THE DERIVE - NEWSLETTER #50

THE BULLETIN OF THE DERIVE® USER GROUP + CAS-TI

Contents:

June 2003

Materialien - Materials

Als langjähriges Mitglied der DUG wende ich mich mit einer Bitte an Sie:
Ich habe auf meiner Homepage (Adresse siehe unten, Rubrik Mathematik) eine Reihe von Grundlagenartikeln zum Einsatz des TI-92/Voyage 200 sowie Unterrichtsbeispiele veröffentlicht, die sicherlich einige Fachkolleginnen und -kollegen interessieren dürften. Ich bitte Sie daher, im DNL auf diese Adresse hinzuweisen.

Vielen Dank!

Mit freundlichen Grüßen
Jürgen Wagner, Ahnatal
`http://j.wagner.bei.t-online.de`

Groebner Bases

Dear Derivers!
at the following link you can find a new utility file for Derive 5.06 (and other versions.) that allows computation of Groebner bases and other related functions:

`http://www.science.unitn.it/~perotti/groebner.htm`

Regards,
Alessandro Perotti

Announcement

Our good friend Carl Leinbach asked for publishing the following announcement and it is a pleasure to do this:

In January at the Joint Meetings in Phoenix, Arizona (Jan 7 - 10) Ed Connors and I will be co chairs of a Contributed Paper Session on "The effective use of Computer Algebra Systems in the Teaching of Mathematics." We have a very limited number of slots for 20 minute talks (15 minutes + 5 for questions). Also, could you mention the session in the DUG Newsletter? The session is on January 7 in the afternoon. Anyone interested in submitting a paper should send me their name, affiliation, e-mail address, the title, and a short (one or two paragraph) abstract. They can e-mail me this information. Ed Connors and I will make the decision by mid October, so we need the abstracts by mid September at the latest. We have only 9 - 12 slots, so the process should be competative.
(email: leinbach@gettysburg.edu)

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 (+ the "Moon"-file) from

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

Dear DUG Members,

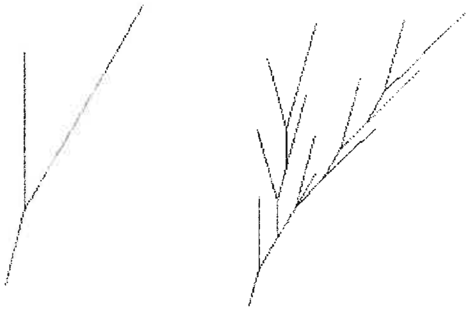
This is a very special issue of the *DNL*. Looking at the front page you can see, that you have *DNL*#50, the "Golden Issue", in your hands. I am sure that in our times this is a very rare event for a private initiative founded in 1991 to reach an age of life of more than 12 years, especially on a field of information technology. I´ll take the occasion to give my thanks to all of you who have helped making the *DUG* a lively group of not only CAS-interested people, but of friends all over the world.

We have a second celebration: Johann Wiesenbauer presents his "Silver" Titbits #25. On behalf of the *DUG* I´d like to express our gratefulness for sharing his rich experience and knowledge with us.

Inspecting the list of contents of this *DNL* I must add some comments:

The first "Challenge for Programmers" found five members who felt incited to face the challenge. Special appreciation to you: Terence Etchells from England, one (novice) colleague from Germany and a remarkable "Senior Trio" from Japan. The Japanese friends sent their findings by surface mail, because we had some problems with the email connection.

The new challenge (page 39) is an introduction to an article planned for *DNL*#51. I felt myself challenged to program *Lindenmayer-Systems.* (Aristide Lindenmayer invented a string rewriting system which can be used to abbreviate Turtle Commands for generating nature like (fractal) structures).



Karsten Schmidt submitted a contribution on a special matrix transformation. He promised to continue in a later issue with applications of the Moore-Pentrose-Inverse.

I am very happy to redeem an old debt presenting Georg Aue`s note on Linear Programming. This note is from DOS-*DERIVE* times but his "handicraft"-modelling is of everlasting validity.

Benno Grabinger sent an interesting contribution for all of us, who want to know how "randomly" the random numbers of our TI-CAS and *DERIVE* really are. With his permission, for what I am very grateful, I made some little amendments to include some *DERIVE*-links and -screen shots.

This *DNL* is again written very tight. I had to postpone an article which was the reaction on a strange request posed in the *DERIVE* Newsgroup: "Is it possible in *DERIVE* finding maximum and minimum points of a graph in a similar way, as it can be done on a graphing calculator (TI-83)?"

I found a way to simulate a GC on the 2D-Plot Window without using means of calculus (for the user – Calculus remains hidden in the background). So one can do the investigations of a graph even in secondary one level.

I am very proud of Peter Hofbauer´s application of the *DNL*-born *DERIVE* statistics tools. It shows a meaningful linkage between school and real-life mathematics. Many thanks, Peter.

In addition to the interesting announcements on the Information Page, I´d like to mention that René Hugleshofer will present the results of a special CAS-group (Forbes (UK), Herweyers (BEL), Hugelshofer (SUI), Schomacker (DK), Böhm (AUT) at a Conference in Reims. I´ll tell more about this project in fall – when we hopefully will have it finalized.

Finally I´d like to wish you all a wonderful summer – winter, of course for the southern hemisphere. Maybe that some of us will meet at any occasion at any meeting or conference or just in holidays.

With my best regards
Josef

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: September 2003
Deadline 15 August 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
2D- & 3D-Visualization of Moebius Transformations, T. Comar, USA
Boson Algebra with *DERIVE*, F. Fernandez, ARG
Lindenmayer-Systems, J. Böhm, 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

Matthieu Gouin woopee77@yahoo.com

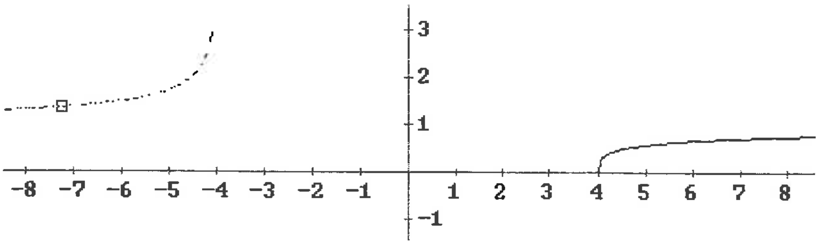
`y = (x - 4)^(1/4)/(x + 4)^(1/4)`

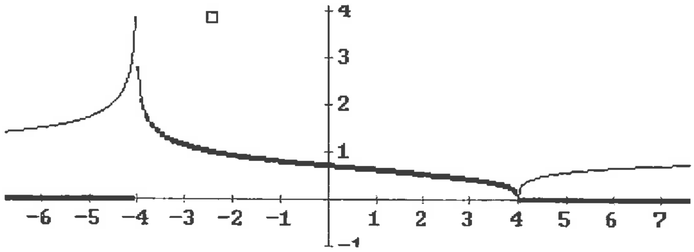
While tracing this equation in *DERIVE*, I notice that the left part of the graph (from -4 to the left) is dotted. Also, if I try to trace the graph, it says : "not real and finite".

However, if I substitute values in the expression, for example -5, it simplifies to y = sqrt(3) which is a real value.

Do you have an idea of this behavior ?

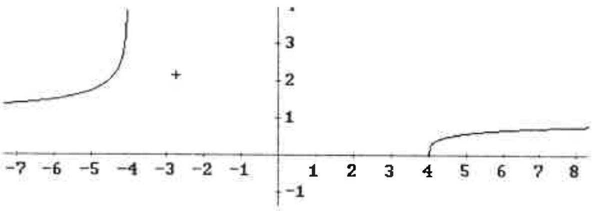
DNL: *I believe that the problem lies in graphing a quotient of complex numbers. First I set* `Options > Plot Real and Imaginary Parts` *in order to receive the graph as shown below. It is interesting to follow the path of the cursor box!!*

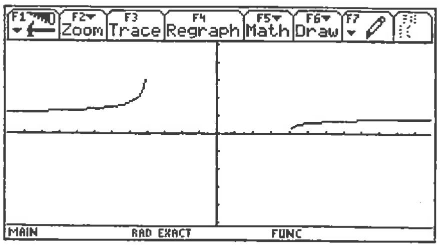




Then I rewrote the function and got the expected graph. In this case obviously DERIVE first evaluates the expression under the root which turns out to be positive for x < -4 or x > 4.
As you can see the TI has no problem to present the function as defined by Matthieu.

$$y = \left(\frac{x-4}{x+4}\right)^{1/4}$$





Ron Larham

It's nice to be able to report that Derive (v5.05 and v4.02) manage to do this weeks NewScientist Enigma without any additional analysis, while Mathematica managed to crash trying, and Maxima ran out of BigNum Space.

$$\text{MOD}(7777777^{7777777}, 100000) = 47697$$

DNL: *Message of MuPAD Pro 2.5:*

```
• modp(7777777^7777777,100000)
 Error: Overflow/underflow in arithmetical operation
```

Equations

Rick Nungester

DERIVE says:

$$\mathrm{SOLVE}(\mathrm{ATAN}(\mathrm{TAN}(x)) - x,\ x,\ \mathrm{Real}) = -\frac{\pi}{2} \le x < \frac{\pi}{2}$$

but $x = -\pi/2$ doesn´t satisfy the equation: $\mathrm{ATAN}\left(\mathrm{TAN}\left(-\frac{\pi}{2}\right)\right) = \pm\frac{\pi}{2}$.

Shouldn´t the real solution be $-\pi/2 < x < \pi/2$?

DERIVE says:

$$\mathrm{SOLVE}(\mathrm{LN}(x^2) - 2\cdot\mathrm{LN}(x),\ x,\ \mathrm{Real}) = x \ge 0$$

but $x = 0$ doesn´t satisfy the equation: $\left[\mathrm{LN}(0^2),\ \mathrm{LN}(0),\ \mathrm{LN}(0^2) - 2\cdot\mathrm{LN}(0)\right]$

$$[-\infty,\ \mathrm{LN}(0),\ ?]$$

Shouldn´t the real solution be $x > 0$?
Also, why does LN(0^2) simplify to -∞, but LN(0) doesn´t?

Albert Rich

Subtle and interesting points you raise.

1. Derive simplifies ATAN(TAN(x)) to

$$\mathrm{ATAN}(\mathrm{TAN}(x)) = x - \pi\cdot\mathrm{FLOOR}\left(\frac{x}{\pi} + \frac{1}{2}\right)$$

which is valid everywhere except for $x = k * \pi/2$, where k is odd. This is the reason the point $x = -\pi/2$ is included in the solution. However, if Derive did not make this transformation, it would be unable to provide any solutions to the equation.

2. The limit of LN(x^2) as x approaches 0 from the left or right is minus infinity. Whereas, LN(x) as x approaches 0 from the right is minus infinity, but the limit as x approaches 0 from the left is minus infinity + $\pi * i$. This is the reason LN(0) does not simplify further. However, when solving equations, finite imaginary parts added to real infinities are ignored. Thus the solution of LN(x^2)=2*LN(x) includes 0.

Because of anomalies like these that are difficult to remedy, I recommend always verifying the endpoints when Derive returns an interval as the solution to an equation or inequality.

Hope this helps.
Albert D. Rich

Rick Nungester

Thank you for your comments, but now I have bigger problems -- my faith in Derive is shaking... (:-)

"Valid everywhere except" says to me it isn't always true, and is a software defect. I've read other contributors refer to Derive's strict attention to domain issues and not simplifying unless something is ALWAYS true. The simplification above isn't always true, it is just usually true. Derive itself disagrees with it:

`ATAN(TAN(pi/2))` simplifies to `+/-pi/2` (correct).
`(pi/2)-pi*FLOOR((pi/2)/pi+1/2)` simplifies to `-pi/2` (correct).

These are different results.

This is the reason the point x=-π/2 is included in the solution. However, if Derive did not make this transformation, it would be unable to provide any solutions to the equation.

This sounds like providing useful usually-true results is sometimes, but not always, more important than strict mathematical truth. When Derive chooses one approach over the other is now unclear. Isn't there is a correct simplification of ATAN(TAN(*x*)) that gets the *x*=*k* ∗ π/2 (*k* odd) points correct also?

2. The limit of LN(x^2) as x approaches 0 from the left or right is minus infinity......

Thank you, this now makes sense to me.

Because of anomalies like these that are difficult to remedy, ...

I like Derive and use it often, but that last comment surprises me. Is it in the Derive Help or Reference Manual? It seems like a "disclaimer" without clear boundaries. What other circumstances are there I might expect "not exactly right" results?

Thanks again for your comments,
Rick Nungester

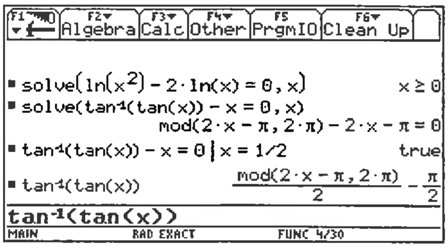
DNL(Josef)

I tried MuPAD solving both equations and didn´t receive any answer!
I also entered both equations into the Voyage 200. You can see its results below.

Then I tried – just for fun – another equation which one can easily solve manually finding that it is a contradiction. MuPAD doesn´t give any answer, but fortunately *DERIVE* and the TIs:

$$\mathrm{SOLVE}(\mathrm{LN}(x^2) - 2\cdot\mathrm{LN}(2\cdot x) = 0,\ x,\ \mathrm{Real}) = \mathrm{false}$$

give a correct answer.



```
■ solve(ln(x²) - 2·ln(x) = 0, x)                x ≥ 0
■ solve(tan⁻¹(tan(x)) - x = 0, x)
                      mod(2·x - π, 2·π) - 2·x - π = 0
■ tan⁻¹(tan(x)) - x = 0 | x = 1/2                true
                          mod(2·x - π, 2·π)     π
■ tan⁻¹(tan(x))           ─────────────────  -  ─
                                  2             2
tan⁻¹(tan(x))
MAIN            RAD EXACT          FUNC 4/30
```

A Limit from Sweden

David Sjöstrand

Hi,
One of my students asked me about this. *DERIVE* 5.06 simplifies the limit given below to the correct value, BUT approximates the same expression to the wrong value 0.

$$\lim_{x\to\pi/4} \frac{\mathrm{TAN}(x) - 1}{\mathrm{SIN}(x) - \dfrac{1}{\sqrt{2}}} = 2\cdot\sqrt{2} \qquad \mathrm{APPROX}\left(\lim_{x\to\pi/4} \frac{\mathrm{TAN}(x) - 1}{\mathrm{SIN}(x) - \dfrac{1}{\sqrt{2}}}\right) = 0$$

Can someone explain this?

Albert Rich

Hello David,

TAN(pi/4) approximates to 1, which is exact so the numerator of your example approximates to exactly 0. However, SIN(pi/4) approximates to 0.7071067811..., which is not exact so the denominator of your example approximates to a small number. The quotient of 0 and a number, no matter how small, approximates to 0.

This is an excellent example of the pitfalls of numerical approximations, and the advantages of exact mode.

Hope this helps.

Aloha,
Albert D. Rich, Co-author of Derive

Vladimir Bondarenko vvb@mail.strace.net

Good Day.

Agree! This example is really pretty.

Now let's have a look at Maple 8.01 command Limit which is an inert form of limit() and can be used for numeric evaluation of limits.

```
> Limit(sin(z)/z, z=0);
  Limit(sin(z)/z,z = 0)
> evalf(Limit(sin(z)/z, z=0));
  1.000000000
```

What about the approximation of the limit at hand?

```
> evalf(Limit((tan(z)-1)/(sin(z)-1/sqrt(2)), z=Pi/4));
  2.828427125
```

which coincides with the expected value.

Let's try to get more precision.

```
> evalf(Limit((tan(z)-1)/(sin(z)-1/sqrt(2)), z=Pi/4), 50);
  2.8284271247461900976033774484193961571393437507539
```

You can say that I used not an approximation to the original limit but got instead the output from a special numerical limit function. Okay, let it be so.

Thus, let's us consider

```
> evalf(limit((tan(z)-1)/(sin(z)-1/sqrt(2)), z=Pi/4), 50);
  2.8284271247461900976033774484193961571393437507538
```

which again is alright.

What about MuPAD 2.5.2 ?

```
> float(limit((tan(z)-1)/(sin(z)-1/sqrt(2)), z=PI/4));
  2.828427125
> DIGITS := 50:
> float(limit((tan(z)-1)/(sin(z)-1/sqrt(2)), z=PI/4));
  2.8284271247461900976033774484193961571393437507539
```

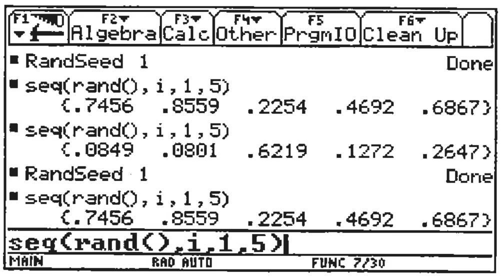
The User Forum will be continued on page 34

The TI-(& *Derive-*) Random Number Generator

Benno Grabinger, BennoGrabinger@t-online.de
Neustadt/Weinstraße, Germany

The random number generator (*RNG*) implemented into the *TI*-CAS machines represents a sophisticated coupling of two singular generators with period lengths $p_1 = 2^{31} - 86$ and $p_2 = 2^{31} - 250$. By this coupling one can achieve a period length with an order of magnitude $p = 2.3 \cdot 10^{18}$. The process is described detailed in [1]. Here we will demonstrate that the *TI*s are using this procedure. Among others it makes possible to predict the random numbers generated by the *TI*.

Figure 1 shows how to initialise the TI-*RNG* with "1" and how to produce subsequently twice a sequence of five random numbers in [0;1]. Figure 1 also shows, that after a new initialization using 1 again, one obtains the same "random numbers".



```
■ RandSeed 1                                 Done
■ seq(rand(), i, 1, 5)
       {.7456   .8559   .2254   .4692   .6867}
■ seq(rand(), i, 1, 5)
       {.0849   .0801   .6219   .1272   .2647}
■ RandSeed 1                                 Done
■ seq(rand(), i, 1, 5)
       {.7456   .8559   .2254   .4692   .6867}
seq(rand(),i,1,5)
MAIN           RAD AUTO            FUNC 7/30
```

Figure 1

I. Random Number Generators

A *RNG* is defined by a finite set of states *S*, a function $f: S \to S$ and an initial state s_0 (called *seed*).

The random numbers are produced by the iteration $s_i = f(s_{i-1})$, **$i = 1,2,3,\ldots..$**

Then a function $g: S \to\,]0;1[$ mapps state s_i on a number between 0 and 1 in order to obtain uniformly distributed random numbers in [0;1].

The concepts will be explained the following example.

Example 1
Linear congruence generator (*Lehmer*)

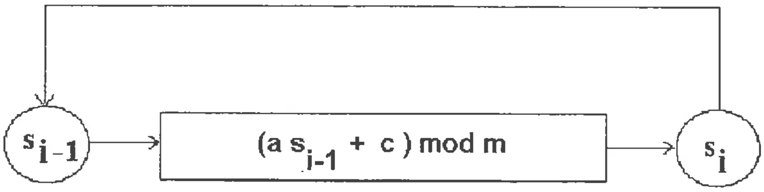
$$f(s) = (a \cdot s + c)\bmod m,\ \ 0 < a, c < m,\ S = \{0,1,2,\ldots..,m-1\}$$

$$g(s) = \frac{s}{m}$$

The sequence of random numbers is generated by an iteration as follows

$$s_i = (a \cdot s_{i-1} + c)\bmod m \quad \text{with} \quad 0 \le s_0 \quad \text{and} \quad a, c < m.$$

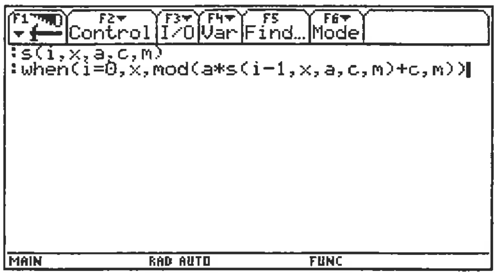
s_0 is an arbitrary initial number, which initialises the *RNG*. "mod" indicates that each new random number is the remainder of the integer division of $(a \cdot s_{i-1} + c)$ by *m*. The next diagram visualises the random number generating process.



The quality of the *RNG* depends on the choice of *a*, *c* and *m*. This will be illustrated by the next two examples.

