THE DERIVE - NEWSLETTER #112

ISSN 1990-7079

THE BULLETIN OF THE

DERIVE

USER GROUP

+ CAS-TI

Contents:

December 2018

For our German speaking and understanding members:

Ich habe eine schöne Webseite eines Mathematikers in der Schweiz gefunden:

http://www.armin-p-barth.ch/index.php

Es finden sich zahlreiche Artikel und Beiträge zur Mathematik und Didaktik, sowie aus den von Armin Barth betreuten Kolumnen *Café Mathematik* und *Schwamm drüber*.
Armin Barth schreibt auch Kriminalgeschichten.

Warum ich das erwähne? Es gibt auch ein DUG-Mitglied, das eifrig Kriminalromane schreibt. Thomas Himmelbauer hat schon sechs Romane veröffentlicht. Sie spielen mehrheitlich in seiner Heimat, im schönen Südburgenland. Sein jüngstes Werk ist erst kürzlich erschienen: „Tod im Zickenwald“.

http://www.federfrei.at

Mail from Danny Ross Lunsford

Hi Professor Böhm,

Very happy to see the Derive UG still going strong. I think I will have some content soon, regarding use of Clifford algebras in particular and matrix algebras in general in Derive. One particular thing I may start with is the singular value decomposition of an arbitrary non-singular 2x2 complex matrix. This has a direct application in relativity and the math is fascinating and well-handled by Derive. It also is a good lesson in Derive programming and parameter setting.

I would like to implement a Derive workspace for recursively creating matrix algebras for all the possible Clifford algebras. I suppose that is what I am working toward.

I also have been doing a lot of comparison of the capabilities of Derive with those of a well-known APL, and have (sort of) implemented a shape and reshape function for arbitrary arrays, and am working on the idea of nested arrays in Derive. However, these ideas are at present half-baked. A lot more understanding of Derive's outer product has to take place. It is a shame that development on Derive has stopped!

Thanks for all your efforts! PS I attached two files to give a flavor of some of the above work.

VectorMatrixFunctions is an expansion of the file included with Derive with my additions annotated by DRL. The most interesting part of this for you will probably be the SHAPE and PERM functions, which respectively create arrays of arbitrary shape, and an array of all possible permutations. Dirac is my Clifford/matrix algebra workspace. They are not for direct publication at the moment, it's a work in progress.

-drl

"I write a little. I erase a lot." - Chopin

BTW, I am and was not a professor, I was a school teacher on a Secondary College for Business Administration, Josef. VectorMatrixFunctions will be presented in the next DNL.

Dear DUG Members,

Welcome to DNL#112 – late, but hopefully not too late. The main reason for the delay is that I wanted to wait for some useful feedback on articles in this issue. Another reason is our trip to Morocco from end of December to begin of January.

Most of the contributions in this DNL are satisfactions of long-time debts – articles which have waited very long on my pile of papers to be published.

Roland Schröder's "Regula Falsi" is the last paper in his series of articles which appeared in many DNLs. There was another paper which was similar to Pierre Charland's "Change Count". So, we agreed to skip Roland's treatment. In an earlier DNL you can find this problem in one of Johann Wiesenbauer's columns.

Josef Lechner's article was intended as a TI-Nspire-material. We added the DERIVE implementation. Josef refers to an article of Prof. Rump. I must admit that there are some details in Rump's one-page paper which I cannot follow.

I came across a strange problem with DERIVE's DISPLAY-function. Günter Schödl gave very useful advice. Maybe that one of you made similar experiences.

The German version of the Scientific American (Spektrum der Wissenschaft) presents a monthly column "Freistetters Formelwelt" (Freistetter's World of Formulae). In the February issue I found a short story about Strogatz' "Love affairs and differential equations" which in a mathematical way deals with the most famous lovers in literature, Romeo and Julia. I "googled" this header and was more than surprised to find a lot of downloadable pdf-files which are engaged in this Strogatz' one-page article (first link below).

I can imagine that this matter might be interesting for students and teachers as well – especially for Math-German- or Math-English-teachers …

Find below a selection of recommended links:

http://ai.stanford.edu/~rajatr/articles/SS_love_dEq.pdf

http://educ.jmu.edu/~strawbem/Sample_Report.pdf

http://sections.maa.org/okar/papers/2010/israel.pdf

http://sprott.physics.wisc.edu/pubs/paper277.pdf

https://www.lagrange.edu/resources/pdf/citations/2016/27_Jones_Math.pdf

http://modelling.martinsewell.com/Alle99.pdf

Best regards and wishes

Josef

The *DERIVE-NEWSLETTER* is the Bulletin of the *DERIVE* & CAS-*TI User Group*. It is published at least four times a year with a content of 40 pages minimum. The goals of the *DNL* are to enable the exchange of experiences made with *DERIVE*, *TI*-CAS and other CAS as well to create a group to discuss the possibilities of new methodical and didactical manners in teaching mathematics.

Editor: Mag. Josef Böhm
D´Lust 1, A-3042 Würmla, Austria
Phone: ++43-(0)660 31 36 365
e-mail: nojo.boehm@pgv.at

Contributions:
Please send all contributions to the Editor. Non-English speakers are encouraged to write their contributions in English to reinforce the international touch of the *DNL*. It must be said, though, that non-English articles will be warmly welcomed nonetheless. Your contributions will be edited but not assessed. By submitting articles, the author gives his consent for reprinting it in the *DNL*. The more contributions you will send, the more lively and richer in contents the *DERIVE* & CAS-*TI Newsletter* will be.

Next issue: March 2019

Preview: Contributions waiting to be published

Some simulations of Random Experiments, J. Böhm, AUT, Lorenz Kopp, GER
Wonderful World of Pedal Curves, J. Böhm, AUT
Simulating a Graphing Calculator in *DERIVE*, J. Böhm, AUT
Cubics, Quartics – Interesting features, T. Koller & J. Böhm, AUT
Logos of Companies as an Inspiration for Math Teaching
Exciting Surfaces in the FAZ
BooleanPlots.mth, P. Schofield, UK
Old traditional examples for a CAS – What´s new? J. Böhm, AUT
Mandelbrot and Newton with *DERIVE*, Roman Hašek, CZK
Tutorials for the NSpireCAS, G. Herweyers, BEL
Dirac Algebra, Clifford Algebra, D. R. Lunsford, USA
A New Approach to Taylor Series, D. Oertel, GER
Statistics of Shuffling Cards, Charge in a Magnetic Field, H. Ludwig, GER
Factoring Trinomials, D. McDougall, CAN
Selected Lectures from TIME 2016
More Applications of TI-Innovator^{TM} Hub and TI-Innovator^{TM} Rover
Surfaces and their Duals, Cayley Symmetroid, J. Böhm, AUT
Affine Mappings –Treated Systematically, H. Nieder, GER
Three planes in space – and how they can intersect, J. Böhm, AUT
Goodies for TI-NspireCAS (3), Don Phillips, USA
and others

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

Francisco Marcelo Fernández, Argentina fernande@quimica.unlp.edu.ar

Dear Josef,
I have unsuccessfully tried to factorize a fourth-order polynomial using the simplify/factor command. However, the polynomial can be factorized as shown in the attached file.
Best regards, Marcelo

```
The polynomial

                  2                                8
        4    3    x ·(62337182076360565082813671875·π  - 26958302109890393212008267776)
#1:     x  - x  + ──────────────────────────────────────────────────────────────────── +
                                                                  8
                                 166232485536961506887503125000·π

        8
  x·(26958302109890393212008267776 - 20779060692120188360937890625·π )
  ────────────────────────────────────────────────────────────────────── +
                                                       8
                  332464971073923013775006250000·π

                             8                         4                                           ~
  (7297565336750390625·π  + 124262203680000000·π  - 9361906584944377856)·(72975653367~
  ──────────────────────────────────────────────────────────────────────────────────~
                                                                                                   ~
                                                          13633141720100055583611350039062500~

         8                         4
  50390625·π  - 124262203680000000·π  - 9361906584944377856)
  ──────────────────────────────────────────────────────────
         16
  000000·π

can be factorized as

                                  8  2               4                   4                      ~
        (116761045388006250000·π ·x  - 253125000·π ·x·(230639102001·π  + 19636496384) + 729756~
#2:    ─────────────────────────────────────────────────────────────────────────────────────────~
                                                                                                 ~
                                                                                                 ~

                   8                         4                                             ~
  5336750390625·π  + 124262203680000000·π  - 9361906584944377856)·(116761045388006250 0~
  ───────────────────────────────────────────────────────────────────────────────────────~
                                                                                           ~
                                        1363314172010005558361135003906250000000 0·π~

      8  2             4                             4                          8    ~
  00·π ·x  + 253125000·π ·x·(19636496384 - 230639102001·π ) + 7297565336750390625·π  - ~
  ─────────────────────────────────────────────────────────────────────────────────────~
  16                                                                                   ~
                                                                                       ~

                          4
  124262203680000000·π  - 9361906584944377856)
  ─────────────────────────────────────────────


but Derive appears to be unable to do it
```

Problems with DERIVE's DISPLAY function (Günter's special trick!)

I don't know why, but from one day to the other I had problems with the DISPLAY function: it does not work.
I miss the output of 3 between #2 and #3.

```
       test(a) :=
         Prog
#1:        DISPLAY(a)
           a·a

#2:    test(3)

#3:    9
```

I sent a mail to our DERIVE expert, Günter Schödl. He gave some advice and finally the last one helped:

Hallo
Was du noch probieren kannst:
Rechte Maustaste auf die
Setup.exe => Eigenschaften =>

Dort kannst du jetzt versch. Kompatibilitätseinstellungen ausprobieren =>

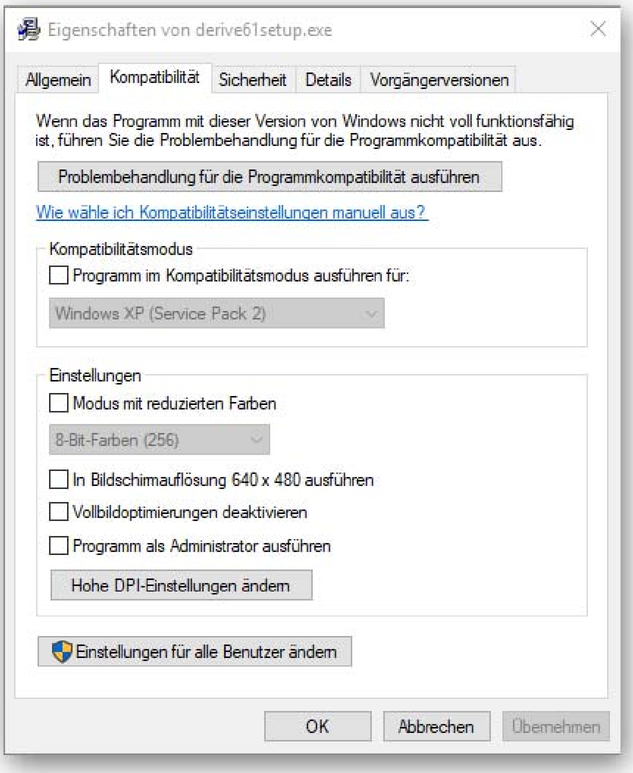
Ich glaub zwar nicht, dass das hilft aber versuch's einmal.

