THE DERIVE - NEWSLETTER #27

THE BULLETIN OF THE

DERIVE®

USER GROUP

+ TI 92

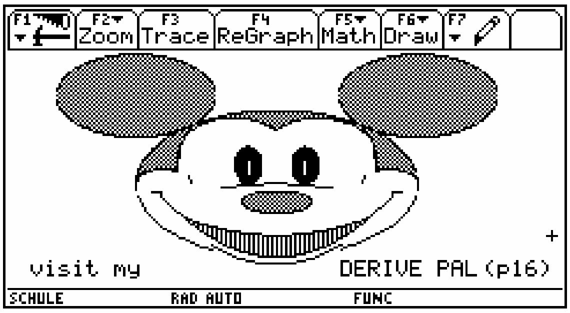
Contents:

revised version 2011 **September 1997**

Learning Mathematics through the *TI-92*
J.S. Berry, E.Graham, A.J. Watkins

This book develops the foundation mathematics needed by scientists and engineers through the use of the Texas Instruments *TI-92*. The authors emphasise its use as an investigative tool to introduce, and help students to understand, basic concepts in mathematics, and as a problem solving tool for solving real problems from the world of science and engineering.

The book is written primarily for students who have not studied mathematics in depth at school, and will prepare them for a foundation course in science or engineering. However, the book also provides an introduction to the use of the *TI-92* for those students who are already familiar with scientific functions and calculus.
L.M. provides an exciting and unique opportunity for students to learn and use mathematics in a refreshing way. The experience of the authors in teaching engineering students at Plymouth shows how their approach provides the understanding and confidence to succeed in learning mathematics.
ISBN 0-86238-489-3, 367 pages, Chartwell-Bratt



Great *TI-92* Programs - Vol 1
Bernhard Kutzler & David R. Stoutemyer

Includes Version 2.0 diskette covering:
Interactive lessons for learning how to operate the *TI-92*, Useful online help with instructive examples.

Solving ODEs and systems of (non-) linear equations,
Implicit plotting,
Turtle Graphics,
Unit Conversion,
PreCalculus Mathematics,
Integration Applications,
and more.

You need Graph LINK to use this book.
ISBN 3-901769-00-5, 64 pages, bk teachware

Nonlinear Physics with *MAPLE*
R.H.Enns & G.C.McGuire

Nonlinear Physics is at the cutting edge of research and applications in fields as diverse as medicine, mathematics, biology and chemistry, computer science, engineering, electronics, and physics, to name just several.
The text represents an introductory survey of the basic concepts and applied mathematical methods of nonlinear science. It is accompanied by a laboratory manual containing 28 activities from Spin Toy Pendulum to Mapping a forced nonlinear oscillator.
Extensive use is made of a CAS (*MAPLE V*). For an experienced *DERIVE* user it should not be too difficult to adjust the proposed techniques.
ISBN 3-7643-3977-2, Birkhäuser

A Bundle of TI-92 books: orders to: jglynn@mathware.com

Discovering Math on the *TI-92*	Ch. Brueningsen and others
Geometric Explorations with the *TI-92*	M. Keyton
Geometric Investigations for the Classroom	Ch. Von der Embse, A. Engebretsen
Graphing Calculators: Quick & Easy for the *TI-92*	D. Lawrence
Introduction to the *TI-92*: 37 Experiments	E. Andersen, Ch. Lund
Investigating Advanced Algebra with the *TI-92*	Dr. Brendan Kelly
Investigating Calculus with the *TI-92*	Dr. Brendan Kelly
Investigating Statistics with the *TI-92*	Dr. Brendan Kelly
Learning Programming with the *TI-92*	W.Ellis, E.Lodi, St. Blasberg
Skill & Practice Masters in Algebra *TI-92*	D.Lawrence

The **Second US DERIVE User Group Meeting** will take place in the frame of the ICTCM ´97 Chicago on Sunday, November 9 morning. Details will be announced during the Conference. I am looking forward to meeting some of you.

Visit the Mickey Mouse DERIVE pal on page 21!

A DERIVE-party - from my point of view

My parents were the local organisers of the DERIVE Symposium ´97, which took place in the beginning of August in Sweden. Even though I´m a non-DERIVER, I had great fun all the time. In fact, I think everyone enjoyed themselves, even between and after the lectures. When the day´s work was finished, the party began. One evening, there was a party with loads of rules. One rule was that at least someone from each country had to make a speech during the dinner. Another was that it was absolutely forbidden to sit next to your husband or wife. And if you didn´t want to make a speech, you had to sing a song. It seemed like Bernhard Kutzler knew the Swedish law extremely well...

The last evening was maybe the nicest of them all. All the delegates came to our home for one last party, to celebrate the successful symposium. We had a nice dinner in our garden, with Swedish meatballs and potato-salad, and after the meal, we enjoyed some wonderful songs. Bernhard Kutzler and Helmut Heugl performed a great DERIVE-version of Frank Sinatra´s "My Way", and then, Josef Böhm, sang some kind of Austrian song, with help from Helmut. Bernhard translated the verses into English, and the rest of us tried to join in when that was possible.
After that, Sergey Biryukov sang two Russian songs, and Tomass Romanovskiss was persuaded to sing something in Latvian, and that was so nice to hear. The German and Austrian gang sang a few lovely songs like a professional choir. When it was time to say goodbye, goodnight and "see you in Gettysburg...", we all went out in the garden again, took each others hands and another beautiful song was performed. Then it was over, and everyone was gone.
This were, from my point of view, the best days of the summer. Thank you so much for coming, and making this symposium a party. At last, here are the words to one of the two great DERIVE-songs that Bernhard Kutzler wrote. If you wish to know the music to it or complain in any other way, you have to contact him.

The DERIVE-party, the DERIVE-party
Is now over, is now over
It was successful, it was successful
Thanks to David and Yvonne

Even though everything was great during the DERIVE-week, the sight of Terence Etchells in my dad´s red swimming trunks, was absolutely the best ever...
Josefine Sjöstrand, Onsala, Sweden

Find here the scientific program - in order of appearence. I´ll try to produce one special DNL containing some lectures of the DERIVE Symposium 1997, FUN in Teaching Mathematics.

David R.Stoutemyer (USA)	A brief history of portable computer algebra
Terence Etchells (UK)	To Booleanly go where no Math has gone before!
Tomass Romanovskis (LAT)	Kepler´s Equation? Why not? With *DERIVE*!
Wolfgang Pröpper (GER)	The *TI-92* as a Medium in Math Classes
Otto Wurnig (AUT)	Normal distributions with CAS
Tatyana Oleinik (UKR)	A mathematical course with *DERIVE* on economical problems
Helmut Heugl (AUT)	The influence of the CAS on the path of the students "into" trigonometry
Barbara Leitherer (USA)	More exciting Trig with the *TI-92*
David Sjöstrand (SWE)	Computer Geometry and Computer Algebra
Vladimir Rovenskiy (RUS)	Teaching Geometry of Curves and Surfaces with CAS
Johann Wiesenbauer (AUT)	On the Fascination of Primes and its Use in Classroom Teaching
Carl Leinbach (USA)	Why hold Back the Good Stuff?
Sergey Biryukow (RUS)	FROM FUN TO JOY
Bernhard Kutzler (AUT)	Teaching Maths in the Computer Age
Halvor Devold (NOR)	The changing Face of Mathematics
Matthias Kawski (USA)	CAS in Vector Calculus: A radically new approach based on visualization
Detlev Kirmse (GER)	A holistic experimental approach to classic functions in middle schools
Robert Hill (USA)	Mathematical encounters of the everyday kind
Josef Böhm (AUT)	From hard work to much FUN & Optimization - A Window Shuttle

The *DERIVE-NEWSLETTER* is the Bulletin of the *DERIVE User Group*. It is published at least four times a year with a contents of 40 pages minimum. The goals of the *DNL* are to enable the exchange of experiences made with *DERIVE* as well as to create a group to discuss the possibilities of new methodical and didactical manners in teaching mathematics.
We include now a section dealing with the use of the *TI-92* and we try to combine these modern technologies.

Editor: Mag. Josef Böhm
A-3042 Würmla
D´Lust 1
Austria
Phone/FAX: 43-(0)660 3136 365
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 Newsletter* will be.

Next issue: December 1997
Deadline: 15 November 1997

Preview: Contributions for the next issues

3D-Geometry, Reichel, AUT
Algebra at A-Level, Goldstein, UK
Graphic Integration, Linear Programming, Various Projections. Böhm, AUT
A Utility file for complex dynamic systems, Lechner, AUT
Examples for Statistics, Roeloffs, NL
Solving Word problems (Textaufgaben) with DERIVE, Böhm, AUT
About the "Cesaro Glove-Osculant", Halprin, AUS
Hidden lines, Weller, GER
Fractals and other Graphics, Koth, AUT
Experimenting with GRAM-SCHMIDT, Schonefeld, USA
Implicit Multivalue Bivariate Function 3D Plots, Biryukov, RUS
Parallel Curves, Wunderling, GER
Quaternion Algebra, Sirota, RUS
150 Years of π´s 250 decimal places, Romanovskis, LAT

The TI-92 Section, Waits a.o.

and
Setif, FRA; Vermeylen, BEL; Leinbach, USA; Halprin, AUS; Speck, NZL; Weth, GER; Wiesenbauer, AUT; Aue, GER; Pröpper, GER; Koller, AUT; Mitic, UK; Tortosa, ESP; Santonja, ESP;Schorn, GER and ....

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

Mag. Martin Mayr, Innsbruck, AUT

Working with *DERIVE* (DOS version 2.57 and *DfW* 4.02) I came across the following bug:

$$\int z^{-\frac{v+1}{2}}\,dz = \frac{2\left(z^{\frac{1-v}{2}}-1\right)}{1-v} \qquad \text{but it should be:} \quad = 2\,\frac{z^{\frac{1-v}{2}}}{1-v}.$$

DNL:

If you decompose the result you will obtain:

$$= \frac{2\,z^{\frac{1-v}{2}}}{1-v} + \underbrace{\frac{2}{v-1}}_{\text{Integration Constant C}}$$

This is the DERIVE 6 result:

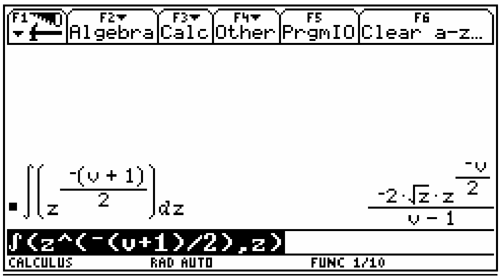
$$\int z^{-(v+1)/2}\,dz = \frac{2\cdot z^{(1-v)/2}}{1-v} + \frac{2}{v-1}$$

It is a good experience for students that sometimes the DERIVE- or TI-expressions will not meet their expectations. Then they have to check the equivalence of their "handmade" results with the CAS-results.
Calculating the defined integral you will find the constant C reduced and the correct result.:

$$\int z^{-(v+1)/2}\,dz = \frac{2\cdot\left(z^{(1-v)/2}-1\right)}{1-v}$$

$$\int_a^b z^{-(1+v)/2}\,dz = \frac{2\cdot\left(a^{(1-v)/2}-b^{(1-v)/2}\right)}{v-1}$$

As you can see the TI-92 shows the result in the expected form. (But here also the "average" student will not see the equivalence at his first view).



Troels Ring, Denmark ring@post1.tele.dk

Dear friends of *DERIVE*. I´m struggling with expressions towards the mean and was reading the paper by Davis (AM J Epidemiol 1976; 104:493-98) and trying to reproduce the theoretical results in *DERIVE*.

First the variables are declared:

```
#1:   "NORMDIST.MTH"                         User
#2:   σ :ε Real (0, ∞)                       User
#3:   μ :ε Real                              User
#4:   a :ε Real                              User
```

Then the normal distribution is defined

$$\#5:\quad \int_{-\infty}^{\infty} \frac{1}{\sqrt{(2\cdot\pi\cdot\sigma^2)}}\cdot\hat{e}^{-1/2\cdot((x-\mu)/\sigma)^2}\,dx \qquad \text{User}$$

and its integrated probability is estimated correctly:

#6: 1 Simp(#5)

and so is its mean expectation:

$$\#7:\quad \int_{-\infty}^{\infty} \frac{x\cdot 1}{\sqrt{(2\cdot\pi\cdot\sigma^2)}}\cdot \hat{e}^{-1/2\cdot((x-\mu)/\sigma)^2}\,dx = \mu \qquad \text{User=Simp(User)}$$

Then I tried to substitute for μ = 70:

$$\#8:\quad \int_{-\infty}^{\infty} \frac{x\cdot 1}{\sqrt{(2\cdot\pi\cdot\sigma^2)}}\cdot \hat{e}^{-1/2\cdot((x-70)/\sigma)^2}\,dx = 70 \qquad \text{User=Simp(User)}$$

and got μ = 70 back as expected mean without, of course, the need for knowing σ. But when I happened to know σ - and substituted it:

$$\#9:\quad \int_{-\infty}^{\infty} \frac{x\cdot 1}{\sqrt{(2\cdot\pi\cdot 5^2)}}\cdot \hat{e}^{-1/2\cdot((x-70)/5)^2}\,dx \qquad \text{User}$$

I got an improper estimate of the expected mean:

#10: 3.78815 Approx(User)

which depended on σ:

$$\#11:\quad \int_{-\infty}^{\infty} \frac{x\cdot 1}{\sqrt{(2\cdot\pi\cdot 10^2)}}\cdot \hat{e}^{-1/2\cdot((x-70)/10)^2}\,dx = 12.775 \qquad \text{User=Approx(User)}$$

Because of this deficiency it was not possible to estimate the expectations for truncated normal variates. How comes that expectation which doesn´t need to know σ is affected by it anyway? Troels Ring

DNL: *See the following DERIVE results:*

$$\#14:\quad \int_{-\infty}^{\infty} \frac{x\cdot 1}{\sqrt{(2\cdot\pi\cdot 5^2)}}\cdot \hat{e}^{-1/2\cdot((x-70)/5)^2}\,dx = 70 \qquad \text{User=Simp(User)}$$

μ

$$\#15:\quad \int_{-\infty}^{\infty} \frac{x\cdot 1}{\sqrt{(2\cdot\pi\cdot 10^2)}}\cdot \hat{e}^{-1/2\cdot((x-70)/10)^2}\,dx = 70 \qquad \text{User=Simp(User)}$$

and read Terence Etchells´ answer. (He is our specialist for numerical affairs). At this occasion: Our best congratulations to Terence for extending his familiy with a second daughter Suzanne.

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

Hi All
The Normal distribution problem posted by Troels Ring can be explained thus.

Troels uses Approx to evaluate an integral with parameters. *DERIVE*´s integral appoximator can only work on definite integrals of a single variable i.e. INT(x^2,x,0,1) not integrals with parameters like INT(a*x^2,x,0,1). So *DERIVE* sees that there is more than one variable in the expression and works exactly (i.e. attempts to find the integrand symbolically) even though the user has specified Approx (i.e. use the methods of Gauss, Romberg etc.).

In the last example given by Troels, there are no parameters in the integrand hence the quadrature methods coded into *DERIVE* snap into action. I used *DERIVE* for Windows Version 4.05 and got a Dubious Accuracy message and no result. Clearly, the limits **-inf** and **inf** have posed a problem for the quadrature methods in *DERIVE* for this integral; it realises (what ever that means!) that the accuracy of the result cannot be relied on. Earlier DOS versions have bleeped Dubious Accuracy warnings but have still given a result or certain problematic integrals. The answer, as in the case of Troels´ example, is not to be relied on! I suspect that Dubious Accuracy warnings were being bleeped as *DERIVE* approximated the integral. I am right???

Cheers

Bengt Mansson, Sweden

Does anyone know of *DERIVE* packages (.MTH - files) for calculation of Galois Groups?
===============
bengtmn@algonet.se
http://www.algonet.se/rbengtmn/www/www.htm

Howard Whitson, USA **WHITSTON@ltu.edu**

As I was working on examples of Floyd´s Algorithm for my Data Structures Class (Floyd´s Algorithm deals with the shortes path between two given vertices using matrices), I got wondering if I could make *DERIVE* do the individual matrices for the algorithm. Now my programming with *DERIVE* is a bit weak (read -- next to non-existing), but i am willing to try if someone can point the way for me (or better yet, someone has already done it and one of you could point me in the direction of the source).
Thank you for your assistance in this matter.
H.W. Lawrence Tech. University

DNL: *Leonardo Volpi gave an answer in the DERIVE News Group. He also sent a DOC-file with an extended explanation. Many thanks. If anyone is interested then please contact Leonardo or me.*

Leonardo Volpi, Firenze, Italy **VOLPI_LEONARDO@mailbox.enel.it**

A graph comprises a set of vertices V1, V2 ...... Vn and a set of edges connecting vertices. It can be modeled as an adjacency matrix A, in which each element A_{ij} is the length connect from i to j vertices. Floyd´s algorithm takes as input a $(n \times n)$ matrix A, modeling a given graph, and computes an $(n \times n)$ matrix S with S_{ij} the length of the shortest path from Vi to Vj or an infinite value ∞ if there is no path.

The sequential Floyd´s procedure is:

$$S_{ij}(0) = \begin{cases} A_{ij} & \text{if } A_{ij} \neq 0 \\ 0 & \text{if } A_{ij} = 0 \text{ and } i = j \\ \infty & \text{if } A_{ij} = 0 \text{ and } i \neq j \end{cases}$$

for $k = 1..n$

for $i = 1 .. n, \ \ j .. n$

$$S_{ij}(k+1) = \min(S_{ij}(k), S_{ik}(k) + S_{kj}(k))$$

This algorithm derives the matrix S in n steps. At each step k, it constructs an intermediate matrix $S(k)$ containing the best known shortest distance between each pair of nodes. Note that, generally speaking, the distance between i and j may be different from the one between j and i (non symmetrical graph).

In *DERIVE* I have developed and tested the following function:

```
#1:   FLOYD.MTH

      FLOYD_INIT(a) := VECTOR(VECTOR(IF(i = j, 0, IF(a       = 0, ∞, a      )), j, 1, DIMENSION(a)),
#2:                                                [i, j]          [i, j]
        i, 1, DIMENSION(a))
```

`FLOYD_INIT` produces the $S_{ij}(0)$ matrix

```
      FLOYD_STEP(a, k) := VECTOR(VECTOR(MIN(a      , a      + a      ), j, 1, DIMENSION(a)), i, 1,
#3:                                         [i, j]   [i, k]   [k, j]
        DIMENSION(a))
```

`FLOYD_STEP` produces the intermediate matrix at step k from the intermededeiate k–1 matrix A

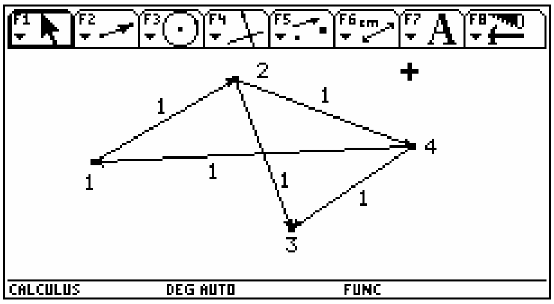
```
      FLOYD_SEQ_AUX(s, k) :=
        If k > DIMENSION(s)
#4:        s
           FLOYD_SEQ_AUX(FLOYD_STEP(s, k), k + 1)

#5:   FLOYD_SEQ(a) := FLOYD_SEQ_AUX(FLOYD_INIT(a), 1)
```

Finally `FLOYD_SEQ` automates the whole process and gives the final result matrix from the initial graph matrix A.

Repeating the test with larger matrices (on Pentium 75 and *DERIVE* edu version) we get the following results:

Matrix	Average Computation Time
15 × 15	1.6 sec
18 × 18	3.0 sec
20 × 20	3.6 sec
25 × 25	7.2 sec

DNL: *Leonardo adds an example with a given matrix A. I try to illustrate this matrix with the graph belonging to that matrix. (I do that on the TI-92, of course).*



$$\#6: \quad a := \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}$$

$$\#7: \quad \text{FLOYD\_INIT}(a) = \begin{bmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 1 & 1 \\ \infty & \infty & 0 & \infty \\ 1 & \infty & 1 & 0 \end{bmatrix}$$

It is easy to verify the solution.

$$\#8: \quad \text{FLOYD\_SEQ}(a) = \begin{bmatrix} 0 & 1 & 2 & 2 \\ 2 & 0 & 1 & 1 \\ \infty & \infty & 0 & \infty \\ 1 & 2 & 1 & 0 \end{bmatrix}$$

<u>Klaus Herdt, Osnabrück, Germany</u> **herdt@fem04.mb.fh-osnabrueck.de**

Does anybody know a possibility to compute within a *DERIVE* program an eigenvector of a matrix A when the eigenvalue r is given?
EXACT_EIGENVECTOR(..) - from the Utility file VECTORS.MTH - does not solve the problem because it leads (mathematically correct) to a subspace of A with respect to r.
Also EXACT_EIGENVECTOR(..)/ABS(EXACT_EIGENVECTOR(..)) does not help because it contains a system variable @nn, which always changes.

<u>DNL:</u> *As I am only a Secondary School teacher I have no problems with eigenvectors - because I don´t need them, I hardly remember what they might be good for. So I intended to ask "eigenvalue specialists" among the DERIVE family – Carl Leinbach and Robert Hill – for help. But then I faced the challenge and tried to find a solution for the problem:*

Load first the Utility file VECTORS.MTH

```
"SPEIV.MTH"

a:=[[-1,-3,-3],[0,2,-1],[2,2,5]]

EIGENVALUES(a)=[w=1,w=2,w=3]

EXACT_EIGENVECTOR(a,3)=[[x1=0,x2=@1,x3=-@1]]

AUX(a,e):=VECTOR(RHS((EXACT_EIGENVECTOR(a,e)) SUB [1,i]),i,1,DIMENSION(a))

COEFF(u_):=IF(DIMENSION(u_)>1,u_ SUB 2*z_,IF(u_>=0,u_ SUB 1,u_ SUB 1,z_))

SPEIV(a,e,z):=LIM(VECTOR(COEFF((FACTORS(AUX(a,e))) SUB i_),i_,1,
               DIMENSION(a)),z_,z)
```

```
AUX(a,3)=[0,@11,-@12]

SPEIV(a,3,2)=[0,2,-2]

SPEIV(a,3,-0.5)=[0,-1/2,1/2]

EXACT_EIGENVECTOR(a,1)=[[x1=@31,x2=-@31/3,x3=-@31/3]]

SPEIV(a,1,4)=[4,-4/3,-4/3]

EXACT_EIGENVECTOR(a,2)=[[x1=@41,x2=-@41,x3=0]]

SPEIV(a,2,10)=[10,-10,0]
```

The trick is first to apply AUX() producing a vector consisting of the eigenvector´s components. These components are factorized with FACTORS.