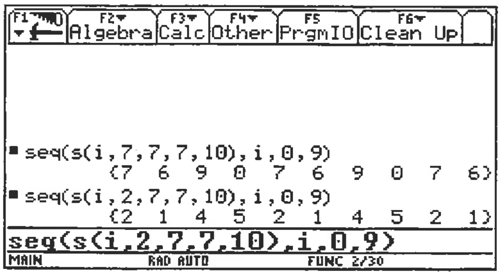
Example 1.1

Choosing $s_0 = a = c = 7$ and $m = 10$ we receive the sequence {7, 6, 9, 0, 7, 6, 9, 0, …..}, which is obviously no random sequence. Arrived at the fourth number we have finished a cycle. Starting with $s_0 = 2$ leads to another sequence with period length 4. (Figures 2 & 3).



```
:s(i,x,a,c,m)
:when(i=0,x,mod(a*s(i-1,x,a,c,m)+c,m))
```

Figure 2



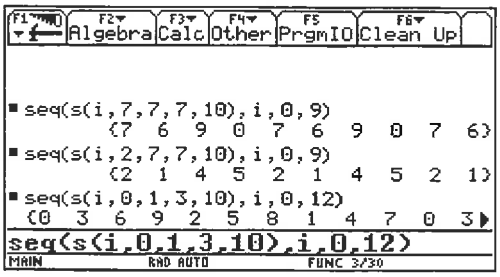
```
■ seq(s(i,7,7,7,10),i,0,9)
            {7  6  9  0  7  6  9  0  7  6}
■ seq(s(i,2,7,7,10),i,0,9)
            {2  1  4  5  2  1  4  5  2  1}
seq(s(i,2,7,7,10),i,0,9)
```

Figure 3

Example 1.2

The generator $s_i = (s_{i-1} + 3) \bmod 10$ with $s_0 = 0$ gives the sequence {0,3,6,9,2,5,8,1,4,7,0,3,…}. In a process forming remainders modulo 10 only the 10 remainders 0, 1, …, 9 can appear. So we can say that this *RNG* has maximum period length (Figure 4).

This can easily be done with *DERIVE,* too.



```
■ seq(s(i,7,7,7,10),i,0,9)
            {7  6  9  0  7  6  9  0  7  6}
■ seq(s(i,2,7,7,10),i,0,9)
            {2  1  4  5  2  1  4  5  2  1}
■ seq(s(i,0,1,3,10),i,0,12)
  {0  3  6  9  2  5  8  1  4  7  0  3▸
seq(s(i,0,1,3,10),i,0,12)
```

Figure 4

```
s(i, x, a, c, m) := (ITERATES(MOD(a·x_ + c, m), x_, x, i))
                                                          i
VECTOR(s(i, 7, 7, 7, 10), i, 10)
          VECTOR(s(i, 7, 7, 7, 10), i, 10) = [7, 6, 9, 0, 7, 6, 9, 0, 7, 6]
          VECTOR(s(i, 2, 7, 7, 10), i, 10) = [2, 1, 4, 5, 2, 1, 4, 5, 2, 1]
       VECTOR(s(i, 0, 1, 3, 10), i, 12) = [0, 3, 6, 9, 2, 5, 8, 1, 4, 7, 0, 3]
```

Comment

It can be shown [3] that the following conditions are necessary and sufficient for a maximum period length of a linear congruence generator.

- c and m are relatively prime.
- $a - 1$ is a multiple of p for each prime factor p of m.
- $a - 1$ is a multiple of 4 if m is a multiple of 4.

Example 1.3

The *RNG* implemented in *DERIVE* has all the properties of the comment from above and looks as follows:

$$s_i = (2654435721 \cdot s_{i-1} + 1) \bmod 2^{32}.$$

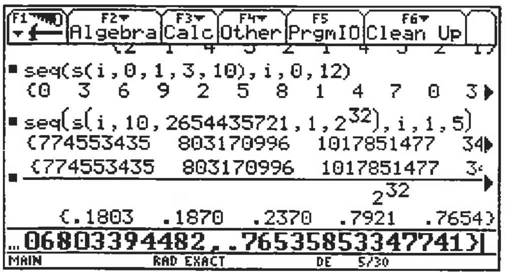
(More information can be found in [2], Chapter 40.)

We initialize this *RNG* using the PCU-time as seed and show the first 10 random numbers – without division by 2^{32}. Then we use the `s()` -function and we observe that the same numbers are generated.

```
                                 RANDOM(0) = 1292520975

              32
VECTOR(RANDOM(2  ), i, 10)

[547478792, 1691357513, 271827218, 3100759203, 1590744636, 2179531293, 32298630,
 3287163831, 128162800, 198370161]

                                               32
VECTOR(s(i, 1292520975, 2654435721, 1, 2  ), i, 2, 11)

[547478792, 1691357513, 271827218, 3100759203, 1590744636, 2179531293, 32298630,
 3287163831, 128162800, 198370161]
```

In Figure 5 we reproduce the *DERIVE* random numbers on the *TI* taking $x = 10$ as initial value for the procedure.

Attention in the *DERIVE* session. One has to "lay the same seed" s_0=10 for each experiment.



```
■ seq(s(i,0,1,3,10),i,0,12)
  {0  3  6  9  2  5  8  1  4  7  0  3▶
■ seq(s(i,10,2654435721,1,2^32),i,1,5)
  {774553435  803170996  1017851477  34▶
  {774553435  803170996  1017851477  3‹
■ ─────────────────────────────────────▶
                              2^32
     {.1803  .1870  .2370  .7921  .7654}
…06803394482,.765358533477 41}|
```

Figure 5

```
                                 RANDOM(-10) = 10
              32
VECTOR(RANDOM(2  ), i, 1, 5)
                  [774553435, 803170996, 1017851477, 3401906302, 3287189871]
                                 RANDOM(-10) = 10
              32
 VECTOR(RANDOM(2  ), i, 1, 5)
 ─────────────────────────────
              32
             2
                      [0.180339, 0.187002, 0.236987, 0.792068, 0.765358]
                                 RANDOM(-10) = 10
VECTOR(RANDOM(1), i, 1, 5)
                      [0.180339, 0.236987, 0.765358, 0.641051, 0.468350]
```

Comment: It is interesting that each second random number is used for RANDOM(1)??

Example 2
Multiplicative linear congruence generator (*MLCG*)

Setting $c = 0$, we create an *MLCG*, which has maximum period length $m - 1$ if m is prime and a is a primitive element modulo m [3].

II. Coupled Random Number Generators

Basics are the two following comments ([1]):

Comment 1

Let X_1 and X_2 two discrete independent random variables. Additionally it is assumed that X_1 is uniformly distributed on $\{0,1,2,\ldots,d-1\}$, with $d > 0$ and integer, i.e.

$$P(X_1 = n) = \frac{1}{d},\ n \in \{0,1,2,\ldots\ldots,d-1\}.$$

Then random variable $X = (X_1 + X_2) \bmod d$ is uniformly distributed on $\{0,1,2,\ldots, d-1\}$.

Proof

$X = n$ if $X_1 + X_2 = n + k \cdot d$, i.e. $P(X=n) = \sum_{k=0}^{\infty} P(X_1 + X_2 = n + k \cdot d)$.

Values which can be taken by X_2 are noted as $\{a, a+1, a+2, \ldots., b\}$. If $X_2 = j$, then X_1 must equal $(n - j) + k \cdot d$, that $X_1 + X_2 = n + k \cdot d$ becomes true. The fact that the domain for X_1 is $\{0,1,2,., d-1\}$. is equivalent to $X_1 = (n - i) \bmod d$.

Hence $P(X=n) = \sum_{j=a}^{b} P(X_2 = j) \cdot P(X_1 = (n-j) \bmod d) = \frac{1}{d} \sum_{j=a}^{b} P(X_2 = j) = \frac{1}{d} \cdot 1 = \frac{1}{d}$.

The next comment is very obvious and does not need special explanations:

Comment 2

Let f_1 and f_2 two generators with periods p_1 and p_2: $s_{1,i} = f_1(s_{1,i-1})$, $s_{2,i} = f_2(s_{2,i-1})$.

We inspect the sequence $\{s_i = (s_{1,i}, s_{2,i}) \mid i \geq 0\}$. This sequence has a period length $p = \operatorname{lcm}(p_1,p_2)$

Taking two *MLCG*s each of them with maximum period lengths

$$s_{1,i} = a_1 \cdot s_{1,i-1} \bmod m_1 \quad \text{and} \quad s_{2,i} = a_2 \cdot s_{2,i-1} \bmod m_2 \quad (m_1, m_2 \text{ relatively prime}).$$

Coupling these two generators results in a new generator z.

$$z_i = (s_{1,i} - s_{2,i}) \bmod (m_1 - 1).$$

According to comment 1 random variable Z is an uniformly distributed random number z_i on $\{0,1,\ldots.,(m_1-1)-1\}$ and following comment 2 the period of Z is $\operatorname{lcm}(m_1 - 1, m_2 - 1)$.

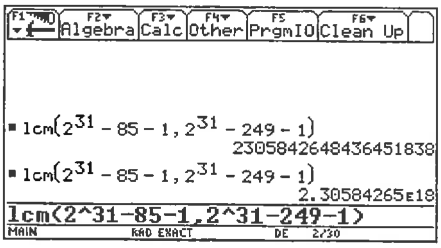
III. The TI-Random Number Generator

In the TI-*RNG* the two generators

$$s_{1,i} = (40014 s_{1,i-1}) \bmod (2^{31} - 85) \quad \text{and} \quad s_{2,i} = (40692 s_{1,i-1}) \bmod (2^{31} - 249)$$

are coupled to the generator $z_i = (s_{1,i} - s_{2,i}) \bmod (2^{31} - 85 - 1)$.

Both single generators have periods of an order of magnitude 2^{31}, the coupled generator has – according to comment 2 from above – a period length of $\operatorname{lcm}(2^{31} - 86, 2^{31} - 250) \approx 2.3 \cdot 10^{18}$! (for comparison: $2^{31} \approx 2.1 \cdot 10^9$. See Figure 6)



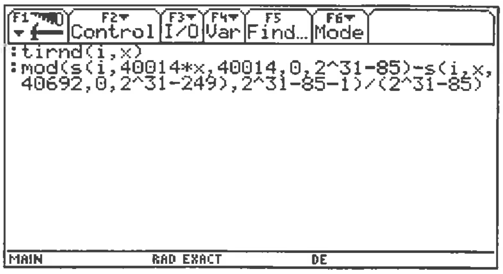
```
■ lcm(2^31 - 85 - 1, 2^31 - 249 - 1)
                    2305842648436451838
■ lcm(2^31 - 85 - 1, 2^31 - 249 - 1)
                         2.30584265E18
lcm(2^31-85-1,2^31-249-1)
MAIN          RAD EXACT          DE   2/30
```

Figure 6

We define a new function (coupling two special varieties of `s(i,x,a,c,m)` from above) to receive `tirnd(i,x)` with i = number of iteration (recursion) and x = initial value generating s_2. Generator s_1 is initialized with $40014x$.

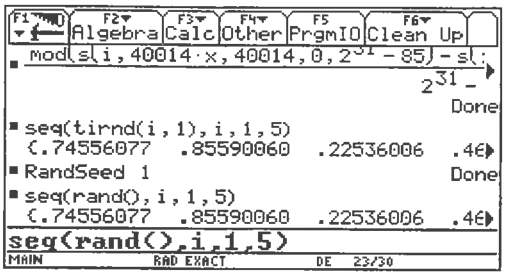
The coupled generator `tirnd(i,x)` can be obtained as described above (formula for z_i). The random numbers are divided by $2^{31} - 85$ in order to receive in [0,1] uniformly distributed random numbers.

```
mod(s(i,40014x,40014,0,2^31-85)-s(i,x,4092,0,2^31-249),2^31-85-1)/
   (2^31-85)→tirnd(i,x)
```



```
:tirnd(i,x)
:mod(s(i,40014*x,40014,0,2^31-85)-s(i,x,
 40692,0,2^31-249),2^31-85-1)/(2^31-85)
```

Figure 7



```
■ mod(s(i,40014·x,40014,0,2^31 - 85) - s(:
  ─────────────────────────────────────────▸
                                    2^31 -
                                      Done
■ seq(tirnd(i,1),i,1,5)
  {.74556077  .85590060  .22536006  .46▸
■ RandSeed 1                          Done
■ seq(rand(),i,1,5)
  {.74556077  .85590060  .22536006  .46▸
seq(rand(),i,1,5)
```

Figure 8

As you can see in Figure 8 we produce a sequence of five random numbers initialized by $x = 1$ and then we initialize the built-in random generator using "seed" = 1 and receive the predicted "random ?????" numbers.

So we have described how the CAS-*TI* built-in random number generator works.

Finally, we try as we did before with the *DERIVE*-random numbers on the TI. We implement the *TI-RNG* in *DERIVE*:

```
tirand(i, x) :=


                                         31                         31           31     ~
  MOD(s(i + 1, 40014·x, 40014, 0, 2   - 85) - s(i + 1, x, 40692, 0, 2   - 249), 2   - 8~
 ───────────────────────────────────────────────────────────────────────────────────────~
                                                  31                                     ~
                                                 2   - 85                                ~

 6)
 ──


VECTOR(tirand(i, 1), i, 10)
[0.745560, 0.855900, 0.225360, 0.469229, 0.686690, 0.0849241, 0.0800630, 0.621907,
  0.127215, 0.264651]
```

References:

[1] Pierre L'Ecuyer, Efficient and Portable Combined Random Number Generators, Communications of the ACM, June 1988, Volume 31, Number 6

[2] Benno Grabinger, Stochastik interaktiv mit DERIVE 5, bk-teachware SR-29, 2002

[3] D. E. Knuth, The Art of Computer Programming, Vol 2, 1981

Boson Algebra

Francisco M. Fernández

Dear Josef,

I am attaching a manuscript entitled "Derive and boson algebra" so that you consider its publication in Derive Newsletter.

Two of my undergraduate students who are presently taking a course on Physical Chemistry have agreed to do some special theoretical work based on Derive. They have no experience on programming, and one of them had never used a PC before she started. After this experiment I expect more students will be interested in CAS.

Best regards, Marcelo.

(Have you ever herad about BOSON Algebra? You will read about in the next DNL. (Josef)

An Introduction to the Moore-Penrose Inverse of a Matrix

Karsten Schmidt, Hannover, Germany, ks@karstenschmidt.de

Introduction

Associated with real numbers are 4 basic operations we all have been accustomed with since our early years in school: addition, subtraction, multiplication, and division. In matrix algebra you can perform similar operations for 3 of these 4: you can add or subtract two matrices ***A*** and ***B*** if they have the same dimension, i.e. the same number of rows and columns. You can also multiply two matrices, ***A*** and ***B***, provided the number of columns of ***A*** is equal to the number of rows of ***B***. This, by the way, provides an intuitive insight why, in general, $\boldsymbol{AB} \neq \boldsymbol{BA}$ since, depending on the dimensions of ***A*** and ***B***, ***AB*** might exist while ***BA*** might not.

Consider the following three matrices ***A***, ***B***, both having 2 rows and 2 columns, and ***C*** with 3 rows and 2 columns:

$$\underset{2\times2}{\boldsymbol{A}} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}; \quad \underset{2\times2}{\boldsymbol{B}} = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}; \quad \underset{3\times2}{\boldsymbol{C}} = \begin{pmatrix} 1 & 2 \\ 2 & 0 \\ 2 & 4 \end{pmatrix}$$

The calculation, for example, of

$$\boldsymbol{A}+\boldsymbol{B} = \begin{pmatrix} 2 & 4 \\ 5 & 8 \end{pmatrix}; \quad \boldsymbol{B}-\boldsymbol{A} = \begin{pmatrix} 0 & 0 \\ -1 & 0 \end{pmatrix}; \quad \boldsymbol{AB} = \begin{pmatrix} 5 & 10 \\ 11 & 22 \end{pmatrix}; \quad \boldsymbol{BA} = \begin{pmatrix} 7 & 10 \\ 14 & 20 \end{pmatrix}$$

is straightforward. In multiplication we get the element in row *i* and column *j* of the product ***AB*** by calculating $\sum_{k=1}^{n} a_{ik}b_{kj}$, where *n* denotes both the number of columns of ***A*** and the number of rows of ***B***. Since ***C*** is of a different dimension, this matrix cannot be added to or subtracted from ***A*** or ***B***, and the product ***BC***, for example, does not exist. The product ***CB***, on the other hand, does exist since the number of columns in ***C*** is equal to the number of rows in ***B***:

$$\underbrace{\underset{3\times2}{\boldsymbol{C}}\,\underset{2\times2}{\boldsymbol{B}}}_{3\times2} = \begin{pmatrix} 5 & 10 \\ 2 & 4 \\ 10 & 20 \end{pmatrix}$$

However, you cannot divide a matrix ***B*** by another matrix ***A***, regardless of their dimensions. Since dividing a real number *b* by another real number *a* (provided that $a \neq 0$) is equivalent to multiplying *b* by the reciprocal of *a*, denoted by a^{-1}, we could use the "reciprocal" of a matrix to circumvent the lack of division. There is indeed such a reciprocal of certain matrices, called the "inverse" and denoted by A^{-1}.

The unique inverse of a matrix ***A***, which satisfies the condition

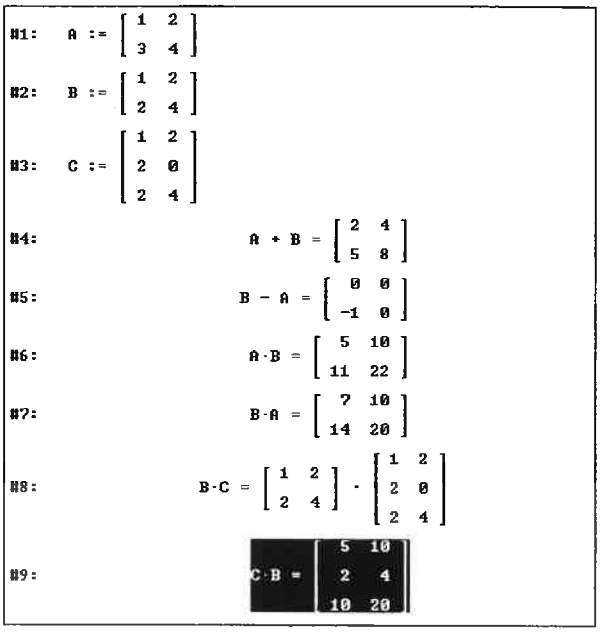
$$\underset{n\times n}{A^{-1}}\,\underset{n\times n}{A} = AA^{-1} = \underset{n\times n}{\boldsymbol{I}}$$

exists if and only if ***A*** is square and nonsingular (recall that $a^{-1}a = aa^{-1} = 1$). Here, $\underset{n\times n}{\boldsymbol{A}}$ denotes a square matrix ***A*** with *n* rows and *n* columns and ***I*** represents the identity matrix

$$\underset{n\times n}{I} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{pmatrix}$$

whose role in matrix algebra is similar to that of the "1" in the system of real numbers ($IA = AI = A$, for example). A matrix A is nonsingular if and only if its determinant is different from 0, i.e. A^{-1} exists if and only if $\det(A) \neq 0$. A square matrix that has a determinant of 0 is called a singular matrix. Recall that there is only one singular real number, 0.