Try this: right mouse click on Setup.exe => Properties =>

There you can try different compatibility settings.

I don't believe that this will help, but it is worth a try!

Günter


Eigenschaften von derive61setup.exe

Allgemein | Kompatibilität | Sicherheit | Details | Vorgängerversionen

Wenn das Programm mit dieser Version von Windows nicht voll funktionsfähig ist, führen Sie die Problembehandlung für die Programmkompatibilität aus.

Problembehandlung für die Programmkompatibilität ausführen

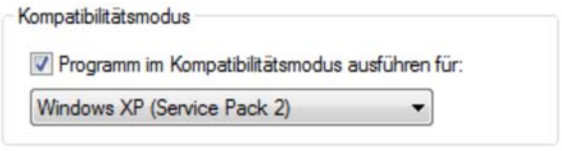
Wie wähle ich Kompatibilitätseinstellungen manuell aus?

Kompatibilitätsmodus
☐ Programm im Kompatibilitätsmodus ausführen für:
Windows XP (Service Pack 2)

Einstellungen
☐ Modus mit reduzierten Farben
8-Bit-Farben (256)
☐ In Bildschirmauflösung 640 x 480 ausführen
☐ Vollbildoptimierungen deaktivieren
☐ Programm als Administrator ausführen
Hohe DPI-Einstellungen ändern

Einstellungen für alle Benutzer ändern

OK | Abbrechen | Übernehmen


Kompatibilitätsmodus
☑ Programm im Kompatibilitätsmodus ausführen für:
Windows XP (Service Pack 2)

Dear Günter,
Your advice was – as always - super.
The system offered to work with Service Pack 2 – and immediately DISPLAY worked as expected.
Many thanks.

I will include this in the next DNL as „Günter's Special TRICK".

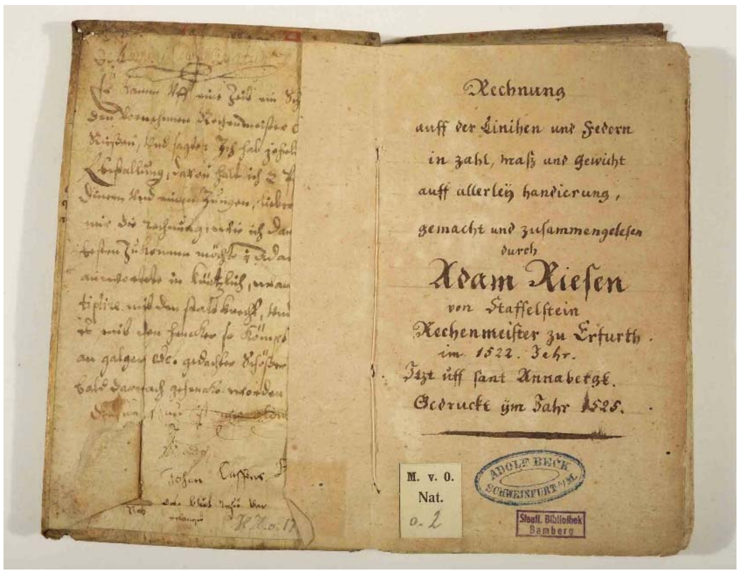
Regards
Josef

```
        test(a) :=
          Prog
#1:         DISPLAY(a)
            a·a

#2:     test(3)

3

#3:     9
```

Regula Falsi

Roland Schröder, Germany

"Regula falsi“ is the "Rule of the Wrong“, i.e. „ … of the wrong approach“. The wrong approach or the intentionally made mistake is always then part of a successful heuristic method when the mistake can be corrected just bevor end of the calculation, because its quantity or its structure are known, and we don't forget that we made a mistake before. Let's remember the Ethiopian farmer, who performing the "Abyssinian Multiplication“ (DNL#73) corrects the – intended - mistake at bisecting at the very end of his calculation.



[illegible]

Rechnung
auff der Linihen und Federn
in zahl, maß und gewicht
auff allerley handierung,
gemacht und zuſammengeleſen
durch
Adam Rieſen
von Staffelſtein
Rechenmeiſter zu Erfurth
im 1522. Jehr.
Jtzt uff ſant Annabergk.
Gedruckt ym Jahr 1525.

http://digital.bib-bvb.de/

Adam Ries (also Ries or Risen) poses in his "Rechenbüchlin“ (1522) a task entitled "Regula falsi“ which would read in our language of today similar like:

> "Somebody makes investments until the amount has doubled. Then he takes away one guilder (Gulden) and invests the remaining amount again until it has doubled. Then he takes away two guilders and invests the remaining a third time until it has doubled. When he now takes away three guilders 10 guilders remain. What was his seed capital?"

In today's secondary school mathematics, we would solve the problem as follows:

Amount at the beginning		x
Doubling and removing 1 guilder		$2x – 1$
Second doubling and removing 2 guilders		$2(2x-1) – 2$
Third doubling and removing 3 guilders		$2(2(2x – 1) – 2) – 3$
10 guilders remain		$2(2(2x – 1) – 2) – 3 = 10$
Solving	$8x – 11 = 10$	$x = 2\ {}^{5}/_{8}$

As one guilder was eight shillings, the "somebody" started with two guilders and five shillings. Adam Ries proposes quite another solving process: he writes – translated in the language of today:

> "Assume that he had started with three guilders, then after doubling and removing one guilder, five guilders remain. After next doubling and removing two guilders he would have eight guilders invested. The next step leads to 13 guilders – but it should lead to the final amount of only ten guilders which makes a difference of plus three. (Ries calls this difference "*lie*".) Another assumption, that it had been four guilders at the beginning would result in a "*lie*" of 11 guilders"

Adam Ries writes down both lies together with the difference of both lies (11 – 3 = 8) in table:

$$\begin{array}{ccc} 3 & +3 & \\ & & 8 \\ 4 & +11 & \end{array}$$

Then he multiplies the numbers placed in the vertices of the square crosswise, subtracts the products and divides this difference by 8: $(3\cdot 11 - 4\cdot 3):8 = 2\,{}^{5}/_{8}$.

Ries does not explain, why this method always gives the correct result. Our school mathematics can give the explanation: The assignment

assumed starting amount (x) → *difference from final amount (lie; y)*

is, as we could show above a linear assignment. Using Ries' instruction we have already two pairs of numbers. We can set up the equation of the linear function (equation of a line with two points given):

$$(*) \qquad \frac{11-3}{4-3} = \frac{y-3}{x-3}$$

We ask which argument r (correct starting amount) will give the error 0? Substituting point (r/0) in equation (*) gives $r = 2\,{}^{5}/_{8}$ (with or without DERIVE).

Choosing other pairs (u/v) and (w/z) instead of these which have been found by Ries, then we receive performing the same procedure as above:

$$(**) \qquad \frac{z-v}{w-u} = \frac{0-v}{r-u} \quad (**)$$

We solve equation (**) for r and get:

$$(***) \qquad r = \frac{uz-vw}{z-v} \quad (***)$$

This exactly what Ries has written and calculated in – generalized – form:

$$\begin{array}{ccc} u & v & \\ & & z-v \\ w & z & \end{array}$$

This quadratic arrangement of numbers *u, v, w, z* is called determinant and is calculated exactly in such a way as Ries demands in his solving procedure. So, we can suppose that he knew the concept of determinants. DERIVE provides the following command to calculate the determinant:

```
DET([[u,v],[w,z]]) or DET([u,v;w,z]).
```

For pairs [assumption/lie] = [*u, v*] and [*w, z*] DERIVE gives the starting amount (solution of the problem from above) as expression:

```
DET([u,v;w,z])/(z-v)
```

$$\frac{\text{DET}\begin{bmatrix} u & v \\ w & z \end{bmatrix}}{z - v}$$

This expression which Ries only applied on linear functions can be used for an approximation method to find zeros of nonlinear functions. Let F(*x*) the expression of a function with a zero – with change of signs - lying in interval [u, w] then

```
r(u,w):=DET([u,f(u);w,f(w)])/(F(w)-f(u))
```

$$r(u,\ w) := \frac{\text{DET}\begin{bmatrix} u & F(u) \\ w & F(w) \end{bmatrix}}{F(w) - F(u)}$$

gives a position *r* which is closer at the zero as *u* or *w*. This approach can be extended to a recursion if *u* or *w* are substituted by *r*. In case of F(*r*) < 0, interval [*u*, *w*] must be replaced by [*r*, *w*], otherwise by [*u*, *r*]. You may compare how the "Regula falsi" is represented in a formulary.

This is the DERIVE-extension to the recursion:

```
#1:   Precision := Approximate

#2:   Notation := Decimal

      reg_falsi(eq, v, u, w, eps, r, fu, fw) :=
        Prog
          Loop
            fu := LIM(eq, v, u)
            fw := LIM(eq, v, w)
            r := DET([u, fu; w, fw])/(fw - fu)
#3:         DISPLAY(r)
            If ABS(u - r) < eps ∨ ABS(w - r) < eps
               RETURN r
            If r < 0
               u := r
               w := r
```

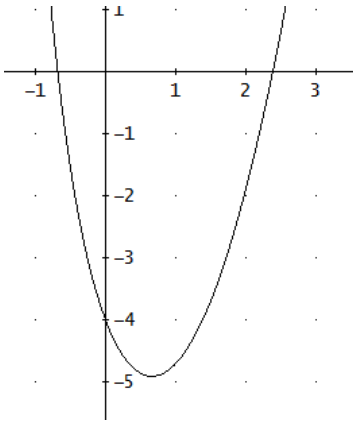
$$\#4: \quad \text{reg\_falsi}\left(\text{COS}(x),\ x,\ 1,\ 3,\ \frac{1}{1000}\right)$$

```
1.706141463

1.565033038

1.571125072

1.570777784

#5:   1.570777784
```

Assume that we would like to find the zeros of the following function:

#6: $\text{EXP}(-2\cdot x) + x^2 - \text{SIN}(x) - 5$

We plot the graph for estimating the intervals containing possible zeros and then call the function:



#7: $\text{reg\_falsi}\left(\text{EXP}(-2\cdot x) + x^2 - \text{SIN}(x) - 5,\ x,\ 2,\ 3,\ \frac{1}{1000}\right)$

```
2.328732587
2.389195374
2.382465614
2.383199709
```

#8: 2.383199709

We can check our routine using DERIVE's NSOLVE-command. And then we will remove the DISPLAY-command from the code, because it is not necessary to have all intermediate approximations. We are only interested in the final result.