Try FACTORS(AUX(a,3))!!

*The components of FACTORS()can appear in three different forms: [@n], [Number] or [@n, Number]. COEFF(u_) converts [@n] to z_, [Number] to Number and [@n,Number] to Number * z_. So COEFF([0]) → 0, COEFF([@10]) → z_ and COEFF([@10,1/3]) gives z_/3. Finally using SPEIV(..) the variable z_ in [0,z_,1/3*z_], will be substituted by z.*

Things have changed since 1997:The result of EIGENVALUES is a vector of the values and EXACT_EIGENVECTOR gives a vector.

#1: $A := \begin{bmatrix} -1 & -3 & -3 \\ 0 & 2 & -1 \\ 2 & 2 & 5 \end{bmatrix}$

#2: EIGENVALUES(A) = [1, 2, 3]

#3: $\text{EXACT\_EIGENVECTOR}(A,\ 3) = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}$

#4: $\dfrac{\text{EXACT\_EIGENVECTOR}(A,\ 3)}{|\text{EXACT\_EIGENVECTOR}(A,\ 3)|} = \begin{bmatrix} 0 \\ \dfrac{\sqrt{2}}{2} \\ -\dfrac{\sqrt{2}}{2} \end{bmatrix}$

#5: $\text{EXACT\_EIGENVECTOR}(A,\ 1) = \begin{bmatrix} 3 \\ -1 \\ -1 \end{bmatrix}$

Reinhard Schaeck, Berlin, Germany schaeck@ibm.net

I am used to write programs in several languages (PASCAL, JAVA, ReXX, C etc.) and I have problems to write a simple function with *DERIVE*´s capabilities, for example:

Let n be an integer and s(n) the sum of its digits

Is there any book about *DERIVE* focussing on defining similar functions?

Terence Etchells, Liverpool, UK T.A.ETCHELLS@livjm.ac.uk

I am not aware of a book that discusses this. You may try Bernhard Kutzler´s book Improving Mathematics Teaching with *DERIVE*. Contact Philip Yorke at Chartwell Bratt for info: philip@chartwel.demon.co.uk

However, I have a solution for this problem. I enjoyed solving it. We use the ITERATES function to construct a vector of digits of n and sum them. The first thing we need is how many digits there are.

```
FLOOR(LOG(n,10))
```

will do this.

An efficient method of extracting the units digits is divide by 10; find the FLOOR, multiply by 10 and subtract this from the original number e.g. 1243 / 10 = 123.4, the FLOOR is 123, multiply by 10 = 1230, substract it from the original 1234 – 1230 = 4. This is automated with

```
n - 10 * FLOOR(n/10)
```

We now iterate this procedure using a 2-dimensioned vector of extracted units and the number with the unit extracted e.g.

```
[[1234,1234],[123,4],[12,3],[1,2],[0,1]]
ITERATES([FLOOR(p/10),p*10*FLOOR(p/10)],[p,q],[n,n],FLOOR(LOG(n,10))+1)
```

putting 1234 into this function for n gives

$$\begin{bmatrix} 1234 & 1234 \\ 123 & 4 \\ 12 & 3 \\ 1 & 2 \\ 0 & 1 \end{bmatrix}$$

To extract the second column transpose the matrix (using `) and extract the second row with SUB 2 (or ELEMENT( ) for very old version *DERIVE* users).

Use the SUM function to add the elements of this vector and subtract the original number. There you have it:

```
S(n):=SUM(ITERATES([FLOOR(p/10),p-10*FLOOR(p/10)],[p,q],[n,n],
      FLOOR(LOG(n,10))+1)` SUB 2)-n
S(1234)
10
```

Good Luck!

Reinhard Schaeck

First of all thanks for your answer an solution. It gave me some ideas for testing and the chance to plunge a little deeper into the ITERATES function. I found another solution, and just in case you´re still interested I append it here:

```
S1(n):=SUM(VECTOR(FLOOR(n/10^j)-10*FLOOR(n/10^(j+1)),j,0,
       FLOOR(LOG(n,10))+1))
S1(1234)=10
```

This one (if right) works without the ITERATES function and therefore seems to be a little bit easier. For a "conventional" programmer functions like ITERATE are really overwhelming.

Terence Etchells

It is right and is a much more elegant solution than mine.

However, if you use your function to evaluate S1(9^9^3) it takes 29.2 seonds to compute the answer of 3249.

My function takes 0.6 seconds to compute the same answer (using *DERIVE* for Windows, Win95, Pentium 75mhz, 16MB RAM)

Quite a speed up in performance, wouldn´t you say! Although the ITERATE function is a little daunting it is a vehicle for minimising repetition of calculation. Basically your function keeps working on the large number 9^9^3 whereas the ITERATES function works on successively smaller numbers (divisions of 10 each time).

See some tests done by the editor:

9^(9^3)=439328503696464329829774782657072712058010308177126710621676697750466344744764029773301412612482563729435064854354299095570379503452515853238520182740967398746503532324400000659505126023955913142968176998364877699089666171297275956245407453033190168644894850576346492691458695174281789557994923607783461486426448617667076393901104477324982631297641034277093818692823488603426279473674368943609268871793467206677285688478458498235002859256706389043030847945506577080623430066283504397583789044245429585982964571774605868466160379567432725704121260940939343217905975847365096315872153240969882363435363449775254393010368267343970426230801390250903399147001650831878665172798468587509747439118815689

S(9^(9^3))

"Memory full with Classic DERIVE 3.* (for DOS)"

S(9^(9^3))=3249

"0.3 sec with DERIVE XM"

S1(9^(9^3))=3249

"27.5 sec with Classic DERIVE 3.*"

These times are from 1997. S needs 0.031 sec and S1 needs now 0.84 sec.

Wim de Jong, London, UK W.Dejong@uel..ac.uk

I found Terence´s *DERIVE* programme for the sum of the digits in a nonnegative integer interesting and amusing. For teaching purposes it may be a little too complicated. The following variation on Terence´s theme is less concise, but perhaps somewhat easier to follow. For an integer a, it produces the number N(a) of digits in a, the digit D(a,k) corresponding to 10^(k-1) and the sum S2(a) of the digits.

```
V(a):=ITERATES(FLOOR(x/10),x,a)
N(a):=DIMENSION(V(a))-2
D(a,k):=ELEMENT(V(a),k)-10*ELEMENT(V(a),k+1)
S2(a):=SUM(D(a,k),k,1,N(a))
S2(1234)=10
S2(9^(9^3))=3249
```

S2 needs now 0.064 sec. With DERIVE 6 we have one more – very fast - way to find the sum of the digits applying a string operation:

#1: $SD(n) := \Sigma(NAME\_TO\_CODES(n)) - 48 \cdot DIM(NAME\_TO\_CODES(n))$

#2: $SD\left(9^{9^{3}}\right) = 3249$

SD needs 0.000 sec.

Christof Scheper earl@wohnheim.wad.org

I have to find out the inverse of a matrix but I need the steps between (similar to the method when calculating without the computer) e.g:

$$\begin{pmatrix} 2 & 3 & 1 & 1 & 0 & 0 \\ 1 & 2 & -1 & 0 & 1 & 0 \\ 2 & 1 & 2 & 0 & 0 & 1 \end{pmatrix} \text{ to (step one) } \begin{pmatrix} 1 & 2 & -1 & 0 & 1 & 0 \\ 0 & -1 & 3 & 1 & -2 & 0 \\ 0 & -3 & 4 & 0 & -2 & 1 \end{pmatrix}$$

I know it can be done with a combination of SCALE_ELEMENT() and FORCE() from VECTOR.MTH but that is not very fast and easy. The command PIVOT() is no great help, too. I could do it manually in the same time .... Any comments welcome. Regards Christof

Wim de Jong, London, UK W.Dejong@uel..ac.uk

Christof, the quick way for your example:

(i) enter the expression a:=[2,3,1;1,2,-1;2,1,2]

(ii) enter ROW_REDUCE(a,identity_matrix(3)) = (include the equality operator = if you want the answer [id(3),a^-1] directly

$$\#3:\quad ROW\_REDUCE(a, IDENTITY\_MATRIX(3)) = \begin{bmatrix} 1 & 0 & 0 & -1 & 1 & 1 \\ 0 & 1 & 0 & \frac{4}{5} & -\frac{2}{5} & -\frac{3}{5} \\ 0 & 0 & 1 & \frac{3}{5} & -\frac{4}{5} & -\frac{1}{5} \end{bmatrix}$$

The following is the slow method. It is educationally preferable if you want students to practise elementary row operations in an interactive way:

(i) load the Utility file VECTOR.MTH (not necessary with recent DERIVE Versions!)
(ii) enter the augmented matrix [[2,3,1,1,0,0], [1,2,-1,0,1,0], [2,1,2,0,0,1]] as #1, say
(iii) enter SWAP_ELEMENTS(#1,1,2) and Simplify to obtain the resulting matrix as #3
(iv) enter SUBTRACT_ELEMENTS(#3,2,1,2) and Simplify to obtain the resulting matrix as #5
(v) enter SUBTRACT_ELEMENTS(#5,3,1,2) and Simplify to obtain what you have got in your note (as #7)
(vi) enter and Simplify SCALE_ELEMENT(#7,2,-1)

and so on......
Your next two steps being SUBTRACT_ELEMENTS(#9,1,2,2) and
SUBTRACT_ELEMENTS(#11,3,2,-3).
After that at most three more steps elementary row operations will be needed to get the required result.

$$\mathrm{SUBTRACT\_ELEMENTS}\left(\begin{bmatrix} 1 & 0 & 5 & 2 & -3 & 0 \\ 0 & 1 & 0 & \frac{4}{5} & -\frac{2}{5} & -\frac{3}{5} \\ 0 & 0 & 1 & \frac{3}{5} & -\frac{4}{5} & -\frac{1}{5} \end{bmatrix}, 1, 3, 5\right)$$

$$\begin{bmatrix} 1 & 0 & 0 & -1 & 1 & 1 \\ 0 & 1 & 0 & \frac{4}{5} & -\frac{2}{5} & -\frac{3}{5} \\ 0 & 0 & 1 & \frac{3}{5} & -\frac{4}{5} & -\frac{1}{5} \end{bmatrix}$$

DNL: *See another application of SUBTRACT_ELEMENTS in DNL#25, page 43.*

Please excuse a mistake. In the last DNL I had a certain Barbara Vincent in the listing of the Gettysburg Conference Committee. It should read ***Barbara Victor****. I am sorry and I hope that you will forgive me. Please take a symbolical flower from me. Josef*

At the occasion of a Conference held in Münster, Germany, 20 - 23 May we met Guido Pinkernell from the University Münster who showed us a very strange operator - **: = =**, *which helped him using the random numbers generator. You might know that unfortunately RANDOM() used in function generates new random numbers at each call of the function. So if one wants to work with a list of random numbers he has to store that list under a variable´s name in an additional assignation.*
Guido overcame that disadvantage. Heinz-Rainer Geyer listened - like me - fascinated Guido´s explanations and some weeks later I received an email-contribution from Heinz-Rainer and then another one from Guido Pinkernell - both dealing with : = = , which Heinz-Rainer calls "delayed assignment".

The Delayed Assigment

Heinz Rainer Geyer, St. Katharinen, Germany; HeinzRainer@t-online.de

04 June 1997

Hallo Josef,

It is night and I am working on a strange thing. I am sure you are remembering Guido Pinkernell´s

:==.

Try the following:

#1: $\text{MAL}(a,\ n) :\equiv \text{VECTOR}([[a := \text{RANDOM}(100),\ *,\ a]]_1,\ i,\ n)$

#2: $\text{MAL}(20,\ 3) = \begin{bmatrix} 33 & * & 20 \\ 58 & * & 20 \\ 23 & * & 20 \end{bmatrix}$

The first component is variable, the second one constant!?!? I intended to produce the value for a once in each "multiplication" and use it twice. Do you have an idea??

I had no idea. But as time goes by: some days later

10 June 1997

Hallo Josef,

some reasoning helped to get on the track of this mystery. You are not allowed to have the variable among the function´s parameters. So I could finish my *DERIVE* - program for Mental Arithmetic. The "delayed assignment" was a valueable support.

So I tried MAL() once more:

#3: $\text{MAL2}(n) :\equiv \text{VECTOR}([[a := \text{RANDOM}(100),\ *,\ a]]_1,\ i,\ n)$

#4: $\text{MAL2}(3) = \begin{bmatrix} a := 89 & * & 89 \\ a := 43 & * & 43 \\ a := 27 & * & 27 \end{bmatrix}$

Load MENTAL.MTH as Utility file, then use the functions: PLUSS, MINUSS, MULTS, MIS, DIS, DIVIS, SQUARED and MIXED. CHECK and CHECK2 are functions to check the results. Look at the examples below:

```
#1:  "MENTAL.MTH
#2:  TEST(g):=IF(RHS(g)=LHS(g),"right","wrong","?????????")
#3:  CHECK(v):=VECTOR([v SUB i,TEST(v SUB i)],i,1,DIMENSION(v))
#4:  CHECK2(v):=[VECTOR([v SUB i,TEST(v SUB i)]
     SUB 2,i,1,DIMENSION(v))]`
#5:  op:=["+","-","*"]
#6:  PROB(s,a_):=[[RANDOM(s+1),a_,RANDOM(s+1)]]
#7:  PROBS(n,s,a_):=VECTOR([RANDOM(s+1),a_,RANDOM(s+1)]`,i,1,n)
#8:  PLUS(s):=PROB(s,op SUB 1)
#9:  PLUSS(n,s):=PROBS(n,s,op SUB 1)
#10: MINUS(s):=PROB(s,op SUB 2)
```

```
#11: MINUSS(n,s):=PROBS(n,s,op SUB 2)
#12: MULT(s):=PROB(s,op SUB 3)
#13: MULTS(n,s):=PROBS(n,s,op SUB 3)
#14: MI(s):=[b_:==RANDOM(s+1),"-",IF(b_<2,0,RANDOM(b_),RANDOM(b_))]
#15: MIS(n,s):=VECTOR(MI(s),i,1,n)
#16: DI(s):=[(c_:==RANDOM(s+1)+1)*RANDOM(s+1),":",c_]
#17: DIS(n,s):=VECTOR(DI(s),i,1,n)
#18: TEILER(s):=VECTOR(IF(s/i=FLOOR(s/i),i,0),i,1,
     FLOOR(SQRT(s)+0.5))
#19: TEIL(s):=SELECT(k/=0,k,TEILER(s))
#20: TEIL1(s):=VECTOR(s/k,k,REVERSE_VECTOR(TEIL(s)))
#21: TM(s):=APPEND(TEIL(s),TEIL1(s))
#22: T(s):=IF(SQRT(s)/=FLOOR(SQRT(s)),TM(s),DELETE_ELEMENT(TM(s),
     DIMENSION(TM(s))/2))
#23: DIVI(s):=[[d_:==RANDOM(s)+2,":",IF(PRIME(d_),2,(T(d_)) SUB
              (RANDOM(DIMENSION(T(d_))-1)+2))]]
#24: DIVIS(n,s):=VECTOR(ELEMENT(DIVI(s),1)`,i_,1,n)
#25: SQUARE(s):=[[RANDOM(s+1),"^",2]]
#26: SQUARES(n,s):=VECTOR(ELEMENT(SQUARE(s),1)`,i_,1,n)
#27: MIX(n,s):=VECTOR([o_:==RANDOM(4)+1,IF(o_<4,PROB(s,op SUB o_),
               [DI(s)])],i_,1,n)
#28: MIXED(n,s):=VECTOR((DELETE_ELEMENT((MIX(n,s)) SUB j_,1))
     SUB [1,1], j_,1,n)
```

Various examples:

$$\text{PLUSS(3, 50)} = \begin{bmatrix} 11 & \text{"+"} & 4 \\ 9 & \text{"+"} & 4 \\ 45 & \text{"+"} & 27 \end{bmatrix} \qquad \text{MINUSS(3, 100)} = \begin{bmatrix} 68 & \text{"-"} & 81 \\ 11 & \text{"-"} & 40 \\ 51 & \text{"-"} & 60 \end{bmatrix}$$

$$\text{MULTS(4, 100)} = \begin{bmatrix} 0 & \text{"*"} & 72 \\ 74 & \text{"*"} & 93 \\ 65 & \text{"*"} & 24 \\ 69 & \text{"*"} & 96 \end{bmatrix} \qquad \text{DIVIS(5, 20)} = \begin{bmatrix} 2 & \text{":"} & 2 \\ 4 & \text{":"} & 4 \\ 20 & \text{":"} & 2 \\ 20 & \text{":"} & 5 \\ 10 & \text{":"} & 10 \end{bmatrix}$$

$$\text{DIS(5, 20)} = \begin{bmatrix} 18 & \text{":"} & 18 \\ 400 & \text{":"} & 20 \\ 272 & \text{":"} & 17 \\ 56 & \text{":"} & 4 \\ 45 & \text{":"} & 5 \end{bmatrix} \qquad \text{MIS(3, 100)} = \begin{bmatrix} 8 & \text{"-"} & 5 \\ 96 & \text{"-"} & 85 \\ 70 & \text{"-"} & 27 \end{bmatrix}$$

If you would try MENTAL.MTH with later versions of *DERIVE*, it would not work because of several reasons: first of all the “Delayed Assignment” is not accepted by *DERIVE*. You may enter it but is will be shown and treated as an ordinary assignment :=.

Operation(n,s) gives a list of n operations z1 op z2 with s = max(z1,z2). In MIS() you never obtain a negative result. In DIS() s = max(divisor,quotient).

$$\#50:\ \mathrm{MIXED}(3,\ 20) = \begin{bmatrix} 8 & "+" & 20 \\ 17 & "*" & 1 \\ 42 & ":" & 7 \end{bmatrix}$$

$$\#51:\ \mathrm{CHECK}\left[8 + 20 = 28,\ 17\cdot 1 = 17,\ \frac{42}{7} = 8\right] = \begin{bmatrix} 28 = 28 & "right" \\ 17 = 17 & "right" \\ 6 = 8 & "wrong" \end{bmatrix}$$

$$\#52:\ \mathrm{CHECK2}\left[8 + 20 = 28,\ 17\cdot 1 = 17,\ \frac{42}{7} = 8\right] = \begin{bmatrix} "right" \\ "right" \\ "wrong" \end{bmatrix}$$

I was not able to reproduce an output similar to #51 because recent versions of *DERIVE* don't show the unsimplified calculation – at the moment when presenting 25 + 15 it will be simplified to 40. But there are also some improvements. My adapted version for DERIVE 6 is following working with short programs instead of the Delayed Assigment. Josef

mental_new.mth

```
#1:   test(g) :=
        If g = true
          "right"
          "wrong"
          "???"

#2:   check(v) := [VECTOR(test(s, v   ), i, DIM(v))]'
                                  i,1

#3:   op := [ + ,  - ,  * , :]

#4:   prob(s, a_) := [APPEND(STRING(RANDOM(s + 1)), a_, STRING(RANDOM(s + 1)))]

#5:   probs(n, s, a_) := VECTOR(prob(s, a_), i, n)

#6:   plus(s) := prob(s, op )
                           1

#7:   pluss(n, s) := probs(n, s, op )
                                   1

#8:   minus(s) := prob(s, op )
                            2

#9:   minuss(n, s) := probs(n, s, op )
                                    2

#10:  mult(s) := prob(s, op )
                           3

#11:  mults(n, s) := probs(n, s, op )
                                   3

#12:  mi(s, b_) :=
        Prog
          b_ := RANDOM(s + 1)
          [b_, "-", IF(b_ < 2, 0, RANDOM(b_), RANDOM(b_))]

#13:  mis(n, s) := VECTOR(mi(s, b_), i, n)

#14:  di(s, c_) :=
        Prog
          c_ := RANDOM(s + 1) + 1
          [APPEND(STRING(c_·(RANDOM(s + 1) + 1)), " : ", STRING(c_))]

#15:  dis(n, s) := VECTOR(di(s, c_), i, n)
```

$$\#16:\quad \text{teiler(s)} := \text{VECTOR}\left(\text{IF}\left(\frac{s}{i} = \text{FLOOR}\left(\frac{s}{i}\right),\ i,\ 0\right),\ i,\ \text{FLOOR}(\sqrt{s} + 0.5)\right)$$

#17: teiler(20) = [1, 2, 0, 4]

#18: teil(s) := SELECT(k ≠ 0, k, teiler(s))

$$\#19:\quad \text{teil1(s)} := \text{VECTOR}\left(\frac{s}{k},\ k,\ \text{REVERSE\_VECTOR(teil(s))}\right)$$

#20: tm(s) := APPEND(teil(s), teil1(s))

#21: tm(20) = [1, 2, 4, 5, 10, 20]

```
       t(s) :=
         If √s ≠ FLOOR(√s)
#22:        tm(s)
            DELETE_ELEMENT(tm(s), DIM(tm(s))/2)

       divi(s, d_) :=
         Prog
#23:       d_ := RANDOM(s) + 2
           [d_, ":", IF(PRIME(d_), 2, (t(d_))↓(RANDOM(DIM(t(d_)) - 1) + 2))]

#24:   divi(20) = [18, :, 6]

#25:   divis(n, s) := VECTOR(divi(s, d_), i, n)

#26:   square(s) := [RANDOM(s + 1), ^, 2]

#27:   squares(n, s) := VECTOR(square(s), i, n)

       mix(n, s, o_) :=
         Prog
           o_ := RANDOM(4) + 1
#28:       If o_ < 4
              prob(s, op↓o_)
              di(s)

#29:   mixed(n, s) := VECTOR(mix(n, s), i, n)
```

Now the examples (do you see the difference?):

#30: RANDOM(0) = 2598965251

$$\#31:\quad \text{pluss}(3,\ 50) = \begin{bmatrix} 17 + 5 \\ 0 + 21 \\ 15 + 11 \end{bmatrix}$$

$$\#32:\quad \text{mults}(4,\ 100) = \begin{bmatrix} 29 * 5 \\ 42 * 98 \\ 33 * 11 \\ 3 * 31 \end{bmatrix}$$

$$\#33:\quad \text{dis}(5,\ 20) = \begin{bmatrix} 77 : 7 \\ 110 : 11 \\ 180 : 20 \\ 133 : 19 \\ 143 : 11 \end{bmatrix}$$

$$\#34:\quad \text{minuss}(3,\ 100) = \begin{bmatrix} 32 - 94 \\ 6 - 97 \\ 55 - 29 \end{bmatrix}$$

$$\#35:\quad \text{divis}(5,\ 20) = \begin{bmatrix} 15 & : & 5 \\ 16 & : & 8 \\ 17 & : & 2 \\ 5 & : & 2 \\ 21 & : & 7 \end{bmatrix}$$

$$\#36:\quad \text{mis}(3,\ 100) = \begin{bmatrix} 38 & - & 23 \\ 15 & - & 12 \\ 3 & - & 2 \end{bmatrix}$$

$$\#37:\quad \text{mixed}(5,\ 30) = \begin{bmatrix} 5 - 3 \\ 13 * 0 \\ 8 - 25 \\ 203 : 29 \\ 29 * 28 \end{bmatrix}$$

$$\#38:\quad \text{check}\begin{bmatrix} 5 - 3 = 2 \\ 13 \cdot 0 = 0 \\ 8 - 25 = 17 \\ \dfrac{203}{29} = 7 \\ 29 \cdot 28 = 822 \end{bmatrix} = \begin{bmatrix} \text{right} \\ \text{right} \\ \text{wrong} \\ \text{right} \\ \text{wrong} \end{bmatrix}$$

The next contribution is only of of nostalgic – historic interest. As I noted above this assignment is no more existing in later versions of *DERIVE*. As you can read Albert Rich was not very happy with it – and so it was and remained undocumented.