Calculating the inverse of a matrix by hand is quite tedious, even more so than the product of two matrices. Algorithms for the calculation of A^{-1} can be found in any standard textbook on matrix algebra (cf. for example Schmidt & Trenkler (1998, pp. 38-40)). However, using *DERIVE* makes things easy when it comes to the calculation of matrix products, inverses, and determinants:



#1: $A := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

#2: $B := \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$

#3: $C := \begin{bmatrix} 1 & 2 \\ 2 & 0 \\ 2 & 4 \end{bmatrix}$

#4: $A + B = \begin{bmatrix} 2 & 4 \\ 5 & 8 \end{bmatrix}$

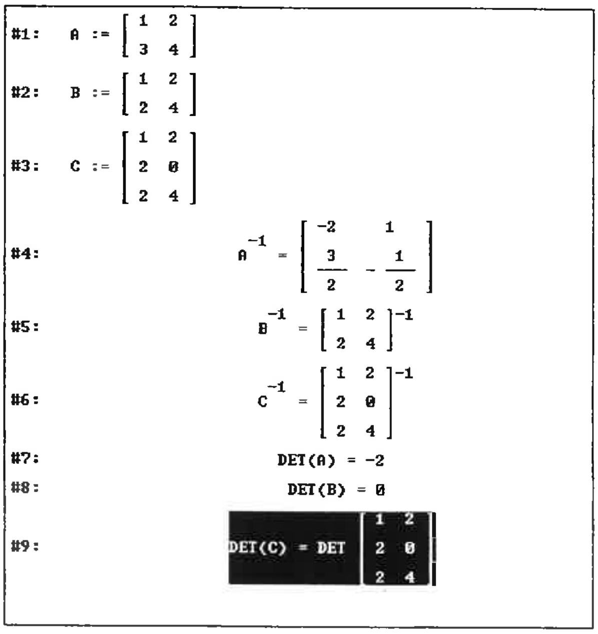
#5: $B - A = \begin{bmatrix} 0 & 0 \\ -1 & 0 \end{bmatrix}$

#6: $A \cdot B = \begin{bmatrix} 5 & 10 \\ 11 & 22 \end{bmatrix}$

#7: $B \cdot A = \begin{bmatrix} 7 & 10 \\ 14 & 20 \end{bmatrix}$

#8: $B \cdot C = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \cdot \begin{bmatrix} 1 & 2 \\ 2 & 0 \\ 2 & 4 \end{bmatrix}$

#9: $C \cdot B = \begin{bmatrix} 5 & 10 \\ 2 & 4 \\ 10 & 20 \end{bmatrix}$



#1: $A := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

#2: $B := \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}$

#3: $C := \begin{bmatrix} 1 & 2 \\ 2 & 0 \\ 2 & 4 \end{bmatrix}$

#4: $A^{-1} = \begin{bmatrix} -2 & 1 \\ \frac{3}{2} & -\frac{1}{2} \end{bmatrix}$

#5: $B^{-1} = \begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix}^{-1}$

#6: $C^{-1} = \begin{bmatrix} 1 & 2 \\ 2 & 0 \\ 2 & 4 \end{bmatrix}^{-1}$

#7: DET(A) = -2

#8: DET(B) = 0

#9: $\text{DET(C)} = \text{DET} \begin{bmatrix} 1 & 2 \\ 2 & 0 \\ 2 & 4 \end{bmatrix}$

From the left screenshot (dealing with addition, subtraction, and multiplication) it is clear that the product BC does not exist. *DERIVE's* way of telling us this is by refusing to simplify the requested product. Instead, it just substitutes the two matrices for the respective symbols and leaves it at that. From the right screenshot (dealing with inverses and determinants of matrices) it becomes clear that both B^{-1} and C^{-1} do not exist. With respect to B, which is a square matrix, we find that its determinant is 0, i.e. B is a singular matrix. C, on the other hand, is not even a square matrix; this also implies that you cannot compute $\det(C)$ since the determinant is only defined for square matrices.

Finally, there is another operation defined for matrices: transposition. The transpose of a matrix A, denoted by A', is found by interchanging rows and columns of A. This implies that if A has m rows and n columns, A' has n rows and m columns. Consider the transpose of our 3 examples:

$$A' = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}; \quad B' = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}; \quad C' = \begin{pmatrix} 1 & 2 & 2 \\ 2 & 0 & 4 \end{pmatrix}$$

If a matrix is equal to its transpose (as B is) it is called a symmetric matrix. Obviously, only square matrices can be symmetric.

Moore-Penrose Inverse

From the previous section we know that matrices which are not square, or are square but have a determinant equal to 0, do not have an inverse. However, for any matrix $\underset{m\times n}{A}$, if it is nonsingular, singular, or not even square, there exists a unique matrix with properties related to those of the inverse of a nonsingular matrix. This matrix is the Moore-Penrose inverse, denoted by $\underset{n\times m}{A^+}$, which satisfies the four conditions

$$AA^+A = A \tag{1}$$

$$A^+AA^+ = A^+ \tag{2}$$

$$(A^+A)' = A^+A \tag{3}$$

$$(AA^+)' = AA^+ \tag{4}$$

Conditions (3) and (4) require both A^+A and AA^+ to be symmetric matrices. Note that the dimension of A^+ is equal to the dimension of A'.

The concepts of the Moore-Penrose inverse and, more generally, the so-called generalized inverses go back to Moore (1920) and Penrose (1955). Greville (1960), Rao (1962), Rao & Mitra (1971) and Ben-Israel & Greville (1974) are some standard references related to generalized inverses and to the Moore-Penrose inverse. Note that generalized inverses are matrices which only satisfy condition (1). Therefore, in general the number of generalized inverses of a matrix is infinite.

Consider the special case of A being a nonsingular matrix. Then its inverse A^{-1} exists. Now substitute A^{-1} for A^+ in conditions (1) to (4). Since $A^{-1}A = AA^{-1} = I$, it is easy to verify that all 4 conditions are satisfied. That is, if A is a nonsingular matrix, we have $A^+ = A^{-1}$. (Moreover, in this case A^{-1} is the only generalized inverse.)

We now proceed to an algorithm for the computation of the Moore-Penrose inverse. We start with a fairly simple formula for the calculation of the Moore-Penrose inverse if $A = \underset{n\times 1}{a}$, i.e. if A is a (column) vector:

$$a^+ = \begin{cases} \frac{1}{a'a}a' & \text{if } a \neq 0 \\ 0' & \text{if } a = 0 \end{cases} \tag{5}$$

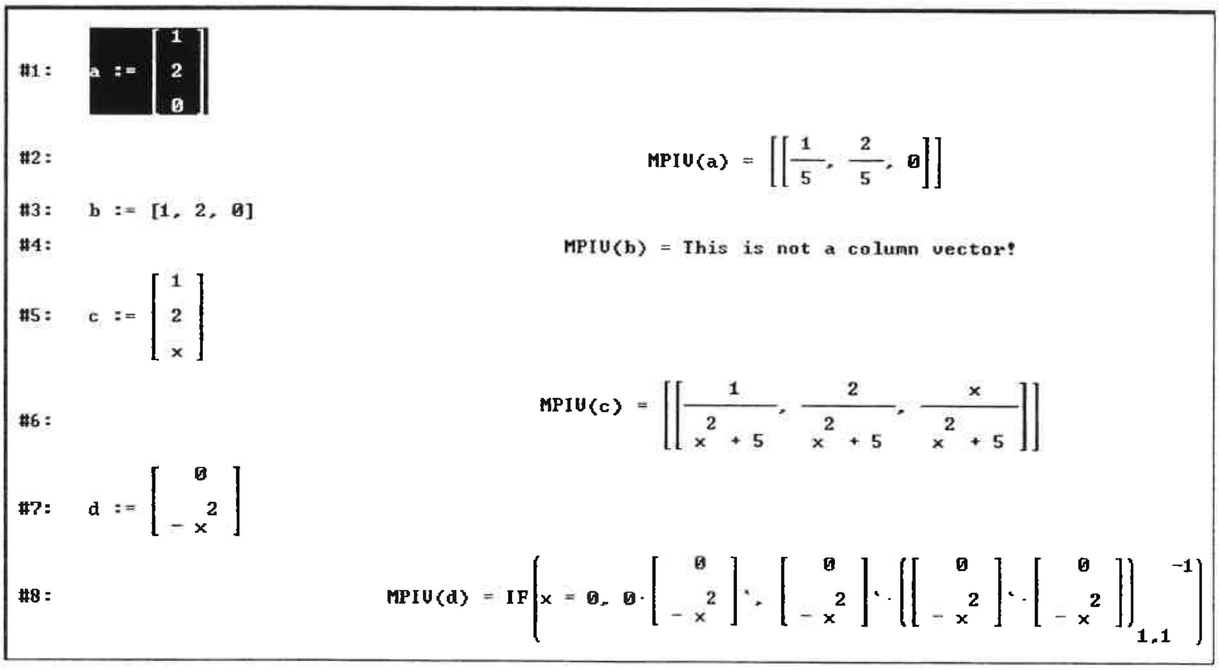
A vector is nothing else but a matrix with only one column, and should, therefore, be declared in *DERIVE* as such (i.e. do not use the symbol [■,■,■], but [⋮⋮⋮], and set the number of columns equal to one).

The function MPIV given below can be used to compute the Moore-Penrose inverse of a vector a. The function first checks if the actual parameter that has been passed on is indeed a (column) vector. If not, an error message is printed on the screen. If the parameter turns out to be a vector the function tests if a is a vector of zeros by computing $a'a$ and checking if this is equal to 0. If so, the Moore-Penrose inverse of $a = 0$ is simply $a^+ = 0'$. If $a'a$ is greater than 0, a^+ equals the transpose of a (a row vector) divided by $a'a$.

The reason `(a`·a)↓1↓1` is used instead of just `a`·a` is that while the product $\underbrace{\underset{1\times n}{\boldsymbol{a}'}\,\underset{n\times 1}{\boldsymbol{a}}}_{1\times 1}$ is in fact a real number (a "scalar"), DERIVE actually returns a matrix with one row and one column (easily to identify by the two pairs of brackets around the scalar). The `(  )↓1↓1` part "extracts" the real number from this matrix and so allows a comparison with another real number, and also a division of a' by this real number.

```
MPIV(a) :=
  If DIM(a`) = 1
     If (a`·a)↓1↓1 = 0
        0·a`
        a`/(a`·a)↓1↓1
     "This is not a column vector!"
```

Now consider the next screenshot:



#1: $a := \begin{bmatrix}1\\2\\0\end{bmatrix}$

#2: $\text{MPIV(a)} = \left[\left[\frac{1}{5}, \frac{2}{5}, 0\right]\right]$

#3: b := [1, 2, 0]

#4: MPIV(b) = This is not a column vector!

#5: $c := \begin{bmatrix}1\\2\\x\end{bmatrix}$

#6: $\text{MPIV(c)} = \left[\left[\frac{1}{x^2+5}, \frac{2}{x^2+5}, \frac{x}{x^2+5}\right]\right]$

#7: $d := \begin{bmatrix}0\\-x^2\end{bmatrix}$

#8: $\text{MPIV(d)} = \text{IF}\left(x = 0, 0\cdot\begin{bmatrix}0\\-x^2\end{bmatrix}^`, \begin{bmatrix}0\\-x^2\end{bmatrix}^`\cdot\left(\left[\begin{bmatrix}0\\-x^2\end{bmatrix}^`\cdot\begin{bmatrix}0\\-x^2\end{bmatrix}\right]_{1,1}\right)^{-1}\right)$

A vector $\boldsymbol{a}$ is declared (as a matrix with 3 rows and 1 column) and its Moore-Penrose inverse is computed with the `MPIV`-function. Let's do this manually according to (5):

$$\underset{3\times 1}{\boldsymbol{a}} = \begin{pmatrix}1\\2\\0\end{pmatrix};\quad \boldsymbol{a}'\boldsymbol{a} = \begin{pmatrix}1 & 2 & 0\end{pmatrix}\begin{pmatrix}1\\2\\0\end{pmatrix} = 1^2 + 2^2 + 0^2 = 5;\quad \boldsymbol{a}^+ = \tfrac{1}{\boldsymbol{a}'\boldsymbol{a}}\boldsymbol{a}' = \begin{pmatrix}\frac{1}{5} & \frac{2}{5} & 0\end{pmatrix}$$

What happens if you declare the same vector with the symbol [▪,▪,▪] is demonstrated in statements #3 and #4.

If a vector has symbolic elements it could be that the `MPIV`-function would not compute the Moore-Penrose inverse. Consider first vector $\boldsymbol{c}$; although it contains the symbol x, the `MPIV`-function finds the Moore-Penrose inverse since for any value of x we have $\boldsymbol{c} \neq \boldsymbol{0}$. Vector $\boldsymbol{d}$, on the other hand, would be a vector of zeros if the second element, $-x^2$, equaled 0. Therefore, the `MPIV`-function would not be able to compute $\boldsymbol{d}^+$ and thus *DERIVE* only simplifies `MPIV(d)` as far as possible.

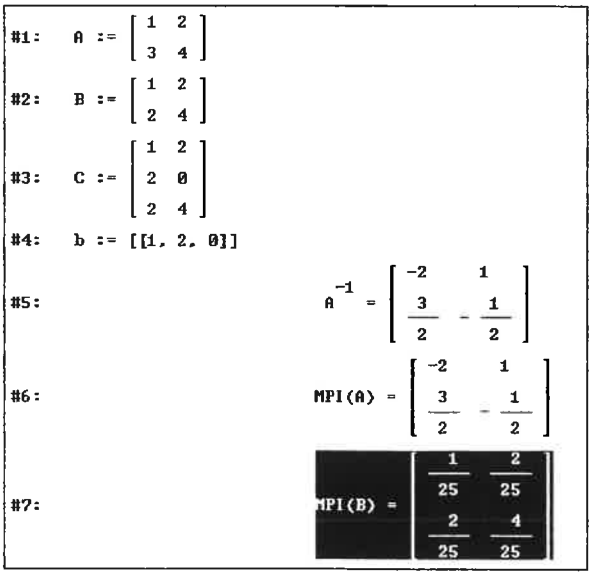
The function MPI given below for the computation of the Moore-Penrose inverse of any matrix is based on the Greville algorithm, which computes the Moore-Penrose inverse in a finite number of steps. A description of the Greville algorithm can be found in Schmidt & Trenkler (1998, pp. 115-116), for example; in Schmidt (1998) and Schmidt (2000) you can also find a description of this algorithm, along with earlier versions of the MPI-function.

The Greville algorithm starts by computing the Moore-Penrose inverse of the first column of A according to (5); therefore, the MPI-function starts by calling the MPIV-function with the first column of A. The result is the first row of A^+ (which is only an intermediate result).

```
MPI(A, APLUS, aj, dt, c, bt, J) :=
  Prog
    APLUS := MPIV(A COL [1])
    J := 2
    Loop
      If J > DIM(A`)
         RETURN APLUS
      aj := A COL [J]
      dt := aj`·APLUS`·APLUS
      c := (IDENTITY_MATRIX(DIM(A)) - A COL [1, ..., J - 1]·APLUS)·aj
      bt := MPIV(c) + (1 - MPIV(c)·c)/(1 + dt·aj)·dt
      APLUS := APPEND(APLUS - APLUS·aj·bt, bt)
      J :+ 1
```

The Greville algorithm and hence the MPI-function then proceed to the second column of A and compute the second intermediate A^+ by transforming the previous result and appending another row. This is repeated for all columns of A. After as many steps as the number of columns of A the Greville algorithm has found A^+. Note that in each step the MPI-function is called. Hence, the MPI-function might not be able to find A^+.

Consider the following two screenshots:



#1: $A := \begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$

#2: $B := \begin{bmatrix}1 & 2\\ 2 & 4\end{bmatrix}$

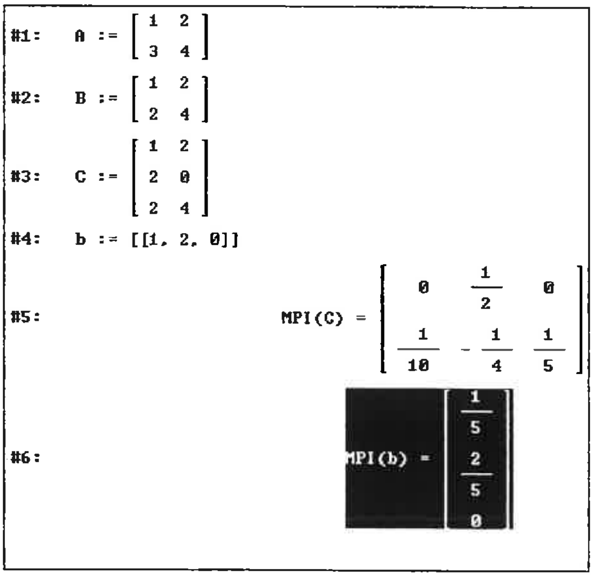
#3: $C := \begin{bmatrix}1 & 2\\ 2 & 0\\ 2 & 4\end{bmatrix}$

#4: b := [[1, 2, 0]]

#5: $A^{-1} = \begin{bmatrix}-2 & 1\\ \frac{3}{2} & -\frac{1}{2}\end{bmatrix}$

#6: $\mathrm{MPI}(A) = \begin{bmatrix}-2 & 1\\ \frac{3}{2} & -\frac{1}{2}\end{bmatrix}$

#7: $\mathrm{MPI}(B) = \begin{bmatrix}\frac{1}{25} & \frac{2}{25}\\ \frac{2}{25} & \frac{4}{25}\end{bmatrix}$



#1: $A := \begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$

#2: $B := \begin{bmatrix}1 & 2\\ 2 & 4\end{bmatrix}$

#3: $C := \begin{bmatrix}1 & 2\\ 2 & 0\\ 2 & 4\end{bmatrix}$

#4: b := [[1, 2, 0]]

#5: $\mathrm{MPI}(C) = \begin{bmatrix}0 & \frac{1}{2} & 0\\ \frac{1}{10} & -\frac{1}{4} & \frac{1}{5}\end{bmatrix}$

#6: $\mathrm{MPI}(b) = \begin{bmatrix}\frac{1}{5}\\ \frac{2}{5}\\ 0\end{bmatrix}$

The Moore-Penrose inverse has been computed for the 3 matrices given in the introduction and, in addition, for the row vector

$$\underset{1\times 3}{\boldsymbol{b}} = \begin{pmatrix} 1 & 2 & 0 \end{pmatrix}$$

Note that $\boldsymbol{b}$ must be declared as a matrix with 1 row and 3 columns, not as a vector.

For the nonsingular matrix $\boldsymbol{A}$ also the inverse has been computed which is identical to the Moore-Penrose inverse. To find out if the matrix given in statement #7 is indeed the Moore-Penrose inverse of $\boldsymbol{B}$ we check conditions (1) to (4):

$$(1) \qquad \boldsymbol{BB^+B} = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}\begin{pmatrix} \frac{1}{25} & \frac{2}{25} \\ \frac{2}{25} & \frac{4}{25} \end{pmatrix}\begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}\underbrace{\begin{pmatrix} \frac{1}{5} & \frac{2}{5} \\ \frac{2}{5} & \frac{4}{5} \end{pmatrix}}_{\boldsymbol{B^+B}} = \begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix} = \boldsymbol{B}$$

$$(2) \qquad \boldsymbol{B^+BB^+} = \begin{pmatrix} \frac{1}{25} & \frac{2}{25} \\ \frac{2}{25} & \frac{4}{25} \end{pmatrix}\begin{pmatrix} 1 & 2 \\ 2 & 4 \end{pmatrix}\begin{pmatrix} \frac{1}{25} & \frac{2}{25} \\ \frac{2}{25} & \frac{4}{25} \end{pmatrix} = \begin{pmatrix} \frac{1}{25} & \frac{2}{25} \\ \frac{2}{25} & \frac{4}{25} \end{pmatrix}\underbrace{\begin{pmatrix} \frac{1}{5} & \frac{2}{5} \\ \frac{2}{5} & \frac{4}{5} \end{pmatrix}}_{\boldsymbol{BB^+}} = \begin{pmatrix} \frac{1}{25} & \frac{2}{25} \\ \frac{2}{25} & \frac{4}{25} \end{pmatrix} = \boldsymbol{B^+}$$

Conditions (3) and (4) are also satisfied since $\boldsymbol{B^+B}$ and $\boldsymbol{BB^+}$ (both indicated in the above equations) are apparently symmetric matrices.

