THE DERIVE - NEWSLETTER #51

THE BULLETIN OF THE

DERIVE®

USER GROUP

+ CAS-TI

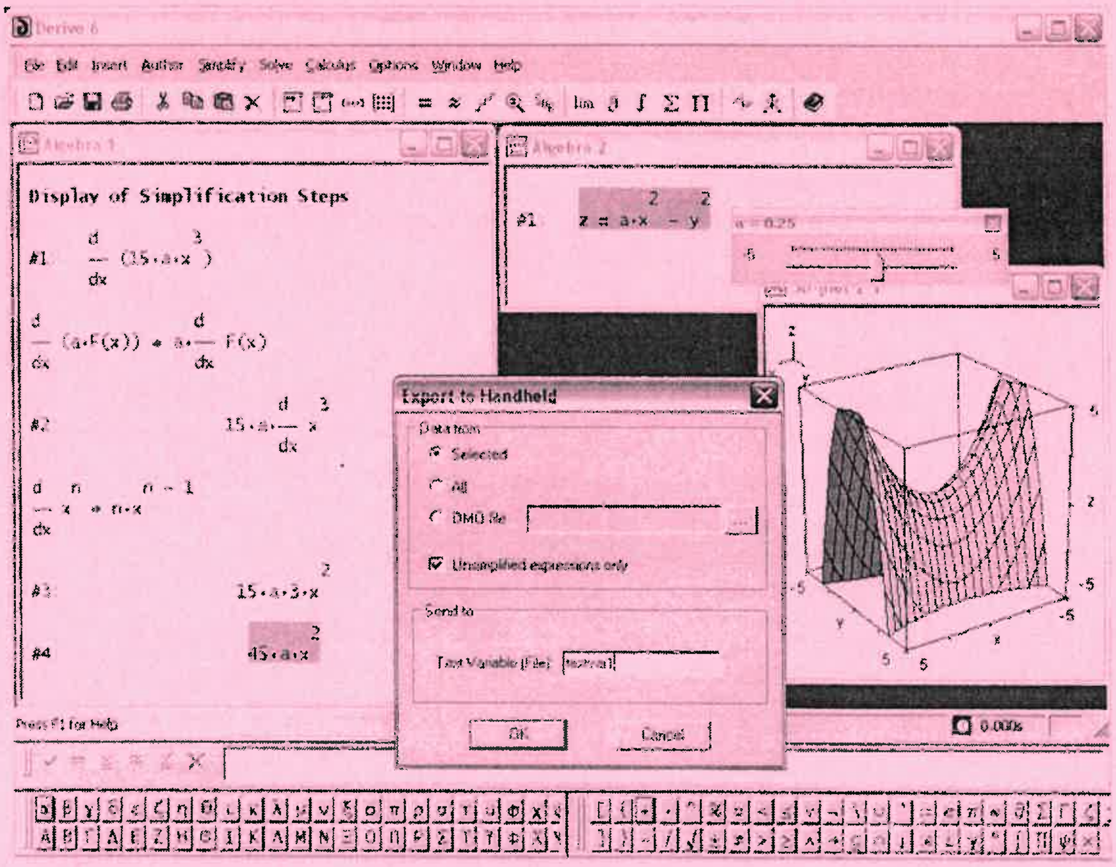
Contents:

September 2003

By the end of October the new Derive 6 will be released. Currently, Derive 6 runs only under Windows2000 and WindowsXP.

New Features in Derive 6:

- display the steps in the simplification of an expression with optional display of transformation rules
- communicate with TI CAS calculators: import data from and export data to TI-89, TI-92+, Voyage200 calculators
- make plots dynamic: animate expression plots with slider bars
- let plots be labelled with defining equations
- rotate 3D plots with mouse
- get support via improved and extended online help
- customize menus, toolbars, and shortcut key
- profit from numerous other improvements, including
  * optional multi-line editing
  * fully scaleable Derive Unicode font
  * state variables now saved in DfW files
  * parentheses matching
  * different styles for connected points
  * controllable display of 3D mesh lines
  * controllable point size of 3D data-point plots
  * function for computing Groebner bases


Derive 6
File Edit Insert Author Simplify Solve Calculus Options Window Help
Algebra 1
Algebra 2
Press F1 for Help
0.000s

Display of Simplification Steps

#1 $\frac{d}{dx}(15\cdot a\cdot x^3)$

$\frac{d}{dx}(a\cdot F(x)) \to a\cdot\frac{d}{dx}F(x)$

#2 $15\cdot a\cdot\frac{d}{dx}x^3$

$\frac{d}{dx}x^n \to n\cdot x^{n-1}$

#3 $15\cdot a\cdot 3\cdot x^2$

#4 $45\cdot a\cdot x^2$

#1 $z = a\cdot x^2 - y^2$

a = 0.25

Export to Handheld

Data from: [✓ Selected] All DMO file

[✓] Unsimplified expressions only

Send to: Text Variable (File): [illegible]

OK Cancel

Dear friends,

after the summer of the century at least for most of the European countries we are heading fall 2003. This *DNL*#51 provides a smaller content than usual, only 32 pages instead of 44. The reason is the following: Very surprisingly the Austrian Post increased mailing charges for sending books and journals by approx 100% in June. So I decided to produce a smaller *DNL*#51 to keep the mailing charges lower now and to add the missing pages to *DNL*#52 in December which will be sent together with the traditional diskette. I hope that you will understand to my decision.

Three colleagues faced the Lindenmayer-Challenge from *DNL*#50. Phillips MacDonald was the first to answer, Stefan Welke sent the most extended contribution and Rüdeger Baumann presents his own way to produce the Snake Kolam = Sierpinski Curve. Many thanks to you all and congratulations to your success. My own full program will be given in *DNL*#52. There is a wonderful website treating among others L-Systems, which I strongly recommend to visit:

`www.fh-lueneburg.de/u1/gym03/homepage/faecher/mathe/chaos/linde/linde.htm.`

As promised in the last *DNL*, Tim Comar presents Möbius Transformations, which I tried to transfer to the TI-CAS. This is one more example that in most cases ideas are not fixed on one platform and that it is often not very difficult to realize a *DERIVE*-contribution on the handheld and vice versa. I do not know the reason, but TI-CAS contributions for publication in the DNL are very rare.

On pages 3 and 4 you can find an extended abstract of Paul Drijvers PhD paper which appeared in printed form. The whole DUG-family congratulates Paul to his academic success. Paul has been a *DERIVIAN* and CAS-enthusiast from the very first hour. And what is much more important: he has been a very good and cooperative friend for years. Many thanks, Paul, and the best wishes for the future.

2004 is the year for the next big *DERIVE* & TI-CAS-Conference which will be hold in the time 15 – 18 July in Montreal, Canada. Michel Beaudin and his team are busy to install a website. It would be great to meet many of you next summer in Maple-Leaf-State.

The last important point which I have to address in my letter is the release of *DERIVE Version 6*. DfW6 will be released very soon. Bernhard Kutzler provided an overview of the most exciting new features of version 6.

From my point of view the most innovative and impressing feature is the connectivity between PC and Handheld CAS. This could initiate a new era of CAS-supported Maths education. We are no longer fixed on either using the PC or the handheld device, but we can easily switch between the platforms with no syntax and other problems and difficulties.

Best regards

Josef [illegible]

Dear DUG-members,
please inform us about new publications on the use of *DERIVE* and/or the CAS-TIs.
We also appreciate all information about interesting websites.

`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.

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 with a contents of 44 pages minimum. The goals of the *DNL* are to enable the exchange of experiences made with *DERIVE* and the *TI*-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.

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

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.

Next issue:	December 2003
Deadline	15 November 2003

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-Encription, 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
The "Joseph-Game", Rüdeger Baumann, GER
Simulating a Graphing Calculator in *DERIVE*, J.Böhm, AUT
Lindenmayer-Systems, J. Böhm, AUT
Cellurar Automata – "Rule 90" a.o., D.Sjöstrand/J.Böhm, SWE-AUT
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

Learning Algebra in a Computer Algebra Environment

Design research on the understanding of the concept of parameter

Dr. Paul Drijvers

On 25 September 2003, Utrecht University (the Netherlands) awarded Paul Drijvers a PhD degree for his thesis 'Learning algebra in a computer algebra environment'. The following abstract provides a brief overview of his work.

It is well known that algebra is a difficult topic in the school mathematics curriculum, and is often experienced as a stumbling-block. One of the directions in which solutions to the problems with the learning of algebra can be sought is the integration of information technology (IT) into mathematics education. Although originally not developed for educational purposes, a computer algebra system is an IT tool that seems promising because of its algebraic power: it expands expressions such as (a + b)2 and solves equations such as 3x + 5 = 7. By doing so, a computer algebra system carries out operations, which are seen as the core of mathematics, and which students so far have to master by hand. The basic aim of this study, therefore, is to investigate whether computer algebra use can contribute to the understanding of algebra. This leads to the following main research question:

How can the use of computer algebra promote the understanding of algebraic concepts and operations?

To answer this question, the concept of parameter was chosen as the central topic. A teaching sequence for high-ability students in ninth and tenth grades (14- to 16-year-olds) was developed, in which the use of computer algebra was integrated. Following the research paradigm of developmental research, this sequence was tested in three cycles of teaching experiments, in which over one hundred students were involved and over one hundred lessens were observed. During these teaching experiments the students all had a TI-89 symbolic calculator at their disposal. The data consisted of observations, interviews, tests and written student work. These data were analysed to investigate how the use of computer algebra influenced the students' learning, how the students developed thinking schemes for machine techniques and how this supported their algebraic insight.

The analysis yielded mixed results. On the one hand, computer algebra turned out to be a complex tool for students of this level, which puts high demands on the user: students had to explain to the machine exactly what should be done, and in some cases got results that were hard to understand. Also, computer algebra sometimes seemed a black box to them. On the other hand computer algebra use offered opportunities for the learning. The computer algebra system could be used as an environment for algebraic experimentation, which induced the discovery of algebraic relations. Furthermore, the use of computer algebra required that the student was aware of the problem solving procedure and the sub-procedures that are part of it. Entering formulas and interpreting results contributed to the development of 'symbol sense', the for algebra so important feeling for the structure and meaning of expressions and formulas.

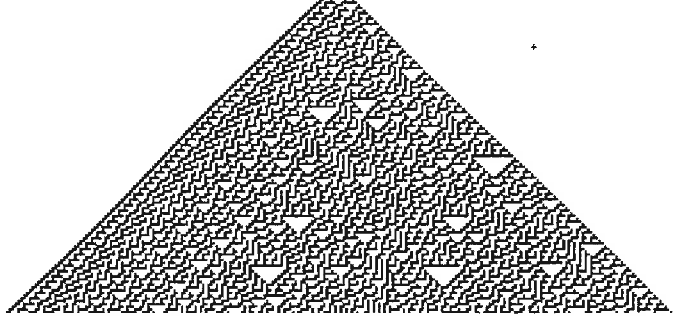
These findings underline the relevance of the dual learning process which students go through while using computer algebra: they have to learn machine techniques for solving types of problems, but this is related to a conceptual development. This so-called instrumental genesis can be promoted by appropriate student assignments and by teacher behaviour that pays attention to the possibilities of the machine in relation to the conceptual insights behind it. There is no reason to fear that students no longer need to think while using computer algebra: technical skills and algebraic insight develop simultaneously.