The email contact between Guido Pinkernell and Al Rich should introduce Guido´s contribution. Josef

AN UNKNOWN ASSIGMENT OPERATOR OF DERIVE

Guido Pinkernell, Münster, Germany

Tuesday, 08. July 1997 13:04:46
Message
From: guido.pinkernell@uni-muenster.de,BBG
Subject: [Fwd: Derive assignment operators]
To: Josef Boehm

Lieber Josef,

auf meine Anfrage an die DERIVE Hersteller habe ich gerade folgende Antwort erhalten. Eine Kopie ging laut Mail-Kopf auch an David Stoutemyer. Ich denke, ich werde jetzt selbst einen Arbeitsbegriff erfinden und wie gehabt verfahren.

Herzliche Gruesse

Guido

Albert Rich wrote:

Hello Guido Pinkernell
Westfaelische Wilhelms-Universitaet, Einsteinstrasse 62
Institut fuer Didaktik der Mathematik 48149 Muenster
fon:+49 251 83-33769 fax:-38350 email:guido.pinkernell@uni-muenster.de

On Wed, 02 Jul 1997 09:58:46 +0200, you wrote:

> Could you please give me the English expressions for the assigment operators
>
> := and := =
>
> Among, at least German colleagues of mine, the latter seems to be widely unknown so that I have been asked to give a short introduction. However, further inquiries showed that there seems to be uncertainty about the correct English terms so that I would like to have DERIVEs official expressions for distinguishing those two operators.

I have not documented or named the :== operator primarily because I am not happy with having to confuse users with two different assignment operators.

However, if I have to distinguish between the := and the :== assignment operators, I usually just say "the colon equal operator" for the former and "the colon equal equal operator" for the latter.

My goal in the next version of Derive is to support multi-line function definitions with the ability to assign values to local variables. Obviously, this will require the := = type of assignment where the result of simplifying the right side is assigned to the variable on the left. Thus, I am considering making the := operator function the same as the current := = operator, and then eliminating the := = operator.

I would greatly appreciate knowing if you or any of your colleagues foresee any conceptual problems with making the above change to Derive.

Aloha,
Albert D. Rich,
Applied Logician
Soft Warehouse, Inc.

1 Prelimimaries

Some time ago in a private communication, one of the creators of DERIVE, Albert D. Rich, mentioned an assignment operator that helped with problems I had while programming stochastic experiments. This operator, **:==**, seems to be widely unknown so that it should be a good idea to give a short introduction at this place. However, Albert said that the effect of the assignment operator in the next version of DERIVE may change slightly. So the discussion of the assignment operators in this note may only be relevant to version 4 of DERIVE. But it still describes an interesting phenomenon in programming with DERIVE that seems to have gone unnoticed for a considerable while.

2 Some effects of DERIVE´s assigment operator :=

With **f := 1** every simplification of **f** gives **1** as output, as expected (Figure 1). To reflect on the functional programming principle of DERIVE, **g** has not been assigned the number **0** but a function with the constant value **0**. In the daily work with DERIVE, though, this detail seems of no great importance.

#1:	f := 1	User
#2:	f	User
#3:	1	Simp(#2)
#4:	1	Simp(#2)
#5:	1	Simp(#2)

Figure 1:

But when working with DERIVE´s random number generator it does make a difference. Having defined **g := RANDOM(2)**, the subsequent simplifications of **g** result in different numbers (Figure 2) which, at least to those who are used to procedural languages like BASIC, indeed must come unexpected. This phenomenon is easily explained by recalling the idea of functional programming: **g** has not been assigned a number that has been computed by **RANDOM(2)**. **g** stands for the function **RANDOM(2**) itself, and to simplify **g** means to call a random number generator which every time produces a new random number.

#1:	g := RANDOM(2)	User
#2:	g	User
#3:	0	Simp(#2)
#4:	0	Simp(#2)
#5:	1	Simp(#2)

Figure 2:

In some cases it is necessary to be able to recall random numbers that have been generated at an earlier stage of work. Maybe the best example is the empirical law of large numbers. Here one constantly observes the relative frequency of some random event, e.g "toss a fair coin and it shows ①". For this its present value is based upon all results that the coin has shown up to now.

A simulation with DERIVE is not easy for the reasons said above. Or how could our "coin" **h := RANDOM(2)** recall the number it has produced in the fourth round, say, when every time it will be asked it produces a new number?

This touches the general problem of applying concepts of procedural programming like indices and loops to functional programming languages which in their purest form do not provide commands for such ideas. In DERIVE, some authors manage with generating a list **L** of random numbers as a database from which they are now able to pick the numbers as elements **L sub i** for evaluation (Figure 3, cf. [1]). But such a list, once created, is necessarily finite, while the idea of the law of large numbers requires the experiment to be repeated as often as it is needed. One can solve this problem by various means, e.g. by computing another list **M** of random numbers and appending this to **L**. This requires some extra programming which can easily be avoided by means of the assignment operator **:==**.

```
#1:  h := RANDOM(2)                          User
#2:  VECTOR(h, i, 1, 10)                     User
#3:  [0, 1, 1, 0, 0, 1, 0, 1, 1, 1]          Simp(#2)
#4:  L := [0, 1, 1, 0, 0, 1, 0, 1, 1, 1]     User
#5:  L SUB 5                                 User
#6:  0                                       Simp(#5)
#7:  0                                       Simp(#5)
#8:  0                                       Simp(#5)
```

Figure 3:

3 "Early evaluation" with :==

```
#1:  g :== RANDOM(2)          User
#2:  0                        Simp(#1)
#3:  g                        User
#4:  0                        Simp(#3)
#5:  0                        Simp(#3)
#6:  0                        Simp(#3)
```

Figure 4:

The existence of the assignment operator **:==** in DERIVE seems to be widely unknown. All textbooks on DERIVE which I was able to check do not mention it, and in a survey of the assignment operators of different computer algebra systems a similar operator has been said to exist for MATHEMATICA though not for DERIVE [2]. In lack of an official term by DERIVE to distinguish **:==** from **:=** I will use "early evaluation" in contrast to the term "lazy evaluation", wellknown in a different context, which describes the non-evaluating effect of **:=**. In the following we will give a short demonstration on how our early evaluation operator works and how it helps with demonstrating of the law of great numbers.

While in **g:=RANDOM(2)** the variable **g** has been assigned the function **RANDOM(2)**, simplifying **g:==RANDOM(2)** will lead to evaluating **RANDOM(2)** *before* the resulting random number will be assigned to **g**. Every subsequent simplification of **g** repeats this random number (Figure 4), which now is the result that we had expected earlier in Figure 2. Now, for a demonstration of the law of great numbers we therefore can program DERIVE as shown in the next DERIVE - listing TOSSES.MTH.

```
#1:  coin := RANDOM(2)                                                User
#2:  TOSS(series,n):=(series:==VECTOR(coin,i,1,n))                    User
#3:  ACTUALIZE(series1,series2,newseries):=(newseries:==              User
     APPEND(series1,series2))
#4:  EVALUATE(series):= AVERAGE(series)                               User
#5:  TOSS(v, 10)=[0,0,1,0,1,1,0,0,1,0]                     User=Simp(User)
#6:  v=[0,0,1,0,1,1,0,0,1,0]                               User=Simp(User)
                     2
#7:  EVALUATE(v)  = ───                                     User=Simp(User)
                     5
#8:  TOSS(w,10)=[1,1,1,1,0,1,0,0,0,0]                      User=Simp(User)
#9:  v=[0,0,1,0,1,1,0,0,1,0]                               User=Simp(User)
#10: w=[1,1,1,1,0,1,0,0,0,0]                               User=Simp(User)
#11: ACTUALIZE(v,w,z)                                                 User
#12: z=[0,0,1,0,1,1,0,0,1,0,1,1,1,1,0,1,0,0,0,0]           User=Simp(User)
                      9
#13: EVALUATE(z)  = ────                                    User=Simp(User)
                     20

#14: TOSS(u,10)=[0,0,1,0,1,1,1,1,1,0]                      User=Simp(User)
#15: z=[0,0,1,0,1,1,0,0,1,0,1,1,1,1,0,1,0,0,0,0]           User=Simp(User)
#16: u=[0,0,1,0,1,1,1,1,1,0]                               User=Simp(User)
#17: ACTUALIZE(u,z,y)=[0,0,1,0,1,1,1,1,1,0,0,0,1,0,1,      User=Simp(User)
     1,0,0,1,0,1,1,1,1,0,1,0,0,0,0]
#18: y=[0,0,1,0,1,1,1,1,1,0,0,0,1,0,1,1,0,0,1,0,1,1,       User=Simp(User)
     1,1,0,1,0,0,0,0]
                     1
#19: EVALUATE(y)  = ───                                     User=Simp(User)
                     2
```

Figure 5:

The underlying idea of the program is to define functions that reflect the real actions of exploring random experiments, which here are: (a) toss a coin **[n]** times and denote the results in a list, (b) evaluate the list by computing the relative frequency of a chosen event, here "the coin shows ①", and (c) if thought necessary, continue tossing the coin and add results to the preceeding list to create a new actual list for evaluation.

The application of early evaluation here is twofold. In line **#2** it causes the coin to be thrown before the results are assigned to a list named [**series]**. In line **#3** it causes the new list **[series2]** to be appended to the preceeding list **[series1]** before the result is named **[newseries]**. Line **#4** defines a fairly simple function for evaluating the list. There are easy modifications for evaluating different events within the same or even different stochastic experiments.

This note was intended to give a short introduction to what we have called "early evaluation" in DERIVE. Stochastics seems to be the main field for applying **:==** though we can expect early evaluation to enhance programming DERIVE in other fields as well.

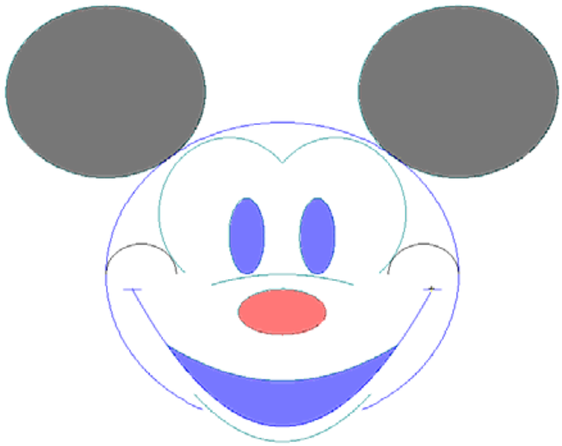
References

[1] Benno Grabinger. *Stochastik mit* Derive. Dümmler, Bonn, 1994.

[2] Reinhard Köhler. Mathematische Standardsoftware und Informatik - Zur Problematik des Zuweisungsoperators und des Gleichheitszeichens in Derive. Der mathematische und naturwissenschaftliche Unterricht, 48:195 – 198, 1995

[3] Guido Pinkernell. Zur Simulation langer Versuchsserien im Stochastikunterricht mit Derive. *Proceedings of the Meeting "Computer-Algebra im Mathematikunterricht" in Münster (Germany) 1997* (to appear)

Visit my TI-92 pal on the Information page, Mickey.



Mickey Mouse was presented by Sergey Biryukow on the International DERIVE Symposium 1997 FUN in Teaching Mathematics.

I believe that this could be a nice problem for students applying implicit, explicit and parameter form of conics, Josef

This is the DERIVE 6 Mickey Mouse

#1: $[10\cdot\cos(t) + 2,\ 10\cdot\sin(t)]$

$-1.1 \le t \le \pi+1.1$

#2: $\left[(x - 12)^2 + (y - 12)^2 \le 32,\ (x + 8)^2 + (y - 12)^2 \le 32,\ \sqrt{4 - (x - 10)^2},\ \sqrt{4 - (x + 6)^2}\right]$

#3: $\left[(x - 4)^2 + \dfrac{(y - 2.5)^2}{2.5^2} \le 1,\ x^2 + \dfrac{(y - 2.5)^2}{2.5^2} \le 1,\ \dfrac{(x - 2)^2}{2.5^2} + \dfrac{(y + 2.5)^2}{1.5^2} \le 1\right]$

#4: $\left[t,\ \dfrac{t^2 - 4\cdot t - 84}{8}\right]$

$-3 \le t \le 7$

#5: $\left[t,\ \sqrt{4\cdot(t + 35) - t^2} - 12\right]$

$-2 \le t \le 6$

#6: $[4\cdot\cos(t) + 5,\ 5\cdot\sin(t) + 4]$

$-0.9 \le t \le 2.4$

#7: $[4\cdot\cos(t) - 1,\ 5\cdot\sin(t) + 4]$

$0.72 \le t \le 4.02$

#8: $\left[t,\ \dfrac{t^2 - 4\cdot t - 122}{18}\right]$

$-4.57 \le t \le 8.57$

#9: $\left[t,\ \dfrac{t^2 - 4\cdot t - 76}{8}\right]$

$-6.5 \le t \le 10.5$

#10: $\left[\begin{bmatrix} 10 & -1 \\ 11 & -1 \end{bmatrix}, \begin{bmatrix} -7 & -1 \\ -6 & -1 \end{bmatrix}\right]$

#11: $\dfrac{x^2 - 4\cdot x - 122}{18} \ge y \ge \dfrac{x^2 - 4\cdot x - 76}{8}$

SOME OPPORTUNITIES IN 3D GRAPHING USING DERIVE

Neil Stahl, Wisconsin, USA

For years I felt computer graphics would be of benefit in demonstrating important concepts in calculus, such as tangent lines and tangent planes as well as direction fields. When I obtained *DERIVE* I developed a few programs to enable me to do that. The programs give one the ability to demonstrate these concepts in class, and explaining the programs to interested students gives an opportunity to introduce some additional concepts about transformations and computer graphics.

I am pleased at the opportunity you are offering me to distribute this work more widely.

```
#1:   showall(u) := VECTOR(ISOMETRIC(u ), k, DIM(u))
                                      k

      show(u) :=
        If DIM(u↓1) = 3
#2:        showall(u)
           ?
           ISOMETRIC(u)

#3:   rotx(α) := VECTOR(ROTATE_X(α)·u , k, DIM(u))
                                     k

#4:   roty(α) := VECTOR(ROTATE_Y(α)·u , k, DIM(u))
                                     k

#5:   rotz(α) := VECTOR(ROTATE_Z(α)·u , k, DIM(u))
                                     k

      rx(u, α) :=
        If DIM(u↓1) = 3
#6:        rotx(u, α)
           ?
           ROTATE_X(α)·u

      ry(u, α) :=
        If DIM(u↓1) = 3
#7:        roty(u, α)
           ?
           ROTATE_Y(α)·u

      rz(u, α) :=
        If DIM(u↓1) = 3
#8:        rotz(u, α)
           ?
           ROTATE_Z(α)·u

#9:   slices(v3, u, a, b, m) := VECTOR(v3, u, a, b, (b - a)/m)

#10:  grids(a, b, m, c, d, n) := VECTOR(VECTOR([x, y], x, a, b, (b - a)/m), y, c, d, (d - c)/n)

#11:  vfield(v2, a, b, m, c, d, n) := VECTOR(VECTOR([x, y] + t_·v2, a, x·a, b, (b - a)/m), y, c, d, (d - c)/n)

#12:  dfield(u, a, b, m, c, d, n) := VECTOR(VECTOR([x, y] + t_·[1, u]/√(1 + u^2), x, a, b, (b - a)/m), y, c, d, (d - c)/n)
```

You should remember that in 1997 we had no 3D-Plot Window available. We needed projections of 3D-objects on the *xy*-plane. ISOMETRIC(…) gives an isometric mapping.

SHOWALL transforms each of a ´vector´ (list) of 3D objects isometrically.

SHOW transforms one 3D object OR a list of them.

ROTX, **ROTY, ROTZ** apply a rotation matrix to each element in a list of 3D objects.

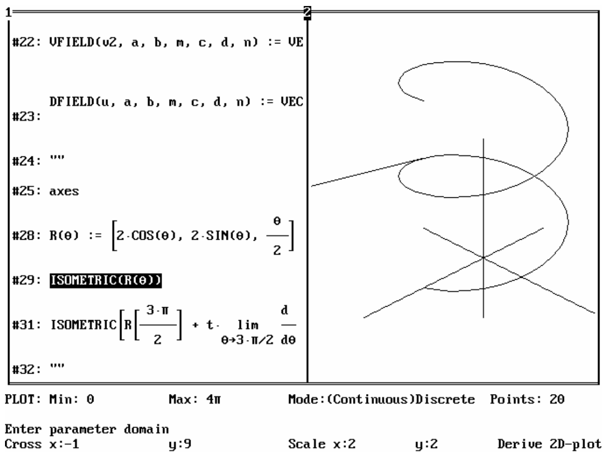
RX, **RY, RZ** apply a rotation matrix to one element OR each in a list.

SLICES evaluates v3 for variable u taking m values from a to b.

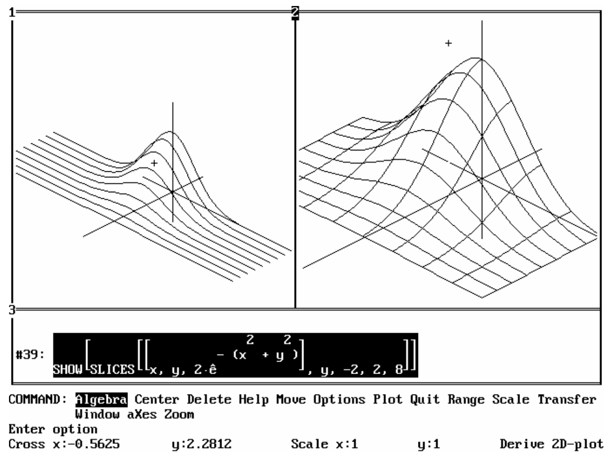
GRID generates a lattice of 2D points for a ≤ x ≤ b and c ≤ y ≤ d.

VFIELD can be used to plot elements of the 2D vector field v2 at points indicated as in **GRID.**

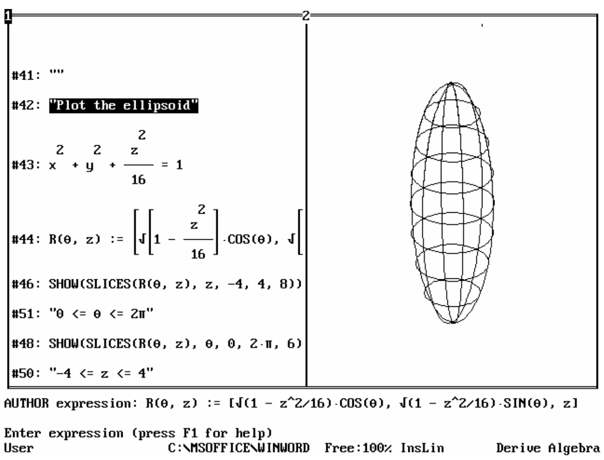
DFIELD plots the direction field u (probably depending on x and/or y) at points indicated as in **GRID**.

