it?) has attracted the attention of the man in the street even more. As you may have read in the newspapers they found a new record prime with more than six million digits. Needless to say that it is again a Mersenne prime, to be more precise number forty in the list of known Mersenne primes, namely

$$2^{20,996,011} - 1$$

and that it has been found by the GIMPS project (for details see <http://www.mersenne.org>). By the way, there is an easy puzzle, which put many people on the spot though, as to the computation of the exact number of digits of this prime. May I ask you to try it using Derive and compare your solution with the one given at the end of this article!?

What puts me on the spot right now is the fact that I promised to continue with my update of polynomial routines, which I started in the last column, but would rather like to talk about the mathematical background of that discovery which is an intriguing blend of ideas from algebra and number theory (hence fitting this series to a T !). Fortunately, two facts come to my help. First, Josef didn't restrict the number of my pages this time, so I can cover both topics without arousing his indignation. Second, there is a close connection between the two topics, since surprisingly efficiently testing Mersenne numbers for primality has a lot to do with efficiently multiplying two polynomials, as I will explain in detail later.

For now, let's continue with the update of the "polyroutines" mentioned above. Note that you will need some of the routines defined in my Titbits #25, whose definitions I don't repeat here to save space. (You will find all routines together in the accompanying DfW-file though!). Furthermore, as a general reference for the mathematical background [1] comes in handy again in case there is any need.

We have already introduced a routine `polyirr?(u,p,x)` in order to determine whether a given polynomial $u \in \mathbb{Z}_p[x]$ is irreducible or not, as well as a routine `irrpolym(m,p)`, whose output is a random irreducible polynomial of degree m in $\mathbb{Z}_p[x]$. In many applications we need irreducible polynomials $f(x)$ though, which fulfill the additional condition that x is generator in multiplicative cyclic group of the field $\mathbb{Z}_p[x]/(f(x))$. They are called primitive polynomials and again for every given positive integer m primitive polynomials of degree m do exist. As a first project, let's supplement our routines mentioned above by the corresponding routines for primitive polynomials.

In the routine `polyprimitive?(u,p,x)` below it is first checked whether u is actually irreducible. If this first condition is fulfilled, then the second condition is checked whether $x^{n/q} \neq 1 \pmod u$ for every prime divisor q of $n = p^m - 1$, where m is the degree of u .

```

polyprimitive?(u, p, x, n_, q_) :=
  Prog
    If ¬ polyirr?(u, p, x)
      RETURN false
    n_ := p^POLY_DEGREE(u, x) - 1
    q_ := (FACTORS(n_)) COL 1
    Loop
      If TERMS(polypower(x, n_/FIRST(q_), u, p, x)) = [1]
        RETURN false
      q_ := REST(q_)
      If q_ = [] exit

```

```

primitivepoly(m, p, x, u_) :=
  Loop
    u_ := Σ(RANDOM(p)·x^k_, k_, 0, m - 1) + x^m
    If polyprimitive?(u_, p, x)
      RETURN u_

```

```

polyprimitive?(x4 + x + 1, 2) = true

```

```

polyirr?(x4 + x3 + x2 + x + 1, 2) = true

```

```

polyprimitive?(x4 + x3 + x2 + x + 1, 2) = false

```

If you want to set up the environment for the calculations in a field \mathbb{F}_{p^m} , where a generator of the multiplicative group is known in advance, then you should modify the routine `setupfield(q)` in the last Titbits accordingly:

```

setupfield(q) :=
  Prog
    If ¬ PRIME_POWER?(q)
      RETURN "q must be a prime power!"
    p := FIRST(FACTORS(q))
    m := p↓2
    p := FIRST(p)
    mp := primitivepoly(m, p, x)
    "ok"

```

```

setupfield(16) = ok

```

```

p = 2

```

```

m = 4

```

```

mp = x4 + x3 + 1

```

Using the same routines `plus(u,v,p)`, `minus(u,v,p)`, `times(u,v,mp,p)`, `div(u,v,mp,p)` for the four basic operations $+$, $-$, \cdot , $/$ in a field as in the first part of this paper, you can again perform all the calculations in the corresponding field you want. The only difference is that the element `p`, which corresponds to the polynomial `x` in the polynomial representation, will always be a generator of the cyclic multiplicative group of the field now.

