THE DERIVE - NEWSLETTER #23

ISSN 1990-7079

THE BULLETIN OF THE

DERIVE USER GROUP

Contents:

1 Letter of the Editor

2 Editorial - Preview

3 *DERIVE* User Forum (including JORDAN.MTH)

Nurit Zehavi and David Ben-Chaim
9 Visualizing a Special Line in the 3D Space

G P Speck
13 Some Notes on *DERIVE* 2.6´s Functions and Limits

Josef Böhm
22 The Trigonometric Super Box

Bernhard Wadsack
31 DREIECK.MTH - TRIANGLE.MTH (2)

Carl Leinbach & Marvin Brubaker
37 Carl’s and Marvin’s Laboratory (3)

41 ERF-Functions and *DERIVE*’s Plotting Accuracy

43 **ACDC 2**

Johann Wiesenbauer
46 Titbits (8)

B.Kutzler, B.Waits & F.Demana
49 The TI-92 Corner (W.Pröpper, C.Leinbach, M.Brubaker)

revised version 2010 September 1996

[1] **Exámenes con *DERIVE***, José Luis Llorens Fuster
DERISOFT (FAX: 907 330 371), 214 pages, ISBN 84-921694-1-9

[2] ***DERIVE:* Aplicaciones matemáticas para PC,** Carrillo & Llamas
RA-MA, Madrid, 1994

[3] **Cómo hacer Matematicás con *DERIVE*,** Castro Chadid
Reverté Colombiana, 1993

[4] **Aplicaciones de *DERIVE:* Análisis Matemático-I (Cálculo),** J.L.Llorens Fuster
Servicio de Publicaciones de la Universidad Politécnica de Valencia, ref. 93735. Valencia, 1993

[5] **Aplicaciones de *DERIVE:* Álgebra Lineal (Fundamentos),** J.L.Llorens Fuster
Public. de la Universidad Politécnica de Valencia, ref. 94753. Valencia, 1994

[6] **Introducción al uso de *DERIVE*: Aplicaciones al Àlgebra Lineal y Cálculo Infinitesimal,**
J.L.Llorens Fuster, Public. de la Universidad Politécnica de Valencia, ref. 95638. Valencia, 1995

[7] **Cálculo Matemático con *DERIVE* para PC,** Paulorrogàn & .Pérez,
RA-MA, Madrid, 1994

[9] **Analysis mit *DERIVE*,** Lehr- und Arbeitsbuch für die Jahrgangsstufe 11/1, Hans Jürgen Kayser
Dümmler Verlag, 1996, 157 Seiten, ISBN 3-427-45231-X; Dümmlerbuch 4523

Ich erhielt mehrere Anfragen, da das Buch von **Larry Gilligan "Mastering the TI-92"** für den deutschen Sprachraum offensichtlich schwer erhältlich sein soll. Die ISBN ist 0-9626661-9-X, erschienen bei GILMAR Publishing, P.O.Box 6376, Cincinnati, OH 45206, USA. Sie können das Buch aber noch bequemer erwerben bei **bk-teachware, FAX ++43 (0)7236 6065 80**.

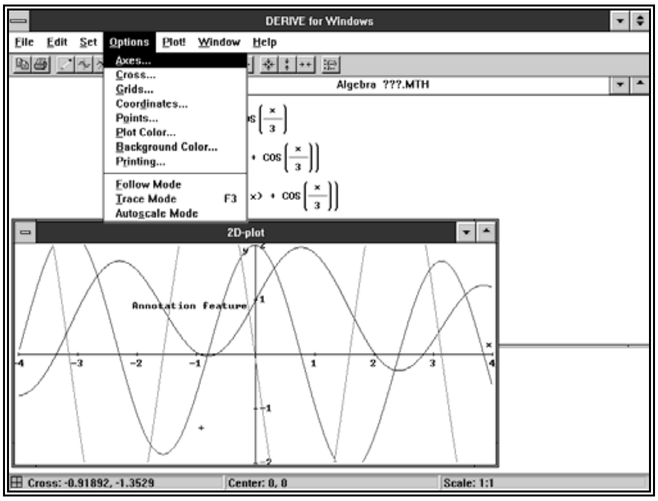
Exchange for DERIVE - Teaching materials in the DNL.

The wheel has not to be invented twice.

Börse für DERIVE Unterrichts-materialien im DNL.

Das Rad muss nicht zweimal erfunden werden.

I can offer:
Binomial Theorem, GCD & LCM, System of coordinates, Modelling Word problems with *DERIVE* (all in English and German*),* SET.EXE, MENGE.EXE (a program to teach and to practise operations with sets including VENN diagrams, in English and German).
I have produced a paper „Einführung in die Matrizenrechnung mit *DERIVE*“ - in German. If you are interested I would send you the paper on a diskette in MSWord6-format.



This is DfW
DERIVE for WINDOWS.

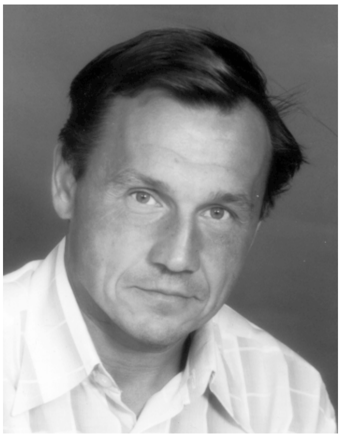
The first screen shot using DfW, demonstrating an ALGEBRA- and a PLOT window.

One of the new features: You are able to add annotations to your plots.

Liebe DUG Mitglieder,

Es war wirklich schön, so viele von Ihnen in Bonn zu treffen oder persönlich kennen zu lernen. Fast alle, die zum ersten Mal eine DERIVE Konferenz besuchten, waren einmal mehr begeistert von der familiären Atmosphäre, vom hohen Niveau und vom ungebrochenen Enthusiasmus der Teilnehmer. Sie haben es genossen, mit David Stoutemyer, mit Bert Waits, um nur zwei zu nennen, und vielen anderen, die sie aus dem DNL kennen, zu sprechen und zu diskutieren. Ich wurde einige Male angesprochen: "Josef, zeige mir bitte den ..., ich möchte doch wirklich wissen, wer das ist!"

Nicht nur J.Wiesenbauer nahm einige Ideen von Bonn nach Hause. Auch Josef Lechner und Eugenio Roanes haben ihre Kontakte vertieft und sind nun dabei, gemeinsam, LOGO in DERIVE zu implementieren. Leider konnte ich bei so vielen Vorträgen nicht, die mich interessiert hätten nicht dabei sein, aber dafür freue ich mich auf den Tagungsband.



Abschließend möchte ich namens der DUG den beiden Organisatoren, Bärbel Barzel und Leo Klingen für Ihre große Mühe herzlich danken.

Vor Bonn hatte ich die Gelegenheit, an einer T^3-Ausbildung - Teachers Training for Technology - in Columbus, Ohio, teilzunehmen. Diese, von B.Waits und F.Demana gegründete Initiative soll nun unter anderem auch auf Europa ausgedehnt werden. Bei dieser Gelegenheit darf ich Sie ersuchen, TI-92 Beiträge an Bert Waits oder an mich zu schicken. (Vergessen Sie aber bitte nicht das klassische DERIVE!)

Thomas Weth kommt mit seinem Kurvenlexikon wieder im nächsten DNL zu Wort. Gerade bei seinen Beiträgen lässt sich eine sinnvolle Verbindung zwischen einem CAS und der interaktiven Geometrie sowohl auf dem PC als auch auf dem TI-92 herstellen.

Ich möchte Sie nochmals darauf hinweisen, dass Sie die .MTH-files zu diesem DNL von SWHH´s web page herunter laden können. Danke, Al Rich.

Vom renommierten Verlag BIRKHÄUSER AG wurde ich gebeten, für unsere deutschsprachigen Mitglieder eine Information über die Fachzeitschrift "Elemente der Mathematik" beizulegen. Diesem Wunsch komme ich gerne nach.

Bis zum nächsten DNL - der „Weihnachtsnummer" mit einigen Zugaben - grüßt Sie herzlichst

Josef Böhm

Nur für Eingeweihte: Habt Ihr das Mäppchen gefunden?

Dear DUG Members,

It was really nice to meet so many of you in Bonn and to make your acquaintance. Most of those who attended a DERIVE Conference the first time were fascinated from the familiar atmosphere, the high level and the unbroken enthusiasm of the participants. They enjoyed talking, discussing and joking with David Stoutemyer and Bert Waits, to mention two people, and with many others who they were knowing from the DNL. It happened several times that I was asked "Oh, Josef, please show me ...., I really would like to know who he (she) is!"

Not only Johann Wiesenbauer took some ideas back home. Josef Lechner and Eugenio Roanes for example, have renewed their contacts and are now busy to implement LOGO in DERIVE. Unfortunately I was not able to attend all the lectures I was interested in. But I am looking forward to studying the Proceedings.

Finally I want to express many thanks on behalf of the DUG to the organizers, Bärbel Barzel and Leo Klingen for their tremendous efforts to make our stay in Bonn as pleasant as possible.

The week before Bonn I had the possibility to attend a T^3 - Conference - Teachers Training for Technology - in Columbus, Ohio. T^3, founded by B.Waits and F.Demana will be established now in Europe, too. On this occasion I’d like to invite you to submit TI-92 papers to B.Waits, B.Kutzler or to me. (But don’t forget the classic DERIVE!)

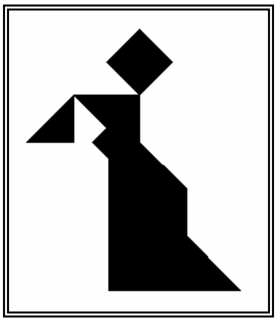
Thomas Weth will continue his "Lexicon of Curves" in the next issue. Especially his articles are excellent examples how to link a CAS with interactive geometry which can be realized now on a PC as well as on the TI-92.

I want to point out once more that you can download the .MTH-files belonging to this issue from SWHH´s web page. Many thanks, Al Rich.

Until the next DNL - the "Christmas Issue" with some extras -

Sincerely yours

Josef Böhm



Tangrams
with *DERIVE*
(in one of the next ACDCs)

The *DERIVE-NEWSLETTER* is the Bulletin of the *DERIVE User Group*. It is published at least four times a year with 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.

Editor: Mag. Josef Böhm
A-3042 Würmla
D´Lust 1
Austria
Phone/FAX: 43-(0)660 3136365
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.

Preview: Contributions for the next issues

Graphic Integration, Linear Programming, Böhm, AUT
LOGO in DERIVE, Lechner & Roanes Lozano, AUT & ESP
3D-Geometry, Reichel, AUT
Parallel- and Central Projection, Böhm, AUT
Algebra at A-Level, Goldstein, UK
Tilgung fremderregter Schwingungen, Klingen, GER
A Utility file for complex dynamic systems, Lechner, AUT
Examples for Statistics, Roeloffs, NL
Linear Mappings and Computer Graphics, Kümmel, GER
Solving Word problems (Textaufgaben) with DERIVE, Böhm, AUT
DERIVE and ACROSPIN, Schorn & Böhm, AUT/GER
Line Searching with DERIVE, Collie, UK
About the "Cesaro Glove-Osculant", Halprin, AUS
3rd-order Differential Equations, Guitiérez, ESP
Tangrams with DERIVE, Población, ESP

The TI-92 Section, Waits a.o.
and
Setif, FRA; Vermeylen, Belgium; Leinbach, USA; Halprin, AUS
Weth, GER; Wiesenbauer, AUT; Keunecke, GER; Aue, GER;
Weller; GER, Stahl, USA; Mitic, UK; Koth, AUT; and ....

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

Victor Hermans, Voorschoten, Netherlands

We have recently met at the *DERIVE* Conference in Bonn. I liked the atmosphere very much and I have found a few colleagues teaching the same subjects as I do. My favourite interest is financial math. I have recently visited the ICME at Sevilla and last year I was at the CIAEM Conference at Berlin and there was no lecture on financial math. The subject seems too practical for mathematicians and too difficult for economists. So my question is: could we start a user group of financial math teachers, whatever the platform? Personally, I have tried to do this in Holland - with Hewlett-Packard financial calculators - with no success. People are just not interested in any machine in the classroom. By the way; there are no graphic calculators and there is no *DERIVE* in Dutch secondary education, and there will not be anything like that on the central examinations for the next four years.

.....

......

***DNL:**. I think we could establish a "Financial Math Working Group" within the DUG if there are other interested members, too. Or we start the Group as a pair of enthusiasts. I would like to ask you for a "pilot contribution" to make clear some topics for this group. As I am a teacher on a "Handelsakademie" (= "College for Business Administration") I am very interested in your idea and I am looking forward to hearing from you in the near future.*

Jan Vermeylen, Kapellen, Belgium

Have you got any experience with the link you can make between *DERIVE* and the programming language C++? Where could I find more information or examples?

***DNL:** Are there any members with experience in combining DERIVE results and C++-applications. I used very successfully the linkage to BASIC programming ACD.EXE to convert DERIVE results into ACD-files for ACROSPIN. I'll present this program in the next DNL. Any other applications would be appreciated.*

George Feeman, Al Ain, United Arab Emirates

We have come across this bug in Ver.3.00 and 3.10:

$\frac{2}{3+x}-\frac{2}{x}$ Simplifies to $-\frac{6}{x(x+3)}$

$\frac{2}{3+x}-\frac{3}{x+1}$ Simplifies to $-\frac{x+7}{(x+1)(x+3)}$ and so on, but

$\frac{2}{3+x}-\frac{3}{x}$ **DOES NOT** Simplify - the output is the same expression.

Can you tell me why? Why this exception to this simple algorithm?

***DNL:** At the occasion of the Bonn DERIVE Conference I discussed this "bug" simplifying the sum of two simple rational expressions with David Stoutemyer. By the way: that's not a "bug" of only DERIVE 3.x versions. DERIVE 2.58 returns the same results. That's a summary of David's answer:*

Computer Algebra is among others the art of recognizing and using patterns. DERIVE always looks for patterns to apply any simplification. (Josef's question: "What is the definition of simple???")

In $\frac{2}{3+x}-\frac{2}{x}$ *DERIVE recognizes the same numerators and then tries to apply a rule.*

In $\frac{2}{3+x}-\frac{3}{x+1}$ *DERIVE discovers two denominators of the form* **(x + constant)** *and then applies the rule to add two fractions using common denominators, but ....*

in $\frac{2}{3+x}-\frac{3}{x}$ *DERIVE cannot find any pattern:* $\frac{3}{x}$ *is treated rather as* $3\,x^{-1}$ *than as* $\frac{3}{x+c}$

Maybe you will not find this explanation very satisfactory. Nevertheless it is always interesting to look behind the curtain of a CAS.

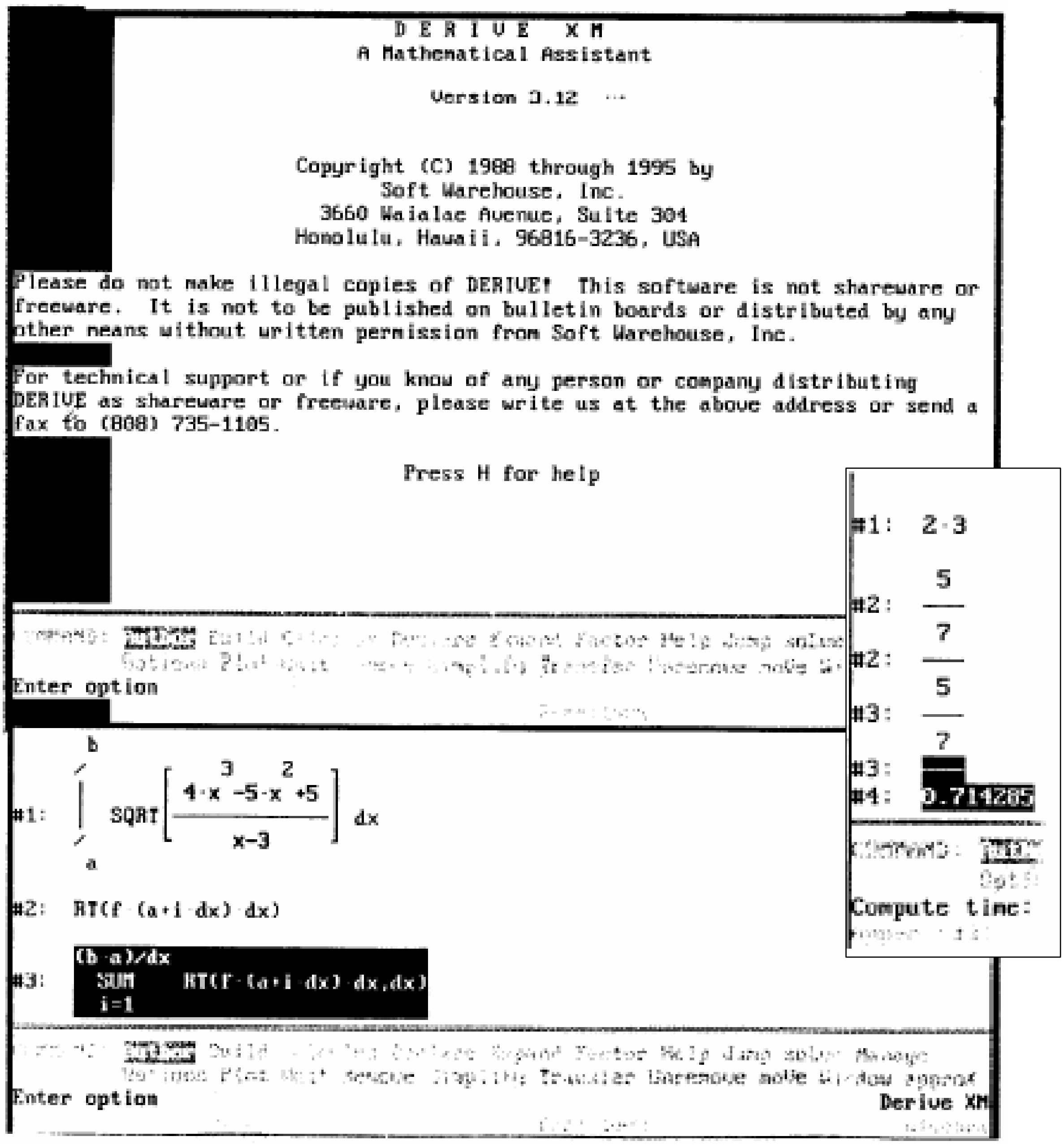
Helmut Wunderling, Berlin, Germany

Anbei Bildschirmausdrucke von *DERIVE*. Die oberen Bilder zeigen, wie unter Softwindows auf einem Powermac die Aufbauroutinen des Bildschirms nicht vollständig funktionieren. Das hat zur Folge, dass bei Ausdrücken, die wechselnd ein- und mehrzeilig sind, die Lesbarkeit dahin ist (vgl. Ausschnitt).

Der untere Teil zeigt, dass Version 3.12 die Sonderzeichen für Integrale, Wurzeln, Summen usw. nicht bereitstellt und zwar auch auf PC (z.B. Hewlett Packard) und das, obwohl die Version 2.59 z.B. das nun wieder anstandslos tut.

Ich bräuchte aber für Veröffentlichungen gute Bilder. Was kann ich tun??

Attached you can find screen dumps of *DERIVE*. The upper pictures (full screen and part of the screen) show that using Softwindows on a Powermac the building up routines of the screen will not work properly. From this follows that expressions with alternately one and more lines are not readable. The dump on the bottom shows that version 3.12 does not provide the special characters for sum, integral, roots etc, and that even on a PC (Hewlett Packard) although I don't have any problems using 2.59. For a publication I need nice screen dumps. What shall I do???



```
                          D E R I V E   X M
                     A Mathematical Assistant

                          Version 3.12


                  Copyright (C) 1988 through 1995 by
                        Soft Warehouse, Inc.
                    3660 Waialae Avenue, Suite 304
                  Honolulu, Hawaii, 96816-3236, USA

Please do not make illegal copies of DERIVE!  This software is not shareware or
freeware.  It is not to be published on bulletin boards or distributed by any
other means without written permission from Soft Warehouse, Inc.

For technical support or if you know of any person or company distributing
DERIVE as shareware or freeware, please write us at the above address or send a
fax to (808) 735-1105.

                          Press H for help

COMMAND: Author Build Calculus Declare Expand Factor Help Jump solve
         Options Plot Quit Remove Simplify Transfer Unremove moVe W
Enter option
```

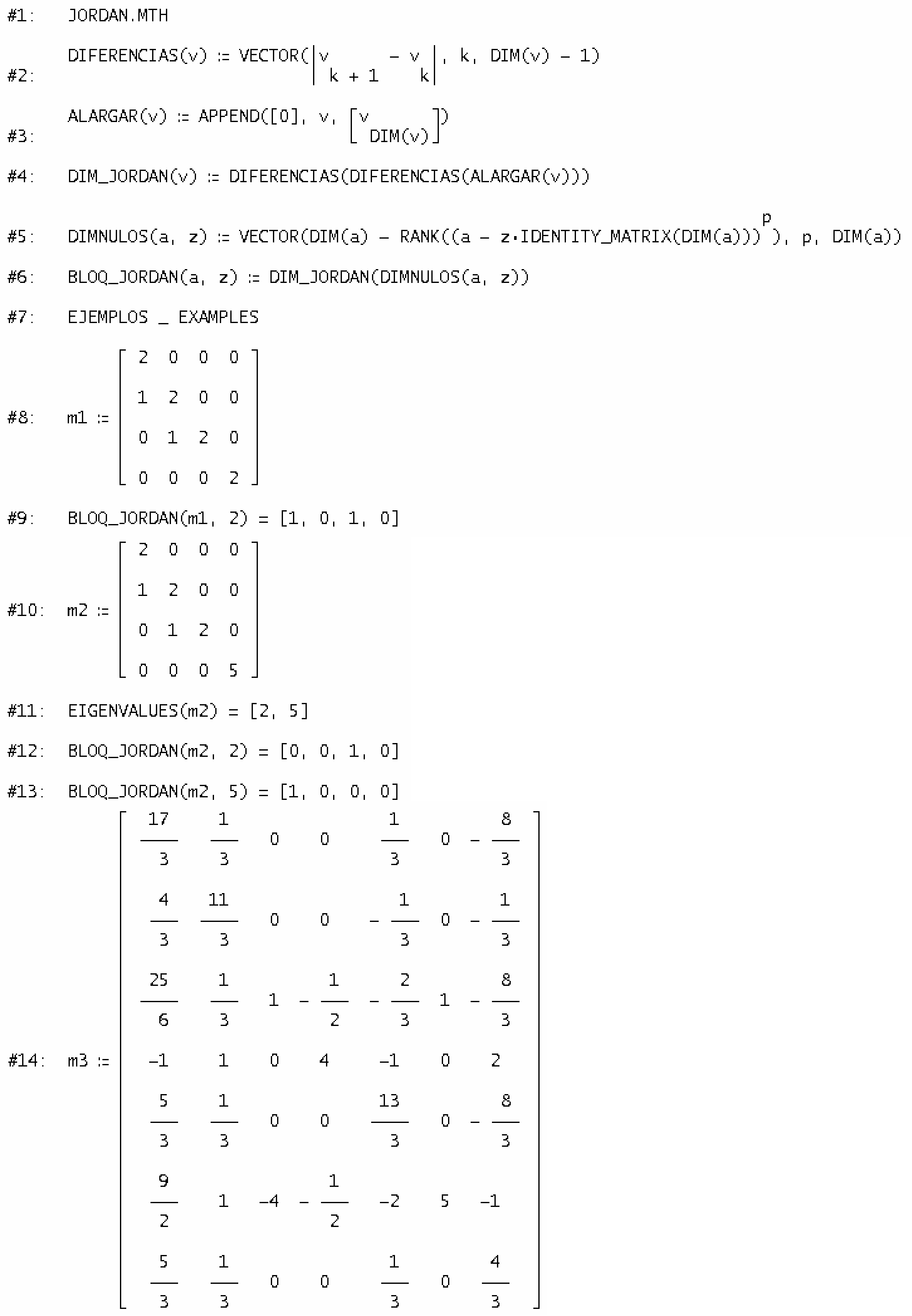
```
#1:  2·3

      5
#2:  ───
      7
#2:  ───
      5
#3:  ───
      7
#3:
#4:  0.714285

COMMAND: Author
         Opti
Compute time:
```

```
         b
        ⌠         ┌    3     2    ┐
#1:     │  SQRT   │ 4·x  -5·x  +5 │ dx
        │         │ ─────────────  │
        ⌡         └     x-3       ┘
         a

#2:  RT(f·(a+i·dx)·dx)

     (b-a)/dx
#3:    SUM     RT(f·(a+i·dx)·dx,dx)
       i=1

COMMAND: Author Build Calculus Declare Expand Factor Help Jump solve Manage
         Options Plot Quit Remove Simplify Transfer Unremove moVe Window approX
Enter option                                                        Derive XM
```

Miguel de Guzmán, Madrid, Spain

I am sending a small program connected with the Jordan form I have described by email. I presume that somebody has thought of this before. Otherwise perhaps it would be useful for some people.

Dada **a** y un autovalor **z** se escribe **BLOQ_JORDAN(a,z)** y resulta un vector de numeros enteros no negativos **[numero de cajas de dimension 1, numero de cajas de dimension 2, ...]**



#1: JORDAN.MTH

#2: $\text{DIFERENCIAS}(v) := \text{VECTOR}\left(\left|v_{k+1} - v_k\right|, k, \text{DIM}(v) - 1\right)$

#3: $\text{ALARGAR}(v) := \text{APPEND}([0], v, [v_{\text{DIM}(v)}])$

#4: DIM_JORDAN(v) := DIFERENCIAS(DIFERENCIAS(ALARGAR(v)))

#5: $\text{DIMNULOS}(a, z) := \text{VECTOR}(\text{DIM}(a) - \text{RANK}((a - z\cdot\text{IDENTITY\_MATRIX}(\text{DIM}(a)))^p), p, \text{DIM}(a))$

#6: BLOQ_JORDAN(a, z) := DIM_JORDAN(DIMNULOS(a, z))

#7: EJEMPLOS _ EXAMPLES

#8: $m1 := \begin{bmatrix} 2 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 2 \end{bmatrix}$

#9: BLOQ_JORDAN(m1, 2) = [1, 0, 1, 0]

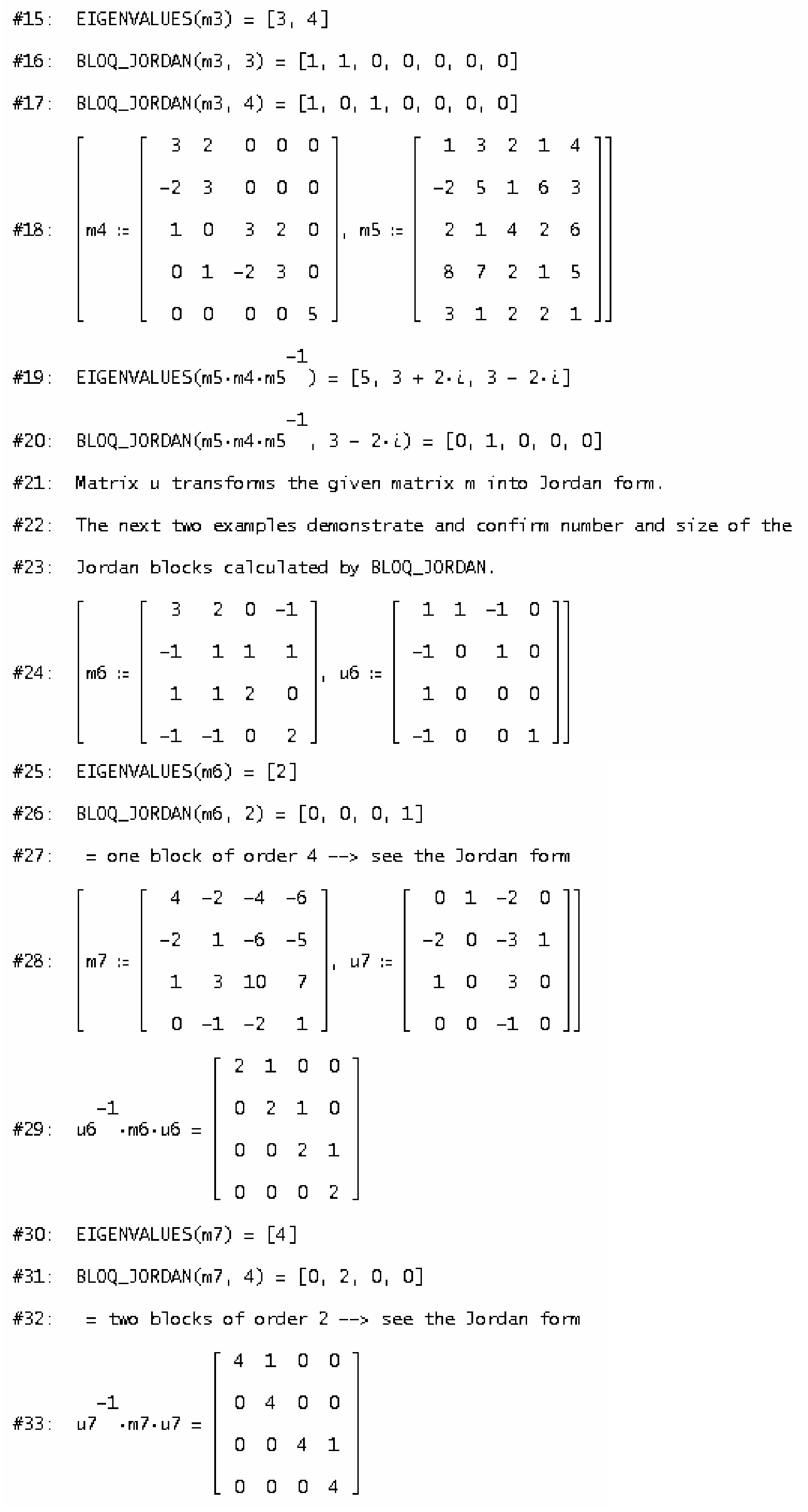
#10: $m2 := \begin{bmatrix} 2 & 0 & 0 & 0 \\ 1 & 2 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix}$

#11: EIGENVALUES(m2) = [2, 5]

#12: BLOQ_JORDAN(m2, 2) = [0, 0, 1, 0]

#13: BLOQ_JORDAN(m2, 5) = [1, 0, 0, 0]

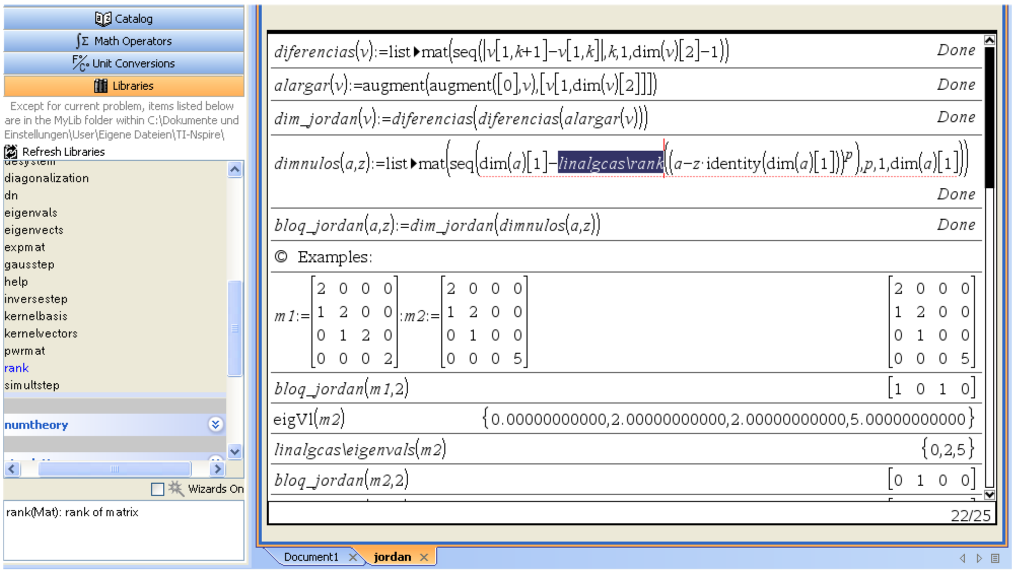
#14: $m3 := \begin{bmatrix} \frac{17}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 & -\frac{8}{3} \\ \frac{4}{3} & \frac{11}{3} & 0 & 0 & -\frac{1}{3} & 0 & -\frac{1}{3} \\ \frac{25}{6} & \frac{1}{3} & 1 & -\frac{1}{2} & -\frac{2}{3} & 1 & -\frac{8}{3} \\ -1 & 1 & 0 & 4 & -1 & 0 & 2 \\ \frac{5}{3} & \frac{1}{3} & 0 & 0 & \frac{13}{3} & 0 & -\frac{8}{3} \\ \frac{9}{2} & 1 & -4 & -\frac{1}{2} & -2 & 5 & -1 \\ \frac{5}{3} & \frac{1}{3} & 0 & 0 & \frac{1}{3} & 0 & \frac{4}{3} \end{bmatrix}$



```
#15:  EIGENVALUES(m3) = [3, 4]

#16:  BLOQ_JORDAN(m3, 3) = [1, 1, 0, 0, 0, 0, 0]

#17:  BLOQ_JORDAN(m3, 4) = [1, 0, 1, 0, 0, 0, 0]

      [        [  3  2   0  0  0 ]          [  1  3  2  1  4 ]]
      [        [                 ]          [                ]]
      [        [ -2  3   0  0  0 ]          [ -2  5  1  6  3 ]]
      [        [                 ]          [                ]]
#18:  [ m4 :=  [  1  0   3  2  0 ], m5 :=   [  2  1  4  2  6 ]]
      [        [                 ]          [                ]]
      [        [  0  1  -2  3  0 ]          [  8  7  2  1  5 ]]
      [        [                 ]          [                ]]
      [        [  0  0   0  0  5 ]          [  3  1  2  2  1 ]]

                                 -1
#19:  EIGENVALUES(m5·m4·m5  ) = [5, 3 + 2·i, 3 - 2·i]

                                 -1
#20:  BLOQ_JORDAN(m5·m4·m5  , 3 - 2·i) = [0, 1, 0, 0, 0]

#21:  Matrix u transforms the given matrix m into Jordan form.

#22:  The next two examples demonstrate and confirm number and size of the

#23:  Jordan blocks calculated by BLOQ_JORDAN.

      [        [  3   2  0  -1 ]         [  1  1  -1  0 ]]
      [        [               ]         [               ]]
      [        [ -1   1  1   1 ]         [ -1  0   1  0 ]]
#24:  [ m6 :=  [               ], u6 :=  [               ]]
      [        [  1   1  2   0 ]         [  1  0   0  0 ]]
      [        [               ]         [               ]]
      [        [ -1  -1  0   2 ]         [ -1  0   0  1 ]]

#25:  EIGENVALUES(m6) = [2]

#26:  BLOQ_JORDAN(m6, 2) = [0, 0, 0, 1]

#27:   = one block of order 4 --> see the Jordan form

      [        [  4  -2  -4  -6 ]          [  0  1  -2  0 ]]
      [        [                ]          [               ]]
      [        [ -2   1  -6  -5 ]          [ -2  0  -3  1 ]]
#28:  [ m7 :=  [                ], u7 :=   [               ]]
      [        [  1   3  10   7 ]          [  1  0   3  0 ]]
      [        [                ]          [               ]]
      [        [  0  -1  -2   1 ]          [  0  0  -1  0 ]]

                          [ 2  1  0  0 ]
                          [            ]
         -1               [ 0  2  1  0 ]
#29:  u6   ·m6·u6 =       [            ]
                          [ 0  0  2  1 ]
                          [            ]
                          [ 0  0  0  2 ]

#30:  EIGENVALUES(m7) = [4]

#31:  BLOQ_JORDAN(m7, 4) = [0, 2, 0, 0]

#32:   = two blocks of order 2 --> see the Jordan form

                          [ 4  1  0  0 ]
                          [            ]
         -1               [ 0  4  0  0 ]
#33:  u7   ·m7·u7 =       [            ]
                          [ 0  0  4  1 ]
                          [            ]
                          [ 0  0  0  4 ]
```

References: ***Elementary Linear Algebra with DERIVE,***
R.J.Hill and T.A.Keagy, Chartwell-Bratt, 1995.