Let's do the same check for $\boldsymbol{C}$:

$$(1) \qquad \boldsymbol{CC^+C} = \begin{pmatrix} 1 & 2 \\ 2 & 0 \\ 2 & 4 \end{pmatrix}\begin{pmatrix} 0 & \frac{1}{2} & 0 \\ \frac{1}{10} & -\frac{1}{4} & \frac{1}{5} \end{pmatrix}\begin{pmatrix} 1 & 2 \\ 2 & 0 \\ 2 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 2 & 0 \\ 2 & 4 \end{pmatrix}\underbrace{\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}}_{\boldsymbol{C^+C}} = \begin{pmatrix} 1 & 2 \\ 2 & 0 \\ 2 & 4 \end{pmatrix} = \boldsymbol{C}$$

$$(2) \qquad \boldsymbol{C^+CC^+} = \begin{pmatrix} 0 & \frac{1}{2} & 0 \\ \frac{1}{10} & -\frac{1}{4} & \frac{1}{5} \end{pmatrix}\begin{pmatrix} 1 & 2 \\ 2 & 0 \\ 2 & 4 \end{pmatrix}\begin{pmatrix} 0 & \frac{1}{2} & 0 \\ \frac{1}{10} & -\frac{1}{4} & \frac{1}{5} \end{pmatrix} = \begin{pmatrix} 0 & \frac{1}{2} & 0 \\ \frac{1}{10} & -\frac{1}{4} & \frac{1}{5} \end{pmatrix}\underbrace{\begin{pmatrix} \frac{1}{5} & 0 & \frac{2}{5} \\ 0 & 1 & 0 \\ \frac{2}{5} & 0 & \frac{4}{5} \end{pmatrix}}_{\boldsymbol{CC^+}} = \begin{pmatrix} 0 & \frac{1}{2} & 0 \\ \frac{1}{10} & -\frac{1}{4} & \frac{1}{5} \end{pmatrix} = \boldsymbol{C^+}$$

Again, conditions (3) and (4) are also satisfied since $\boldsymbol{C^+C}$ and $\boldsymbol{CC^+}$ (both indicated in the above equations) are apparently symmetric matrices.

Finally, let's check if `MPI(b)` gives indeed the Moore-Penrose inverse of $\boldsymbol{b}$:

$$(1) \qquad \boldsymbol{bb^+b} = \begin{pmatrix} 1 & 2 & 0 \end{pmatrix}\begin{pmatrix} \frac{1}{5} \\ \frac{2}{5} \\ 0 \end{pmatrix}\begin{pmatrix} 1 & 2 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 2 & 0 \end{pmatrix}\underbrace{\begin{pmatrix} \frac{1}{5} & \frac{2}{5} & 0 \\ \frac{2}{5} & \frac{4}{5} & 0 \\ 0 & 0 & 0 \end{pmatrix}}_{\boldsymbol{b^+b}} = \begin{pmatrix} 1 & 2 & 0 \end{pmatrix} = \boldsymbol{b}$$

$$(2) \qquad \boldsymbol{b^+bb^+} = \begin{pmatrix} \frac{1}{5} \\ \frac{2}{5} \\ 0 \end{pmatrix}\begin{pmatrix} 1 & 2 & 0 \end{pmatrix}\begin{pmatrix} \frac{1}{5} \\ \frac{2}{5} \\ 0 \end{pmatrix} = \begin{pmatrix} \frac{1}{5} \\ \frac{2}{5} \\ 0 \end{pmatrix}\underbrace{\begin{pmatrix} 1 \end{pmatrix}}_{\boldsymbol{bb^+}} = \begin{pmatrix} \frac{1}{5} \\ \frac{2}{5} \\ 0 \end{pmatrix} = \boldsymbol{b^+}$$

Conditions (3) and (4) are satisfied as well since $\boldsymbol{b^+b}$ and $\boldsymbol{bb^+}$ (both indicated in the above equations) are apparently symmetric matrices. Note that $\boldsymbol{bb^+}$ is a scalar, i.e. a matrix with one row and one column, and thus always symmetric.

References

Ben-Israel, A. & T.N.E. Greville (1974), *Generalized Inverses: Theory and Applications*, New York (Wiley).

Greville, T.N.E. (1960), Some Applications of the Pseudoinverse of a Matrix, *SIAM Review 2*, 15-22.

Moore, E.H. (1920), On the Reciprocal of the General Algebraic Matrix (Abstract), *Bulletin of the American Mathematical Society 26*, 394-395.

Penrose, R. (1955), A Generalized Inverse for Matrices, *Proceedings of the Cambridge Philosophical Society 51*, 406-413.

Rao, C.R. (1962), A Note on a Generalized Inverse of a Matrix with Applications to Problems in Mathematical Statistics, *Journal of the Royal Statistical Society B 24*, 152-158.

Rao, C.R. & S.K. Mitra (1971), *Generalized Inverse of Matrices and Its Applications*, New York (Wiley).

Schmidt, K. (1998), The Use of a Computer Algebra System in Modern Matrix Algebra, *Proceedings of the 3rd International DERIVE and TI-92 Conference.*

Schmidt, K. & G. Trenkler (1998), *Moderne Matrix-Algebra*, Berlin (Springer).

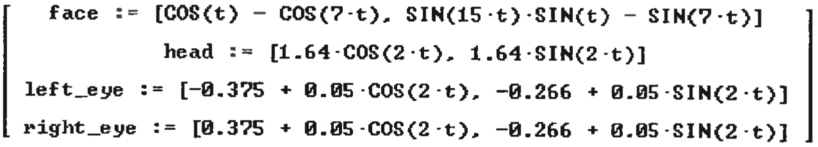
Schmidt, K. (2000), An Application of the Moore-Penrose Inverse of a Matrix to Linear Regression, *Proceedings of the 4th International DERIVE-TI89/92 Conference.*

A Note from David Halprin

davidlaz@net2000.com.au

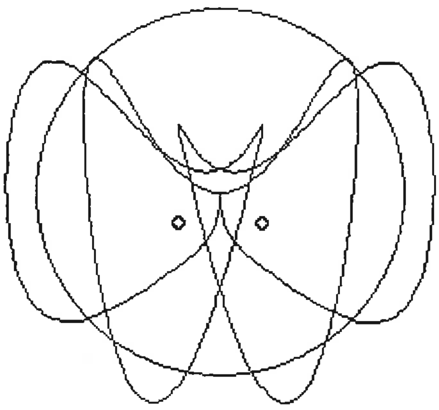
Hi Josef
Just a short note, to tell you that I received *DNL*#47 yesterday and I feel that I should congratulate you on the high standard, that you always produce.
I read your paper on parameters and faces first, and I found it fascinating. By coincidence, only one week ago, while *DNL* was en route, I was dabbling with parametrically expressed spae curves, which are generated by a point in simple harmonic motion being rotated about the three axes simultaneously, and the three projections onto the coordinate planes of that very space curve. I found one that generated a face with spectacles, so I encircled it and added two circles for eyes, and voila, a caricature of me in my better days.

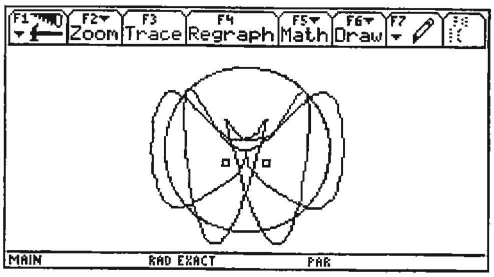
Bis auf zu später
David Halprin



```
[   face := [COS(t) - COS(7·t), SIN(15·t)·SIN(t) - SIN(7·t)]    ]
[          head := [1.64·COS(2·t), 1.64·SIN(2·t)]               ]
[ left_eye := [-0.375 + 0.05·COS(2·t), -0.266 + 0.05·SIN(2·t)]  ]
[ right_eye := [0.375 + 0.05·COS(2·t), -0.266 + 0.05·SIN(2·t)]  ]
```

0 <= t <= PI





Descriptive Statistics with *DERIVE* –
- A Real Life Application

Peter Hofbauer, peter.hofbauer@schule.at
Horn, Lower Austria

History

In fall/winter 2002/2003 OA Dr. Dietmar Weixler from the Waldviertelklinikum Horn (WVK - Hospital in a town in northern Lower Austria) made general inquiries among all doctors who worked in operation theatres. The aim of this inquiry was to investigate the atmosphere in the op-theaters of the WVK.

Most of the answers had to be given by a scale from 1 (= does not bother at all) to 10 (= maximum bothering).

I present a sample of questions from the questionnaire:

I. Which circumstances are influencing my feeling and my performance in the operation theatre in a negative way. Please mark a number between 1 (= does not influence) and 10 (= maximum negative influence).

Part A: Sound

Quest 3 Signal sounds of the anaesthesia control

1 2 3 4 5 6 7 8 9 10

Part B: Illumination, Movement

Quest 1 Quality of illumination

1 2 3 4 5 6 7 8 9 10

II. Which circumstances are influencing my feeling and my performance in the operation theatre in a positive way. Please mark a number between 1 (= maximum positive) and 10 (= maximum negative).

Quest 1 Music (assumption: according to my habits)

1 2 3 4 5 6 7 8 9 10

Data Analyse

For evaluation of the data I was given an Excel table. I intended to use Excel for further working. We fixed the expected graphic representations and numeric statistical measures. Very soon I found that the numbers delivered by Excel could not be correct from my point of view (eg. median, quartile).

(According Excel the 3rd quartile of [1,1,1,1,1,1,1,2,2,2] = 1.75 !!)

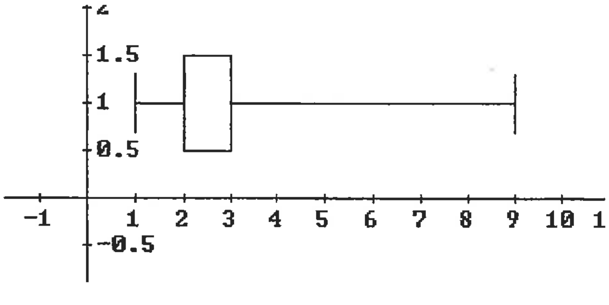
So I decided to use *DERIVE* for calculating the statistics. I had not to "invent the wheel again", but came back to Josef's *DERIVE* – tool (DNL#45 & 46) which – with some slight changes – completely fulfilled our expectations. I appreciated the possibility to enter the data in a list, to receive all requested numbers in one single step and to immediately represent them graphically.

Data Import

I want to underline the fact that it is very easy to import the data from an Excel-table to *DERIVE*: The data were exported from Excel into a text-file (Separation of the columns by comma. We had to take in account the non-answered questions because they were interpreted by Excel as zero and would have led to incorrect results). The text-file could be imported directly to *DERIVE*. Having performed the necessary calculations and plots we had no problems to import the results into the final report as a Word document.

The following *DERIVE*-section shows the collected data from questionnaire Part A and one special treatment of answers for Question 5.

```
abschnittA := [[4, 8, 8, 3, 5, 4, 4, 8, 3, 6, 2, 2, 1, 2, 1, 3, 5, 6, 1, 2, 3, 10,
  2, 4, 2, 1, 5, 2], [7, 9, 8, 7, 4, 10, 3, 10, 5, 9, 3, 4, 1, 2, 1, 5, 9, 8, 1, 6,
  5, 10, 6, 10, 3, 1, 9, 3], [1, 1, 3, 5, 1, 2, 1, 2, 2, 1, 1, 1, 1, 2, 1, 2, 6, 5,
  1, 2, 2, 6, 2, 9, 2, 1, 5, 1], [5, 4, 5, 6, 2, 2, 2, 2, 2, 3, 1, 4, 1, 1, 10, 2,
  9, 8, 2, 3, 4, 9, 3, 10, 1, 1, 5, 5], [2, 3, 3, 6, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1,
  1, 2, 9, 5, 1, 2, 2, 3, 6, 2, 1, 1, 2], [1, 3, 6, 3, 4, 2, 1, 5, 2, 3, 6, 2, 1,
  1, 1, 4, 1, 1, 1, 2, 2, 2, 3, 1, 1, 1, 2], [7, 2, 6, 6, 7, 9, 8, 8, 8, 5, 7, 3,
  1, 2, 10, 4, 4, 6, 1, 4, 4, 10, 10, 7, 2, 1, 5, 4], [10, 2, 8, 10, 8, 7, 9, 9, 5,
  10, 3, 1, 1, 3, 3, 6, 10, 7, 1, 3, 3, 4, 9, 3, 3, 1, 5, 7], [1, 1, 4, 1, 1, 3, 5,
  3, 1, 2, 2, 1, 3, 2, 5, 2, 10, 1, 1, 8, 1, 10, 10, 3, 2, 10, 1, 1], [6, 10, 9, 1,
  1, 10, 1]]

quest5 := abschnittA
                    5

                 [ Datensätze:    27 ]
                 [   Minimum:      1 ]
                 [   Maximum:      9 ]
                 [ Modalwert(e)  [2] ]
KENNZ(quest5) =  [ 1.Quartile:     2 ]
                 [    Median:      2 ]
                 [ 3.Quartile:     3 ]
                 [ Halbweite:      1 ]

BOXPLOT(quest5, 1)
```

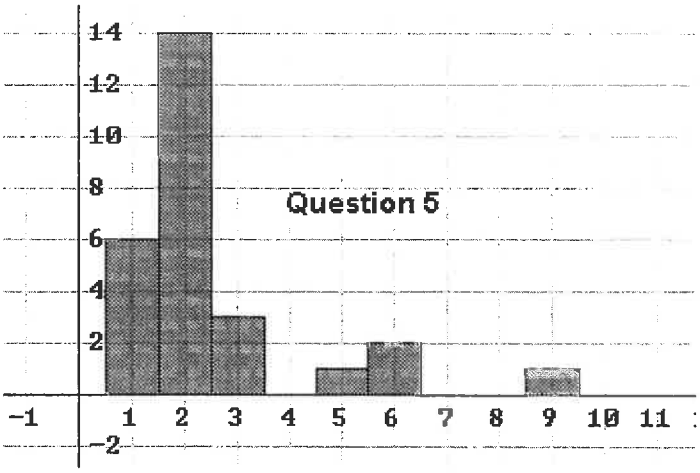


Comment

Special attention must be directed to the fact that in this case a real-life application can be treated using basic statistic methods only. All the statistical measures used are part of secondary 1 level curriculum. But it must be added that publication of these numbers and diagrams cannot be considered to be complete without an analysis of the underlying method of collecting the data (inquiry). And especially this critical inspection of the results might lead students of secondary 2 level to a deeper understanding of statistics. (See the respective comment given in the final report).

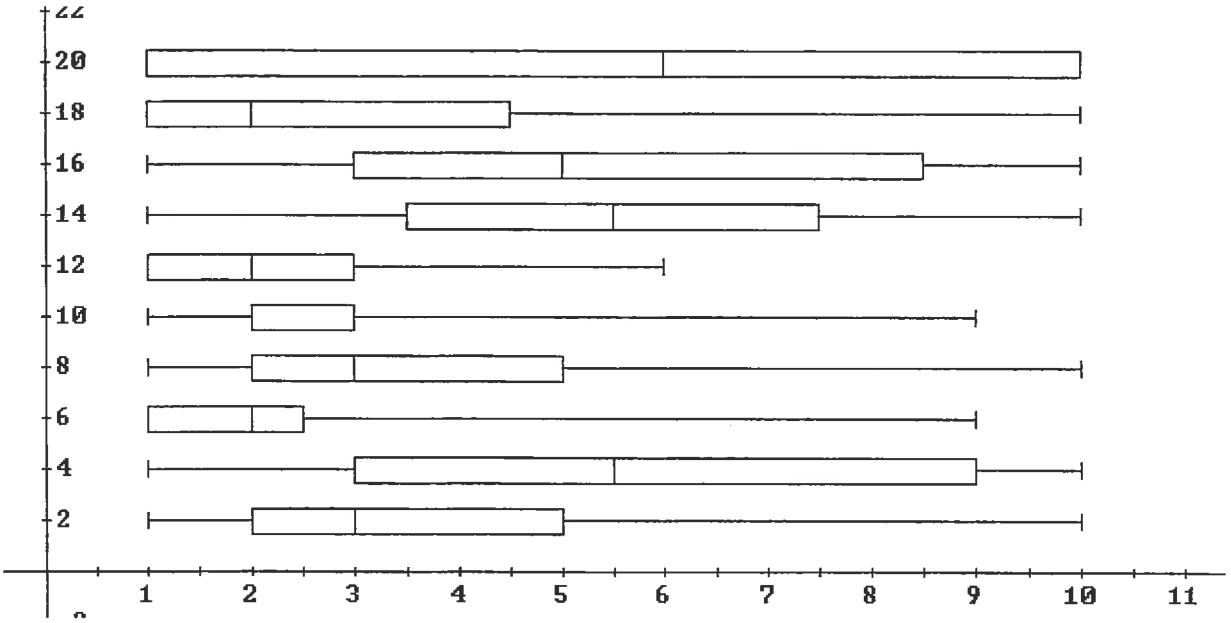
The next page shows a *DERIVE*-histogram, the statistics and box plots for all questions of Part A.

```
histosh(quest5, 0.5, 10.5, 10)
```



```
VECTOR(KENNZ(abschnittA ), i, DIM(abschnittA))
                       i
```

```
[[ Datensätze:   28  ]  [ Datensätze:        28          ]  [ Datensätze:  28  ]  [
 [  Minimum:      1  ]  [  Minimum:           1          ]  [  Minimum:     1  ]  [
 [  Maximum:     10  ]  [  Maximum:          10          ]  [  Maximum:     9  ]  [
 [ Modalwert(e)  [2] ], [ Modalwert(e)  [1, 3, 9, 10]    ], [ Modalwert(e) [1] ], [
 [ 1.Quartile:    2  ]  [ 1.Quartile:         3          ]  [ 1.Quartile:   1  ]  [
 [   Median:      3  ]  [   Median:          5.5         ]  [   Median:     2  ]  [
 [ 3.Quartile:    5  ]  [ 3.Quartile:         9          ]  [ 3.Quartile:  2.5 ]  [
 [ Halbweite:     3  ]  [ Halbweite:          6          ]  [ Halbweite:   1.5 ]  [
  Datensätze:   28  ]  [ Datensätze:  27 ]  [ Datensätze:  27 ]  [ Datensätze:
   Minimum:      1  ]  [  Minimum:     1 ]  [  Minimum:     1 ]  [  Minimum:
   Maximum:     10  ]  [  Maximum:     9 ]  [  Maximum:     6 ]  [  Maximum:
  Modalwert(e)  [2] ], [ Modalwert(e) [2] ], [ Modalwert(e) [1] ], [ Modalwert(e)
```



Kategorie 1 / Abschnitt A (Schall) / Frage 5

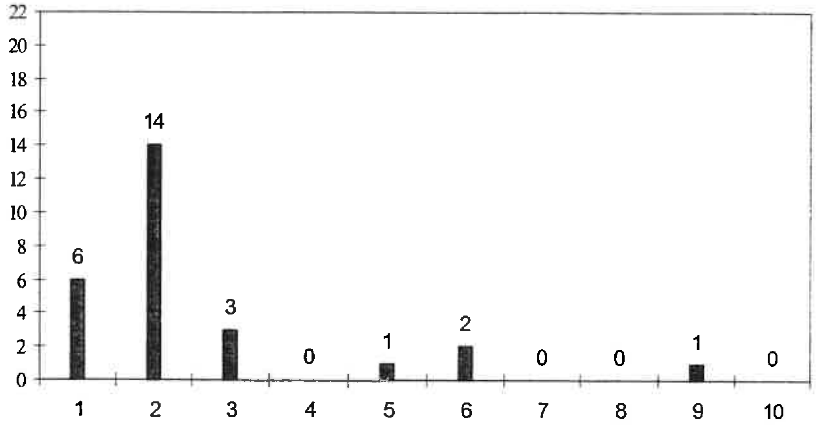
Geräusche durch sonstige Anästhesiegeräte
(Cellsaver, Narkosegasabsaugung, Bairhugger etc.)

Kennzahlen

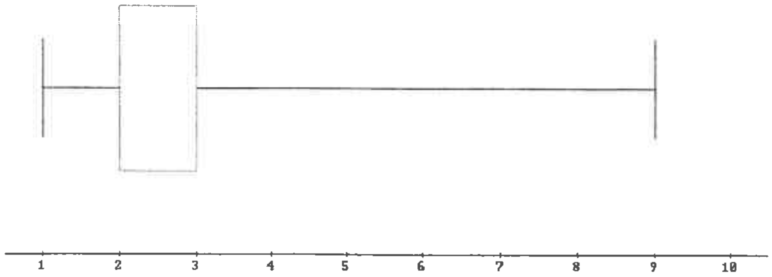
Datensätze	27
Minimum	1
Maximum	9
Modalwert	2
1.Quartile	2
Median	**2**
3.Quartile	3
Halbweite (IQR)	1

Säulendiagramm

(absolute Häufigkeiten – absolute frequencies)



Graphische Darstellung der Kennzahlen



Anmerkungen (Comments):

Keine (non)

Excerpt from the Evaluation and Final Report:

.... Using a scale (1 – 10) in this questionnaire we must indicate the characteristic of this kind of gathering data. The choice of a rating scale (more or less arbitrary) allows only limited usual description of data by mean and standard deviation.