Hence, in our example above 2 is a generator of the finite field with 16 elements, which you get by the setup above, as you can check yourself by using the following multiplication table. (Again the usual index row and index column of such a table is not given here explicitly, but it is mirrored anyway in the second row and column, respectively.)

VECTOR(VECTOR(times(a, b), a, 0, 15), b, 0, 15)

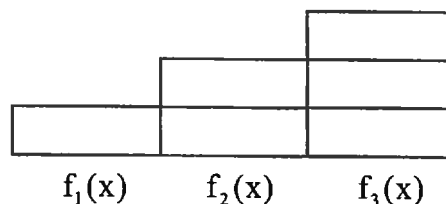
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	2	4	6	8	10	12	14	9	11	13	15	1	3	5	7
0	3	6	5	12	15	10	9	1	2	7	4	13	14	11	8
0	4	8	12	9	13	1	5	11	15	3	7	2	6	10	14
0	5	10	15	13	8	7	2	3	6	9	12	14	11	4	1
0	6	12	10	1	7	13	11	2	4	14	8	3	5	15	9
0	7	14	9	5	2	11	12	10	13	4	3	15	8	1	6
0	8	9	1	11	3	2	10	15	7	6	14	4	12	13	5
0	9	11	2	15	6	4	13	7	14	12	5	8	1	3	10
0	10	13	7	3	9	14	4	6	12	11	1	5	15	8	2
0	11	15	4	7	12	8	3	14	5	1	10	9	2	6	13
0	12	1	13	2	14	3	15	4	8	5	9	6	10	7	11
0	13	3	14	6	11	5	8	12	1	15	2	10	7	9	4
0	14	5	11	10	4	15	1	13	3	8	6	7	9	2	12
0	15	7	8	14	1	9	6	5	10	2	13	11	4	12	3

Now let's turn to the problem of factoring a given polynomial $f(x) \in \mathbb{Z}_p[x]$. Since $\mathbb{Z}_p[x]$ is a factorial ring, we can assume that $f(x)$ has a representation of the form

$$f(x) = f_1(x)f_2(x)^2 \dots f_r(x)^r$$

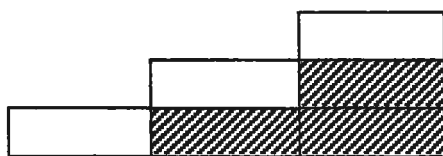
where $f_k(x)$ is simply the product of all irreducible factors of $f(x)$ with multiplicity k , $k=1,2,\dots,r$. This is called the squarefree factorization of $f(x)$, since all the factors $f_k(x)$ are obviously squarefree, and it is obtained by the program below..

In order to demonstrate its simple idea let's assume for the moment that $r=3$ and $p>3$, which will make things a little easier. It may also help you to visualize the factorization $f(x) = f_1(x)f_2(x)^2 f_3(x)^3$ as a set of dominoes arranged in the following way:



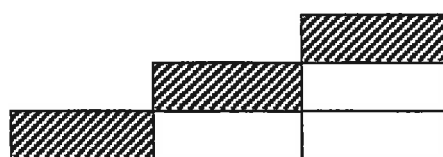
Let's do it step by step:

1. Step:



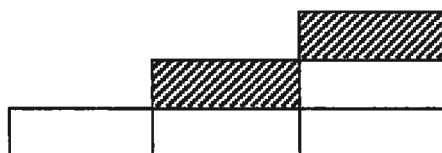
How can we get hold of the part hatched in the figure above? Well, it is simply the part that $f(x)$ and its first derivative $f'(x)$ have in common, i.e. $f_{-}(x) := \gcd(f(x), f'(x))$, isn't it?