#9: $\text{NSOLVE}(\text{EXP}(-2\cdot x) + x^2 - \text{SIN}(x) - 5 = 0,\ x)$

#10: $x = -0.6812688887$

#11: $\text{NSOLVE}(\text{EXP}(-2\cdot x) + x^2 - \text{SIN}(x) - 5 = 0,\ x,\ 2,\ 3)$

#12: $x = 2.383127361$

#13: $\text{reg\_falsi}\left(\text{EXP}(-2\cdot x) + x^2 - \text{SIN}(x) - 5,\ x,\ -1,\ 0,\ \frac{1}{1000}\right)$

#16: $\text{reg\_falsi}\left(\text{EXP}(-2\cdot x) + x^2 - \text{SIN}(x) - 5,\ x,\ -1,\ 0,\ \frac{1}{1000}\right) = -0.6809771520$

#17: $\text{reg\_falsi}\left(\text{EXP}(-2\cdot x) + x^2 - \text{SIN}(x) - 5,\ x,\ -1,\ 0,\ \frac{1}{100000}\right) = -0.6812651889$

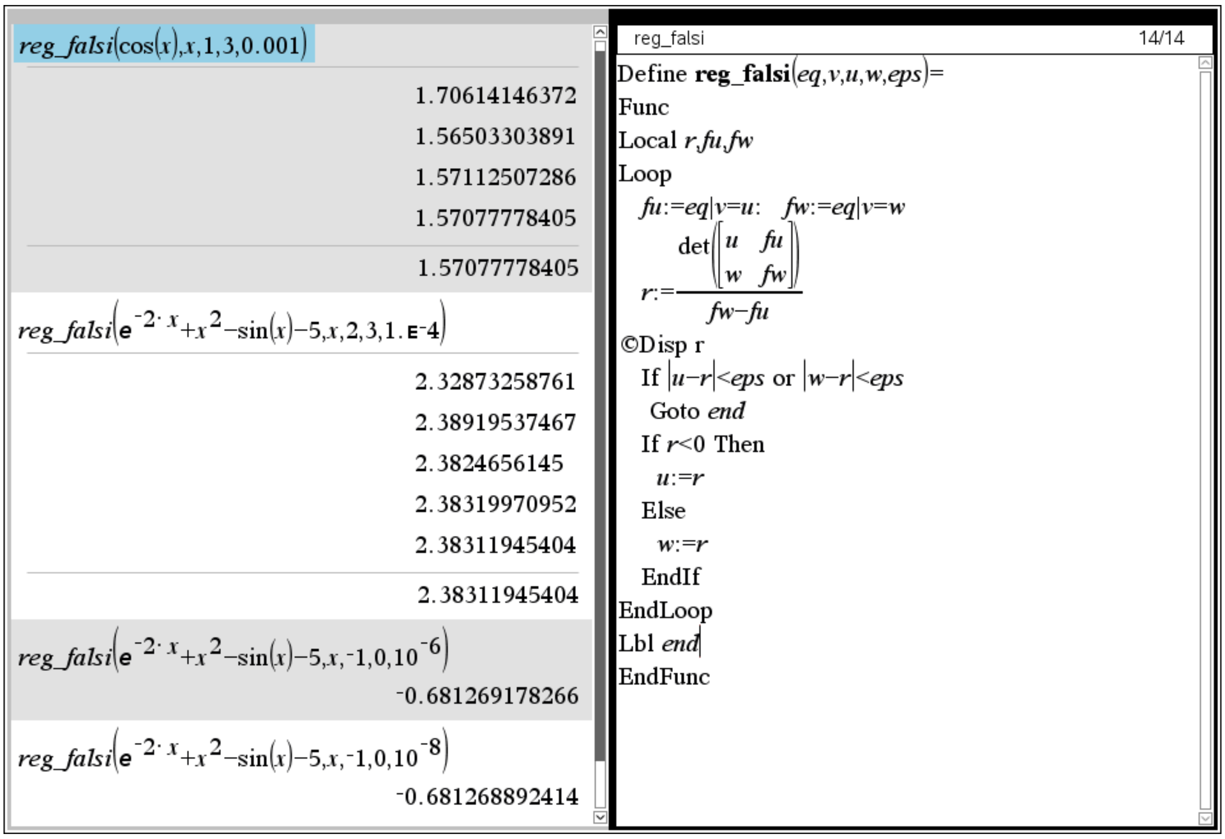
#18: $\text{reg\_falsi}(\text{EXP}(-2\cdot x) + x^2 - \text{SIN}(x) - 5,\ x,\ -1,\ 0,\ 10^{-6}) = -0.6812691782$

#19: PrecisionDigits := 20

#20: NotationDigits := 10

#21: $\text{reg\_falsi}(\text{EXP}(-2\cdot x) + x^2 - \text{SIN}(x) - 5,\ x,\ -1,\ 0,\ 10^{-10}) = -0.6812688887$

The next page shows the TI-NspireCX CAS implementation.

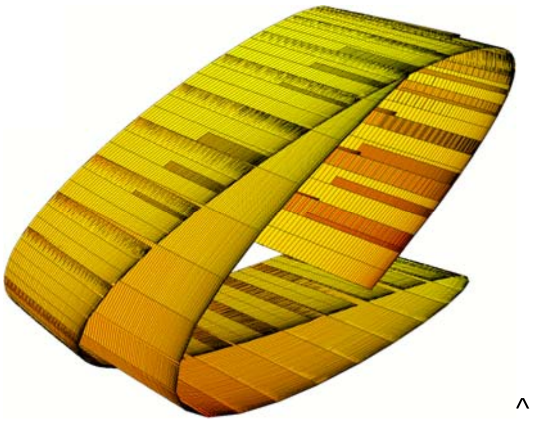


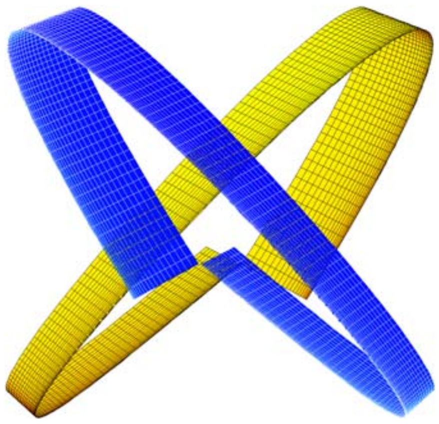
```
reg_falsi(cos(x),x,1,3,0.001)
                                1.70614146372
                                1.56503303891
                                1.57112507286
                                1.57077778405
                                1.57077778405
```
$reg\_falsi(e^{-2\cdot x}+x^2-\sin(x)-5,x,2,3,1.E-4)$
```
                                2.32873258761
                                2.38919537467
                                2.3824656145
                                2.38319970952
                                2.38311945404
                                2.38311945404
```
$reg\_falsi(e^{-2\cdot x}+x^2-\sin(x)-5,x,-1,0,10^{-6})$
```
                               -0.681269178266
```
$reg\_falsi(e^{-2\cdot x}+x^2-\sin(x)-5,x,-1,0,10^{-8})$
```
                               -0.681268892414
```

reg_falsi 14/14

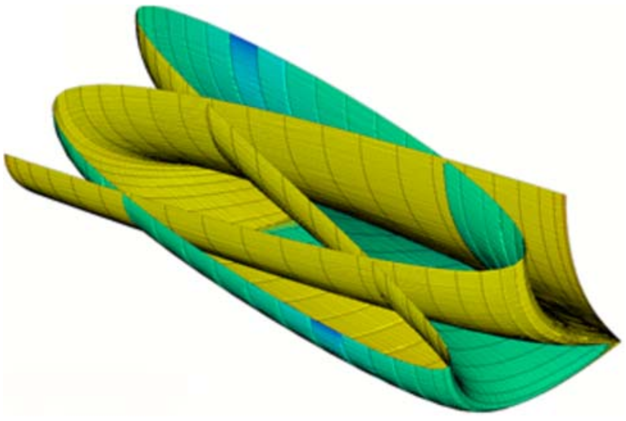
```
Define reg_falsi(eq,v,u,w,eps)=
Func
Local r,fu,fw
Loop
  fu:=eq|v=u:   fw:=eq|v=w
  r:= $\frac{\det\begin{pmatrix} u & fu \\ w & fw \end{pmatrix}}{fw-fu}$
©Disp r
  If |u-r|<eps or |w-r|<eps
   Goto end
  If r<0 Then
    u:=r
  Else
    w:=r
  EndIf
EndLoop
Lbl end
EndFunc
```

Three graphs from Pierre Charland's gallery (unfortunately without generating functions).





^



Rolling three dice and more … extended

Pierre Charland

Following the "Rolling three dice and more ..." article in DNL#62 p.34,

I wrote some functions to throw dice and compute
-- the frequency of 1-results, like two '1's , four '2's, ...
-- the frequency of 2-results, like two '1 followed by 2's, 5 '1 followed by 3's, ...

it works with
-- 1 (or many) dice
-- with faces 1..6 , or a..b, or a..b step d

```
NOTATION: dice(a,b,d) == dice with face values {a,a+d,a+2d,..., upto b}

NOTATION: dice(1,6,1) == regular dice with face values {1,2,3,4,5,6}
```

Throwing 1 dice 20 times would give

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

with 1-frequency

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 2 & 2 & 5 & 2 & 7 \end{bmatrix}$$

and 2-frequency (pair-frequency):

$$\begin{bmatrix} \# & 1 & 2 & 3 & 4 & 5 & 6 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 2 & 1 & 0 & 0 & 1 & 0 & 0 \\ 3 & 0 & 0 & 0 & 1 & 1 & 0 \\ 4 & 0 & 1 & 0 & 0 & 0 & 4 \\ 5 & 0 & 1 & 0 & 1 & 0 & 0 \\ 6 & 1 & 0 & 1 & 2 & 1 & 2 \end{bmatrix}$$

(the 1 and 4 in row#4 mean that we have 1 pair (4,2) and 4 pairs (4,6) of consecutive numbers)

I am attaching DiceSum.dfw

The dfw file has examples with display() output.

```
>> sum distribution array for d dice(1..6,1), as in (DNL-62 p.36) >> ex:DiceSumDistr(3) >>

                                  1    2    3    4    5    6 d
DiceSumDistr(d) := TABLE(POLY_COEFF((x  + x  + x  + x  + x  + x ) , x, k), k, d, 6·d)'

DiceSumDistr(3)
```

$$\begin{bmatrix} 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 \\ 1 & 3 & 6 & 10 & 15 & 21 & 25 & 27 & 27 & 25 & 21 & 15 & 10 & 6 & 3 & 1 \end{bmatrix}$$

We will generalize in two ways:

```
>> sum distribution array for d dice(v1..v2,dv) >> ex:DiceSumDistrX(1, 6, 1, 3) >>

DiceSumDistrX(v1, v2, dv, ddice) :=
  Prog
    If v1 = v2 ∨ dv = 0
       RETURN [ddice·v1; 1]
    If SIGN(v2 - v1) ≠ SIGN(dv)
       RETURN [ddice·v1; 1]
    nv1 := FLOOR(ABS((v2 - v1)/dv)) + 1
    If nv1 = 1
       RETURN [ddice·v1; 1]
    nvd := ddice·nv1 - (ddice - 1)
    k1 := ddice·v1
    k2 := k1 + (nvd - 1)·dv
    t := TABLE(POLY_COEFF(Σ(x^i, i, v1, v2, dv)^ddice, x, k), k, k1, k2)'
    RETURN t

DiceSumDistrX(1, 6, 1, 3)

[ 3  4  5   6   7   8   9  10  11  12  13  14  15  16  17  18 ]
[ 1  3  6  10  15  21  25  27  27  25  21  15  10   6   3   1 ]

DiceSumDistrX(2, 8, 2, 3)

[ 6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 ]
[ 1  0  3  0   6   0  10   0  12   0  12   0  10   0   6   0   3   0   1 ]
```