... It is also necessary to point out that given evaluation scales can cause bias.

... Missing data records were ignored and reduced the number of the data to be evaluated.

Visualising Linear Programming Problems (& an old Debt)

(1) Question of an Austrian colleague:

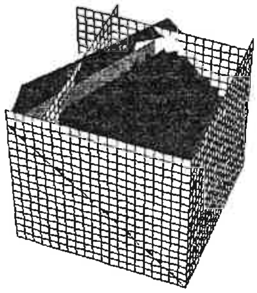
Is it possible in *DERIVE* to represent a Linear Optimisation Problem in three dimensions. I´d like to represent the goal funktion $z = 5x + 8y$ only above the region which is described by the set of constraints:

$$x \geq 0 \text{ AND } y \geq 0 \text{ AND } x \leq 9 \text{ AND } y \leq 8 \text{ AND } 5x + 4y \leq 60 \text{ AND } x + 2y \leq 18.$$

Attempt One: Edit goal function and constraints (as equations) and plot with

0 x 10, 0 y 10,0 z 90.

```
[z = 5·x + 8·y, x + 2·y = 18, 5·x + 4·y = 60, y = 0, x = 9, y = 8, x = 0]
```



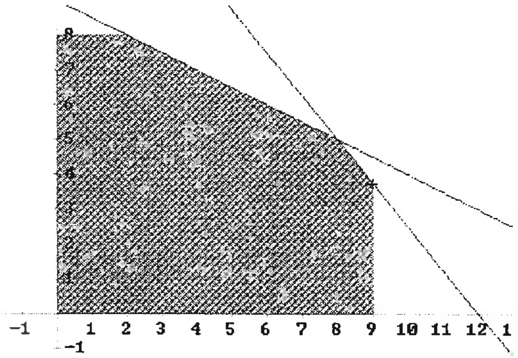
You can imagine that I was not really satisfied with this model.

So I tried .....

Attempt Two: I inspected the problem in the *xy*-plane first.

I define the region of the goal function plane and add the boundary points of the restricted area in plane as point list to have a “top view”.

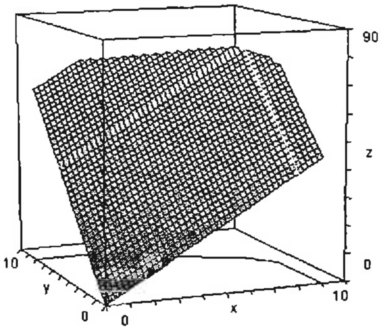
```
0 ≤ x ≤ 9 ∧ 0 ≤ y ≤ 8 ∧ 5·x + 4·y ≤ 60 ∧ x + 2·y ≤ 18
```

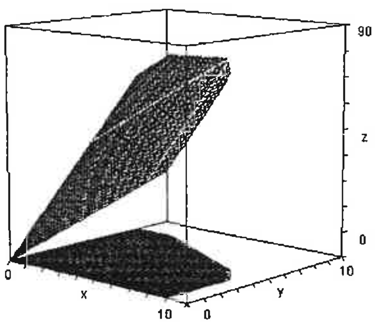


```
SOLVE(5·x + 4·y = 60 ∧ x + 2·y = 18, [x, y]) = (x = 8 ∧ y = 5)
```

```
IF(0 ≤ x ≤ 9 ∧ 0 ≤ y ≤ 8 ∧ 5·x + 4·y ≤ 60 ∧ x + 2·y ≤ 18, 5·x + 8·y)
```

$$\begin{bmatrix} 0 & 0 & 0 \\ 9 & 0 & 0 \\ 9 & \frac{15}{4} & 0 \\ 8 & 5 & 0 \\ 2 & 8 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$





Plotting a Boolean expression the coloured base polygon together with the "goal plane".
Plotting the Boolean expression in the plane adds a coloured base polygon:

```
IF(0 ≤ x ≤ 9 ∧ 0 ≤ y ≤ 8 ∧ 5·x + 4·y ≤ 60 ∧ x + 2·y ≤ 18, 0)
```

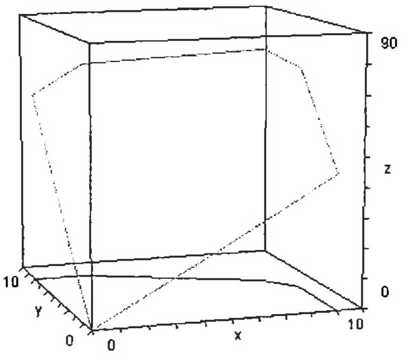
Another nice impression is given by plotting

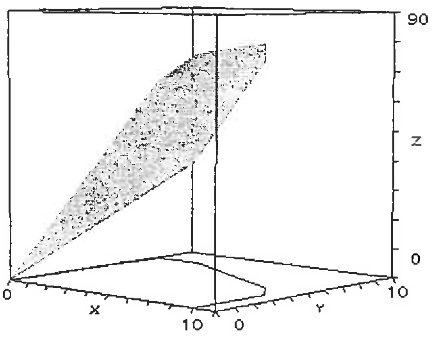
```
IF(0 ≤ x ≤ 9 ∧ 0 ≤ y ≤ 8 ∧ 5·x + 4·y ≤ 60 ∧ x + 2·y ≤ 18, 5·x + 8·y, 0)
```

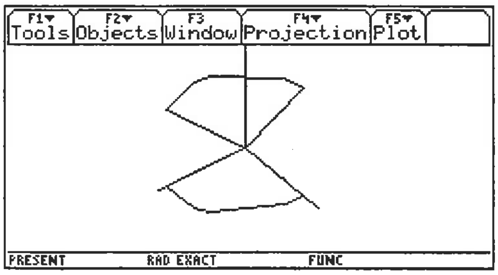
An additional very helpful feature is offered by applying the TRACE-option. One can trace the optimal solution (i.e. the highest point of the plane) and read off in the bottom left corner: `Cross:8,5,80.`

Attempt Three: We calculate the edges of the polygon in the goal-function plane and plot this polygon together with the base:

```
ziel(x, y) := 5·x + 8·y
simpl3d := VECTOR([v_1, v_2, ziel(v_1, v_2)], v_, simpl)
```







Using an appropriate tool (see *DNL#...*) one can represent the figures on the TI-92/Voyage 200, too.

And finally I apply a function to fill the polygon (see the respective contribution in this *DNL*):

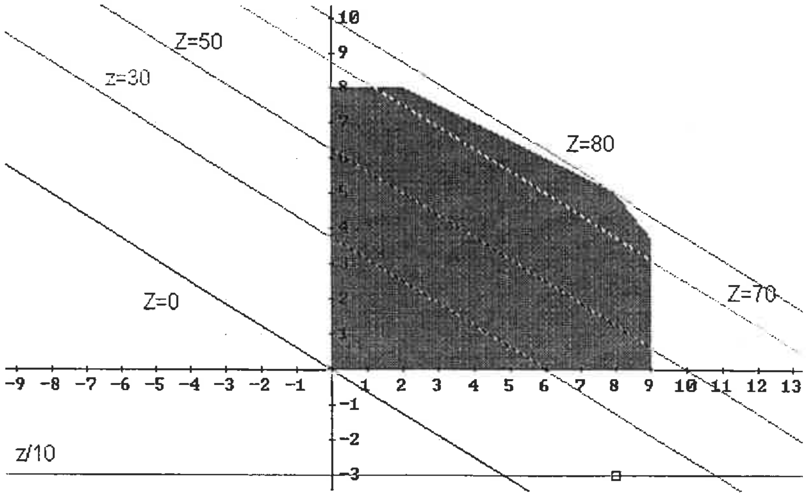
`poly_3d(simpl3d)`

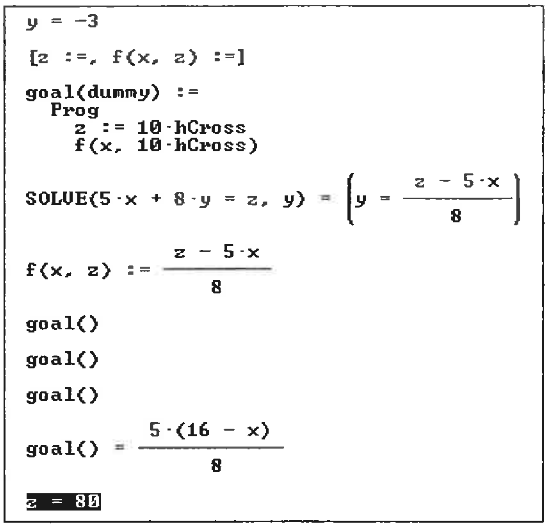
(2) Question in a *DERIVE* course:

Is it possible in *DERIVE* to visualise the various goal function lines simulating the shift process how it is done by hands using something like a slide bar?

My answer was: "Yes, it is, Use a very simplified form of my gglider() from the *DNL*. Take the example from above:

Move the "Trace Box" along the reference line `z/10` at any position and simply plot `goal()`.





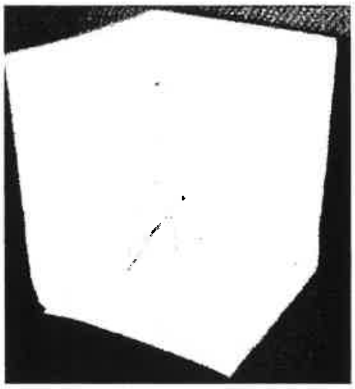
```
y = -3

[z :=, f(x, z) :=]

goal(dummy) :=
  Prog
    z := 10·hCross
    f(x, 10·hCross)

                                    ⎧    z - 5·x⎫
SOLVE(5·x + 8·y = z, y) =           ⎨y = ───────⎬
                                    ⎩       8   ⎭

            z - 5·x
f(x, z) := ───────
               8

goal()

goal()

goal()

          5·(16 - x)
goal() = ──────────
              8

z = 80
```

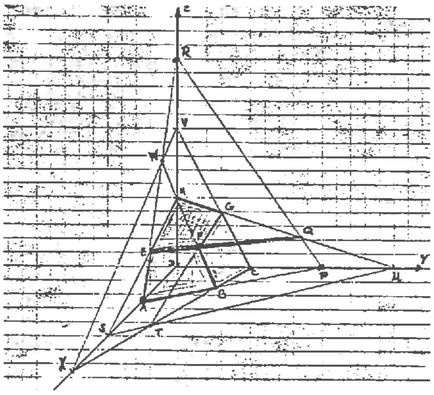
The *DERIVE* code for the "gliding" goal function line.

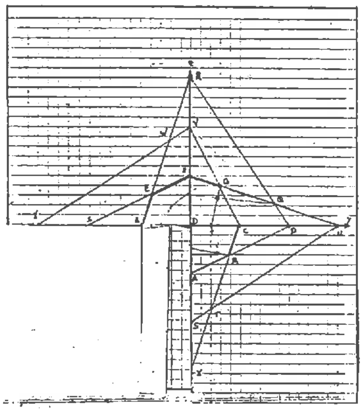
(3) An Old Debt

Reading the Preview of future contributions you could alsways find Georg AUE, Germany. Georg has been a *DUG*-member since the very beginning in 1991. Several years ago – in times of DOS-*DERIVE* – he sent a letter – there was no email at these times – concerning the simplex algorithm. Together with hisletter was a diskette containing two examples how to solve LP-Problems in 3 variables performing the necessary matrix trnas-formations step by step.

He wrote that his students had problems to imagine the 3D-representation and so they first made a sketch in oblique view and then in top-, front- and side view. The latter could be folded to a real 3D-model using glue and little pieces of wire (paper clips). He included a photo of the model.

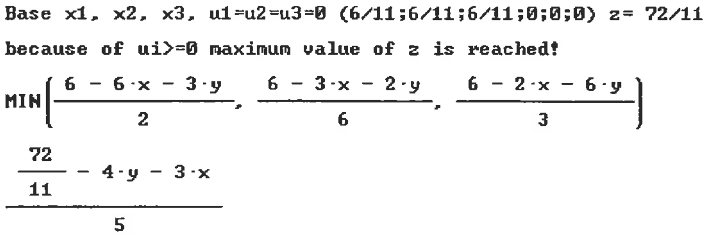




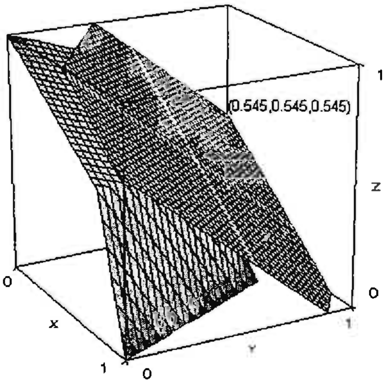


Now in times of *DfW* one can give an impressive 3-D model very easily. The first line is Georg´s solution, followed by the line producing the model of the simplex determined by three planes (assumed *x*, *y*, *z* >=0) and the final solution.

$$6x + 3y + 2z \leq 6$$
$$3x + 2y + 6z \leq 6$$
$$2x + 6y + 3z \leq 6$$
$$3x + 4y + 5z = \text{Maximum}$$



```
Base x1, x2, x3, u1=u2=u3=0 (6/11;6/11;6/11;0;0;0) z= 72/11

because of ui>=0 maximum value of z is reached!

    ⎛ 6 - 6·x - 3·y   6 - 3·x - 2·y   6 - 2·x - 6·y ⎞
MIN ⎜ ─────────────, ─────────────, ───────────── ⎟
    ⎝       2               6               3       ⎠

 72
 ── - 4·y - 3·x
 11
──────────────
       5
```



Richards Challenge (1)

Mrs Britta Brigge from Hannover sent a letter and wrote that she was introduced to Programming with *DERIVE* and to the *DNL* by her colleague Rüdiger Baumann and she immediately did her first steps in Programming facing Richard´s Challenge.
Here are her remarkable findings:

At first I defined function `RFolge()`, which is Richards Folge (Richard´s Sequence) as follows:

```
RFolge(n0 := 9371, length := 10, k, numb) :=
  Prog
    k := FLOOR(LOG(n0, 10))
    numb := FLOOR(n, 10) + 10^k·MOD(FLOOR(MOD(n, 100), 10) + MOD(n, 10), 10)
    If length < ∞
       ITERATES(numb, n, n0, length)
       ITERATES(numb, n, n0)
```

and received the numbers as given in the *DNL*:

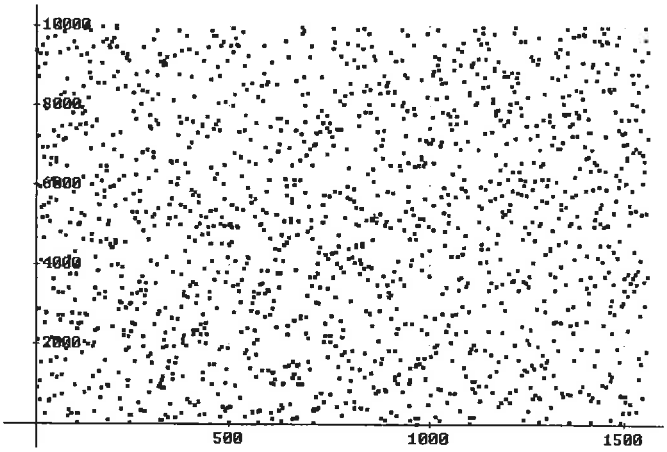
```
RFolge() = [9371, 8937, 893, 2089, 7208, 8720, 2872, 9287, 5928, 592, 1059]

DIM(RFolge(9371, ∞)) - 1 = 1560
```

The period length appeared as 1560 assuming that there is no preperiod, what I didn´t check.

It seems to be that the numbers appear more or less chaotic (similar to a random number generator).
To visualise my impression I represented the sequence in the coordinate system.
Using and applying

```
repr(v) := VECTOR([i, v ], i, DIM(v))
                       i

repr(RFolge(9371, ∞))
```

I received the following graphic representation



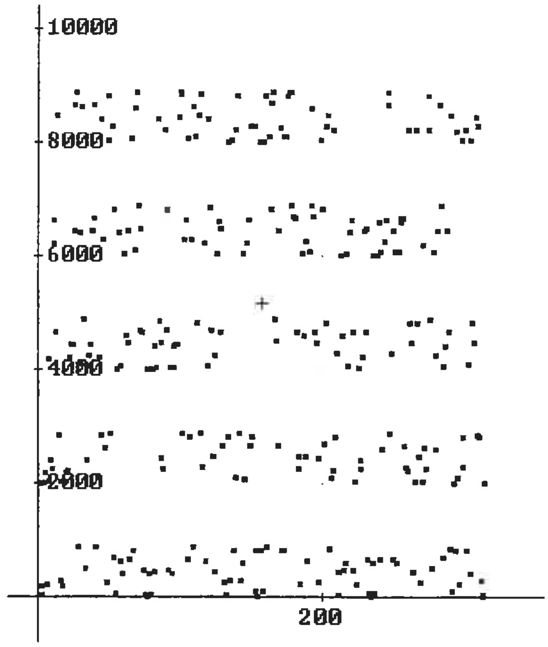
Now I had to answer the question if all sequences would have the same period length and would show a similar behaviour. I tried other initial values 1000, 2000, ……, 9000:

```
VECTOR([1000·k, DIM(RFolge(1000·k, ∞)) - 1], k, 9)`

         ┌ 1000  2000  3000  4000  5000  6000  7000  8000  9000 ┐
         └ 1560   312  1560   312   15    312  1560   312  1560 ┘
```

So we have other period lengths. Funny is the sequence starting with 5000:

```
RFolge(5000, ∞)
   [5000, 500, 50, 5005, 5500, 550, 5055, 505, 5050, 5505, 5550, 5555, 555, 55, 5, 5000]
```

and its graphic representation. The numbers are not uniformly distributed they remain in certain intervals.



My conjecture is: If the initial number consists of digits 0 and 5, the period length is 15; if it consists of digits 2, 4, 6, 8, 0 (even digits) only then the period length is 312, in all other cases it is 1560.

To verify (proof?) this conjecture, I define the function `Periods()`:

```
Periods(n := 10, n1 := 99, Set_ := {}) :=
  Loop
    If n > n1
       RETURN Set_
    Set_ := Set_ ∪ {DIM(RFolge(n, ∞)) - 1}
    n :+ 1

Periods() = {3, 4, 12, 20, 60}
```

For two-digit numbers we find 5 different period lengths. The shortest periods show the numbers composed of 0 and 5, which can be seen as follows:

```
SELECT(DIM(RFolge(numb, ∞)) - 1 = 3, numb, 10, 99) = [50, 55]

RFolge(55, ∞) = [55, 5, 50, 55]

RFolge(50, ∞) = [50, 55, 5, 50]
```

Three-digit initial values result in possible period lengths as follows:

```
Periods(100, 999) = {4, 7, 24, 28, 168}

RFolge(268, ∞) = [268, 426, 842, 684, 268]
```

Exact four sequences have period with length 4: [268, 426, 684, 842]. In this case not the numbers composed of 0 and 5 give the shortest periods. These numbers [500, 505, 550, 555] return sequences with periods 7, 96 numbers have periods 28 and the remaining 768 numbers period 168.

```
[                         VECTOR(k, k, {4, 7, 24, 28, 168})
[ VECTOR(DIM(SELECT(DIM(RFolge(numb, ∞)) - 1 = j, numb, 100, 999)), j, {4, 7, 24, 28,

  168}) ]

                               [ [4, 7, 24, 28, 168] ]
                               [ [4, 4, 96, 28, 768] ]
```

For five digit numbers I found out the following: there are 7 different period lengths with the distribution as shown in the table:

Period lengths	3	7	21	781	2343	5467	16401
Number of cycles	2	4	10	2500	8748	19996	58740

The minimum initial value with period 3 is 50550, because ...

```
RFolge(50550, ∞) = [50550, 55055, 5505, 50550]
RFolge(50050, ∞) = [50050, 55005, 55500, 5550, 50555, 5055, 505, 50050]
RFolge(50000, ∞) = [50000, 5000, 500, 50, 50005, 55000, 5500, 550, 50055, 5005, 50500,
  5050, 50505, 55050, 55505, 55550, 55555, 5555, 555, 55, 5, 50000]
```

... the minimum initial value giving period 7 is 50050, minimum initial value with period 21 is 50000, etc.

Richards Challenge (2)

I received another (surface-) mail from Japan:

A Solution to Richard´s CHALLENGE (#49, p.39) from the Japanese Senior Trio:

Yoshihiro NAKANO (80 yrs old)
Kiyoshi YAMASHITA (78 yrs old)
Toshio NISHIKAWA (69 yrs old)

We investigated the periodicity in Richard´s CHALLENGE using the 10-digit initial number 1234567890.