2. Step:



How can we get hold of the part hatched in the figure above. Well, it is simply the quotient $g_{-}(x) := f(x)/f_{-}(x)$, isn't it?

3. Step:



How can we get hold of the part hatched in the figure above? Well, it is simply the part that $f_{-}(x)$ and $g_{-}(x)$ have in common, i.e. $h_{-}(x) := \gcd(f_{-}(x), g_{-}(x))$, isn't it?

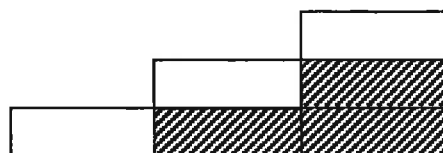
4. Step:



How can we get hold of the part hatched in the figure above? Well, it is simply the quotient $u_{-}(x) = g_{-}(x)/h_{-}(x)$, isn't it? Note that we have found $f_1(x)$ so far!

5. Step:

How can we reduce the original problem to a simpler problem, i.e. to a problem with a smaller value of r than $r=3$?



Well, we actually revert to the figure in step 1 by modifying $f_{-}(x)$ accordingly and the new $g_{-}(x)$ becomes $h_{-}(x)$. Actually, we have thus the same situation as at the beginning but with $r=2$.


```

sf_factors(f, p, x, k_ := 1, f_, g_, h_, u_, v_ := 1) :=
  Prog
  If NUMBER?(f)
    RETURN f
  f_ := polygcd(f, D(f, x), p, x)
  g_ := polyquot(f, f_, p, x)
  Loop
    If NUMBER?(g_) exit
    h_ := polygcd(f_, g_, p, x)
    u_ := polyquot(g_, h_, p, x)
    v_ := u_k_
    g_ := h_
    f_ := polyquot(f_, h_, p, x)
    k_ := k_ + 1
  f := polyquot(f, v_, p, x)
  f := Σ(POLY_COEFF(f, x, i_)·x^(i_/p), i_, 0, POLY_DEGREE(f), p)
  v_·sf_factors(f, p, x)p

```

$$\begin{aligned}
 & \text{sf_factors}(2 \cdot x^{16} + 2 \cdot x^{11} + x^9 + 2 \cdot x^8 + x^7 + x^6 + x^4 + 2 \cdot x^2 + 2 \cdot x + 1, 3) \\
 & 2 \cdot (x + 1) \cdot (x + 2)^3 \cdot (x^2 + 1)^2 \cdot (x^2 + 2 \cdot x + 2)^4
 \end{aligned}$$

If you have a look at the program above you will see that it performs exactly those five steps again and again. But you will also find out that I haven't yet told you the whole truth. In fact, if $p < r$ and $f(x)$ has irreducible factors of a multiplicity that is divisible by p then things are slightly more complicated. If, for example, $p=2$ then $f_2(x)$ has the same multiplicity in $f(x)$ and $f'(x)$ and as a consequence the "top layer" $h_2(x)$ in step 2 has a "hole" in the place of $f_2(x)$. I leave it to you to show that after performing the same cycle of 5 steps as before the column of $f_2(x)$ won't get smaller but stays always the same size. Hence, after some time we arrive at a polynomial that consists only of irreducible factors with multiplicities divisible by p . This is not really a problem though: We simply compute the p -th root of that polynomial (see the line before the last in the program) and apply the routine `sf_factors()` recursively to the result. Again see the program for details!