In the following you can see how to transfer this utility to TI-NspireCAS. The rank function is not implemented in the base system but it is available via the *linalgcas*-library. The screenshots below should be informative enough that you can reproduce the procedure on your own.



rank(Mat): rank of matrix

diferencias(*v*):=list▸mat(seq(|*v*[1,*k*+1]−*v*[1,*k*]|,*k*,1,dim(*v*)[2]−1)) *Done*

alargar(*v*):=augment(augment([0],*v*),[*v*[1,dim(*v*)[2]]]) *Done*

dim_jordan(*v*):=*diferencias*(*diferencias*(*alargar*(*v*))) *Done*

dimnulos(*a*,*z*):=list▸mat(seq(dim(*a*)[1]−*linalgcas\rank*((*a*−*z*·identity(dim(*a*)[1]))^p),*p*,1,dim(*a*)[1])) *Done*

bloq_jordan(*a*,*z*):=*dim_jordan*(*dimnulos*(*a*,*z*)) *Done*

© Examples:

$m1:=\begin{bmatrix}2&0&0&0\\1&2&0&0\\0&1&2&0\\0&0&0&2\end{bmatrix}:m2:=\begin{bmatrix}2&0&0&0\\1&2&0&0\\0&1&0&0\\0&0&0&5\end{bmatrix}$ $\begin{bmatrix}2&0&0&0\\1&2&0&0\\0&1&0&0\\0&0&0&5\end{bmatrix}$

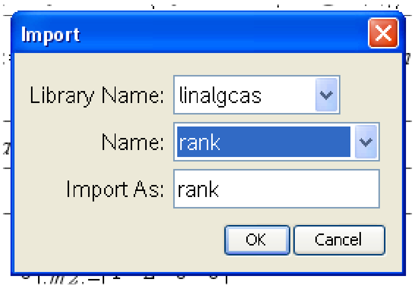
bloq_jordan(*m1*,2) $\begin{bmatrix}1&0&1&0\end{bmatrix}$

eigVl(*m2*) {0.00000000000,2.00000000000,2.00000000000,5.00000000000}

linalgcas\eigenvals(*m2*) {0,2,5}

bloq_jordan(*m2*,2) $\begin{bmatrix}0&1&0&0\end{bmatrix}$

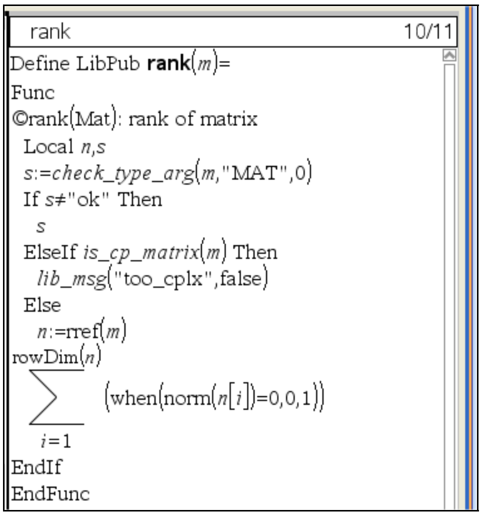
22/25



Library Name: linalgcas

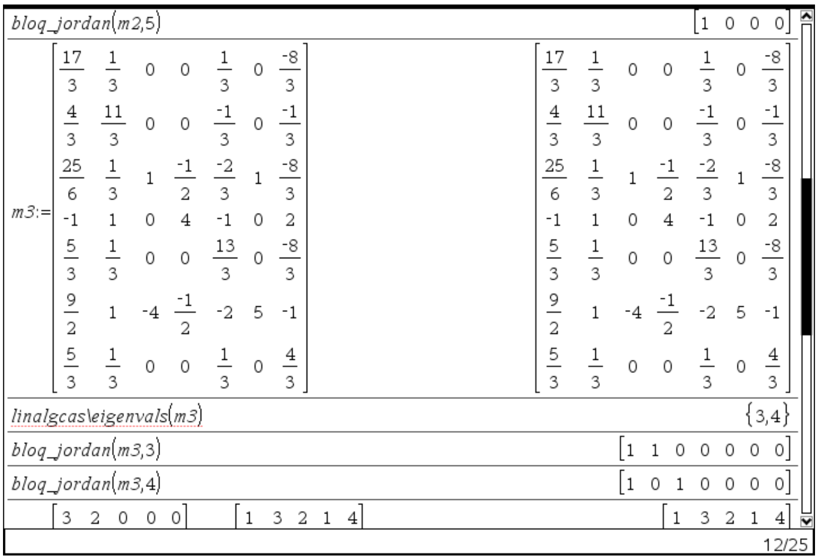
Name: rank

Import As: rank



rank 10/11

```
Define LibPub rank(m)=
Func
©rank(Mat): rank of matrix
 Local n,s
 s:=check_type_arg(m,"MAT",0)
 If s≠"ok" Then
   s
 ElseIf is_cp_matrix(m) Then
   lib_msg("too_cplx",false)
 Else
   n:=rref(m)
```
$$\sum_{i=1}^{\mathrm{rowDim}(n)} (\mathrm{when}(\mathrm{norm}(n[i])=0,0,1))$$
```
EndIf
EndFunc
```



bloq_jordan(*m2*,5) $\begin{bmatrix}1&0&0&0\end{bmatrix}$

$m3:=\begin{bmatrix}\frac{17}{3}&\frac{1}{3}&0&0&\frac{1}{3}&0&\frac{-8}{3}\\\frac{4}{3}&\frac{11}{3}&0&0&\frac{-1}{3}&0&\frac{-1}{3}\\\frac{25}{6}&\frac{1}{3}&1&\frac{-1}{2}&\frac{-2}{3}&1&\frac{-8}{3}\\-1&1&0&4&-1&0&2\\\frac{5}{3}&\frac{1}{3}&0&0&\frac{13}{3}&0&\frac{-8}{3}\\\frac{9}{2}&1&-4&\frac{-1}{2}&-2&5&-1\\\frac{5}{3}&\frac{1}{3}&0&0&\frac{1}{3}&0&\frac{4}{3}\end{bmatrix}$ $\begin{bmatrix}\frac{17}{3}&\frac{1}{3}&0&0&\frac{1}{3}&0&\frac{-8}{3}\\\frac{4}{3}&\frac{11}{3}&0&0&\frac{-1}{3}&0&\frac{-1}{3}\\\frac{25}{6}&\frac{1}{3}&1&\frac{-1}{2}&\frac{-2}{3}&1&\frac{-8}{3}\\-1&1&0&4&-1&0&2\\\frac{5}{3}&\frac{1}{3}&0&0&\frac{13}{3}&0&\frac{-8}{3}\\\frac{9}{2}&1&-4&\frac{-1}{2}&-2&5&-1\\\frac{5}{3}&\frac{1}{3}&0&0&\frac{1}{3}&0&\frac{4}{3}\end{bmatrix}$

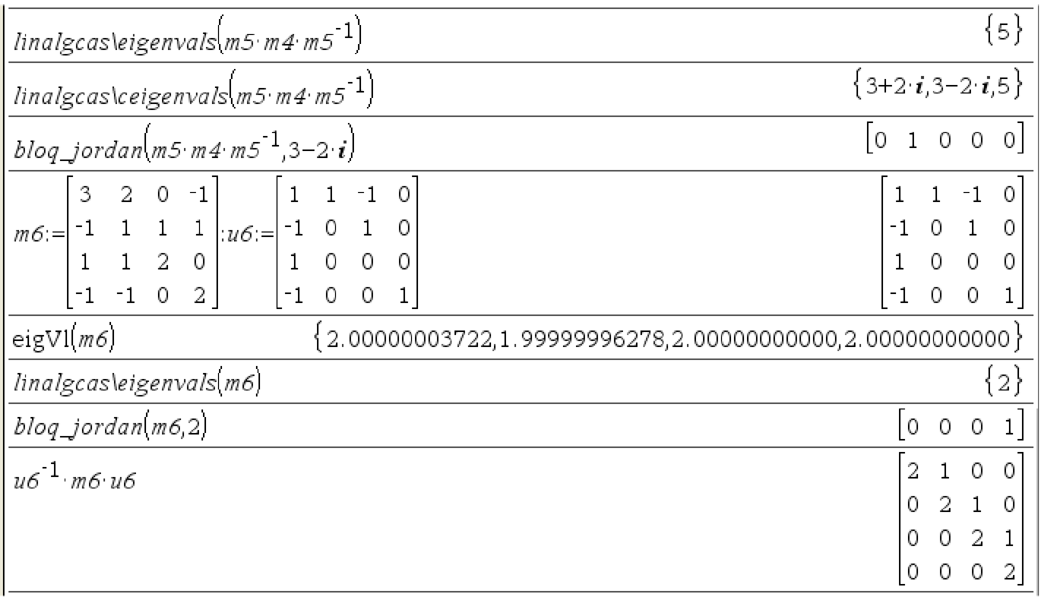
linalgcas\eigenvals(*m3*) {3,4}

bloq_jordan(*m3*,3) $\begin{bmatrix}1&1&0&0&0&0&0\end{bmatrix}$

bloq_jordan(*m3*,4) $\begin{bmatrix}1&0&1&0&0&0&0\end{bmatrix}$

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

12/25



$linalgcas\backslash eigenvals(m5\cdot m4\cdot m5^{-1})$ $\{5\}$

$linalgcas\backslash ceigenvals(m5\cdot m4\cdot m5^{-1})$ $\{3+2\cdot \boldsymbol{i},3-2\cdot \boldsymbol{i},5\}$

$bloq\_jordan(m5\cdot m4\cdot m5^{-1},3-2\cdot \boldsymbol{i})$ $\begin{bmatrix}0 & 1 & 0 & 0 & 0\end{bmatrix}$

$$m6:=\begin{bmatrix}3 & 2 & 0 & -1\\ -1 & 1 & 1 & 1\\ 1 & 1 & 2 & 0\\ -1 & -1 & 0 & 2\end{bmatrix}:u6:=\begin{bmatrix}1 & 1 & -1 & 0\\ -1 & 0 & 1 & 0\\ 1 & 0 & 0 & 0\\ -1 & 0 & 0 & 1\end{bmatrix} \qquad \begin{bmatrix}1 & 1 & -1 & 0\\ -1 & 0 & 1 & 0\\ 1 & 0 & 0 & 0\\ -1 & 0 & 0 & 1\end{bmatrix}$$

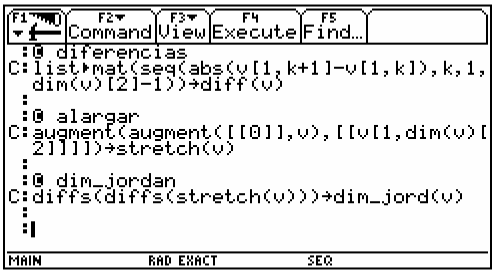
$\mathrm{eigVl}(m6)$ $\{2.00000003722,1.99999996278,2.00000000000,2.00000000000\}$

$linalgcas\backslash eigenvals(m6)$ $\{2\}$

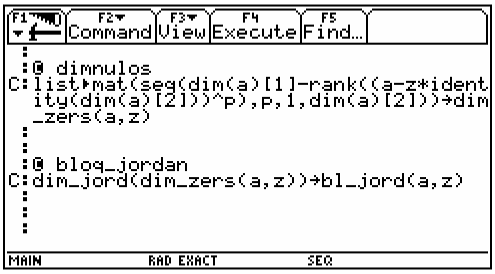
$bloq\_jordan(m6,2)$ $\begin{bmatrix}0 & 0 & 0 & 1\end{bmatrix}$

$$u6^{-1}\cdot m6\cdot u6 \qquad \begin{bmatrix}2 & 1 & 0 & 0\\ 0 & 2 & 1 & 0\\ 0 & 0 & 2 & 1\\ 0 & 0 & 0 & 2\end{bmatrix}$$

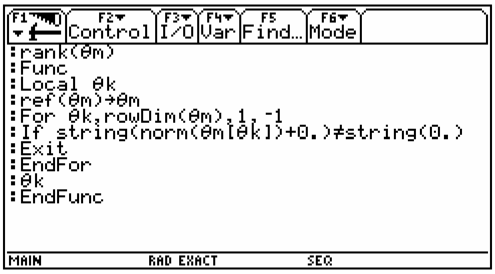
For accomplishing this contribution see the same procedures performed with the TI-92 / Voyage 200.



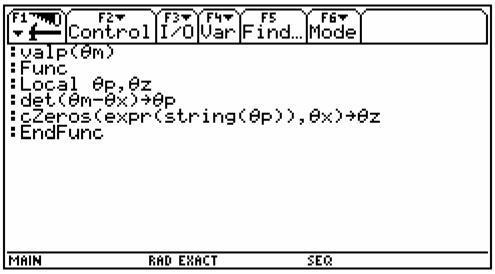
```
 :@ diferencias
C:list▸mat(seq(abs(v[1,k+1]-v[1,k]),k,1,
  dim(v)[2]-1))→diff(v)
 :
 :@ alargar
C:augment(augment([[0]],v),[[v[1,dim(v)[
  2]]]])→stretch(v)
 :
 :@ dim_jordan
C:diffs(diffs(stretch(v)))→dim_jord(v)
 :
 :
```



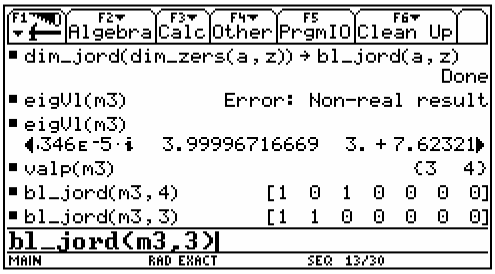
```
 :
 :@ dimnulos
C:list▸mat(seq(dim(a)[1]-rank((a-z*ident
  ity(dim(a)[2]))^p),p,1,dim(a)[2]))→dim
  _zers(a,z)
 :
 :
 :@ bloq_jordan
C:dim_jord(dim_zers(a,z))→bl_jord(a,z)
 :
 :
 :
```



```
:rank(θm)
:Func
:Local θk
:ref(θm)→θm
:For θk,rowDim(θm),1,-1
:If string(norm(θm[θk])+0.)≠string(0.)
:Exit
:EndFor
:θk
:EndFunc
```



```
:valp(θm)
:Func
:Local θp,θz
:det(θm-θx)→θp
:cZeros(expr(string(θp)),θx)→θz
:EndFunc
```



```
■ dim_jord(dim_zers(a,z)) → bl_jord(a,z)
                                     Done
■ eigVl(m3)         Error: Non-real result
■ eigVl(m3)
  ◀.346E-5·i   3.99996716669   3. + 7.62321▶
■ valp(m3)                           {3   4}
■ bl_jord(m3,4)         [1  0  1  0  0  0  0]
■ bl_jord(m3,3)         [1  1  0  0  0  0  0]
bl_jord(m3,3)
```

Visualizing a Special Line in the 3-D Space

Nurit Zehavi & David Ben-Chaim
Weizmann Institute of Science, Israel

The motivation

The second author shared with the first author a paper that he had submitted for publication, entitled, A Special Point in the 2-D Plane and a Special Line in the 3-D Space. The paper explores systems of linear equations that are written in the standard form with coefficients occurring as consecutive terms of arithmetic sequences. He challenged her: Can we prepare an activity for teachers based on this paper using Derive?

The result

It is easy to verify, algebraically and graphically, that the ordered pair (-1, 2) satisfies any linear equation in two variables that when written in the standard form the coefficients are consecutive numbers from an arithmetic sequence.[1)]

The solution of the general 3-by-3 system of linear equations with arithmetic sequences for coefficients ,

$$a1.x + (a1 + d1)y + (a1 + 2d1)z = a1 + 3d1$$
$$a2.x + (a2 + d2)y + (a2 + 2d2)z = a2 + 3d2$$
$$a3.x + (a3 + d3)y + (a3 + 2d3)z = a3 + 3d3$$

is parametric [@1, -2@1 - 1, @1 +2].

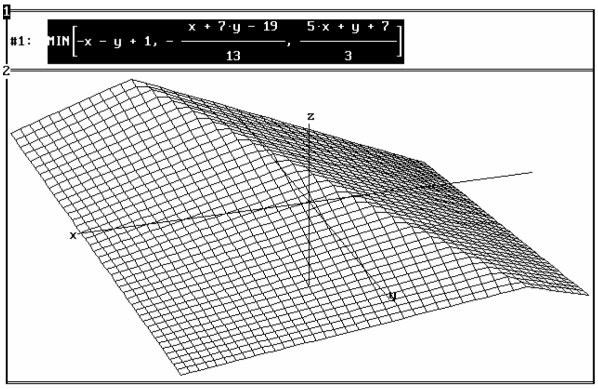
The graphical representation of a linear equation in three variables is a plane. The interpretation of the algebraic solution above is that all the planes that represent equations with "arithmetic coefficients" intersect in a Special Line (t, -2t – 1, t + 2) (choosing t to be the parameter). How can we demonstrate this finding applying Derive?

Derive can place only one surface at a time in the 3D-plot window. However, we superimposed several planes in Acrospin and watched for the intersection line while running Acrospin. The special line in this representation was not very convincing. Coincidentally we received the card from Soft Warehouse with an invitation to "surf" the net to their web page. In the Derive-Questions-and-Answers section it was suggested to plot the minimum of two surfaces in order to see their intersection. Aha!

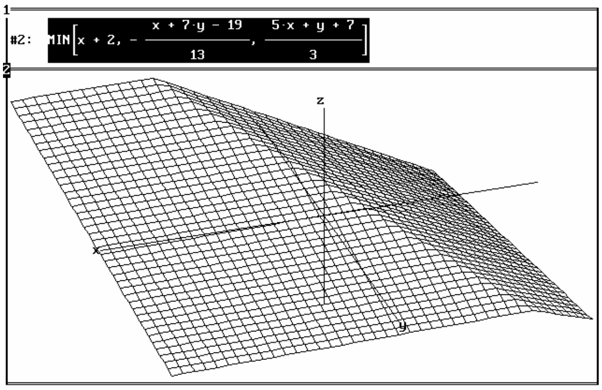
We applied the method to an example taken from Ben-Chaim's paper:

$$x + y + z = 1$$
$$x + 7y + 13z = 19$$
$$5x + y - 3z = -7$$

The equations were modified to the explicit form z = u(x, y) and the minimum of the three planes was plotted (see figure 1). The following figures were plotted by teachers who explored the situation. In figure 2 the first equation was replaced by another one with arithmetic coefficients, x - z = -2, and the special line appeared again. But do we see the same picture? Some teachers studied the 3D plots and compared coefficients, others ran the Acrospin file (it is helpful to plot each of the three planes in a different colour), and a few issued the dot product of the vectors orthogonal to each of the planes and compared the angles between the planes. In figure 3, the first equation was replaced by $z = y$ so that the three planes intersect in one point (-1, 1, 1). Now a question was raised: Is it possible to see the special line even if one of the equations does not fulfil the arithmetic coefficients condition?



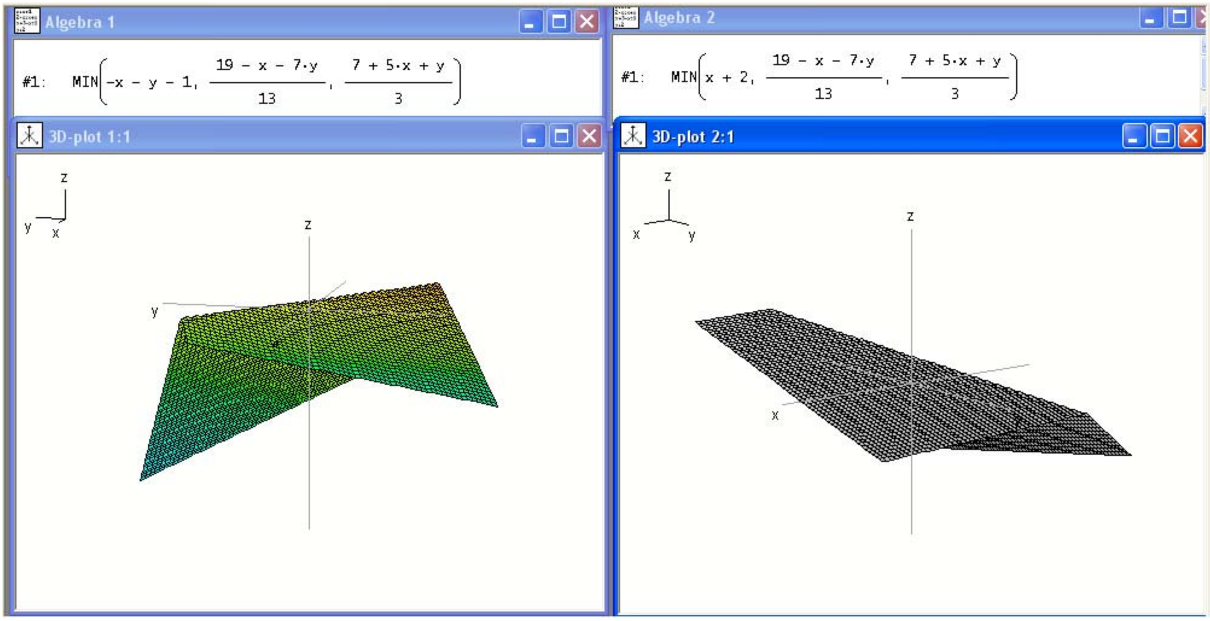
#1: $\mathrm{MIN}\left[-x - y + 1, -\frac{x + 7\cdot y - 19}{13}, \frac{5\cdot x + y + 7}{3}\right]$



#2: $\mathrm{MIN}\left[x + 2, -\frac{x + 7\cdot y - 19}{13}, \frac{5\cdot x + y + 7}{3}\right]$

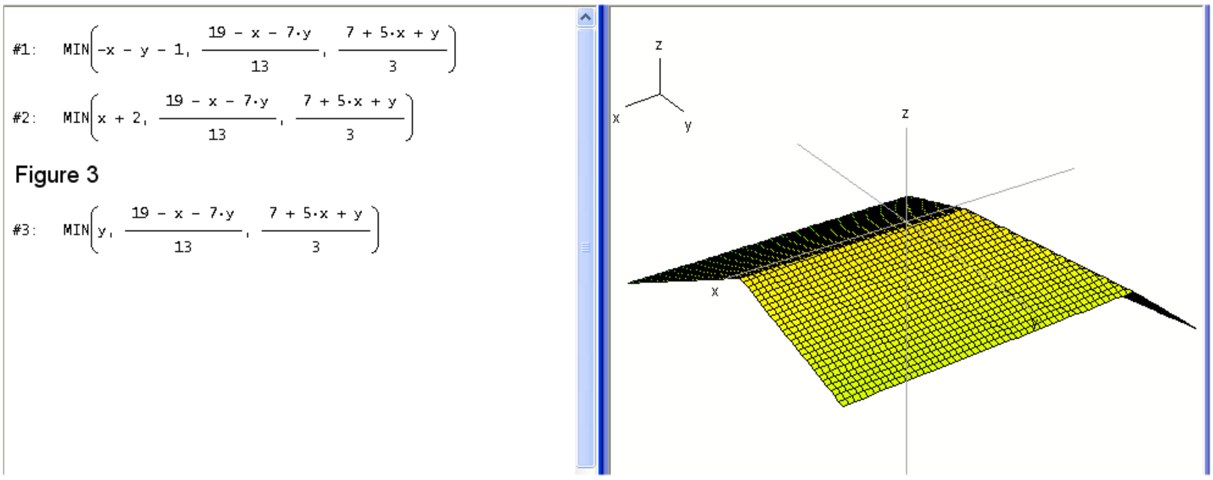
Figure 1 `MIN(-x-y+1,-(x+7y-19)......)`

Figure 2



#1: $\mathrm{MIN}\left(-x - y - 1, \frac{19 - x - 7\cdot y}{13}, \frac{7 + 5\cdot x + y}{3}\right)$

#1: $\mathrm{MIN}\left(x + 2, \frac{19 - x - 7\cdot y}{13}, \frac{7 + 5\cdot x + y}{3}\right)$



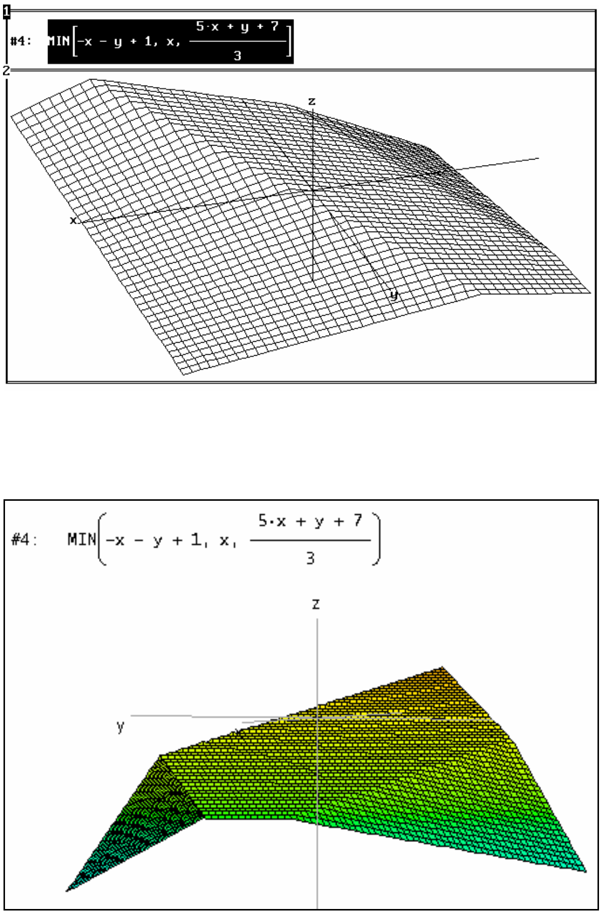
#1: $\mathrm{MIN}\left(-x - y - 1, \frac{19 - x - 7\cdot y}{13}, \frac{7 + 5\cdot x + y}{3}\right)$

#2: $\mathrm{MIN}\left(x + 2, \frac{19 - x - 7\cdot y}{13}, \frac{7 + 5\cdot x + y}{3}\right)$

Figure 3

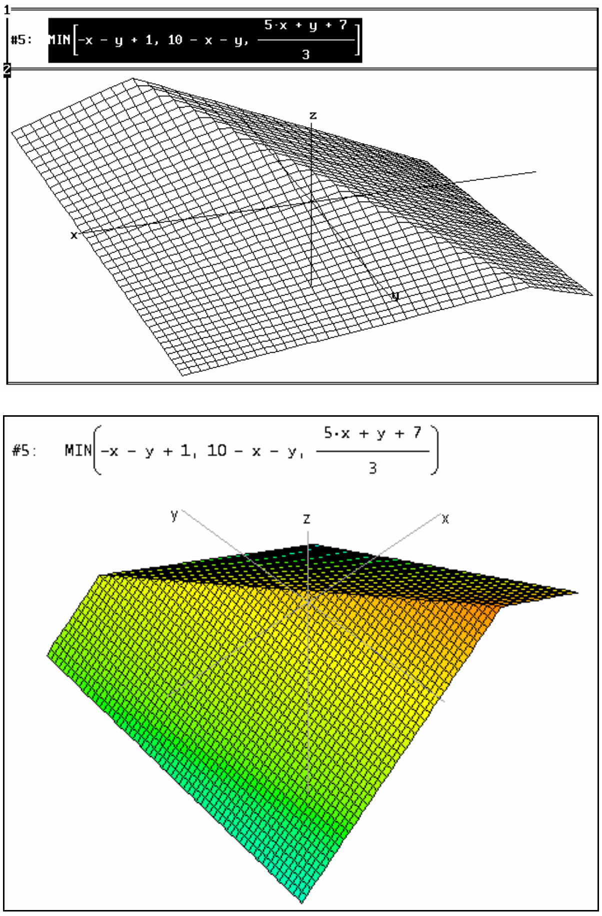
#3: $\mathrm{MIN}\left(y, \frac{19 - x - 7\cdot y}{13}, \frac{7 + 5\cdot x + y}{3}\right)$

The equations , ax + by + cz = d, to consider must be represented by planes that are parallel to the identified special line [0, -1, 2] + t[1, -2, 1] (written in vector notation). Thus the dot product [1, -2, 1].[a, b, c] = a -2b + c must be zero. In figure 4 such equation, z = x, was chosen to replace the middle equation. Not surprisingly, the plot in figure 5 is the same as in figure 1, but the coefficients of x + y + z = 10 are not consecutive numbers of an arithmetic sequence. In fact, the plane that represents x + y + z = 10 is parallel to one of the planes seen in figure 5. In conclusion, the special line activity provides opportunities for teachers to revise and extend their knowledge by using the possibilities offered by the software.



#4: $\mathrm{MIN}\left(-x - y + 1,\ x,\ \frac{5\cdot x + y + 7}{3}\right)$

Figure 4



#5: $\mathrm{MIN}\left(-x - y + 1,\ 10 - x - y,\ \frac{5\cdot x + y + 7}{3}\right)$

Figure 5

Nurit and David mentioned in their paper the ACROSPIN animation of their objects. As some of you might not know how to combine several *DERIVE* 3D-plots in different layers and colours in one ACROSPIN-file I'll explain how to do:

Edit the first plane:

```
pl1:=-x-y+1,
```

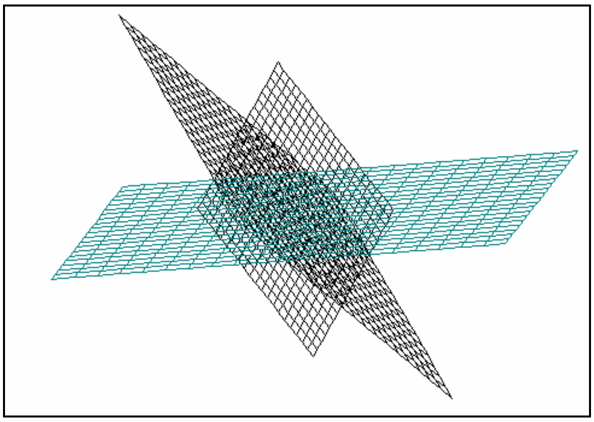
Plot the grid in an appropriate colour (Grids 20) in the 3D-plot window and then press **T**ransfer **A**crospin **S**ave: MYNAME. Now you may **R**un **A**crospin to view your result so far. Switch back to the Algebra window, highlight or edit the second plane:

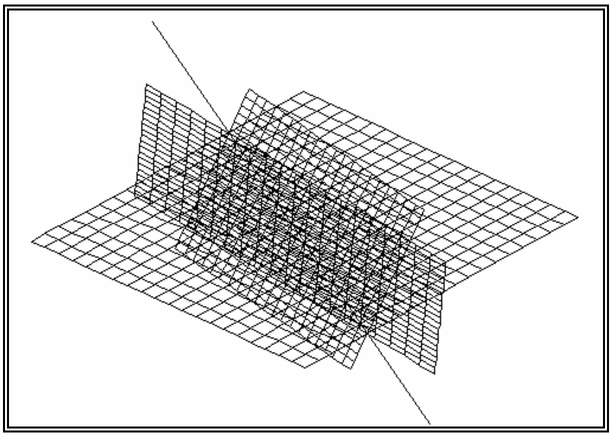
```
pl2:=-(x+7*y-19)/13
```

Plot it again in an appropriate colour. Then Transfer Acrospin Save, confirm MYNAME.ACD pressing · . Skip "Overwrite existing file" pressing N and confirm "Append to existing file" pressing Y. Then go back once more to the Algebra window and repeat the same procedure once more with the third plane.

```
pl3:=(5*x+y+7)/3
```

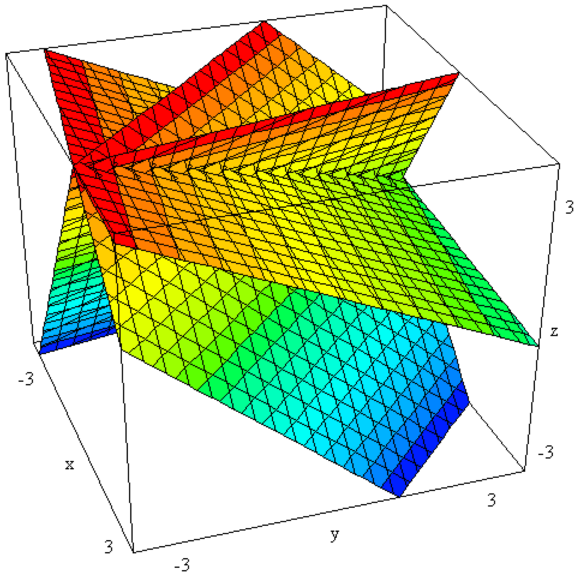
Now run Acrospin.