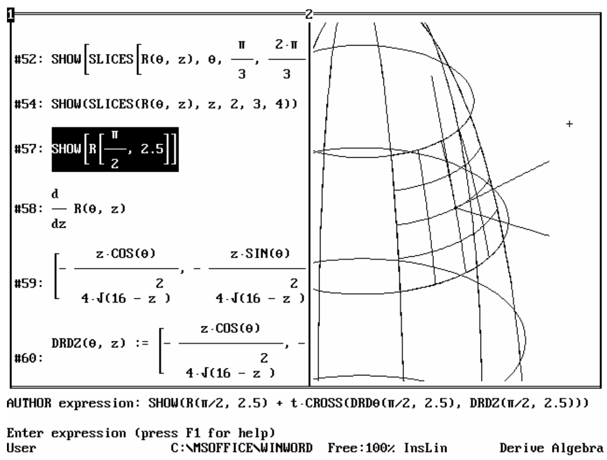


Simplify and Plot #25, #29 and #31

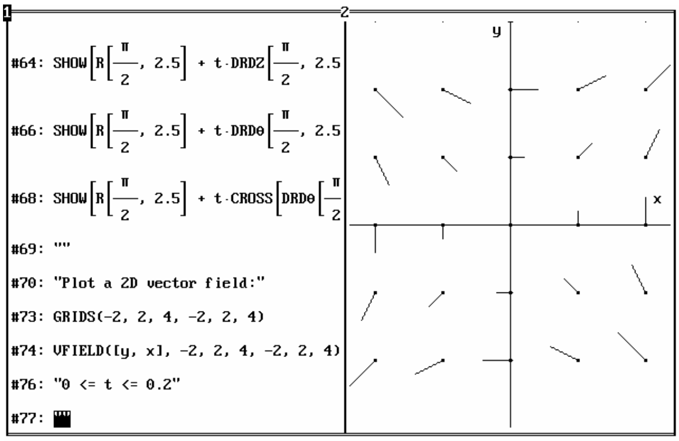


SHOW(SLICES([x,y,2*ê^(-(x^2+y^2))],x,0,2,8))
SHOW(SLICES([x,y,2*ê^(-(x^2+y^2))],y,-2,2,8))



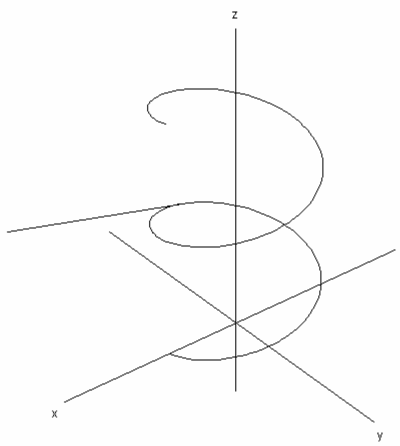


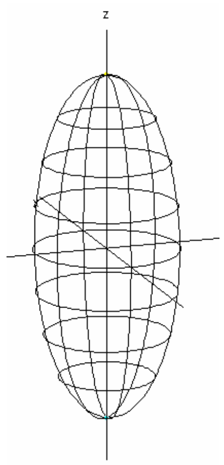
SHOW(R(π/2,2.5)+t*DRDZ(π/2,2.5))
SHOW(R(π/2,2.5)+t*DRDθ(π/2,2.5))
SHOW(R(π/2,2.5)+t*CROSS(DRDθ(π/2,2.5),DRDZ(π/2,2.5)))

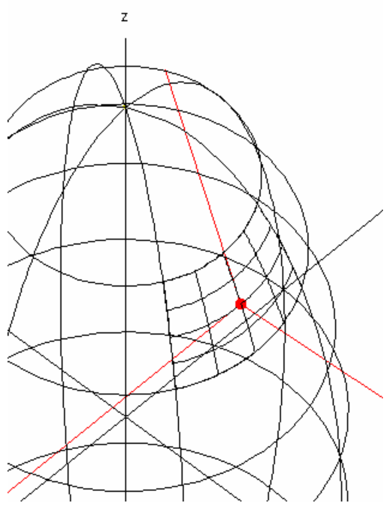


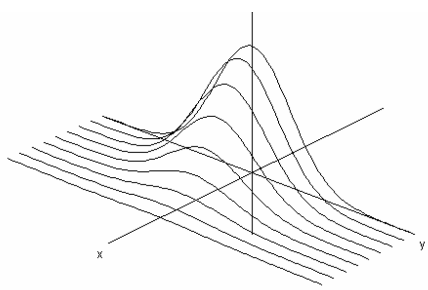
Neil Stahl, Professor, Math
University of Wisconsin Center-Fox Valley
Midway Road, Menasha, Wisconsin, USA 54952-8002
email: nstahl@uwcmail.uwc.edu

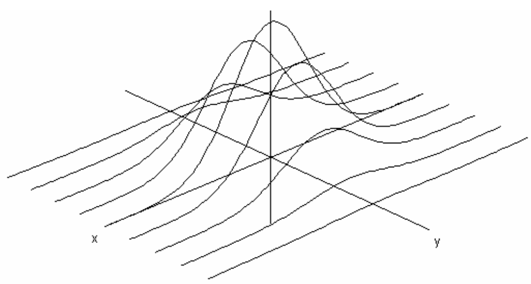
It might be nice to compare with the 3D-plots produced with *DERIVE* 6.

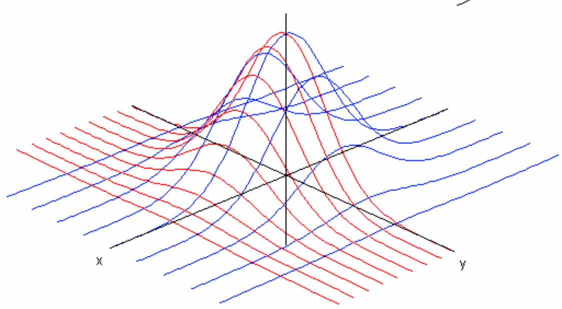












A CAS Competition

presented by

David R. Stoutemyer, Honolulu, Hawaii

The TI-92 tied for first place over MATHEMATICA in a problem-solving competition at the 1997 International Conference on Computer Algebra (ISAAC 97).

The competition was announced about three weeks before the meeting, then the five systems that accepted the challenge (Mathematica, Macsyma, Reduce, muPad and the TI-92) were sent ten very difficult problems. Team members were not allowed to tell people outside their team their solutions before the presentation at the meeting. (with Al Rich away traveling and David Stoutemyer already working with the TI-92 team, this precluded a *DERIVE* team. Otherwise the *DERIVE* solutions would be similar to the TI-92 solutions.)

The results demonstrate that the TI-92 is appropriate for industrial-strength research problems as well as education.

The Problems

1. What is the 4 significant digit approximation to the condition number of the 256 by 256 Hilbert matrix?
   Notes: The n by n Hilbert matrix has entries $h_{i,j} = 1/(i + j - 1)$ for $1 \le i \le n$, $1 \le j \le n$. The condition number of a non-singular matrix A is defined to be $Norm_2(A)\ Norm_2(A^{-1})$, where $Norm_2$ is the vector induced matrix norm defined by $Norm_2(A) = max(norm_2(Ax)/norm_2(x))$ over all non-zero vectors $x \in R^n$, and where the $norm_2$ is the Euclidean vector norm defined by $norm_2(x) = \sqrt{\sum_{i=1}^{n} x_i^2}$.

2. What is the value of $P = \int_1^6 x^{x^{x}} dx$ to 7 significant digits?

3. What is $\sum_{n=1}^{\infty} (n^{\pi} + n^2 + n^{\sqrt{2}} + 1)^{-1/3}$ to 14 significant digits?

4. What is the coefficient of x^{3000} in the expansion of the polynomial

   $$(x+1)^{2000}\ (x^2 + x + 1)^{1000}\ (x^4 + x^3 + x^2 + x + 1)^{500}$$

   to 13 significant digits?

5. What is the largest zero of the 1000th Laguerre polynomial to 12 significant digits?
   Note: The Laguerre Polynomials satisfy the following recurrence relation:

   $$L_0(x) = 1,\ \ L_1(x) = -x + 1$$

   $$L_n(x) = \frac{2n-1-x}{n} L_{n-1}(x) \ - \ \frac{n-1}{n} L_{n-2}(x) \ \text{ for } n > 1.$$

6. Find a lexicographic groebner basis for the (four) polynomial system:

$$8\ w^2 + 5wz - 4wy + 2wz + 3w + 5x^2 + 2xy - 7xz - 7x + 7y^2 - 8yz - 7y + 7z^2 - 8z + 8\ ;$$

$$3w^2 - 5wx - 3wy - 6wz + 9w + 4x^2 + 2xy - 2xz + 7x + 9y^2 + 6yz + 5y + 7z^2 + 7z + 5\ ;$$

$$-2w^2 + 9wx + 9wy - 7wz - 4w + 8x^2 + 9xy - 3xz + 8x + 6y^2 - 7yz + 4y - 6z^2 + 8z + 2\ ;$$

$$7w^2 + 5wx + 3wy - 5wz - 5w + 2x^2 + 9xy - 7xz + 4x - 4y^2 - 5yz + 6y - 4z^2 - 9z + 2\ .$$

7. What is $\int_0^1 x^2 Li_3\left(\frac{1}{x+1}\right) dx$ where $Li_n(z)$ is the polylogarithm function $\sum_{k=1}^{\infty} \frac{z^k}{k^n}$?

7. Let

$f(x)$ = $\tan(\tanh(\sin(x)))$ $+ \tanh(\sin(\tan(x)))$ $+ \sin(\tan(\tanh(x)))$ $- \tan(\sin(\tanh(x)))$ $- \sin(\tanh(\tan(x)))$ $- \tanh(\tan(\sin(x)))$ $- \tan(\sinh(\tanh(x)))$ $- \sinh(\tanh(\tan(x)))$ $- \tanh(\tan(\sinh(x)))$ $+ \tan(\tanh(\sinh(x)))$ $+ \tanh(\sinh(\tan(x)))$ $+ \sinh(\tan(\tanh(x)))$

$g(x)$ = $\sinh(\tanh(\sin(x)))$ $+ \tanh(\sin(\sinh(x)))$ $+ \sin(\sinh(\tanh(x)))$ $- \sinh(\sin(\tanh(x)))$ $- \sin(\tanh(\sinh(x)))$ $- \tanh(\sinh(\sin(x)))$ $- \tan(\sinh(\sin(x)))$ $- \sinh(\sin(\tan(x)))$ $- \sin(\tan(\sinh(x)))$ $+ \tan(\sin(\sinh(x)))$ $+ \sin(\sinh(\tan(x)))$ $+ \sinh(\tan(\sin(x)))$

What is $\lim_{x \to 0} \frac{f(g(x))}{g(f(x))}$ to 9 significant digits.

9. Find the largest eigenvalue λ to 13 significant digits for the integral equation

$$\int_0^1 \exp(x + y + x^2 + xy + y^2 + x^2 y^2) f(y)dy = \lambda f(x)$$

10. Consider the following initial value problem:

$$\frac{d^2 y}{dx^2} = x^3 + y^3 + \frac{dy^3}{dx}, \quad y(0) = 0, \ y'(0) = 0.$$

Find the smallest positive number *r* such that the solution has a derivative singularity at *x* = *r*. Calculate *r* to 13 significant digits. Is *y*(*r*) infinite or finite? If *y*(*r*) is finite then com pute it to 13 significant digits.

David Stoutemyer presented the problems and his way to solve them at the DERIVE Symposium ´97 at Särö, Sweden. It is obvious that all the problems cannot be solved by editing them and then pressing the ENTER-key. An excellent tool needs an excellent master to show its full capabilities. It might be a challenge for some of you trying to solve the problems using the TI, DERIVE or any other CA-systems. I´ll present some other problem´s solutions in the next DNLs. Josef

A TI-92 Solution to the ISSAC '97 Challenge Problem 3

David R. Stoutemyer, July 1997

Let $f(n)$ denote the summand and m denote an integer ≥ 1. Summing the first $m-1$ terms then applying the Euler-Maclaurin formula to the remaining terms gives:

$$\sum_{n=1}^{\infty} f(n) = \sum_{1}^{m-1} f(n) + \int_m^{\infty} f(n)dn + \frac{f(m)}{2} - \frac{f'(m)}{12} + \frac{f'''(m)}{720} - \frac{f^{(5)}(m)}{30240} + \ldots$$

The integral will be done by the built-in **nInt()** function. However, for an infinite upper limit it replaces the integration variable with its reciprocal, which doesn't work well for this integrand. Therefore, the built-in **nSolve()** function was used to determine that $1 + b^{\sqrt{2}} < 5\text{e-}14 \cdot (b^2 + b^{\pi})$ for $b >$ 5e7, where 5e-14 is float_epsilon for the TI-92. The integral was then split into a numeric integration from m to b, together with an exact symbolic integral from b to ∞ of a truncated power series for $(n^{\pi} + n^2)^{-1/3}$. Derived with the assistance of the TI-92, this tail integral is

$$-\sum_{j=0}^{k}\frac{C(-1/3,j)b^{1-\pi/3+(2-\pi)j}}{1-\pi/3+(2-\pi)j}$$

where $C(i,j)$ is the number of combinations of i things taken j at a time. Substituting the above value of b reveals that the third term is less than float_epsilon times the sum of the first two terms.

Initial experiments suggested that the sum was approximately 21. Then,

nSolve (-*f'*(*m*)/12 = 21*float_epsilon, *m*)

and

nSolve(*f'''*(*m*)/720 = 21*float_epsilon, *m*)

suggested that it might be necessary to include the derivative terms through *f'''*(*m*)/720 to achieve a practically small value of *m*. Moreover, that derivative is already complicated enough to discourage also including the next derivative term to further reduce *m*.

To reduce cumulative rounding error, the miscellaneous terms were added in order of increasing magnitude and the partial sum was done in reverse order.

For every *m* the two terms of the symbolic integral contribute about -2e-10 and 9.2. Here is a table showing the other contributions and total value together with the computing time as a function of *m*:

m	*f*(*m*)/2	-*f'*(*m*)/12	*f'''*(*m*)/720	integral	partial sum	total	sec
1	3e-1	3e-2	-8e-5	11.7	0	**21.193**365806583	109
2	2e-1	1e-2	-2e-4	11.2	0.63	**21.19323**6374751	105
4	1e-1	4e-3	-2e-5	10.6	1.3	**21.193239**967167	97
8	5e-2	1e-3	-2e-6	10.0	2.0	**21.193240**366886	92
16	3e-2	3e-4	-1e-7	9.4	2.6	**21.193240377**510	85
32	1e-2	7e-5	-7e-9	8.1	3.2	**21.19324037770**8	78
64	6e-3	2e-5	-4e-10	8.2	3.8	**21.193240377711**	73
128	3e-3	4e-6	-3e-11	7.7	4.3	21.193240377711	68
256	2e-3	1e-6	-2e-12	7.1	4.9	"	82
512	7e-4	2e-7	-1e-13	6.6	5.4	"	94
1024	4e-4	6e-8	-6e-15	6.1	5.9	"	143

The initial decline in computing time is because the more difficult quadratures dominate the increasing partial-sum times for small *m*.

The third derivative term becomes neglibible for $m \geq 256$, so we didn't need it after all, but without it we wouldn't have a good idea of the error due to truncating the derivative terms.

The least accurate component is almost certainly the numeric integration, which contributes a significant portion to the result. **nInt()** typically returns about 12 significant digits for well-behaved problems (sometimes more), and nInt() issues a warning if it seems likely that there are less than 6 significant digits. Therefore, the above converged value is probably not correct to more than about 12 digits – perhaps less. The totals might begin to increase in accuracy for much higher *m* as the quadrature contribution becomes less significant, but I don't have the patience to check that.

Although the requested 14 significant digits is impractical with this 14-digit arithmetic, it appears that this technique would permit the requested accuracy with IEEE 16-digit floating-point arithmetic and a correspondingly more ambitious numeric integrator.

email: swh@aloha.com

Complex Numbers - Fundamental Operations and their Representation

Bernhard Wadsack, Vienna, Austria

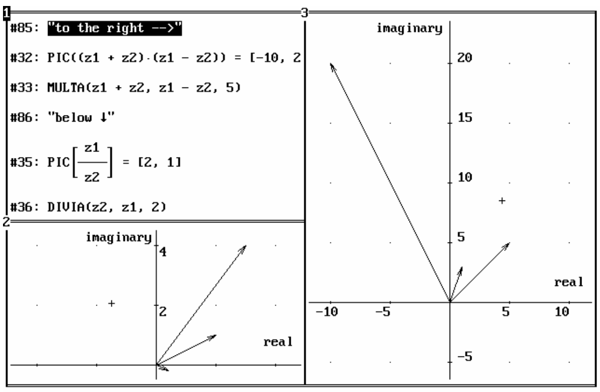
Using COMPLEX.MTH enables the students to represent the four fundamental operations, exponentiating and taking roots with complex numbers in the complex plane.

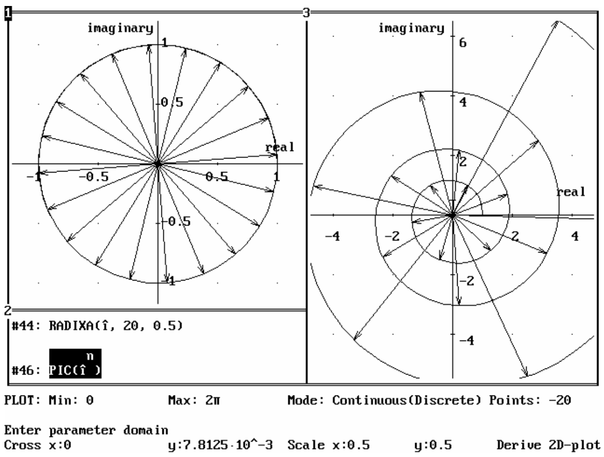
For plotting the vector representation it is recommended to set Options - State - Mode: CONNECTED and Size: SMALL (for the .....A - functions) and Size: LARGE otherwise.

Having loaded the utility file COMPLEX.MTH you have the following functions at your disposal:

PIC(z)	represents the complex number z = x + *i* y as the point Z(x,y)
LOC(z)	position vector OZ
ARROW(z,sc)	position vector OZ with an arrow head (sc = scale)
ADD(z1,z2)	position vectors of z1, z2 and z1 + z2
ADDA(z1,z2,sc)	position vectors of z1, z2 and z1 + z2 with arrow heads
SUBTR(z1,z2) and SUBTRA(z1,z2,sc)	difference
MULT(z1,z2) and MULTA(z1,z2,sc)	product
DIVI(z1,z2) and DIVIA(z1,z2,sc)	quotient
POT(z,n) and POTA(z,n,sc)	all powers of z with n = 1(1)n
POT1(z1,n,inc) and POT1A(z,n,inc,sc)	all powers of z with n = 1(inc)n
RADIX(z1,n) and RADIXA(z,n,sc)	all n^{th} roots of z

Some examples:






```
z1=3 + 4i; z2 = 2 + i

position vector of z1/z2 = (2,1)

position vector of z2/z1--> graph
```

```
POTA(0.5+#i,18,2)
PIC((0.5+#1)^n)       Plot with 0≤n≤18

RADIXA(#i,20,0.5)
PIC(#i^n)             Plot with 0≤n≤2π
```

#1: *** Complex Numbers with DERIVE ***

#2: [x :∊ Real, y :∊ Real, sc := 1]

#3: PIC(z) := [RE(z), IM(z)]

#4: LOC(z) := [[0, 0], PIC(z)]

#5: $\mathrm{SP}(z,\ sc) := \dfrac{0.15\cdot sc\cdot z}{|z|}\cdot\left(\mathrm{COS}\left(\dfrac{\pi}{12}\right) + i\cdot\mathrm{SIN}\left(\dfrac{\pi}{12}\right)\right)$

#6: $\mathrm{SP1}(z,\ sc) := \dfrac{0.15\cdot sc\cdot z}{|z|}\cdot\left(\mathrm{COS}\left(-\dfrac{\pi}{12}\right) + i\cdot\mathrm{SIN}\left(-\dfrac{\pi}{12}\right)\right)$

#7: ARROW(z, sc) := [[0, 0], PIC(z), PIC(z − SP(z, sc)), PIC(z), PIC(z − SP1(z, sc))]

#8: ADD(z1, z2) := [LOC(z1), LOC(z2), LOC(z1 + z2)]

#9: ADDA(z1, z2, sc) := [ARROW(z1, sc), ARROW(z2, sc), ARROW(z1 + z2, sc)]

#10: SUBTR(z1, z2) := [LOC(z1), LOC(z2), LOC(z1 − z2)]

#11: SUBTRA(z1, z2, sc) := [ARROW(z1, sc), ARROW(z2, sc), ARROW(z1 − z2, sc)]

#12: MULT(z1, z2) := [LOC(z1), LOC(z2), LOC(z1·z2)]

#13: MULTA(z1, z2, sc) := [ARROW(z1, sc), ARROW(z2, sc), ARROW(z1·z2, sc)]

#14: $\mathrm{DIVI}(z1,\ z2) := \left[\mathrm{LOC}(z1),\ \mathrm{LOC}(z2),\ \mathrm{LOC}\left(\dfrac{z1}{z2}\right)\right]$

#15: $\mathrm{DIVIA}(z1,\ z2,\ sc) := \left[\mathrm{ARROW}(z1,\ sc),\ \mathrm{ARROW}(z2,\ sc),\ \mathrm{ARROW}\left(\dfrac{z1}{z2},\ sc\right)\right]$

#16: $\mathrm{POT}(z1,\ n) := \mathrm{VECTOR}(\mathrm{LOC}(z1^{k}),\ k,\ 1,\ n)$

#17: $\mathrm{POTA}(z,\ n,\ sc) := \mathrm{VECTOR}(\mathrm{ARROW}(z^{k},\ sc),\ k,\ 1,\ n)$

#18: $\mathrm{POT1}(z1,\ n,\ inc) := \mathrm{VECTOR}(\mathrm{LOC}(z1^{k}),\ k,\ 1,\ n,\ inc)$

#19: $\mathrm{POT1A}(z,\ n,\ inc,\ sc) := \mathrm{VECTOR}(\mathrm{ARROW}(z^{k},\ sc),\ k,\ 1,\ n,\ inc)$

#20: $\mathrm{RADIX}(z1,\ n) := \mathrm{VECTOR}(\mathrm{LOC}(z1^{1/n}\cdot e^{i\cdot 2\cdot\pi\cdot k/n}),\ k,\ 0,\ n-1)$

#21: $\mathrm{RADIXA}(z,\ n,\ sc) := \mathrm{VECTOR}(\mathrm{ARROW}(z^{1/n}\cdot e^{i\cdot 2\cdot\pi\cdot k/n},\ sc),\ k,\ 0,\ n-1)$

#22: Examples:

#23: [z1 := 3 + 4·i, z2 := 2 + i]