Okay, for a complete factorization we are left with the task of splitting up those squarefree factors $f_k(x)$ as well. In view of the previous results we may assume that $f(x)$ is squarefree from now on. The next and comparably easy step is to find a factorization of $f(x)$ into a product of polynomials $A_d(x)$ with the property that each of its irreducible factors of $A_d(x)$ in $\mathbb{Z}_p[x]$ has the same degree for some fixed d with $1 \leq d \leq \lfloor m/2 \rfloor$, where m is again the degree of $f(x)$. We simply use the fact here that the polynomial $x^{p^d-1} - x$ contains all irreducible polynomials in $\mathbb{Z}_p[x]$ of degree d exactly once. Hence, we can get $A_d(x)$ by computing the $\gcd(f(x), x^{p^d-1} - x)$ for the above values of $d=1,2,\dots, \lfloor m/2 \rfloor$, and removing the found factors from $f(x)$ after each step. There is a small subtlety though as regards the computation of the gcd above. Of course, before computing it the term x^{p^d-1} should be replaced by its value mod $f(x)$ using our `polypower()` again! For the other details I refer you to the following program. (Note that "dd" in its name stands for "distinct degree".)

```

dd_factors(f, p, x, d_ := 1, m_, u_ := 1, v_, w_) :=
  Prog
    w_ := x
    Loop
      m_ := POLY_DEGREE(f, x)
      If d_ > m_/2
        RETURN f·u_
      w_ := polypower(w_, p, f, p, x)
      v_ := polygcd(f, w_ - x, p, x)
      u_ := v_
      f := polyquot(f, v_, p, x)
      d_ := d_ + 1

```

$$\text{dd_factors}(x^{16} - x, 2) = x \cdot (x + 1) \cdot (x^2 + x + 1) \cdot (x^{12} + x^9 + x^6 + x^3 + 1)$$

As you may conclude from the example above $x^{16} - x$ splits up into two linear factors, one quadratic factor and 3 factors of degree 4, where the latter are not yet separated. This separation is the hard work in practice and everything we have done so far has been child's play by comparison. There are many different methods to achieve this goal (see for example Berlekamp's method in [1]) and all have different advantages and drawbacks, but to cut a long story short I'll only update the method by Cantor-Zassenhaus already introduced in my Titbits #13. Note that for the following programs it is tacitly assumed that the given polynomial f consists only of irreducible factors of some fixed degree d .

```

polysplit(f, d, p, x, t_) :=
  Loop
    If POLY_DEGREE(f, x) = d
      RETURN f
    t_ := Σ(RANDOM(p)·x^k_, k_, 0, 2·d - 1)
    t_ := polygcd(f, polypower(t_, (p^d - 1)/2, f, p, x) - 1, p, x)
    t_ := [t_, polyquot(f, t_, p, x)]
    If SELECT(NUMBER?(u_), u_, t_) = []
      RETURN polysplit(t_↓1, d, p, x)·polysplit(t_↓2, d, p, x)

```

$$\text{polysplit}(x^{12} + x^9 + x^6 + x^3 + 1, 4) = (x^4 + x + 1) \cdot (x^4 + x^3 + 1) \cdot (x^4 + x^3 + x^2 + x + 1)$$

Let's turn to promise now I gave in the introduction, namely to show how testing Mersenne numbers, i.e. numbers of the form $M_p = 2^p - 1$, where p is a prime, for primality, is connected up with the problem of multiplying two polynomials as fast as possible. First, let me remind you that if exclude the prime $p=2$ then there is the incredibly simple Lucas-Lehmer test for those numbers (see e.g. my Titbits #22). All you have to do is to set $s:=4$ and perform the assignment $s:=s^2 - 2 \bmod M_p$ exactly $p-2$ times. Then M_p is prime if and only if $s=0$ for the resulting value of s . I have also already mentioned that the reduction mod M_p is simply a matter of shifting and adding for a binary computer. Hence, what remains to be done and is the really hard part is the squaring of s . (Remember that we are talking about an s that has the size of several million digits currently!)

Assuming that s is represented as a polynomial in the base B , that is

$$s = s_0 + s_1 B + \dots + s_{n-1} B^{n-1}$$

where B is the word size of the computer, say $B=2^{32}$ or $B=2^{64}$, and n is large enough to cover all numbers below M_p , then we have indeed the problem of a fast polynomial multiplication. Experts in this field already know that this is the point where FFT (=Fast Fourier Transform) inevitably comes into play. The basic idea behind it is very simple. Instead of multiplying the polynomials using the school method, i.e. the Cauchy multiplication for which $O(n^2)$ multiplications are needed, we evaluate the polynomials we want to multiply in $2n-1$ suitably chosen places $x_0, x_1, \dots, x_{2n-2}$, carry out the pointwise multiplication for those places and solve the interpolation problem afterwards to get the product of the given polynomials.