We can combine two, three or even more dice with different numbers and values of faces:

```
>> sum distribution array for dice([[d1],..,[dk]]) >>
>> ex:DiceSumDistrA([[1,2,3,4,5,6],[1,2,3]]) >>

DiceSumDistrA(dice) :=
  Prog
    ndice := DIM(dice)
    poly := 1
    n := 1
    Loop
      If n > ndice exit
      dicen := SORT(dice↓n)
      polyn := Σ(x^dicen↓i, i, 1, DIM(dicen))
      poly := poly·polyn
      n := n + 1
    polyt := REVERSE(TERMS(EXPAND(poly)))
    poly1 := SUBST(polyt, x, 1)
    polyd := VECTOR((STRING(polyt↓i/poly1↓i))↓3, i, 1, DIM(polyt))
    RETURN [polyd, poly1]

DiceSumDistrA([[1, 2, 3, 4, 5, 6], [1, 2, 3]])

[ 2  3  4  5  6  7  8  9 ]
[ 1  2  3  3  3  3  2  1 ]

DiceSumDistrA([[1, 2, 3, 4], [10, 20]])

[ 11  12  13  14  21  22  23  24 ]
[  1   1   1   1   1   1   1   1 ]

DiceSumDistrA([[1, 2, 3, 4], [10, 20], [1, 2]])

[ 12  13  14  15  16  22  23  24  25  26 ]
[  1   2   2   2   1   1   2   2   2   1 ]
```

In the last example we have three – irregular - dice with faces {1,2,3,4}, {10,20}, {1,2}. (You can interpret the latter two as coins with faxes 10, 20 and 1,2 respective.

The next functions simulate random dice throws.

Comment: when Pierre Charland wrote his functions, it was not known how to produce different random numbers at every new start of the program. One way is to simplify RANDOM(0) before calling the first routine or – what I prefer – to include a “dummy” variable and include simplifying RANDOM(0) within the function. Josef

```
                                      ⎛ddice                                  ⎞
DiceThrowD_(ddice, ntimes) := VECTOR⎜  ∑    (1 + RANDOM(6)), i, 1, ntimes⎟
                                      ⎝ j=1                                   ⎠

DiceThrowD(ddice, ntimes, dummy) :=
  Prog
    dummy := RANDOM(0)
    VECTOR(∑(1 + RANDOM(6), j, 1, ddice), i, 1, ntimes)
```

`DiceThrowD_()` is Pierre’s original function, `DiceThrowD()` is my version.

I made all functions “really” randomly generating the throws. If you prefer the other version, just remove the “dummy-line”. Josef

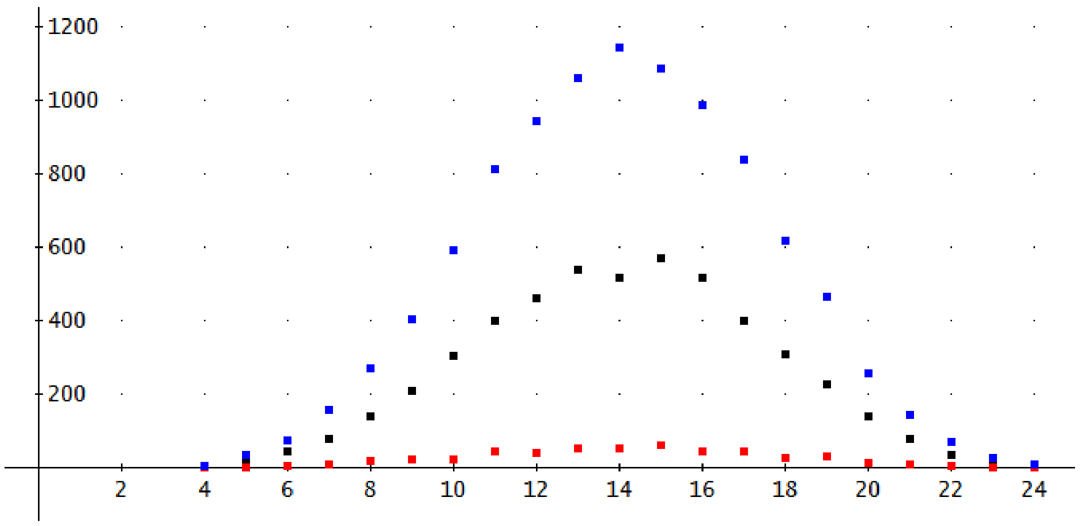
```
DiceThrowD(2, 20)

[6, 6, 5, 7, 8, 3, 12, 12, 8, 8, 11, 6, 10, 10, 10, 9, 7, 5, 11, 8]

DiceThrowD(4, 20)

[13, 12, 13, 14, 12, 12, 14, 19, 8, 16, 16, 7, 13, 9, 9, 8, 15, 16, 11, 9]
```

Now we can start experimenting:

```
>> random result 1-frequency array for d dice(1..6,1) thrown n times >>

DiceThrowSumDistrD1(ddice, ntimes, dummy) :=
  Prog
    dummy := RANDOM(0)
    counts := [VECTOR(i, i, ddice, ddice·6), VECTOR(0, i, ddice, ddice·6)]
    offset := ddice - 1
    j := 1
    Loop
      If j > ntimes exit
      val := ∑(1 + RANDOM(6), i, 1, ddice)
      k := val - offset
      counts↓2↓k := counts↓2↓k + 1
      j := j + 1
    RETURN counts

DiceThrowSumDistrD1(3, 50)

⎡ 3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 ⎤
⎣ 1  0  3  3  4  4  3  11   9   4   3   3   2   0   0   0 ⎦

DiceThrowSumDistrD1(4, 100)

⎡ 4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 ⎤
⎣ 0  0  0  1  1  4   5   9   9  10  18  11  14   5   5   3   2   2   1   0   0 ⎦
```

We throw four regular dice 500, 5000 and 10000 times and plot the distribution of the sums of the faces:

```
DiceThrowSumDistrD1(4, 500)'

DiceThrowSumDistrD1(4, 5000)'

DiceThrowSumDistrD1(4, 10000)'
```



Distribution of the pairs maybe much more interesting:

```
DiceThrowSumDistrD12(ddice, ntimes, dummy) :=
  Prog
    dummy := RANDOM(0)
    w := VECTOR(i, i, ddice, ddice·6)
    dw := (6 - 1)·ddice + 1
    f1 := VECTOR(0, i, ddice, ddice·6)
    f12 := VECTOR(VECTOR(0, a, dw + 2), b, dw + 2)
    d1 := -ddice + 1
    d12 := -ddice + 3
    p := ∑(1 + RANDOM(6), i, 1, ddice)
    f1↓(p + d1) := f1↓(p + d1) + 1
    i := 2
    Loop
      If i > ntimes exit
      q := ∑(1 + RANDOM(6), i, 1, ddice)
      f1↓(q + d1) := f1↓(q + d1) + 1
      f12↓(p + d12)↓(q + d12) := f12↓(p + d12)↓(q + d12) + 1
      p := q
      i := i + 1
    f12↓1↓1 := ntimes
    f12↓1↓2 := "."
    f12↓2↓1 := "."
    f12↓2↓2 := "#"
    i := 1
    Loop
      If i > dw exit
      f12↓2↓(i + 2) := w↓i
      f12↓(i + 2)↓2 := w↓i
      f12↓1↓(i + 2) := f1↓i
      f12↓(i + 2)↓1 := f1↓i
      i := i + 1
    RETURN f12
```

We throw three dice 50 times and add the faces:

```
DiceThrowSumDistrD12(3, 50)

⎡ 50   .  0  0  3  1  3  8  9   7   9   5   2   1   0   1   1   0 ⎤
   .   #  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18
   0   3  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0
   0   4  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0
   3   5  0  0  0  0  1  0  1   1   0   0   0   0   0   0   0   0
   1   6  0  0  0  0  0  0  1   0   0   0   0   0   0   0   0   0
   3   7  0  0  2  0  0  0  0   1   0   0   0   0   0   0   0   0
   8   8  0  0  0  0  0  1  4   0   1   1   0   0   0   0   0   0
   9   9  0  0  1  0  0  3  0   1   1   2   1   0   0   0   0   0
   7  10  0  0  0  0  1  1  0   0   0   2   0   1   0   1   1   0
   9  11  0  0  0  1  0  1  1   0   5   0   1   0   0   0   0   0
   5  12  0  0  0  0  1  1  2   1   0   0   0   0   0   0   0   0
   2  13  0  0  0  0  0  0  0   1   1   0   0   0   0   0   0   0
   1  14  0  0  0  0  0  1  0   0   0   0   0   0   0   0   0   0
   0  15  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0
   1  16  0  0  0  0  0  0  0   0   1   0   0   0   0   0   0   0
   1  17  0  0  0  0  0  0  0   1   0   0   0   0   0   0   0   0
⎣  0  18  0  0  0  0  0  0  0   0   0   0   0   0   0   0   0   0 ⎦
```

The distribution table says, that we have:

no pairs starting with 3, 4, 15 and 18; 3 pairs starting with 5: (5,7), (5,9), (5,10), …, 9 pairs starting with 11: (11,6), (11,8), (11,9), 5 times (11,11) and (11,13); …

The next three functions followed by sample calls are:

```
>> random result array for d dice(v1..v2,dv) thrown n times >>
>> random result 1-frequency array for d dice(v1..v2,dv) thrown n times >>
>> random result 1,2-frequency array for d dice(v1..v2,dv) thrown n times >>
```

```
DiceThrowDX(1, 6, 1, 3, 50)

[9, 6, 10, 12, 6, 10, 10, 14, 13, 6, 15, 13, 12, 12, 10, 15, 8, 8, 11, 11, 9, 11, 7,
  12, 8, 13, 9, 9, 7, 9, 7, 8, 10, 8, 8, 8, 8, 12, 11, 11, 9, 9, 12, 13, 7, 5, 11, 10,
  10, 11]

DiceThrowDX(2, 6, 2, 3, 50)

[14, 10, 8, 8, 10, 12, 16, 10, 12, 8, 8, 12, 12, 14, 10, 10, 10, 14, 14, 10, 8, 8, 14,
  18, 12, 12, 14, 12, 8, 12, 12, 10, 10, 8, 14, 10, 18, 12, 8, 16, 10, 16, 12, 12, 8,
  14, 16, 10, 12, 14]
```