The right Acrospin animation containing the planes together with the intersection line was produced using my ACD.EXE tool. (Next DNL!!). Josef

[1] E. Sawada, Extending Algebra Concepts with Technology, DNL#12, December 1993



Some Notes on DERIVE 2.6´s Functions and Limits

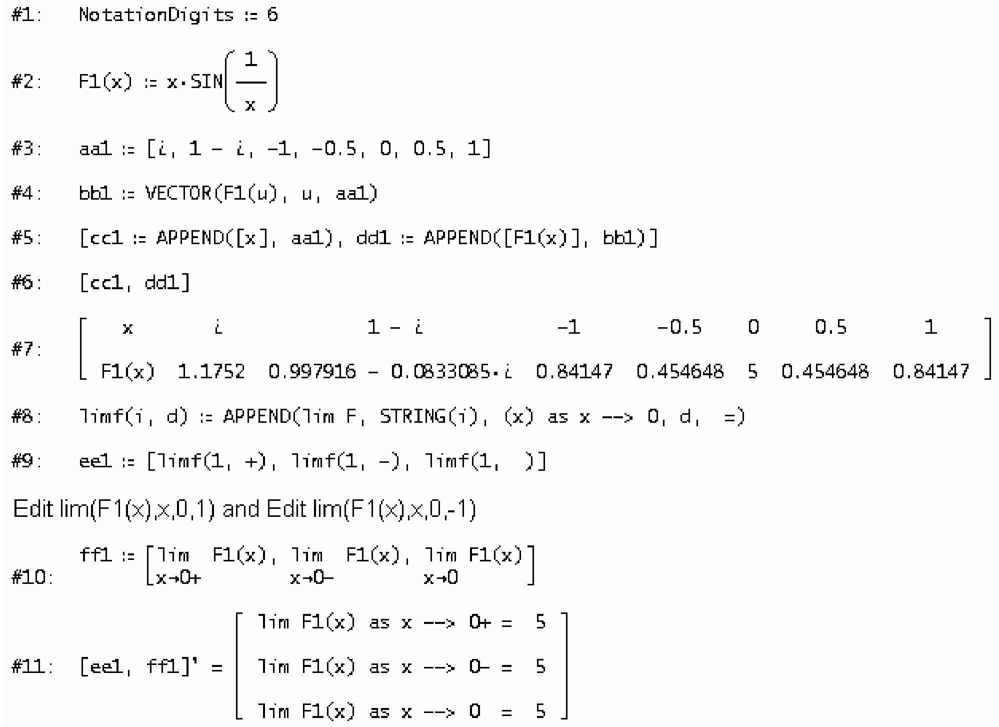
G P Speck, Wanganui, New Zealand

Using *DERIVE* 2.6, we have the following results for specific functions, as shown on the printout belonging to these examples:

(If there are any differences using other - more recent - *DERIVE* versions I've added them. Maybe that Al Rich or David Stoutemyer will have some comments on Mr Speck's paper?)

Example 1: F1(x) = x SIN(1/x). For this function F1(x), all of the *DERIVE* 2.6 functional values and limits agree with the classical results; e.g.,

lim F1(x) as x → 0+ = lim F1(x) as x → 0– = lim F1(x) as x → 0.



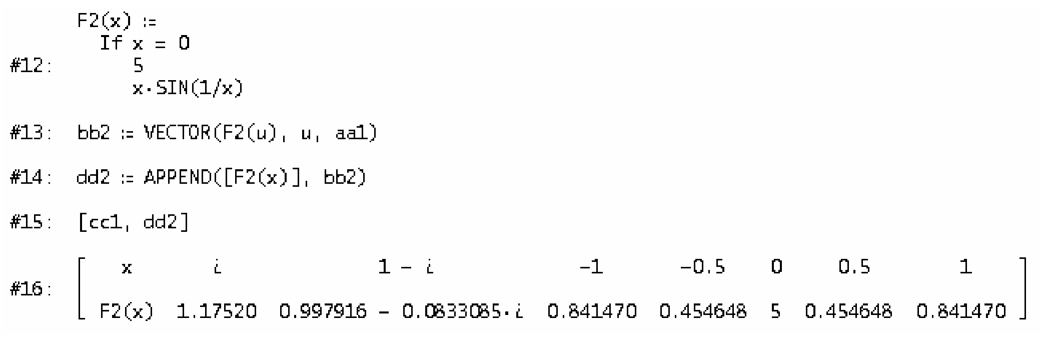
```
#1:   NotationDigits := 6

                    ⎛ 1 ⎞
#2:   F1(x) := x·SIN⎜ ─ ⎟
                    ⎝ x ⎠

#3:   aa1 := [î, 1 - î, -1, -0.5, 0, 0.5, 1]

#4:   bb1 := VECTOR(F1(u), u, aa1)

#5:   [cc1 := APPEND([x], aa1), dd1 := APPEND([F1(x)], bb1)]

#6:   [cc1, dd1]

      ⎡   x       î             1 - î            -1      -0.5     0     0.5       1     ⎤
#7:   ⎢                                                                                 ⎥
      ⎣ F1(x)  1.1752  0.997916 - 0.0833085·î  0.84147  0.454648  5  0.454648  0.84147 ⎦

#8:   limf(i, d) := APPEND(lim F, STRING(i), (x) as x --> 0, d,  =)

#9:   ee1 := [limf(1, +), limf(1, -), limf(1,  )]
```

Edit lim(F1(x),x,0,1) and Edit lim(F1(x),x,0,-1)

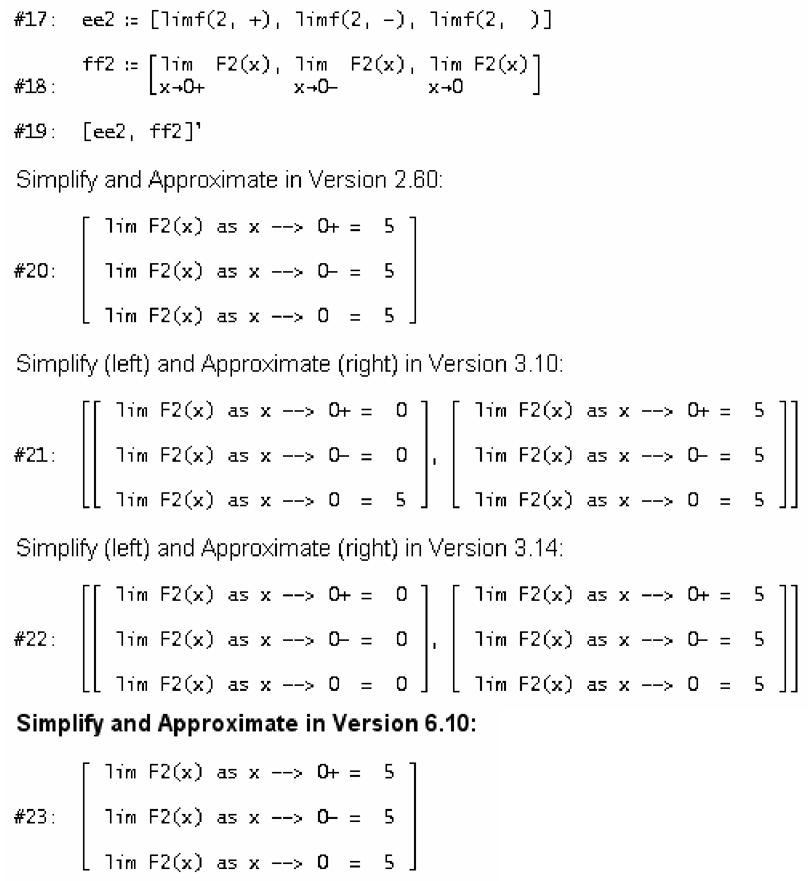
```
      ff1 := ⎡lim  F1(x), lim  F1(x), lim F1(x)⎤
#10:         ⎣x→0+        x→0-        x→0      ⎦

                       ⎡ lim F1(x) as x --> 0+ =  5 ⎤
                       ⎢                            ⎥
#11:  [ee1, ff1]' =    ⎢ lim F1(x) as x --> 0- =  5 ⎥
                       ⎢                            ⎥
                       ⎣ lim F1(x) as x --> 0  =  5 ⎦
```

Example 2: F2(x) = IF(x = 0, 5, x SIN(1/x)). For this function F2(x), we have the following *DERIVE* 2.6 limit results. We see that each of these results DISAGREE with the corresponding classical calculus limit results. each of which is 0 if we work with approXimate.

The function F(x):=IF(x=0,0,x SIN(1/x)) has all limits equal to 0 as x tends to 0, as it is the case for the classical limits.



```
      F2(x) :=
        If x = 0
#12:       5
           x·SIN(1/x)

#13:  bb2 := VECTOR(F2(u), u, aa1)

#14:  dd2 := APPEND([F2(x)], bb2)

#15:  [cc1, dd2]

      ⎡   x       î              1 - î             -1       -0.5     0     0.5       1      ⎤
#16:  ⎢                                                                                     ⎥
      ⎣ F2(x)  1.17520  0.997916 - 0.0833085·î  0.841470  0.454648  5  0.454648  0.841470 ⎦
```



```
#17:  ee2 := [limf(2, +), limf(2, -), limf(2,  )]

      ff2 := ⎡lim  F2(x), lim  F2(x), lim F2(x)⎤
#18:         ⎣x→0+        x→0-        x→0      ⎦

#19:  [ee2, ff2]'
```

Simplify and Approximate in Version 2.60:

```
       ⎡ lim F2(x) as x --> 0+ =  5 ⎤
#20:   ⎢ lim F2(x) as x --> 0- =  5 ⎥
       ⎣ lim F2(x) as x --> 0  =  5 ⎦
```

Simplify (left) and Approximate (right) in Version 3.10:

```
       ⎡⎡ lim F2(x) as x --> 0+ =  0 ⎤  ⎡ lim F2(x) as x --> 0+ =  5 ⎤⎤
#21:   ⎢⎢ lim F2(x) as x --> 0- =  0 ⎥, ⎢ lim F2(x) as x --> 0- =  5 ⎥⎥
       ⎣⎣ lim F2(x) as x --> 0  =  5 ⎦  ⎣ lim F2(x) as x --> 0  =  5 ⎦⎦
```

Simplify (left) and Approximate (right) in Version 3.14:

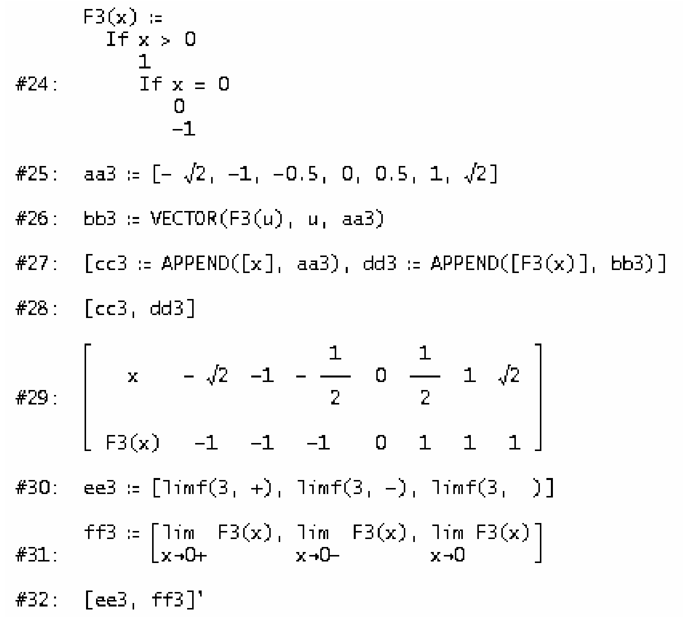
```
       ⎡⎡ lim F2(x) as x --> 0+ =  0 ⎤  ⎡ lim F2(x) as x --> 0+ =  5 ⎤⎤
#22:   ⎢⎢ lim F2(x) as x --> 0- =  0 ⎥, ⎢ lim F2(x) as x --> 0- =  5 ⎥⎥
       ⎣⎣ lim F2(x) as x --> 0  =  0 ⎦  ⎣ lim F2(x) as x --> 0  =  5 ⎦⎦
```

Simplify and Approximate in Version 6.10:

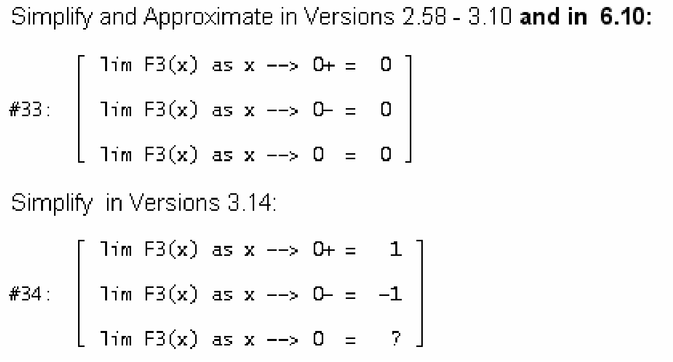
```
       ⎡ lim F2(x) as x --> 0+ =  5 ⎤
#23:   ⎢ lim F2(x) as x --> 0- =  5 ⎥
       ⎣ lim F2(x) as x --> 0  =  5 ⎦
```

Example 3:

F3(*x*) = IF(*x* > 1, IF(*x* = 0, 0, -1)). For this function F3(*x*), we have the following *DERIVE* 2.6 limit results. The left and right hand *DERIVE* 2.60 limits agree with the classical calculus results, but the two-sided *DERIVE* 2.60 limit DOES NOT AGREE with the classical result.



```
      F3(x) :=
        If x > 0
           1
#24:       If x = 0
              0
              -1

#25:  aa3 := [- √2, -1, -0.5, 0, 0.5, 1, √2]

#26:  bb3 := VECTOR(F3(u), u, aa3)

#27:  [cc3 := APPEND([x], aa3), dd3 := APPEND([F3(x)], bb3)]

#28:  [cc3, dd3]

      ⎡                      1        1         ⎤
      ⎢   x    - √2  -1  - ───   0   ───   1  √2 ⎥
#29:  ⎢                      2        2         ⎥
      ⎢                                         ⎥
      ⎣ F3(x)   -1   -1    -1    0    1    1   1 ⎦

#30:  ee3 := [limf(3, +), limf(3, -), limf(3,  )]

      ff3 := ⎡lim  F3(x), lim  F3(x), lim F3(x)⎤
#31:         ⎣x→0+        x→0-        x→0      ⎦

#32:  [ee3, ff3]'
```



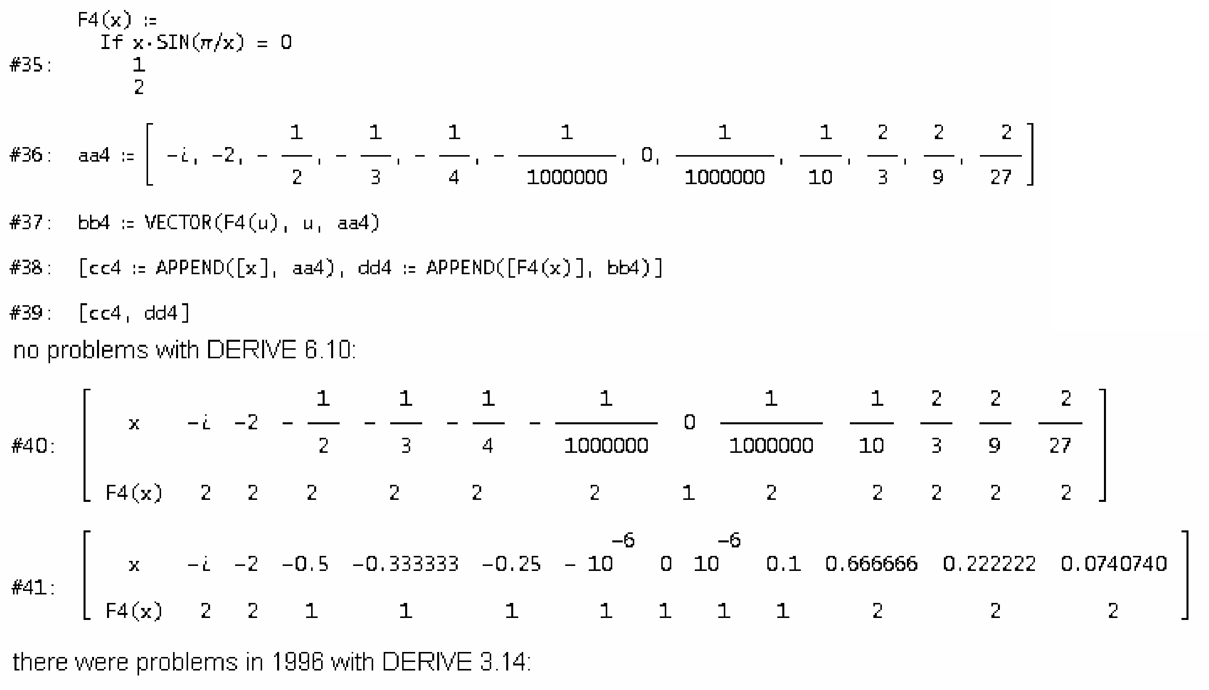
Simplify and Approximate in Versions 2.58 - 3.10 **and in 6.10:**

```
       ┌ lim F3(x) as x --> 0+ =  0 ┐
#33:   │ lim F3(x) as x --> 0- =  0 │
       └ lim F3(x) as x --> 0  =  0 ┘
```

Simplify in Versions 3.14:

```
       ┌ lim F3(x) as x --> 0+ =   1 ┐
#34:   │ lim F3(x) as x --> 0- =  -1 │
       └ lim F3(x) as x --> 0  =   ? ┘
```

Example 4: F4(x) = IF(x SIN(π/x) = 0,1,2). For this function F4(x), the *DERIVE* 2.6 limits with the value 1 DO NOT AGREE with the corresponding classical calculus limits, while those with the value 2 DO AGREE with the classical limits; also the lim F4(x) as $x \to 0$ as given below (in approXimate Mode and if interpreted appropriately) DOES AGREE with the classical calculus limit.



```
       F4(x) :=
         If x·SIN(π/x) = 0
#35:        1
            2

                         1     1     1        1                1        1   2   2    2
#36:  aa4 := [ -ί, -2, - ─, - ─, - ─, - ───────, 0, ───────, ──, ─, ─, ──]
                         2     3     4     1000000          1000000   10   3   9   27

#37:  bb4 := VECTOR(F4(u), u, aa4)

#38:  [cc4 := APPEND([x], aa4), dd4 := APPEND([F4(x)], bb4)]

#39:  [cc4, dd4]
```

no problems with DERIVE 6.10:

```
       ┌                  1     1     1        1           1        1   2   2    2  ┐
       │  x    -ί  -2  - ─  - ─  - ─  - ───────  0  ───────  ──  ─  ─  ──  │
#40:   │                  2     3     4     1000000      1000000   10   3   9   27  │
       │                                                                           │
       └ F4(x)  2   2     2     2     2        2     1      2       2   2   2    2  ┘

       ┌                                              -6       -6                                   ┐
       │  x    -ί  -2  -0.5  -0.333333  -0.25  - 10    0  10    0.1  0.666666  0.222222  0.0740740 │
#41:   │                                                                                             │
       └ F4(x)  2   2    1       1        1        1    1   1     1      2         2          2     ┘
```

there were problems in 1996 with DERIVE 3.14:

This is from 1996

```
       ┌  "x"     -î   -2   -  1    -  1    -  1    -    1                  0
       │                     ───      ───      ───     ───────
       │                      2        3        4      1000000
#12:   │
       │                                                                      ┌  π ┐
       │ "F4(x)"   2    2      1        1        1         1          IF│0·SIN│──│  =
       └                                                                      └  0 ┘

                        1          1     2     2     2   ┐
                     ───────      ──     ─     ─    ──   │
                     1000000      10     3     9    27   │
                                                         │
           0, 1, 2┘      1          1     2     2     2   ┘
```

Simplify in Version 3.14:

```
      ┌                                  1      1      1           1                   1
         "x"     -î   -2   -  ───   -  ───   -  ───   -  ─────────────   0   ─────────────
#15:  │                                  2      3      4        1000000           1000000
      └ "F4(x)"   2    2       1        1        1            1           1         1

          1      2      2      2   ┐
        ────   ───   ───    ───   │
         10      3      9     27   │
          1      2      2      2   ┘
```

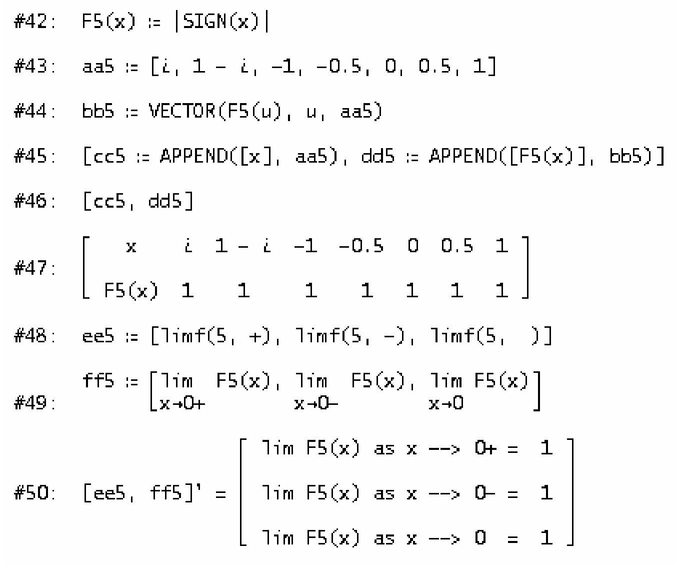
approXimate in Version 3.14:

```
      ┌                                  1                       -6        -6             2      2
         "x"     -î   -2   -0.5    -  ───    -0.25    - 10      0   10     0.1    ───    ───
#17:  │                                  3                                                 3      9
      └ "F4(x)"   2    2     1        1          1          2      1    2       1      2      2

          2  ┐
        ────  │
         27   │
          2  ┘
```

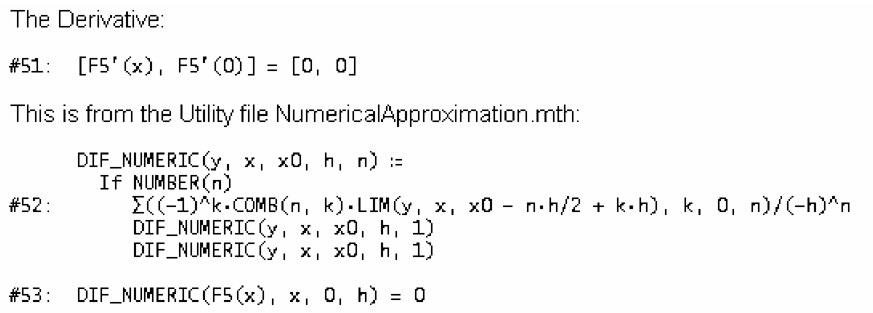
In DNL#23 from 1996 follows a thorough investigation of left- and right sided limits. As this is of no importance now I am omitting this part and go ahead with G P Specks 5th example:

Example 5:

F5(*x*) = | SIGN(*x*) |. F5(0) = | SIGN(0) |.Simplifies or approXimates to 1, even though SIGN(0) is undefined and when Simplified or approXimated gives SIGN(0). (note, however, that √ SIGN^2(0) returns SIGN(0), as should be the case.). Each of the *DERIVE* 2.60 limit results DOES AGREE with the corresponding classical calculus limit. However, the following *DERIVE* 2.60 derivative results (which are of course special limits) are NOT in agreement with the classical calculus derivative at *x* = 0.



```
#42:  F5(x) := |SIGN(x)|

#43:  aa5 := [ί, 1 - ί, -1, -0.5, 0, 0.5, 1]

#44:  bb5 := VECTOR(F5(u), u, aa5)

#45:  [cc5 := APPEND([x], aa5), dd5 := APPEND([F5(x)], bb5)]

#46:  [cc5, dd5]

      ┌   x     ί  1 - ί  -1  -0.5  0  0.5  1 ┐
#47:  │                                        │
      └ F5(x)  1    1      1    1    1   1   1 ┘

#48:  ee5 := [limf(5, +), limf(5, -), limf(5,  )]

      ff5 := ┌lim  F5(x), lim  F5(x), lim F5(x)┐
#49:         └x→0+        x→0-        x→0      ┘

                     ┌ lim F5(x) as x --> 0+ =  1 ┐
#50:  [ee5, ff5]' =  │ lim F5(x) as x --> 0- =  1 │
                     └ lim F5(x) as x --> 0  =  1 ┘
```



The Derivative:

```
#51:   [F5'(x), F5'(0)] = [0, 0]
```

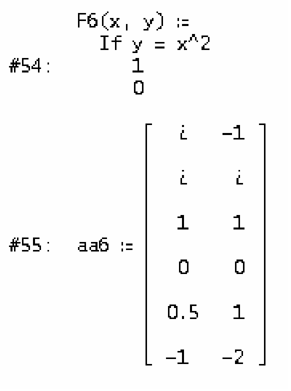
This is from the Utility file NumericalApproximation.mth:

```
       DIF_NUMERIC(y, x, x0, h, n) :=
         If NUMBER(n)
#52:        Σ((-1)^k·COMB(n, k)·LIM(y, x, x0 - n·h/2 + k·h), k, 0, n)/(-h)^n
            DIF_NUMERIC(y, x, x0, h, 1)
            DIF_NUMERIC(y, x, x0, h, 1)

#53:   DIF_NUMERIC(F5(x), x, 0, h) = 0
```

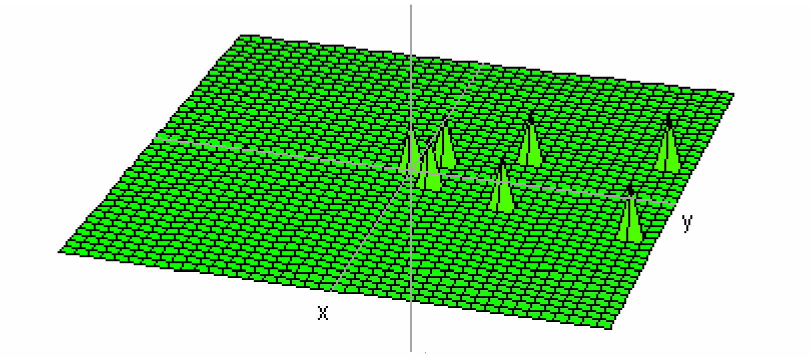
Example 6:

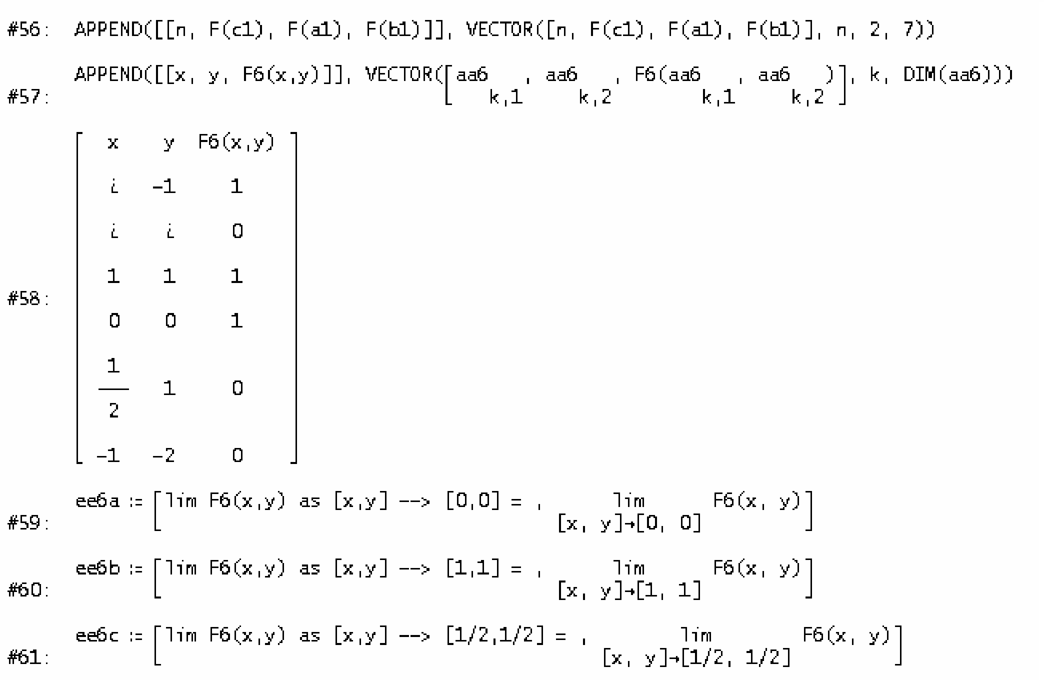
F6(x, y) = IF(y = x^2, 1, 0). Each of the first two *DERIVE* 2.60 limits below DISAGREES with the corresponding classical limit for a function of two variables, since the classical calculus limit fails to exist in each case. The third limit below AGREES with the corresponding classical calculus one.

The same comments can made be here for the LIM2 results as were made for the LIM results above. Note that neither the *DERIVE* 2.60 LIM nor LIM2 (being limits along rays through the point in question) are designed to handle the classical limit of a function of the F6 type, even in the absence of the difficulties shown here.

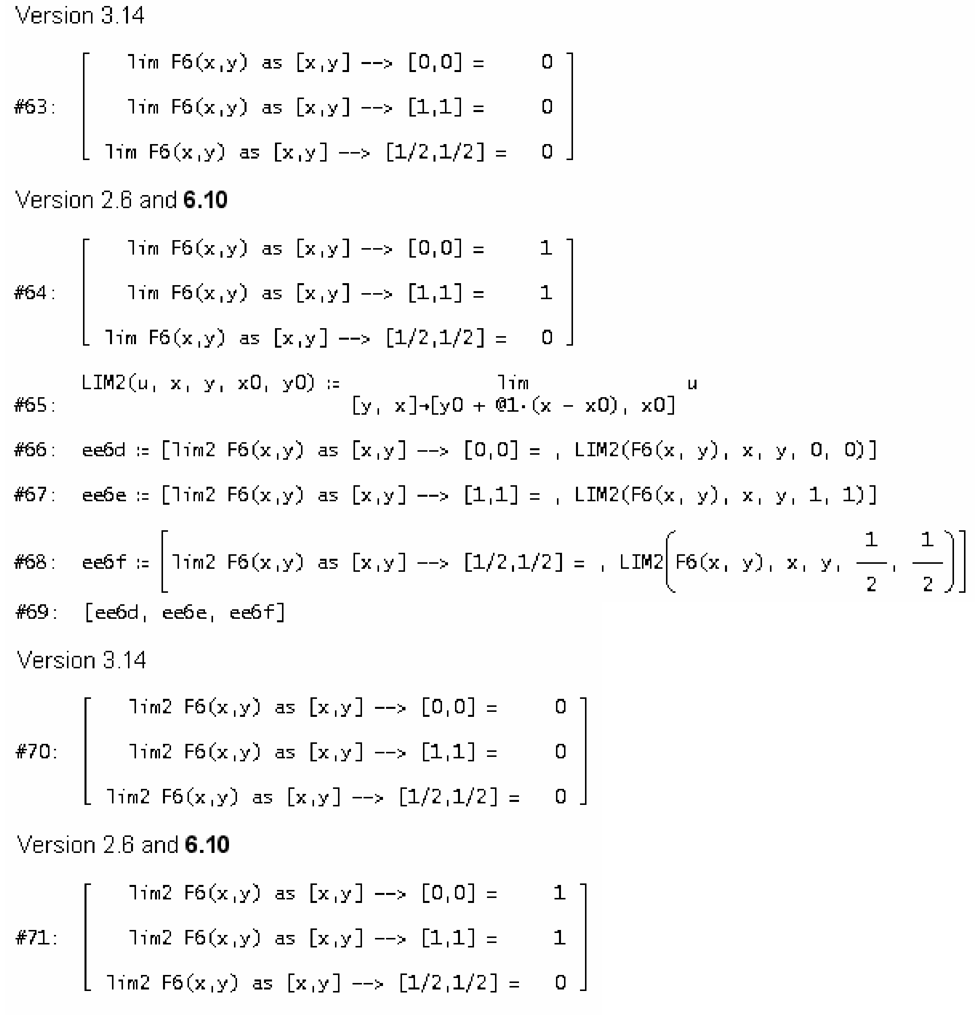


```
       F6(x, y) :=
         If y = x^2
#54:        1
            0

                 ┌  ¿    -1  ┐
                 │           │
                 │  ¿    ¿   │
                 │           │
                 │  1    1   │
                 │           │
#55:   aa6 :=    │  0    0   │
                 │           │
                 │  0.5  1   │
                 │           │
                 └  -1   -2  ┘
```





```
#56:   APPEND([[n, F(c1), F(a1), F(b1)]], VECTOR([n, F(c1), F(a1), F(b1)], n, 2, 7))

       APPEND([[x, y, F6(x,y)]], VECTOR(┌aa6   , aa6   , F6(aa6   , aa6   )┐, k, DIM(aa6)))
#57:                                    └   k,1     k,2        k,1     k,2 ┘

       ┌  x    y   F6(x,y) ┐
       │                   │
       │  ¿   -1     1     │
       │                   │
       │  ¿    ¿     0     │
       │                   │
       │  1    1     1     │
#58:   │                   │
       │  0    0     1     │
       │                   │
       │  1                │
       │ ───   1     0     │
       │  2                │
       │                   │
       └ -1   -2     0     ┘

       ee6a := ┌lim F6(x,y) as [x,y] --> [0,0] = ,      lim      F6(x, y)┐
#59:           └                                   [x, y]→[0, 0]          ┘

       ee6b := ┌lim F6(x,y) as [x,y] --> [1,1] = ,      lim      F6(x, y)┐
#60:           └                                   [x, y]→[1, 1]          ┘

       ee6c := ┌lim F6(x,y) as [x,y] --> [1/2,1/2] = ,       lim         F6(x, y)┐
#61:           └                                       [x, y]→[1/2, 1/2]          ┘
```



```
Version 3.14

       ┌   lim F6(x,y) as [x,y] --> [0,0] =      0 ┐
#63:   │   lim F6(x,y) as [x,y] --> [1,1] =      0 │
       └ lim F6(x,y) as [x,y] --> [1/2,1/2] =    0 ┘

Version 2.6 and 6.10

       ┌   lim F6(x,y) as [x,y] --> [0,0] =      1 ┐
#64:   │   lim F6(x,y) as [x,y] --> [1,1] =      1 │
       └ lim F6(x,y) as [x,y] --> [1/2,1/2] =    0 ┘

       LIM2(u, x, y, x0, y0) :=             lim               u
#65:                          [y, x]→[y0 + @1·(x - x0), x0]

#66:   ee6d := [lim2 F6(x,y) as [x,y] --> [0,0] = , LIM2(F6(x, y), x, y, 0, 0)]

#67:   ee6e := [lim2 F6(x,y) as [x,y] --> [1,1] = , LIM2(F6(x, y), x, y, 1, 1)]

                                                                           1   1
#68:   ee6f := [lim2 F6(x,y) as [x,y] --> [1/2,1/2] = , LIM2(F6(x, y), x, y, ─, ─)]
                                                                           2   2
#69:   [ee6d, ee6e, ee6f]

Version 3.14

       ┌   lim2 F6(x,y) as [x,y] --> [0,0] =      0 ┐
#70:   │   lim2 F6(x,y) as [x,y] --> [1,1] =      0 │
       └ lim2 F6(x,y) as [x,y] --> [1/2,1/2] =    0 ┘

Version 2.6 and 6.10

       ┌   lim2 F6(x,y) as [x,y] --> [0,0] =      1 ┐
#71:   │   lim2 F6(x,y) as [x,y] --> [1,1] =      1 │
       └ lim2 F6(x,y) as [x,y] --> [1/2,1/2] =    0 ┘
```

A number of the difficulties illustrated in the preceding examples can be traced to the way in which *DERIVE* 2.60 limits behave relative to functions using the IF( ) statement - however, as Example 5 shows, this is not the only difficulty. The way in which *DERIVE* 2.60 limits react relative to the IF( ) statement functions given above can be determined without too much analysis; but other more complicated multiple IF( ) statement functions can be displayed where it is not nearly so easy to predict just what the *DERIVE* 2.60 limit result will be.