The period is found as 1 736 327 236 or 2 * 2 * 7 * 19 * 31 * 127 * 829.

We used APL-like language J (version J4.06a) of K.E.Iverson in Toronto, Canada.

The time of calculation is 469441 sec (or 5 d 10 h 24 m 1 s) by IBM PC (Aptiva, 300 MHz, WIN 98).

Our proposal to CHALLENGE:

How many kinds of periodicity are there in the for example 3-digit initial numbers (100 – 999)?

Part 2:

This letter is a continuation of the former letter. We have investigated “How many kinds of periodicity are there in Richard´s Numbers from the 1-digit to the 10-digit case?”.

Part 3:

This time we would like to send you the result using *DERIVE* (Ver. 2.5 from 1993).

```
RICH10(n) := ITERATE([MOD(ELEMENT(v, 9) + ELEMENT(v, 10), 10), ELEMENT(v, 1), ELEMENT(v,
  2), ELEMENT(v, 3), ELEMENT(v, 4), ELEMENT(v, 5), ELEMENT(v, 6), ELEMENT(v, 7),
  ELEMENT(v, 8), ELEMENT(v, 9)], v, [1, 2, 3, 4, 5, 6, 7, 8, 9, 0], n)
```

The results for various data vectors v are:

[5,0,0,5,5,5,0,5,0,0]	7	calculation time < 1 sec
[0,5,5,5,5,0,5,5,5,5]	127	3 sec
[5,5,5,5,5,5,5,5,5,5]	889	15 sec
[2,0,0,0,0,0,0,0,0,0]	1 953 124	9 hours
[1,2,3,4,5,6,7,8,9,0]	1 736 327 236	5 d 10 h 24 m

Digits	Period	How many members	Examples
1	1	1-digit & all "0" (trivial)	
2	3	3	05, 50, 55
	4	4	24, 48 62, 86
	12	12	12, 17, 29, 31, 36, …, 40, 42
	20	20	02, 04, 06, 08, …, 84, 88
	$60 = 2^2*3*5$	60	01, 03, 07, 09, …, 99
3	4	4	268, 426, 684, 842
	7	7	005, 050, 055, 500, 505, 550, 555
	28	28	100, …
	24	120 = 24 * 5	002, …
	$168 = 2^3*3*7$	840 = 168 * 5	007, ...
4	15	15	0005, 0050, …, 5550, 5555
	312	624 = 312 * 2	0002, 0004, …, 8888
	1560	9360 = 1560 * 6	0001, 0003, …, 9999
5	3	3	05505, 50550, 55055
	7	7	00505, 05055, 05550, 50050, ...
	21	21	00005, 00050, ..., 55555
	781	3124 = 781 * 4	00002, 00004, ..., 88888
	2343	9372 = 2343 * 4	01101, 01103, ..., 99899
	5467	21868 = 5467 * 4	00101, 00103, ..., 99988
	16401 = 3*7*11*71	65604 = 16401 * 4	00001, 00003, ..., 99999
6	63	63 (all "5-family")	000005, …, 050505, …, 505050
	3124	63 (all "2-family")	000002, …, 020202, …, 202020
	$196812 = 2^2*3^2*7*11*71$	others	
7	127	127 (all "5-family")	5000000, ...
	5208	127 (all "2-family")	2000000, ...
	661416 = $=2^3*3*7*11*31*127$	others	
8	3	3	05505505, 50550550, 55055055
	63	252 (all other "5-fam.")	50000000, ...
	2232	255 (all "2-family")	20000000, ...
	$15624 = 2^3*3^3*7*31$	others	
9	73	511 (all "5-family")	
	121836	511 (all "2-family")	
	8894028	others	
10	7	7	special members of the "5-family"
	127	127	other members of the "5-family"
	889 = 7*127		rest of the "5-family"
	1953124	1023 ("2-family)	
	1736327236 = $=2^2*7*19*31*127*829$	others	

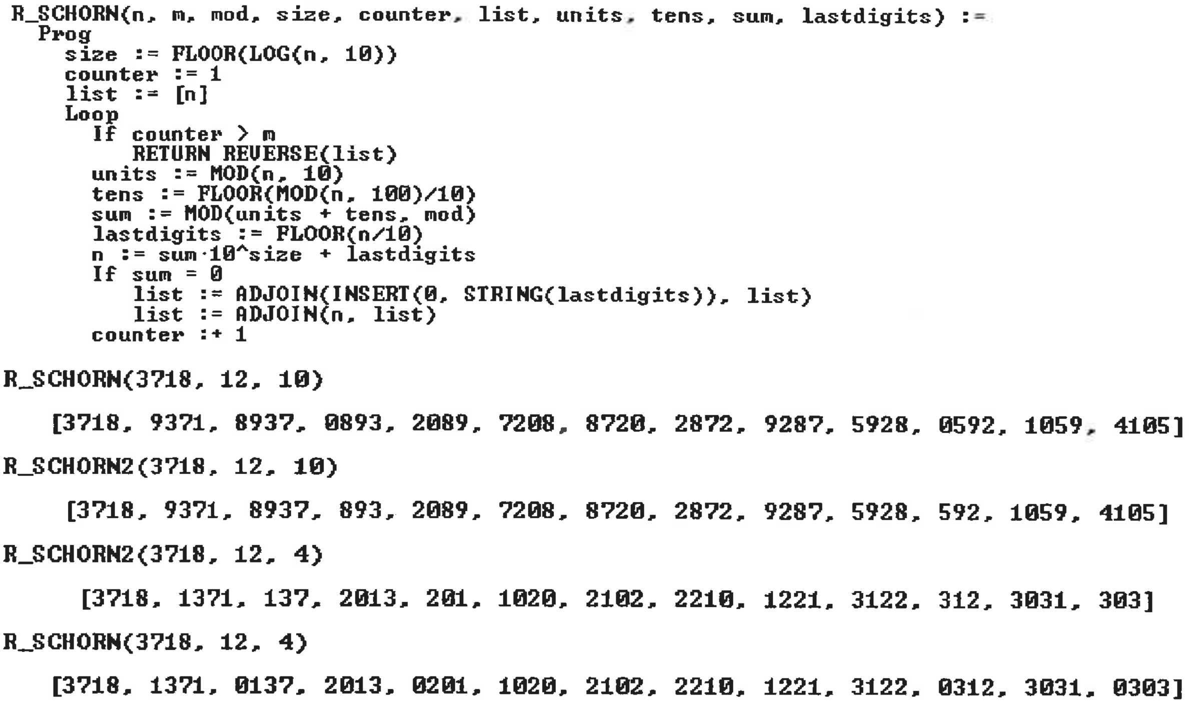
We have noticed that intimate parallelism exists between the period and the numbers of the members.

nakano@mta.biglobe.ne.jp

The results differ from Mrs. Brigge´s because the Japanese Seniors accepted leading zeros as parts of the numbers (strings).

Richards Challenge (3)

The first answer of all came from Terence Etchelles. He sent a DERIVE file containing two functions One of them showed the leading zero in the output, the other did not. The complete file is included in the files of this issue.



```
R_SCHORN(n, m, mod, size, counter, list, units, tens, sum, lastdigits) :=
  Prog
    size := FLOOR(LOG(n, 10))
    counter := 1
    list := [n]
    Loop
      If counter > m
         RETURN REVERSE(list)
      units := MOD(n, 10)
      tens := FLOOR(MOD(n, 100)/10)
      sum := MOD(units + tens, mod)
      lastdigits := FLOOR(n/10)
      n := sum·10^size + lastdigits
      If sum = 0
         list := ADJOIN(INSERT(0, STRING(lastdigits)), list)
         list := ADJOIN(n, list)
      counter :+ 1

R_SCHORN(3718, 12, 10)

   [3718, 9371, 8937, 0893, 2089, 7208, 8720, 2872, 9287, 5928, 0592, 1059, 4105]

R_SCHORN2(3718, 12, 10)

    [3718, 9371, 8937, 893, 2089, 7208, 8720, 2872, 9287, 5928, 592, 1059, 4105]

R_SCHORN2(3718, 12, 4)

     [3718, 1371, 137, 2013, 201, 1020, 2102, 2210, 1221, 3122, 312, 3031, 303]

R_SCHORN(3718, 12, 4)

   [3718, 1371, 0137, 2013, 0201, 1020, 2102, 2210, 1221, 3122, 0312, 3031, 0303]
```

Fill your 3D-polygons with *DERIVE*

Josef Böhm

In GRAPHICS.MTH you can find the useful function `POLYGON_FILL(pointlist)` which makes possible to present a filled polygon in R^2 and R^3. I can imagine that it is not so easy to explain this function including crossproducts to students (- to make the "Black Box" white). I believe that the following function is possibly easier to understand:

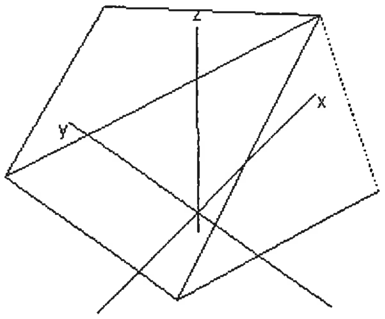
$$\text{poly\_3d(v)} := \text{VECTOR}\left(\begin{bmatrix} v_1 & v_i \\ v_1 & v_{i+1} \end{bmatrix}, i, 2, \text{DIM(v)} - 1\right)$$

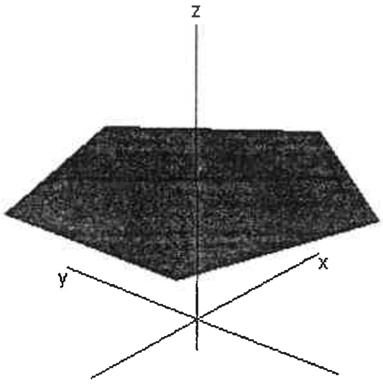
`v` is the point list which generates the polygon. `poly_3d(v)` divides the figure into triangular areas which all have one edge – the first point of the list – in common. We can demonstrate this by defining and then plotting a pentagon first as wire grind and then by customizing the plot settings:

$$\text{penta} := \text{VECTOR}\left([4\cdot\text{COS(t)}, 4\cdot\text{SIN(t)}, 4], t, 0, \frac{8\cdot\pi}{5}, \frac{2\cdot\pi}{5}\right)$$

`poly_3d(penta)`

For the filled polygon set `Insert Plot > Plot Color > Scheme > Custom` and then set all colors the same.

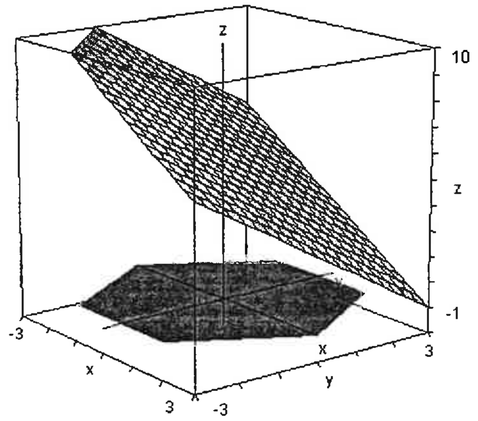




As second example I´d like to present the intersection of prism with hexagonal base and a plane.

Plane: x + y + z = 6

$$\#9: \quad \left[a := [3,\ 0,\ 0],\ b := \left[\frac{3}{2},\ \frac{3}{2}\cdot\sqrt{3},\ 0\right],\ c := \left[-\frac{3}{2},\ \frac{3}{2}\cdot\sqrt{3},\ 0\right],\ d := [-3,\ 0,\ 0],\ e := \left[-\frac{3}{2},\ -\frac{3}{2}\cdot\sqrt{3},\ 0\right],\ f := \left[\frac{3}{2},\ -\frac{3}{2}\cdot\sqrt{3},\ 0\right]\right]$$



The figure shows the base and the intersecting plane.

```
hexa := [a, b, c, d, e, f]
poly_3d(hexa)
z = 6 - x - y
```

In two steps we can add the intersection figure which is another hexagon.

The lateral faxes accomplish the figure.

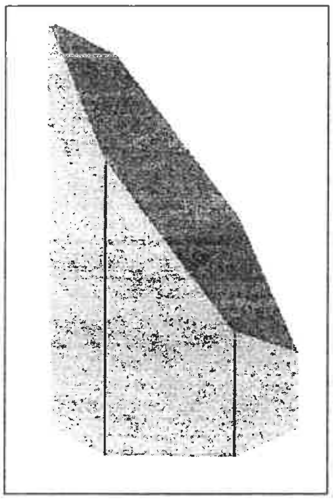
$$\text{hexa1} := \text{VECTOR}([v_1,\ v_2,\ 6 - v_1 - v_2],\ v,\ \text{hexa})$$

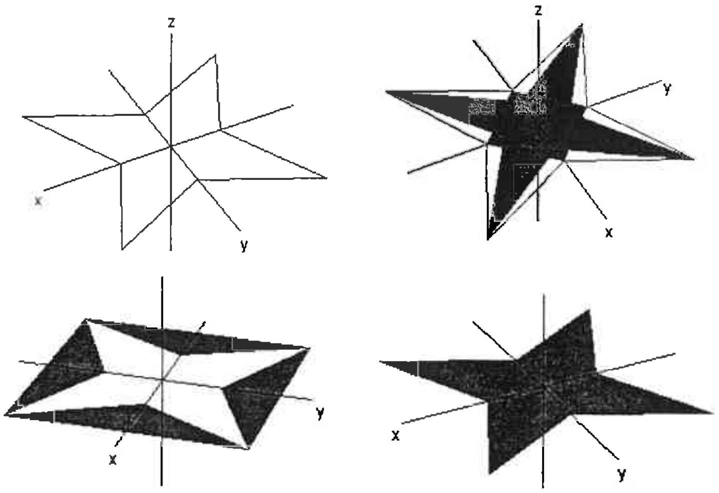
$$\text{poly\_3d(hexa1)}$$

$$\text{APPEND(VECTOR(poly\_3d}([\text{hexa}_i,\ \text{hexa}_{i+1},\ \text{hexa1}_{i+1},\ \text{hexa1}_i]),\ i,\ 1,\ 5),\ \text{poly\_3d}([\text{hexa}_6,\ \text{hexa}_1,\ \text{hexa1}_1,\ \text{hexa1}_6]))$$

We define a star by its points and give different presentations by varying the order of the points in the pointlis.t

```
[s1 := [4, 4, 0], s2 := [0, 2, 0], s3 := [-4, 4, 0], s4 := [-2, 0, 0], s5 := [-4, -4,
  0], s6 := [0, -2, 0], s7 := [4, -4, 0], s8 := [2, 0, 0]]

star := [s1, s2, s3, s4, s5, s6, s7, s8, s1]
```





Variations of a Star

Given are the edges of a star. Use `poly_3d()` to produce the various figures.

Ways to write with *DERIVE* and TI

By Milton Lesmes Acosta, Bogotá, Colombia

It is known that *DERIVE* does not work as a word processor. So I would like to give some ideas about programming to start writing and modifying the fonts.

I believe that is important to report that this ideas are the result of an effort to teach mathematical concepts like linear equations, functions, matrices, starting with problems. The idea was developed by two of my students Yenni Andrea Castillo and Francy Angélica Riveros from the Universidad Distrital Francisco José de Caldas facing the problem to design symbols with the use of technology. They -- as students for becoming teachers - proposed a didactic unit to develop mathematical concepts and competences in Colombian secondary schools.

First, to the point with *DERIVE*. Matrices with parametric equations as entries for each character have to be constructed (a decision has to be taken as a result of the design), for example:

$$E := \begin{pmatrix} 0 & 3\tau \\ 2\tau & 0 \\ 2\tau & \frac{3}{2} \\ 2\tau & 3 \end{pmatrix} \qquad A := \begin{pmatrix} \tau & 3\tau \\ \frac{13\tau}{10}+\frac{7}{20} & 1 \\ \tau+1 & 3-3\tau \\ 0 & 0 \end{pmatrix} \qquad C := \begin{pmatrix} 1+\cos(\frac{3\pi\tau}{2}+\frac{\pi}{4}) & \frac{3}{2}+\frac{3}{2}\sin(\frac{3\pi\tau}{2}+\frac{\pi}{4}) \\ 1+\cos(\frac{3\pi\tau}{2}+\frac{\pi}{4}) & \frac{3}{2}+\frac{3}{2}\sin(\frac{3\pi\tau}{2}+\frac{\pi}{4}) \\ 1+\cos(\frac{3\pi\tau}{2}+\frac{\pi}{4}) & \frac{3}{2}+\frac{3}{2}\sin(\frac{3\pi\tau}{2}+\frac{\pi}{4}) \\ 1+\cos(\frac{3\pi\tau}{2}+\frac{\pi}{4}) & \frac{3}{2}+\frac{3}{2}\sin(\frac{3\pi\tau}{2}+\frac{\pi}{4}) \end{pmatrix}$$

as a decision to take matrices 4×2 and parametric functions with $0 \le \tau \le 1$ to build characters in the subset $[0,2]\times[0,3]$ of the plane.

Maybe that this is new for you:
You can define the matrices as "E":=, "I":=, etc. Strings can be used as variable names!!

$$\#2:\quad \left[E := \begin{bmatrix} 0 & 3\cdot\tau \\ 2\cdot\tau & 0 \\ 2\cdot\tau & \frac{3}{2} \\ 2\cdot\tau & 3 \end{bmatrix},\ I := \begin{bmatrix} 1 & 3\cdot\tau \\ 1 & 3\cdot\tau \\ 1 & 3\cdot\tau \\ 1 & 3\cdot\tau \end{bmatrix},\ F := \begin{bmatrix} 0 & 3\cdot\tau \\ 0 & 3\cdot\tau \\ 2\cdot\tau & \frac{3}{2} \\ 2\cdot\tau & 3 \end{bmatrix},\ A := \begin{bmatrix} \tau & 3\cdot\tau \\ \frac{13\cdot\tau}{10} + \frac{7}{20} & 1 \\ \tau + 1 & 3 - 3\cdot\tau \\ 0 & 0 \end{bmatrix}\right]$$

$$\#3:\quad \left[L := \begin{bmatrix} 0 & 3\cdot\tau \\ 0 & 0 \\ 0 & 0 \\ 2\cdot\tau & 0 \end{bmatrix},\ N := \begin{bmatrix} 0 & 3\cdot\tau \\ 2\cdot\tau & 3 - 3\cdot\tau \\ 0 & 0 \\ 2 & 3\cdot\tau \end{bmatrix},\ Z := \begin{bmatrix} 2\cdot\tau & 3\cdot\tau \\ 2\cdot\tau & 0 \\ 2\cdot\tau & 3 \\ 0 & 0 \end{bmatrix},\ T := \begin{bmatrix} 1 & 3\cdot\tau \\ 1 & 3\cdot\tau \\ 2\cdot\tau & 3 \\ 2\cdot\tau & 3 \end{bmatrix}\right]$$

(Comment: The quotes " are not displayed in the Algebra-Window, but in the Edit line.)
You can construct your own characters, I am sure you get the idea, and with auxiliary definitions like:

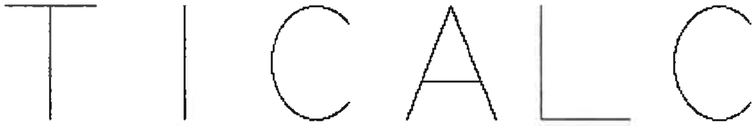
$$\sigma(\lambda, \epsilon) := \begin{bmatrix} \epsilon + \lambda_{1,1} & \lambda_{1,2} \\ \epsilon + \lambda_{2,1} & \lambda_{2,2} \\ \epsilon + \lambda_{3,1} & \lambda_{3,2} \\ \epsilon + \lambda_{4,1} & \lambda_{4,2} \end{bmatrix}$$

$$\text{write(text)} := \text{VECTOR}(\sigma(\text{text}_k,\ 3\cdot(k - 1)),\ k,\ 1,\ \text{DIM(text)})$$

With `text` being a string, you can write. For example with the given matrices for "T", "I", "C", "A" and "L", you enter

`write("TICALC")`

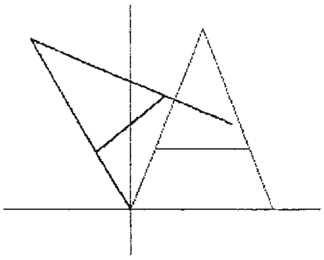
switch to 2D-Plot-Window and get the plot.