We will show only the code of the first function.

```
DiceThrowDX(v1, v2, dv, ddice, ntimes) :=
  Prog
    If v1 = v2 ∨ dv = 0
       RETURN VECTOR(ddice·v1, i, 1, ntimes)
    If SIGN(v2 - v1) ≠ SIGN(dv)
       RETURN VECTOR(ddice·v1, i, 1, ntimes)
    nv1 := FLOOR(ABS((v2 - v1)/dv)) + 1
    If nv1 = 1
       RETURN VECTOR(ddice·v1, i, 1, ntimes)
    v := VECTOR(Σ(v1 + dv·RANDOM(nv1), j, 1, ddice), i, 1, ntimes)
    RETURN v
```

`DiceThrowDX(2,6,2,3,50)` simulates 50 throws with 3 nonregular dice with faces {2, 4, 6}.

```
DiceThrowSumDistrDX1(2, 6, 2, 3, 50)
```

$$\begin{bmatrix} 6 & 8 & 10 & 12 & 14 & 16 & 18 \\ 3 & 7 & 14 & 15 & 7 & 4 & 0 \end{bmatrix}$$

```
DiceThrowSumDistrDX12(2, 6, 2, 3, 50)
```

$$\begin{bmatrix} 50 & . & 0 & 8 & 12 & 7 & 18 & 1 & 4 \\ . & \# & 6 & 8 & 10 & 12 & 14 & 16 & 18 \\ 0 & 6 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 8 & 8 & 0 & 0 & 4 & 2 & 0 & 0 & 1 \\ 12 & 10 & 0 & 0 & 2 & 2 & 6 & 1 & 1 \\ 7 & 12 & 0 & 2 & 1 & 0 & 2 & 0 & 2 \\ 18 & 14 & 0 & 4 & 5 & 1 & 8 & 0 & 0 \\ 1 & 16 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 4 & 18 & 0 & 1 & 0 & 1 & 2 & 0 & 0 \end{bmatrix}$$

```
>> 1-frequency table of an array >> ex:Frequency1([5,2,2,5,2]) >>
>> 2-frequency table of an array >> ex:Frequency2([5,2,2,5,2]) >>
>> 1,2-frequency table of an array >> ex:Frequency12([5,2,2,5,2]) >>
```

The last functions enable analyzing a given array of throws. Intermediate results are displayed as text objects using DERIVE's `DISPLAY`-function.

The "`DISPLAYs`" are the same in all functions. It is no problem to remove them. In the following examples I left the messages in the first function (Frequency1()) and removed them in the other two ones.

We throw two dice 20 times and analyze the outcomes:

```
v1 := DiceThrowD(2, 20)

v1 := [6, 11, 9, 9, 8, 4, 6, 3, 4, 8, 4, 6, 7, 6, 8, 9, 4, 12, 8, 6]
```

We proceed treating this series of throws in the three ways described above
:

```
#96:  Frequency1(v1)

[6, 11, 9, 9, 8, 4, 6, 3, 4, 8, 4, 6, 7, 6, 8, 9, 4, 12, 8, 6]

[3, 4, 4, 4, 4, 6, 6, 6, 6, 6, 7, 8, 8, 8, 8, 9, 9, 9, 11, 12]

[3, 4, 6, 7, 8, 9, 11, 12]

      ┌ 3  4  6  7  8  9  11  12 ┐
#97:  │                          │
      └ 1  4  5  1  4  3   1   1 ┘

#98:  Frequency2(v1)

      ┌  #  3  4  6  7  8  9  11  12 ┐
      │  3  0  1  0  0  0  0   0   0 │
      │  4  0  0  2  0  1  0   0   1 │
      │  6  1  0  0  1  1  0   1   0 │
#99:  │  7  0  0  1  0  0  0   0   0 │
      │  8  0  2  1  0  0  1   0   0 │
      │  9  0  1  0  0  1  1   0   0 │
      │ 11  0  0  0  0  0  1   0   0 │
      └ 12  0  0  0  0  1  0   0   0 ┘

#100: Frequency12(v1)

      ┌ 20   .  1  4  5  1  4  3   1   1 ┐
      │  .   #  3  4  6  7  8  9  11  12 │
      │  1   3  0  1  0  0  0  0   0   0 │
      │  4   4  0  0  2  0  1  0   0   1 │
      │  5   6  1  0  0  1  1  0   1   0 │
#101: │  1   7  0  0  1  0  0  0   0   0 │
      │  4   8  0  2  1  0  0  1   0   0 │
      │  3   9  0  1  0  0  1  1   0   0 │
      │  1  11  0  0  0  0  0  1   0   0 │
      └  1  12  0  0  0  0  1  0   0   0 ┘
```

Geometric Patterns for Rational Numbers

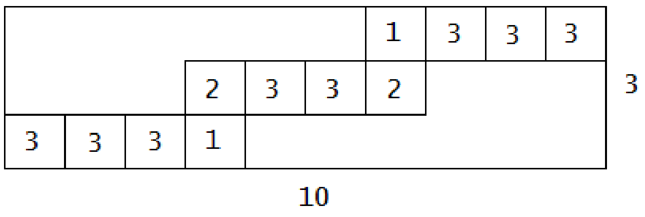
Josef Lechner, Austria

This is a teaching unit to get an unusual presentation of rational numbers and subsequently to build a bridge to matrix calculation.

Problem – Rational Patterns (Hooks) and Matrices

Investigate rational hooks, i.e. graphical arrangements of positive fraction numbers.

On the right you can see the hook for $\frac{10}{3}$.



The procedure for filling the table is as follows:

(1) Let's consider first only positive rational numbers with numerator n ≥ denominator d.

- Begin at bottom left with the denominator d (and proceed in right direction).
- Proceed with d as often as possible before reaching the numerator, then fill in the remainder (the sum in one row gives the numerator).
- Make one step upwards and fill in the next cell to accomplish the denominator.
- Go right and fill in the denominator until you reach again the numerator and finish with the remainder, one step up, …
- Finally, you receive an arrangement of d rows with sum n and n columns with sum d.

Tasks:

a) Construct the rational hooks for $\frac{7}{3}, \frac{12}{5}$ and $\frac{17}{7}$.

b) Check, that integers, i.e. rational numbers with denominator 1 give hooks with one row.

c) Check that extended fractions (gcd(n,d) ≠ 1) lead to hooks of the same form (same numbers, too?). What about numbers of rows and columns?

d) Show using self-chosen fractions that all hooks have a symmetric form.

e) Represent two hooks in matrix form.

(2) Now let's consider positive rational numbers ≤ 1 (with numerator n ≤ denominator d).

- Begin at bottom left with the numerator n (and proceed in upwards direction).
- Proceed with d as often as possible before reaching the numerator, then fill in the remainder (the sum in one column gives the numerator).
- Make one step right and fill in the next cell to accomplish the numerator.
- Go up and fill in the numerator until you reach again the denominator and finish with the remainder, one step right, …
- Finally, you receive an arrangement of d rows with sum n and n columns with sum d.

Tasks:

f) Construct the rational hooks for $\frac{3}{10}, \frac{5}{12}$ and $\frac{7}{22}$.

g) Define the relation between the Zig-Zag-patterns of a fraction and its reciprocal value.

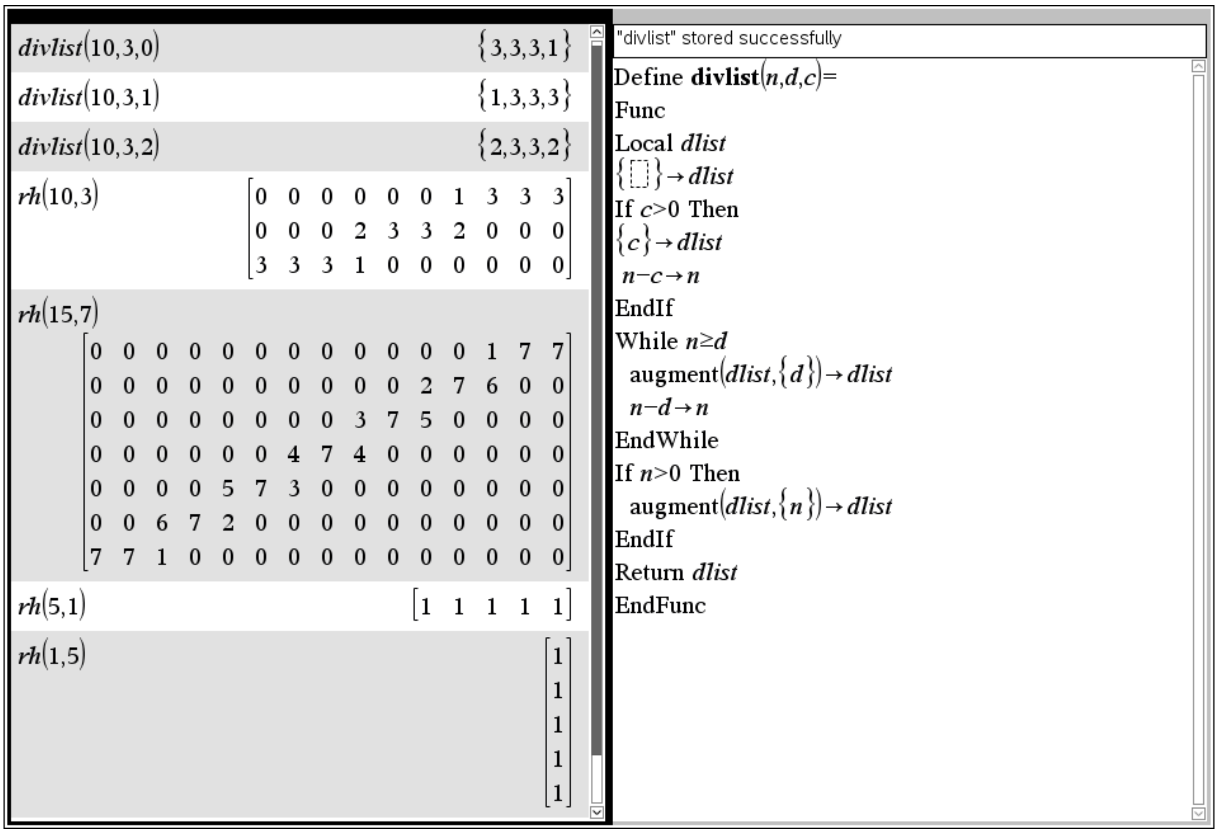
h) Present the patterns as matrices und investigate the properties of the matrix.

More tasks to follow

Proposal how to realize the patterns in classroom

After the students have constructed some hooks with paper and pen, we can focus on an algorithmic treatment of the representation of the zig-zags (as a matrix).