In the case of the DFT (=Discrete Fourier Transform) we choose $x_i = \omega^i$, $i=0,1,\dots,2n-2$, for some fixed primitive n -th root of unity ω in the given ring R of coefficients. For example, in the classical case where $R=\mathbb{C}$, the field of complex numbers, we could take $\omega = e^{2i\pi/n}$ or $\omega = e^{-2i\pi/n}$. In more general rings R , the statement " ω is an ' n -th primitive root of unity'" means that ω is a an n -th root of unity in the ring R . Moreover none of the values $\omega^k - 1$, $0 < k < n$, must be a zero divisor in R and n must be invertible in R .

Ok, let's start with the classical case $R=\mathbb{C}$ in order to see what I'm talking about. The evaluation for a polynomial for the first n powers ω^k , $k=0,1,\dots,n-1$, is simply a multiplication with the matrix Ω whose definition be seen in the following initialization routine. What is less obvious, but can be proven without great efforts, is the amazingly simple form of the inverse matrix, called Ω_* in the following program, which looks very similar to Ω apart from the fact that it uses the inverse root of unity ω^{-1} and has also got a constant factor $1/n$. (By the way the latter fact is why we require the invertibility of n .)

```

init(n, i_ := 1, j_ := 1) :=
  Prog
    OMEGA := IDENTITY_MATRIX(n)
    OMEGA_* := OMEGA
    Loop
      j_ := 1
      Loop
        OMEGA[i_]j_ := EXP(2*i_*(j_ - 1)*pi/n)
        OMEGA_*[i_]j_ := EXP(- 2*i_*(j_ - 1)*pi/n)
        j_ := j_ + 1
        If j_ > n exit
      i_ := i_ + 1
      If i_ > n exit

```

```

DFT(a) := a.OMEGA

```

```

DFT_inv(a) :=  $\frac{1}{\text{DIM}(a)}$ .a.OMEGA_*

```

```

init(8) = true

```

$$\text{OMEGA} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & \frac{\sqrt{2}}{2} + \frac{\sqrt{2} \cdot i}{2} & i & -\frac{\sqrt{2}}{2} + \frac{\sqrt{2} \cdot i}{2} & -1 & -\frac{\sqrt{2}}{2} - \frac{\sqrt{2} \cdot i}{2} & -i & \frac{\sqrt{2}}{2} - \frac{\sqrt{2} \cdot i}{2} \\ 1 & i & -1 & -i & 1 & i & -1 & -i \\ 1 & -\frac{\sqrt{2}}{2} + \frac{\sqrt{2} \cdot i}{2} & -i & \frac{\sqrt{2}}{2} + \frac{\sqrt{2} \cdot i}{2} & -1 & \frac{\sqrt{2}}{2} - \frac{\sqrt{2} \cdot i}{2} & i & -\frac{\sqrt{2}}{2} - \frac{\sqrt{2} \cdot i}{2} \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & -\frac{\sqrt{2}}{2} - \frac{\sqrt{2} \cdot i}{2} & i & \frac{\sqrt{2}}{2} - \frac{\sqrt{2} \cdot i}{2} & -1 & \frac{\sqrt{2}}{2} + \frac{\sqrt{2} \cdot i}{2} & -i & -\frac{\sqrt{2}}{2} + \frac{\sqrt{2} \cdot i}{2} \\ 1 & -i & -1 & i & 1 & -i & -1 & i \\ 1 & \frac{\sqrt{2}}{2} - \frac{\sqrt{2} \cdot i}{2} & -i & -\frac{\sqrt{2}}{2} - \frac{\sqrt{2} \cdot i}{2} & -1 & -\frac{\sqrt{2}}{2} + \frac{\sqrt{2} \cdot i}{2} & i & \frac{\sqrt{2}}{2} + \frac{\sqrt{2} \cdot i}{2} \end{bmatrix}$$

a := [1, ..., 8]

b := DFT(a)