The thesis is structured as follows. Chapter 1 contains the research questions and explains the aims and backgrounds of the study. In Chapter 2 the research design and methodology are described. Key words are design research and hypothetical learning trajectory. Chapters 1 and 2 together indicate what the research is about and how it is conducted. Chapters 3, 4 and 5 form the theoretical part of the thesis. They treat the main themes of the study: algebra in general, the concept of parameter in particular and the possible roles of computer algebra. Chapter 3 concerns algebra in general. It sketches different views on algebra and describes the standpoint of this study. The theoretical issues of symbol sense, symbolizing, the process-object duality and Realistic Mathematics Education are addressed. Chapter 4 zooms in on the concept of parameter. After a brief historical perspective, a conceptual analysis of the parameter is given. Then the higher level understanding of the concept of parameter is described. This is connected to the theoretical notions from Chapter 3. Chapter 5 deals with the tool that students use in this research project: computer algebra. Besides an overview of previous research in this domain, it contains a description of the theory of instrumentation that will be used in Chapter 10 in particular. Chapters 6 - 10 form the empirical part of the dissertation. Chapters 6, 7 and 8 describe the development of the hypothetical learning trajectory and the classroom experiences during the three subsequent research cycles. Chapter 9 concerns the contribution of computer algebra use to the understanding of the concept of parameter. In Chapter 10, the results concerning the instrumentation of computer algebra are presented.
Chapter 11, finally, answers the main research question. After that, the results and the methodology are discussed. Also, the relevance of the theoretical framework and the generalizability of the findings are evaluated. The chapter ends with recommendations for teaching, for software design and for further research.

Drijvers, P.H.M. (2003). *Learning algebra in a computer algebra environment. Design research on the understanding of the concept of parameter.* Utrecht: CD-β Press. ISBN 90-73346-55-X

For an extended summary, downloads and hard copy see www.fi.uu.nl/~pauld/dissertation.

Two results of cellurar automata – "Rule 90" and others



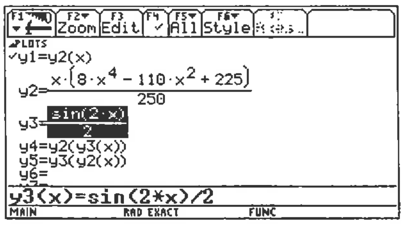
A not well-known Interaction between CAS and Dynamic Geometry on the TI-92/Voyage 200

We will design a dynamic geometry model (CABRI), which plots the graph of any function defined in the [Y=]-Editor for further use with special respect to possible animations as

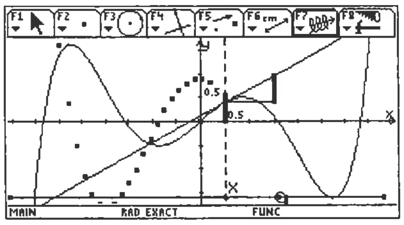
(1) to plot the function graph,
(2) to move a point on the graph and simultaneously visualize the slope,
(3) to move the point together with the tangent (or the perpendicular to the tangent),
(4) to transfer the slope for simultaneous plotting the derivative as a locus,
(5) to plot the envelope of the perpendicular lines,
(6) to plot pedal curves as a locus,
....

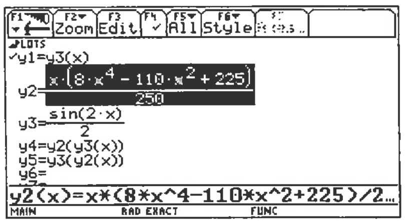
After changing the function in the [Y=]-Editor, the model will immediately be adapted.
See an example:

At first we show a polynomial function of degree 5 together with the slope and tangent in the moving – or animated – point creating pointwise the graph of the derivative.

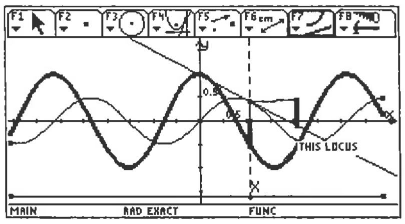


```
PLOTS
✓y1=y2(x)
 y2=x·(8·x^4 − 110·x^2 + 225)/250
 y3=sin(2·x)/2
 y4=y2(y3(x))
 y5=y3(y2(x))
 y6=
y3(x)=sin(2*x)/2
MAIN          RAD EXACT          FUNC
```





```
PLOTS
✓y1=y3(x)
 y2=x·(8·x^4 − 110·x^2 + 225)/250
 y3=sin(2·x)/2
 y4=y2(y3(x))
 y5=y3(y2(x))
 y6=
y2(x)=x*(8*x^4-110*x^2+225)/2…
MAIN          RAD EXACT          FUNC
```



Then we change the function and we can see the same model applied on a sine function. Here we plotted the locus of the derivative points in thick style.

How can we achieve this? I assume that you have some experience in working with the CABRI-Application on the TI. If not the I strongly recommend to discover this wonderful tool.

(1) Define an appropriate function in the [Y=]-Editor, e.g. `y1(x) = sin(2x)/2`.

(2) Switch to the Geometry-Application, draw a horizontal segment on the bottom of the screen and fix point X on it. This point is moveable and forms the reference point for all appearing function arguments.

(3) Draw a vertical line through X and intersect with the x-axis giving auxiliary point T.

(4) Show coordinates of X ([F6]).

(5) Via [F6]: `Calculate` calculate the function value y1(x-coordinate of X).

(6) Check: Grab point X with [✋], move it along the segment and you should observe the resepctive function value in R: . . . . on the screen.

(7) Transfer this function value to the vertical line passing T ([F4]: `Measurement Transfer`), which results in a point P of the function graph. Check: Drag or Animate X and observe the run of point P.

(8) $\boxed{\text{F4}}$: Locus, Plot the locus of point P. You should see the function graph. If necessary increase the number of locus points ($\boxed{\blacklozenge}$ $\boxed{\text{F}}$). Check: Define another function as `y1(x)` and go back to the Geometry Application and hopefully you will see the changed graph.

(9) Hide the function Value ($\boxed{\text{F7}}$).

Now CAS enters the stage – we visualize the slope:

(10) Draw a horizontal line through P.

(11) With $\boxed{\text{F6}}$`:Calculate` "calculate" the value 1 and transfer this measurement from P to the right on the horizontal line, giving point Q.

(12) Draw a vertical line through Q.

(13) $\boxed{\text{F6}}$`:Calculate d(y1(x),x)|x` = x-coordinate of X.

(14) $\boxed{\text{F4}}$: `Measurement Transfer` this number on the vertical line from Q, giving Q'.

(15) Hide the auxiliary lines, draw the $\boxed{\text{F3}}$`:Triangle P, Q, Q'` and draw the tangent which is the line defined by P and Q'.

(16) Animate X ($\boxed{\text{F7}}$`:Animation`) on the segment and the tangent should move along.

Produce the graph of the derivative:

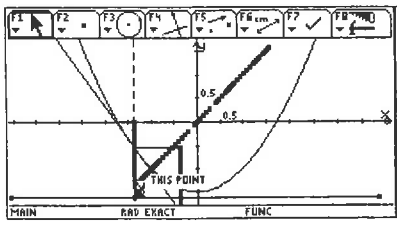
(17) The value of the 1st derivative is still on the screen and can be transferred from T on the vertical line, giving P' which is a point of the first derivative.

(18) Hide the vertical line and create two segments ($\boxed{\text{F2}}$) TP and TP'.

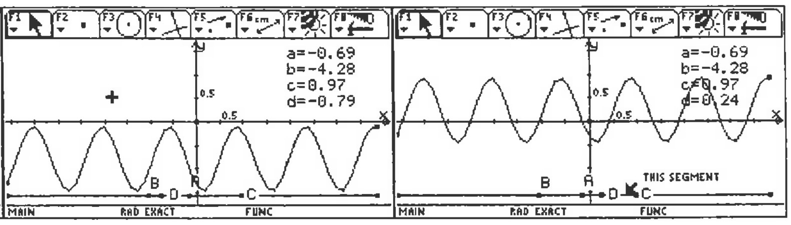
(19) For P' setting $\boxed{\text{F7}}$`:Trace On`, the graph of the 1st derivative will appear pointwise while moving X along the segment on the bottom of the screen. You can individually add some changes. I plot the two segments QQ' and TP' both $\boxed{\text{F7}}$`:Thick` to underline their equivalence.

(20) Applying $\boxed{\text{F4}}$`:Locus` on P' you will find the graph of the first derivative.

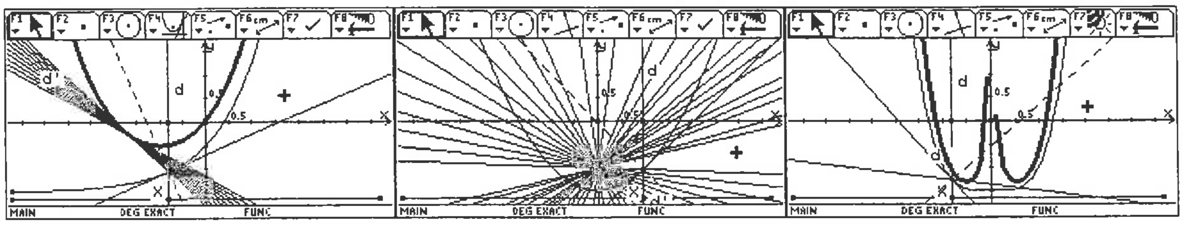
(21) It is no problem to add the second derivative, and, and, and .....



Derivative of a quadratic



Visalization of the influence of parameters a,b,c and d on *a* f(*b*(x+*c*)+*d*



Vertical rays are reflected on various graphs. What is the envelope of the reflected rays?

The figures can be downloaded. Don't forget to define a function as y1(x). Otherwise the model would not work. It makes sense to save the fist part (Plotting the function graph) separately as initial point for other purposes. Many thanks to P. Frachebourg for valueable hints. `http://www.stmaurice.ch/pfrache`.

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

Timothy D. Comar
Benedictine University
Department of Mathematics
tcomar@benedu

Introduction

We describe an activity using *DERIVE* to illustrate the beautiful geometric behavior of Möbius transformations in both two and three dimensions. The activity is designed to meet several important goals: geometric visualization, hands-on activities, reinforcement of basic mathematics, and the establishment of connections between several areas of mathematics. We also provide a brief review of Möbius transformations and hyperbolic geometry and illustrate some *DERIVE* routines from the package **mobmath.mth** [2] created by the author.

A wonderful aspect of *DERIVE* as a pedagogical tool is that it is a versatile computer algebra system that still requires the user to perform a fair amount of basic precalculus mathematics to create more sophisticated examples and explorations. The activity described here can be presented to students at several levels. At a higher level, students could be expected to create from scratch the *DERIVE* routines to construct the examples. At a lower level, the instructor can provide the students with a copy of the collection of the routines in the file **mobmath.mth** [2] and more detailed instructions about constructing the examples. Even though the activity is designed for *DERIVE*, it is easily transferable to any computer algebra system with graphing capabilities.

Geometric Visualization

The primary goal of the activity described in this paper is to obtain deep geometric intuition about the behavior of hyperbolic Möbius transformations. In particular, the activity focuses on developing the inter-relationship between the two and three-dimensional behavior of Möbius transformations.

Hands-on Activities, Reinforcement, Connections

The activity approaches new mathematical content by investigating concrete examples that require the students to perform basic calculations using familiar mathematics: basic geometry, trigonometry, and parametric representations. Even though *DERIVE* is used to perform the necessary calculations, the student must demonstrate a sound understanding of the basic mathematics to be able to enter the appropriate expressions into *DERIVE*. This dependence on precalculus mathematics can help demystify the abstraction of new mathematics as well as reinforce and re-integrate this basic mathematics in advanced courses outside the calculus sequence.

The novel aspect of this investigation between the two and three-dimensional geometry is that the activity introduces the mathematics in a suitable manner for undergraduates, whereas the three-dimensional geometry of Möbius transformations is not part of the standard undergraduate course in complex analysis and is not introduced, if at all, until graduate courses in topology or hyperbolic manifolds. In particular, this activity illustrates connections between the advanced mathematical topics of complex variables and hyperbolic geometry as well as notions from topology and algebra. Moreover, the activity provides a nice illustration of applying basic mathematics, which can help pre-service secondary education students form deeper connections between undergraduate mathematics and the concepts that they will teach in secondary school.