First of all, we define a function, which creates the rows of one hook.



```
divlist(10,3,0)        {3,3,3,1}
divlist(10,3,1)        {1,3,3,3}
divlist(10,3,2)        {2,3,3,2}
rh(10,3)
  [0 0 0 0 0 0 1 3 3 3]
  [0 0 0 2 3 3 2 0 0 0]
  [3 3 3 1 0 0 0 0 0 0]
rh(15,7)
  [0 0 0 0 0 0 0 0 0 0 0 0 1 7 7]
  [0 0 0 0 0 0 0 0 0 0 2 7 6 0 0]
  [0 0 0 0 0 0 0 0 3 7 5 0 0 0 0]
  [0 0 0 0 0 0 4 7 4 0 0 0 0 0 0]
  [0 0 0 0 5 7 3 0 0 0 0 0 0 0 0]
  [0 0 6 7 2 0 0 0 0 0 0 0 0 0 0]
  [7 7 1 0 0 0 0 0 0 0 0 0 0 0 0]
rh(5,1)                [1 1 1 1 1]
rh(1,5)
  [1]
  [1]
  [1]
  [1]
  [1]
```

```
"divlist" stored successfully
Define divlist(n,d,c)=
Func
Local dlist
{□}→dlist
If c>0 Then
{c}→dlist
 n−c→n
EndIf
While n≥d
  augment(dlist,{d})→dlist
  n−d→n
EndWhile
If n>0 Then
  augment(dlist,{n})→dlist
EndIf
Return dlist
EndFunc
```

The parameters are:

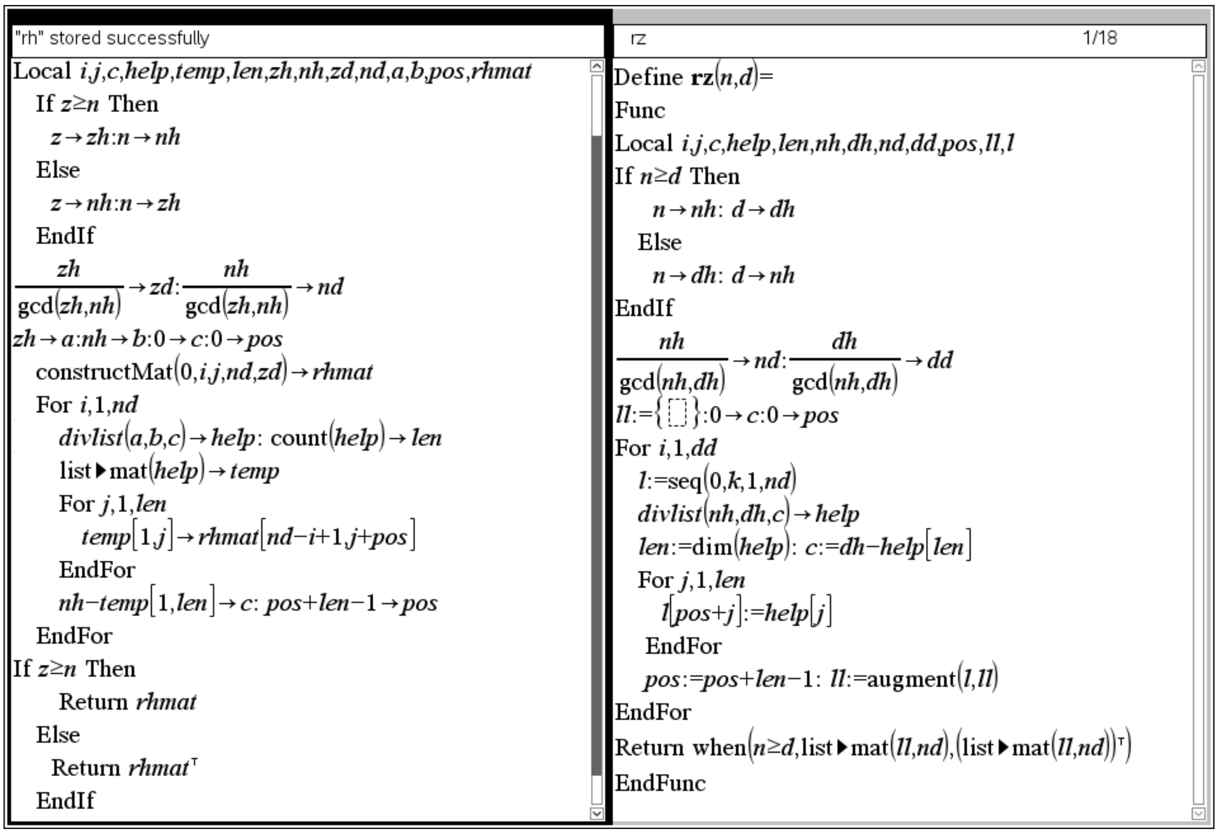
n … the numerator
d … the denominator
c … the carry

Comment: command augment(list1,list2) allows to concatenate two lists.

Function divlist(n,d,c) delivers a list of subsequent summands starting with a possible carry from the line before. It needs only subtractions.

Now, a second function comes into play: rh(n,d). It produces the matrix in order to make the pattern visible. The remaining elements are filled with zeros. It is recommended to use rh(n,d) as black box for students.

Comment: Basically, the lists generated by divlist are embedded on the appropriate place in a zero-matrix (left function code). Another possibility right function code) – without double indices – is making the rows first, appending them and finally converting the list (n · d elements) into a matrix with appropriate dimensions (n × d). A third way will be presented in the appendix where the DERIVE-treatment invites to apply the sophisticated ITERATE-function to do the job.



```
"rh" stored successfully
Local i,j,c,help,temp,len,zh,nh,zd,nd,a,b,pos,rhmat
  If z≥n Then
    z→zh:n→nh
  Else
    z→nh:n→zh
  EndIf
zh/gcd(zh,nh)→zd: nh/gcd(zh,nh)→nd
zh→a:nh→b:0→c:0→pos
  constructMat(0,i,j,nd,zd)→rhmat
  For i,1,nd
    divlist(a,b,c)→help: count(help)→len
    list▶mat(help)→temp
    For j,1,len
      temp[1,j]→rhmat[nd−i+1,j+pos]
    EndFor
    nh−temp[1,len]→c: pos+len−1→pos
  EndFor
If z≥n Then
    Return rhmat
  Else
   Return rhmat^T
  EndIf
```

```
rz                                            1/18
Define rz(n,d)=
Func
Local i,j,c,help,len,nh,dh,nd,dd,pos,l1,l
If n≥d Then
   n→nh: d→dh
  Else
   n→dh: d→nh
EndIf
nh/gcd(nh,dh)→nd: dh/gcd(nh,dh)→dd
l1:={ }:0→c:0→pos
For i,1,dd
  l:=seq(0,k,1,nd)
  divlist(nh,dh,c)→help
  len:=dim(help): c:=dh−help[len]
  For j,1,len
    l[pos+j]:=help[j]
  EndFor
  pos:=pos+len−1: l1:=augment(l,l1)
EndFor
Return when(n≥d,list▶mat(l1,nd),(list▶mat(l1,nd))^T)
EndFunc
```

More tasks and experiments:

i) Multiply the matrices generated of a fraction and its reciprocal value. We will call this multiplication as fraction-hook-multiplication. Is this multiplication commutative?

j) What makes the difference? What are the sums of the rows and columns?

k) Calculate the determinant for both cases: What do you find out?

l) In one case is the determinant ≠ 0. Is it possible to determine the value of the determinant as function of n and d?

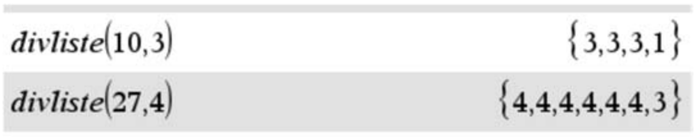
m) Why is the determinant equal zero in the other case?

Didactical comment:

This project shall students make familiar with an unusual presentation of rational numbers. Additionally, they are invited to discover relationships between rational numbers and matrices. They can think about connections between division with remainder and the presentation as fraction resp. matrix. Division with remainder plays a key role for representation of a rational number as a continued fraction.

Methodical comment:

Technological environment is used for experimentation. It is on the teacher to use functions divlist and/or rh/rz as black box. divlist can be result of an algorithmic side step. You might prefer to start with a simplified version (without carry), divliste:



```
divliste(10,3)                    {3,3,3,1}
divliste(27,4)                {4,4,4,4,4,4,3}
```

Possible further extensions:

Can we define an addition for rational hooks?

See e.g. rh(2,3) + rh(1,3) in form augment(rh(2,3),rh(1,3)). Properties of the resulting matrix?

What are the differences between magic squares and rational hooks?

What are the differences magic squares and the product matrices of rational hooks of fractions and their reciprocals? What do they have in common?

Find more investigations in depth in Wolfgang Rump's "Rationale Zahlen und Figuren" (*http://www.iaz.uni-stuttgart.de/LstAGeoAlg/Rump/study/haken.pdf*). The textbook Mathematik 5, Veritas, Linz, C. Brand, A.Dorfmayr, J.Lechner u.a (2009) shows some ideas for this project.

Appendix: The DERIVE implementation

As said above, we will show the DERIVE code with and without the ITERATE-function.

`divlist` is very similar to the Nspire code. As we cannot apply while-wend, we use loop.

The algorithm is quite the same.

```
divlist(a, b, c) :=
  Prog
    If c > 0
       l := [c]
       l := []
    a := a - c
    Loop
      If a ≥ b
         l := APPEND(l, [b])
         exit
      a := a - b
    If a > 0
       RETURN APPEND(l, [a])
       RETURN l
```

`lm(n,d)` generates a zero-matrix with appropriate rows and columns (assumed n > d):

$$\mathrm{lm}(n,\ d) := \mathrm{VECTOR}\left(\mathrm{VECTOR}\left(0,\ i,\ 1,\ \frac{n}{\mathrm{GCD}(n,\ d)}\right),\ j,\ 1,\ \frac{d}{\mathrm{GCD}(n,\ d)}\right)$$

$$\mathrm{lm}(10,\ 3) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

`embed(list, mat, px, py)` puts the `divlist`-generated vector (list) on position `px, py` into the matrix.

$$\mathrm{embed}(\mathrm{list},\ \mathrm{mat},\ \mathrm{px},\ \mathrm{py}) := \left(\mathrm{ITERATE}\left(\left[\mathrm{REPLACE}\left(\mathrm{REPLACE}\left(\mathrm{list}_{i\_},\ \mathrm{mat\_}_{\mathrm{py}},\ i\_ + \mathrm{px} - 1\right),\ \mathrm{mat\_},\ \mathrm{py}\right),\ i\_ + 1\right],\ [\mathrm{mat\_},\ i\_],\ [\mathrm{mat},\ 1],\ \mathrm{DIM}(\mathrm{list})\right)\right)_1$$

$$\mathrm{embed}(\mathrm{divlist}(10,\ 3,\ 0),\ \mathrm{nm},\ 1,\ 3) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3 & 3 & 3 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Another ITERATE does the job to fill in all rows (`rhh`) and finally, `rh` considers if n/d >1 or n/d <1.