[36, -4 - i·(4·√2 + 4), -4 - 4·i, -4 + i·(4 - 4·√2), -4, -4 + i·(4·√2 - 4), -4 + 4·i, -4 + i·(4·√2 + 4)]

DFT_inv(b) = [1, 2, 3, 4, 5, 6, 7, 8]

Let's turn to the FFT, what might be called the "fast version" of DFT although it works only if n is a power of two. It is basically a "divide-and-conquer" algorithm which reduces the problem of finding the value of DFT(a) w.r.t. to chosen ω to the problem of finding DFT(a_{even}) and DFT(a_{odd}) w.r.t. ω^2 , where a_{even} and a_{odd} denote the subvectors of a with even or oddnumbered indices, respectively. (For the details see the following program.) The important point about FFT is that the complexity is now only O(n log n) multiplications and hence considerably better than the corresponding complexity O(n²) for the ordinary DFT.

$\omega :=$

FFT(a, ω , f_, f_, n_, x_) :=

 Prog

 n_ := DIM(a)

 If n_ = 1

 RETURN a

 f_ := FFT(VECTOR(a[i_], i_, 1, n_, 2), ω^2)

 x_ := ITERATES(ω , ω , 1, n_/2 - 1)

 f_ := FFT(VECTOR(a[i_], i_, 2, n_, 2), ω^2)

 f_ := VECTOR(x_₁·f_₁, i_, 1, n_/2)

 APPEND(f_, f_) + APPEND(f_, -f_)

FFT_inv(a, ω) := $\frac{1}{\text{DIM}(a)} \cdot \text{FFT}(a, \omega^{-1})$

$a := [1, \dots, 8]$

$b := \text{FFT}\left(a, \text{EXP}\left(\frac{2 \cdot i \cdot \pi}{8}\right)\right)$

$[36, -4 - i \cdot (4 \cdot \sqrt{2} + 4), -4 - 4 \cdot i, -4 + i \cdot (4 - 4 \cdot \sqrt{2}), -4, -4 + i \cdot (4 \cdot \sqrt{2} - 4), -4 + 4 \cdot i, -4 + i \cdot (4 \cdot \sqrt{2} + 4)]$

$\text{FFT_inv}\left(b, \text{EXP}\left(\frac{2 \cdot i \cdot \pi}{8}\right)\right) = [1, 2, 3, 4, 5, 6, 7, 8]$

Maybe I shouldn't close this topic before giving an explicit demonstration how these ideas can be used for a fast multiplication of two numbers. For this we need another version of FFT working in a residue class ring mod m for a suitably chosen m . We use in the following the fact that $\omega = 2$ is a primitive

n -th root of unity if $n = 2^{v+1}$ and $m = 2^{n/2} + 1$

$\text{fft}(a, \omega, m, f_ , f_ , n_ , x_) :=$

```

Prog
  n_ := DIM(a)
  If n_ = 1
    RETURN a
  f_ := fft(VECTOR(a↓i_, i_, 1, n_, 2), ω^2, m)
  x_ := ITERATES(MOD(ω_·ω, m), ω_, 1, n_/2 - 1)
  f_ := fft(VECTOR(a↓i_, i_, 2, n_, 2), ω^2, m)
  f_ := VECTOR(x_↓i_·f_↓i_, i_, 1, n_/2)
  MOD(APPEND(f_, f_) + APPEND(f_, -f_), m)

```

$\text{fft_inv}(a, \omega, m) := \text{MOD}(\text{INVERSE_MOD}(\text{DIM}(a), m) \cdot \text{fft}(a, \text{INVERSE_MOD}(\omega, m), m), m)$