Möbius Transformations

Let **M** denote the group of Möbius transformations $z \mapsto \frac{az+b}{cz+d}$ acting on $\hat{\mathbf{C}} = \mathbf{C} \cup \{\infty\}$, where a, b, c, d are complex numbers such that $ad - bc = 1$. The group **M** is the group of orientation-preserving conformal automorphisms of $\hat{\mathbf{C}}$ and is called *Möbius Group*. We can further identify **M** with the group **PSL(2,C)** (Projective Special Linear group), the group of 2 × 2 matrices with complex coefficients, determinant 1, modulo the equivalence relation $A \sim -A$. By using the 2 × 2 matrix representation for Möbius transformations, we can easily create short routines in *DERIVE* to calculate properties of the Möbius transformations and obtain parametric representations which we can plot.

In the upper halfspace model of hyperbolic three-space $\mathbf{H}^3$, hyperbolic planes are either vertical planes or hemispheres which are orthogonal to its boundary at infinity, $\hat{\mathbf{C}}$. In both cases, the boundary at infinity of a hyperbolic plane is either a circle or line, which we consider as a circle through ∞; For simplicity, we refer to both circles and lines as *generalized circles*. Each element in **M** is the product of an even number of reflections in circles in $\hat{\mathbf{C}}$. We can extend the action of each element in **M** to an orientation-preserving isometry of the upper halfspace model of hyperbolic three-space $\mathbf{H}^3$ by reflecting the corresponding hyperbolic planes bounded by the generalized circles in $\hat{\mathbf{C}}$ which generate the Möbius transformation. This extension is called the *Poincaré extension* and takes the following form in the upper halfspace model:

$$\tilde{f}(z,t) = \left(\frac{(az+b)\overline{(cz+d)} + a\bar{c}t^2}{|cz+d|^2 + |c|^2 t^2}, \frac{|ad-bc|\,t}{|cz+d|^2 + |c|^2 t^2} \right),$$

where $f(z) = \frac{az+b}{cz+d}$. (See [1] for details.) We thus can identify the goup $\mathrm{Isom}^+(\mathbf{H}^3)$ of orient-tion preserving isometries of $\mathbf{H}^3$ with **PSL(2,C)**. Using the Poincaré extension, we have created a *DERIVE* routine `Poincare_Ext_Image` [2] to obtain parameterizations of images of hyperbolic planes under Möbius transformatsions to geometrically investigate the three-dimensional geometric aspects of Möbius transformations. For ease of exposition, we will freely represent points in the plane either as ordered pairs of the form (x, y) or as complex numbers of the form $x + i\,y$, as convenient. Similarly, we represent points in $\mathbf{H}^3$ either as ordered triples of real numbers of the form (x, y, t) or as ordered pairs $(x + i\,y, t)$ consisting of a complex number in the first coordinate and a positive real number in the second coordinate.

Classification of Möbius Transformations

Note that every element of **M** has at least one fixed point on $\hat{\mathbf{C}}$, and that any element of **M** fixing more than two points is the identity. For $\mathbf{M} - \{id\}$, we have:

1. A *hyperbolic* transformation has two fixed points in $\mathbf{H}^3 \cup \hat{\mathbf{C}}$, both of which are in $\hat{\mathbf{C}}$. Any matrix representing a hyperbolic transformation can be conjugated to one of the form $\begin{bmatrix} \lambda^{1/2} & 0 \\ 0 & \lambda^{-1/2} \end{bmatrix}$, where $|\lambda| \neq 1$. This particular matrix acts on $\mathbf{H}^3$ by $(z,t) \mapsto (\lambda z, |\lambda|\, t)$. The hyperbolic transformation is *purely hyperbolic* if λ is real and positive and is otherwise *loxodromic*.

2. An *elliptic* transformation fixes two points on $\hat{\mathbf{C}}$ as well as each point on the geodesic axis in $\mathbf{H}^3$ joining these fixed points on $\hat{\mathbf{C}}$. Any matrix representing an elliptic transformation can be conjugated to one of the form $\begin{bmatrix} \lambda^{1/2} & 0 \\ 0 & \lambda^{-1/2} \end{bmatrix}$, where $|\lambda| = 1$, $\lambda \neq 1$. This particular matrix acts on $\mathbf{H}^3$ by $(z,t) \mapsto (\lambda z, |\lambda| t)$.

3. A *parabolic* transformation fixes one point on $\hat{\mathbf{C}}$ and none in $\mathbf{H}^3$. Any matrix representing a hyperbolic transformation can be conjugated to one of the form $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$. This particular matrix acts on $\mathbf{H}^3$ by $(z,t) \mapsto (z+1,t)$.

The Activity: Visualization of Hyperbolic Transformations

The transformation that we primarily use in the activity is the pure hyperbolic tranformation $f(z) = 4z$, which has a matrix representation

$$M = \begin{bmatrix} 2 & 0 \\ 0 & \frac{1}{2} \end{bmatrix}.$$

Step 1. Express the hyperbolic transformation f(z) as the product (composition) of two reflections.

Let r_1 be the reflection in the unit circle, centered at the origin of the plane, and let r_2 be reflection in the circle of radius 2 centered at the origin of the plane. Then

$$r_1(z) = \frac{1}{\bar{z}},\ r_2(z) = \frac{4}{\bar{z}},\ \text{and } f(z) = r_2(r_1(z)).$$

Step 2. Let C(r, x, y) denote the circle of radius r centered at the point x + i y in the plane. Plot the circle C(0.25,0.5,0), its image, f(C(0.25,0.5,0)), under the mapping f(z), and then f(f(C(0.25,0.5,0))).

To plot $C(0.25,0.5,0)$, we use the parametric representation

```
[0.25·COS(t)+0.5,0.25·SIN(t)],
```

where $0 \le t \le 2\pi$. We can obtain the image of $C(0.25,0.5,0)$ under $f(z)$ using the user-defined *DERIVE* routine `Plane_Circ_Image(r,a,b,M)` which returns the image of the parametric representation $(r \cos t + a, \sin t + b)$ of the circle of radius r centered at the point (a, b) in the plane under the 2×2 matrix representation M of the Möbius transformation $f(z)$. Now the command

```
Plane_Circ_Image(0.25,0.5,0,M)
```

returns a parametric representation of $f(C(0.25,0.5,0))$:

```
Plane_Circ_Image(0.25,0.5,0,M) = [COS(t) + 2, SIN(t)],
```

where $0 \le t \le 2\pi$. We then use the command

`Plane_Circ_Image(1,2,0,M)` to find $f(f(C(0.25,0.5,0))) = f(C(1,2,0))$, which is

```
Plane_Circ_Image(1,2,0,M) = [4·COS(t) + 8, 4·SIN(t)],
```

where $0 \le t \le 2\pi$. Graphs of $C(0.25,0.5,0)$, $f(C(0.25,0.5,0)) = C(1,2,0)$, and $f(f(C(0.25,0.5,0))) = f(C(1,2,0)) = C(4,8,0)$ are shown in Figure 1.

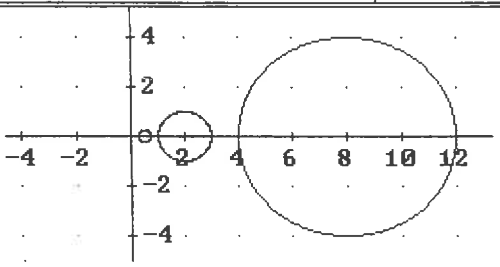
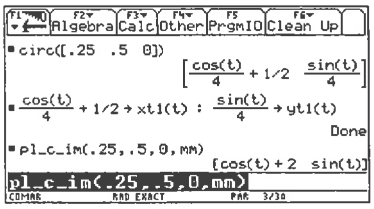
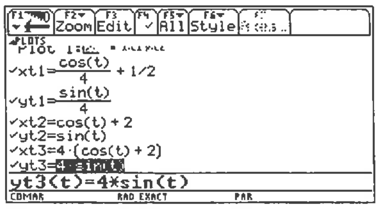


Figure 1:

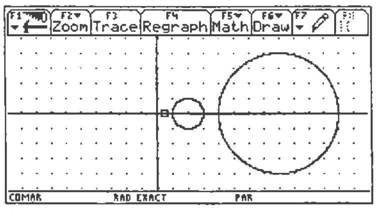
The circles $C(0.25,0.5,0)$, $C(1,2,0)$, and $C(4,8,0)$



```
F1 F2 Algebra F3 Calc F4 Other F5 PrgmIO F6 Clean Up
■circ([.25  .5  0])
                    [cos(t)/4 + 1/2   sin(t)/4]
■cos(t)/4 + 1/2 → xt1(t) : sin(t)/4 → yt1(t)
                                          Done
■pl_c_im(.25,.5,0,mm)
                          [cos(t) + 2  sin(t)]
pl_c_im(.25,.5,0,mm)
COMAR        RAD EXACT        PAR   3/30
```



```
F1 F2 Zoom F3 Edit F4 ✓ F5 All F6 Style
PLOTS
✓xt1=cos(t)/4 + 1/2
✓yt1=sin(t)/4
✓xt2=cos(t) + 2
✓yt2=sin(t)
✓xt3=4·(cos(t) + 2)
✓yt3=4·sin(t)
yt3(t)=4*sin(t)
COMAR        RAD EXACT        PAR
```



The TI-CAS-Realisation

Step 3. Now consider the image of the circle $C(r, a, b)$ with center (a, b) and radius r under f and draw a conclusion about the image of a circle under f.

The command `Plane_Circ_Image(r,a,b,M)` produces

`Plane_Circ_Image(r,a,b,M) = [4·r·COS(t) + 4·a,4·r·SIN(t) + 4·b],`

where $0 \le t \le 2\pi$. We can now conclude in general that $f(z) = 4z$ dilates a circle by a factor 4 and scales the center of the circle by a factor of 4.

We can draw further conclusions from this observation. First, note that f has an inverse, which is $f^{-1}(z) = z/4$. The nth iterate of f is $f^{(n)}(z) = 4^n z$, and the nth iterate of f^{-1} is $f^{(-n)}(z) = (f^{(n)})^{-1}(z) =$ $= (f^{-1})^{(n)}(z) = 4^{-n} z$. If R is the closed annulus in the plane bounded by $C(1,0,0)$ and $C(4,0,0)$ (see Figure 2), and int(R) is the interior of R, then $f^{(n)}(\text{int}(R)) \cap \text{int}(R) = \emptyset$, for each nonzero integer n. Moreover, $f^{(n)}(C(1,0,0)) \to \infty$ and $f^{(-n)}(C(1,0,0)) \to 0$ as $n \to \infty$The points 0 and ∞ in $\hat{\mathbf{C}}$ are the fixed points of f. More specially, 0 is called the *repelling fixed point* of f and ∞ is called the *attracting fixed point* of f. (By convention, let $f^{(0)}(z) = z$ be the identity transformation.) We now have $\cup_{n \in Z} f^n(R) = \mathbf{C} - \{0\}$. This means that R is a *fundamental region for the action of the infinite cyclic group* $\langle f \rangle$ *on* $\mathbf{C} - \{0\}$. Finally, note that the surface obtained by gluing the two boundary circles of R together is a torus.

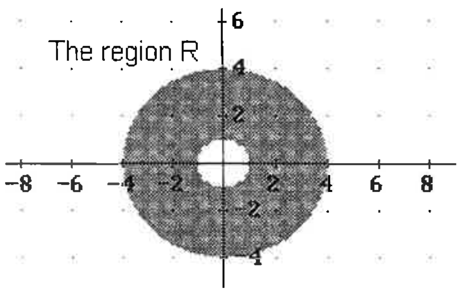


Figure 2:

The closed annulus R bounded by $C(1,0,0)$ and $C(4,0,0)$.