$$\mathrm{rhh}(n,\ d) := \left(\mathrm{ITERATE}\left(\left[\mathrm{embed}(\mathrm{divlist}(n,\ d,\ \mathrm{hc}),\ \mathrm{hmat},\ \mathrm{hpos},\ \mathrm{hpy}),\ \mathrm{hpos} + \mathrm{DIM}(\mathrm{divlist}(n,\ d,\ \mathrm{hc})) - 1,\ \mathrm{hpy} - 1,\ \frac{d}{\mathrm{GCD}(n,\ d)} - \mathrm{ELEMENT}(\mathrm{divlist}(n,\ d,\ \mathrm{hc}),\ \mathrm{DIM}(\mathrm{divlist}(n,\ d,\ \mathrm{hc})))\right],\ [\mathrm{hmat},\ \mathrm{hpos},\ \mathrm{hpy},\ \mathrm{hc}],\ \left[\mathrm{lm}(n,\ d),\ 1,\ \frac{d}{\mathrm{GCD}(n,\ d)},\ 0\right],\ d\right)\right)_1$$

```
rh(n, d) :=
  If n ≥ d
     rhh(n, d)
     rhh(d, n)`
```

It must be admitted that this code is not so easy to explain to students, who have problems with much simpler function codes. So, let's try another way. Just translate function rz(n,d) from above into DERIVE code, considering that for-next loops must treated a little bit the other way round.

```
rz(n, d, i, j, c, help, len, nh, nd, dd, pos, ll, l) :=
  Prog
    ll := []
    c := 0
    pos := 0
    If n ≥ d
       [nh := n, dh := d]
       [dh := n, nh := d]
    [nd := nh/GCD(nh, dh), dd := dh/GCD(nh, dh)]
    i := 1
    Loop
      If i > dd exit
      l := VECTOR(0, k, 1, nd)
      help := divlist(nh, dh, c)
      len := DIM(help)
      c := dh - help↓len
      j := 1
      Loop
        If j > len exit
        l↓(pos + j) := help↓j
        j :+ 1
      pos := pos + len - 1
      ll := APPEND([l], ll)
      i :+ 1
    If n > d
       RETURN ll
       RETURN ll`
```

Examples:

```
[rh(10, 3), rz(10, 3)]
```

$$\left[\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 3 & 3 \\ 0 & 0 & 0 & 2 & 3 & 3 & 2 & 0 & 0 & 0 \\ 3 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 3 & 3 & 3 \\ 0 & 0 & 0 & 2 & 3 & 3 & 2 & 0 & 0 & 0 \\ 3 & 3 & 3 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}\right]$$

$$\text{rz}(3, 7) = \begin{bmatrix} 0 & 0 & 3 \\ 0 & 0 & 3 \\ 0 & 2 & 1 \\ 0 & 3 & 0 \\ 1 & 2 & 0 \\ 3 & 0 & 0 \\ 3 & 0 & 0 \end{bmatrix}$$

I tried to find a way to enter rz(x) or rh(x) with x being the rational number. I was not successful. But when I sent the article to Josef Lechner he sent a dfw-file which accepts fractions as parameter (e.g. rz(3/7) and rz(7/3)). See file rational_hooks_impr.dfw. Josef

Change Count

Pierre Charland

Here is a program to compute the #ways to give change, (which is the number of partitions into some given parts). ChangeCount.dfw is actually my third version.

The 1st version was very short and simple, using Derive functions to multiply polynomials, but it was incredibly slow.

This new version, (which uses basically the same method, except that I had to code every-thing) is much faster.

ChangeCount(vec,n) returns [v1,v2,v3] where

[v1,v2,v3] = [amount, #ways, cumul #ways]
[v1,v2,v3] = [n, #partition(n) with parts in vec, #partition(0...n) with parts in vec]

for example,

```
#11:  ChangeCount([1, 5], 10)

n=1 cn=1

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

n=2 cn=5

[1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 3]
```

#12:

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 1 & 1 & 1 & 1 & 1 & 2 & 2 & 2 & 2 & 2 & 3 \\ 1 & 2 & 3 & 4 & 5 & 7 & 9 & 11 & 13 & 15 & 18 \end{bmatrix}$$

So, column 6,2,9 means using parts [1,5], there are 2 ways to get sum = 6:
(1+1+1+1+1+1) & (1+5),
using parts [1,5], there are 9 ways to get sum <= 6

```
#15:  ChangeCount([1, 3, 10], 20)

n=1 cn=1

[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

n=2 cn=3

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

n=3 cn=10

[1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 5, 5, 6, 7, 7, 8, 9, 9, 10, 11, 12]
```

#16:

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 & 17 & 18 & 19 & 20 \\ 1 & 1 & 1 & 2 & 2 & 2 & 3 & 3 & 3 & 4 & 5 & 5 & 6 & 7 & 7 & 8 & 9 & 9 & 10 & 11 & 12 \\ 1 & 2 & 3 & 5 & 7 & 9 & 12 & 15 & 18 & 22 & 27 & 32 & 38 & 45 & 52 & 60 & 69 & 78 & 88 & 99 & 111 \end{bmatrix}$$

Parts can be repeated, (as if of different color) so ChangeCount([1, 1], 10) returns

```
ChangeCount([1, 1], 10)
```

$$\begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\ 1 & 3 & 6 & 10 & 15 & 21 & 28 & 36 & 45 & 55 & 66 \end{bmatrix}$$

The fifth column [4, 5, 15] tells us that we having white and black coins available can get 5 different collections of colors for four coins:

(w,w,w,w) (b,w,w,w), (b,b,w,w), (b,b,b,w) and (w,w,w,w).

Remember combinatorics:

What is the number of ways taking *k* balls from a box containing *n* elements without regard to order and with replacement?

$$\binom{n+k-1}{k} \rightarrow \binom{2+4-1}{4} = \binom{5}{4} = 5$$

```
      a(n, k := 3, d := [1, 3, 10]) :=
        If n < 0
           0
#1:        If k = 1
              1
              a(n, k - 1) + a(n - d↓k, k)

#2:   a(20) = 12

#3:   VECTOR(a(n_), n_, 20)

#4:   [1, 1, 2, 2, 2, 3, 3, 3, 4, 5, 5, 6, 7, 7, 8, 9, 9, 10, 11, 12]

      a(n, k := 6, d := [1, 2, 5, 10, 20, 50]) :=
        If n < 0
           0
#5:        If k = 1
              1
              a(n, k - 1) + a(n - d↓k, k)

#6:   a(50) = 451

#7:   Σ(VECTOR(a(n_), n_, 50)) = 6148
```

```
(ChangeCount([1, 2, 5, 10, 20, 50], 50))↓↓51 = [50, 451, 6149]
```

Comment: Johann Wiesenbauer gave a solution to this problem in one of his famous Titbits: You can find it together with Rüdeger Baumann’s solution in DNL#48 (from 2002), Josef.

This is another contribution which has been waiting a very long time for its appearance in the DNL. It reminds us on a time when we couldn't admire 3D-plots with DERIVE. We either used the commands provided by the utility file GRAPHICS.MTH (ISOMTERIC, COPROJECTION, …) or we produced own functions for mapping 3D-objects on a 2D-plane.

I believe that even now, when we have 3D-plotting available in almost all systems (see exception at the end of Peter's contribution) it might be interesting and useful to demonstrate how this works. What we are calculating now (or something similar) is done by the programs in the hidden background – much faster but the basics should be the same.

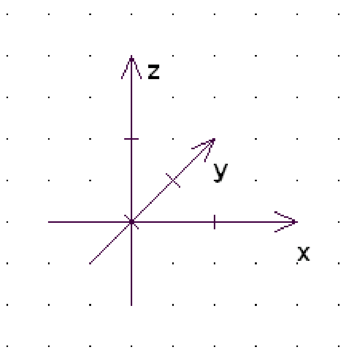
Peter produced tools for performing three oblique views of 3D-objects. He called them "Schrägbild 1", "Schrägbild 2" and "Schrägbild 3". "Schrägbild 3" is also known as "Trimetrie" or "Trimetic Projection".

As Peter's file is very long, I restrict its presentation on the trimetric projection (dfw-file ThreeD_New.dfw). I tried to add English comments. (Dreidimensionalneu.dfw is the full file with German comments.)

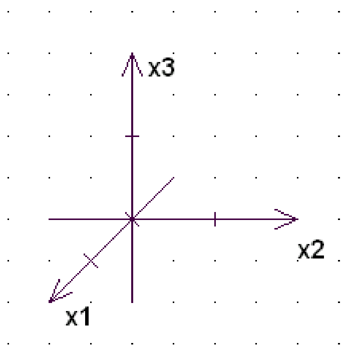
Tools for 3D-Problems

Peter Lüke-Rosendahl, Germany

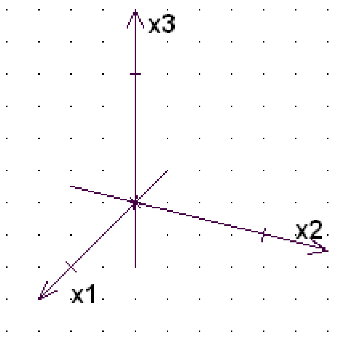
The sketches given below show the axes of coordinates for the three projections which are presented:



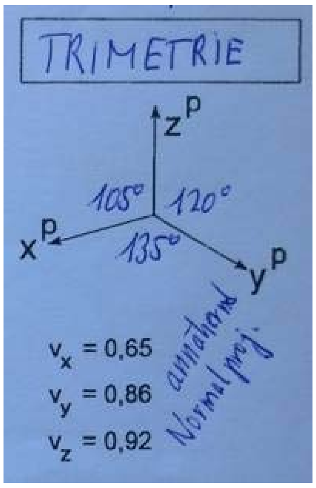
SB1



SB2



SB3 – Trimetric Projection



Comment Josef:

In trimetric projection, the viewing direction is such that all of the three axes of space appear unequally foreshortened (Wikipedia).

Peter's trimetric axes have the benefit that they can easily be drawn on a grid paper. The unit points are marked. His functions are based on this system.

The system depicted at left hand side is another often used system of axes for trimetric projection.

These are the functions provided for SB1, SB2 and SB3:

SB1_Punkt(rV)	Darstellung eines Punktes
SB1(Start,End)	Achsenkreuz mit Einheiten; in schräge Richtungen zwei Einheiten mehr
SB1_Strecke(rU,rV)	Strecke von U nach V
SB1_ORTS_VEKT(rV)	Ortsvektor mit Pfeil zum Punkt V
SB1_VEKT(rU,rV)	Vektorpfeil von U nach V
SB1_E_3P(rU,rV,rW)	3D-Darstellung der durch U,V und W festgelegten Ebene
SB1_E_Puv(rU,u,v)	3D-Darstellung der durch U und zwei Richtungsanteile festgelegten Ebene
SB1_g_2P(rU,rV)	3D-Darstellung der durch U und V festgelegten Geraden
SB1_g_PR(rU,v)	3D-Darstellung der durch U und Richtungsvektor v festgelegten Geraden
SB2_ …	
SB3_ …	

In the following file ThreeD_New.dfw I used appropriate English function names starting with Tr_ instead of SB3_ (Josef).

The functions will be explained when they are introduced and used the first time.