#24: z1 + z2 = 5 + 5·i

#25: z1 − z2 = 1 + 3·i

#26: z2 − z1 = −1 − 3·i

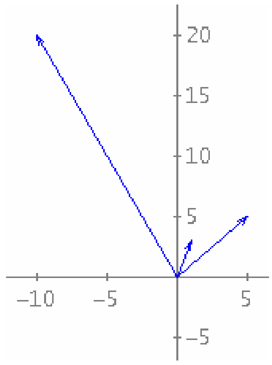
#27: z1·z2 = 2 + 11·i

#28: $\dfrac{z1}{z2} = 2 + i$

#29: $\dfrac{z2}{z1} = \dfrac{2}{5} - \dfrac{i}{5}$

These are the plots in DERIVE 6:

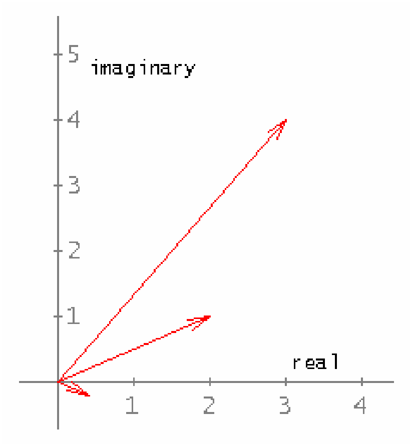
```
#30:  Notation := Decimal

#31:  NotationDigits := 3

#32:  PIC((z1 + z2)·(z1 - z2)) = [-10, 20]

#33:  MULTA(z1 + z2, z1 - z2, 5)
```



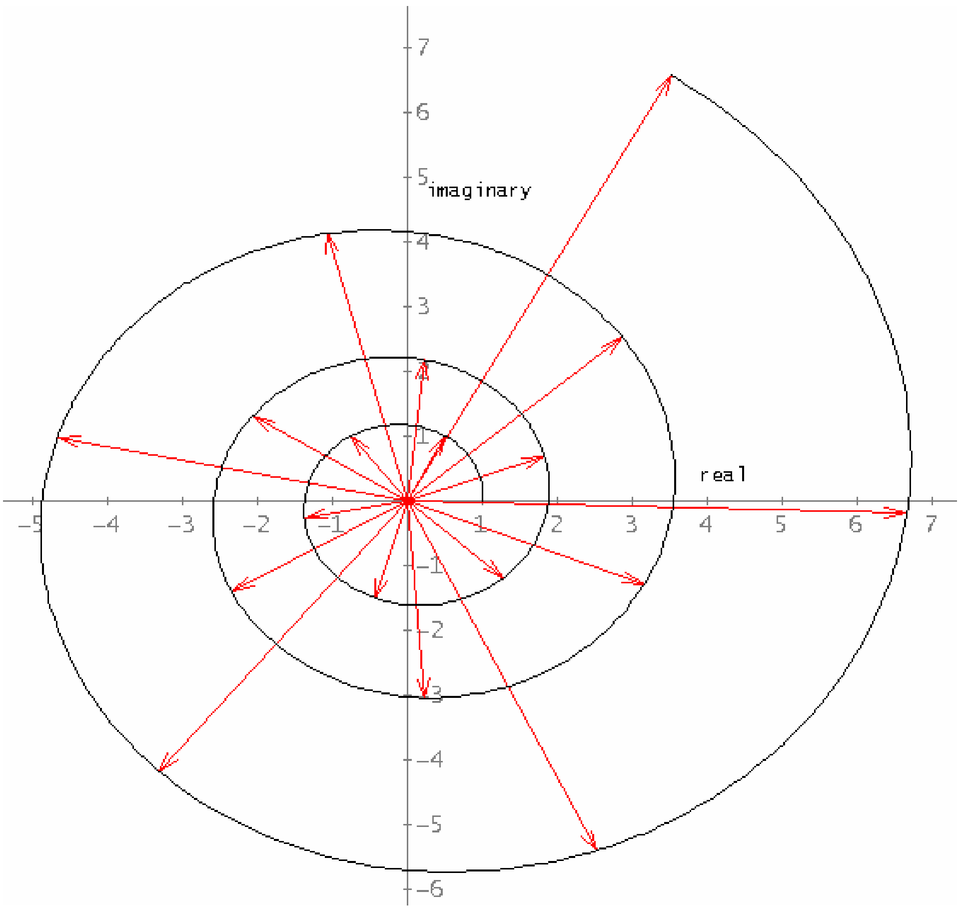
#34: $\text{PIC}\left(\dfrac{z2}{z1}\right) = [0.4,\ -0.2]$

```
#35:  DIVIA(z2, z1, 2)

#37:  POTA(0.5 + i, 18, 2)
```

#38: $\text{PIC}((0.5 + i)^n)$

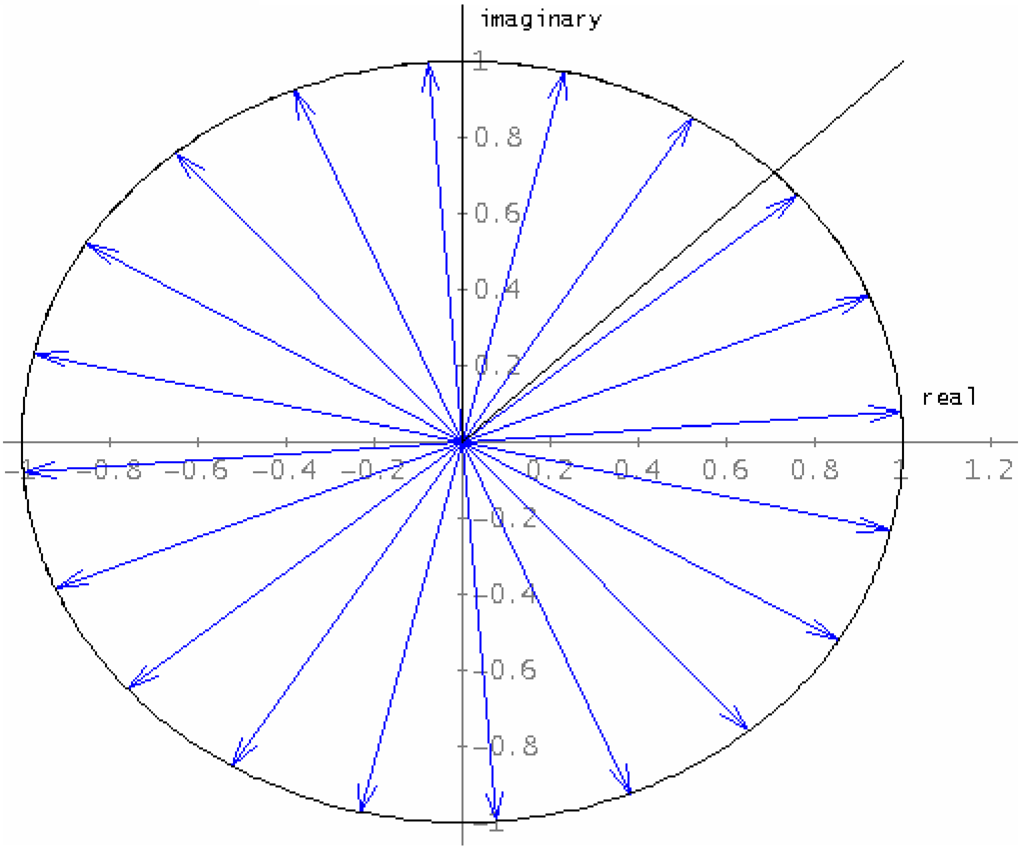


The “Complex Spiral”



#40: RADIXA(ί, 20, 0.5)

#41: $\text{PIC}(ί^n)$

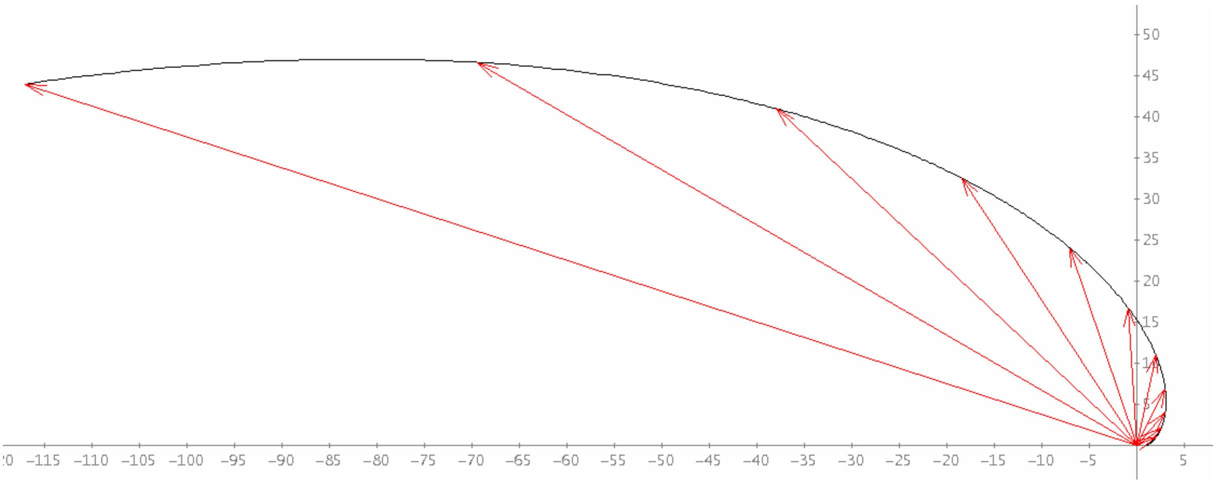


#54: RADIX(-117 + 44·ί, 6)

#55: $\left[\begin{bmatrix}0 & 0\\2 & 1\end{bmatrix}, \begin{bmatrix}0 & 0\\\frac{989}{7382} & \frac{9513}{4262}\end{bmatrix}, \begin{bmatrix}0 & 0\\-\frac{5432}{2911} & \frac{5251}{4262}\end{bmatrix}, \begin{bmatrix}0 & 0\\-2 & -1\end{bmatrix}, \begin{bmatrix}0 & 0\\-\frac{989}{7382} & -\frac{9513}{4262}\end{bmatrix}, \begin{bmatrix}0 & 0\\\frac{5432}{2911} & -\frac{5251}{4262}\end{bmatrix}\right]$

#56: $\text{PIC}((2 + ί)^n)$

#57: POT1A(2 + ί, 6, 0.5, 15)



Darstellung von Polyedern - Platonische und archimedische Körper im Mittelpunkt

Huber Weller, Vogelsang, Germany

In einem Kurs Lineare Algebra wird u.a. auch die Darstellung, Beschreibung und Untersuchung von räumlichen Objekten angestrebt. Dabei geht es "um eine psychologische Auffrischung und Ausformung grundlegender Raumvorstellungen, wie sie jeder normale Mensch von Kindheit an über seine Handlungen an räumlichen Gebilden aufbaut und im Laufe seiner geistigen Entwicklung immer klarer prägt, ausdifferenziert und miteinander verzahnt." (Wittmann, Elementargeometrie und Wirklichkeit, S.50).

Im Unterricht der Oberstufe habe ich immer wieder festgestellt, daß Schüler wissen wollten, wie Computergraphiken entstehen und wie man so etwas macht. Deshalb habe ich mich entschlossen, in einem Kurs DERIVE einzusetzen, um den mathematischen Hintergrund der Bewegungen bzw. der Abbildungen näher zu untersuchen. Durch die mögliche Verbindung der drei Werkzeugebenen (numerische, symbolische und grafische) wird eine Integration des raumgeometrischen Wissens unterstützt, wobei insbesondere die Window-Shuttle-Technik extensiv genutzt wird. Während des ganzen Lehrgangs haben uns die besonders schönen Körper (die platonischen und archimedischen) begleitet. Sie werden zunächst als Kantenmodelle hergestellt, danach zeichnerisch und schließlich mit DERIVE auf dem Bildschirm dargestellt.

Für diese Arbeit am Computer habe ich versucht zwei Prinzipien zu verfolgen:

1. Die Notation soll möglichst dicht an der üblichen Notation bleiben.
2. Die Darstellung der Körper auf dem Bildschirm sollte sich an der Tätigkeit des Zeichnens orientieren (Punkte werden miteinander verbunden).

Wegen der einfacheren Eingabe in DERIVE werden Punkte und Vektoren als ZEILENvektoren beschrieben.

In a Linear Algebra course usually representation, description and investigation of 3D-objects are included. We try to refresh basic skills in spatial imagination. Teaching in upper secondary schools since long I could notice that students are interested in creating computer graphics - "How is this possible?". So I used DERIVE to investigate the mathematical background of mappings and motions. Using the connection of three levels of tools - a Window Shuttle between numerical, symbolical and graphical techniques - I could support an integration of spatial-geometric knowledge. We specially dealt with the Platonic and the Archimedian Solids. Having produced models we drawed the solids by hand and then we represented them on the screen. Working with DERIVE I tried to follow two principles:

1. *Keep the notation as close as possible to the common one.*
2. *Presentation on the screen should be very similar to the drawing activity, i.e. points are connected.*

Points and vectors are written as ROW vectors.

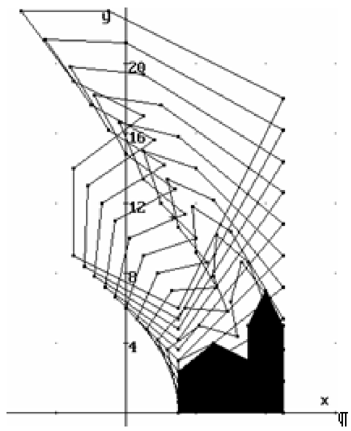
2D-Abbildungen - *2D Mappings*

Zunächst werden ebene Figuren als Punktfolgen im 2D-Grafikfenster gezeichnet (Connected- Modus). Daran schließt sich die Untersuchung von 2D-Abbildungen an. Da die Figuren durch Punktfolgen (d.h. durch eine Matrix) repräsentiert werden, kann das Bild der gesamten Figur durch eine einzige Matrizen-multiplikation berechnet werden. Dieser eher statische Vorgang kann mit dem VECTOR-Befehl dynamisiert werden, indem eine Bildfolge erzeugt wird, wobei die Einheitsmatrix schrittweise in die gegebene Matrix verändert wird:

Z.B. soll die durch die Matrix $\begin{bmatrix} 1 & 2 \\ -2 & 1 \end{bmatrix}$ beschriebene Abbildung visualisiert werden:

First of all plane figures are drawn as sequences (lists) of points (= matrices) in the 2D graphic window. Then we continue with investigations of 2D mappings, which can be described by a multiplication of matrices. This more static process will become dynamically producing a sequence of mappings using the VECTOR command. The unity matrix will be changed step by step to the given one.

For example lets visualize the mapping given by $\begin{bmatrix} 1 & 2 \\ -2 & 1 \end{bmatrix}$:




```
"Drehstreckung "
"Die Kirche : "
kirche:=[[3,0],[3,3],[5,4],[7,3],[7,5],
        [8,7],[9,5], [9,0],[3,0]]
VECTOR(kirche*[[1,k],[-k,1]],k,0,2,0.2)
```

Am Ende dieses Abschnitts müssen die Schüler auf jeden Fall wissen:

1. Eine Abbildung wird durch eine MATRIX beschrieben.
2. Die Hintereinanderausführung von Abbildungen wird beschrieben durch die Multiplikation der entsprechenden Matrizen.
3. Die Bilder der EINHEITSVEKTOREN stehen in den ZEILEN der Matrix.

At the end of this section the students should know:

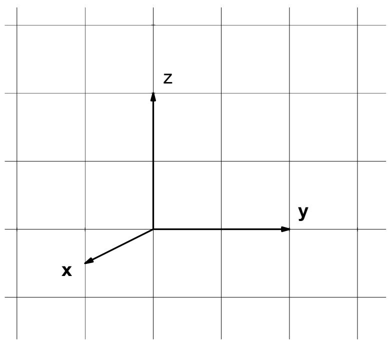
1. *A mapping is decribed by a MATRIX.*
2. *An execution of a sequence of mappings is decribed by a product of the respective matrices.*
3. *The pictures of the UNITY VECTORS can be found in the ROWS of the matrices.*

3D-Abbildungen - *3DMappings*

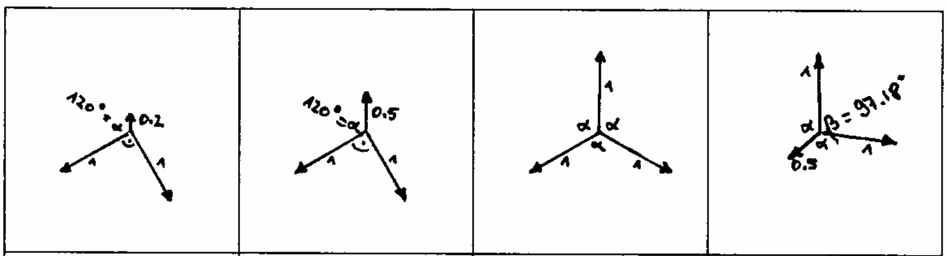
Zur grafischen Darstellung der dreidimensionalen Objekte werden die Erkenntnisse über lineare Abbil-dungen und Matrizen genutzt. Die Tätigkeit des Zeichnens wird als Abbildung des dreidimensionalen Raums in die Ebene aufgefaßt, d.h. nur durch die Bilder der Einheitsvektoren ist die Projektionsmatrix bestimmt. Bei der grafischen Darstellung müssen 3D-Koordinaten in 2D-Koordinaten transformiert werden. Wir benutzen eine Form der Darstellung, die wir auch schon beim Anfertigen der Bilder mit Papier und Bleistift benutzt haben:

[1 , 0 , 0] ==> [-0.5 , -0.25]
[0 , 1 , 0] ==> [1 , 0]
[0 , 0 , 1] ==> [0 , 1]

Der Name **Kavalierprojektion** für diese Art der Darstellung entstammt dem mittelalterlichen Militärwesen. Kavaliere (Reiter) sind Teile der Festungsaufbauten gewesen, deren Seitenansicht so unverzerrt dargestellt wurde. Will man den Grundriß unverzerrt darstellen, so benutzt man die **Militärprojektion**, bei der die 3.Achse verkürzt dargestellt wird. Diese Form der Darstellung wird heute oft in der Architektur und bei anschaulichen Stadtplänen benutzt. Daneben werden **Isometrie** (eine Kugel wird als Kreis dargestellt) und **Dimetrie** verwendet. Siehe hierzu auch Elschenbroich/Meiners S.26 ff.



*The name **cavalier projection** has its origin in the medieval military affairs. Cavaliers were parts of fortifications. In this projection their sideviews could be presented free from distortion. If you want to have an undistorted topview then you should use **military projection** with a shortened 3rd axis. This form of representation is still used in architecture and for vivid town maps. We also use **isometric projection** (a sphere´s picture is a circle) and **dimetric projection** [1].*



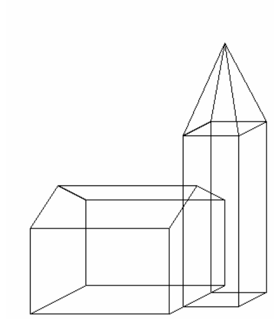
Militärprojektion 1 | Militärprojektion 2 | Isometrie | Dimetrie

```
"3DKIRCHE.MTH
"Various views of the church - The house´s vertices"
[a:=[4,0,0],b:=[4,5,0],c:=[0,5,0],d:=[0,0,0],e:=[4,0,3],f:=[4,5,3]]
 [g:=[0,5,3],h:=[0,0,3],i:=[2,0,4],j:=[2,5,4]]
"Die Eckpunkte des Turms - The tower´s vertices"
[k:=[3,5,0],l:=[3,7,0],m:=[1,7,0],n:=[1,5,0],o:=[3,5,6]]
[p:=[3,7,6],q:=[1,7,6],r:=[1,5,6],s:=[2,6,9]]
"The points connected show the whole solids."
kirche:=[a,b,f,e,a,d,h,i,e,f,j,i,h,g,j,g,c,d,c,b]
turm:=[k,l,p,o,k,n,r,o,s,r,q,s,p,q,m,l,m,n]
```

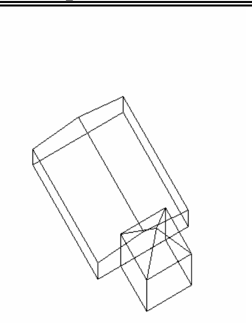
```
"Kavalierprojektion - Cavalier projection"

kav:=[[-0.5,-0.25],[1,0],[0,1]]

[kirche*kav,turm*kav]

"Militärprojektionen - Military projections"

mil1:=[[-COS(30*deg),-SIN(30*deg)],[COS(60*deg),
      -SIN(60*deg)],[0,0.2]]

mil2:=[[-COS(30*deg),-SIN(30*deg)],[COS(60*deg),
      -SIN(60*deg)],[0,0.5]]

[kirche*mil1,turm*mil1]

[kirche*mil2,turm*mil2]

"Isometrie - Isometric projection"

iso:=[[-COS(30*deg),-SIN(30*deg)],[COS(30*deg),-SIN(30*deg)],[0,1]]

[kirche*iso,turm*iso]

"Dimetrie - Dimetric projection"

dim:=[[-0.5*COS(41.41*deg),-0.5*SIN(41.41*deg)],[COS(7.18*deg),
     -SIN(7.18*deg)],[0,1]]

[kirche*dim,turm*dim]
```

Schräge Parallelprojektionen

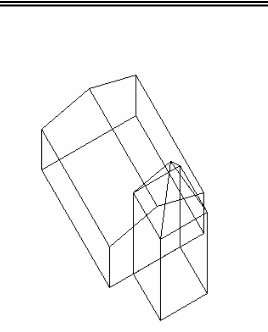
Vorsicht ! Optische Täuschungen !