We now turn to the three-dimensional viewpoint. Recall that hyperbolic planes in the upper halfspace model of hyperbolic three-space $\mathbf{H}^3$ are either vertical planes in the upper halfspace or hemispheres orthogonal to the boundary plane at infinity ($\mathbf{C}$), which meet $\mathbf{C}$ along their (eqitorial) boundary circles. As mentioned earlier in this note, Möbius transformations can be extended from (conformal) Mappings of the plane to isometries of $\mathbf{H}^3$ by reflecting in the hyperbolic planes bounded by the generalized circles in $\hat{\mathbf{C}}$. To visualize this notion through our example, we continue with the next step in our activity.

Step 4. Plot the hyperbolic planes $HP(1,0,0)$ and $HP(2,0,0)$ that lie over the circles $C(1,0,0)$ and $C(2,0,0)$. Compute the product (composition) of the reflections in these two hyperbolic planes.

To plot the hyperbolic planes, we use the command `HP(r,a,b)` to plot the hyperbolic plane whose boundary at infinity is $C(r, a, b)$. Hence

```
HP(1,0,0)  =  [COS(s)SIN(t),SIN(s)SIN(t),COS(t)]
```
and
```
HP(2,0,0)  =  [2COS(s)SIN(t),2SIN(s)SIN(t),2COS(t)],
```
where $0 \le s \le 2\pi$ and $0 \le t \le \pi/2$.

The formula for the reflection in $HP(r, a, b)$ is

$$\phi_{(r,a,b)}(x,y,z) = (a,b,0) + \left(\frac{r^2}{(x-a)^2 + (y-b)^2 + z^2}\right)^2 (x-a, y-b, z).$$

We can verify (using *DERIVE*, for instance) that

$$\phi_{(2,0,0)}\left(\phi_{(1,0,0)}(x,y,z)\right) = (4x,4y,4z),$$

which is indeed the Poincaré extension $\tilde{f}$ of f to $\mathbf{H}^3$. As the transformation f is a dilatation by a factor of 4 away from the origin of points in the plane $\mathbf{C} = \mathbf{R}^2$, the extended transformation is an Euclidean dilatation by a factor 4 away from (0,0,0) of points in $\mathbf{H}^3$.

Step 5. Confirm the behavior of $\tilde{f}$ by computing $\tilde{f}$ (HP(1,0,0)).

To do this, we use the command `Poincare_Ext_ImageV(A,vec)`, which produces the image of the ordered triple `vec` under a 2×2 matrix representation of a Möbius transformation A. The matrix representation for $\tilde{f}$ is indeed the same matrix M representing f. Now

```
Poincare_Ext_ImageV(M,[COS(s)SIN(t),SIN(s)SIN(t),COS(t)])
```
gives
```
[4COS(s)SIN(t),4SIN(s)SIN(t),4COS(t)],
```
where 0 s 2π and 0 t $\pi/2$.

Arguing as before, we can see that the closed region D in $\mathbf{H}^3$ bounded by HP(1,0,0) and HP(4,0,0) is a fundamental region for the action of $\langle \tilde{f} \rangle$ on $\mathbf{H}^3$. (See Figure 3.)

Now, by gluing the boundaries HP(1,0,0) and HP(4,0,0) together, we obtain an open solid torus V. The boundary of V at infinity is called the *conformal boundary* of V and is just the torus constructed above! This boundary at infinity is called the conformal boundary because Möbius transformations, which behave as hyperbolic isometries, act as conformal maps on the boundary at infinity of $\mathbf{H}^3$, which can be identified as $\hat{\mathbf{C}}$.

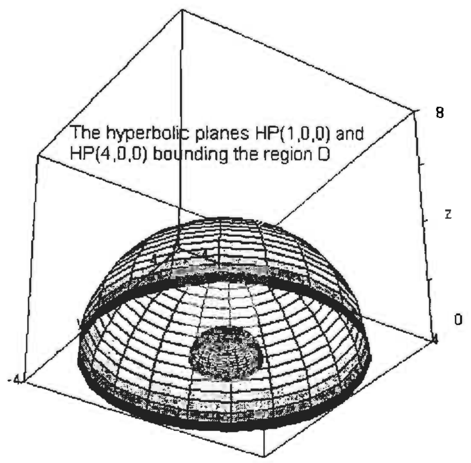


Figure 3

Recall that every nontrivial, non-parabolic Möbius transformation has a geodesic axis joining its two fixed points on $\hat{\mathbf{C}}$.

Let us denote by A the geodesic axis of $\tilde{f}$ joining 0 to ∞.

to be continued in DNL#52

Visualizing a Special Envelope in 3-D:
Sometimes you have to assist your mathematical assistant

Nurit Zehavi
Department of Science Teaching, The Weizmann Institute of Science
Rehovot 76100, ISRAEL
nurit.zehavi@weizmann.ac.il

In DNL#23 we visualized a special line in the 3-D space (Zehavi and Ben-Chaim, 1996). The special line, *[t, -2t – 1, t + 2]*, happened to be the line of intersection of all planes that when written in the standard form, $ax + by + cz = d$, the coefficients a, b, c, and d are consecutive terms of arithmetic sequences. At that time *Derive* version 3 did not allow superimposing surfaces. This limitation motivated us to develop an alternative method for visualizing a specific line in the 3-D space, namely plotting one *surface* obtained by applying the MIN (or MAX) function on a pair (or more) of explicit equations of planes that intersect in this line. In *Derive* version 5 it is easy to illustrate the span of planes for a two-parameter family of planes where the coefficients form an arithmetic progression, $ax + by + (2b - a)z = 3b - 2a$ as shown in figure 1.

```
VECTOR(VECTOR(a·x + b·y + (2·b - a)·z = 3·b - a, a, -5, 5, 3), b, -5, 5, 3)
```

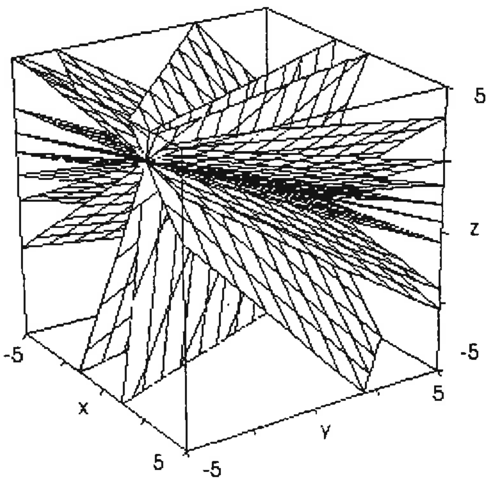
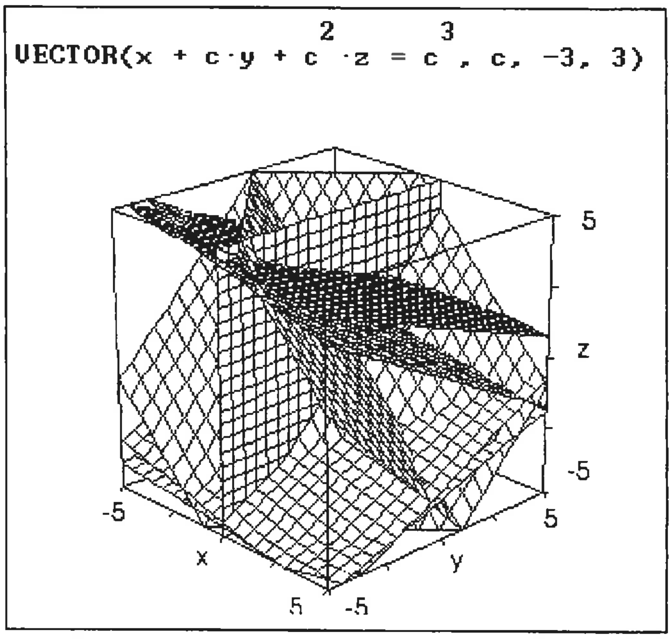


Figure 1. A family of planes intersecting in the special line

When mathematics teachers discover the special line obtained by "arithmetic" linear equations, they most frequently suggest exploring "geometric" equations, namely linear equations in which the coefficients form a geometric sequence (Sawada, 1993; Ben-Chaim, 1997). We start with the 2-D plane; for simplicity we explore a one-parameter "geometric" family of equations by $x + cy = c^2$. We can show and prove that the parabola $x + \frac{y^2}{4} = 0$ is the *envelope* of the family $x + cy = c^2$.

In general, for a given family of curves, a curve that is *tangent* at each of its points to a curve of the family is called the *envelope* of the given family of curves (Boltyanskii, 1964).

Can we see an envelope when we move on to the 3-dimensional family of planes? It is hard to even imagine the envelope in Figure 2.



$$\text{VECTOR}(x + c\cdot y + c^2\cdot z = c^3,\ c,\ -3,\ 3)$$

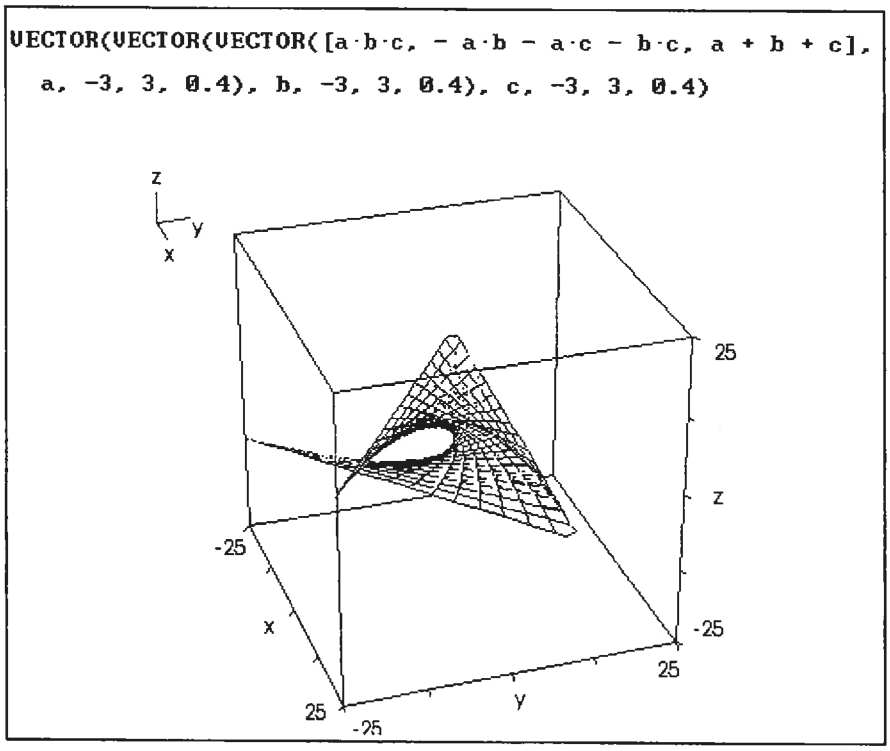
Figure 2. Visualizing a special family of planes

However, we can see three planes intersecting in one point. And indeed,

$$\text{SOLUTIONS}\left(\left[x + a\cdot y + a^2\cdot z = a^3,\ x + b\cdot y + b^2\cdot z = b^3,\ x + c\cdot y + c^2\cdot z = c^3\right],\ [x,\ y,\ z]\right)$$

$$[[a\cdot b\cdot c,\ - a\cdot b - a\cdot c - b\cdot c,\ a + b + c]]$$

Let us look for a visual pattern when we plot a long list of such intersection points (Figure 3):



```
VECTOR(VECTOR(VECTOR([a·b·c, - a·b - a·c - b·c, a + b + c],
  a, -3, 3, 0.4), b, -3, 3, 0.4), c, -3, 3, 0.4)
```

Figure 3. Connected intersection points of triplets of planes

It is worth trying to make conjectures on how the shape of the envelope of the planes should look before proceeding to the formal manipulations.