I have created a variety of simple *DERIVE* "programs" to bypass some of the one and two variable limit difficulties, but have not come up with a totally satisfactory solution to the inherent limit problems. Example 5 above illustrates a particularly annoying problem in that *DERIVE* returns a numerical value for the composition of two pre-defined functions at an independent variable value which "shouldn't" be in the domain of the composition. How many other examples of this kind are prevalent? None? Several?

If we were confident that *DERIVE* will produce appropriate functional values for all predefined and author defined (using IF( ) where needed) functions and compositions of such functions, then we would be in a position to seriously tackle the resolution of the *DERIVE* limit difficulties. Notwithstanding the difficulties raised herein, for the very great majority of *DERIVE* functions of whatever type that I have used, correct functional values are returned. As long as this is the case, *DERIVE* does provide two very powerful tools for investigating important features, including the limit, of non-trivial functions: the easy display of a table of functional values and functional plots. As an illustration, Example 7 in the printout that follows shows the verification (by way of a table) of an important feature of the very interesting non-analytic function

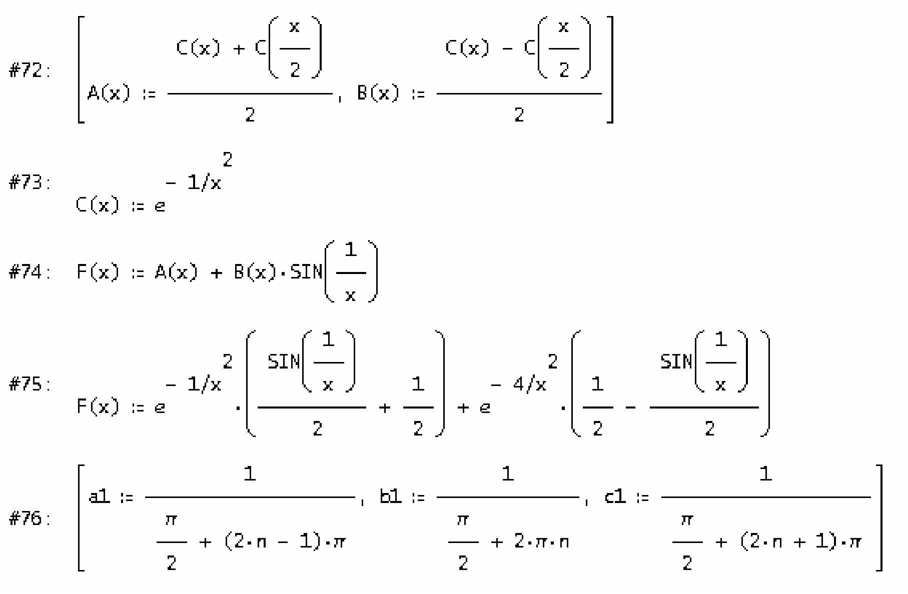
Example 7: An example of a non-elementary type of local minimum

$$F(x) = A(x) + B(x)\sin\left(\frac{1}{x}\right) \qquad \text{where}$$

$$A(X) = \frac{C(x) + C\left(\frac{x}{2}\right)}{2} \text{ and } B(x) = \frac{C(x) - C\left(\frac{x}{2}\right)}{2} \quad \text{with}$$

$$C(x) = e^{-\frac{1}{x^2}} \quad \text{for } x \neq 0 \quad \text{and}$$

$$F(x) = 0 \quad \text{for } x = 0.$$



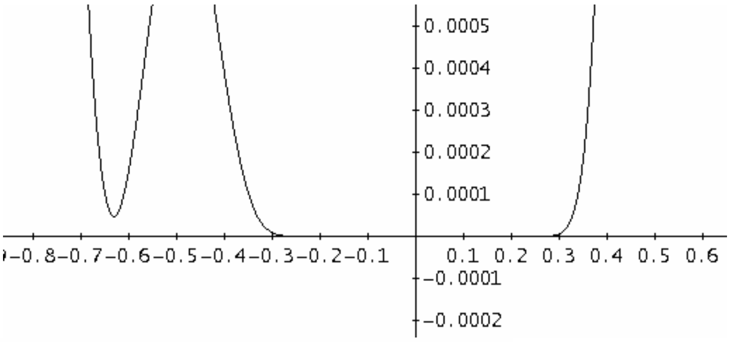
#72: $\left[A(x) := \frac{C(x) + C\left(\frac{x}{2}\right)}{2},\ B(x) := \frac{C(x) - C\left(\frac{x}{2}\right)}{2}\right]$

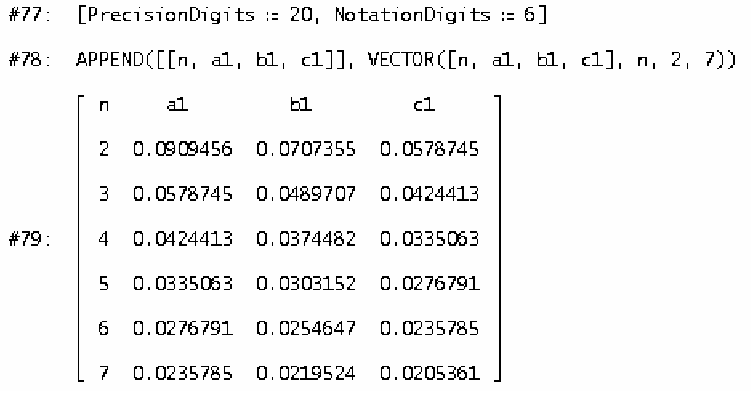
#73: $C(x) := e^{-1/x^2}$

#74: $F(x) := A(x) + B(x)\cdot \mathrm{SIN}\left(\frac{1}{x}\right)$

#75: $F(x) := e^{-1/x^2}\cdot\left(\frac{\mathrm{SIN}\left(\frac{1}{x}\right)}{2} + \frac{1}{2}\right) + e^{-4/x^2}\cdot\left(\frac{1}{2} - \frac{\mathrm{SIN}\left(\frac{1}{x}\right)}{2}\right)$

#76: $\left[a1 := \frac{1}{\frac{\pi}{2} + (2\cdot n - 1)\cdot\pi},\ b1 := \frac{1}{\frac{\pi}{2} + 2\cdot\pi\cdot n},\ c1 := \frac{1}{\frac{\pi}{2} + (2\cdot n + 1)\cdot\pi}\right]$





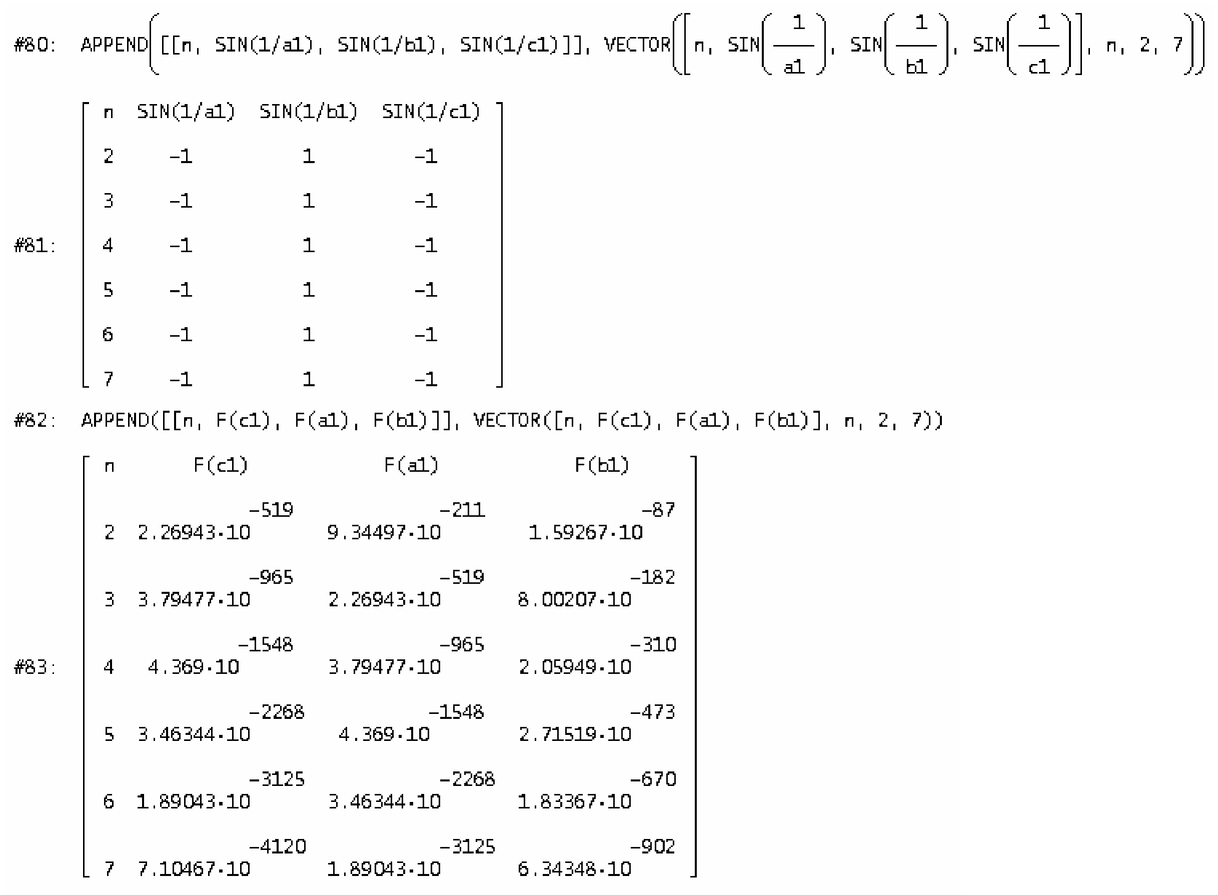
#77: [PrecisionDigits := 20, NotationDigits := 6]

#78: APPEND([[n, a1, b1, c1]], VECTOR([n, a1, b1, c1], n, 2, 7))

#79:

n	a1	b1	c1
2	0.0909456	0.0707355	0.0578745
3	0.0578745	0.0489707	0.0424413
4	0.0424413	0.0374482	0.0335063
5	0.0335063	0.0303152	0.0276791
6	0.0276791	0.0254647	0.0235785
7	0.0235785	0.0219524	0.0205361

0



```
                                                            ⎛   1  ⎞     ⎛  1  ⎞     ⎛  1  ⎞
#80:  APPEND([[n, SIN(1/a1), SIN(1/b1), SIN(1/c1)]], VECTOR([n, SIN⎜ ── ⎟, SIN⎜ ── ⎟, SIN⎜ ── ⎟], n, 2, 7))
                                                            ⎝  a1  ⎠     ⎝  b1 ⎠     ⎝  c1 ⎠

       ⎡ n  SIN(1/a1)  SIN(1/b1)  SIN(1/c1) ⎤
       ⎢                                    ⎥
       ⎢ 2     -1          1         -1     ⎥
       ⎢                                    ⎥
       ⎢ 3     -1          1         -1     ⎥
       ⎢                                    ⎥
#81:   ⎢ 4     -1          1         -1     ⎥
       ⎢                                    ⎥
       ⎢ 5     -1          1         -1     ⎥
       ⎢                                    ⎥
       ⎢ 6     -1          1         -1     ⎥
       ⎢                                    ⎥
       ⎣ 7     -1          1         -1     ⎦

#82:  APPEND([[n, F(c1), F(a1), F(b1)]], VECTOR([n, F(c1), F(a1), F(b1)], n, 2, 7))

       ⎡ n      F(c1)          F(a1)          F(b1)      ⎤
       ⎢                                                 ⎥
       ⎢               -519           -211            -87⎥
       ⎢ 2  2.26943·10      9.34497·10      1.59267·10   ⎥
       ⎢                                                 ⎥
       ⎢               -965           -519           -182⎥
       ⎢ 3  3.79477·10      2.26943·10      8.00207·10   ⎥
       ⎢                                                 ⎥
       ⎢             -1548            -965           -310⎥
#83:   ⎢ 4   4.369·10       3.79477·10      2.05949·10   ⎥
       ⎢                                                 ⎥
       ⎢              -2268           -1548          -473⎥
       ⎢ 5  3.46344·10       4.369·10       2.71519·10   ⎥
       ⎢                                                 ⎥
       ⎢              -3125           -2268          -670⎥
       ⎢ 6  1.89043·10      3.46344·10      1.83367·10   ⎥
       ⎢                                                 ⎥
       ⎢              -4120           -3125          -902⎥
       ⎣ 7  7.10467·10      1.89043·10      6.34348·10   ⎦
```

It can be proved for our F(x) defined above for all real x:

F(x) has derivatives of all orders at each real x;
F(x) has a relative minimum at x = 0;
a1 > b1 > c1 and F(c1) < F(a1) < F(b1) for n ≥ 2.

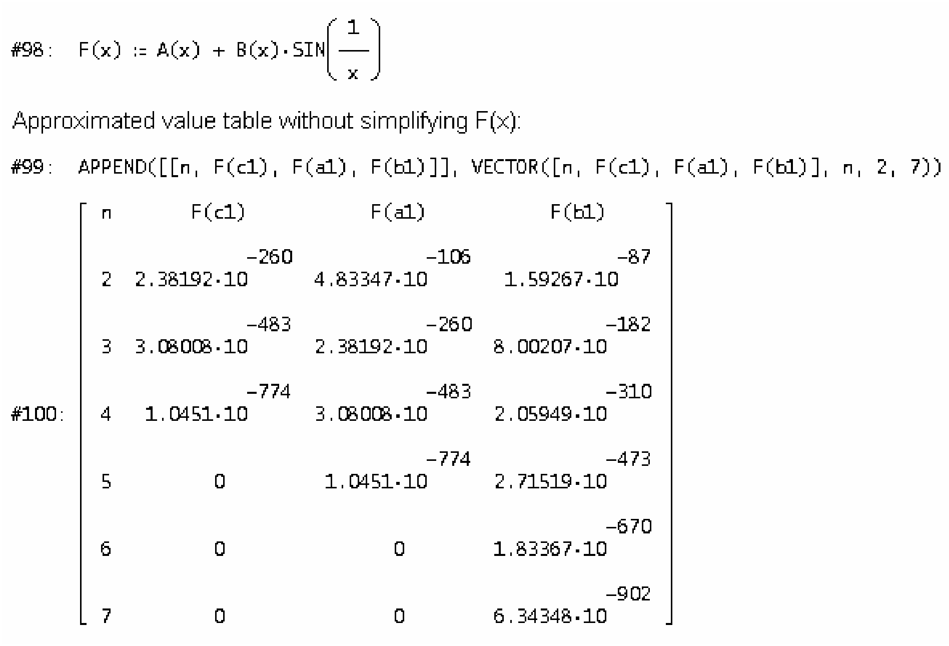
Thus, F'(x) is not ≥ 0 throughout any interval (0,b), nor is F'(x) ≤ 0 throughout any interval (a,0).

The *DERIVE* matrices displayed above provide simple verification that a1 > b1 > c1 and F(c1) < F(a1) < F(b1) for 2 ≤ n ≤ 7. Such simple *DERIVE* verification is very handy indeed in providing confidence in one’s theoretical results for "intricate" functions such as our F(x) given here.

The following is a "graphical" view of the relative magnitudes of a1, b1, c1 and F(a1), F(b1), F(c1).

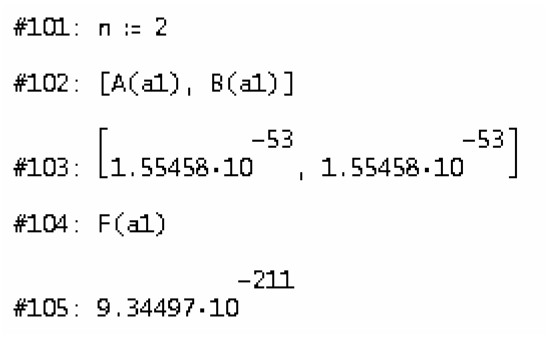
```
  F |
    |     for n >= 2:
    |
    |              *
    |
    |
    |                   *
    |
    |    *
    |
    |
   -|----|---------|----|-------- x
         c1   <    b1 < a1
       F(c1)  <  F(a1) < F(b1)
```

A word of caution! If we define F(x) in the following manner without Simplifying F(x) as was done above, then the matrix of values for F(a1), F(b1) and F(c1) may be totally inaccurate!!



```
                              ⎛ 1 ⎞
#98:  F(x) := A(x) + B(x)·SIN ⎜ ─ ⎟
                              ⎝ x ⎠

Approximated value table without simplifying F(x):

#99:  APPEND([[n, F(c1), F(a1), F(b1)]], VECTOR([n, F(c1), F(a1), F(b1)], n, 2, 7))

       ⎡ n      F(c1)          F(a1)          F(b1)      ⎤
       ⎢                                                 ⎥
       ⎢              -260            -106            -87⎥
       ⎢ 2  2.38192·10     4.83347·10      1.59267·10    ⎥
       ⎢                                                 ⎥
       ⎢              -483            -260           -182⎥
       ⎢ 3  3.08008·10     2.38192·10     8.00207·10     ⎥
       ⎢                                                 ⎥
       ⎢              -774            -483           -310⎥
#100:  ⎢ 4   1.0451·10     3.08008·10     2.05949·10     ⎥
       ⎢                                                 ⎥
       ⎢                              -774           -473⎥
       ⎢ 5        0         1.0451·10     2.71519·10     ⎥
       ⎢                                                 ⎥
       ⎢                                             -670⎥
       ⎢ 6        0              0        1.83367·10     ⎥
       ⎢                                                 ⎥
       ⎢                                             -902⎥
       ⎣ 7        0              0        6.34348·10     ⎦
```

We see that the F(c1) and F(a1) values in this matrix are totally different from the correct function values in the simplified F(x) matrix above, while the F(b1) values agree for the two matrices. A classical problem in numerical analysis underlies the difficulty here:
For our function F(x) e^(-4/x^2) is so much smaller than e^(-1/x^2) for a1 with n = 2, for example, that the evaluation of A(a1) with PrecisionDigits: 20 will give the value A(a1) = (4.83347 10^-106)/2 and similarly for B(a1). If we used PrecisionDigits: 100 in this case the following shows that the *DERIVE* F(a1) value would be the correct value:

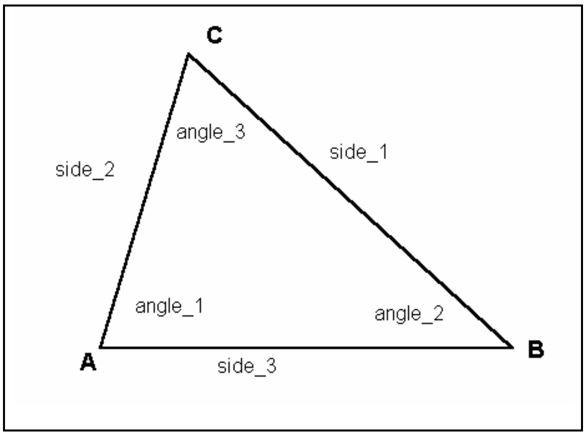


```
#101:  n := 2

#102:  [A(a1), B(a1)]

       ⎡              -53               -53⎤
#103:  ⎣1.55458·10      , 1.55458·10       ⎦

#104:  F(a1)

                   -211
#105:  9.34497·10
```

The Trigonometric Super Box

Josef Böhm, Würmla, Austria

In DNL#9 from March 1993 I presented a "Black Box" to solve oblique triangles. At the occasion preparing a workshop for the DERIVE Conference 1996 I tried to extent this "Black Box" to a real "Superbox" in order to build a "Circuit of Boxes" for this special subject.



For all functions and formulae we define a triangle with common names for sides and angles:

(I based my workshop on the problem:

ΔABC(BC = 120, AC = 80, ∠ABC = 20°)).

This is the most difficult case, because we have to consider two solutions!!

1 The White Box

In the White Box phase we would derive the Sine Rule and the Cosine Rule. The scientific pocket calculator is sufficient support at that time. The first step using *DERIVE* could be editing the Cosine Rule and the Sine Rule. The students must know which of the both rules applies for solving a problem, which variables can be substituted and then let *DERIVE* solve the respective equation. I'd like to call this position as

2 Not a Box at all

The start of solving the given problem could look like as follows:

Not a Box at All

```
User:
#1:   InputMode ≔ Word

User:
        2      2      2
#2:   s3  = s1  + s2  - 2·s1·s2·COS(a3)

User:
#3:   s1·SIN(a2) = s2·SIN(a1)

User:
#4:   80·SIN(α°) = 120·SIN(20°)

Solve(#4,α):
#5:   SOLVE(80·SIN(α°) = 120·SIN(20°), α, Real)

Approx(#5):
#6:   α = -210.8658824 ∨ α = 149.1341175 ∨ α = 30.86588247
```

Take care that you set Manage Trigonometry Angle: Radian (in DERIVE 6: Options > Mode Settings > Angular unit > Radians). So you have to substitute for the angle a2 in line #3 **α deg**. In the next phase we provide four functions which give the solutions of the equations in lines #2 and #3 for the angle or the side. This package of functions represents the

3 The "Gray" Box

`cs_a3(side_1, side_2, angle_3)`	returns side_3
`ca_s1(side_1, side_2, side_3)`	returns angle_1
`ss_a2(side_1, angle_1, angle_2)`	returns side_2

`sa_s2(side_1, side_2, angle_1, k)`	returns angle_2 (k = 1 ⇒ angle_2 ≤ 90°,
	k = 2 ⇒ 90° < angle_2 ≤ 180°,
	k = 1 by default.)

The students must still know which of the rules applies. They also have to consider the case with the possibility of obtaining two solutions. To explain the "names" of the functions:

`cs_a3` = use the **C**osine Rule to find the **S**ide opposite to **A**ngle**_3**.

(You will find the "Gray Box functions" in the DERIVE printout at the end of the contribution.)

4 The "Black Box"

These four functions from the "Gray Box" can be developed together with the students. The next ones are more difficult. We use the Gray Box Functions and combine them to a real "Black Box" to solve the triangles in one step. It will depend on the students´ skill in working with *DERIVE* to produce the next package. This could be much easier if the maths teacher is also the information technology teacher. We can build in some control structures (considering the triangle's inequality, considering one, two or none solution in case of SSA, ...). I could imagine to produce the most difficult module together with the students and let them produce the remaining ones in groups by their own.

Last year I was short of time teaching trigonometry. I only had time to deal with the trigonometric functions and their applications in connection with right angled triangles. But I didn't want to miss the applications of solving oblique triangles in plane and space. I wanted the students to use strategies for calculating polygons and problems in 3D-space. So I provided this Black Box in a Utility file and an instruction similar to the following:

Load the Utility file TRIGO.mth. You then will have four functions available for solving triangles. The provided functions are:

```
sss(side_1, side_2, side_3)
sas(side_1, angle_3, side_2)
asa(angle_2, side_1, angle_3)
ssa(side_1, side_2, angle_1)
```

(*I don't want to repeat that part from DNL#9. I use these functions for building the SUPER BOX.*)

But according to Bernhard Kutzler [2,3] the real Super Box would be a function like

```
        triangle(side_1, side_2, side_3, angle_1, angle_2, angle_3)
```

with three parameters known and three unknown. *DERIVE* should do all the remaining work: check if the triangle really exists, decide the case and calculate the missing sides and angles. Then our example should look like that:

```
        triangle(120, 80, side_ab, angle_bac, 20°, angle_abc).
```

I faced the challenge to develop such a function and we reproduced the ideas in the workshop.

5 The "Super Box"

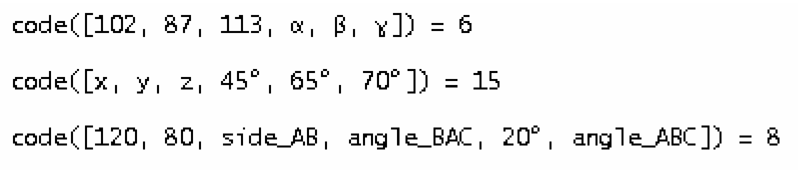
Depending on the unknown parameters in `TRIANGLE(…)` *DERIVE* has to decide which of the Black Box functions applies. Additionally we want to implement the until now untreated case of SAA. (This is new even for the participants of the Workshop in Bonn). In the Black Box phase we would calculate the third angle by hand and then apply ASA. That is now the moment when not only the - not too difficult - implementing of mathematical rules will take place but when you also - perhaps with your students - have to consider an algorithm. In the following I will present one possibility (I didn't find a better one):

I set up a table with 6 columns for the parameters and sufficient rows to represent all the possible combinations to define a triangle from **SSS** to **AAA -** (next page). As $\binom{6}{3} = 20$ we need 20 rows. Surprisingly a pattern emerges. This pattern is the key to solve the problem. The parameters have to be edited as a vector v and the existence of values for its elements creates the code for the application of well known tools. While `C2(v)` picks out the SAA-case, `CODE(v)` gives a unique relation between the code numbers and the remaining problems (SSS, ASA, AAA and SSA). The `IF`-construct returns the value 0 if *DERIVE* cannot decide whether the expression is 0 or not. And that is the case, if the expression is a not predesignated variable or a given constant value.

```
c2(v):=IF(v SUB 1*v SUB 4*v SUB 5>=0 OR (v SUB 1*v SUB 4*v SUB 6>=0 OR
        (v SUB 2*v SUB 4*v SUB 5>=0 OR (v SUB 2*v SUB 5*v SUB 6>=0 OR
        (v SUB 3*v SUB 4*v SUB 6>=0 OR v SUB 3*v SUB 5*v SUB 6>=0)))),1,0,0)
```

```
code(v):=SUM(VECTOR(IF(v SUB i=0,i,i,0),i,6))
```

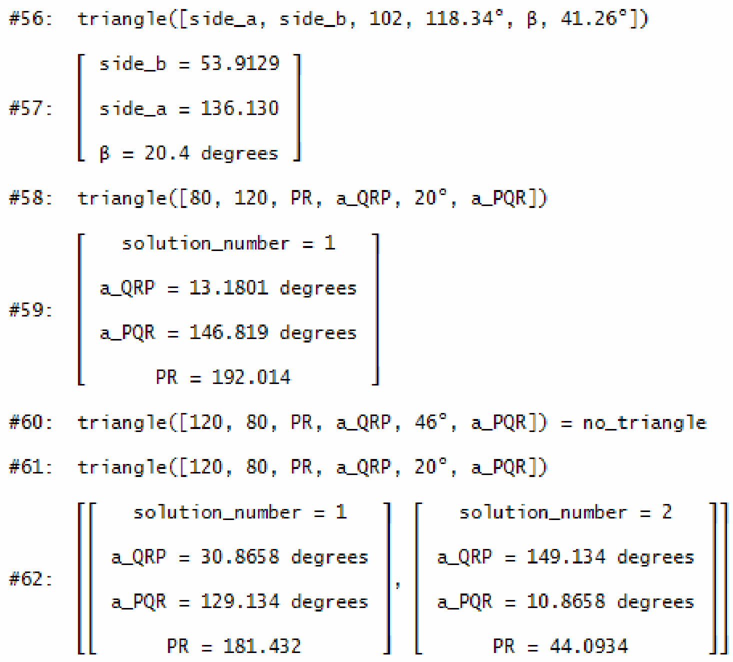
For example see some codes:



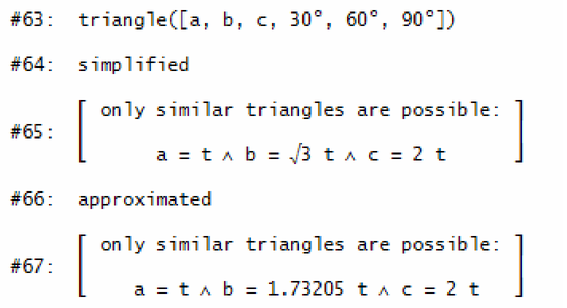
```
code([102, 87, 113, α, β, γ]) = 6

code([x, y, z, 45°, 65°, 70°]) = 15

code([120, 80, side_AB, angle_BAC, 20°, angle_ABC]) = 8
```

For the cases SAS and ASA only a permutation of the parameters is necessary. AAA is special: *DERIVE* returns the proportions of the sides (see one example!). The most difficult problem occurs in the cases of SSA and SAA, because there is no unique relation between the code numbers and the given sides and angles. But as you can see in the special printed numbers in the table there is an unique incidence of pairs of numbers. We can use the fact that the product of these pairs is defined and greater 0 to identify the problem. See first the table followed by some examples :

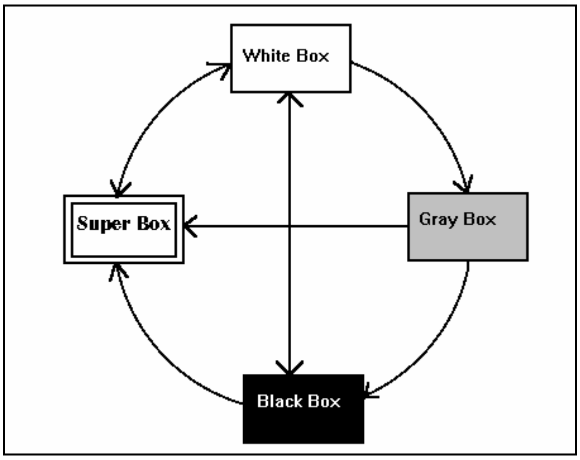
side_1 = v_1	side_2 = v_2	side_3 = v_3	angle_1=a_1	angle_2=a_2	angle_3=a_3	Σ = Code	Case
①			4	⑤		10	***SAA***
①			4		⑥	11	***SAA***
	②		④	5		11	***SAA***
	②			5	⑥	13	***SAA***
		③	④		6	13	***SAA***
		③		⑤	6	14	***SAA***
1	2	3				**6**	***SSS***
1	2				6	**9**	***SAS***
1		3	5			**9**	***SAS***
	2	3	4			**9**	***SAS***
1				5	6	**12**	***ASA***
	2		4		6	**12**	***ASA***
		3	4	5		**12**	***ASA***
			4	5	6	**15**	***AAA***
1	②		④			7	***SSA***
①	2			⑤		8	***SSA***
	2	③		⑤		10	***SSA***
	②	3			⑥	11	***SSA***
1		③	④			8	***SSA***
①		3			⑥	10	***SSA***



```
#56:  triangle([side_a, side_b, 102, 118.34°, β, 41.26°])

       ┌ side_b = 53.9129 ┐
#57:   │ side_a = 136.130 │
       └ β = 20.4 degrees ┘

#58:  triangle([80, 120, PR, a_QRP, 20°, a_PQR])

       ┌   solution_number = 1    ┐
       │ a_QRP = 13.1801 degrees  │
#59:   │ a_PQR = 146.819 degrees  │
       └      PR = 192.014        ┘

#60:  triangle([120, 80, PR, a_QRP, 46°, a_PQR]) = no_triangle

#61:  triangle([120, 80, PR, a_QRP, 20°, a_PQR])

       ┌┌   solution_number = 1    ┐  ┌   solution_number = 2    ┐┐
       ││ a_QRP = 30.8658 degrees  │  │ a_QRP = 149.134 degrees  ││
#62:   ││ a_PQR = 129.134 degrees  │, │ a_PQR = 10.8658 degrees  ││
       └└      PR = 181.432        ┘  └      PR = 44.0934        ┘┘
```



```
#63:  triangle([a, b, c, 30°, 60°, 90°])

#64:  simplified

      ⎡ only similar triangles are possible: ⎤
#65:  ⎢                                      ⎥
      ⎣      a = t ∧ b = √3 t ∧ c = 2 t      ⎦

#66:  approximated

      ⎡ only similar triangles are possible: ⎤
#67:  ⎢                                      ⎥
      ⎣    a = t ∧ b = 1.73205 t ∧ c = 2 t   ⎦
```

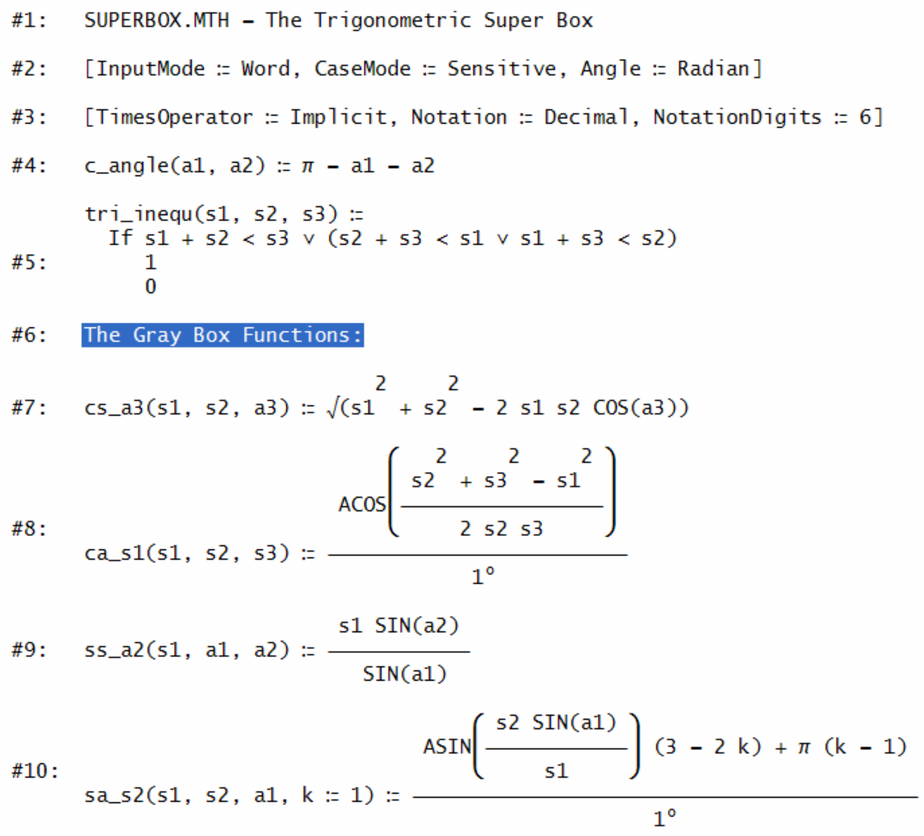


CIRCUIT OF BOXES

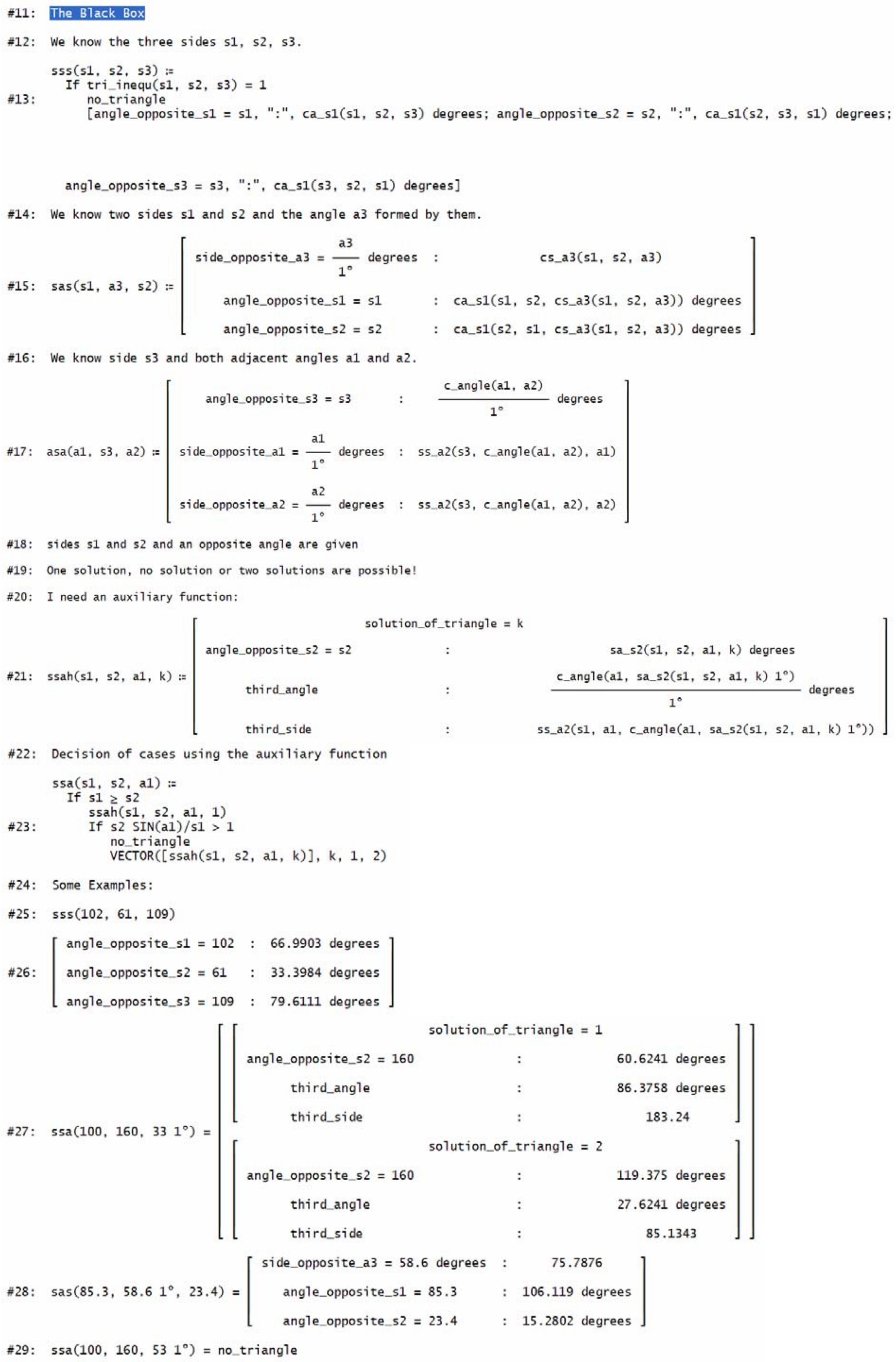
The workshop and, of course, this contribution should demonstrate that you are able to "program" even in *DERIVE*. You can use control structures and "error catching".