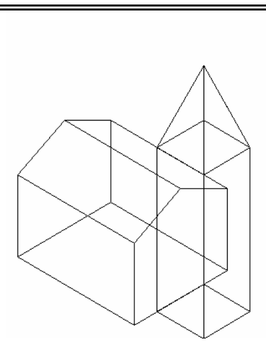
KAVALIERPROJEKTION



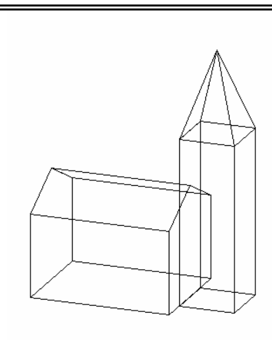
MILITÄRPROJEKTION 1



MILITÄRPROJEKTION 2



ISOMETRIE



DIMETRIE

From Linear Inequalities to Linear Programming (2)

Josef Böhm, Würmla, Austria

In DNL#26 I presented the visualization of linear inequalities and systems of linear inequalities. You know that a linear programming problem consists of a linear function which should obtain a maximum or minimum value under consideration of a system of linear constraints.

Let´s have an easy example:

Find the maximum value of	$z = x + 2y$
under the constraints	$x + 4y \le 24$ $2x + 3y \le 23$ $3x + 2y \le 27$
and usually	$x \ge 0,\ y \ge 0$

The following functions are provided:

`z := goal function`	
`rm := [constraint1, constraint2, ........]`	
`L(n)`	gives the boundary line of constraint #n in list rm
`ALL_INT_SECT`	returns the segments of the restriction lines in the 1st quadrant
`SIMPLEX`	plots the feasible region
`S_PKT(i,j)`	intersection point of restriction line i with restriction line j
`ISO(x,y)`	line parallel to the goal function passing the point P(x,y)
`ZIEL(x,y)`	value of the goal function in the point P(x,y)
`KNOTEN`	all existing intersection points (nodes) of the restriction lines including the axes) among one another
`BEREICH(xm,ym)`	shows the feasible region for rm with $0 \le x \le xm$ and $0 \le y \le ym$
`BER_INT(xm,ym)`	shows only the points with integer coordinates within BEREICH
`TOTAL`	returns a list of all nodes together with the value of the goal function in this point + comment whether feasible or not
`TOTZUL`	returns a list of all feasible nodes + value of the goal function
`maxloes, minloes`	gives the maximum (minimum) solution

All constraints – except the nonnegativity conditions – must be collected in the list (vector) `rm`, the goal function must be defined as `z:=`

```
#30:  z := x + 2·y
#31:  [r1 := x + 4·y ≤ 24, r2 := 2·x + 3·y ≤ 23, r3 := 3·x + 2·y ≤ 27, r4 := x ≥ 0, r5 := y ≥ 0]
#32:  rm := [r1, r2, r3]
```

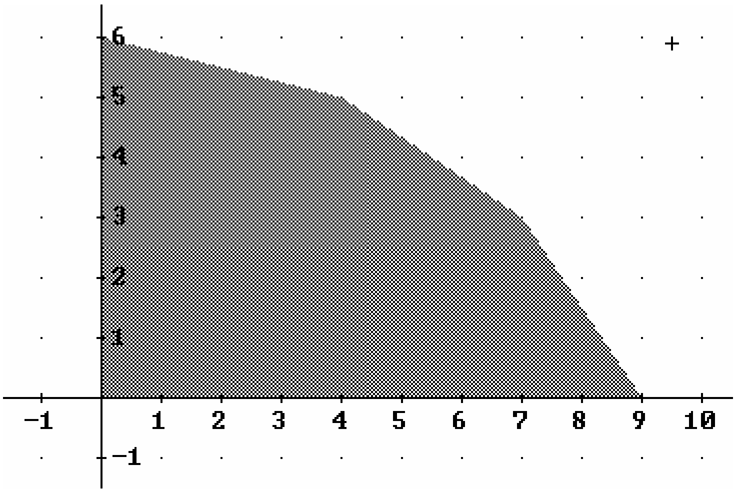
In 1997 we could not represent inequalities graphically, so I produced some functions for shading the respective half planes (see DNL#26). Here we use the functionality of DERIVE 6

We represent the feasible region

```
#33:  r1 ∧ r2 ∧ r3 ∧ r4 ∧ r5
```

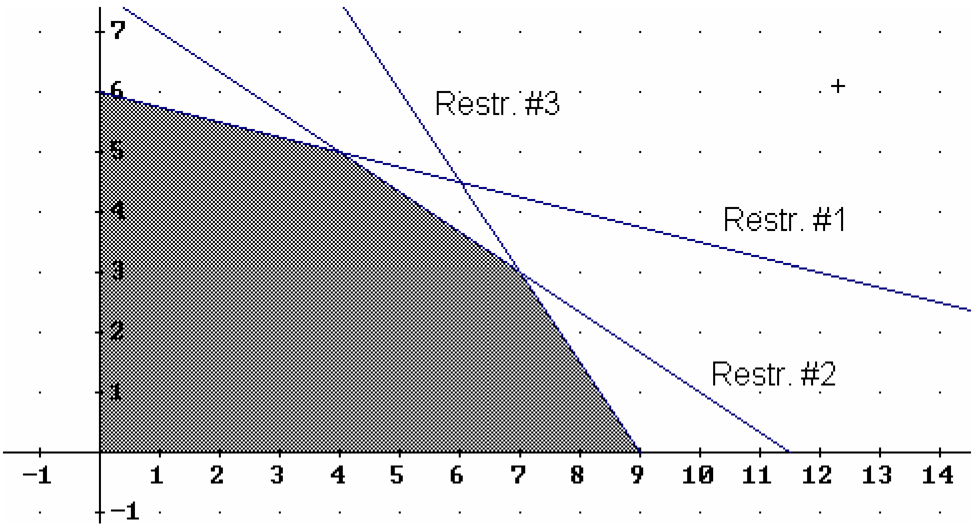
or simply plot SIMPLEX:

```
#34:  SIMPLEX
```



If you would like to see all boundary lines then:

```
#35:  ALL_INT_SECT
```

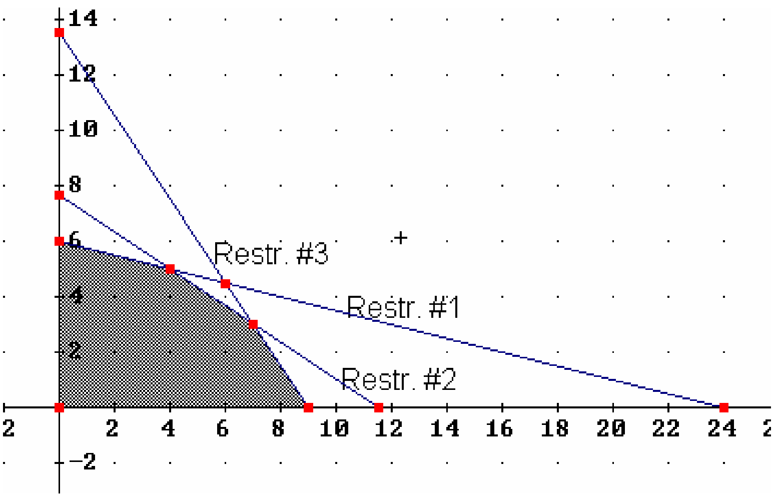


We can calculate the intersection points of the lines (vertices of the feasible region):

```
#36:  S_PKT(1, 2) = [4, 5]

#37:  S_PKT(2, 3) = [7, 3]

#38:  S_PKT(1, 3) = [6, 4.5]
```

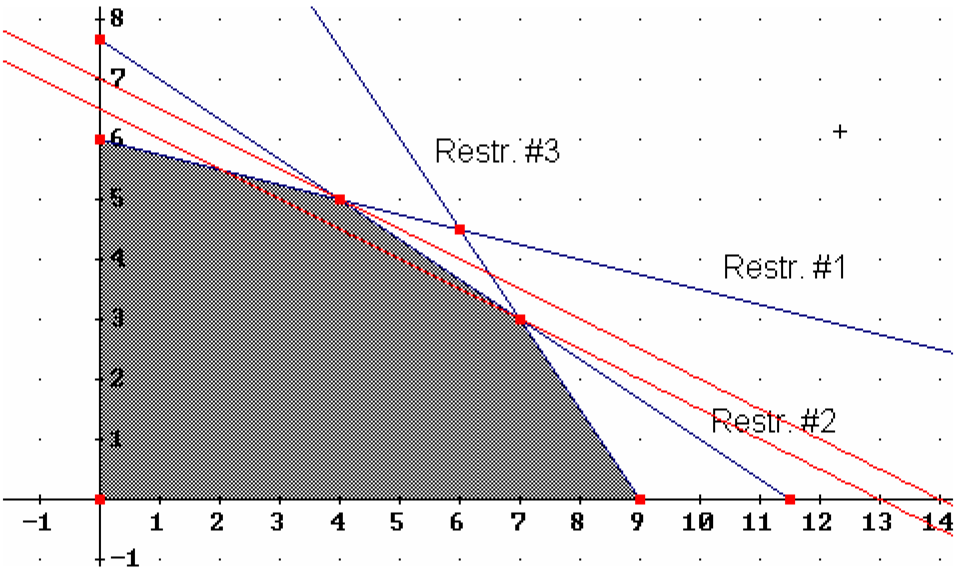
Function KNOTEN returns a list containing all $\binom{5}{2}$ possible intersection points. The solution must be found among these points

#39: KNOTEN = $\begin{bmatrix} 4 & 5 \\ 6 & \frac{9}{2} \\ 7 & 3 \\ 24 & 0 \\ \frac{23}{2} & 0 \\ 9 & 0 \\ 0 & 6 \\ 0 & \frac{23}{3} \\ 0 & \frac{27}{2} \\ 0 & 0 \end{bmatrix}$



Let us plot isogoal lines passing the intersection points of contraints (#1,#2) and (#2,#3). (The isogoal lines are the sets of points giving the same function value for the goal function, eg x + 2y = 10, x + 2y = 20, …)

```
#40:  [ISO(S_PKT(1, 2)), ISO(S_PKT(2, 3))]
```



We can observe that the line passing (4,5) (maximum distance from the origin but containing at least one point of the feasible region) gives the maximum value. We need only calculate this maximum value: 4 + 2·5 = 14. Or:

```
#41:  ZIEL([4, 5]) = 14
```

TOTAL returns a list of all nodes (= KNOTEN) together with the respective function values and a comment if feasible or not. TOTZUL selects the feasible solutions.

$$\#42:\quad \text{TOTAL} = \begin{bmatrix} x & y & \text{Goal} & \text{feasible} \\ 4 & 5 & 14 & \text{YES} \\ 6 & \frac{9}{2} & 15 & \text{NO} \\ 7 & 3 & 13 & \text{YES} \\ 24 & 0 & 24 & \text{NO} \\ \frac{23}{2} & 0 & \frac{23}{2} & \text{NO} \\ 9 & 0 & 9 & \text{YES} \\ 0 & 6 & 12 & \text{YES} \\ 0 & \frac{23}{3} & \frac{46}{3} & \text{NO} \\ 0 & \frac{27}{2} & 27 & \text{NO} \\ 0 & 0 & 0 & \text{YES} \end{bmatrix}$$

$$\#43:\quad \text{TOTZUL} = \begin{bmatrix} x & y & \text{Goal} \\ 4 & 5 & 14 \\ 7 & 3 & 13 \\ 9 & 0 & 9 \\ 0 & 6 & 12 \\ 0 & 0 & 0 \end{bmatrix}$$

Maximum and Minumum are given directly by:

$$\#44:\quad \text{maxloes} = \begin{bmatrix} x & y & \text{Maximum} \\ 4 & 5 & 14 \end{bmatrix}$$

$$\#45:\quad \text{minloes} = \begin{bmatrix} x & y & \text{Maximum} \\ 0 & 0 & 0 \end{bmatrix}$$

Just as a reminder on DOS times (from the DNL #27/1997)

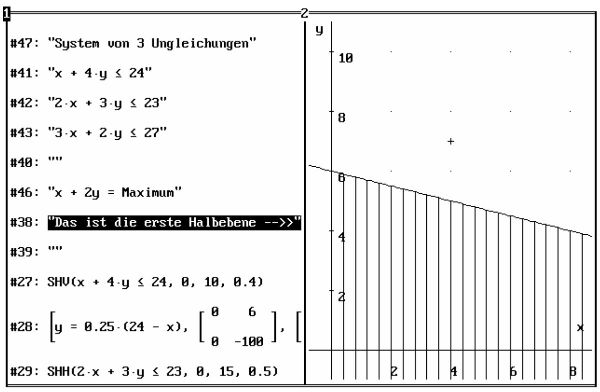


Figure 1

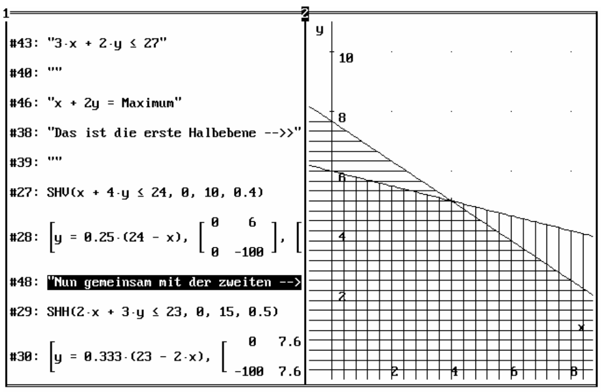


Figure 2

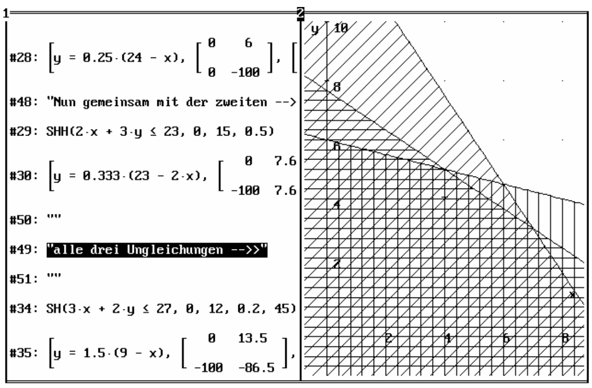


Figure 3

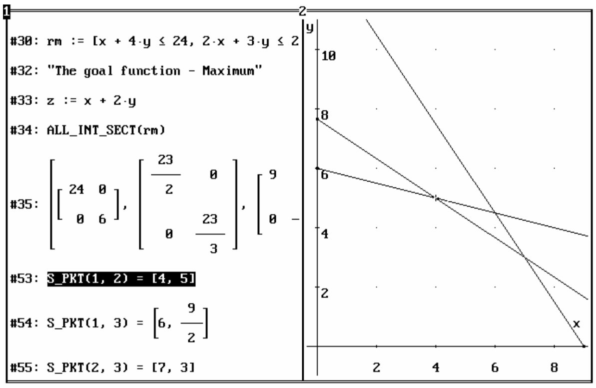


Figure 4

A Mixed Problem containing more constraints (and not all of the kind ≤)!!

Goal function: $z = 150\,x + 180\,y$

Constraints:

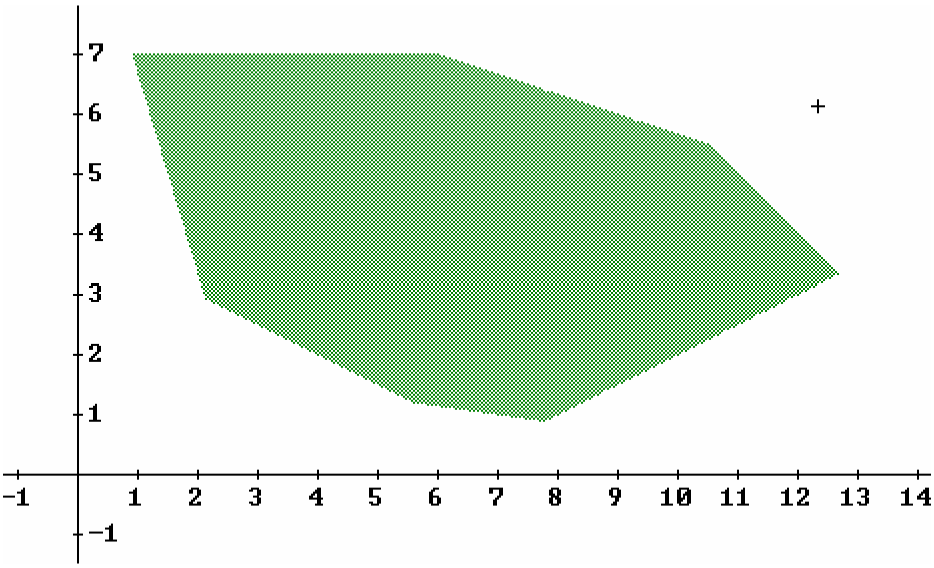
$$\begin{aligned} x - 2y &\le 6 \\ x + y &\le 16 \\ y &\le 7 \\ x + 3y &\le 27 \\ 10x + 3y &\ge 30 \\ x + 7y &\ge 14 \\ x + 2y &\ge 8 \end{aligned}$$

Find the feasible region.

Find maximum and minimum value of z.

Find maximum and minimum for integer solutions.

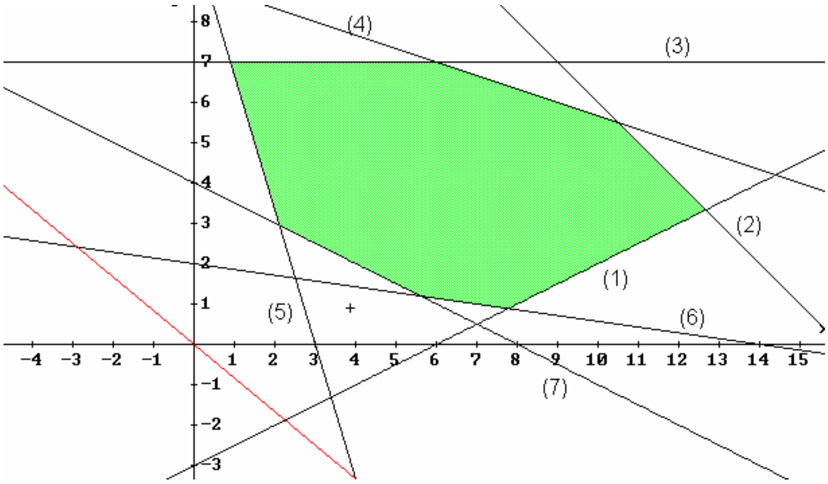
```
#46:  z := 150·x + 180·y
#47:  [r1 := x - 2·y ≤ 6, r2 := x + y ≤ 16, r3 := y ≤ 7, r4 := x + 3·y ≤ 27]
#48:  [r5 := 10·x + 3·y ≥ 30, r6 := x + 7·y ≥ 14, r7 := x + 2·y ≥ 8]
#49:  rm := [r1, r2, r3, r4, r5, r6, r7]
#50:  SIMPLEX
```



We identify the constraints and label them in the plot window.

Then we draw a iso line through the origin and try to find Maximum and Minimum by inspection.

```
#51:  L(2)
#52:  L(3)
#53:  L(4)
#54:  L(5)
#55:  L(6)
#56:  L(7)
#57:  z = 0
```



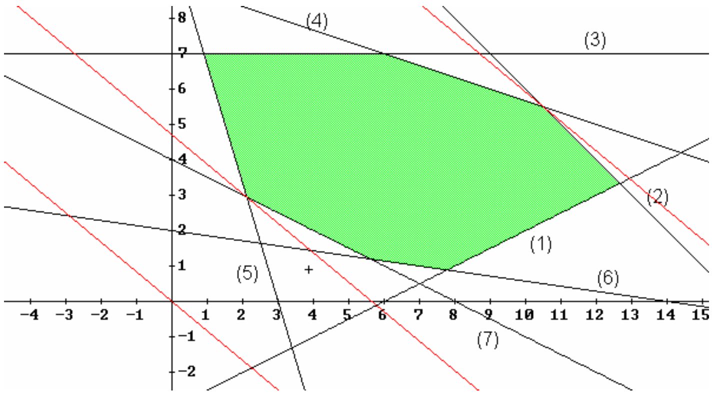
It seems to be that the maximum is reached in the interestion point of (2) and (4), the minimum in (7) ∩ (5). Hence we add the iso lines first and then calculate their intersection points:

#60: S_PKT(2, 4) = $\left[\frac{21}{2}, \frac{11}{2}\right]$

#61: ZIEL $\left[\frac{21}{2}, \frac{11}{2}\right]$ = 2565

#62: S_PKT(5, 7) = $\left[\frac{36}{17}, \frac{50}{17}\right]$

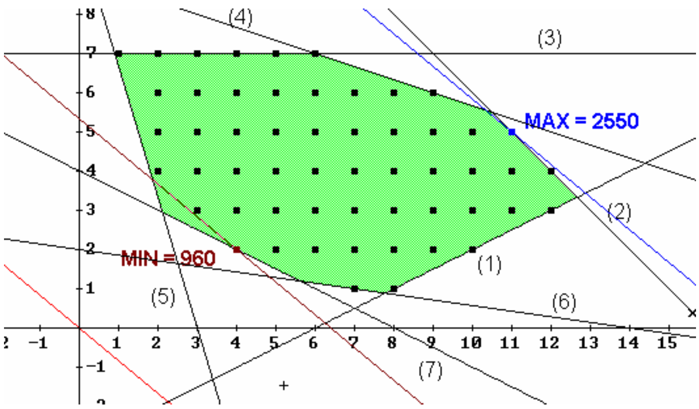
#63: ZIEL $\left[\frac{36}{17}, \frac{50}{17}\right]$ = $\frac{14400}{17}$



The maximum = 2565 is given for (x = 21/2, y = 11/2).
The minimum = 14400/17 is geven for (x = 36/17, y = 50/17).