The method used in deferential geometry for finding the envelope for a given family (if it exists) is solving simultaneously the two equations $[f(x,y,z,c)=0, \frac{d}{dc}f(x,y,z,c)=0]$ and eliminating c.

$$f(x,y,z,c) := x + c\cdot y + c^2\cdot z - c^3$$

$$\left[f(x,y,z,c) = 0,\ \frac{d}{dc}f(x,y,z,c) = 0\right]$$

$$\left[x + c\cdot y + c^2\cdot z - c^3 = 0,\ y + 2\cdot c\cdot z - 3\cdot c^2 = 0\right]$$

Derive (version 5) does not have an *eliminate* command. We could use another CAS for that matter, or we could "help" *Derive* in the manipulations as demonstrated in Figure 4.

$$\text{SOLVE}\left(\left[x + c\cdot y + c^2\cdot z - c^3 = 0,\ y + 2\cdot c\cdot z - 3\cdot c^2 = 0\right],\ [x,\ y]\right)$$

$$\left[x = c^2\cdot(z - 2\cdot c)\ \wedge\ y = c\cdot(3\cdot c - 2\cdot z)\right]$$

To eliminate c we solve $y = c\cdot(3\cdot c - 2\cdot z)$ for c

$$\text{SOLVE}(y = c\cdot(3\cdot c - 2\cdot z),\ c)$$

$$c = \frac{z - \sqrt{(3\cdot y + z^2)}}{3}\ \vee\ c = \frac{\sqrt{(3\cdot y + z^2)} + z}{3}$$

Substitute the two solutions in x = c^2·(z − 2·c)

$$x - \left(\frac{z - \sqrt{(3\cdot y + z^2)}}{3}\right)^2\cdot\left(z - 2\cdot\frac{z - \sqrt{(3\cdot y + z^2)}}{3}\right) = 0$$

$$x - \left(\frac{\sqrt{(3\cdot y + z^2)} + z}{3}\right)^2\cdot\left(z - 2\cdot\frac{\sqrt{(3\cdot y + z^2)} + z}{3}\right) = 0$$

Denote the 'roots' by x1, x2

$$x1 := \left(\frac{z - \sqrt{(3\cdot y + z^2)}}{3}\right)^2\cdot\left(z - 2\cdot\frac{z - \sqrt{(3\cdot y + z^2)}}{3}\right)$$

$$x2 := \left(\frac{\sqrt{(3\cdot y + z^2)} + z}{3}\right)^2\cdot\left(z - 2\cdot\frac{\sqrt{(3\cdot y + z^2)} + z}{3}\right)$$

Expand the equation 27(x−x1)(x−x2)=0

$$27\cdot(x - x1)\cdot(x - x2) = 0$$

$$27\cdot x^2 + 18\cdot x\cdot y\cdot z + 4\cdot x\cdot z^3 - 4\cdot y^3 - y^2\cdot z^2 = 0$$

Figure 4. Finding the envelope

Now that we have an expression for the envelope, we want to visualize it; but *Derive* so far cannot plot an implicit expression in 3-D. This is the time to use *DPGraph* as suggested in DNL#46 (Wonisch, 2002). In Figure 5 we see the surface determined by the equation of the envelope. How is it related to Figure 3?

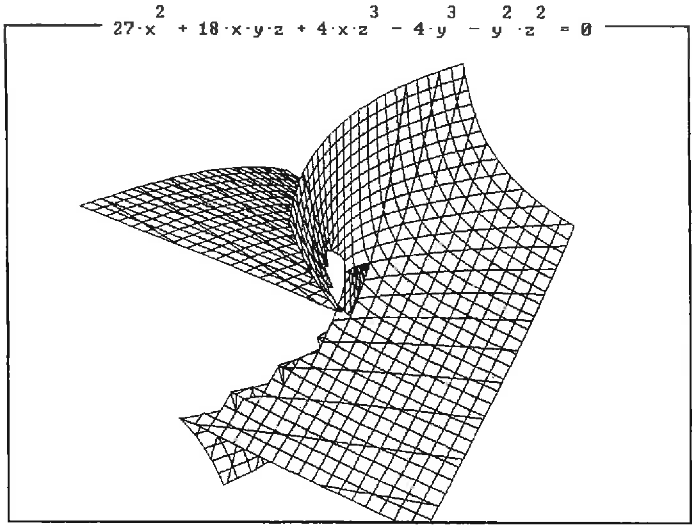

$27 \cdot x^2 + 18 \cdot x \cdot y \cdot z + 4 \cdot x \cdot z^3 - 4 \cdot y^3 - y^2 \cdot z^2 = 0$

Figure 5. Visualizing the envelope

We return to *Derive* to make sure that the equations of tangent planes to this surface belong to the "geometric" family of planes. For example:

$$\mathrm{TANGENT\_PLANE}\left(27 \cdot x^2 + 18 \cdot x \cdot y \cdot z + 4 \cdot x \cdot z^3 - 4 \cdot y^3 - y^2 \cdot z^2,\ [x, y, z],\ \left[\frac{5}{27}, 1, 1\right]\right)$$

$$27 \cdot x - 9 \cdot y + 3 \cdot z = -1$$

A nice graphic feature of *DPGraph* is the options to change parameters interactively and prepare animations. In Figure 6 we added the parametric equation $x + cy + c^2z = c^3$, and set $c = 0.33$ to fit the tangent plane above. In the scrollbar menu we marked the variable c, thus enabling changing the values of the parameter using the scrollbar. Finally we chose to color the graphs *BySteepness* (by the gradient) to create a didactic animation wherein the color of the animated plane is changing to match the color of the envelope where the animated tangent touches it.

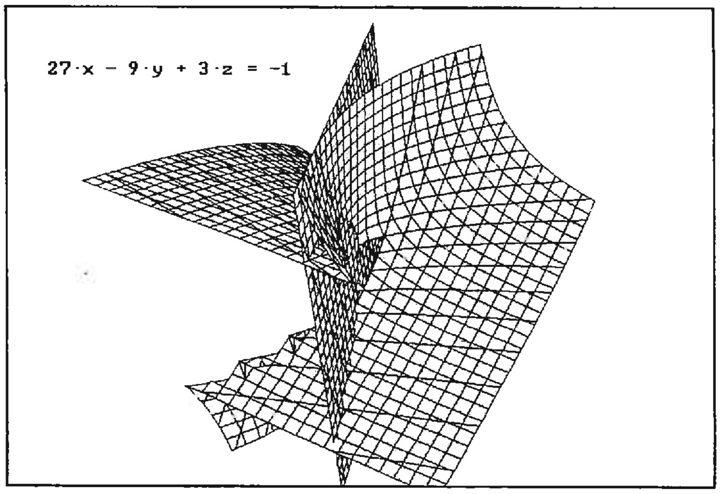


Figure 6. Animating tangent planes

An additional didactic animation can be designed by plotting in *DPGraph* three planes:

$[x + ay + a^2z = a^3, x + by + b^2z = b^3, x + cy + c^2z = c^3]$.

In figure 7 we set *a = 1.5, b = -1, c = 0.5*. Their intersection point is at [-0.75, 1.25, 1]. We can animate each plane and follow interactively the new intersection point.

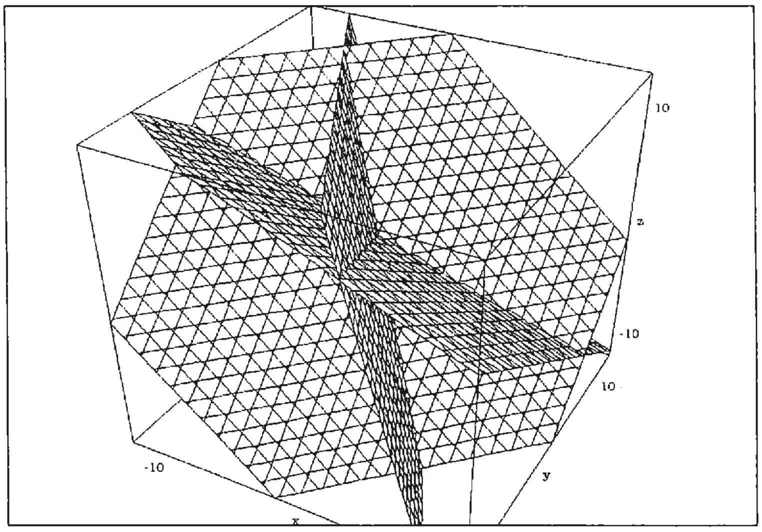


Figure 7. Animating three intersecting planes

References

Ben-Chaim, D (1997). A special point in the 2-D plane and a special line in the 3-D space. *Int. Journal of Mathematical Education in Science and in Technology* 28(30), 345-355.

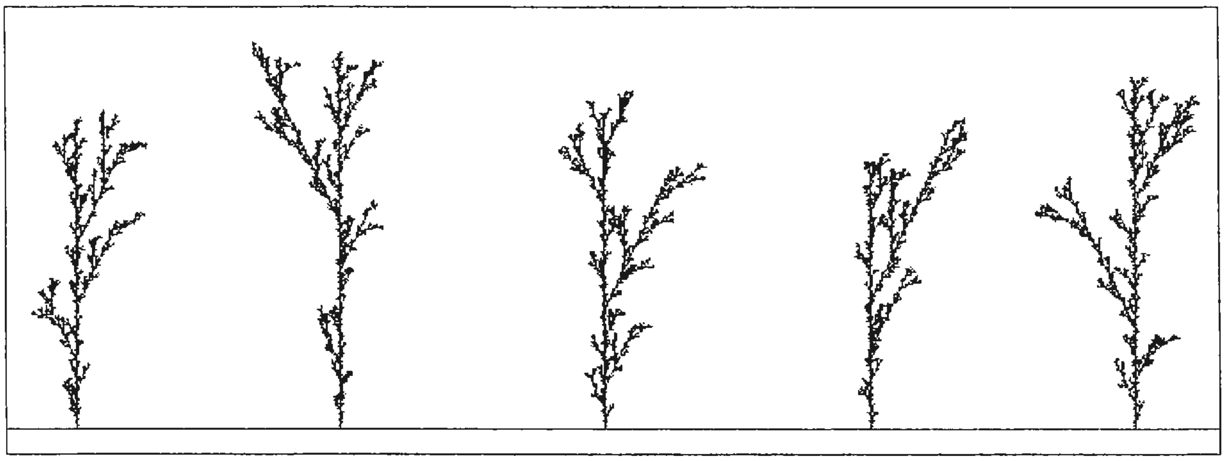
Boltyanskii, V. C. (1964). *Envelopes*. New York: The Macmillan Company.

Sawada, E. Y (1993). Extending algebra concepts with technology. *The Derive Newsletter* #12, 14-19.

Spivak, M. (1979). *A Comprehensive Introduction to Differential Geometry* (Vol 3), Berkeley: Publish or Perish..

Wonisch, R. (2002). DPGraph animates Derive. *The Derive Newsletter* #46, 8-16.

Zehavi, N. & Ben-Chaim, D. (1996). Visualizing a special line in the 3-D space. *The Derive Newsletter* #23, 7-9.