Now you can multiply, rotate each letter and finally to try writing in Derive as a word processor in the 2D-plot Window.

perform a rotation of the "A":

$$\#8:\quad A\cdot\begin{bmatrix} \text{COS}\left(\frac{\pi}{4}\right) & \text{SIN}\left(\frac{\pi}{4}\right) \\ -\ \text{SIN}\left(\frac{\pi}{4}\right) & \text{COS}\left(\frac{\pi}{4}\right) \end{bmatrix}$$

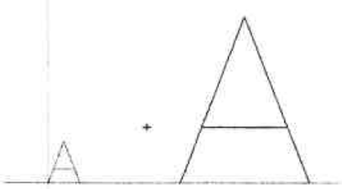


Which transformation is given by the following calculation

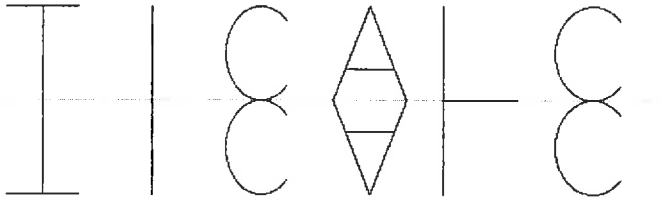
#9: A

#10: 4·σ(A, 2)

Can you perform the same transformation using σ(....., ...)?

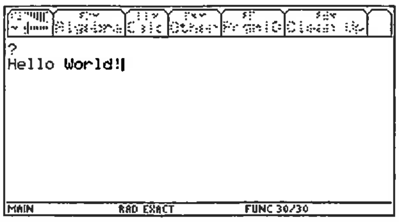


Produce the following picture:

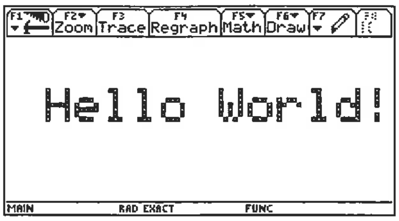


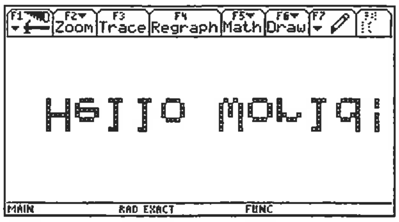
And now with the TI-92 (PLUS/Voyage 200). Check and try to develop the idea what we want to see presented on the TI screen:

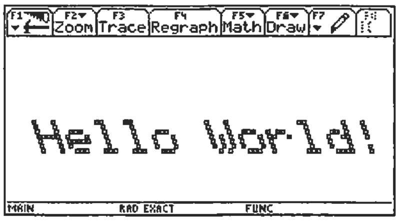
Call program `cartel()` from the home screen.



```
?
Hello World!
```







```
Cartel()
Prgm
ClrIO
ClrDraw
SetGraph("Axes","Off")
InputStr a
PxlText a,1,1
For i,0,10
For j,0,100
If PxlTest(i,j)=true Then
Pxloff i,j
PxlCrcl 3*i+30,3*j+20,1
EndIf
EndFor
EndFor
EndPrgrm
```

The idea is a simple one to use the transformation of the plane

$$T(x,y)=3(x,y)+(30,20)=(3x+30,3y+20)$$

Where and how to change the program to receive other outputs of the given input?

DERIVE- and CAS-TI User Forum 2

Mathematica 4.2.1 ?

```
In[1]  := N[Limit[(Tan[z] - 1)/(Sin[z] - 1/Sqrt[2]), z -> Pi/4], 50]

Out[1] = 2.8284271247461900976033774484193961571393437507539

In[2]  := << NumericalMath`NLimit`
In[2]  := NLimit[(Tan[z]-1)/(Sin[z]-1/Sqrt[2]),z->Pi/4,WorkingPrecision->50]

Out[3] = 2.828368236944547674091879890306081603l
```

ADR> This is an excellent example of the pitfalls of numerical approximations, and the advantages of exact mode.

Okay, let it be so. How then Maple, Mathematica and MuPAD managed not to fall into a snare? Couldn't Derive 5.07 use the same strategy? (hopefully, yes)

DS>> Can someone explain this.

Yes, and the explanation is simple. It's a rare bug in Derive.

In quality of the expert, figures at my fingertip, I am happy to announce that it is Derive that is the least error-prone system as compared with Maple 8.01, Mathematica 4.2.1, and MuPAD 2.5.2.

So you a had a lucky strike to identify a bug I had not identified yet (I envy you with the blackest envy, yes ;)

By the way, if you are interested in various bugs and workarounds in commercial computer algebra systems you may wish to visit my sites in progress

`http://www.cas-testing.org/` Symbolic Testing Group Official Home (95% ready)
`http://maple.bug-list.org/` Maple Bugs Encyclopaedia (20% ready)

Very soon, they will be updated dramatically (which will give you a numerically expressed flavor about how EXCELLENT Derive is :)

Au revoir,

Mathematical and Production Director
Symbolic Testing Group

David Sjöstrand

Hi again,

Thank you Al and Vladimir very much for your answers

If I skip lim then (TAN(PI/4 + h) - 1)/(SIN(PI/4 + h) - 1/sqrt(2)) approximates to 2.82842712.... if h=10^-50 and PrecisionDigits := 100, which makes sense.

I also would like to mention my students Joseph Bentham and Magnus Roeding at Elof Lindaelv´s gymnasium who found this anomaly when they were working with the limit that I had asked them to compute.

Best regards, David Sjöstrand

Cable Problem

Rick Nungester

Given a 100m cable hanging between two equal-height anchors, drooping at the low point 10m below the anchors, what is the span between the anchors? How do I solve this using Derive?

I get to solving these two equations for a (catenary constant) and s(span):

```
a*COSH(s/(2*a)) - a = 10
2*a*SINH(s/(2*a)) = 100
```

But then what? (Given span and droop, length can be found by solving only a single-var iteration. Similarly, given length and span, droop can be found by solving only a single-var iteration. The problem here seems harder.)

Related question: Given length and droop, is there an algebraic solution for the RATIO length/span? This is related to a recent post in newsgroup geometry.puzzles titled "2 poles and a rope".

Rick

Ron Larham

1. Put cosh(*s*/(2*a*)) = *ch*.
2. Use the appropriate identity ot replace sinh(*s*/(2*a*)) by an expression in *ch*.
3. Solve the first equation for *a* in terms of *ch*.
4. Substitute this value of *a* into the second equation.
5. Solve resulting equation for *ch*.
6. Use value of *ch* to find *a*.
7. Use value of *ch* and *a* to find *s*.

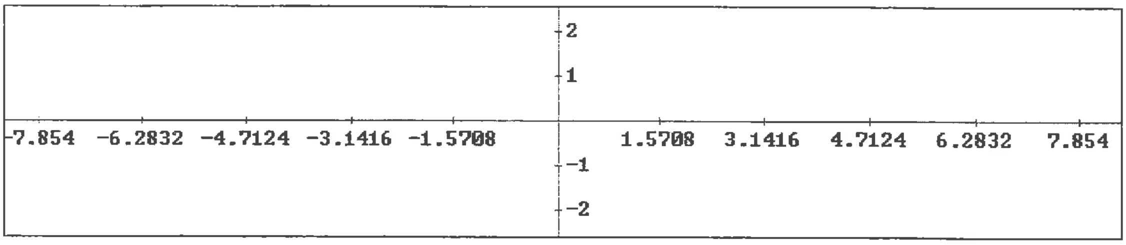
π-Scaling

Louis F. Lowell louis.lowell@verizon.net

Hello all,

I am using Derive 5.05 and would like to have the horizontal scale on the plot screen in multiples of pi/2. I don't recall how this is done, and cannot find information about this in the help file. Can someone refresh my memory?

DNL: I gave the advice to set Plot region > Horizontal length 6PI and 12 Intervals which results in the following 2D-Plot Window (Josef).

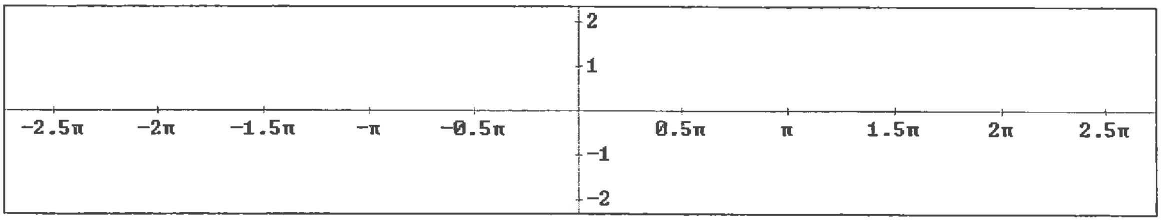


But Abert Rich had a better advice:

Albert Rich

The following is from the Derive 5 on-line topic for the 2D-plot window's Options > Display > Axes command:

The axis scale factors make it possible to display axis labels in a unit appropriate for the expression being plotted. For example, when plotting trigonometric functions, it is often convenient to make the horizontal axis labels multiples of pi. To do this, use the Options > Display > Axes command to set the horizontal axes scale factor to p by clicking on it in the Greek symbol toolbar. Then click on the zoom out icon or press the F6 key to obtain simple multiples of pi.



Michel Gouy

Hello,

I write with *DERIVE* : `eq1:= 2x + 3y + 5 = 4`.

Is-it possible to simplify `eq1` in `2x + 3y = -1`?

Thank you

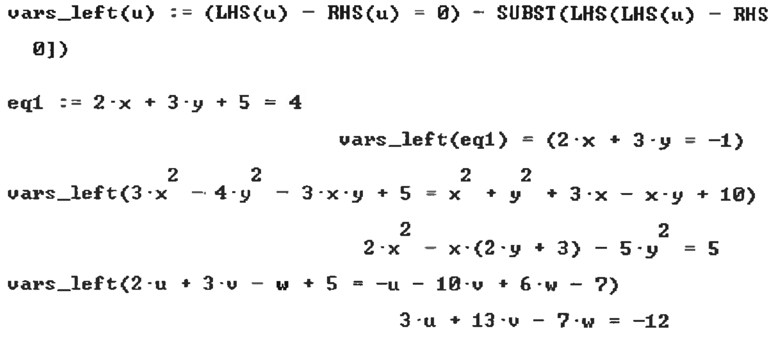
__DNL:__

Hi, Michel,

I am sure that DERIVE doesn´t "simplify" as you like without giving any "orders" what to do.

At another occasion you might prefer to "simplify" `equ1` to `2x + 3y + 1 = 0`.

I wrote a little function, which does the work (putting all variables on the left hand side of the equation) and which can easily be adapted for other "simplifications".



```
vars_left(u) := (LHS(u) - RHS(u) = 0) - SUBST(LHS(LHS(u) - RHS(u)), VARIABLES(u), [0,
  0])

eq1 := 2·x + 3·y + 5 = 4
```

$$\text{vars\_left}(\text{eq1}) = (2\cdot x + 3\cdot y = -1)$$

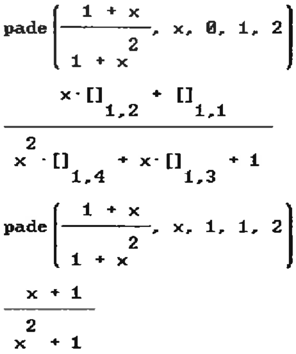
$$\text{vars\_left}(3\cdot x^2 - 4\cdot y^2 - 3\cdot x\cdot y + 5 = x^2 + y^2 + 3\cdot x - x\cdot y + 10)$$

$$2\cdot x^2 - x\cdot(2\cdot y + 3) - 5\cdot y^2 = 5$$

$$\text{vars\_left}(2\cdot u + 3\cdot v - w + 5 = -u - 10\cdot v + 6\cdot w - 7)$$

$$3\cdot u + 13\cdot v - 7\cdot w = -12$$

__Francisco M. Fernández__ http://www.conicet.gov.ar/webue/cequinor/mick.htm,framfer@isis.unlp.edu.ar



$$\text{pade}\left(\frac{1 + x}{1 + x^2}, x, 0, 1, 2\right)$$

$$\frac{x\cdot[\,]_{1,2} + [\,]_{1,1}}{x^2\cdot[\,]_{1,4} + x\cdot[\,]_{1,3} + 1}$$

$$\text{pade}\left(\frac{1 + x}{1 + x^2}, x, 1, 1, 2\right)$$

$$\frac{x + 1}{x^2 + 1}$$

Hi Derivians,

I had a problem with `pade.mth` that appears in DNL #49. Apparently it does not work properly on rational functions for some initial value of the variable (x=0 in my example), but it does if you change the expansion point. You may say that it is foolish to look for a rational approximation to a rational function, but I was just testing the program.

Best regards, Marcelo

__Johann Wiesenbauer__

Hi Marcelo,

Well, after a lot of debugging I know the problem with my pade.mth now (it has something to do with the built-in function TERMS, whose behaviour is sometimes unpredictable), but I don't know how to fix it without a decrease of performance.

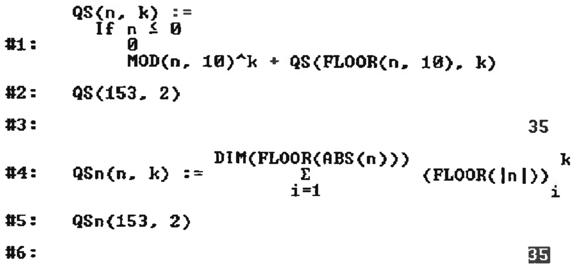
For the time being, I can offer you the following fix though, which is considerably slower, but on the other hand far more transparent. (Hence, there might be people out there who appreciate this version even more!)

(You can download the updated program pade() from our websites, Josef)

__Don Phillips__

Just got DNL #48 over the weekend. It was a great issue, as always! It was good to see my article in print. You did a wonderful job with it; I liked the used of the TI-89 screen shots to show that my programs came up with the same answers.

On page 24 there is a routine to calculate the sum of the kth powers of the integers of a number, QS(n,k). There is another way to do this.



```
       QS(n, k) :=
         If n ≤ 0
#1:         0
            MOD(n, 10)^k + QS(FLOOR(n, 10), k)

#2:    QS(153, 2)

#3:                                                      35
```

$$\#4:\quad QSn(n, k) := \sum_{i=1}^{DIM(FLOOR(ABS(n)))} (FLOOR(|n|))_i^k$$

```
#5:    QSn(153, 2)

#6:                                                      35
```

!!!Help!!! A mail from a German class arrived having a very dramatic title:

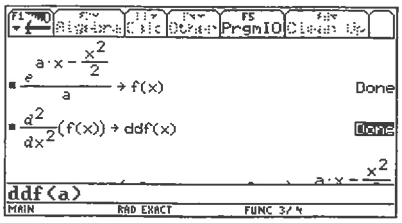
In our endexamination (Abitur, Matura, Baccalaureat) we investigated the family of functions $f_a(x)=\frac{1}{a}e^{-\frac{1}{2}x^2+ax}$. Using their TI-92PLUS all pupils found the correct second derivative

$f_a''(x)=\frac{(x^2-2ax+a^2-1)e^{-\frac{x^2}{2}+ax}}{a}$. Calculating the 2nd derivative for $x = a$ the expected result is

$f_a''(a)=\frac{-e^{\frac{a^2}{2}}}{a}$ which most of us (37 out of 42) obtained but the remaining 5 received

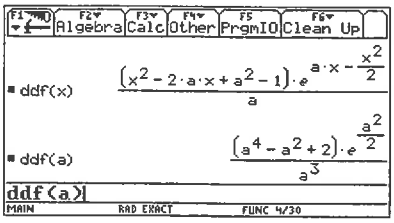
$f_a''(a)=\frac{(a^4-a^2+2)e^{\frac{a^2}{2}}}{a^3}$. How is this possible?

My answer was (supported by some screen shots of Thomas Himmelbauer):



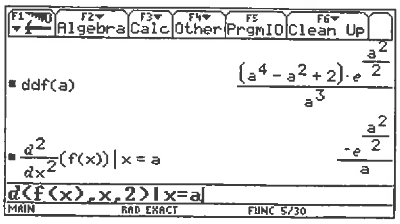
- $\frac{e^{a\cdot x-\frac{x^2}{2}}}{a}\to f(x)$ Done
- $\frac{d^2}{dx^2}(f(x))\to ddf(x)$ Done

```
ddf(a)
```



- $ddf(x)$ $\frac{(x^2-2\cdot a\cdot x+a^2-1)\cdot e^{a\cdot x-\frac{x^2}{2}}}{a}$
- $ddf(a)$ $\frac{(a^4-a^2+2)\cdot e^{\frac{a^2}{2}}}{a^3}$

```
ddf(a)
```

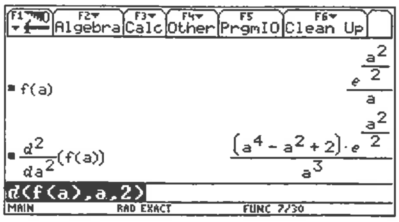


- $ddf(a)$ $\frac{(a^4-a^2+2)\cdot e^{\frac{a^2}{2}}}{a^3}$
- $\frac{d^2}{dx^2}(f(x))\,|\,x=a$ $\frac{-e^{\frac{a^2}{2}}}{a}$

```
d(f(x),x,2)|x=a
```

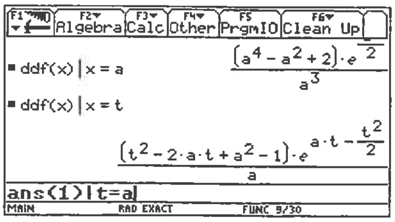
I recommend to differentiate *f*(*x*) twice and then substitute for *x* = *a*. But what is the reason for the "wrong" result? The answer is not so difficult to understand. Obviously the *TI* first evaluates *f*(*a*) and then differentiates with respect to *a*.

We will confirm this:



- $f(a)$ $\frac{e^{\frac{a^2}{2}}}{a}$
- $\frac{d^2}{da^2}(f(a))$ $\frac{(a^4-a^2+2)\cdot e^{\frac{a^2}{2}}}{a^3}$

```
d(f(a),a,2)
```



- $ddf(x)\,|\,x=a$ $\frac{(a^4-a^2+2)\cdot e^{\frac{a^2}{2}}}{a^3}$
- $ddf(x)\,|\,x=t$ $\frac{(t^2-2\cdot a\cdot t+a^2-1)\cdot e^{a\cdot t-\frac{t^2}{2}}}{a}$

```
ans(1)|t=a
```

We obtain the "wrong" result again predefining the 2nd derivative as you can see. `ddf(x)` seems to be ok, but substituting for *x* leads also to the not expected outcome.

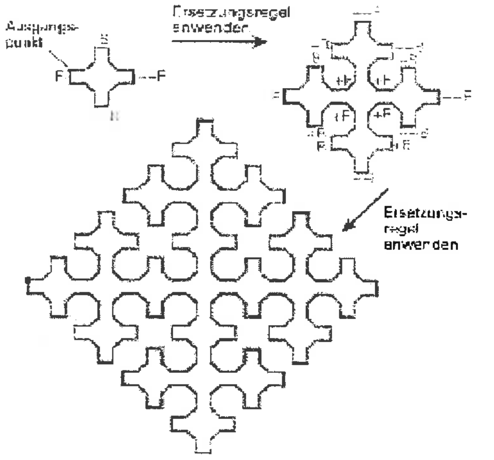
DERIVE does it well.

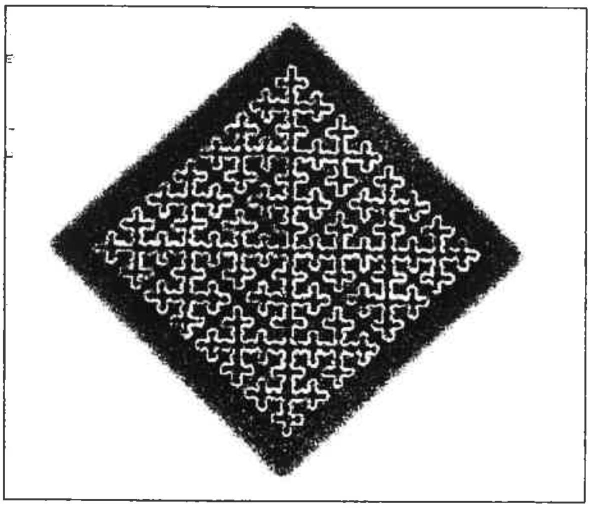
#1: $$f(x) := \frac{ê^{a\cdot x - x^2/2}}{a}$$

#2: $$[f''(x),\ f''(a)] = \left[\frac{ê^{a\cdot x - x^2/2}\cdot(x^2 - 2\cdot a\cdot x + a^2 - 1)}{a},\ -\frac{ê^{a^2/2}}{a}\right]$$

I sent an extended answer to the students but .... I never heard if it arrived!!??