Very often only integer solutions are wanted. So one has to inspect the integer grid points in order to find the right iso lines.

```
#66:  BER_INT(17, 7)
#67:  ISO([11, 5])
#68:  [11, 5]
#69:  ZIEL([11, 5]) = 2550
#70:  ISO([4, 2])
#71:  [4, 2]
#72:  ZIEL([4, 2]) = 960
```

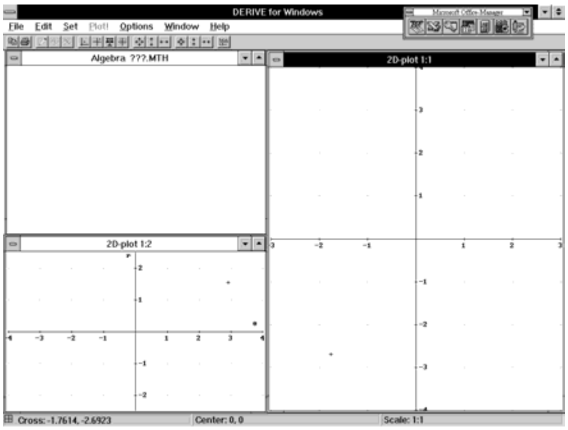


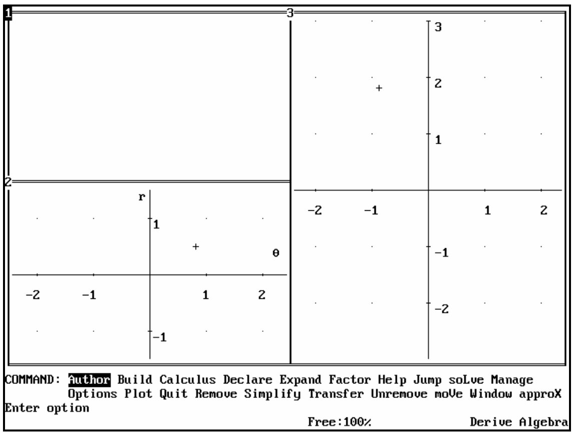
The plot should explain the procedure.

Polar Coordinates

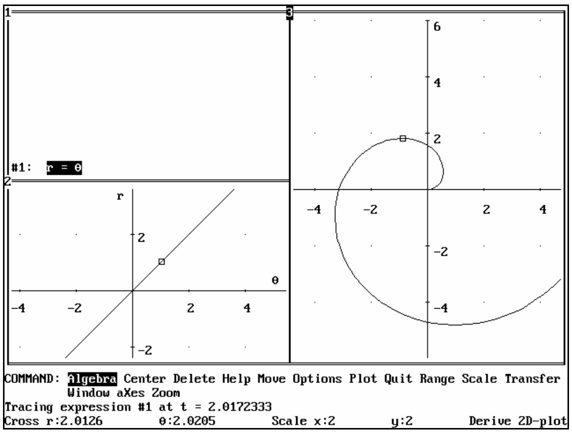
To work with the exercises in this section set up your screen to look like the one below (*DERIVE* for DOS = *DfD* or *DERIVE for Windows = DfW*).

When you are in the right 2D-window - the large one = window 3 - choose ***O****ptions and switch the* ***C****oordinates to* ***P****olar and make the a****X****es labels blank for this window. In the smaller 2D-window - = window 2 - let the* ***C****oordinates* ***R****ectangular and set the a****X****es labels to θ for the horizontal and r for the vertical axes. For further advise use your manual and/or the Help function.*





Author in the Algebra Window $\quad$ $r = \theta$



In window 2 plot this equation in rectangular co-ordinates. In window 3 plot in polar coordinates. Note that when you plot in polar coordinates, you must specify a Min and Max value. Choose 0 for the Min and 2π for the Max. Turn on the Trace and follow the cursor. Note in the lower left corner that *DERIVE* tells you the *r* and θ setting. Also note that it gives you a value for a parameter, *t*. How do *t* and θ differ? When do they agree?

Return to window 1 and repeat the above exercise for the expression

$$r = 2\sin(\theta)$$

What curve does this generate? Can you use the Polar Transformations

$$x = r\cos(\theta) \quad \text{and} \quad y = r\sin(\theta)$$

to transform this curve to rectangular coordinates?

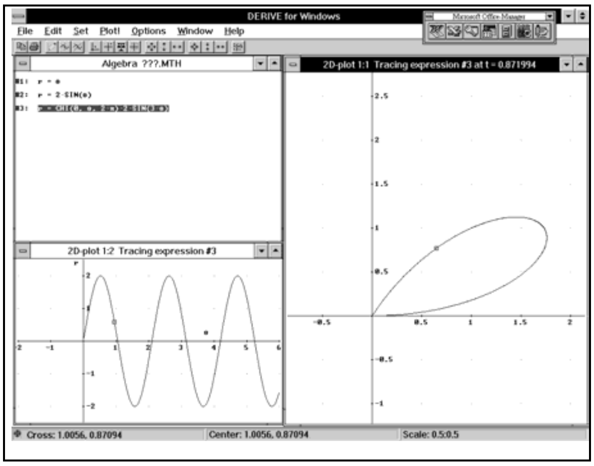
Hint: multiply both sides of the original equation by *r*.

A more interesting curve can be generated by the expression

$$r = 2\sin(3\theta)$$

Author this expression and also **A**uthor an expression to limit the plot in window 2 from 0 to 2π

$$r = \text{CHI}(0, \theta, 2\pi)\ 2\sin(3\theta).$$



Plot this expression in window 2. Return to the Algebra window and highlight the expression without the CHI function. Plot this expression in window 3 with θ going from 0 to $\frac{\pi}{3}$.

Trace this curve.

What is the location of the point corresponding to $r = 0.6$ and $\theta = 0.92$?

Plot this curve again for θ going from 0 to 2π.

As exercises we suggest you plot the curves $r = 2\ \sin(k\ \theta)$ for $k = 2, 3, 4, 5$. What observations can you make about this family of curves? What is the result of changing *sin* to *cos*?

We can use the identities $x^2 + y^2 = r^2$ and $\theta = atan\left(\frac{x}{y}\right)$

to convert a polar expression to an expression in rectangular coordinates. Above you converted

$$r = 2\sin(\theta) \text{ to } x^2 + y^2 = 2y.$$

What does $r = 2\cos(\theta)$ become?

We can use these identities in *DERIVE* together with the **M**anage/**S**ubstitute option to convert expressions from polar form to an equation in x and y. Let´s look at

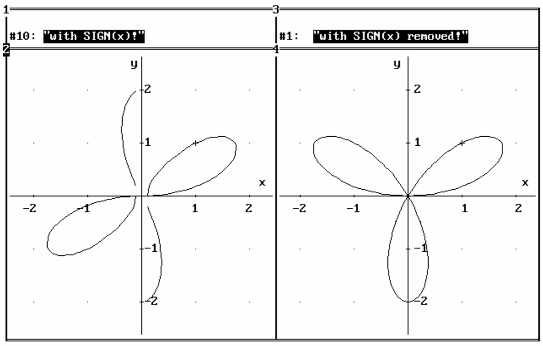
$$\boldsymbol{r = 2\sin(3\theta).}$$

Before we do any conversion let´s first expand $sin(3\theta)$ in terms of $sin(\theta)$ and $cos(\theta)$. Under the **M**anage menu, choose Trigonometry. Set the Direction to Expand.

Now highlight the right side of the polar expression and choose **E**xpand. The expression becomes

$$r = 2\,(4\,sin(\theta)\,cos^2(\theta) - sin(\theta)).$$

With this expression highlighted choose **M**anage/**S**ubstitute and replace **r** with $r = \sqrt{x^2 + y^2}$ and θ with $atan\left(\frac{x}{y}\right)$. **S**implify the result.



There will be an extraneous SIGN(x) in the expression and you may want to make the expression a little more palatable by multiplying both sides with $\left(x^2 + y^2\right)^{3/2}$.

Plot the final result in the two dimensional plot window with the Coordinates set to Rectangular. Even with a 486 machine this takes a while. Note that *DERIVE* has some problems in this plot with the points near the origin.

As a parting exercise try this one. It combines a polar plot, and a rectangular plot all on the same screen.

Dorothy and Toto are in the Land of Oz at the origin. They followed the polar curve r = θ *from* θ = 0 *to* θ = 4π *radians. They were joined by the Tinman and followed a parametric plot* [*t*, 4 *sin t*] *from t* = 4π *to* t = 10π. *At this point they were joined by the Cowardly Lion and Scare Crow and followed a curve* $y = (x - 10\pi)^2$ *and followed it until they entered the Emerald City at x = 12π. Plot the path of their trip in one window. Is the path continuous? What is the distance that each member of the group traveled?*

Titbits from Algebra and Number Theory

by Johann Wiesenbauer, Vienna

Back again! Today I couldn't be better in a better mood. As you may remember, things were quite different when I wrote the last column in this series. At that time, I was really up in arms complaining about the state of the utility file NUMBER.MTH which, in my opinion, left a lot to be desired. Well, things have changed dramatically since. Today a new version of DfW has become available and when Al Rich announced it to some members of the DERIVE-community in an e-mail communication, he wrote that "the utility file NUMBER.MTH included with DfW version 4.06 has been extensively revised. These enhancements are primarily due to suggestions and contributions made by Johann Wiesenbauer... " Now I am in a dilemma. Normally, I would have said that those enhancements are the best thing since sliced bread (many thanks to Carl Leinbach for telling me this idiom and others at the DERIVE-conference in Sweden!) but since I am more or less responsible for them this might be inappropriate. Therefore my only comment on the new update of NUMBER.MTH is (quoting Al from the e-mail again): "If you are at all interested in number theory, check it out!"

Anyway, may I offer you my help in doing so? After all, the corresponding parts of the help file have not yet been updated and some of you might not even have the new update of NUMBER.MTH. Ready? Let´s start then with the first new function

```
EXTENDED_GCD(a,b):=ITERATE([v_ SUB 2,[v_ SUB 4,v_ SUB 6]],v_,
   ITERATE(IF(MOD(v_ SUB 1,v_ SUB 2)=0,v_,[v_ SUB 2,MOD(v_ SUB 1,
   v_ SUB 2),v_ SUB 4,v_ SUB 3-FLOOR(v_ SUB 1,v_ SUB 2)*v_ SUB 4,
   v_ SUB 6,v_ SUB 5-FLOOR(v_ SUB 1,v_ SUB 2)*v_ SUB 6]),v_,[a,b,1,0,0,1]),1)
```

Its output is a vector [d,[x,y]], where d = gcd(a,b) and x,y are integers such that 1=xa+yb. This is essentially a revised version of EEA(a,b) in DNL#13, p27, based on the Extended Euclidean Algorithm, where I used the ITERATE(...,1)-trick to select the three numbers above from an 6-element vector. You will encounter this programming trick a number of times subsequently. As sure as eggs is eggs, most programmers would have used one of those ugly _AUX-functions instead. If you belong to them then I strongly urge you to rethink your policy. Believe me, your functions will become shorter, more self-contained and also more beautiful (though, admittedly, beauty is in the eye of the beholder!) without detracting from speed and readability!

What is this function actually good for? Well primarily you could use it to solve Diophantine equations of the form $ax + by = c$. This could be done in two steps:

Step 1: Assuming that EXTENDED_GCD(a,b) = [d,[u,v]] check whether d is a divisor of c. If not, then obviously the given Diophantine equation has no solutions. Otherwise continue with

Step 2: Multiply the equation $au + bv = d$ with c/d to see that $x = cu/d,\ y = cv/d$ or more generally $x = (cu+bt)/d,\ y=(cv-at)/d$ is a solution of the original equation for any integer t.

The following routine will do this automatically for you:

```
LIN_DIOPHANT(a,b,c):=ITERATE(IF(MOD(c,e_ SUB 1)=0,
   e_ SUB 2*c/e_ SUB 1+[b*@,-a*@]),e_,EXTENDED_GCD(a,b),1)
```

(Herein I used @ instead of t to avoid any conflict with previously defined variables.)

For instance, you could use it to solve tasks of the following sort (cf. [1], p25),

A woman has two egg timers due to two previous relationships. The first can time an interval of exactly 5 minutes and the second an interval of exactly 11 minutes. Unfortunately, her current lover insists on having a 3-minute egg for breakfast. Can you help her?

Of course, we can. All we have to do is to compute

```
LIN_DIOPHANT(5,11,3)=[11*@-6,3-5*@]
```

Setting e.g. @=0 we get the solution $x = -6$, $y = 3$ of $5x + 11y = 3$, which leads to the equation $3 \cdot 11 - 6 \cdot 5 = 3$. Therefore she could solve the problem by starting both egg timers at the same time, putting in the egg after 6 cycles of the 5-minute timer and removing it after 3 cycles of the 11-minute timer. This means that the whole procedure will last 33 minutes. Is this the best possible solution, what do you think? (Check other values of @!)

Of course, there are more serious "applications" of EXTENDED_GCD(a,b) than the previous one. In particular, if $(a,m)=1$ then $ax+ym=1$ means that x is an inverse of a mod m. Since the value of y was actually calculated to no purpose here, an own routine for the computation of the inverse of a mod m was included in NUMBER.MTH that makes no use of EXTENDED_GCD(a,b). If you are a regular reader of this column, you will know it by now (apart from the use of indexed variables):

```
INVERSE_MOD(a,m):=IF(GCD(a,m)=1,MOD((ITERATE(IF(MOD(v_ SUB 1,
   v_ SUB 2)=0,v_,[v_ SUB 2,MOD(v_ SUB 1,v_ SUB 2),v_ SUB 4,v_ SUB 3–
   FLOOR(v_ SUB 1,v_ SUB 2)*v_ SUB 4]),v_,[a,m,1,0])) SUB 4,m)) [1]
```

I don´t think that it is possible to overestimate the importance of this very function. Whenever linear congruences must be solved it comes into play and mathematics abounds of applications. For instance, it is needed in the well-known RSA-cryptosystem to calculate the decoding key (cf.[4]). The next three functions can also be viewed as applications of INVERSE_MOD(a,m). The first is our good old powermod-function

```
POWER_MOD(a,n,m):=MOD(IF(n>=0,n,INVERSE_MOD(a,m))^ABS(n),m)
```

that can compute a^n mod m now even if $a < 0$ (of course, the latter only on condition that $(a,m) = 1$, otherwise the output will be a question mark!). Also the other two functions

```
SOLVE_MOD(u,x,m):=ITERATE(IF(MOD(LIM(RHS(u)-LHS(u),x,0),t_)=0,
   ITERATES(s_+m/t_,s_,MOD(INVERSE_MOD(DIF(LHS(u),x)/t_,m/t_)*
   LIM(RHS(u)-LHS(u),x,0)/t_,m/t_),t_-1),[]),t_,GCD(DIF(LHS(u),x),m),1)
```

and

```
CRT(a,m):=ITERATE(MOD(a*VECTOR(p_/m_*INVERSE_MOD(p_/m_,m_),
   m_,m),p_),p_,PRODUCT(m_,m_,m),1)
```

have already been introduced, namely in DNL#26, p26, and in DNL#24, p40 respectively. To see these routines at work we will show as an example how they can be used to solve algebraic congruences, i.e. congruences of the form $f(x) = 0$ mod m, where $f(x)$ is a polynomial with integer coefficients.

To do this we need two propositions from number theory (cf. [3]).

Proposition 1: If u is any solution of $f(x) \equiv 0 \bmod p^{e-1}$ for an $e>1$, then $u + v\,p^{e-1}$, where v is any solution of the linear congruence

$$f'(u)\,y \;\equiv\; -\frac{f(u)}{p^{e-1}} \bmod p$$

will be a solution of $f(x) \equiv 0 \bmod p^e$ and all solutions of $f(x) \equiv 0 \bmod p^e$ can be obtained in this way.

Proposition 2: If $f(x) \equiv 0 \bmod m_1 m_2 \ldots\ldots m_r$ is an algebraic congruence where $m_1, m_2, \ldots., m_r$ are pairwise coprime, then all its solutions can be obtained by solving the congruence systems

$$x \equiv a_1 \bmod m_1$$
$$x \equiv a_2 \bmod m_2$$
$$\ldots$$
$$x \equiv a_r \bmod m_r$$

[1] The red printed functions have been implemented in DERIVE in the meanwhile.

for all r-tuples $(a_1, a_2, ..., a_r)$, where $f(a_i) \equiv 0 \mod m_i$, i = 1,2,...., r.

As an example we will try to find all solutions of

$$x^3 + 5x^2 - 11x + 3 \mod 106121003.$$

We start by setting

```
F(x):=x^3+5*x^2-11*x+3
m:=106121003
DIF(F(x),x)=3*x^2+10*x-11
FF(x):=3*x^2+10*x-11
```

and notice that

```
FACTOR(m)=101^3*103
```

In view of Proposition 1 we first determine the solution of the given congruence mod 101 and mod 103, respectively

```
SELECT(MOD(F(x_),101)=0,x_,0,100)=[11,93]
SELECT(MOD(F(x_),103)=0,x_,0,102)=[4]
```

Because of the equations

```
SOLVE_MOD(FF(11)*y=-F(11)/101,y,101)=[38]
11+101*38=3849
SOLVE_MOD(FF(3849)*y=-F(3849)/101^2,y,101)=[47]
3849+47*101^2=483296
```

we see that the first solution 11 mod 101 leads to the solution 483296 mod 101^3 of the given congruence, whereas the second branch starting with the solution 93 mod 101 "dies out" because of

```
SOLVE_MOD(FF(93)*y=-F(93)/101,y,101)=[]
```

Finally, we put together our solutions mod 101^3 and mod 103 by means of Proposition 2 and get

```
CRT([483296,4],[101^3,103])=2543898
```

i.e. the given congruence has the unique solution 2543898 mod m. (Check it!) If you are a very ambitious programmer, you could try to write a program that solves a given algebraic congruence by carrying out all the steps above automatically. (Good luck!)

Let´s turn to the next new routine (again not really new to regular readers of this column!), which is

```
SORT(v):=REVERSE_VECTOR(LIM(TERMS(v*VECTOR(x_^k_,k_,v)),x_,1))
```

Believe it or not, this very short routine can sort any list of different (!) rational numbers in ascending order and is lightning fast into the bargain. (If you want to sort in descending order it becomes even shorter: Simply leave out REVERSE_VECTOR!)

How is this possible? Well, if you take the trouble of looking into the code you will surely discover that I have simply stolen the internal sort-routine of DERIVE. This is really an exception of the rule, you see. Usually you have no access to all the hidden treasures that are used by DERIVE internally which can be quite annoying at times.

By the way, if the rational numbers in the list which you want to sort are not necessarily distinct there is no need to despair. In this case you should use the slightly more complicated routine

```
SORT2(v):=REVERSE_VECTOR(LIM(TERMS(EXPAND(VECTOR(k_*x_^APPROX(k_),
    k_,v)*VECTOR(y_^k_,k_,DIMENSION(v)))),[x_,y_],[1,1]))
```

Usually this routine will sort even a list of real numbers, but in contrast to what I said in DNL#23, p39, there may be exceptions for a rather strange reason. For some real numbers *c* DERIVE doesn´t recognize that *x c* is a polynomial! You can test this for a given real number *c* by checking whether it simplifies *x c* to *c x* or not! In this sense you shouldn´t include in your list real numbers like e^{π}, π^{e}, ...etc, but simple numbers like *e*, π, $\sqrt{2}$, ... don´t cause problems.

A good old acquaintance is also the next routine

```
FAREY(n):=SORT((ITERATE(IF(v_ SUB 1=v_ SUB 2 AND v_ SUB 2=n,
   v_,[IF(v_ SUB 1=v_ SUB 2,1,v_ SUB 1+1),IF(v_ SUB 1=v_ SUB 2,
   v_ SUB 2+1,v_ SUB 2),IF(GCD(v_ SUB 1,v_ SUB 2)=1,APPEND(
   v_ SUB 3,[v_ SUB 1/v_ SUB 2]),v_ SUB 3)]),v_,[1,1,[]])) SUB 3)
```

which computes the Farey-fractions of order n, i.e. the fractions in the interval (0,1] (some authors include 0, mind you!) whose denominator doesn´t exceed n ordered by size. If you are not yet familiar with them, you should check the following simple example:

FAREY(6)=[1/6,1/5,1/4,1/3,2/5,1/2,3/5,2/3,3/4,4/5,5/6,1]

Farey-fractions have a lot of intriguing properties (cf. [2]). For instance, every fraction in this series with two neighbours *a/b* and *c/d* can be computed by the easy formula *(a+c)/(b+d)*. This works also for the first one, if you consider the fraction 0/1 as its left neighbour. Actually this property can be used to construct the Farey fractions of order n+1 from those of order n by inserting a new fraction between two "old" ones using the formula above whenever the denominator of the resulting fraction doesn´t exceed n+1. In the example above this leads to

FAREY(7)=[1/7,1/6,1/5,1/4,2/7,1/3,2/5,3/7,1/2,4/7,3/5,2/3,5/7,3/4,4/5,5/6,6/7,1]

(Check it!) Another interesting property of Farey-fractions is the fact that the equation $bc - ad = 1$ holds for two neighbouring Farey-fractions $a/b < c/d$. This could be used (at least in theory) to solve linear Diophantine equations of the form $bx - ay = 1$ ($a \le b$) in natural numbers which do not exceed *b*. Just take $x=c$, $y=d$, where *c/d* is the Farey-fraction following immediately after *a/b* in the series of Farey-fractions of order *b*!

There are a lot of interesting conjectures dealing with Farey-fractions. Let me just mention one of them. Take FAREY(6) from above which has 12 elements. A priori, the "most regular" selection of an ordered 12-element subset of (0,1] would certainly be

[1/12, 1/6, 1/4, 1/3, 5/12, 1/2, 7/12, 2/3, 3/4, 5/6, 11/12, 1]

where all points have the same distance 1/12 from each other. To measure the deviation of the actual arrangement in FAREY(6) from the one above, we could sum up the absolute differences of corresponding elements. In our example this yields

SUM(ABS(v_),v_,FAREY(6) – [1,2,...,12]/12)=4/15

More generally, one could define a function

```
DISCREPANCY(n):=ITERATE(SUM(ABS(v_),v_,
   s_– [1,2,...,DIMENSION(s_)]/DIMENSION(s_)),s_,FAREY(n),1)
```

that measures the deviation of FAREY(n) from the most regular arrangement of an equally sized ordered subset of (0,1]. Here are the first 10 values of this function

VECTOR(DISCREPANCY(n_),n_,1,10)=[0,0,1/6,1/6,11/30,4/15,43/90,61/110,173/315,187/360]

which seem to indicate that the function doesn´t grow very fast. Actually, the conjecture I was talking about claims that its growth is so slow that for every $r > 1/2$ there exists a real constant $C > 0$ such that DISCREPANCY(n) < Cn^r for all *n*. Just in case you want to get your teeth into it, be warned: Surprisingly enough, this conjecture is equivalent to Riemann´s hypothesis (RH for short), one of the most outstanding conjectures in mathematics and therefore it isn´t exactly a pushover!