A meadow of "Stochastic Lindenmayer Plants"

DERIVE AND BOSON ALGEBRA

Francisco M. Fernández
Cequinor, Facultad de Ciencias Exactas, Universidad Nacional de La Plata
framfer@isis.unlp.edu.ar

Boson algebra is suitable for solving many problems in theoretical physics. It is spanned by the so-called creation $\hat{a}^{\dagger}$ and annihilation $\hat{a}$ operators that satisfy the commutation relation

$$[\hat{a},\hat{a}^{\dagger}]=\hat{a}\hat{a}^{\dagger}-\hat{a}^{\dagger}\hat{a}=\hat{1} \tag{1}$$

where $\hat{1}$ is the identity operator that we will omit from now on. In order to make this letter sufficiently self-contained we outline some of the well-known properties of the boson algebra.

There is a set of eigenvectors $\{|n>,\quad n=0,1,\ldots\}$ of the number operator $\hat{n}=\hat{a}^{\dagger}\hat{a}$ that satisfy

$$\hat{n}|n>=n|n>,\ \hat{a}|n>=\sqrt{n}|n-1>,\ \hat{a}^{\dagger}|n>=\sqrt{n+1}|n+1> \tag{2}$$

In standard bracket notation, the second and third equalities in equation (2) are equivalent to

$$<n|\hat{a}^{\dagger}=\sqrt{n}<n-1|,<n|\hat{a}=\sqrt{n+1}<n+1| \tag{3}$$

The set of vectors $\{|n>,\quad n=0,1,\ldots\}$ is orthonormal and complete; we write the scalar product between two of them in terms of the Kronecker delta function: $<m\,|\,n>=\delta_{m\,n}$.

Normal order greatly facilitates many calculations involving boson operators. Normal order means to write the creation operator to the left of the annihilation operator everywhere in a given expression. To achieve this goal one simply applies equation (1), rewritten as $\hat{a}\hat{a}^{\dagger}=1+\hat{a}^{\dagger}\hat{a}$, as many times as required.

In what follows we specialize in powers of the operator

$$\hat{s}=c_1\hat{a}+c_2\hat{a}^{\dagger} \tag{4}$$

For example, to write $\hat{s}^2$ in normal order we proceed as follows:

$$\begin{aligned}\hat{s}^2&=c_1^2\hat{a}^2+c_1c_2(\hat{a}\hat{a}^{+}+\hat{a}^{+}\hat{a})+c_2^2\hat{a}^{\dagger 2}=c_1^2\hat{a}^2+c_1c_2(1+2\hat{a}^{+}\hat{a})+c_2^2\hat{a}^{+2}\\&=c_1^2\hat{a}^2+c_1c_2(1+2\hat{n})+c_2^2\hat{a}^{\dagger 2}\end{aligned} \tag{5}$$

For higher powers $\hat{s}^k$ this straightforward procedure soon becomes tedious and error-prone as k increases.

It would be useful to have an algorithm to do such ordering automatically using Derive. Here we propose a recursive procedure based on the following expression:

$$\begin{aligned}\hat{s}^k&=\hat{s}\hat{s}^{k-1}=c_2\hat{a}^{\dagger}\hat{s}^{k-1}+c_1\hat{s}^{k-1}\hat{a}+c_1[\hat{a},\hat{s}^{k-1}]\\&=c_2\hat{a}^{\dagger}\hat{s}^{k-1}+c_1\hat{s}^{k-1}\hat{a}+c_1c_2(k-1)\hat{s}^{k-2}\end{aligned} \tag{6}$$

In the second equality of equation (6) we have used the result $[\hat{a},\hat{s}^{k-1}]=(k-1)\hat{s}^{k-2}$ that follows from induction on $[\hat{a},\hat{s}^{k-1}]=[\hat{a},\hat{s}]\hat{s}^{k-2}+\hat{s}[\hat{a},\hat{s}^{k-2}]$. Equation (6) tells us that if $\hat{s}^{k-1}$ and $\hat{s}^{k-2}$ are in normal order, then $\hat{s}^k$ will be in normal order, too. This conclusion enables us to write a Derive func tion to obtain $\hat{s}^k$ in normal order without bothering ourselves about the noncommutativity of the-boson operators. Therefore, we simply substitute variables a and b for the operators $\hat{a}$ and $\hat{a}^\dagger$, respectively.

The following simple Derive function does the desired calculation recursively:

```
   no(k, c1, c2) :=
      If k < 0
         0
#1:      If k = 0
            1
            c2·b·no(k - 1, c1, c2) + c1·no(k - 1, c1, c2)·a + c1·c2·
            (k - 1)·no(k - 2, c1, c2)
```

We set

```
#2: VariableOrder := [b, a]
```

in order to have the variable b that represents the creation operator to the left of a that represents the annihilation operator. For example

```
#3:   no(2)

        2  2                  2  2
#4:   c2 ·b  + 2·c1·c2·b·a + c1 ·a  + c1·c2
```

gives us the first line in equation (5). For the cubic power we have

```
#5:   no(3)

        3  3            2  2          2             2              2        3  3         2
#6:   c2 ·b  + 3·c1·c2 ·b ·a + 3·c1 ·c2·b·a  + 3·c1·c2 ·b + c1 ·a  + 3·c1 ·c2·a
```

after expansion in terms of the variables a and b.

Another interesting and useful application is the calculation of matrix elements $< m \mid \hat{s}^k \mid n >$. In order to obtain a suitable expression for Derive we write

$$
\begin{aligned}
< m \mid \hat{s}^k \mid n > &= < m \mid (c_1\hat{a} + c_2\hat{a}^\dagger)\hat{s}^{k-1} \mid n > \\
&= c_1\sqrt{m+1} < m+1 \mid \hat{s}^{k-1} \mid n > + c_2\sqrt{m} < m-1 \mid \hat{s}^{k-1} \mid n >
\end{aligned}
\tag{7}
$$

that follows from equations (3). The following function gives the desired matrix elements

```
    matel(m, k, n, c1, c2) :=
       If k = 0
#7:       KRONECKER(m, n)
          c1·√(m + 1)·matel(m + 1, k - 1, n, c1, c2) + c2·√m·matel(m - 1,
          k - 1, n, c1, c2)
```

It is interesting to see how Derive simplifies some expressions for arbitrary *m* and *n*:

```
#9:    matel(m, 1, n)

#10:   c1·√(m + 1)·IF(m = n - 1, 1, 0) + c2·√m·IF(m = n + 1, 1, 0)

#11:   matel(m, 2, n)

         2                                             2
#12:   c1 ·√(m + 1)·√(m + 2)·IF(m = n - 2, 1, 0) + c2 ·√m·√(m - 1)·IF(m = n + 2, 1, 0) +

         2·c1·c2·m·IF(m = n, 1, 0) + c1·c2·IF(m = n, 1, 0)
```

We get more compact expressions if we write n in terms of m; for example, for the first and second powers of $\hat{s}$ we have

```
#14:  matel(m, 1, m) = 0

#15:  matel(m, 1, m + 1) = c1·√(m + 1)

#16:  matel(m, 1, m + 2) = 0

#17:  matel(m, 2, m) = c1·c2·(2·m + 1)

#18:  matel(m, 2, m + 1) = 0

                              2
#19:  matel(m, 2, m + 2) = c1 ·√(m + 1)·√(m + 2)
```

The operators for the dimensionless coordinate and conjugate momentum in quantum mechanics are

$$\hat{q} = \frac{1}{\sqrt{2}}(\hat{a} + \hat{a}^{+}),\ \hat{p} = \frac{i}{\sqrt{2}}(\hat{a}^{+} - \hat{a}) \tag{8}$$

and the dimensionless Hamiltonian operator for the harmonic oscillator is

$$\hat{H} = \frac{1}{2}\left(\hat{p}^2 + \hat{q}^2\right) \tag{9}$$

If we use our Derive function to calculate matrix elements $< m \mid \hat{H} \mid n >$ for arbitrary m and n, we obtain

```
       1        ⎛          î    î ⎞   1        ⎛          1    1 ⎞
#20:  ───·matel ⎜m, 2, n, - ───, ───⎟ + ───·matel ⎜m, 2, n, ───, ───⎟
       2        ⎝          √2   √2 ⎠   2        ⎝          √2   √2 ⎠

      √(-m - 1)·√(-m - 2)·IF(m = n - 2, 1, 0)·SIGN((m + 1)·(m + 2))
#21:  ───────────────────────────────────────────────────────────── +
                                   4

        √(1 - m)·√(-m)·SIGN(m·(m - 1))·IF(m = n + 2, 1, 0)
        ────────────────────────────────────────────────── +
                               4

        √(m + 1)·√(m + 2)·IF(m = n - 2, 1, 0) + √m·√(m - 1)·IF(m = n + 2, 1, 0) + 2·(2·m~
        ─────────────────────────────────────────────────────────────────────────────────~
                                                  4                                      ~
        + 1)·IF(m = n, 1, 0)
        ────────────────────
```

In order to produce a simpler expression we substitute a variable i for the imaginary number and then substitute the latter back:

```
       1        ⎛          i    i ⎞   1        ⎛          1    1 ⎞
#22:  ───·matel ⎜m, 2, n, - ───, ───⎟ + ───·matel ⎜m, 2, n, ───, ───⎟
       2        ⎝          √2   √2 ⎠   2        ⎝          √2   √2 ⎠

        2                                                        2                                ~
      (i  + 1)·√(m + 1)·√(m + 2)·IF(m = n - 2, 1, 0) + √m·(i  + 1)·√(m - 1)·IF(m = n + 2~
#23:  ────────────────────────────────────────────────────────────────────────────────~
                                                        4                             ~

                        2
        , 1, 0) + (1 - i )·(2·m + 1)·IF(m = n, 1, 0)
        ────────────────────────────────────────────

                        (2·m + 1)·IF(m = n, 1, 0)
#24:                    ─────────────────────────
                                    2
```

This is the well-known result $< m \mid \hat{H} \mid n >= \frac{2m+1}{2}\delta_{mn}$.

Finally, I would like to mention another application of Derive to Boson Algebra. It is well-known that one can write

$$\hat{s}^k = \alpha_{k0}(\hat{n}) + \sum_{m=1}^{k}\left[\alpha_{km}(\hat{n})\hat{a}^{\dagger m} + \beta_{km}(\hat{n})\hat{a}^m\right]. \tag{10}$$

As before, we write $\hat{s}^k = \hat{s}^{k-1}\hat{s}$ and assume that $\hat{s}^{k-1}$ has been previously written in the form (10). Then, we take into account that the right multiplications $\hat{a}^j\hat{a}^\dagger = (\hat{n}+j)\hat{a}^{j-1}$ and $\hat{a}^{\dagger j}\hat{a} = (\hat{n}-j+1)\hat{a}^{\dagger j-1}$ have the following counterparts in terms of variables a and b: $(n+j)a^{j-1} = a^{-n}\frac{\partial}{\partial a}a^{j+n}$ and $(n-j+1)b^{j-1} = -b^{n+1}\frac{\partial}{\partial b}b^{j-n-1}$, respectively. The other two possible right multiplications are simpler: $\hat{a}^j\hat{a} = \hat{a}^{j+1}$, $\hat{a}^{\dagger j}\hat{a}^\dagger = \hat{a}^{\dagger j+1}$. If $P(\hat{a},\hat{a}^\dagger)$ is a polynomial of the form (10) and we substitute variables a and b for the operators $\hat{a}$ and $\hat{a}^\dagger$, respectively, then

$$\begin{aligned} P(\hat{a},\hat{a}^\dagger)\hat{a} &\to P(a,0)a - b^{n+1}\frac{\partial}{\partial b}\left[P(0,b)-P(0,0)\right]b^{-n-1} \\ P(\hat{a},\hat{a}^\dagger)\hat{a}^\dagger &\to P(0,b)b + a^{-n}\frac{\partial}{\partial a}\left[P(a,0)-P(0,0)\right]a^n \end{aligned} \tag{11}$$

should give us the correct coefficients of the polynomials resulting from the indicated operator multiplications. The Derive functions

```
                                 n + 1 d  ⎛    1
       ta(x) := a·SUBST(x, b, 0) - b     ·——  ⎜ ——————— ·(SUBST(x, a, 0) - SUBST(x, [a, b],
#25:                                     db  ⎜  n + 1
                                              ⎝ b

         [0, 0]))⎞
                 ⎠

                                    1  d    n
       tb(x) := b·SUBST(x, a, 0) + ——— ·—— (a ·(SUBST(x, b, 0) - SUBST(x, [a, b], [0, 0])))
#26:                                 n   da
                                    a
```

perform the operations in the right-hand sides of equations (11), and

```
         nno(j, c1, c2) :=
           If j = 0
              1
#27:          If j = 1
                 c1·a + c2·b
                 c1·ta(nno(j - 1, c1, c2)) + c2·tb(nno(j - 1, c1, c2))
```

applies them recursively to the powers of $\hat{s}$. For example, after expanding in terms of a and b this function gives us

```
                                  2  2       2  2
#28:   nno(2) = 2·c1·c2·n + c2 ·b  + c1 ·a  + c1·c2
```

which agrees exactly with the second line of equation (5). Analogously, for $\hat{s}^3$ we obtain

```
                3  3          2           2           2
#29:  nno(3) = c2 ·b  + 3·c1·c2 ·n·b + c1 ·a·(c1·a  + 3·c2·(n + 1))

#30:  VariableOrder := [b, a, b]
```