There are many possibilities to change from one box phase to another one. I try to illustrate that fact. You can step into this circuit at any position and then move to all directions.

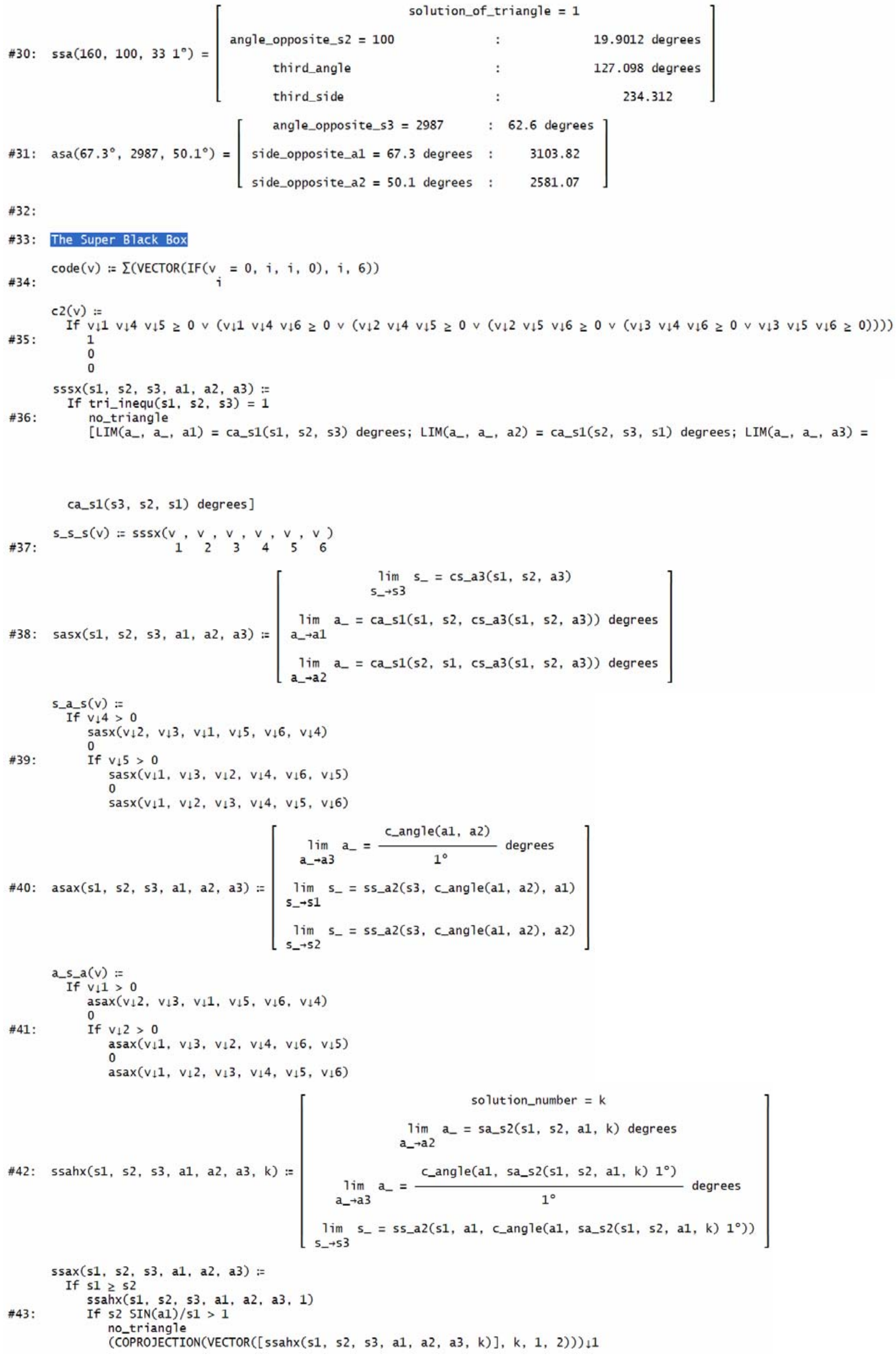
Now, in 2010 with DERIVE 6.10 and its programming features available I would doubtless try writing a compact DERIVE program. There are some differences to the code given in 1996 because of changes in DERIVE´s functionality (e.g. SOLVE in 1996 and SOLVE/SOLUTIONS in later versions). In later versions we can differ between ASIN and ARCSIN. This is also not considered in this package. But the package works as it is.



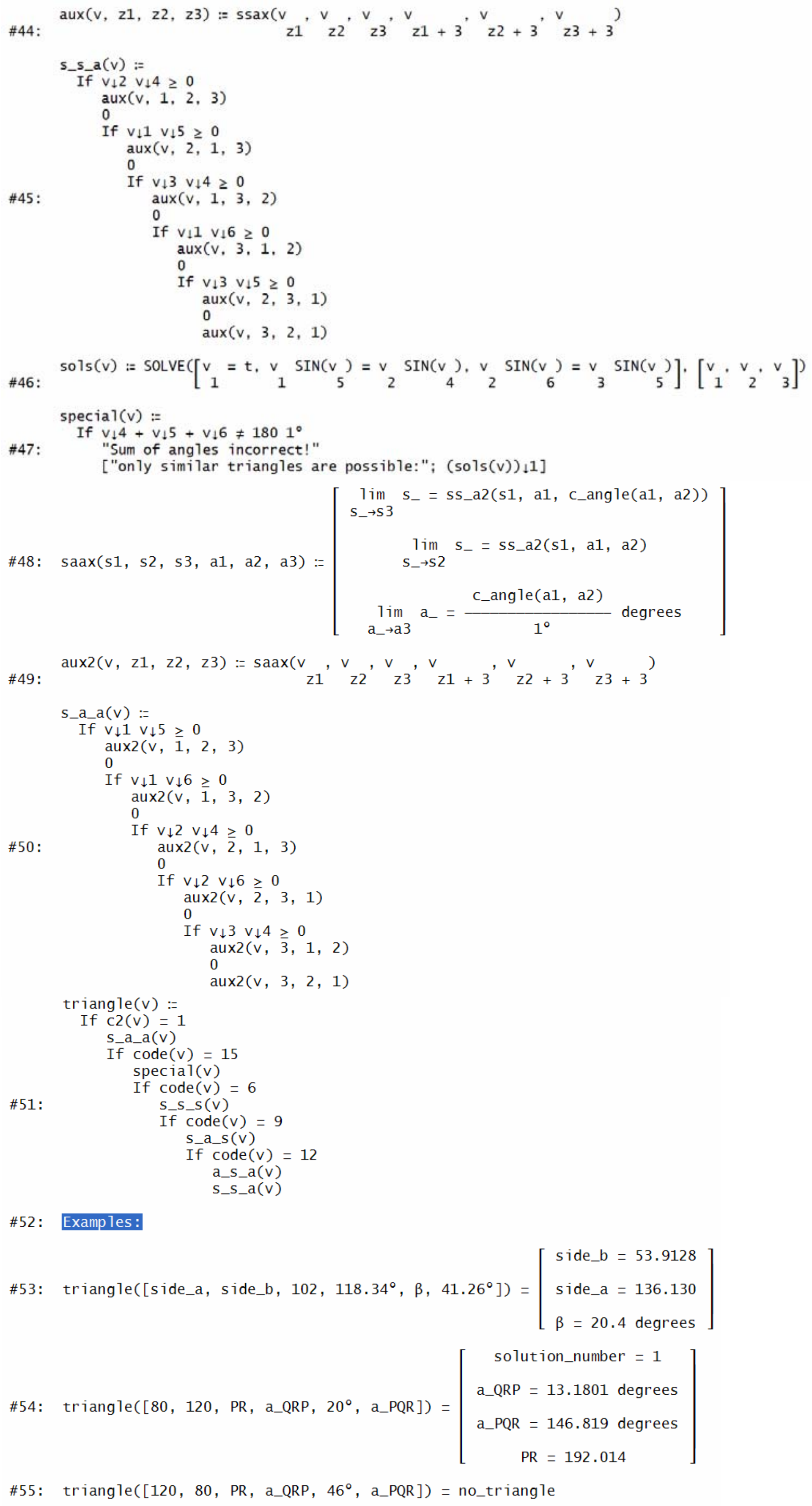
```
#1:   SUPERBOX.MTH - The Trigonometric Super Box

#2:   [InputMode := Word, CaseMode := Sensitive, Angle := Radian]

#3:   [TimesOperator := Implicit, Notation := Decimal, NotationDigits := 6]

#4:   c_angle(a1, a2) := π - a1 - a2

      tri_inequ(s1, s2, s3) :=
        If s1 + s2 < s3 ∨ (s2 + s3 < s1 ∨ s1 + s3 < s2)
#5:        1
           0

#6:   The Gray Box Functions:

                                  2      2
#7:   cs_a3(s1, s2, a3) := √(s1  + s2  - 2 s1 s2 COS(a3))

                                  ⎛   2      2      2 ⎞
                                  ⎜ s2  + s3  - s1   ⎟
                              ACOS⎜──────────────────⎟
#8:                               ⎝     2 s2 s3      ⎠
      ca_s1(s1, s2, s3) := ─────────────────────────
                                      1°

                              s1 SIN(a2)
#9:   ss_a2(s1, a1, a2) := ──────────────
                               SIN(a1)

                                    ⎛ s2 SIN(a1) ⎞
                                ASIN⎜────────────⎟ (3 - 2 k) + π (k - 1)
#10:                                ⎝     s1     ⎠
      sa_s2(s1, s2, a1, k := 1) := ──────────────────────────────────────────
                                                  1°
```



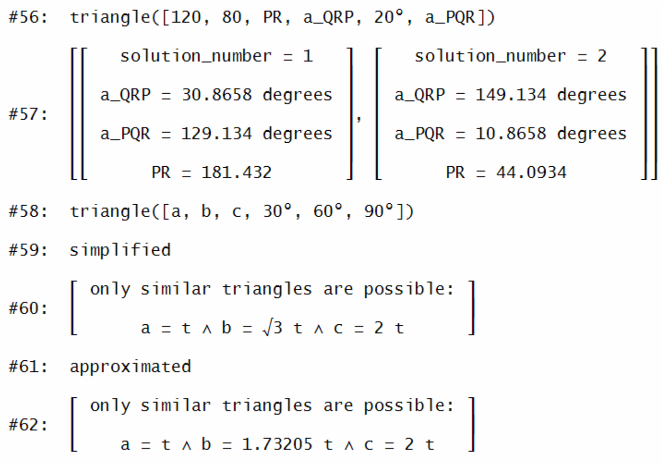
```
#11:  The Black Box

#12:  We know the three sides s1, s2, s3.

      sss(s1, s2, s3) :=
        If tri_inequ(s1, s2, s3) = 1
#13:       no_triangle
           [angle_opposite_s1 = s1, ":", ca_s1(s1, s2, s3) degrees; angle_opposite_s2 = s2, ":", ca_s1(s2, s3, s1) degrees;


        angle_opposite_s3 = s3, ":", ca_s1(s3, s2, s1) degrees]

#14:  We know two sides s1 and s2 and the angle a3 formed by them.

                         ┌                      a3                                          ┐
                         │ side_opposite_a3 = ──── degrees  :           cs_a3(s1, s2, a3)   │
                         │                      1°                                          │
#15:  sas(s1, a3, s2) := │                                                                  │
                         │     angle_opposite_s1 = s1       :  ca_s1(s1, s2, cs_a3(s1, s2, a3)) degrees │
                         │                                                                  │
                         └     angle_opposite_s2 = s2       :  ca_s1(s2, s1, cs_a3(s1, s2, a3)) degrees ┘

#16:  We know side s3 and both adjacent angles a1 and a2.

                         ┌                                        c_angle(a1, a2)          ┐
                         │     angle_opposite_s3 = s3      :    ───────────────── degrees  │
                         │                                              1°                 │
                         │                        a1                                       │
#17:  asa(a1, s3, a2) := │ side_opposite_a1 = ──── degrees  :  ss_a2(s3, c_angle(a1, a2), a1) │
                         │                        1°                                       │
                         │                        a2                                       │
                         │ side_opposite_a2 = ──── degrees  :  ss_a2(s3, c_angle(a1, a2), a2) │
                         └                        1°                                       ┘

#18:  sides s1 and s2 and an opposite angle are given

#19:  One solution, no solution or two solutions are possible!

#20:  I need an auxiliary function:

                          ┌                         solution_of_triangle = k                                       ┐
                          │ angle_opposite_s2 = s2          :                  sa_s2(s1, s2, a1, k) degrees         │
                          │                                              c_angle(a1, sa_s2(s1, s2, a1, k) 1°)        │
#21:  ssah(s1, s2, a1, k) :=│      third_angle                 :          ──────────────────────────────────── degrees │
                          │                                                          1°                            │
                          │                                                                                        │
                          └      third_side                  :        ss_a2(s1, a1, c_angle(a1, sa_s2(s1, s2, a1, k) 1°)) ┘

#22:  Decision of cases using the auxiliary function

      ssa(s1, s2, a1) :=
        If s1 ≥ s2
           ssah(s1, s2, a1, 1)
#23:       If s2 SIN(a1)/s1 > 1
              no_triangle
              VECTOR([ssah(s1, s2, a1, k)], k, 1, 2)

#24:  Some Examples:

#25:  sss(102, 61, 109)

      ┌ angle_opposite_s1 = 102  :  66.9903 degrees ┐
#26:  │ angle_opposite_s2 = 61   :  33.3984 degrees │
      └ angle_opposite_s3 = 109  :  79.6111 degrees ┘

                                ┌┌                    solution_of_triangle = 1                 ┐┐
                                ││ angle_opposite_s2 = 160             :           60.6241 degrees ││
                                ││      third_angle                    :           86.3758 degrees ││
                                │└      third_side                     :              183.24       ┘│
#27:  ssa(100, 160, 33 1°) =    │┌                    solution_of_triangle = 2                 ┐│
                                ││ angle_opposite_s2 = 160             :           119.375 degrees ││
                                ││      third_angle                    :           27.6241 degrees ││
                                └└      third_side                     :              85.1343      ┘┘

                                  ┌ side_opposite_a3 = 58.6 degrees  :      75.7876    ┐
#28:  sas(85.3, 58.6 1°, 23.4) =  │    angle_opposite_s1 = 85.3      :  106.119 degrees │
                                  └    angle_opposite_s2 = 23.4      :  15.2802 degrees ┘

#29:  ssa(100, 160, 53 1°) = no_triangle
```



```
                                  ┌                solution_of_triangle = 1                ┐
                                  │                                                        │
                                  │ angle_opposite_s2 = 100          :       19.9012 degrees│
#30:  ssa(160, 100, 33 1°) =      │                                                        │
                                  │      third_angle                 :       127.098 degrees│
                                  │                                                        │
                                  └      third_side                  :          234.312    ┘

                                     ┌    angle_opposite_s3 = 2987      :  62.6 degrees ┐
                                     │                                                  │
#31:  asa(67.3°, 2987, 50.1°) =      │ side_opposite_a1 = 67.3 degrees  :     3103.82   │
                                     │                                                  │
                                     └ side_opposite_a2 = 50.1 degrees  :     2581.07   ┘

#32:

#33:  The Super Black Box

      code(v) := Σ(VECTOR(IF(v  = 0, i, i, 0), i, 6))
#34:                           i

      c2(v) :=
        If v↓1 v↓4 v↓5 ≥ 0 ∨ (v↓1 v↓4 v↓6 ≥ 0 ∨ (v↓2 v↓4 v↓5 ≥ 0 ∨ (v↓2 v↓5 v↓6 ≥ 0 ∨ (v↓3 v↓4 v↓6 ≥ 0 ∨ v↓3 v↓5 v↓6 ≥ 0))))
#35:       1
           0
           0

      sssx(s1, s2, s3, a1, a2, a3) :=
        If tri_inequ(s1, s2, s3) = 1
#36:       no_triangle
           [LIM(a_, a_, a1) = ca_s1(s1, s2, s3) degrees; LIM(a_, a_, a2) = ca_s1(s2, s3, s1) degrees; LIM(a_, a_, a3) =



        ca_s1(s3, s2, s1) degrees]

      s_s_s(v) := sssx(v , v , v , v , v , v )
#37:                     1   2   3   4   5   6

                                             ┌             lim  s_ = cs_a3(s1, s2, a3)              ┐
                                             │            s_→s3                                     │
                                             │                                                      │
                                             │   lim  a_ = ca_s1(s1, s2, cs_a3(s1, s2, a3)) degrees │
#38:  sasx(s1, s2, s3, a1, a2, a3) :=        │  a_→a1                                               │
                                             │                                                      │
                                             │   lim  a_ = ca_s1(s2, s1, cs_a3(s1, s2, a3)) degrees │
                                             └  a_→a2                                               ┘

      s_a_s(v) :=
        If v↓4 > 0
           sasx(v↓2, v↓3, v↓1, v↓5, v↓6, v↓4)
           0
#39:       If v↓5 > 0
              sasx(v↓1, v↓3, v↓2, v↓4, v↓6, v↓5)
              0
              sasx(v↓1, v↓2, v↓3, v↓4, v↓5, v↓6)

                                            ┌                c_angle(a1, a2)              ┐
                                            │   lim  a_ =  ───────────────── degrees      │
                                            │  a_→a3              1°                      │
                                            │                                             │
#40:  asax(s1, s2, s3, a1, a2, a3) :=       │  lim  s_ = ss_a2(s3, c_angle(a1, a2), a1)   │
                                            │ s_→s1                                       │
                                            │                                             │
                                            │  lim  s_ = ss_a2(s3, c_angle(a1, a2), a2)   │
                                            └ s_→s2                                       ┘

      a_s_a(v) :=
        If v↓1 > 0
           asax(v↓2, v↓3, v↓1, v↓5, v↓6, v↓4)
           0
#41:       If v↓2 > 0
              asax(v↓1, v↓3, v↓2, v↓4, v↓6, v↓5)
              0
              asax(v↓1, v↓2, v↓3, v↓4, v↓5, v↓6)

                                              ┌                      solution_number = k                          ┐
                                              │                                                                   │
                                              │               lim  a_ = sa_s2(s1, s2, a1, k) degrees              │
                                              │              a_→a2                                                │
                                              │                                                                   │
#42:  ssahx(s1, s2, s3, a1, a2, a3, k) :=     │                     c_angle(a1, sa_s2(s1, s2, a1, k) 1°)          │
                                              │       lim  a_ =  ──────────────────────────────────────── degrees │
                                              │      a_→a3                        1°                              │
                                              │                                                                   │
                                              │    lim  s_ = ss_a2(s1, a1, c_angle(a1, sa_s2(s1, s2, a1, k) 1°))  │
                                              └   s_→s3                                                           ┘

      ssax(s1, s2, s3, a1, a2, a3) :=
        If s1 ≥ s2
           ssahx(s1, s2, s3, a1, a2, a3, 1)
#43:       If s2 SIN(a1)/s1 > 1
              no_triangle
              (COPROJECTION(VECTOR([ssahx(s1, s2, s3, a1, a2, a3, k)], k, 1, 2)))↓1
```



```
      aux(v, z1, z2, z3) := ssax(v  , v  , v  , v       , v       , v       )
#44:                               z1   z2   z3   z1 + 3    z2 + 3    z3 + 3

      s_s_a(v) :=
        If v↓2 v↓4 ≥ 0
           aux(v, 1, 2, 3)
           0
           If v↓1 v↓5 ≥ 0
              aux(v, 2, 1, 3)
              0
              If v↓3 v↓4 ≥ 0
#45:             aux(v, 1, 3, 2)
                 0
                 If v↓1 v↓6 ≥ 0
                    aux(v, 3, 1, 2)
                    0
                    If v↓3 v↓5 ≥ 0
                       aux(v, 2, 3, 1)
                       0
                       aux(v, 3, 2, 1)

      sols(v) := SOLVE([v  = t, v  SIN(v ) = v  SIN(v ), v  SIN(v ) = v  SIN(v )], [v , v , v ])
#46:                     1       1      5      2      4    2      6      3      5     1   2   3

      special(v) :=
        If v↓4 + v↓5 + v↓6 ≠ 180 1°
#47:       "Sum of angles incorrect!"
           ["only similar triangles are possible:"; (sols(v))↓1]

                                            ⎡   lim   s_ = ss_a2(s1, a1, c_angle(a1, a2)) ⎤
                                            ⎢ s_→s3                                       ⎥
                                            ⎢                                             ⎥
                                            ⎢        lim   s_ = ss_a2(s1, a1, a2)         ⎥
#48:  saax(s1, s2, s3, a1, a2, a3) :=       ⎢      s_→s2                                  ⎥
                                            ⎢                                             ⎥
                                            ⎢                  c_angle(a1, a2)            ⎥
                                            ⎢     lim   a_ =  ───────────────── degrees   ⎥
                                            ⎣   a_→a3                1°                   ⎦

      aux2(v, z1, z2, z3) := saax(v  , v  , v  , v       , v       , v       )
#49:                                z1   z2   z3   z1 + 3    z2 + 3    z3 + 3

      s_a_a(v) :=
        If v↓1 v↓5 ≥ 0
           aux2(v, 1, 2, 3)
           0
           If v↓1 v↓6 ≥ 0
              aux2(v, 1, 3, 2)
              0
              If v↓2 v↓4 ≥ 0
#50:             aux2(v, 2, 1, 3)
                 0
                 If v↓2 v↓6 ≥ 0
                    aux2(v, 2, 3, 1)
                    0
                    If v↓3 v↓4 ≥ 0
                       aux2(v, 3, 1, 2)
                       0
                       aux2(v, 3, 2, 1)
      triangle(v) :=
        If c2(v) = 1
           s_a_a(v)
           If code(v) = 15
              special(v)
              If code(v) = 6
#51:             s_s_s(v)
                 If code(v) = 9
                    s_a_s(v)
                    If code(v) = 12
                       a_s_a(v)
                       s_s_a(v)

#52:  Examples:

                                                          ⎡ side_b = 53.9128   ⎤
#53:  triangle([side_a, side_b, 102, 118.34°, β, 41.26°]) = ⎢ side_a = 136.130   ⎥
                                                          ⎣ β = 20.4 degrees   ⎦

                                                   ⎡   solution_number = 1    ⎤
#54:  triangle([80, 120, PR, a_QRP, 20°, a_PQR]) =  ⎢ a_QRP = 13.1801 degrees  ⎥
                                                   ⎢ a_PQR = 146.819 degrees  ⎥
                                                   ⎣       PR = 192.014       ⎦

#55:  triangle([120, 80, PR, a_QRP, 46°, a_PQR]) = no_triangle
```



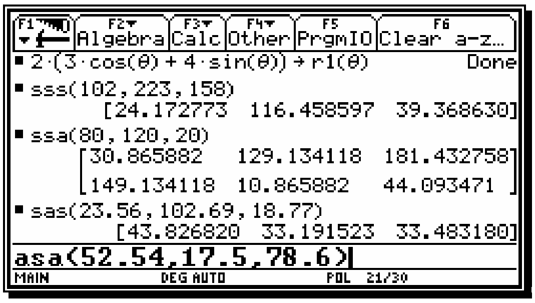
```
#56:  triangle([120, 80, PR, a_QRP, 20°, a_PQR])

       ⎡⎡    solution_number = 1     ⎤ ⎡    solution_number = 2     ⎤⎤
       ⎢⎢                            ⎥ ⎢                            ⎥⎥
       ⎢⎢  a_QRP = 30.8658 degrees   ⎥ ⎢  a_QRP = 149.134 degrees   ⎥⎥
#57:   ⎢⎢                            ⎥,⎢                            ⎥⎥
       ⎢⎢  a_PQR = 129.134 degrees   ⎥ ⎢  a_PQR = 10.8658 degrees   ⎥⎥
       ⎢⎢                            ⎥ ⎢                            ⎥⎥
       ⎣⎣        PR = 181.432        ⎦ ⎣        PR = 44.0934        ⎦⎦

#58:  triangle([a, b, c, 30°, 60°, 90°])

#59:  simplified

       ⎡ only similar triangles are possible: ⎤
#60:   ⎢                                      ⎥
       ⎣     a = t ∧ b = √3 t ∧ c = 2 t       ⎦

#61:  approximated

       ⎡ only similar triangles are possible: ⎤
#62:   ⎢                                      ⎥
       ⎣   a = t ∧ b = 1.73205 t ∧ c = 2 t    ⎦
```

`triangle(v)` is the main function. According to the code numbers it assigns each problem to its solution procedure.

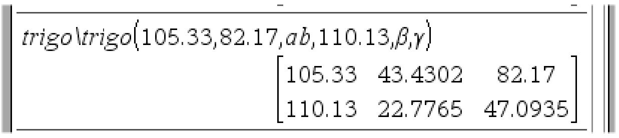
There is one more improvement possible. I will try to append the feature to plot the triangles. I have in mind a function PLOT(v) which returns the coordinates of the vertices. One vertex will be in the origin and one side will lie on the x-axe. Maybe you can find it on the file on the diskette.

I'll close with the same example which was worked through using the Black Box in *DNL#9*. The problem was to solve a Pentagon ABCDE. Do it once more testing the Super Box:

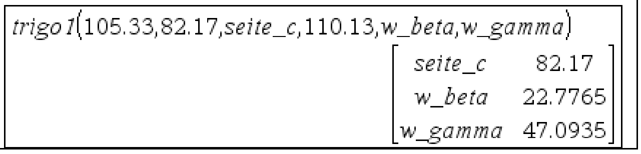
(AB = 30, BC = 52, AE = 30, AD = 65, BD = 60, EC = 68, ∠.EAB = 106.1°; ED = ?, CD = ?
Solution: ED = 45.8171; CD = 36.3114)



```
■ 2·(3·cos(θ) + 4·sin(θ)) → r1(θ)             Done
■ sss(102, 223, 158)
          [24.172773  116.458597  39.368630]
■ ssa(80, 120, 20)
       ⎡30.865882   129.134118  181.432758⎤
       ⎣149.134118  10.865882   44.093471 ⎦
■ sas(23.56, 102.69, 18.77)
           [43.826820  33.191523  33.483180]
asa(52.54,17.5,78.6)
MAIN            DEG AUTO          POL   21/30
```



```
trigo\trigo(105.33,82.17,ab,110.13,β,γ)
                          ⎡105.33  43.4302   82.17 ⎤
                          ⎣110.13  22.7765  47.0935⎦
```



```
trigo1(105.33,82.17,seite_c,110.13,w_beta,w_gamma)
                                   ⎡ seite_c    82.17 ⎤
                                   ⎢ w_beta    22.7765⎥
                                   ⎣w_gamma   47.0935⎦
```

[1] The *DERIVE* News Letter #9, March 1993, p15-p17
[2] Mathematik unterrichten mit *DERIVE*, Bernhard Kutzler, Addison-Wesley 1995, ISBN 3 89319 860 1
[3] Improving Mathematics with *DERIVE*, Bernhard Kutzler, Chartwell-Bratt 1996, ISBN 0 86238 422 2

As you can see on the screen dump I have tried successfully to implement the "Black Box" on the TI-92. I will go further to implement the Superbox, too. Maybe you can find it on the diskette of ´96.

Comment in 2010: I did not develop the Super Box for the TI-92. But I can offer the SUPER BOX for the TI-Nspire in combination with a short introduction in programming the TI-Nspire in combination with developing a "TRIGO-Library" for the next DNL#77. Josef

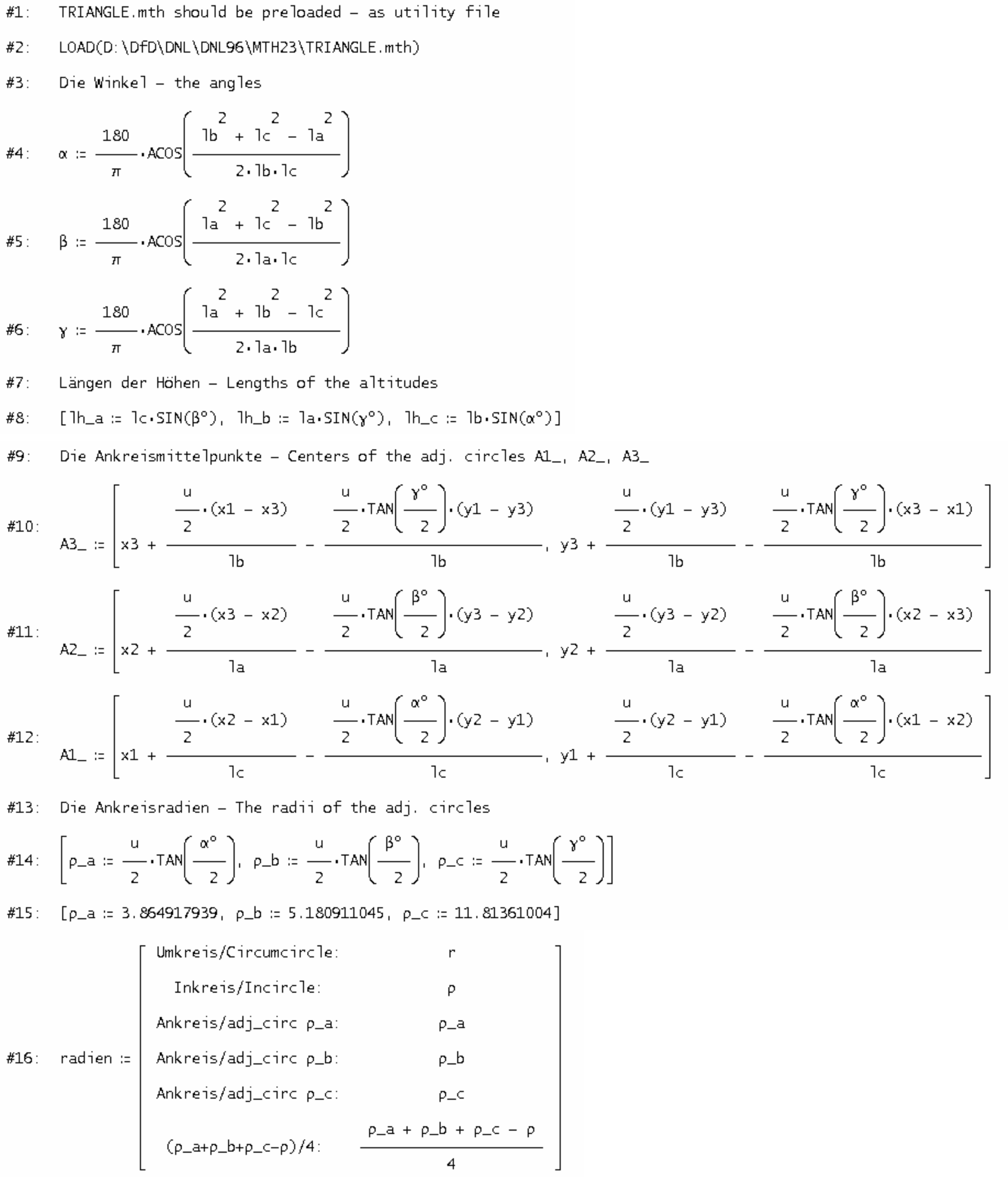
DREIECK.MTH -
- TRIANGLE.MTH (2)

Bernhard Wadsack, Vienna, Austria

This is the second part of Bernhard's article to plot a triangle together with its remarkable points and circles. Bernhard sent a paper which he gives his pupils to work with this Utility file. I add this paper at the end of the contribution.