In the Scientific American (German Edition) I found an interesting article on "Ethnomathematics". In detail the contribution was about south Indian "Kolams". One of them could be represented using the Lindenmayer System. Fortunately I worked out *DERIVE*-procedures for applying L-Systems (next *DNL*) and tried my program – it worked. (This a partial copy of the SA-article.)





As an introduction for L-systems I pose the following challenge:

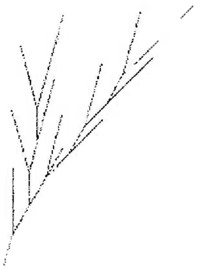
Transformation of a string according to the following rules must be performed:

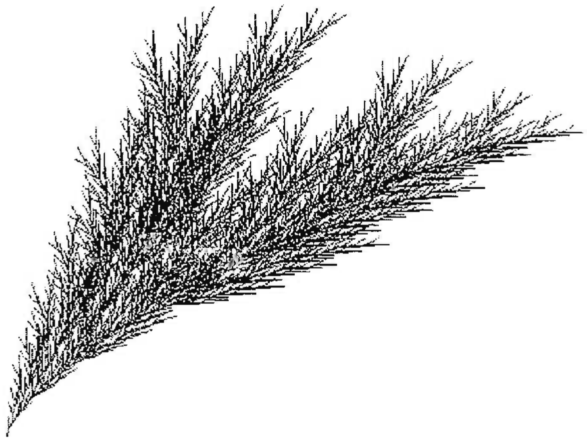
Given is an initial string (say `"ab"`)
`"a"` → `"b"`
`"b"` → `"ac"`
`"c"` → `"abc"`

Applying this rule five times we will have the following list of results:

```
[                          bac                           ]
[                         acbabc                         ]
[                     babcacbacabc                       ]
[               acbacabcbabcacbabcbacabc                 ]
[ babcacbabcbacabcacbacabcbabcacbacabcacbabcbacabc       ]
```

Write a program –*as generalized as possible* – for performing such transformations.





Titbits from Algebra and Number Theory (25)

by Johann Wiesenbauer, Vienna

You may have already read in the editorial that this issue of the DNL is a very special one, being number fifty. Let me take this opportunity to congratulate Josef on the occasion of this anniversary and wish him all the best for the next 50 issues of his beautiful journal, which has served for more than twelve years as a "communication center" for the Derive-community. I would also like to express my gratitude and my pride about the fact that this column – the twentyfifth by the way and hence also sort of special – is by far the longest DNL-series.

On this occasion, I have also scanned all my columns so far and old memories came back when I wrote the very first articles still using Derive 2.56 or something like that. Many routines that first appeared in this series are now included in NUMBER.MTH and COMBINAT.MTH, which I'm very proud of. On the other hand, many programs in older issues of the DNL are now totally obsolete from a programming point of view, as they were written in the old clumsy Derive-code before version 5. In some cases that's no great loss, but there are a few notable exceptions, at least in my opinion. One of them is my column about polynomial arithmetic in the DNL # 30. I think most of these routines are quite useful and deserve an update to Derive 5. This is exactly what I'm going to do in the following. It might be a good idea to have that paper (Titbits #13) at hand, if you can get hold of it, as I will refer occasionally to its contents.

The first two routines with which I started then could be used to reduce a given polynomial u with integer coefficients mod n. There is no need for a conversion though, as those functions are built-in in the meanwhile under the names POLY_MOD(u,n) and POLY_MODS(u,n). (The latter is used by the way, if you want the smallest absolute residues rather than the smallest nonnegative residues mod n.) Pretty much the same goes for the next function

```
                   d
polydeg(u, x) := x·—— LN(DENOMINATOR(FACTOR( lim   u, Trivial, x)))
                   dx                        x→1/x
```

which can be used to compute the degree of a polynomial u in the indeterminate x. Frankly, this jewel of programming is still one of the functions I'm very fond of, because it's both very tricky and incredibly fast. As you might know though, Albert Rich himself has written an equally powerful library function POLY_DEGREE(u,x) in the meanwhile, which has replaced the old obsolete version in MISC.MTH at last. What follows is his "masterpiece of logical deduction", as I called it at one time during a discussion on the Derive-forum, which gave birth to this really nice function.

```
POLY_DEGREE(u, x) :=
  Prog
    If u = 0
       RETURN -1
    If IDENTICAL?(u, x)
       RETURN 1
    If SUM?(u)
       RETURN MAX(VECTOR(POLY_DEGREE(v_, x), v_, TERMS(u)))
    If POWER?(u) ∨ PRODUCT?(u)
       Σ(POLY_DEGREE(v_↓1, x)·v_↓2, v_, FACTORS(u))
       0
```

In that discussion I also suggested a new function for the computation of coefficients of a polynomial, whose definition is also given here for the sake of completeness. (Believe it or not, the old function POLY_COEFF(u,x,n) in MISC.MTH computed those coefficients in the same way, as one would compute the coefficients of a taylor series of a function u, that is it didn't make any use of the polynomial form of u !)

```
                                      n
POLY_COEFF(u, x, n) := SUBST(QUOTIENT(u, x , x), x, 0)
```

As for the next function from my Titbits(13) , which computed the leading coefficient of a polynomial u in the indeterminate x, I only exchanged polydeg(u,x) by POLY_DEGREE(u,x) in the following definition:

$$\text{leadcoeff}(u,\ x) := \lim_{x\to\infty} \frac{u}{x^{\text{POLY\_DEGREE}(u,\ x)}}$$

It shares with POLY_DEGREE(u,x) (and polydeg(u,x) for that matter) the nice property that u must not necessarily be given in expanded form without any significant loss of performance, as you can see in the following computation:

$$\text{leadcoeff}((2\cdot x + 1)^{100},\ x) = 1267650600228229401496703205376$$

This function is often used when we want to transform a given nonzero polynomial u over $\mathbf{Z}_p$ into a monic polynomial by multiplication with a constant $c \neq 0$ mod p. Exactly this is done by the following routine:

```
polynorm(u, p, x) :=
  Prog
    u := POLY_MOD(u, p)
    If u = 0
       RETURN u
    POLY_MOD(INVERSE_MOD(leadcoeff(u, x), p)·u, p)
```

$$\text{polynorm}(3\cdot x^3 + 2\cdot x + 4,\ 5) = x^3 + 4\cdot x + 3$$

The next three functions are more or less self-explanatory. They can be described as the counterparts of the built-in functions QUOTIENT(u,v,x), REMAINDER(u,v,x) and POLY_GCD(u,v,x), but with one additional parameter p, which is supposed to be prime. The small, but crucial change is that all those operations are carried out mod p as far as the coefficients are concerned!

```
polyquot(u, v, p, x) :=
  Prog
    u := FACTOR(QUOTIENT(u, POLY_MOD(v, p), x), Trivial, x)
    POLY_MOD(INVERSE_MOD(DENOMINATOR(u), p)·NUMERATOR(u), p)

polyrem(u, v, p, x) :=
  Prog
    u := FACTOR(REMAINDER(u, POLY_MOD(v, p), x), Trivial, x)
    POLY_MOD(INVERSE_MOD(DENOMINATOR(u), p)·NUMERATOR(u), p)

polygcd(u, v, p, x, r_) :=
  Loop
    If POLY_MOD(v, p) = 0
       RETURN polynorm(u, p, x)
    r_ := polyrem(u, v, p, x)
    u := v
    v := r_
```

$$\text{polyquot}(x^3 + 1,\ 3\cdot x^2 + 3,\ 2) = x$$

$$\text{polyrem}(x^3 + 1,\ 3\cdot x^2 + 3,\ 2) = x + 1$$

$$\text{POLY\_MOD}(x\cdot(3\cdot x^2 + 3) + x + 1,\ 2) = x^3 + 1$$

$$\text{polygcd}(x^3 + 1,\ 3\cdot x^2 + 3,\ 2) = x + 1$$

Furthermore, we'll need occasionally for a polynomial u the inverse of u as well as powers of u, again all mod p, which are computed by the following routines:

```
polyinv(u, v, p, x, q_, r_, s_, u_ := 1, v_ := 0) :=
  Loop
    If v = 0
       RETURN IF(NUMBER?(u) ∧ u ≠ 0, polyquot(u_, u, p, x))
    q_ := polyquot(u, v, p, x)
    r_ := polyrem(u, v, p, x)
    s_ := POLY_MOD(u_ - q_·v_, p)
    u := v
    v := r_
    u_ := v_
    v_ := s_
```

```
          2          3
polyinv(x  + 1, 2·x  + 2·x + 2, 3) = 2·x
```

```
polypower(u, n, v, p, x, w_ := 1) :=
  Loop
    If n = 0
       RETURN w_
    If ODD?(n)
       w_ := polyrem(u·w_, v, p, x)
    u := polyrem(u·u, v, p, x)
    n := FLOOR(n, 2)
```

```
            2               100   3                2
polypower(x  + x + 1, 10     , x  + x + 1, 2) = x  + 1
```

A simple application of polypower( ) is the following routine, which determines for a given polynomial $u \in \mathbf{Z}_p[x]$ of positive degree d whether it is irreducible, i.e. not a product of two polynomials of smaller degree, or not. This is done by checking for all i=1,2,...,[d/2] the condition

$$\gcd(x^{p^i} - x, u) \neq 1$$

because according to theorems of the theory of finite fields, if u is reducible, i.e. has got a divisor of degree $\leq d/2$, this would cause one of these gcd's to be unequal to 1.

```
polyirr?(u, p, x, d_, i_ := 1, u_) :=
  Prog
    u := POLY_MOD(u, p)
    u_ := x
    d_ := POLY_DEGREE(u, x)
    If d_ = 0
       RETURN false
    Loop
      u_ := polypower(u_, p, u, p, x)
      If ¬ NUMBER?(polygcd(u, u_ - x, p, x))
         RETURN false
      i_ :+ 1
      If i_ > d_/2 exit
```

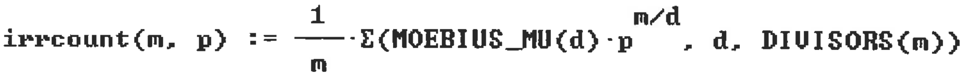
```
           4    3    2
polyirr?(x  + x  + x  + x + 1, 2) = true
```

As for irreducible polynomials, there are a number of interesting facts. In the first place, for every degree d there is always an irreducible polynomial $u \in \mathbf{Z}_p[x]$ of degree m. To be more precise, the number of monic irreducible polynomials of degree m in $\mathbf{Z}_p[x]$ is given by the formula

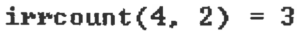
$$N_p(m) = \frac{1}{m}\sum_{d|m} \mu(d) p^{m/d}$$

where μ denotes the Moebius μ - function (cf. [1], p 155)

The Derive-implementation of this formula along with an example looks like this:

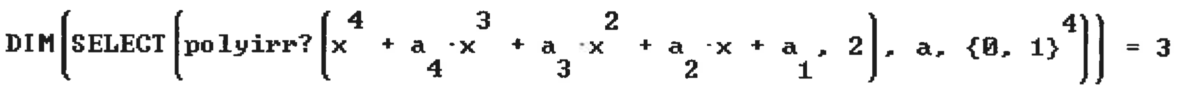


```
                     1                    m/d
irrcount(m, p) := ──·Σ(MOEBIUS_MU(d)·p     , d, DIVISORS(m))
                     m
```



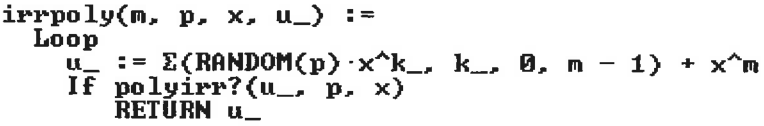
```
irrcount(4, 2) = 3
```

As this example is so small, we can afford it to check the outcome by the following brute-force computation:

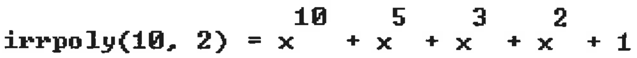


```
              ⎛ 4      3      2                ⎞               4⎞⎞
DIM⎛SELECT⎛polyirr?⎜x  + a ·x  + a ·x  + a ·x + a , 2⎟, a, {0, 1} ⎟⎟ = 3
              ⎝       4      3      2      1       ⎠               ⎠⎠
```

From the formula above the probability of a random monic polynomial of degree m in $\mathbf{Z}_p[x]$ being irreducible is at least 1/(2m) and roughly 1/m. Hence, if we want to find an irreducible polynomial in $\mathbf{Z}_p[x]$ of degree m by testing a number of random monic polynomials of degree m in $\mathbf{Z}_p[x]$, then we should expect about m failures. The following Derive-routine will do all this testing for you.

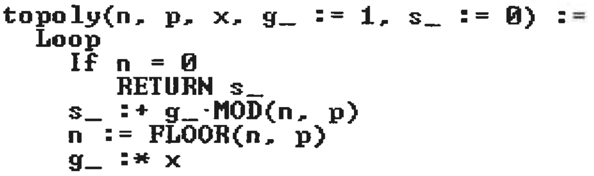


```
irrpoly(m, p, x, u_) :=
  Loop
    u_ := Σ(RANDOM(p)·x^k_, k_, 0, m - 1) + x^m
    If polyirr?(u_, p, x)
       RETURN u_
```

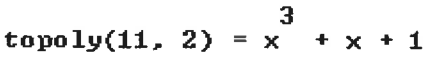


```
                      10    5    3    2
irrpoly(10, 2) = x   + x  + x  + x  + 1
```

If you studied my paper, which I recommended in the beginning, you will already know that every irreducible polynomial f(x) of degree m in $\mathbf{Z}_p[x]$, can be used to generate the (up to isomorphisms unique) field $\mathbf{F}_q$, where $q = p^m$, due to $\mathbf{F}_q \cong \mathbf{Z}_p[x]/(f(x))$. In order to make computations in such a field $\mathbf{F}_q$ easier, it might be a good idea to encode the numbers 0,1,2,..,q-1 as polynomials of degree < m representing the residue classes in that factor ring.

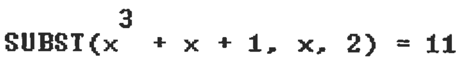


```
topoly(n, p, x, g_ := 1, s_ := 0) :=
  Loop
    If n = 0
       RETURN s_
    s_ :+ g_·MOD(n, p)
    n := FLOOR(n, p)
    g_ :* x
```



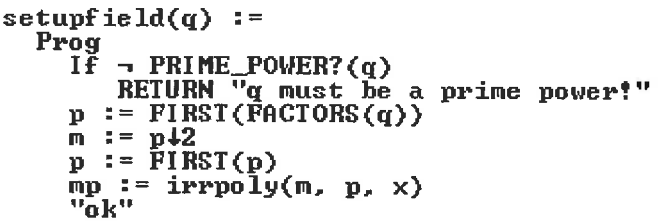
```
                      3
topoly(11, 2) = x   + x + 1
```

You might wonder, why I didn't care about the conversion in the other direction, i.e. from polynomials to numbers. Isn't it important enough? Of course, this is needed as well, but all we must do is to simply substitute p for x to get this kind of conversion:



```
         3
SUBST(x   + x + 1, x, 2) = 11
```

Okay, we are ready now to write sort of a "set up program" for the parameters of a field.



```
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 := irrpoly(m, p, x)
    "ok"
```

```
setupfield(25) = ok

p = 5

m = 2

          2
mp = x  + 2
```

Note that p, m and mp are used as global variables in the following and you have to consider them kind of "protected"! On the other hand, using these "standard parameters" you are allowed to omit them in the call of the routines dealing with the basic operations of a field, which we are going to define now.

```
plus(u, v, p) := SUBST(POLY_MOD(topoly(u, p, x) + topoly(v, p, x), p), x, p)
minus(u, v, p) := SUBST(POLY_MOD(topoly(u, p, x) - topoly(v, p, x), p), x, p)
times(u, v, mp, p) := SUBST(polyrem(topoly(u, p, x)·topoly(v, p, x), mp, p, x), x, p)
div(u, v, mp, p) := SUBST(polyrem(topoly(u, p, x)·polyinv(topoly(v, p, x), mp, p, x), mp, p, x), x, p)
```

Okay, you want to see the multiplication table of the above field with 25 elements in all its glory? Here you are !

```
VECTOR(VECTOR(times(a, b), a, 0, 24), b, 0, 24)
┌ 0   0   0   0   0   0   0   0   0   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  16  17  18  19  20  21  22  23  24 │
│ 0   2   4   1   3  10  12  14  11  13  20  22  24  21  23   5   7   9   6   8  15  17  19  16  18 │
│ 0   3   1   4   2  15  18  16  19  17   5   8   6   9   7  20  23  21  24  22  10  13  11  14  12 │
│ 0   4   3   2   1  20  24  23  22  21  15  19  18  17  16  10  14  13  12  11   5   9   8   7   6 │
│ 0   5  10  15  20   3   8  13  18  23   1   6  11  16  21   4   9  14  19  24   2   7  12  17  22 │
│ 0   6  12  18  24   8  14  15  21   2  11  17  23   4   5  19  20   1   7  13  22   3   9  10  16 │
│ 0   7  14  16  23  13  15  22   4   6  21   3   5  12  19   9  11  18  20   2  17  24   1   8  10 │
│ 0   8  11  19  22  18  21   4   7  10   6  14  17  20   3  24   2   5  13  16  12  15  23   1   9 │
│ 0   9  13  17  21  23   2   6  10  19  16  20   4   8  12  14  18  22   1   5   7  11  15  24   3 │
│ 0  10  20   5  15   1  11  21   6  16   2  12  22   7  17   3  13  23   8  18   4  14  24   9  19 │
│ 0  11  22   8  19   6  17   3  14  20  12  23   9  15   1  18   4  10  21   7  24   5  16   2  13 │
│ 0  12  24   6  18  11  23   5  17   4  22   9  16   3  10   8  15   2  14  21  19   1  13  20   7 │
│ 0  13  21   9  17  16   4  12  20   8   7  15   3  11  24  23   6  19   2  10  14  22   5  18   1 │
│ 0  14  23   7  16  21   5  19   3  12  17   1  10  24   8  13  22   6  15   4   9  18   2  11  20 │
│ 0  15   5  20  10   4  19   9  24  14   3  18   8  23  13   2  17   7  22  12   1  16   6  21  11 │
│ 0  16   7  23  14   9  20  11   2  18  13   4  15   6  22  17   8  24  10   1  21  12   3  19   5 │
│ 0  17   9  21  13  14   1  18   5  22  23  10   2  19   6   7  24  11   3  15  16   8  20  12   4 │
│ 0  18   6  24  12  19   7  20  13   1   8  21  14   2  15  22  10   3  16   9  11   4  17   5  23 │
│ 0  19   8  22  11  24  13   2  16   5  18   7  21  10   4  12   1  15   9  23   6  20  14   3  17 │
│ 0  20  15  10   5   2  22  17  12   7   4  24  19  14   9   1  21  16  11   6   3  23  18  13   8 │
│ 0  21  17  13   9   7   3  24  15  11  14   5   1  22  18  16  12   8   4  20  23  19  10   6   2 │
│ 0  22  19  11   8  12   9   1  23  15  24  16  13   5   2   6   3  20  17  14  18  10   7   4  21 │
│ 0  23  16  14   7  17  10   8   1  24   9   2  20  18  11  21  19  12   5   3  13   6   4  22  15 │
└ 0  24  18  12   6  22  16  10   9   3  19  13   7   1  20  11   5   4  23  17   8   2  21  15  14 ┘
```

(Note that the usual index row and index column can be seen in the second row and column, resp.)

Oh no, space is running out again! So I have to continue this topic in the next column. (Hope you don't consider this a threat!) And as always, if you have any comments or suggestions, please let me know! (j.wiesenbauer@tuwien.ac.at)

[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/)