In order to have the correct order $\hat{a}$ and $\hat{a}^{\dagger}$ to the right of $\hat{n}$ we should expand the outputs in terms of the three variables a, b, and n. However, we have expanded them only in terms of the first two variables in order to show the polynomials of n explicitly.

We clearly appreciate that Derive is suitable for carrying out some simple but tedious operator algebra in quantum mechanics.

Albert Rich´s answer on a question raised in DNL#50

Hello Josef,
Thank you for the timely delivery of the DNL#50 as a pdf file. I used it to look up Benno Grabinger´s article on random number generators to understand the question you raised in TI_RANDOM.dfw. You wondered why

```
        VECTOR(RANDOM(1),i,1,5)
```

produced each second random number generated by

```
        VECTOR(RANDOM(2^32),i,1,5)/2^32.
```

If you reduce the precision from 10 to 6 digits, the first expression above will produce each of the random numbers produced by the second. This phenomenon is due to the fact that when producing random numbers between 0 and 1, the result must have at least the digits of presision number of random digits. Therefore, at 10 digits of precision, the random number generator is called twice in order to generate that many random digits. At 20 digits of precision, three calls are reqired.

Hope this explanation helps.

In the attachment you find an exercise that causes problems when computing with Derive 5.04.
Approximating the first expression to three digits of precision, you get three digits. So that's not the problem of course.

Approximating the second expression to three digits of precision, you get four digits. I don't like that!
Can you explain me what's happening ?
Approximating the second expression to two digits of precision, you get three digits. I could accept this result, because there are three digits before the comma in the result.

$$\left(\frac{365}{5+\sqrt{6}}\right)^{1/3} = 3.65$$

$$100\cdot\left(\frac{365}{5+\sqrt{6}}\right)^{1/3} = 365.9$$

$$100\cdot\left(\frac{365}{5+\sqrt{6}}\right)^{1/3} = 365$$

Hartelike groet
Marie-Jeann Aspeele

Albert Rich´s answer which gives another insight to illuminate *DERIVE*´s interior:

Internally Derive uses binary arithmetic. However users specify digits of precision in decimal digits. Therefore, Derive has to convert the number of decimal digits to the number of binary bits required to achieve the precision. Since this conversion from digits to bits of precision can only be an approximation, Derive conservatively rounds up the number of bits. This may result in slightly more digits of precision than requested.

In addition to the digits of precision, another factor to consider is the digits of notation used to display results. In decimal notation, the digits of notation is normally the maximum number of digits displayed. However, no matter how few digits of notation are requested, Derive displays all the integer digits of noninteger, rational numbers plus at least one digit following the decimal point. This is done to make clear that the number is not an integer.

Josef,

Here is my answer to your challenge. **Transform**() takes 3 arguements: s--the string to be transformed, tm--the transformation matrix, and n--the number of times the string is transformed. In this version, if a given element of the string is not to be transformed, it DOES NOT have to be included in the transformation matrix.

```
        Transform(s, tm, n := 1, l, new, i, j, k, m, a) :=
          Prog
            j := 1
            k := DIM(tm)
            Loop
              If j > n
                 RETURN s
              i := 1
              l := DIM(s)
              new := ""
              Loop
                If i > l exit
#1:             m := 1
                a := false
                Loop
                  If m > k exit
                  If s↓i = tm↓m↓1
                     [new := APPEND(new, tm↓m↓2), a := true]
                  m :+ 1
                If a = false
                   new := APPEND(new, s↓i)
                i :+ 1
              j :+ 1
              s := new
```

#2: $t := \begin{bmatrix} a & b \\ b & ac \\ c & abc \end{bmatrix}$

#3: Transform(ab, t, 5)

#4: babcacbabcbacabcacbacabcbabcacbacabcacbabcbacabc

#5: VECTOR([Transform(ab, t, n)], n, 5)

#6: $\begin{bmatrix} \text{bac} \\ \text{acbabc} \\ \text{babcacbacabc} \\ \text{acbacabcbabcacbabcbacabc} \\ \text{babcacbabcbacabcacbacabcbabcacbacabcacbabcbacabc} \end{bmatrix}$

#7: Transform(a_X_b, t, 5)

#8: babcacbabcbacabc_X_acbacabcbabcacbacabcacbabcbacabc

#9: VECTOR([Transform(a_X_b, t, n)], n, 5)

#10: $\begin{bmatrix} \text{b\_X\_ac} \\ \text{ac\_X\_babc} \\ \text{babc\_X\_acbacabc} \\ \text{acbacabc\_X\_babcacbabcbacabc} \\ \text{babcacbabcbacabc\_X\_acbacabcbabcacbacabcacbabcbacabc} \end{bmatrix}$

MacDonald Phillips

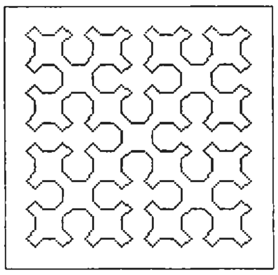
Hallo Josef,

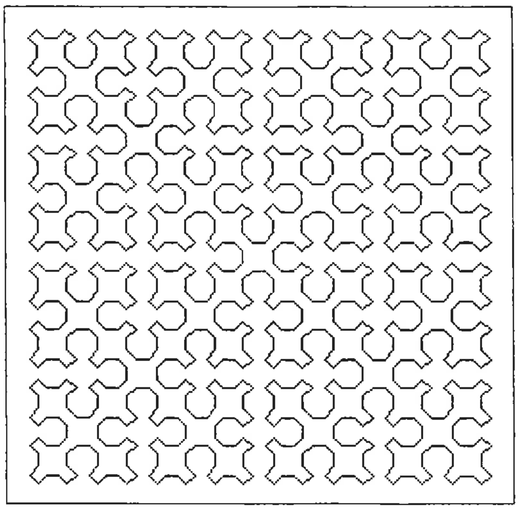
I programmed the curve which you presented in *DNL*#50 (page 39) in traditional way using the "Turtle": (The result is the classic Sierpinski Curve – see *Sagan, H.: Space-filling curves, 1994, p 51*).

Best regards

Rüdeger Baumann

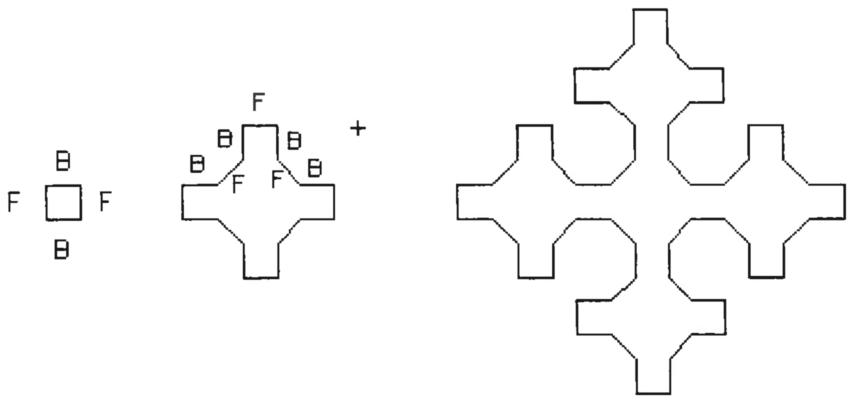
```
#1: Angle:=Degree

#2: [posX:=, posY:=, Richtung:=, Spur:=]

#3: neu:=PROG(posX:=0,posY:=0, Richtung:=0, Spur:=[[0, 0]])

    vor(s_):=PROG(
        posX := posX + s_·COS(Richtung),
#4:     posY := posY + s_·SIN(Richtung),
        Spur := APPEND(Spur, [[posX, posY]]))

#5: links(w_):=PROG(
        Richtung:=MOD(Richtung + w_, 360))

#6: rechts(w_):=links(-w_)

    Kolam(n, s):=
      IF(n > 0,
           PROG(
           Kolam(n - 1, s),
           rechts(45), vor(s), rechts(45),
#7:        Kolam(n - 1, s),
           links(90),  vor(s), links(90),
           Kolam(n - 1, s),
           rechts(45), vor(s), rechts(45),
           Kolam(n - 1, s)))

    Sierp(n, s, i:=0):=PROG(
                        neu,
                        LOOP(
                             i:=i + 1,
#8:                          IF(i > 4,RETURN Spur),
                             PROG(
                                  Kolam(n,s),
                                  rechts(45), vor(s), rechts(45))))

#9: Sierp(3, 1)
```





```
#10: Sierp(4,1)
```

You can download also the English version of the file.
Rüdeger wrote one special program for one special Curve. (See also "TURTLE GRAPHIC in DERIVE" by Lechner & Roanes & Roanes & Wiesenbauer, *DNL*#25, 1997).
Aristide Lindenmayer´s Rewriting technique makes possible to use one program to produce various forms of Curves.
See the following illustration: We start with the square (FBFB) and then replace the Bs by (BFBFBFB) and the Fs by an F. In this figure the Bs and the Fs are substituted in the same way .....