You should set the following Options:

Options Input Word, Plot Options State Connected, Options Auto Change Color No



#1: TRIANGLE.mth should be preloaded - as utility file

#2: LOAD(D:\DfD\DNL\DNL96\MTH23\TRIANGLE.mth)

#3: Die Winkel - the angles

#4: $$\alpha := \frac{180}{\pi}\cdot \mathrm{ACOS}\left(\frac{lb^2 + lc^2 - la^2}{2\cdot lb\cdot lc}\right)$$

#5: $$\beta := \frac{180}{\pi}\cdot \mathrm{ACOS}\left(\frac{la^2 + lc^2 - lb^2}{2\cdot la\cdot lc}\right)$$

#6: $$\gamma := \frac{180}{\pi}\cdot \mathrm{ACOS}\left(\frac{la^2 + lb^2 - lc^2}{2\cdot la\cdot lb}\right)$$

#7: Längen der Höhen - Lengths of the altitudes

#8: [lh_a := lc·SIN(β°), lh_b := la·SIN(γ°), lh_c := lb·SIN(α°)]

#9: Die Ankreismittelpunkte - Centers of the adj. circles A1_, A2_, A3_

#10: $$A3\_ := \left[x3 + \frac{\frac{u}{2}\cdot(x1 - x3)}{lb} - \frac{\frac{u}{2}\cdot \mathrm{TAN}\left(\frac{\gamma°}{2}\right)\cdot(y1 - y3)}{lb},\ y3 + \frac{\frac{u}{2}\cdot(y1 - y3)}{lb} - \frac{\frac{u}{2}\cdot \mathrm{TAN}\left(\frac{\gamma°}{2}\right)\cdot(x3 - x1)}{lb}\right]$$

#11: $$A2\_ := \left[x2 + \frac{\frac{u}{2}\cdot(x3 - x2)}{la} - \frac{\frac{u}{2}\cdot \mathrm{TAN}\left(\frac{\beta°}{2}\right)\cdot(y3 - y2)}{la},\ y2 + \frac{\frac{u}{2}\cdot(y3 - y2)}{la} - \frac{\frac{u}{2}\cdot \mathrm{TAN}\left(\frac{\beta°}{2}\right)\cdot(x2 - x3)}{la}\right]$$

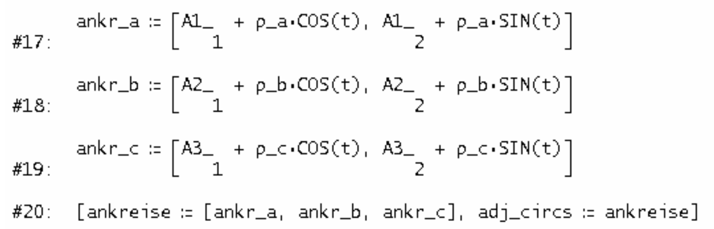
#12: $$A1\_ := \left[x1 + \frac{\frac{u}{2}\cdot(x2 - x1)}{lc} - \frac{\frac{u}{2}\cdot \mathrm{TAN}\left(\frac{\alpha°}{2}\right)\cdot(y2 - y1)}{lc},\ y1 + \frac{\frac{u}{2}\cdot(y2 - y1)}{lc} - \frac{\frac{u}{2}\cdot \mathrm{TAN}\left(\frac{\alpha°}{2}\right)\cdot(x1 - x2)}{lc}\right]$$

#13: Die Ankreisradien - The radii of the adj. circles

#14: $$\left[\rho\_a := \frac{u}{2}\cdot \mathrm{TAN}\left(\frac{\alpha°}{2}\right),\ \rho\_b := \frac{u}{2}\cdot \mathrm{TAN}\left(\frac{\beta°}{2}\right),\ \rho\_c := \frac{u}{2}\cdot \mathrm{TAN}\left(\frac{\gamma°}{2}\right)\right]$$

#15: [ρ_a := 3.864917939, ρ_b := 5.180911045, ρ_c := 11.81361004]

#16: $$\mathrm{radien} := \begin{bmatrix} \text{Umkreis/Circumcircle:} & r \\ \text{Inkreis/Incircle:} & \rho \\ \text{Ankreis/adj\_circ } \rho\text{\_a:} & \rho\_a \\ \text{Ankreis/adj\_circ } \rho\text{\_b:} & \rho\_b \\ \text{Ankreis/adj\_circ } \rho\text{\_c:} & \rho\_c \\ (\rho\_a+\rho\_b+\rho\_c-\rho)/4\text{:} & \frac{\rho\_a + \rho\_b + \rho\_c - \rho}{4} \end{bmatrix}$$



#17: $\text{ankr\_a} := \left[\text{A1\_}_{1} + \rho\text{\_a}\cdot\text{COS}(t),\ \text{A1\_}_{2} + \rho\text{\_a}\cdot\text{SIN}(t)\right]$

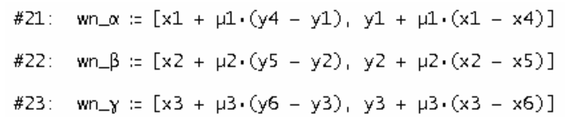
#18: $\text{ankr\_b} := \left[\text{A2\_}_{1} + \rho\text{\_b}\cdot\text{COS}(t),\ \text{A2\_}_{2} + \rho\text{\_b}\cdot\text{SIN}(t)\right]$

#19: $\text{ankr\_c} := \left[\text{A3\_}_{1} + \rho\text{\_c}\cdot\text{COS}(t),\ \text{A3\_}_{2} + \rho\text{\_c}\cdot\text{SIN}(t)\right]$

```
#20:  [ankreise := [ankr_a, ankr_b, ankr_c], adj_circs := ankreise]
```

Man kann die Ankreismittelpunkte auf eine andere einfache Art ermitteln: die folgenden drei Winkelsymmetralen halbieren die Außenwinkel und stehen auf die ursprünglichen Winkelsymmetralen w_α, w_ß und w_δ normal. In deren Schnittpunkten finden sich die Mittelpunkte.

We can find the centres of the adjacent circles in another way: the following three angle bisectors are perpendiculars to w_α, w_ß and w_δ and bisect the external angles. Their intersection points are the centres.



```
#21:  wn_α := [x1 + μ1·(y4 - y1), y1 + μ1·(x1 - x4)]

#22:  wn_ß := [x2 + μ2·(y5 - y2), y2 + μ2·(x2 - x5)]

#23:  wn_γ := [x3 + μ3·(y6 - y3), y3 + μ3·(x3 - x6)]
```

```
"[wn_δ,wn_ß] --> A1_ ,[wn_α,wn_ß] --> A3_ ,[wn_α,wn_δ] --> A2_ !"
```

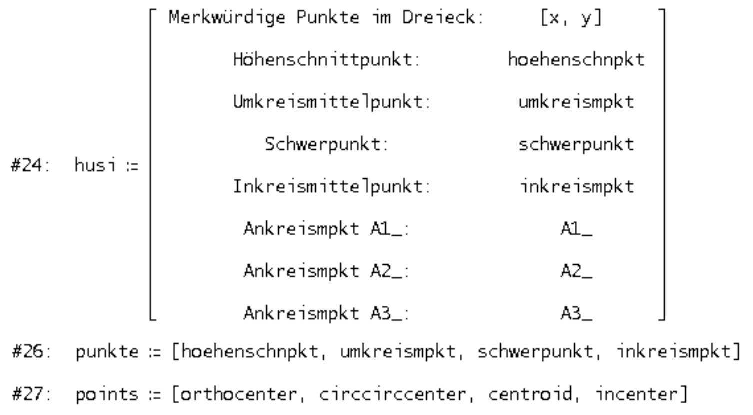
Mit den Koordinaten der Eckpunkte löst man die folgenden drei Gleichungen und substituiert die jeweils erhaltenen Parameter.

Using the coordinates of the vertices you have to solve the three following equations to obtain solutions for the parameters which have to be substituted into the appropriate equations.

```
wn_ß=wn_δ

wn_α=wn_ß

wn_α=wn_δ
```

Nun bilden wir aus allen merkwürdigen Punkten (H, U, S, I) einen Vektor und generieren weitere übersichtliche Tabellen. Im Anschluss daran berechnen wir die Gleichung der Eulerschen Geraden und zeigen, dass der Schwerpunkt die Strecke UH immer im Verhältnis 1:2 teilt.

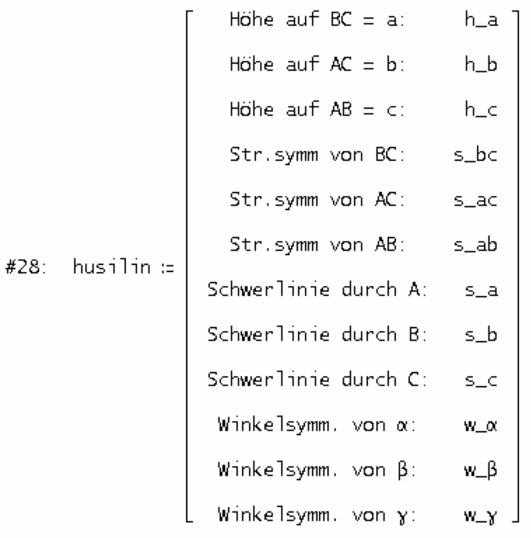
Now we use the points H, U, S and I to create a vector and we generate more tables of results. Then we calculate the equation of Euler's line and we show that the gravity centre S divides the segment UH in the ratio of 1:2.



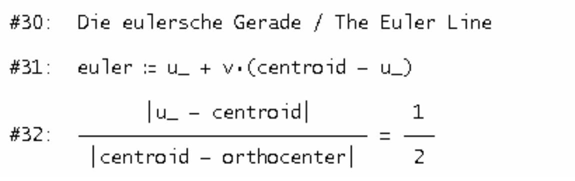
#24: $\text{husi} := \begin{bmatrix} \text{Merkwürdige Punkte im Dreieck:} & [x,\ y] \\ \text{Höhenschnittpunkt:} & \text{hoehenschnpkt} \\ \text{Umkreismittelpunkt:} & \text{umkreismpkt} \\ \text{Schwerpunkt:} & \text{schwerpunkt} \\ \text{Inkreismittelpunkt:} & \text{inkreismpkt} \\ \text{Ankreismpkt A1\_:} & \text{A1\_} \\ \text{Ankreismpkt A2\_:} & \text{A2\_} \\ \text{Ankreismpkt A3\_:} & \text{A3\_} \end{bmatrix}$

```
#26:  punkte := [hoehenschnpkt, umkreismpkt, schwerpunkt, inkreismpkt]

#27:  points := [orthocenter, circcirccenter, centroid, incenter]
```

```
[punkte:=[h_,u_,s_,i_],hoehen:=[h_a,h_b,h_c]]

[streckensymmetralen:=[s_bc,s_ac,s_ab],schwerlinien:=[s_a,s_b,s_c]]

winkelsymmetralen:=[w_α,w_ß,w_δ]
```



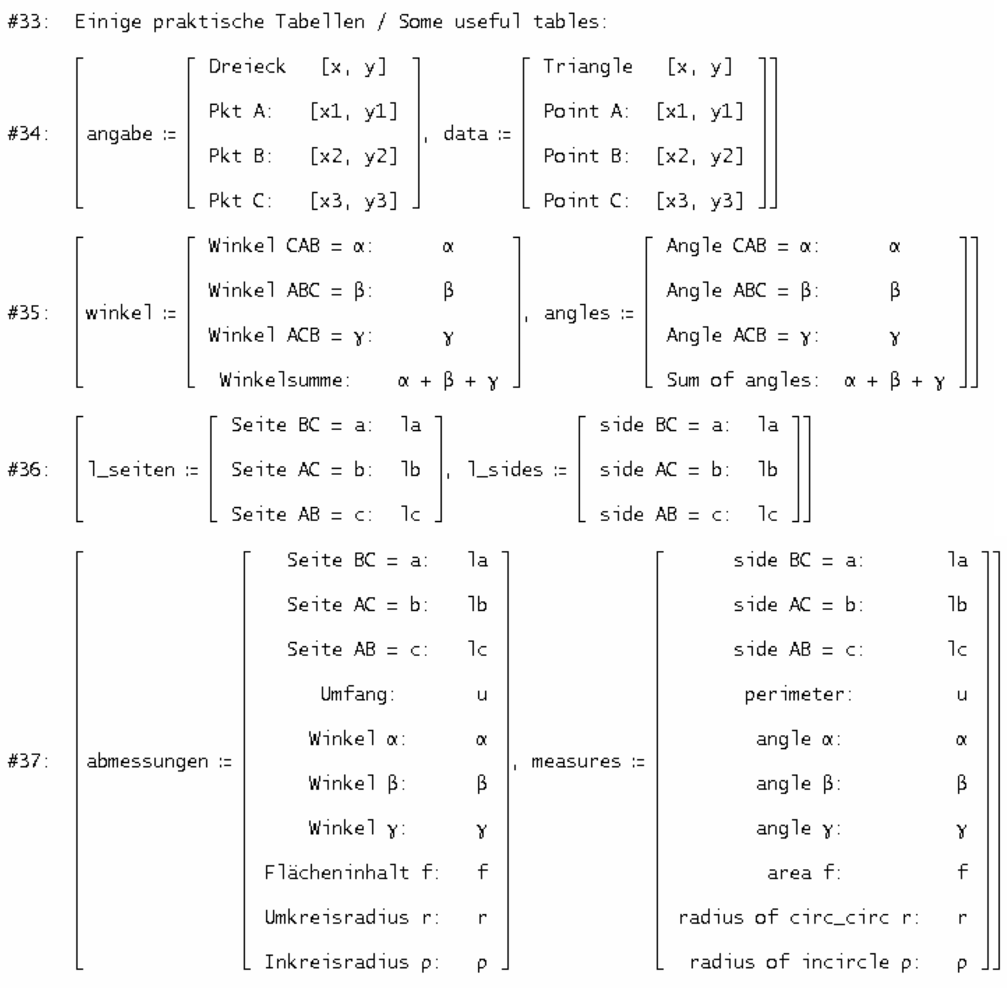
#28: $$\text{husilin} := \begin{bmatrix} \text{Höhe auf BC = a:} & \text{h\_a} \\ \text{Höhe auf AC = b:} & \text{h\_b} \\ \text{Höhe auf AB = c:} & \text{h\_c} \\ \text{Str.symm von BC:} & \text{s\_bc} \\ \text{Str.symm von AC:} & \text{s\_ac} \\ \text{Str.symm von AB:} & \text{s\_ab} \\ \text{Schwerlinie durch A:} & \text{s\_a} \\ \text{Schwerlinie durch B:} & \text{s\_b} \\ \text{Schwerlinie durch C:} & \text{s\_c} \\ \text{Winkelsymm. von } \alpha\text{:} & \text{w\_}\alpha \\ \text{Winkelsymm. von } \beta\text{:} & \text{w\_}\beta \\ \text{Winkelsymm. von } \gamma\text{:} & \text{w\_}\gamma \end{bmatrix}$$



#30: Die eulersche Gerade / The Euler Line

#31: $\text{euler} := \text{u\_} + v\cdot(\text{centroid} - \text{u\_})$

#32: $$\frac{|\text{u\_} - \text{centroid}|}{|\text{centroid} - \text{orthocenter}|} = \frac{1}{2}$$



#33: Einige praktische Tabellen / Some useful tables:

#34: $$\left[\text{angabe} := \begin{bmatrix} \text{Dreieck} & [x, y] \\ \text{Pkt A:} & [x1, y1] \\ \text{Pkt B:} & [x2, y2] \\ \text{Pkt C:} & [x3, y3] \end{bmatrix}, \text{data} := \begin{bmatrix} \text{Triangle} & [x, y] \\ \text{Point A:} & [x1, y1] \\ \text{Point B:} & [x2, y2] \\ \text{Point C:} & [x3, y3] \end{bmatrix}\right]$$

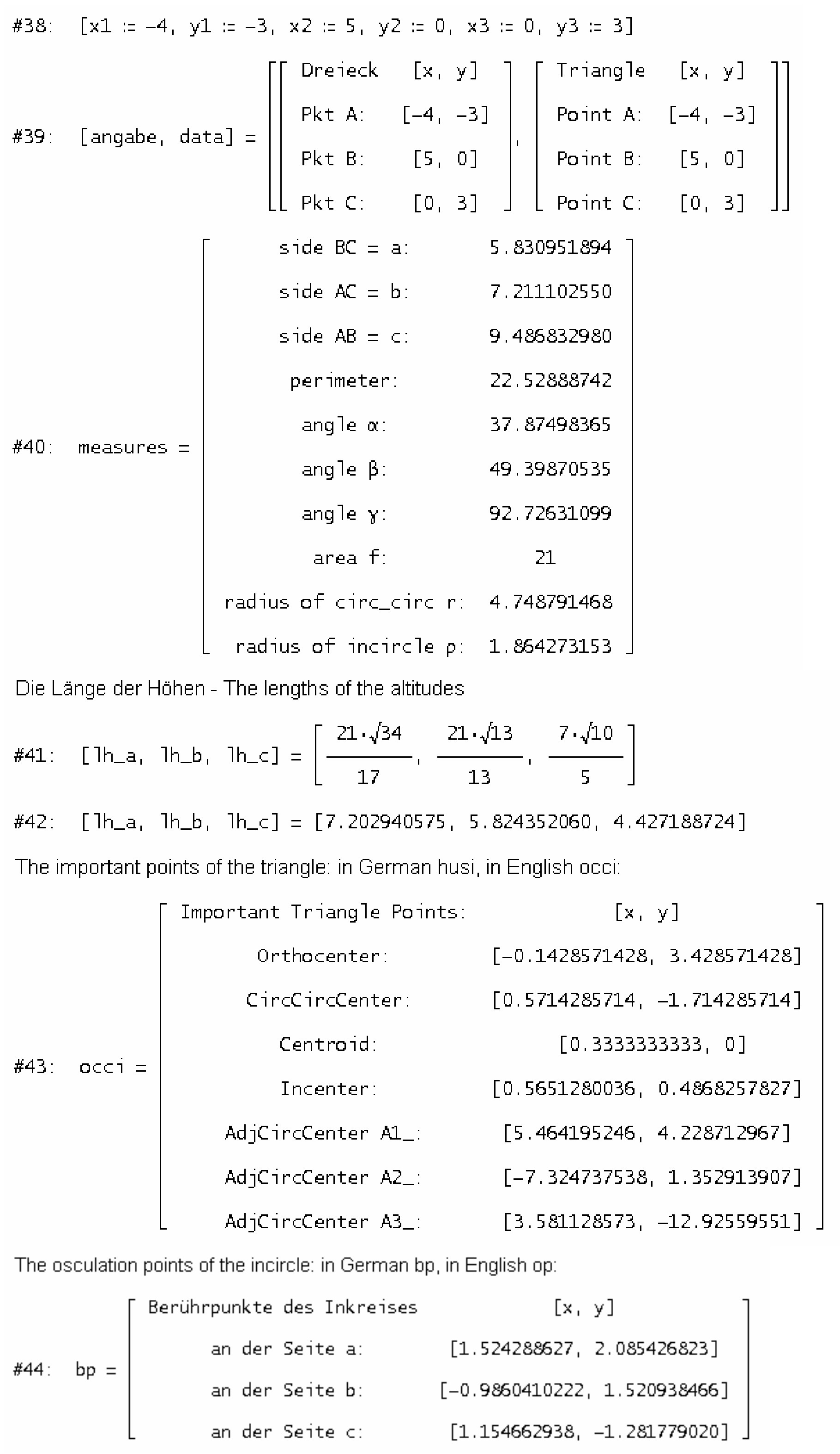
#35: $$\left[\text{winkel} := \begin{bmatrix} \text{Winkel CAB = } \alpha\text{:} & \alpha \\ \text{Winkel ABC = } \beta\text{:} & \beta \\ \text{Winkel ACB = } \gamma\text{:} & \gamma \\ \text{Winkelsumme:} & \alpha + \beta + \gamma \end{bmatrix}, \text{angles} := \begin{bmatrix} \text{Angle CAB = } \alpha\text{:} & \alpha \\ \text{Angle ABC = } \beta\text{:} & \beta \\ \text{Angle ACB = } \gamma\text{:} & \gamma \\ \text{Sum of angles:} & \alpha + \beta + \gamma \end{bmatrix}\right]$$

#36: $$\left[\text{l\_seiten} := \begin{bmatrix} \text{Seite BC = a:} & \text{la} \\ \text{Seite AC = b:} & \text{lb} \\ \text{Seite AB = c:} & \text{lc} \end{bmatrix}, \text{l\_sides} := \begin{bmatrix} \text{side BC = a:} & \text{la} \\ \text{side AC = b:} & \text{lb} \\ \text{side AB = c:} & \text{lc} \end{bmatrix}\right]$$

#37: $$\left[\text{abmessungen} := \begin{bmatrix} \text{Seite BC = a:} & \text{la} \\ \text{Seite AC = b:} & \text{lb} \\ \text{Seite AB = c:} & \text{lc} \\ \text{Umfang:} & \text{u} \\ \text{Winkel } \alpha\text{:} & \alpha \\ \text{Winkel } \beta\text{:} & \beta \\ \text{Winkel } \gamma\text{:} & \gamma \\ \text{Flächeninhalt f:} & \text{f} \\ \text{Umkreisradius r:} & \text{r} \\ \text{Inkreisradius } \rho\text{:} & \rho \end{bmatrix}, \text{measures} := \begin{bmatrix} \text{side BC = a:} & \text{la} \\ \text{side AC = b:} & \text{lb} \\ \text{side AB = c:} & \text{lc} \\ \text{perimeter:} & \text{u} \\ \text{angle } \alpha\text{:} & \alpha \\ \text{angle } \beta\text{:} & \beta \\ \text{angle } \gamma\text{:} & \gamma \\ \text{area f:} & \text{f} \\ \text{radius of circ\_circ r:} & \text{r} \\ \text{radius of incircle } \rho\text{:} & \rho \end{bmatrix}\right]$$

Für eine spezielle Angabe werden einige Ausgaben wiedergegeben.

For one special triangle you can see some results.



```
#38:  [x1 := -4, y1 := -3, x2 := 5, y2 := 0, x3 := 0, y3 := 3]

                        ⎡⎡ Dreieck    [x, y]  ⎤  ⎡ Triangle   [x, y]  ⎤⎤
                        ⎢⎢                    ⎥  ⎢                    ⎥⎥
                        ⎢⎢ Pkt A:   [-4, -3]  ⎥  ⎢ Point A:  [-4, -3] ⎥⎥
#39:  [angabe, data] =  ⎢⎢                    ⎥, ⎢                    ⎥⎥
                        ⎢⎢ Pkt B:    [5, 0]   ⎥  ⎢ Point B:   [5, 0]  ⎥⎥
                        ⎢⎢                    ⎥  ⎢                    ⎥⎥
                        ⎣⎣ Pkt C:    [0, 3]   ⎦  ⎣ Point C:   [0, 3]  ⎦⎦

                  ⎡       side BC = a:       5.830951894 ⎤
                  ⎢       side AC = b:       7.211102550 ⎥
                  ⎢       side AB = c:       9.486832980 ⎥
                  ⎢        perimeter:        22.52888742 ⎥
                  ⎢         angle α:         37.87498365 ⎥
#40:  measures =  ⎢         angle β:         49.39870535 ⎥
                  ⎢         angle γ:         92.72631099 ⎥
                  ⎢          area f:             21      ⎥
                  ⎢ radius of circ_circ r:   4.748791468 ⎥
                  ⎣  radius of incircle ρ:   1.864273153 ⎦
```

Die Länge der Höhen - The lengths of the altitudes

$$\#41:\quad [lh\_a,\ lh\_b,\ lh\_c] = \left[\frac{21\cdot\sqrt{34}}{17},\ \frac{21\cdot\sqrt{13}}{13},\ \frac{7\cdot\sqrt{10}}{5}\right]$$

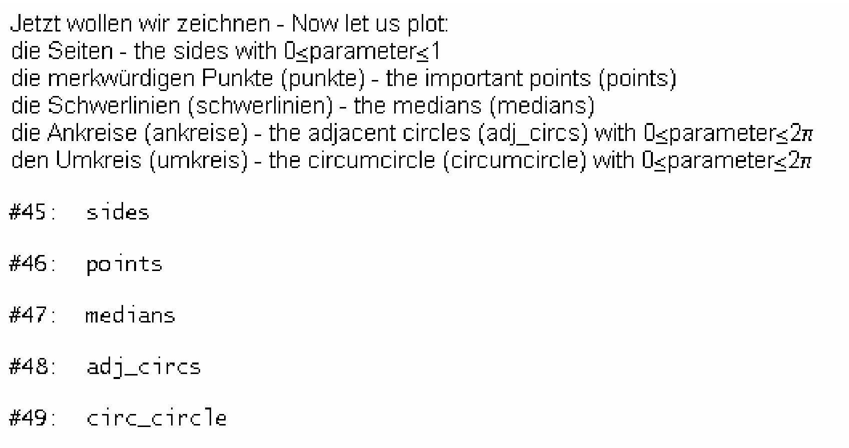
```
#42:  [lh_a, lh_b, lh_c] = [7.202940575, 5.824352060, 4.427188724]
```

The important points of the triangle: in German husi, in English occi:

```
              ⎡ Important Triangle Points:              [x, y]              ⎤
              ⎢       Orthocenter:          [-0.1428571428, 3.428571428]    ⎥
              ⎢      CircCircCenter:        [0.5714285714, -1.714285714]    ⎥
              ⎢         Centroid:                 [0.3333333333, 0]         ⎥
#43:  occi =  ⎢         Incenter:           [0.5651280036, 0.4868257827]    ⎥
              ⎢   AdjCircCenter A1_:         [5.464195246, 4.228712967]     ⎥
              ⎢   AdjCircCenter A2_:         [-7.324737538, 1.352913907]    ⎥
              ⎣   AdjCircCenter A3_:         [3.581128573, -12.92559551]    ⎦
```

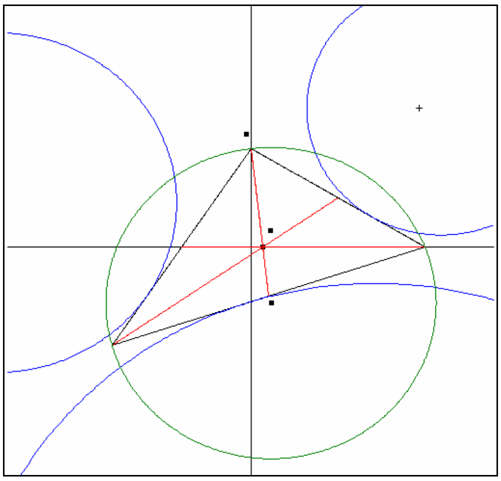
The osculation points of the incircle: in German bp, in English op:

```
            ⎡ Berührpunkte des Inkreises            [x, y]              ⎤
            ⎢       an der Seite a:        [1.524288627, 2.085426823]    ⎥
#44:  bp =  ⎢       an der Seite b:       [-0.9860410222, 1.520938466]   ⎥
            ⎣       an der Seite c:        [1.154662938, -1.281779020]   ⎦
```



Jetzt wollen wir zeichnen - Now let us plot:
die Seiten - the sides with $0 \le parameter \le 1$
die merkwürdigen Punkte (punkte) - the important points (points)
die Schwerlinien (schwerlinien) - the medians (medians)
die Ankreise (ankreise) - the adjacent circles (adj_circs) with $0 \le parameter \le 2\pi$
den Umkreis (umkreis) - the circumcircle (circumcircle) with $0 \le parameter \le 2\pi$

```
#45:   sides

#46:   points

#47:   medians

#48:   adj_circs

#49:   circ_circle
```



eckpunkte	vertices		
dreieck	triangle		
angabe	data		
winkel	angles		
l_seiten	l_sides	Seitenlängen, Lengths of sides	
abmessungen	measures		
bp	op	Inkreisberührpunkte, oscpts of incircle	
fusspunkte	pedpoints	Höhenfußpkte, pedalpoints of altitudes	
seiten	sides	Seiten, sides	$0 \le p \le 1$
umkreismpkt	circcirccenter	Umkreismittelpunkt, circumcenter	
umkreis	circ_circle		$0 \le p \le 2\pi$
hoehenschnpkt	orthocenter		
schwerpunkt	centroid		
inkreis	incircle		$0 \le p \le 2\pi$
inkreismpkt	incenter		
seitsymm	perpbisecs		$0 \le p \le 1$
winkelsymm	angbiss		$0 \le p \le 1$
hoehen	altitudes		$-5 \le p \le 5$
schwerlinien	medians		$0 \le p \le 1$
husi	occi	alle merkw. Punkte, all interesting points	
punkte	points	nur H, U, I, S, only O, C, C, I	
husilin	occilin		$0 \le p \le 1$
euler	euler	eulersche Gerade, Euler line	
Ankreise	adj_circs		$0 \le p \le 2\pi$
radien	radii		

Working sequence for the teaching unit

The remarkable points in the triangle

using the utility file DREIECK.MTH

☞ Turn on your computer and start the CAS *DERIVE* as usual.

☞ Load DREIECK.MTH as an utility file. You will not see the expressions on the screen but they are available for further usage. (Principle of a Black-Box!)
Transfer Load Utility DREIECK ·

☞ Read carefully the instruction how to use DREIECK.MTH (3 pages)!

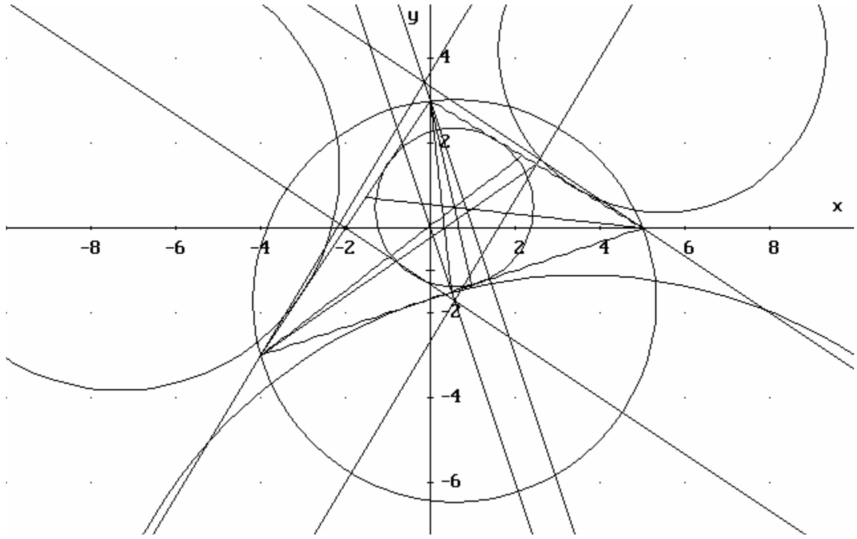
☞ The following triangle shall be calculated and then be plotted using *DERIVE*:
ΔABC[A(–4|–3), B(5|0), C(0|3)]
Author [x1:= –4,y1:= –3,x2:=5,y2:=0,x3:=0,y3:=3] ·

☞ Now execute the *DERIVE* calculations. Keep attention to the instruction (page 1 bottom and page 2 top). The expressions like *angabe, abmessen, ... husilin* have to be edited in the Author line without quotation marks. (That are prepared definitions from the Black-Box!).

☞ Regarding to instruction on page 2 the expressions needed for plotting the triangle like *seiten, schwerlinien, ... ,inkreis, ankreise* have to be Authored and then Simplified or approXimated. Then you can plot them.

☞ For the 2D-plot use an Overlay window. So the full screen is at your disposal for the construction. In the 2D-plot window set *Options Color Auto No* ·

☞ Plot the sides in yellow: *Options Color Plot 14* ·
Plot the gravity lines in light red: *Options Color Plot 12* ·
Plot the altitudes in light green: *Options Color Plot 10* ·
Plot the perpendicular bisectors and the circumcircle in light blue
Plot the angle bisectors, the incircle and the adjacent circles in white!



➨ Important:
Save your work as:
MY_1WORK.MTH on the diskette!

➨ Important:
When you are ready with your picture then show it to the teacher!!

–

An Exercise to Motivate Eigenvalues

Carl Leinbach & Marvin Brubaker, USA

In this section we will give a visual demonstration of the effect of applying a 2 by 2 matrix to a simple two dimensional figure. We will see this figure compressed and turned in the direction of the dominant eigenvector of the matrix.

Before starting this demonstration, create a Plot window aside of the Algebra window and choose the Connected option for Plotting points.

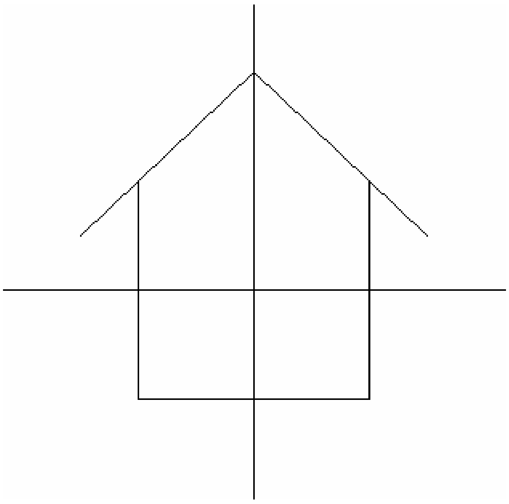
First we create the figure to which we are going to apply our matrix. Enter a 2 by *n* matrix that gives the coordinates of a house. The first row contains the x-coordinates, the second row the *y*-coordinates.

```
h:=[-2,-2,-3,0,0,3,2,2,2,-2;-2,2,1,4,4,1,2,-2,-2,-2]
```

In order to plot the figure represented by our matrix create a vector of 2 by 2 matrices from the 2 by *n* matrix. This gives a vector of pairs of points that we can plot. NOTE a↓i↓j is *DERIVE*'s way of denoting a_{ij}. The down arrow (↓) can be replaced with the phrase SUB. (Or in *DERIVE* 2.x versions use the ELEMENT(a,i,j) command).

```
PMAT(b) := VECTOR([b↓1↓i,b↓1↓(i+1);b↓2↓i,b↓2↓(i + 1)]',i,DIMENSION(b')-1)
```

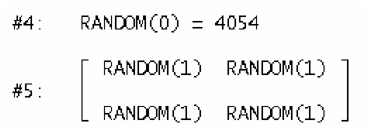
When we apply PMAT to *h* from above and plot the result we see:



For the matrix, *a*, that we will study, generate a random matrix. *DERIVE* has a random number generator called RANDOM. RANDOM(1) generates numbers between 0 and 1. It is almost certain (but not guaranteed) that the following statement will generate a matrix that has a *dominant eigenvalue*[1]. Edit

```
[RANDOM(1),RANDOM(1);RANDOM(1),RANDOM(1)]
```

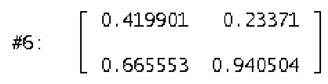
(To be sure to obtain different results at any start of the file Edit and Simplify first the expression RANDOM(0)).