In its original formulation, RH deals with the zeros of the so-called Riemann´s ζ-function, For Re(z)>1 it can be defined by

$$\zeta(z) = \prod_p 1/(1-p^{-z})$$

where p runs through all prime numbers. If you replace $1/(1-p^{-z})$ in this formula by the corresponding geometric series $1 + 1/p + 1/p^2 + ...$ then by expanding it and using the fundamental theorem of number theory about prime factorizations you get another well-known formula for $\zeta(z)$, namely

$$\zeta(z) = \sum_{n=1}^{\infty} n^{-z} \quad (\mathrm{Re}(z) > 1$$

By a well-known method called "analytic continuation" it is possible to define $\zeta(z)$ for all complex numbers z and the resulting function has no singularities except for a pole of order 1 at $z = 1$. Now RH claims that all zeros of $\zeta(z)$ in the so-called "critical strip" $0 \le \mathrm{Re}(z) \le 1$ lie on the "critical line" $\mathrm{Re}(z) = 1/2$. Just to get an impression what this means you could draw

ABS(ZETA(a + #i x))

in the range $0 \le x \le 100$ for various values of a in [0,1] e.g. a = 0.3, 0.5, 0.8 (unfortunately, this doesn´t work in DfW, so you have to use the DOS-version of DERIVE.) The truth of RH would have many important implications in number theory.

What else is new in NUMBER.MTH? Above all, a bunch of functions that should cover most problems dealing with divisibility. Again they are from my e-mail correspondence with Al Rich based in most cases on functions published in previous "Titbits". (Look there for the definition of these functions and examples.) The only exception is DIVISOR_SIGMA(k,n) which is now defined more generally to be the sum of k-th powers of all positive divisors of n, where k is a nonnegative integer. Furthermore, some notations have been changed slightly.

```
PRIME_FACTORIZATION(n):=VECTOR(ITERATE(IF(PRIME(p_),[p_,k_],
    [p_^(k_/(k_+1)),k_+1],[p_^(k_/(k_+1)),k_+1]),[p_,k_],[f_,1]),f_,FACTORS(FACTOR(n)))

DIVISORS(n):=SORT((ITERATE(IF(DIMENSION(v_)=0,[u_,v_],[APPEND([u_]`*
    [ITERATES(v_ SUB 1 SUB 1*b_,b_,1,v_ SUB 1 SUB 2)]),DELETE_ELEMENT(v_)]),
    [u_,v_],[[1],PRIME_FACTORIZATION(n)])) SUB 1)

DIVISOR_SIGMA(k,n):=PRODUCT(SUM(v_ SUB 1^(k*j_),j_,0,v_ SUB 2),v_,
    PRIME_FACTORIZATION(n))

DIVISOR_TAU(n):=PRODUCT(v_ SUB 2+1,v_,PRIME_FACTORIZATION(n))

EULER_PHI(n):=n*PRODUCT(1-1/v_ SUB 1,v_,PRIME_FACTORIZATION(n))

MOEBIUS_MU(n):=PRODUCT(IF(PRIME(k_),-1,0),k_,FACTORS(FACTOR(n)))

SQUAREFREE(n):=IF(MOEBIUS_MU(n)=0,false,true)

CYCLOTOMIC(n,x):=PRODUCT((x^(n/d_)-1)^MOEBIUS_MU(d_),d_,DIVISORS(n))
```

As you might remember, the absence of those functions was the main point of criticism in the last issue. At last they are available now! Unfortunately, space is running out and so I have to continue with the introduction of the new functions the next time. What do you think of them.? E-mail me, if you know of any numbertheoretic function that might be of general interest but is still missing! (j.wiesenbauer@tuwien.ac.at)

[1] Adler A.-Coury J.E., The Theory of Numbers, Jones and Barlett Publishers, 1995

[2] Hardy G.H.-Wright E.M., The Theory of Numbers, Oxford University Press, 4 ed., 1965

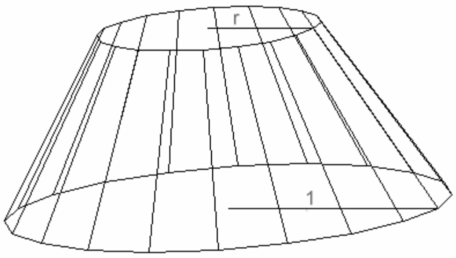
[3] Nöbauer W.- Wiesenbauer J., Zahlentheorie, Prugg-Verlag, Eisenstadt, 1981

[4] Wiesenbauer J., Number Theory with DERIVE, Some Suggestions for Classroom Teaching, in DERIVE in Education, Chartwell-Bratt, 1994

End Point vs. Interior Extrema

Edward T. White, Frostburg, MD
presented by and with TI-92 version by
Carl Leinbach, Gettyburg, PA

In this paper we examine the problem of finding the lateral surface area of a frustrum of a cone having its bottom radius fixed as 1 unit, its top radius r and a fixed height h, as r varies from 0 to 1.

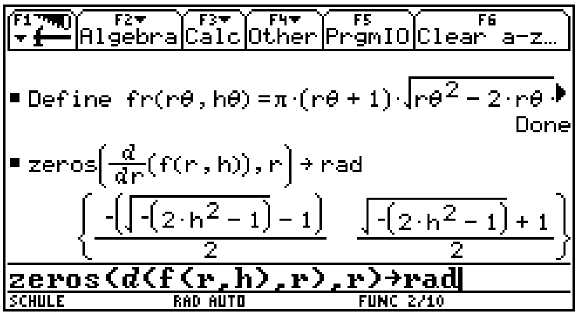


For what value of r is the surface area a maximum?

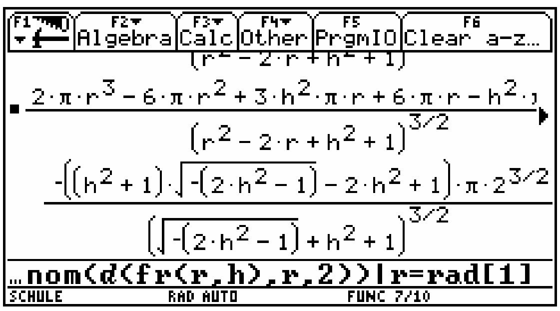
We proceed in the usual way and define a function of r and h which we call Fr, for frustrum

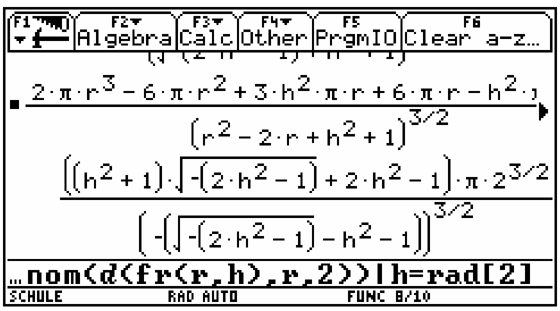
$$\mathrm{Fr}(r,h) \;=\; \pi\,(r+1)\sqrt{h^2+(r-1)^2}$$

The following screen shows the well known process for solving the problem as posed. We differentiate Fr with respect to r, set the result equalt to zero and solve for r. *(To overcome difficulties with the error message "Circular definition" later on I use rθ and hθ as variable names in the function definition. JB)*



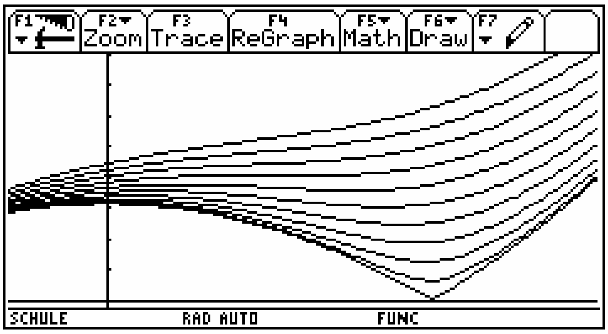
It is apparent that of the two solutions shown in the figure, the first is less than the second. Further investigation shows that when $h \le \frac{1}{\sqrt{2}}$ the interior maximum value of Fr(r,h) lies to the left of its interior minimum value. In the figures shown below rad[1] is the value to the left and rad[2] ist the value to the right.





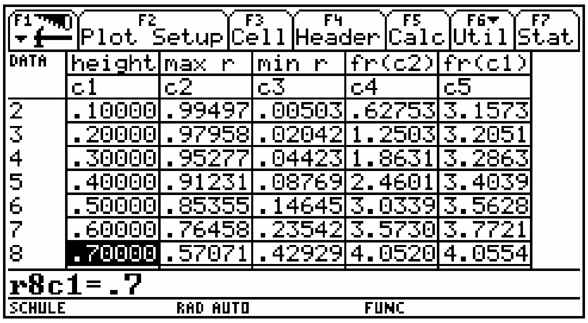
Given the restrictions on h, it is obvious that when r = rad[1] the second derivative is negative and when r = rad[2] the second derivative is positive. (You may want to convince yourself that the denominator is a positive real number in both cases.)

Now comes the interesting part.Let´s look at the lateral surface areas for various choices of h between 0 and 1. The following figure graphs Fr(r,h) for h = 0, 0.1, 0.2, ..., 1 in a window size [-0.2, 1.5] by [-0.2, 8]. *(Use the* [Y=] *facility. You can also create the family of curves with* **seq(fr(x,h),h,0,1,0.1)**, *but plotting is very slow and you are not able then to determine the max and min-values using the* F5*-Math-tool. JB)*



It can be seen that after a certain value of h the minimum for the lateral surface area is at r = 0, a cone, and the maximum is at r = 1, a cylinder. However, there are values of h in between where the maximum and minimum occur for a true frustrum, i.e. one that is neither a cone nor a cylinder. The maximum value starts on the left and as h increases "slides" to the right. The action of the minima is just the opposite. Does the switch from interior maxima and minima occur when these points cross?

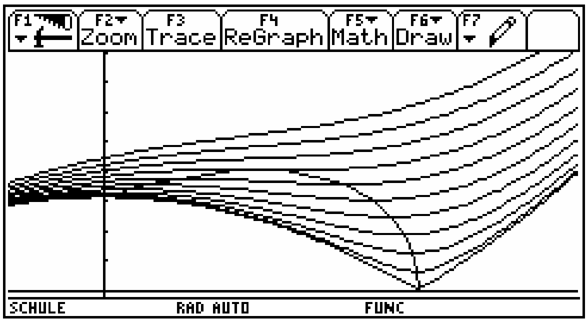
This seems like an intriguing question. Let´s investigate! First let´s look at the maximum and minimum points for the values of h for which we have drawn graphs. The figures below show a table generated in the Data/Matrix editor and then the line graphs of the data in column 4 versus column 2 and in column 5 versus column 3 using the Plot Setup option. These line graphs are superimposed on our original plot.



DATA	height	max r	min r	fr(c2)	fr(c1)
	c1	c2	c3	c4	c5
2	.10000	.99497	.00503	.62753	3.1573
3	.20000	.97958	.02042	1.2503	3.2051
4	.30000	.95277	.04423	1.8631	3.2863
5	.40000	.91231	.08769	2.4601	3.4039
6	.50000	.85355	.14645	3.0339	3.5628
7	.60000	.76458	.23542	3.5730	3.7721
8	.70000	.57071	.42929	4.0520	4.0554

r8c1=.7

SCHULE RAD AUTO FUNC

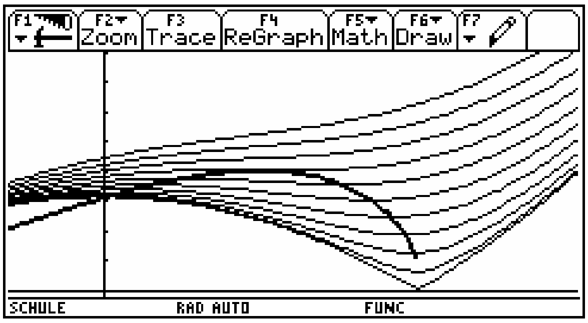


It looks like the hypothesis may have some credence, but what happens in the "gap" in our data?

Returning to the expression for $\frac{\partial}{\partial r}F(r,h)$, we see that it is equal to 0 when

$$h^2 = 2r(1-r).$$

Substituting this value for h^2 into Fr(r,h) we have $Fr(r) = \pi(1+r)\sqrt{1-r^2}$. The graph of this function for $0 \le r \le 1$ superimposed upon the graphs of Fr(r,h) is shown below. It seems to be very similar to the line graphs that were superimposed above. Could it be that the cross over point is at the maximum of Fr(r)?



Once again looks are deceiving. Analysis shows that Fr(r) has a maximum at r = 0.5 and h = $= \frac{1}{\sqrt{2}}$ The numerical data shows that this is not the point where the maxima and minima cross.

In a final effort to put our hypthesis to rest let´s look at the following equations:

Fr(rad[1],h) – Fr(1,h) = 0 (1) and

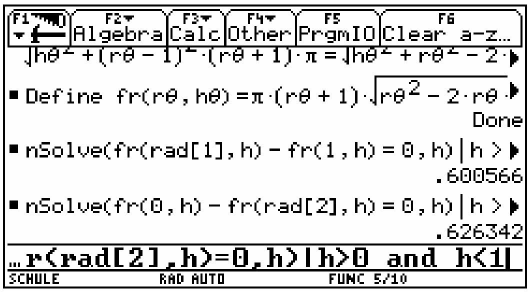
Fr(0,h) – Fr(rad[2],h) = 0 (2).

Solving equation (1) for h should give us the value for h where the maximum shifts to an end point maximum. The solution to (2) should tell us where the minimum shifts. Using the **nsolve(** operator and restricting h to be between 0 and 1, equation (1) gives a value of

h = 0.600566,

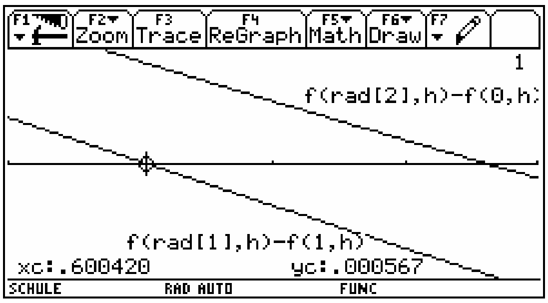
equation (2) yields

h = 0.626342.



These answers are close, but far enough separated that we can be certain that the difference is not merely due to numerical error. To further convince ourselves we graph the left hand side of equations (1) and (2) for 0.59 ≤ h ≤ 0.63. (ymin = – 0.1, ymax = 0.1)

Thus, there is a short interval of h values for which Fr(r,h) (with 0 ≤ h ≤ 1, 0 ≤ r ≤ 1) has a maximum at r = 1 and a minimum in the interior of the interval.



The analysis of this problem required all three aspects of the TI-92. The graphical evidence suggested the problem and helped us to develop a hypothesis, but it certainly could not convince us. The numerical capabilities of the calculator enabled us to look at data and to find solutions to the equations that finally disproved the hypothesis. However, it was the symbolic capability that allowed us to perform the manipulations that gave us the equations that were input for our numerical solvers. Thus all three capabilities, the numerical, the graphical, and the symbolic were important components for doing this analysis.

THE GAUSS-SEIDEL METHOD - A Program for the TI-92.

Leandro Tortosa and Javier Santacruz, Alicante, ESP

Let´s supose that we want to solve the linear system of equations given by

$$A\,X = B$$

and we are assuming that the system has a unique solution. Let´s also suppose that the elements on the main diagonal are not nules; if they don´t, we do some rowswap elementary operation in order not to appear a null element in that place.

Generally speaking, iterative methods are based on performing a partition on the matrix A as

$$A = E - F\,,$$

being E and F square matrices of the same order like A.

By substituting this decomposition of the matrix A in the initial system we´ll have

$$(E - F)\,X = B \;\rightarrow\; E\,X = F\,X + B \;\rightarrow$$

$$X = E^{-1} F\, X + E^{-1} B = T\,X + C\,, \text{ with } T = E^{-1}F \text{ y } C = E^{-1}B.$$

Using the last expression, we can rewrite it and getting the following iterative expression:

$X^{k+1} = E^{-1} F\; X^k + E^{-1} B = T\,X^k + C$

To begin with, we take an initial vector X^0. From this, we generate a sequence of vectors

$$X^{(1)}, X^{(2)}, X^{(3)}, ..., X^{(i)}, ...$$

with the property of converging to the solution of the initial system.

The Gauss-Seidel method is based on the Jacobi method with a slight modification.

The well-known equations that describe the Jacobi method are:

$$(x_j)^{i+1} = \frac{1}{a_{ij}} \cdot \left(b_j - \sum_{k=1}^{j-1} a_{jk} \cdot (x_k)^i - \sum_{k=j+1}^{n} a_{jk} \cdot (x_k)^i \right) \;;\; j = 1, 2, ..., n$$

Without going into details, we can see that the modification we must do to get the Gauss-Seidel method consist of calculating the i-th component of the vector X^{k+1} using the components calculated before in the same step of the iteration, so we will use the components

$$x_1^{k+1}, x_2^{k+1}, ... , x_{i-1}^{k+1}$$

For developing the program in the Ti-92, we use the next expression :

$$Mx = (M - A)x + b \Rightarrow x = (I - M^{-1}A)x + M^{-1}b$$

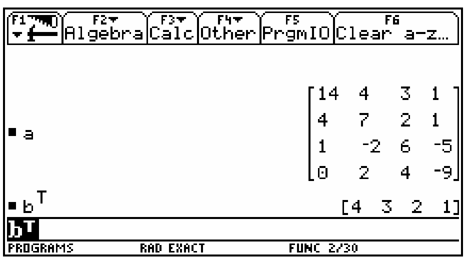
where M is the lower triangular part of A. The iterative expression we get from the last one is

$$x_{n+1} = (I - M^{-1}A)x_n + M^{-1}b$$

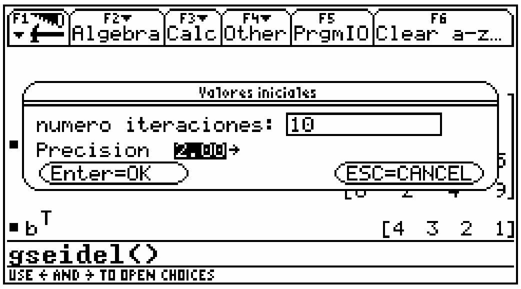
and this is the one we build at the program.

An example :

Let´s consider the system given by Ax = b where A and b are defined like we can see



When we write gseidel() at the editor, the program is executed and the first screen asks us for the number of iterations and precision,



After this step, the next screen asks us for the initial vector, x0. The default option is the null vector ; if we want to change it, we must press 1 and enter de desired one. Finally, it calculates the solution :

A condition for the numerical stability of the solution.

We have given an implementation of the Gauss-Seidel method for Ti-92. But the question arises now is : what´s the matter with the problems of numerical stability when we perform an iterative process, and

We say that a matrix A is strictly diagonal dominant if

$$\left|a_{ii}\right| > \sum_{j=1, j\neq i}^{n} |a_{ij}| \qquad\qquad i = 1,2,\ldots n.$$

We can define the diagonal dominance of a matrix A, δ, like

$$\delta = \max_i \left[\frac{\left|a_{ii}\right|}{\sum_{\substack{j=1 \\ j\neq i}}^{n} \left|a_{ij}\right|} \right].$$

From these definitions, we can say that a matrix is strictly diagonal dominant if δ > 1.

The reason of introducing these concepts is that whenever we prove that in the linear system the matrix of the coefficients is strictly diagonal dominant, we can assure that the system has a solution and the iterative proccess is numerically stable. In other words, when the matrix A is strictly diagonal dominant, we can use the Gauss-Seidel method described above to get the solution, because the iterative proccess is quickly convergent.

We have developed a program Diadom for the TI-92 that shows us if a matrix is strictly diagonal dominant. (We must enter the matrix in the variable m5).

```
Diadom()
Prgm
Local l1,l2,l3,n,s,i,j,k,δ,μ,ε,m,g
dim(m5)→l1
l1[1]→n
l1[2]→s
newList(n)→l2
Disp "Calculo de δω
For i,1,n
```

```
abs(m5[i,i])/(Σ(abs(m5[i,j]),j,1,s)-abs(m5[i,i]))→l2[i]
EndFor
Disp l2
min(l2)→k
max(l2)→δ
If k<1 Then
  Disp "no es estrictamente diagonal"
EndIf
Disp "La dominanza diagonal es"
Disp "δ=",δ
Pause
```

PROGRAM FOR TI-92 FOR SOLVING SYSTEMS OF LINEAR EQUATIONS BY THE GAUSS-SEIDEL METHOD

The program asks you for the number of iterations you want to do and the precision to work. We introduce the matrix of coeficients of the system with label **a** and the vector of the right-hand side of the system like **b.**

```
gseidel()
Prgm
setMode("Exact/Approx","Approximate")
Local n,i,j,k,l,l1,t,siter,pre
{"Fix  Ø","Fix  1","Fix  2","Fix  3","Fix  4","Fix  5","Fix  6","Fix  7","Fix
8","Fix 9","Fix 1Ø","Fix 11","Fix 12"}→l1
dim(a)→n
n[1]→n
newMat(n,1)→xØ
Dialog
Title "Valores iniciales"
Request "numero de iteraciones",siter
DropDown "Precision",seq(string(l),1,Ø,12),pre
EndDlog
pre+1→pre
setMode("Display Digits",l1[pre])
expr(siter)→siter
identity(n)→r
  For i,1,n
    For j,1,i
    a[i,j]→r[i,j]
    EndFor
  EndFor
Disp r
Disp xØ
Disp "Ø = aceptar"
Input "1 = modificar",t
If t=1 Then
   Prompt xØ
EndIf
For u,1,siter,1
   (identity(n)-r^(-1)*a)*xØ+r^(-1)*b→xØ
   If u≠siter Then
     Disp xØ
   EndIf
EndFor
Disp "El resultado es..."
Pause xØᵀ
EndPrgm
```