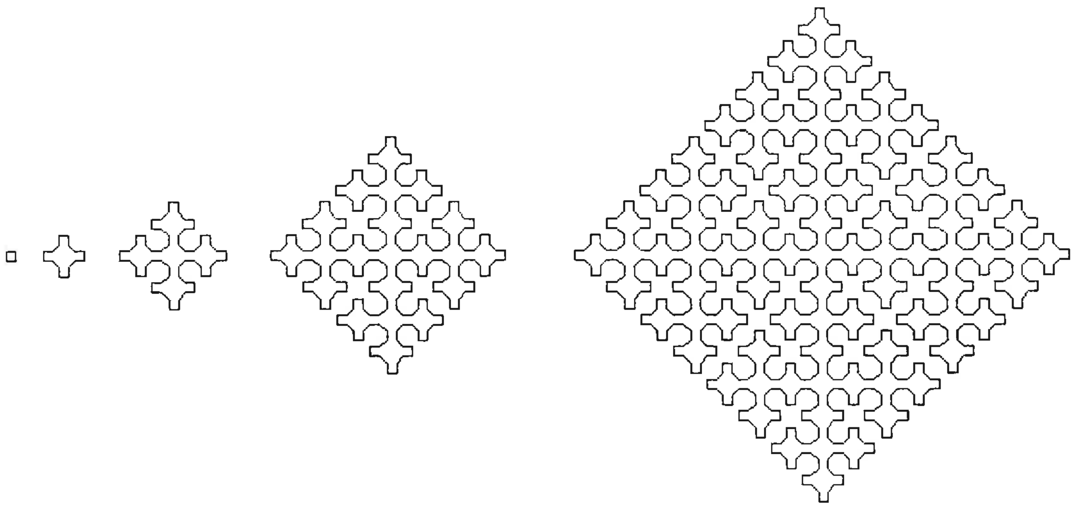


The rules are as follows (one important additional rule will be introduced in *DNL*#52):
B and **F**: go one step forward and plot the segment,
+ make a 45° turn in positive direction,
– make a 45° turn in negative direction,

```
lindsys(45,["B","F"],["B+F+B--F--B+F+B","F"],"B--F--B--F",4)
```

delivers the 4th stage of development of the "Snake Kolam". You will recognize the explanation from above. The syntax of the program is:

`lindsys (turning angle, list of elements to be substituted, substitution rules, starting figure, number of steps, [starting direction, default = 0], [starting point, default = [0,0]]).`



See DNL#52, Josef

Solution to the "New Challenge" from DNL # 50

Stefan Welke, Bonn, Germany, spwelke@aol.com

1 L-Systems

My solution is a straightforward exercise in functional programming. The main part of this exercise is to replace characters in a string by characters or strings, e.g. $"a" \rightarrow "b"$, $"b" \rightarrow "ac"$, and $"c" \rightarrow "abc"$ as given by the actual "new challenge". Thus we start with two functions, (1) to split a string into a vector of its characters, and (2) the inverse functions.

```
#1:   string2vector(str) := VECTOR(CODES_TO_NAME(k), k, NAME_TO_CODES(str))
      vector2string(vec) := CODES_TO_NAME(NAME_TO_CODES(vec)` )
#2:                                                          1
```

Examples:

```
#3:   string2vector("Lindenmayer")
#4:              ["L","i","n","d","e","n","m","a","y","e","r"]
#5:   vector2string(["L","i","n","d","e","n","m","a","y","e","r"])
#6:                          "Lindenmayer"
```

The core idea is to (I) replace the given initial string by its vector of characters, (II) then to substitute the characters according to the given rules, and (III) finally to transform the resulting vector of characters into a string:

$$"ab" \xrightarrow{\text{Step I}} ["a","b"] \xrightarrow{\text{Step II}} ["b",["a","c"]] \xrightarrow{\text{Step II'}} ["b","a","c"] \xrightarrow{\text{Step III}} "bac"$$

There is just one minor problem left, as is indicated in the diagram above: We get a vector, with characters and vectors as elements. So we have to remove the inner square brackets and this is exactly, what the following function does. The function **flatten(v,n)** flattens out nested vectors up to level n. This function proves to be useful in itself apart from our actual problem, the inspiration comes from *Mathematica*, where this is a built-in function.

```
#7:   flatten(v, n := 1) :=
        ITERATE(IF(VECTOR?(v_), PROG(w_ := VECTOR(IF(VECTOR?(k), k, [k]), k, v_),
                APPEND(w_)), v_, v_), v_, v, n)
#8:   flatten([3, [a, b, y ], [2, [t, s]]])
#9:                        [3, a, b, y , 2, [t, s]]
#10:  flatten([3, [a, b, y ], [2, [t, s]]], 2)
#11:                        [3, a, b, y , 2, t, s]
```

Now we are in the position to define the function **replacement(...)**, which combines the steps (II) and (II'):

```
#12:  replacement(expr_, old, new) := flatten(SUBST(expr_, old, new))
#13:  replacement(["a", a, 3], ["a", a, 3], [["a","b"], x, 7·y])
#14:                           ["a","b", x, 7·y]
```

The so called L-Systems are named after the biologist Aristid Lindenmayer. This is also the reason for the name of the following function, that combines steps (I) to (III):

```
      lindenmayer(str_, old_, new_, n := 1) :=
        Prog
#15:      w_ := string2vector(str_)
          ww_ := ITERATE(replacement(z_, old_, new_), z_, w_, n)
          vector2string(ww_)
```

The following examples are self-explanatory:

```
#16:  lindenmayer(ab, [a, b, c], [b, [a, c], [a, b, c]])
#17:                              bac
#18:  lindenmayer(ab, [a, b, c], [b, [a, c], [a, b, c]], 3)
#19:                          babcacbacabc
```

In order to produce a nested list, which represents the intermediate iterations too, we need a slight modification of the last function:

```
      lindenmayer_list(str_, old_, new_, n := 1) :=
        Prog
#20:      w_ := string2vector(str_)
          ww_ := ITERATES(replacement(z_, old_, new_), z_, w_, n)
          VECTOR(vector2string(k), k, ww_)
```

Examples:

```
#21:  lindenmayer_list(ab, [a, b, c], [b, [a, c], [a, b, c]], 3)
#22:                [ab, bac, acbabc, babcacbacabc]
#23:  [lindenmayer_list(ab, [a, b, c], [b, [a, c], [a, b, c]], 5)]`
             ┌                        ab                          ┐
             │                        bac                         │
             │                      acbabc                        │
#24:         │                   babcacbacabc                     │
             │              acbacabcbabcacbabcbacabc              │
             └ babcacbabcbacabcacbacabcbabcacbacabcacbabcbacabc   ┘
```

This is the solution to the "new challenge".

II 2D-Walks

It is fairly easy to create fractal figures with the aid of L-Systems that are iteratively created with an initial string and a set of replacement rules. As an example we use the set $\{1, i, -1, -i\}$ of complex numbers instead of the characters of the alphabet because we can interpret complex numbers as rotations. For example: The sequence $\{1, i, i, -i, -1, 1\}$ tells us to start at the origin, then to move one step forward (1), then turn left (i) and move a step forward, turn left again (i) and move a step forward, now turn right (-i) and move a step forward, then (-1) move a step back, and finally move a further step forward (1) in the actual direction. We can use the function `replacement(...)` to produce this sort of turtle graphics:

```
#25:  walk(start, old_, new_, n) := ITERATE(replacement(z_, old_, new_),
                                              z_, start, n)
#26:  walk([1, î], [1, î, -î], [î, [1, -î], [1, î, -î]], 5)
#27:  [î, 1, î, -î, 1, -î, î, 1, î, -î, î, 1, -î, 1, î, -î, 1, -î, î, 1,
        -î, 1, î, -î, î, 1, î, -î, 1, -î, î, 1, -î, 1, î, -î, 1, -î, î, 1,
        î, -î, î, 1, -î, 1, î, -î]
```

The next two functions are necessary to translate a sequence of complex numbers, the in-structions, into a set of vertices of the corresponding 2D-walk. The third one is the translation function:

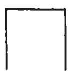
```
#28:  prod_list(v) := VECTOR(  k
                               Π   v , k, 1, DIMENSION(v))
                              j=1   j
#29:  sum_list(v) := VECTOR(  k
                              Σ   v , k, 1, DIMENSION(v))
                             j=1   j
#30:  two_d_walk(start_, old_, new_, n) := sum_list(prod_list(walk(start_,
                                                old_, new_, n)))
#31:  two_d_walk([1, î], [1, î, -î], [î, [1, -î], [1, î, -î]], 5)
#32:  [î, 2·î, -1 + 2·î, -1 + 3·î, -1 + 4·î, 4·î, 5·î, 6·î, -1 + 6·î, -1 +
        7·î, -2 + 7·î, -3 + 7·î, -3 + 8·î, -3 + 9·î, -4 + 9·î, -4 + 10·î,
        -4 + 11·î, -3 + 11·î, -3 + 12·î, -3 + 13·î, -2 + 13·î, -1 + 13·î,
        -1 + 14·î, 14·î, 15·î, 16·î, -1 + 16·î, -1 + 17·î, -1 + 18·î,
        18·î, 19·î, 20·î, 1 + 20·î, 2 + 20·î, 2 + 21·î, 3 + 21·î, 4 +
        21·î, 4 + 20·î, 5 + 20·î, 6 + 20·î, 6 + 21·î, 7 + 21·î, 7 + 22·î,
        7 + 23·î, 8 + 23·î, 9 + 23·î, 9 + 24·î, 10 + 24·î]
```

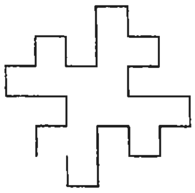
So far we have got a sequence of complex numbers that must be converted to ordinary points:

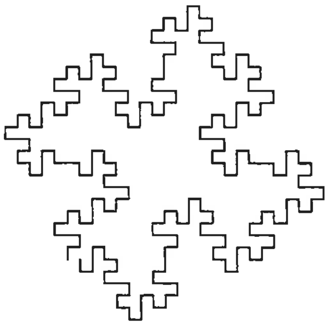
```
#33:  to_real_points(z_) := [RE(z_), IM(z_)]`
#34:  to_real_points([î, 2·î, -1 + 2·î, -1 + 3·î])
                                 [  0  1 ]
                                 [  0  2 ]
#35:                             [ -1  2 ]
                                 [ -1  3 ]
```

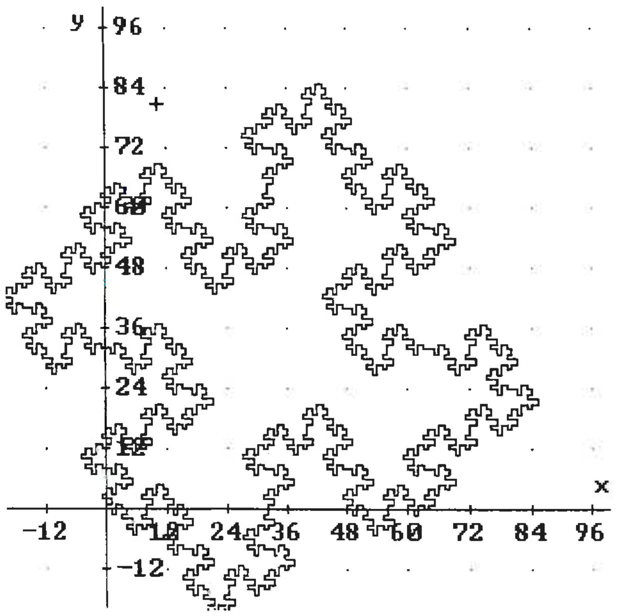
Let us close with a final example:

```
#36:  old:= [1, î, -î]
              [  1  -î  î  î  1  -î  -î  î ]
#37:  new:=   [  î  -î  î  î  1  -î  -î  î ]
              [ -î  -î  î  î  1  -î  -î  î ]
#38:  to_real_points(two_d_walk([1, î, î, î], old, new , 1))
#39:  to_real_points(two_d_walk([1, î, î, î], old, new , 2))
#40:  to_real_points(two_d_walk([1, î, î, î], old, new , 3))
```


Stage 0


Stage 1


Stage 2



Stage 3 (more than 220 sec)

Again I use my rewriting technique shortly described on page 29 together with the plotting routine to obtain a very similar figure. Calculation time differs enormously: Stefan's stage 3 plot takes nearly 4 minutes, my figure takes an instant to be plotted. Josef.

```
lindsys(90, ["F"], ["F+F-F-FF+F+F-F"], "F+F+F+F", 0, 0, [-12, 0])
lindsys(90, ["F"], ["F+F-F-FF+F+F-F"], "F+F+F+F", 1, 0, [-8, -2])
lindsys(90, ["F"], ["F+F-F-FF+F+F-F"], "F+F+F+F", 2, 0, [4, -8])
lindsys(90, ["F"], ["F+F-F-FF+F+F-F"], "F+F+F+F", 3, 0, [48, -32])
```