#4: RANDOM(0) = 4054

#5: $\begin{bmatrix} \text{RANDOM(1)} & \text{RANDOM(1)} \\ \text{RANDOM(1)} & \text{RANDOM(1)} \end{bmatrix}$

Approximate this expression and *then* assign that result to the matrix, *a*. Here is one example of approximating the above



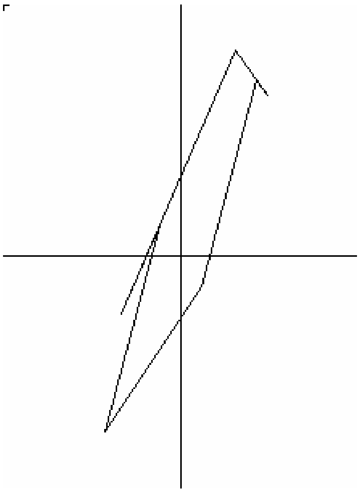
#6: $\begin{bmatrix} 0.419901 & 0.23371 \\ 0.665553 & 0.940504 \end{bmatrix}$

Now, we assign this result to the matrix, *a*. We *must* do things in this order. Otherwise, the matrix *a* will change each time it is applied. Use the F3 key to Edit

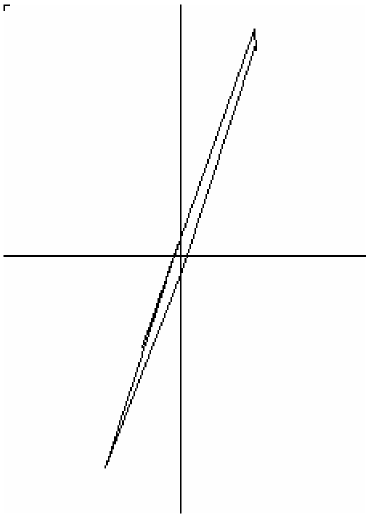
```
a ≔ [0.419901, 0.23371; 0.665553, 0.940504]
```

Let's see what happens to the house under the action of *a*.

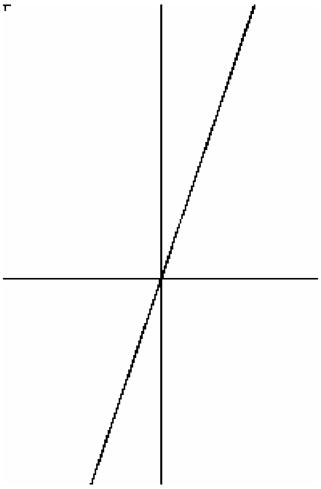
Clear the plot window and first approXimate then **P**lot PMAT(a·h).



Let's apply *a* again. **P**lotting PMAT(a^2·h):



Now we apply the 10-th power of *a* to *h*. Change the scale in the plot window to *x* = 0.02, *y* = 0.02 and **P**lot PMAT(a^10·h):



It is so ´squashed´ it looks like a line. If we apply *a* to *h* any more times we will still get this line. It appears to stay fixed under the action of *a*. If so, any point on the line will just map onto another point on the line, i.e.

$$a \cdot x = \delta x$$

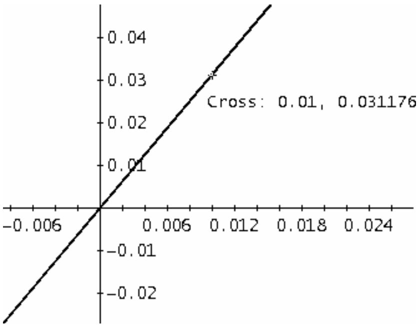
for any point, *x*, on the line we see on the screen.

Let's find *δ*. Move the crosshair to the point with the *x*-coordinate 0.01. Our reading of this *y*-coordinate is 0.0312.

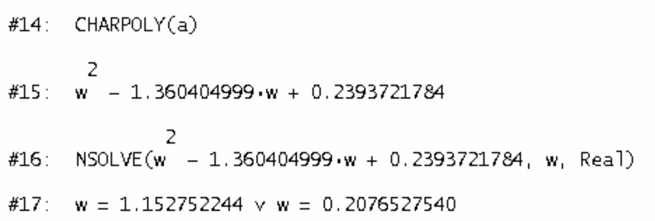
Now let's see what happens to this point under the action of a. Author

`a . [[0.01],[0.0312]]`

The result is: `[0.01149076; 0.03599926]`. Dividing the x-coordinate of the above by 0.01 we have an estimate for δ of δ= 1.14907.

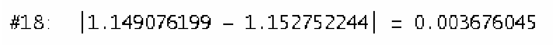


We can check this experimental result by finding the roots of the Characteristic Polynomial. If we Simplify `CHARPOLY(a)`, we have:



```
#14:  CHARPOLY(a)

       2
#15:  w  - 1.360404999·w + 0.2393721784

              2
#16:  NSOLVE(w  - 1.360404999·w + 0.2393721784, w, Real)

#17:  w = 1.152752244 ∨ w = 0.2076527540
```

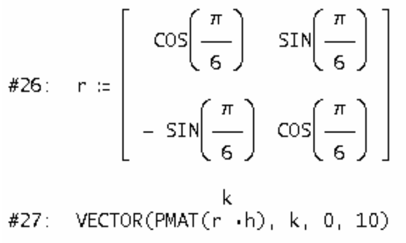
The largest root of this polynomial is w = 1.15275.



```
#18:  |1.149076199 - 1.152752244| = 0.003676045
```

This differs from our experimental result by only 0.00368. Not bad.

Reset the screen and define r as follows:

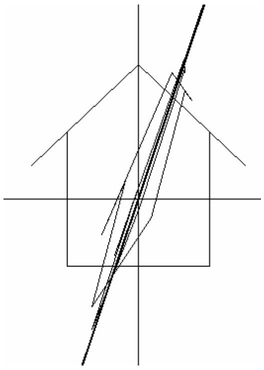


#26: $r := \begin{bmatrix} \mathrm{COS}\left(\frac{\pi}{6}\right) & \mathrm{SIN}\left(\frac{\pi}{6}\right) \\ -\mathrm{SIN}\left(\frac{\pi}{6}\right) & \mathrm{COS}\left(\frac{\pi}{6}\right) \end{bmatrix}$

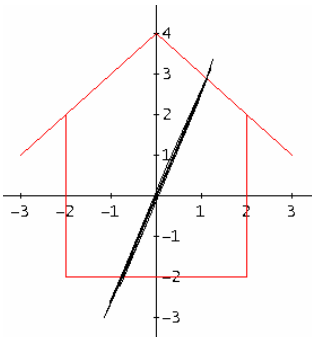
```
                     k
#27:  VECTOR(PMAT(r ·h), k, 0, 10)
```

Repeat the above exercise replacing a with r. Are the results the same? Check out the roots of the Characteristic Polynomial for r.

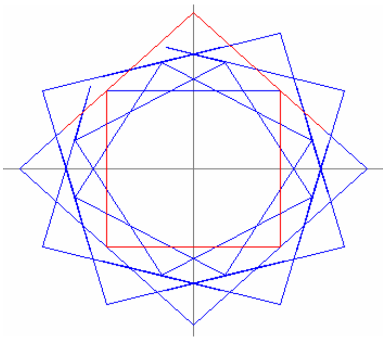
It is very nice to use the VECTOR command to make visible the effect of ongoing multiplying the object matrix h by any "random" matrix a or by the "determined" matrix r. Josef



matrix a



another random matrix



matrix r from above

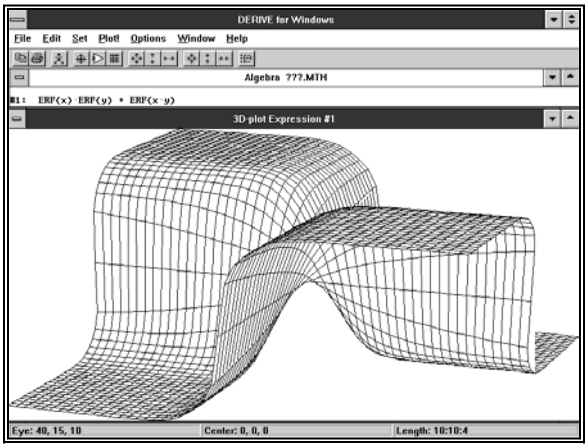
[1] You can find an exciting application of *dominant eigenvectors* in DNL#68, page 12

ERF functions in DfW4 (and DERIVE 6.10)

Another view from the

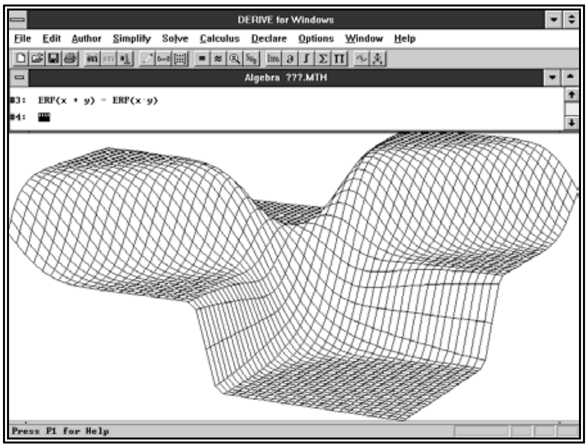
DERIVE Window (DfW 4.0)

Compare DfW4 and DERIVE 6 (below)!



#1: ERF(x)·ERF(y) + ERF(x·y)

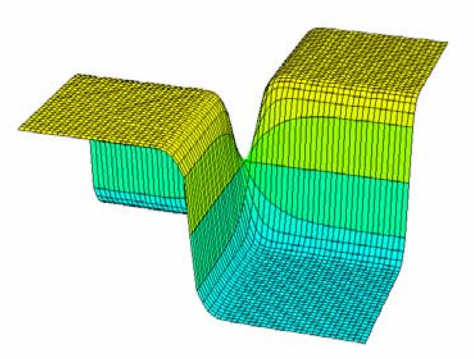
Eye: 40, 15, 10 Center: 0, 0, 0 Length: 10:10:4

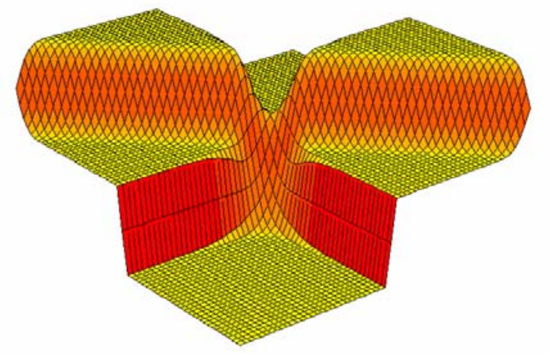

DERIVE for Windows
File Edit Author Simplify Solve Calculus Declare Options Window Help
Algebra 777.MTH

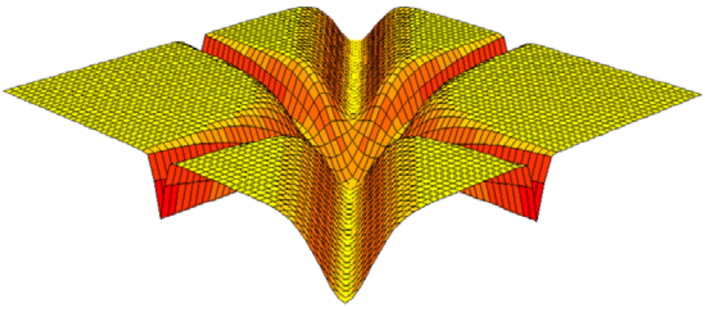
#3: ERF(x + y) - ERF(x·y)

#4:

Press F1 for Help







ERF(x+y)^2 + ERF(x·y)^2

Playing with *DERIVE´s* Plotting Accuracy

Josef Böhm and Charles Reno, Euclid, USA

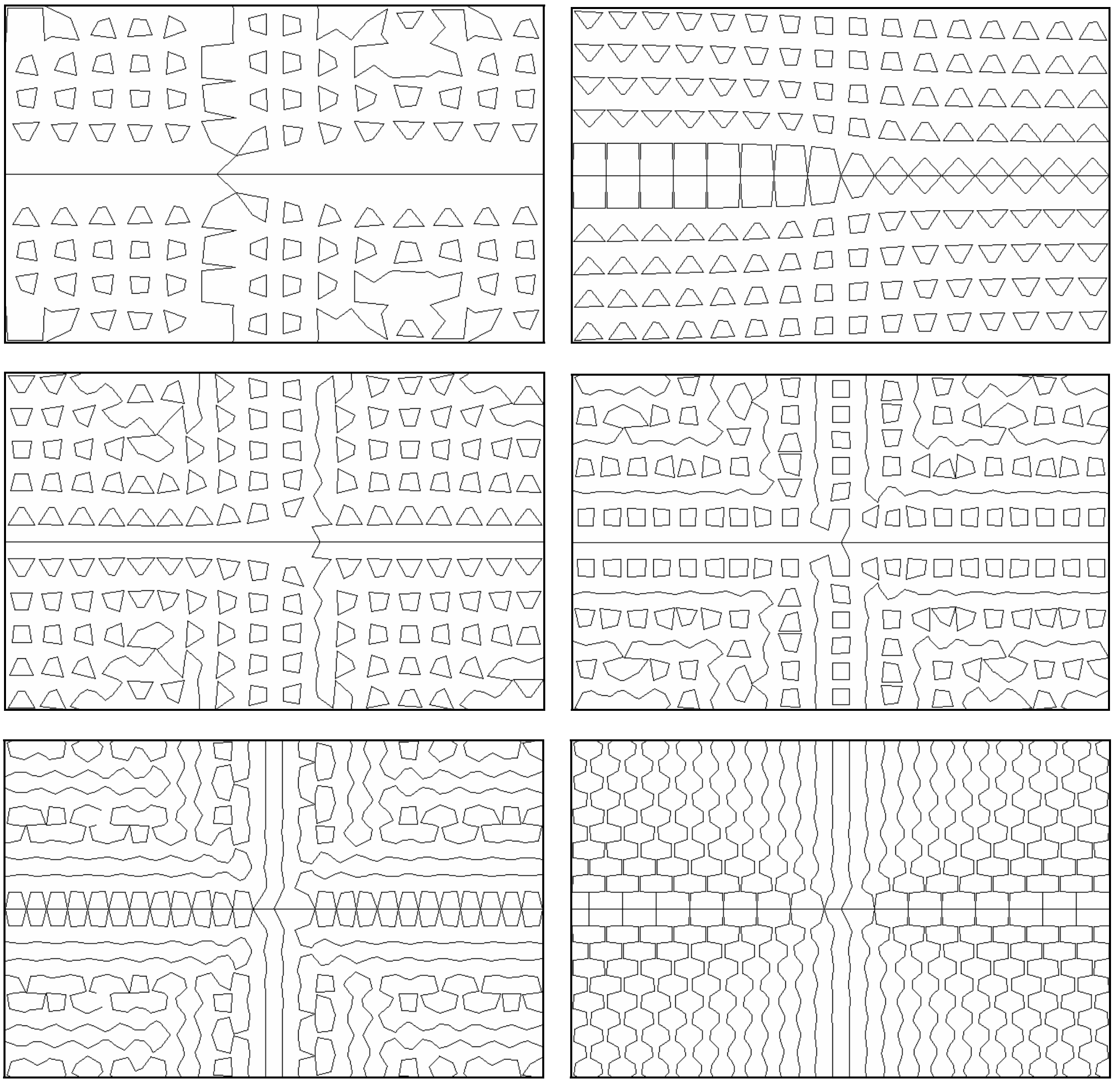
When I attended the T^3 Conference in Columbus, Ohio, I met Charles Reno from Euclid, Ohio and during a break he challenged *DERIVE* to plot the implicit function

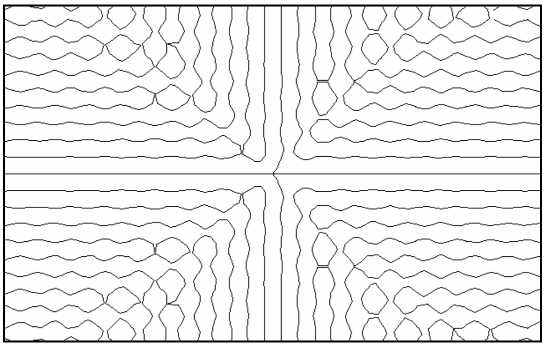
$$x\sin(2y) = y\cos(2x)$$

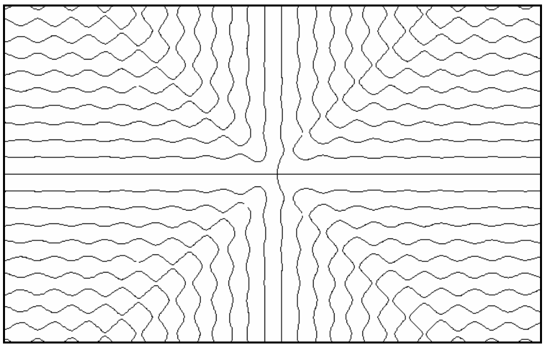
Immediately we tried - in the default Accuracy 7. As we were not really satisfied with the result we increased the accuracy up to 9 and then, because the picture needed too much time to be built up we again decreased the accuracy. So we met many different patterns dependent on the plotting accuracy. It was fascinating to observe *DERIVE´*s plotting algorithm applied on this example.

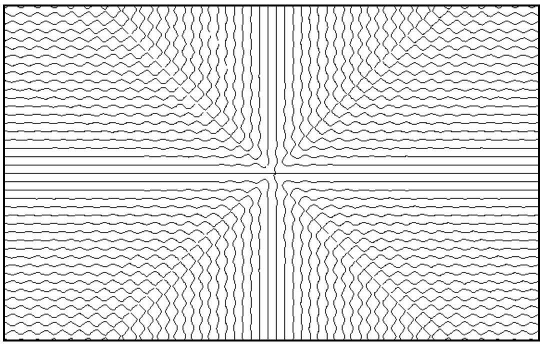
You can follow the sequence of accuracy in the next screen shots. But one warning: if you have the intention to reproduce the plots then you should preserve a lot of time.

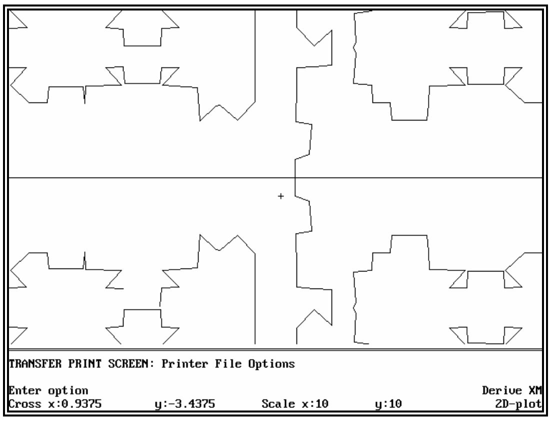
On the next page you can find two plots made by *DERIVE* for WINDOWS. The plot process in DfW is very much faster than in the DOS version - but it seems that you don’t have any possibility to change the accuracy.



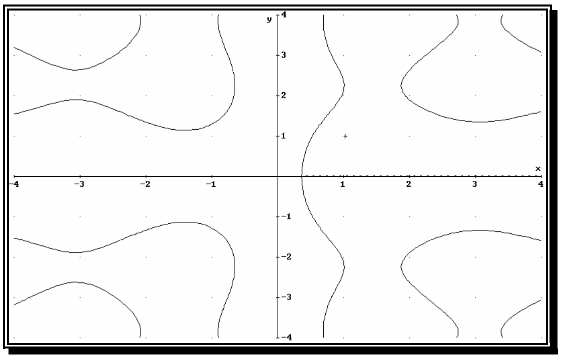


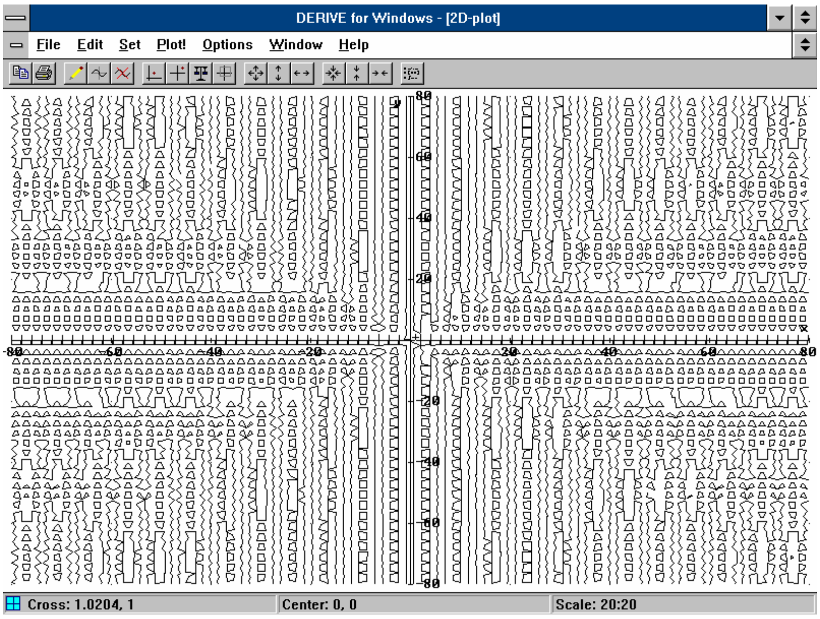






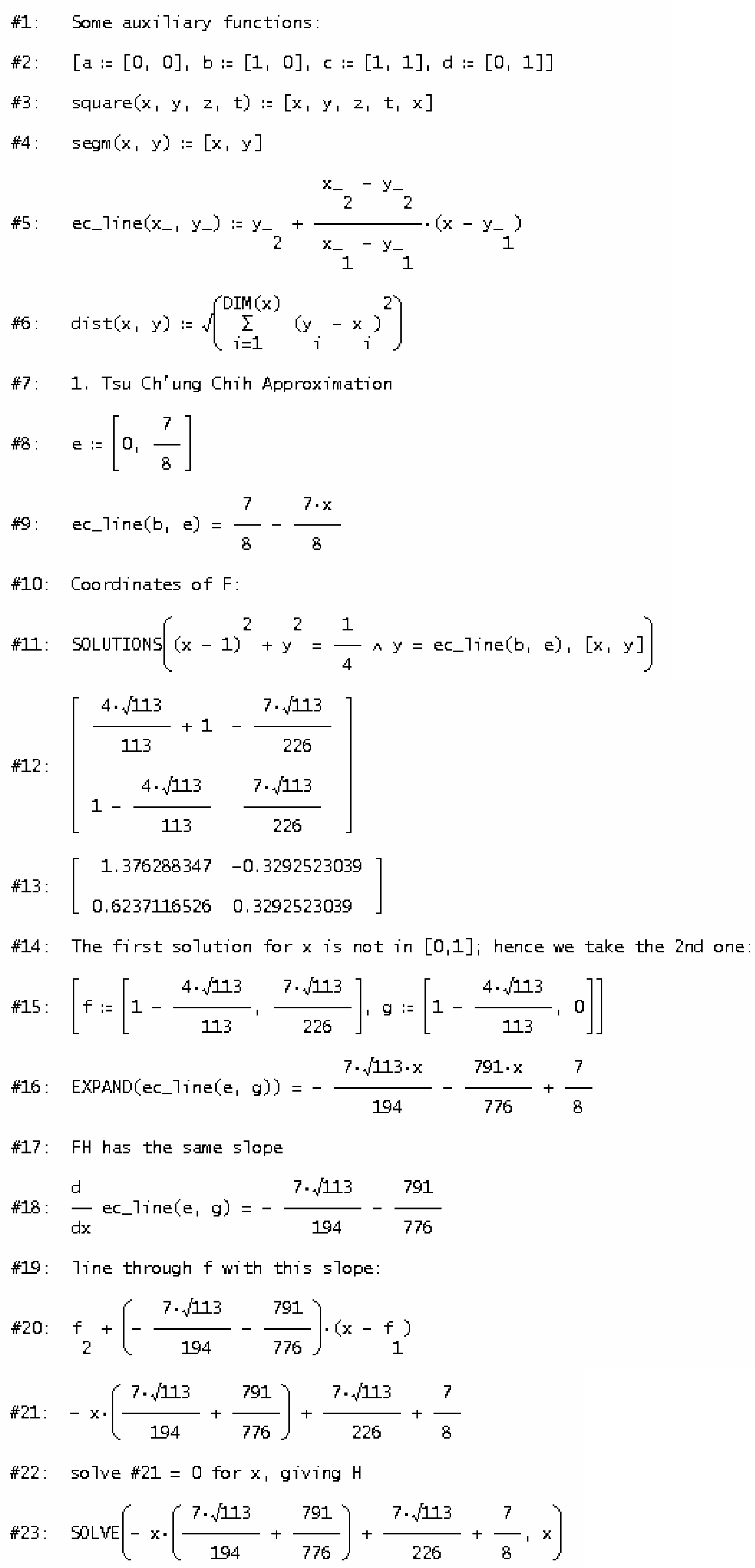
```
TRANSFER PRINT SCREEN: Printer File Options

Enter option                                              Derive XM
Cross x:0.9375      y:-3.4375      Scale x:10      y:10     2D-plot
```




DERIVE for Windows - [2D-plot]
File Edit Set Plot! Options Window Help
80
60
40
20
-20
-40
-60
-80
-80
-60
-40
-20
20
40
60
80

Cross: 1.0204, 1 | Center: 0, 0 | Scale: 20:20

Solution of the problems from ACDC One



#1: Some auxiliary functions:

#2: [a := [0, 0], b := [1, 0], c := [1, 1], d := [0, 1]]

#3: square(x, y, z, t) := [x, y, z, t, x]

#4: segm(x, y) := [x, y]

#5: $\mathrm{ec\_line}(x\_, y\_) := y\_{}_2 + \dfrac{x\_{}_2 - y\_{}_2}{x\_{}_1 - y\_{}_1}\cdot(x - y\_{}_1)$

#6: $\mathrm{dist}(x, y) := \sqrt{\left(\sum_{i=1}^{\mathrm{DIM}(x)} (y_i - x_i)^2\right)}$

#7: 1. Tsu Ch'ung Chih Approximation

#8: $e := \left[0, \dfrac{7}{8}\right]$

#9: $\mathrm{ec\_line}(b, e) = \dfrac{7}{8} - \dfrac{7\cdot x}{8}$

#10: Coordinates of F:

#11: $\mathrm{SOLUTIONS}\left((x - 1)^2 + y^2 = \dfrac{1}{4} \wedge y = \mathrm{ec\_line}(b, e), [x, y]\right)$

#12: $\begin{bmatrix} \dfrac{4\cdot\sqrt{113}}{113} + 1 & -\dfrac{7\cdot\sqrt{113}}{226} \\ 1 - \dfrac{4\cdot\sqrt{113}}{113} & \dfrac{7\cdot\sqrt{113}}{226} \end{bmatrix}$

#13: $\begin{bmatrix} 1.376288347 & -0.3292523039 \\ 0.6237116526 & 0.3292523039 \end{bmatrix}$

#14: The first solution for x is not in [0,1]; hence we take the 2nd one:

#15: $\left[f := \left[1 - \dfrac{4\cdot\sqrt{113}}{113}, \dfrac{7\cdot\sqrt{113}}{226}\right], g := \left[1 - \dfrac{4\cdot\sqrt{113}}{113}, 0\right]\right]$

#16: $\mathrm{EXPAND}(\mathrm{ec\_line}(e, g)) = -\dfrac{7\cdot\sqrt{113}\cdot x}{194} - \dfrac{791\cdot x}{776} + \dfrac{7}{8}$

#17: FH has the same slope

#18: $\dfrac{d}{dx}\,\mathrm{ec\_line}(e, g) = -\dfrac{7\cdot\sqrt{113}}{194} - \dfrac{791}{776}$

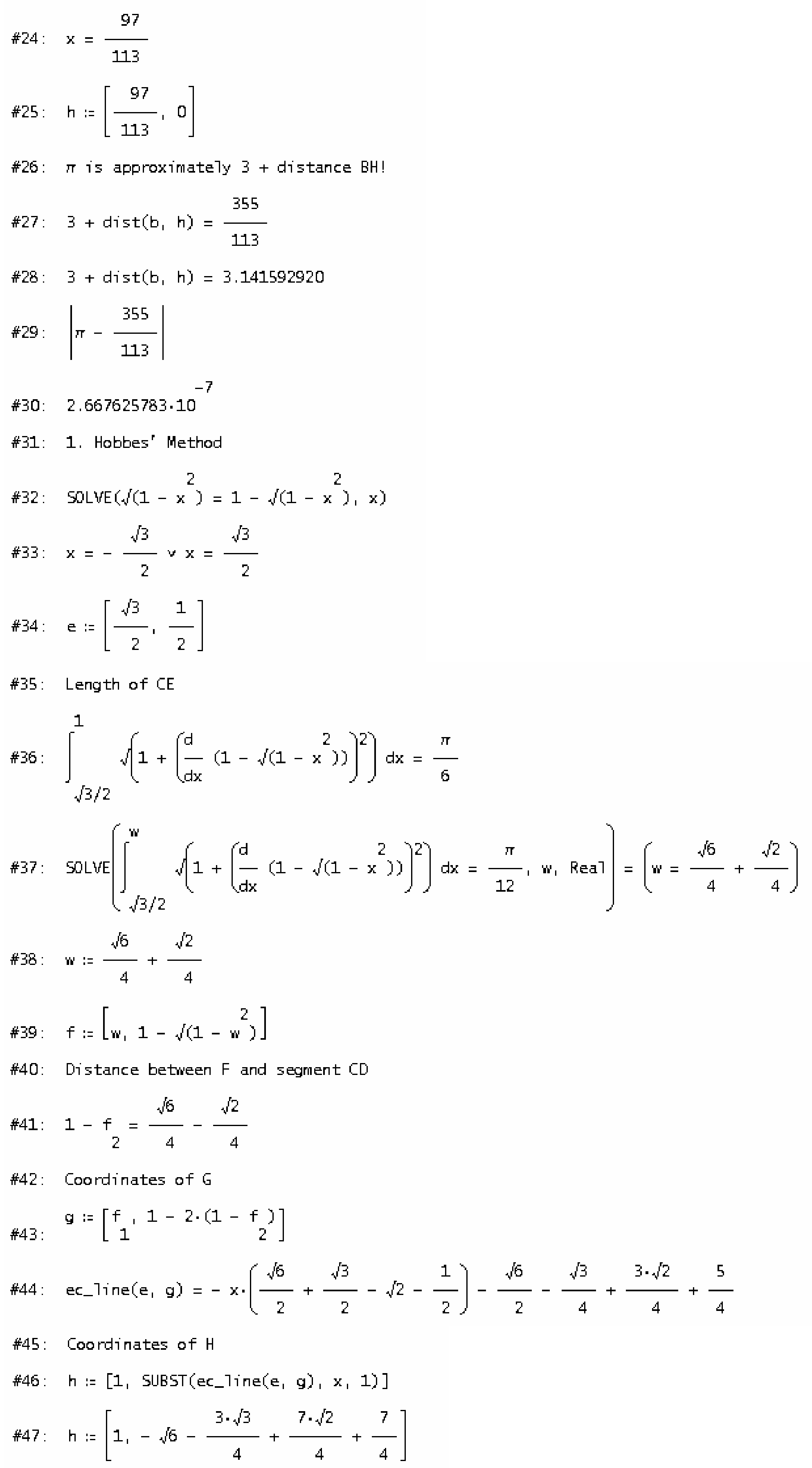
#19: line through f with this slope:

#20: $f_2 + \left(-\dfrac{7\cdot\sqrt{113}}{194} - \dfrac{791}{776}\right)\cdot(x - f_1)$

#21: $-x\cdot\left(\dfrac{7\cdot\sqrt{113}}{194} + \dfrac{791}{776}\right) + \dfrac{7\cdot\sqrt{113}}{226} + \dfrac{7}{8}$

#22: solve #21 = 0 for x, giving H

#23: $\mathrm{SOLVE}\left(-x\cdot\left(\dfrac{7\cdot\sqrt{113}}{194} + \dfrac{791}{776}\right) + \dfrac{7\cdot\sqrt{113}}{226} + \dfrac{7}{8}, x\right)$



#24: $x = \frac{97}{113}$

#25: $h := \left[\frac{97}{113}, 0\right]$

#26: π is approximately 3 + distance BH!