$\text{fft}([1, \dots, 8], 2, 17) = [2, 8, 14, 6, 13, 3, 12, 1]$

$\text{fft_inv}([2, 8, 14, 6, 13, 3, 12, 1], 2, 17) = [1, 2, 3, 4, 5, 6, 7, 8]$

For the conversions of the number representation to the polynomial representation and vice versa I have written the following routines:

$\text{tovector}(x, b := 10, l := 1, d_ := []) :=$

```

Prog
  Loop
    If x = 0 exit
    d_ := INSERT(MOD(x, b), d_, 0)
    x := FLOOR(x, b)
  Loop
    If DIM(d_) ≥ l
      RETURN d_
    d_ := INSERT(0, d_, 0)

```

$\text{tonumber}(v, b := 10, x_ := 0, y_ := 1) :=$

```

Prog
  Loop
    If v = []
      RETURN x_
    x_ := v↓1·y_
    y_ := * b
    v := REST(v)

```

And finally a simple example to see those routines at work:

$$1324563472323451 \cdot 243272343213275 = 322229659646837853175477012025$$

```
x := tovector(1324563472323451, 10, 32)
```

```
[1, 5, 4, 3, 2, 3, 2, 7, 4, 3, 6, 5, 4, 2, 3, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

```
v := tovector(243272343213275, 10, 32)
```

```
[5, 7, 2, 3, 1, 2, 3, 4, 3, 2, 7, 2, 3, 4, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

$$\text{prod}(u, v) := \text{VECTOR}(u_i \cdot v_i, i, 1, \text{DIM}(u))$$

```
fft_inv(prod(fft(x, 2, 65537), fft(y, 2, 65537)), 2, 65537)
```

```
[5, 32, 57, 56, 55, 54, 61, 91, 125, 112, 135, 167, 165, 167, 170, 169, 135, 153, 130, 120, 133, 115, 93, 76, 71,
 45, 27, 19, 10, 2, 0, 0]
```

```
tonumber([5, 32, 57, 56, 55, 54, 61, 91, 125, 112, 135, 167, 165, 167, 170, 169, 135, 153, 130, 120, 133, 115, 93,
76, 71, 45, 27, 19, 10, 2, 0, 0])
```

322229659646837853175477012025

Whow, it works! Note that the vectors x and y corresponding to the two numbers that are to be multiplied have to be filled up with that many zeros such that there won't be any "overflow" due to the polynomial multiplication. This procedure is called zero-padding.

Of course, only the principle could be shown here and many subtleties are involved in practice. In particular, it goes without saying that programming in assembler code is mandatory. Would you believe that those programs are so highly sophisticated that certain versions are used to test Pentium chips before delivery?

Okay, let's turn to a more down-to-the-earth question at the end of the Titbits namely the number of digits for the new record prime. (The length of this article is also a new record by the way, but after all Josef has asked for it and you should write letters of complaint to him!)

Here is the Derive routine that computes number of digits for a Mersenne number M_p , no matter whether it is prime or not.

$$\text{digits}(p) := \text{FLOOR}(p \cdot \text{LOG}(2, 10)) + 1$$

```
digits(20996011) = 6320430
```

Well, this computation should be rather obvious. Nevertheless, just to be on the safe side, I checked this number on the Internet site quoted above, as you should in turn check everything I said on the preceding pages. (But this you did anyway, didn't you?)

Hope you enjoyed the programs as much as I did when writing them. And as always, any comments, questions or suggestions (to j.wiesenbauer@tuwien.ac.at) are welcome!

[1] A. Menezes, P. van Oorschot and S. Vanstone, Handbook of Applied Cryptography, CRC Press, Boca Raton, FL, 1996 (cf. also <http://www.cacr.math.uwaterloo.ca/hac/>)

Additional note: together with Titbits #25 you have a very complete collection of functions and programs for polynomial arithmetic. You can find Johann's files for DERIVE versions 5 and 6 on the diskette. Josef