#27: $3 + \mathrm{dist}(b, h) = \frac{355}{113}$

#28: $3 + \mathrm{dist}(b, h) = 3.141592920$

#29: $\left|\pi - \frac{355}{113}\right|$

#30: $2.667625783\cdot 10^{-7}$

#31: 1. Hobbes' Method

#32: $\mathrm{SOLVE}(\sqrt{(1 - x^2)} = 1 - \sqrt{(1 - x^2)}, x)$

#33: $x = -\frac{\sqrt{3}}{2} \vee x = \frac{\sqrt{3}}{2}$

#34: $e := \left[\frac{\sqrt{3}}{2}, \frac{1}{2}\right]$

#35: Length of CE

#36: $\int_{\sqrt{3}/2}^{1} \sqrt{\left(1 + \left(\frac{d}{dx}(1 - \sqrt{(1 - x^2)})\right)^2\right)}\, dx = \frac{\pi}{6}$

#37: $\mathrm{SOLVE}\left(\int_{\sqrt{3}/2}^{w} \sqrt{\left(1 + \left(\frac{d}{dx}(1 - \sqrt{(1 - x^2)})\right)^2\right)}\, dx = \frac{\pi}{12}, w, \mathrm{Real}\right) = \left(w = \frac{\sqrt{6}}{4} + \frac{\sqrt{2}}{4}\right)$

#38: $w := \frac{\sqrt{6}}{4} + \frac{\sqrt{2}}{4}$

#39: $f := \left[w, 1 - \sqrt{(1 - w^2)}\right]$

#40: Distance between F and segment CD

#41: $1 - f_2 = \frac{\sqrt{6}}{4} - \frac{\sqrt{2}}{4}$

#42: Coordinates of G

#43: $g := \left[f_1, 1 - 2\cdot(1 - f_2)\right]$

#44: $\mathrm{ec\_line}(e, g) = - x\cdot\left(\frac{\sqrt{6}}{2} + \frac{\sqrt{3}}{2} - \sqrt{2} - \frac{1}{2}\right) - \frac{\sqrt{6}}{2} - \frac{\sqrt{3}}{4} + \frac{3\cdot\sqrt{2}}{4} + \frac{5}{4}$

#45: Coordinates of H

#46: $h := [1, \mathrm{SUBST}(\mathrm{ec\_line}(e, g), x, 1)]$

#47: $h := \left[1, - \sqrt{6} - \frac{3\cdot\sqrt{3}}{4} + \frac{7\cdot\sqrt{2}}{4} + \frac{7}{4}\right]$

#48: π is approximately 6·distance CH

#49: $$6\cdot\mathrm{dist}(h,\ c) = 6\cdot\sqrt{6} + \frac{9\cdot\sqrt{3}}{2} - \frac{21\cdot\sqrt{2}}{2} - \frac{9}{2}$$

#50: $$6\cdot\mathrm{dist}(h,\ c) = 3.141924685$$

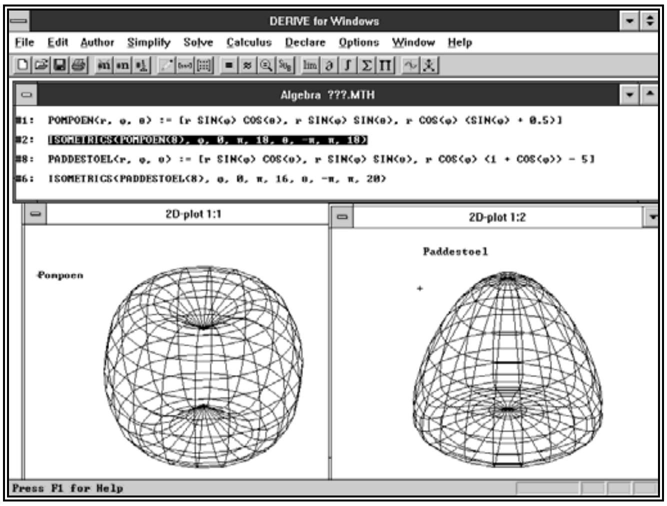
#51: $$|\pi - 6\cdot\mathrm{dist}(h,\ c)| = -\frac{2\cdot\pi - 12\cdot\sqrt{6} - 9\cdot\sqrt{3} + 21\cdot\sqrt{2} + 9}{2}$$

#52: $$|\pi - 6\cdot\mathrm{dist}(h,\ c)| = 0.0003320322787$$

Another DfW Screen Dump (Dfw 4)

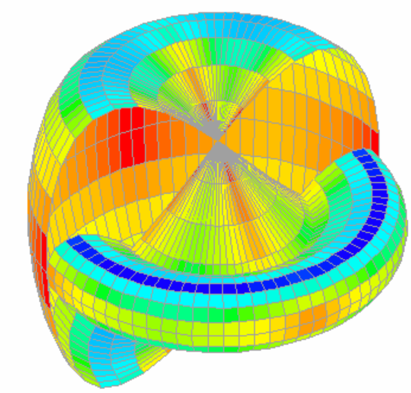
The graphics are - like the others in this and earlier *DSL's* - produced by students of Luk Van den Broeck, Belgium.

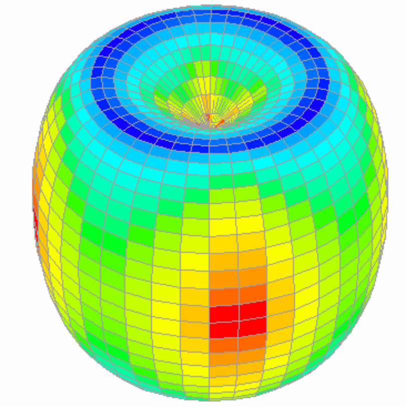
See below the pompoen (= pumpkin) and the paddestoel (= mushroom) in DERIVE 6.

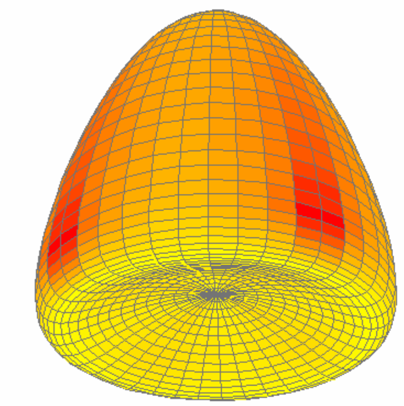


```
#1:  POMPOEN(r, φ, θ) := [r SIN(φ) COS(θ), r SIN(φ) SIN(θ), r COS(φ) (SIN(φ) + 0.5)]
#2:  ISOMETRICS(POMPOEN(8), φ, 0, π, 18, θ, -π, π, 18)
#8:  PADDESTOEL(r, φ, θ) := [r SIN(φ) COS(θ), r SIN(φ) SIN(θ), r COS(φ) (1 + COS(φ)) - 5]
#6:  ISOMETRICS(PADDESTOEL(8), φ, 0, π, 16, θ, -π, π, 20)
```

```
pumpkin(r, s, t) := [r·SIN(s)·COS(t), r·SIN(s)·SIN(t), r·COS(s)·(SIN(s) + 0.5)]

pumpkin(8)

mushroom(r, s, t) := [r·SIN(s)·COS(t), r·SIN(s)·SIN(t), r·COS(s)·(1 + COS(s)) - 5]

mushroom(8)
```







Titbits from Algebra and Number Theory

by Johann Wiesenbauer, Vienna

Hi folks, long time no see! Well, on second thoughts this greeting might be inappropriate in your case as I have just met quite a few of you at the second international DERIVE conference in Bonn this summer. Speaking of Bonn this was really a very special event! As for me it was here that I first got a lot of feedback as regards this column. Boy, modesty forbids me to say more than that I was really flattered! On the other hand I don't want to cover up the fact that I also met people who were in the bad habit of skipping my column regularly being of the opinion that it is strictly for the birds. You happen to be one of them and were just about to turn over? Phew! Please, read on! I promise to bend over backwards to please you this time!

Have you ever regretted the fact that DERIVE has no built-in function to solve even the simplest nonlinear systems of equations? If you agree then the following routine might come in handy:

```
SOLVE2(u,v,x,y):=ITERATE(IF(LHS(l_ SUB 1)=y, APPEND(DELETE_ELEMENT(l_,1),RHS(VECTOR(
[x_,l_ SUB 1], x_, LIM( SOLVE(v,x),y,RHS(l_ SUB 1))))),APPEND(DELETE_ELEMENT(l_,1), RHS(
VECTOR([l_ SUB 1,y_],y_,LIM(SOLVE(v,y),x,RHS(l_ SUB 1))))), l_), l_,ITERATE(SOLVE(g_,IF(
PRODUCT(v_-x,v_, VARIABLES(g_))=0,x,y,y)), g_, ITERATE(RHS(v)*QUOTIENT(u_,LHS(v))+
REMAINDER(u_, LHS(v)),u_,LHS(u)-RHS(u)),1))
```

Here u and v are two polynomial equations in x and y where v should be the "simpler" of the two equations. It is supposed to work whenever

```
RED(u,v):=ITERATE(RHS(v)*QUOTIENT(u_,LHS(v)) + REMAINDER(u_,LHS(v)),u_,LHS(u)-RHS(u))
```

(cf. DNL #20) yields a solvable expression in one variable only.

Using these functions example 1 from DNL #21, p.39, would now look like this:

```
RED(x^2-x*y-y=113,y^2-x*y+x=-41)=126*(y+13)/(y-1)^2-70
SOLVE2(x^2-x*y-y=113,y^2-x*y+x=-41)=[[15,7],[-61/5,-16/5]]
```

I used what I call the iterate(...,1)-trick to "combine RED and solve", as Josef put it in the article quoted above, and it might be of use in other cases as well:

```
SOLVERED(u,v):=ITERATE(SOLVE(u_,VARIABLES(u_)),u_,RED(u,v),1)
```

Actually, it was this preliminary version of SOLVE2 which I distributed to some people at the conference in Bonn. It is still useful in some cases like in my example 2 in the DNL #21, p.39:

```
P(v):=(a*(b-v)+r*t*v^2)/(v^2*(v-b))

SOLVERED(DIF(P(v),v),DIF(P(v),v,2))=[v=3*b,v=inf,v=-inf]
```

Here SOLVE2 won't work because of a very nasty bug in the current version 3.14e of DERIVE. Which one? Well, DERIVE commutes sometimes the operations DIF and LIM where this isn't allowed, e.g. in

```
LIM(DIF(xy,x),y,x)=2x
```

There are other cases where SOLVE2 should work in theory but doesn't in practice because of a memory overflow e.g.

```
SOLVE2(x^2-2*x*y-3*y=1,y^2-3*x*y+2*x=-4)
```

In these cases the modified routine

```
NSOLVE2(u,v,x,y):=ITERATE(IF(LHS(l_ SUB 1)=y,APPEND( DELETE_ELEMENT(l_,1),RHS(VECTOR(
APPROX([x_,l_ SUB 1]), x_, LIM(SOLVE(v,x),y,APPROX(RHS(l_ SUB 1)))))),APPEND(
DELETE_ELEMENT(l_,1),RHS(VECTOR(APPROX([l_ SUB 1,y_]), y_,LIM(SOLVE(v,y),x,APPROX(RHS(
l_ SUB 1)))))),l_),l_, ITERATE(SOLVE(g_,IF(PRODUCT(v_-x,v_,VARIABLES(g_))=0,x,y,y)), g_,
ITERATE(RHS(v)*QUOTIENT(u_,LHS(v))+REMAINDER(u_,LHS(v)),u_,LHS(u)-RHS(u)),1))
```

should give you at least approximate solutions (n stands for numeric!). Unlike the usual numeric procedures you will get all the solutions (even complex ones if any!):

```
NSOLVE2(x^2-2*x*y-3*y=1,y^2-3*x*y+2*x=-4)=
[[3.58397,1.1649],[-1.68707,-4.93439],[-1.04845-0.353058*#i,-0.415254+
0.495038*#i],[-1.04845+0.353058*#i,-0.415254-0.495038*#i]]
```

(*440 sec. on a 486DX2/66; 575 sec. on a 486/33; 110 sec. on a Pentium*)

You might have noticed that I have always omitted the third and fourth parameter so far. In fact, this is allowed if your variables are x and y which is recommended. If you use other variables than x and y you should make sure that they come before any constants in the order of variables (if necessary use the command "Manage Order") e.g.

```
SOLVE2(a^2+b^2=1,b-r/s*a=-1,a,b) =
[[0,-1],[2*r*s/(r^2+s^2),(r^2-s^2)/(r^2+s^2)]]
```

It is not easy to give general rules when SOLVE2(u,v,x,y) is applicable and therefore it is up to you to experiment a little. Even so, the following two cases deserve to be mentioned:

- SOLVE2(p(x,y),ax+by=c), where p(x,y) is a polynomial of degree at most 4 and a,b,c are real numbers. (In the case a=0 reverse the order of x,y by means of "Manage Order"!) In particular you could use SOLVE2(p(x,y),x=c) or SOLVE2(p(x,y),y=c,y,x), respectively, to calculate points on the curve which is implicitly given by p(x,y)=0.
- When dealing with two conic sections try to arrive at SOLVE2(ax^2+bxy+cy^2+dx+ey+f,gy^2+hx+iy+j) with h≠0 or alternatively at SOLVE2(ax^2+bxy+cy^2+dx+ey+f,gx^2+hx+iy+j,y,x) with i≠0 (again reverse the order of x,y in the latter case!) In "school examples" this special form of v can be virtually always achieved by forming suitable linear expressions of the original equations u and v. Since SOLVE2 might take quite long in these cases (or even lead to a memory overflow) you should always try SOLVERED or NSOLVE2 first!

Let me use the remaining space to deal with another "most missing" feature of DERIVE which is sorting a given list of real numbers. Those lists occur in mathematics on many occasions. One example that immediately comes to my mind are the Farey numbers of order n for a given natural number n, i.e. the positive fractions not greater than 1 with a denominator at most n ordered by their size. Assuming that a routine SORT(v) is already available the definition of FAREY(n) could look like this:

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

As for SORT we could take in this case

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

Actually this is nothing else but the sort-routine from my "Titbits(7)" which I believed only sorted different nonnegative integers but which can be taken for different rational numbers as well! Nevertheless I will keep my promise and give you the definition of a sort-routine that sorts arbitrary real numbers:

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

(It goes without saying that in both cases the corresponding definition for DSORT(v), which sorts in descending order, is the same but without applying REVERSE_VECTOR.) It works as the following example shows:

```
SORT([pi,SQRT(3),SQRT(2),#e,SQRT(5),SQRT(2),#e])=
    [SQRT(2),SQRT(2),SQRT(3),SQRT(5),#e,#e,pi]
```

I hear you saying that there has already been published such a function in the DNL #22,p.40, by Sergey Biryukov. Ah, I would have liked to dodge that issue because it forces me to admit that I felt totally at a loss when I studied Sergey's solution. But remembering a story dealing with Alexander the Great when he was in a quite similar situation I cut Sergey's implementation into two halves throwing away one of them and making some changes to the other. Here is the outcome of my doing:

```
SSORT_AUX(v,n,p):=VECTOR(IF(MOD(i+p,2)=0,IF(i=1 OR v SUB (i-1)<=
v SUB i,v SUBi,v SUB (i-1)),IF(i=n OR v SUB i<=v SUB (i+1), v SUB i,v SUB (i+1))),i,n)
```

```
SSORT(v):=ITERATE(SSORT_AUX(SSORT_AUX(w,DIMENSION(v),0),DIMENSION(v),1),w,v)
```

Just one small example to give you an impression of the calculation times: For n=50 there are 774 Farey fractions to sort. It took 7 sec to calculate them and 1.1 to sort them with the first sort-routine, 1.5 sec with the second one and 901.4 sec by means of SSORT(v).

I should also note that I am currently using a Pentium PC with 166 MHz and 32 MB Ram. It's really a powerful machine! I used it to calculate the number

parts(1840926)=245...373 (1505 digits!)

in 12 h 43 min which has been proven to be prime by Morain quite recently . It took DERIVE another 394.3 sec to confirm this. See you next time!(?)

On the diskette Johann gave the solution of the example from DNL#21 using his SOLVE2 routine:

$$\#19:\ \mathrm{SOLVE2}(a^2 + b^2 + c^2 = 9,\ a + b + c = 3,\ a,\ b) = \begin{bmatrix} -\dfrac{\sqrt{3}\cdot\sqrt{(c - 3)}\cdot\sqrt{(-c - 1)} + c - 3}{2} & \dfrac{\sqrt{(c - 3)}\cdot(\sqrt{3}\cdot\sqrt{(-c - 1)} - \sqrt{(c - 3)})}{2} \\ \dfrac{\sqrt{3}\cdot\sqrt{(c - 3)}\cdot\sqrt{(-c - 1)} - c + 3}{2} & -\dfrac{\sqrt{3}\cdot\sqrt{(c - 3)}\cdot\sqrt{(-c - 1)} + c - 3}{2} \end{bmatrix}$$

$$\#20:\ a := -\frac{\sqrt{3}\cdot\sqrt{(c - 3)}\cdot\sqrt{(-c - 1)} + c - 3}{2} \qquad \text{User}$$

$$\#21:\ b := \frac{\sqrt{(c - 3)}\cdot(\sqrt{3}\cdot\sqrt{(-c - 1)} - \sqrt{(c - 3)})}{2} \qquad \text{User}$$

$$\#22:\ a^3 + b^3 + c^3 - 24 = 3\cdot(c^3 - 3\cdot c^2 + 1) \qquad \text{User=Simp(User)}$$

$$\#23:\ \mathrm{SOLVE}(a^3 + b^3 + c^3 = 24,\ c) = \left[c = 1 - 2\cdot\mathrm{SIN}\left[\frac{\pi}{18}\right],\ c = 2\cdot\mathrm{COS}\left[\frac{\pi}{9}\right] + 1,\ c = 1 - 2\cdot\mathrm{COS}\left[\frac{2\cdot\pi}{9}\right]\right]$$

#24: "The following equation together with a^3+2b^3=24 shows that" User

#25: "no two variables, like e.g. b and c, may have the same value:" User

$$\#26:\ \mathrm{SOLVE2}(a^2 + 2\cdot b^2 = 9,\ a + 2\cdot b = 3,\ a,\ b) = \begin{bmatrix} 3 & 0 \\ -1 & 2 \end{bmatrix} \qquad \text{User=Simp(User)}$$

#27: "Therefore apart from permutations this is the unique solution:" User

$$\#28:\ \left[a := 1 - 2\cdot\mathrm{SIN}\left[\frac{\pi}{18}\right],\ b := 2\cdot\mathrm{COS}\left[\frac{\pi}{9}\right] + 1,\ c := 1 - 2\cdot\mathrm{COS}\left[\frac{2\cdot\pi}{9}\right]\right] \text{User}$$

#29: "Let's check all the terms in question:" User

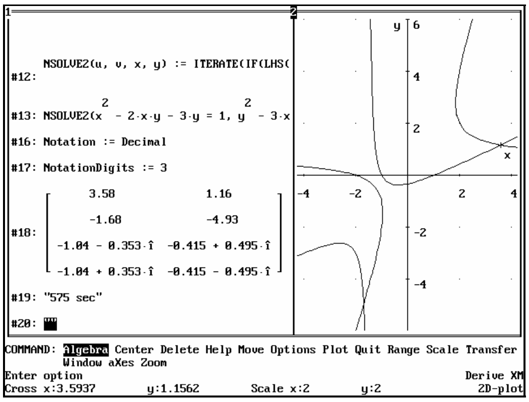
#30: Trigonometry := Collect User

$$\#31:\ \left[a + b + c,\ a^2 + b^2 + c^2,\ a^3 + b^3 + c^3,\ a^4 + b^4 + c^4\right] \qquad \text{User}$$

$$\#32:\ \left[3,\ 4\cdot\mathrm{COS}\left[\frac{\pi}{9}\right] - 2\cdot\sqrt{3}\cdot\mathrm{COS}\left[\frac{\pi}{18}\right] - 2\cdot\mathrm{SIN}\left[\frac{\pi}{18}\right] + 9,\ 24,\ 69\right] \qquad \text{Simp(\#31)}$$

#33: "DERIVE can't cope with a^2+b^2+c^2 but by approximating we arrive at"

#34: [3, 9, 24, 69] Approx(#31)



$$\#12:\ \mathrm{NSOLVE2}(u,\ v,\ x,\ y) := \mathrm{ITERATE}(\mathrm{IF}(\mathrm{LHS}($$

$$\#13:\ \mathrm{NSOLVE2}(x^2 - 2\cdot x\cdot y - 3\cdot y = 1,\ y^2 - 3\cdot x$$

#16: Notation := Decimal

#17: NotationDigits := 3

$$\#18:\ \begin{bmatrix} 3.58 & 1.16 \\ -1.68 & -4.93 \\ -1.04 - 0.353\cdot \hat{\imath} & -0.415 + 0.495\cdot \hat{\imath} \\ -1.04 + 0.353\cdot \hat{\imath} & -0.415 - 0.495\cdot \hat{\imath} \end{bmatrix}$$

#19: "575 sec"

#20:

1
2
y
6
4
2
x
-4
-2
2
4
-2
-4
COMMAND: Algebra Center Delete Help Move Options Plot Quit Range Scale Transfer
Window aXes Zoom
Enter option
Derive XM
Cross x:3.5937 y:1.1562 Scale x:2 y:2
2D-plot

THE TI-92 CORNER

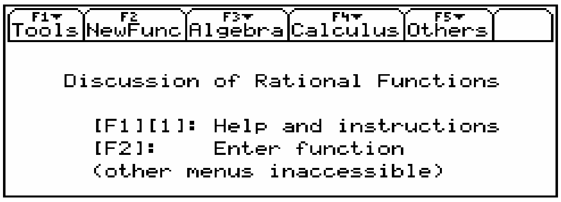
(EDITED BY B. WAITS, F. DEMANA, B. KUTZLER & J. BÖHM)

DIRA - A TI-92 PROGRAM FOR DISCUSSING RATIONAL FUNCTIONS

BY W. PRÖPPER, NÜRNBERG, GERMANY

When using the TI-92 more often to do the necessary computations for discussing functions one soon sees what a boring activity this is. The same operations have to be done again and again:
Enter the term of the function, store it as f(x), let the derivatives be computed and store them with other names, compute zeros of the function and its derivatives and so on. At the end you can get a nice plot.
What was more obvious than using the programmability of the TI-92 to pass all the routine works to the calculator. I was inspired to do so by a little contribution by J. Böhm in DNL #15 from Sept. 1994. There a Derive-utility was introduced, in which only the function's term had to be entered and Derive did the whole discussion.
This contribution shows in a comprised form the possibilities of DIRA when discussing rational functions.

After inputting dira() (ENTER) on the home screen of the TI-92 you see this opening screen:

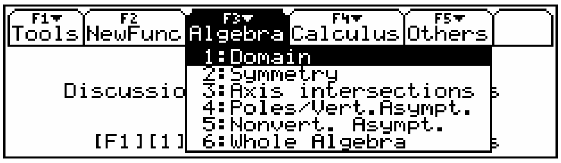


Discussion of Rational Functions

[F1][1]: Help and instructions
[F2]: Enter function
(other menus inaccessible)

fig. 1

Looking into the menus an expert foresees which mathematical activities take place in each item.
Algebraic investigations with (F3):

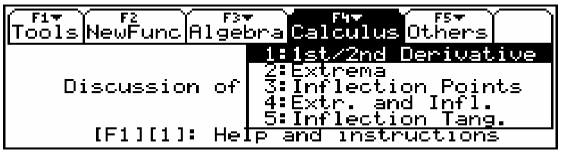


Discussio

[F1][1]

fig. 2

Investigations by means of calculus with (F4):

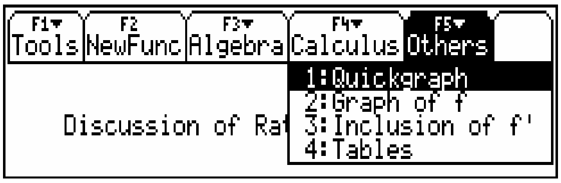


Discussion of

[F1][1]: Help and instructions

fig. 3

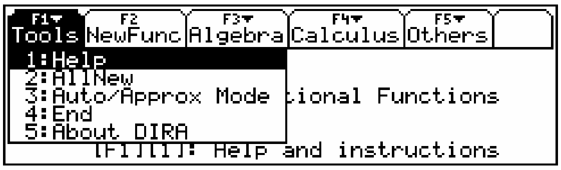
Graphs and tables with (F5):



Discussion of Ra

fig. 4

Tools are found in (F1):

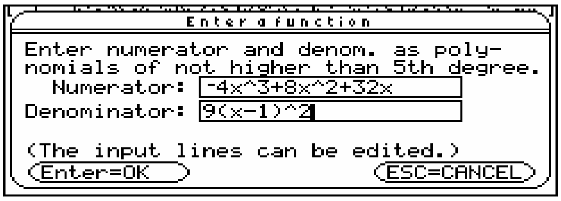


ional Functions

[F1][1]: Help and instructions

fig. 5

To be able working with DIRA a function's term must be entered first via (F2):

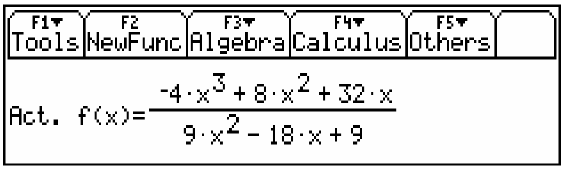


Enter numerator and denom. as polynomials of not higher than 5th degree.
Numerator: -4x^3+8x^2+32x
Denominator: 9(x-1)^2

(The input lines can be edited.)

fig. 6

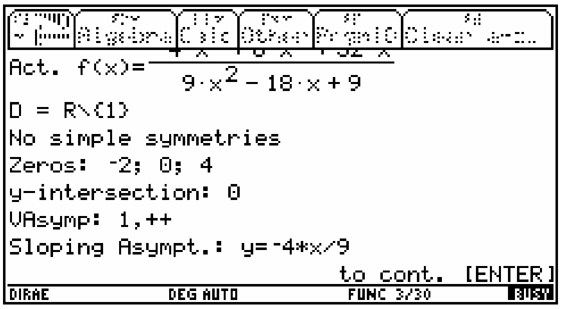
The DIRA-screen shows the actual function:



Act. $f(x)=\dfrac{-4\cdot x^3+8\cdot x^2+32\cdot x}{9\cdot x^2-18\cdot x+9}$

fig. 7

By (F3)(6) all algebra-items are run through:



Act. f(x)= $\dfrac{}{9\cdot x^2-18\cdot x+9}$

D = R\{1}

No simple symmetries

Zeros: ⁻2; 0; 4

y-intersection: 0

VAsymp: 1,++

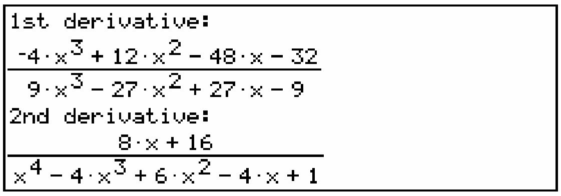
Sloping Asympt.: y=⁻4*x/9

to cont. [ENTER]

DIRAE DEG AUTO FUNC 3/30 BUSY

fig. 8

After that (F4)(1) computes the derivatives:



1st derivative:

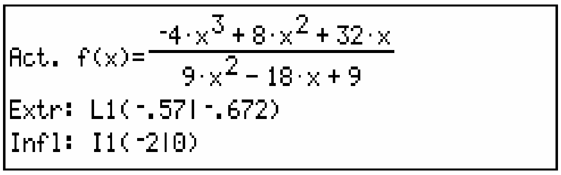
$$\frac{^-4\cdot x^3+12\cdot x^2-48\cdot x-32}{9\cdot x^3-27\cdot x^2+27\cdot x-9}$$

2nd derivative:

$$\frac{8\cdot x+16}{x^4-4\cdot x^3+6\cdot x^2-4\cdot x+1}$$

fig. 9

Extrema and inflection points can be found in one run by calling (F4)(4):



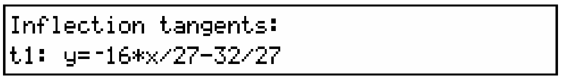
Act. f(x)= $\dfrac{^-4\cdot x^3+8\cdot x^2+32\cdot x}{9\cdot x^2-18\cdot x+9}$

Extr: L1(⁻.571⁻.672)

Infl: I1(⁻2|0)

fig. 10

The tangent at inflection points are found with (F4)(5):



Inflection tangents:

t1: y=⁻16*x/27-32/27

fig. 11

Finally the graphs of the function and its first derivative (dotted): (F5)(2) and (3):

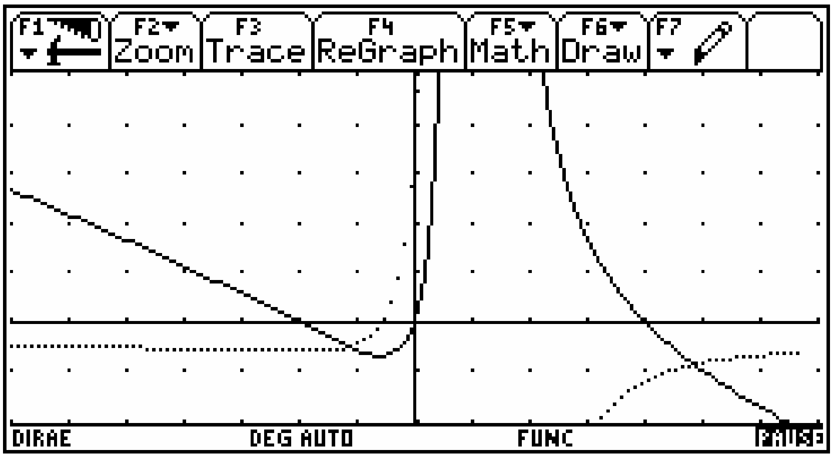


fig. 12

The program with a more detailed documentation will be found on the annual DNL-diskette. By means of the TI-Graph Link it can be transferred from the PC to the TI-92. From autumn 1996 you can find it in its most recent version on the home page of my school with the web address http://www.nuernberg.de/ver/schg/nk/.

Many thanks Wolfgang for your wonderful "Christmas gift". Maybe I'll find a way to put your files in a read- and printable form on the diskette, too. Then our friends who have not GRAPH LINK available can also benefit from your program. See the first lines of DIRA´s HELP subroutine:

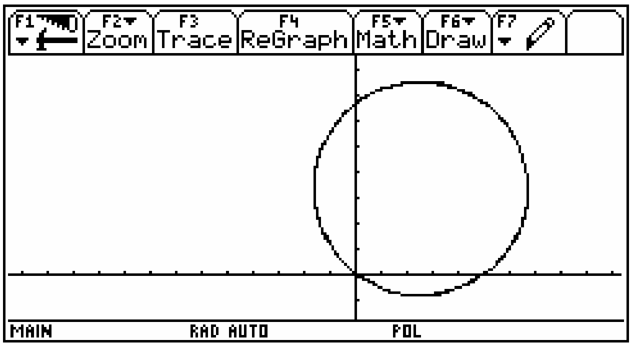
```
()
Prgm
ClrIO
Disp "         H e l p s c r e e n  (1)"
Disp ""
Disp "DIRA is a program which does the routine-"
Disp "work when DIscussing RAtional functions."
Disp ""
Disp "In menu [F3] algebraical methods like"
Disp "computing domains, zeros, poles and"
Disp "asymptotes are applied."
```

Determining Equations of Conic Sections

by M. Brubaker & C. Leinbach, USA

Introduction:

Given the following figure, find the equation of the circle. Where is its center located? What is its radius?



The strategy is obvious. Use the cursor to digitize some points and use these points to determine the equation of the circle. How many points do we need to digitize? The answer becomes clear if we look at the standard equation for a circle.

$$(x-h)^2 + (x-k)^2 = r^2$$

There are three unknowns, the x-coordinate of the center, the y-coordinate of the center, and the radius. We can find the equation by obtaining three points that lie on the circle, substituting, and solving the resulting three equations for the unknowns, h, k, and r. This process is straight forward, but, because of the fact that the unknowns appear to the first and second power in the resulting equations, tedious. We propose a different strategy that can be easily extended to find the equations of other conics in standard form.

Let's suppose that *(x,y)* is a point on the circle, then

$$(x-h)^2 + (x-k)^2 = r^2$$

which we can expand to

$$x^2 + y^2 - 2hx - 2ky + (k^2 + h^2 - r^2) = 0.$$

This has the general form of the homogeneous equation

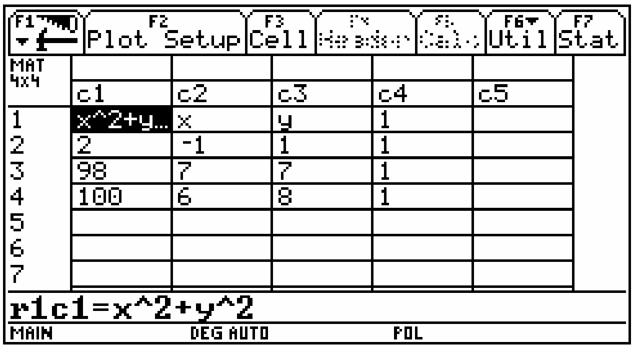
$$A(x^2 + y^2) + Bx + Cy + D = 0. \qquad (1)$$

Because we are dealing with the equation of a circle, x^2 and y^2 terms have the same coefficients. Thus, when we substitute for x and y we have a linear system with unknowns *A, B, C* and *D.* Have we increased the number of points needed to find the equation? The answer is no. The symbolic capabilities of our CAS will show that this the case.

Let's get specific. The figure depicted at the beginning of our discussion was chosen so that we could easily identify that the points (–1,1), (6,8) and (7,7) lie on the circle. Substituting these points into equation (1) above we have:

$$
\begin{aligned}
2A - B + C + D &= 0\\
100A + 6B + 8C + D &= 0 \qquad (2)\\
49A + 7B + 7C + D &= 0
\end{aligned}
$$

This still leaves us short one equation, or does it? We still have equation (1) and our CAS. If we think of *(x,y)* as a general point on the circle, we have a system of four homogeneous equations in four unknowns. From the theory of equations we know that this system will have a nontrivial solution if and only if the determinant of the coefficient matrix is zero. We enter this matrix using the Matrix editor of the TI-92.

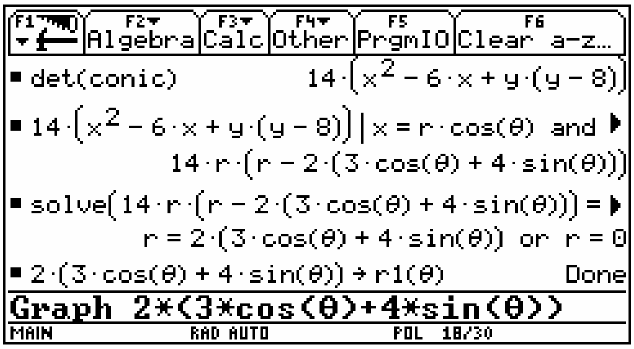


	c1	c2	c3	c4	c5
1	x^2+y…	x	y	1	
2	2	-1	1	1	
3	98	7	7	1	
4	100	6	8	1	
5					
6					
7					

```
r1c1=x^2+y^2
MAIN          DEG AUTO          FUL
```

In the following figure we take the determinant of this matrix (which we called conic). Setting this result equal to 0 yields the equation of the circle. Completing the square will reveal center and radius.

To graph the circle, it is best to use polar equations. Once again the CAS makes this easy. We use local substitution with $x = r.\cos(\theta)$ and $y = r.\sin(\theta)$ and solve the resulting expression for r ($\neq 0$). This generates the graph at the start of this article.



```
■ det(conic)              14·(x² - 6·x + y·(y - 8))
■ 14·(x² - 6·x + y·(y - 8)) | x = r·cos(θ) and ▶
             14·r·(r - 2·(3·cos(θ) + 4·sin(θ)))
■ solve(14·r·(r - 2·(3·cos(θ) + 4·sin(θ))) = ▶
             r = 2·(3·cos(θ) + 4·sin(θ)) or r = 0
■ 2·(3·cos(θ) + 4·sin(θ)) → r1(θ)             Done
Graph 2*(3*cos(θ)+4*sin(θ))
MAIN          RAD AUTO          FUL 18/30
```

This same idea can be extended to the other conics. The general form for a parabola is

$$Ax^2 + Bx + Cy + D = 0 \text{ or } Ay^2 + By + Cx + D = 0$$

and for ellipse or hyperbola (depending on the location of the points) is

$$Ax^2 + By^2 + Cx + Dy + E = 0.$$

Locating three points on the graph of a parabola or four on the graph of an ellipse or hyperbola and entering the appropriate matrix will quickly generate the equation of the conic in question.

Locating three points on the graph of a parabola or four on the graph of an ellipse or hyperbola and entering the appropriate matrix will quickly generate the equation of the conic in question.