ThreeD_New.dfw

```
#1:   [InputMode := Word, CaseMode := Sensitive, NotationDigits := 4]
```

Set off Lines, Labels, Grid and Cross in Options Display Axes

****

Um Pfeile zu zeichnen, Tool von Biryukow:
2D Plots Labelling D-N-L#16; p12; 1994
S. Biryukow's Tool from DNL#16 is useful for plotting arrows

#2: $\left[\text{rot} := 0,\ \text{ROT\_V}(v,\ \text{rot}) := \begin{bmatrix} \text{COS(rot)} & -\text{SIN(rot)} \\ \text{SIN(rot)} & \text{COS(rot)} \end{bmatrix}\cdot v\right]$

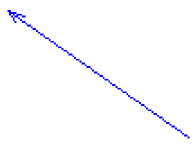
#3: $\text{ARROW}(p1,\ p2,\ l) := \left[p1,\ p2,\ p2 + \text{ROT\_V}\left(\dfrac{l\cdot(p2 - p1)}{|p2 - p1|},\ \dfrac{7}{8}\cdot\pi\right),\ p2,\ p2 + \text{ROT\_V}\left(\dfrac{l\cdot(p2 - p1)}{|p2 - p1|},\ -\dfrac{7}{8}\cdot\pi\right)\right]$

Die Pfeillänge kann hier verändert werden!
You can change length of arrowheads!

```
#4:   l := 0.3
```

Beispiel für die Pfeileingabe:
Examples for an arrow:

```
#5:   ARROW([2, -1], [-3, 4], l)
```



Definitionen für später benötigte Vektoren
Definitions of vectors for later use

```
#6:   [g := [g1, g2, g3], u := [u1, u2, u3], v := [v1, v2, v3]]
```

Ortsvektoren werden mit vorgestelltem 'r' bezeichner; z.B. rP
Positionvectors are initiated bei an 'r'; e.e. rP

```
#7:   [rS := [sx, sy, sz], rT := [tx, ty, tz], rU := [ux, uy, uz]]

#8:   [rV := [vx, vy, vz], rW := [wx, wy, wz]]
```

Für Beispiele
Points for the examples

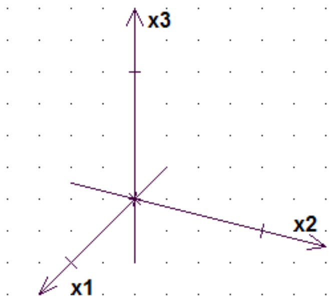
```
#9:   [rA := [2, -9, -2], rB := [8, 0, 1], rC := [0, 6, 3]]
```

ABCD ist ein Parallelogramm, daher rD = rA + (rC - rB)
ABCD is a rhomboid, so rD = rA + (rC - rB)

```
#10:  (rD := rA + (rC - rB)) = rD := [-6, -3, 0]

#11:  [rE := [1, 2, 3], rF := [5, 0, -1], rG := [3, 4, -5], rH := [-1, 6, -1]]

#12:  [rS1 := [6, 7, 1], rS2 := [-2, -1, -3], rP := [1, 1, 8], dir_l := [-1, -1, -2]]
```

****

***   Trimetrie Oblique Projection 3 (Trimetric Projection)   ***
****



Projection of one point:

$$\#13:\quad Tr\_Pt(rV) := \left[-rV_1 + 2\cdot rV_2,\ -rV_1 - \frac{1}{2}\cdot rV_2 + 2\cdot rV_3\right]$$

Trimetrie / Trimetric Projection
Einheiten x1-Achse - Units on x1-axis

$$\#14:\quad Step := \frac{1}{40}$$

```
#15:  Trp1_x1(Start, End) := VECTOR([-x - 3·Step, -x + 3·Step], x, Start, End, 1)

#16:  Trp2_x1(Start, End) := VECTOR([-x + 3·Step, -x - 3·Step], x, Start, End, 1)
```

$$\#17:\quad TrUnits\_x1(Start, End) := VECTOR\left(\left[(Trp1\_x1(Start, End))_i, (Trp2\_x1(Start, End))_i\right], i, 1, DIM(Trp1\_x1(Start, End))\right)$$

Einheiten x2-Achse – **Units on x2-axis**

#18: $\text{Trp1\_x2(Start, End)} := \text{VECTOR}\left(\left[x + \text{Step}, -\frac{1}{4}\cdot x + 4\cdot\text{Step}\right], x, 2\cdot\text{Start}, 2\cdot\text{End}, 2\right)$

#19: $\text{Trp2\_x2(Start, End)} := \text{VECTOR}\left(\left[x - \text{Step}, -\frac{1}{4}\cdot x - 4\cdot\text{Step}\right], x, 2\cdot\text{Start}, 2\cdot\text{End}, 2\right)$

#20: $\text{TrUnits\_x2(Start, End)} := \text{VECTOR}\left(\left[(\text{Trp1\_x2(Start, End)})_i, (\text{Trp2\_x2(Start, End)})_i\right], i, 1, \text{DIM(Trp1\_x2(Start, End))}\right)$

Einheiten x3-Achse – **Units on x3-axis**

#21: Trp1_x3(Start, End) := VECTOR([0 – 3·Step, x], x, 2·Start, 2·End, 2)

#22: Trp2_x3(Start, End) := VECTOR([0 + 3·Step, x], x, 2·Start, 2·End, 2)

#23: $\text{TrUnits\_x3(Start, End)} := \text{VECTOR}\left(\left[(\text{Trp1\_x3(Start, End)})_i, (\text{Trp2\_x3(Start, End)})_i\right], i, 1, \text{DIM(Trp1\_x3(Start, End))}\right)$

#24: Tr_AxUnits(Start, End) := [TrUnits_x1(Start, End), TrUnits_x2(Start, End), TrUnits_x3(Start, End)]

#25: Tr_x1_Ax(Start, End) := ARROW(Tr_Pt([Start, 0, 0]), Tr_Pt([End, 0, 0]), 1)

#26: Tr_x2_Ax(Start, End) := ARROW(Tr_Pt([0, Start, 0]), Tr_Pt([0, End, 0]), 1)

#27: Tr_x3_Ax(Start, End) := ARROW(Tr_Pt([0, 0, Start]), Tr_Pt([0, 0, End]), 1)

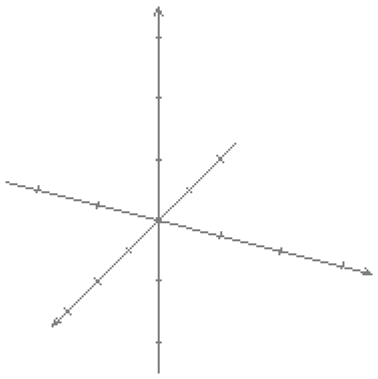
Die Achsen werden jeweils um eine halbe Einheit verlängert
Axes are extended that they will not end with a unit

#28: $$\text{Tr\_x1\_x2\_x3\_Axes(Start, End)} := \left[\begin{array}{l} \text{Tr\_x1\_Ax}\left(\text{Start} - \frac{1}{2}, \text{End} + \frac{1}{2}\right) \\ \text{Tr\_x2\_Ax}\left(\text{Start} - \frac{1}{2}, \text{End} + \frac{1}{2}\right) \\ \text{Tr\_x3\_Ax}\left(\text{Start} - \frac{1}{2}, \text{End} + \frac{1}{2}\right) \end{array}\right]$$

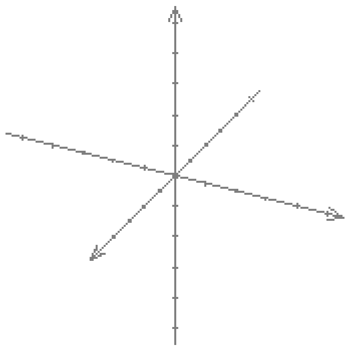
#29: $$\text{Tr(Start, End)} := \left[\begin{array}{c} \text{Tr\_x1\_x2\_x3\_Axes(Start, End)} \\ \text{Tr\_AxUnits(Start, End)} \end{array}\right]$$

Bild zweier Koordinatensysteme in Trimetrie (versch. Achsenlängen):
Plot of two systems of axes of coordinates in trimetric projection:

```
#30:   Tr(-2, 3)
```



```
#31:   l ≔ 1

#32:   Tr(-5, 5)
```

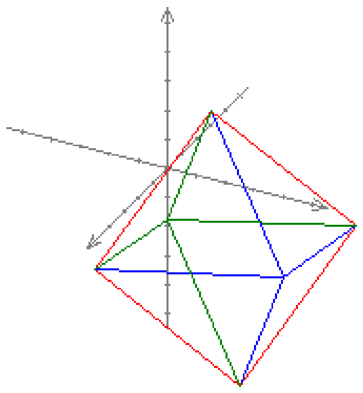


Ein Oktaeder aus den Punkten von oben
Projection of an octahedron using the points from above

```
#33:   Tr_Segm(rU, rV) ≔ [Tr_Pt(rU), Tr_Pt(rV)]
```

$$\#34:\quad \left[\begin{bmatrix} \text{Tr\_Segm(rE, rF)} \\ \text{Tr\_Segm(rF, rG)} \\ \text{Tr\_Segm(rG, rH)} \\ \text{Tr\_Segm(rH, rE)} \end{bmatrix}, \begin{bmatrix} \text{Tr\_Segm(rS1, rE)} \\ \text{Tr\_Segm(rS1, rF)} \\ \text{Tr\_Segm(rS1, rG)} \\ \text{Tr\_Segm(rS1, rH)} \end{bmatrix}, \begin{bmatrix} \text{Tr\_Segm(rS2, rE)} \\ \text{Tr\_Segm(rS2, rF)} \\ \text{Tr\_Segm(rS2, rG)} \\ \text{Tr\_Segm(rS2, rH)} \end{bmatrix}\right]$$

Highlight and plot one component of #34 after the other and change the color.

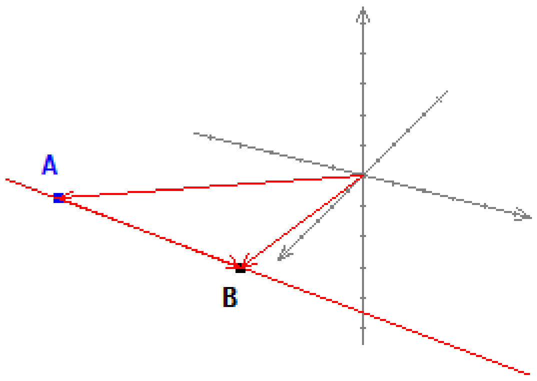


I skip presenting some calculations in the file which we do not need now.

```
**************************************************

***    Zweites Beispiel - Second example      ***
***    Grundaufgaben - Basic Problems         ***
**************************************************

#69:   Tr_Pos_Vect(rV) ≔ ARROW([0, 0], Tr_Pt(rV), l)

#70:   Tr_Vect(rU, rV) ≔ ARROW(Tr_Pt(rU), Tr_Pt(rV), l)
```

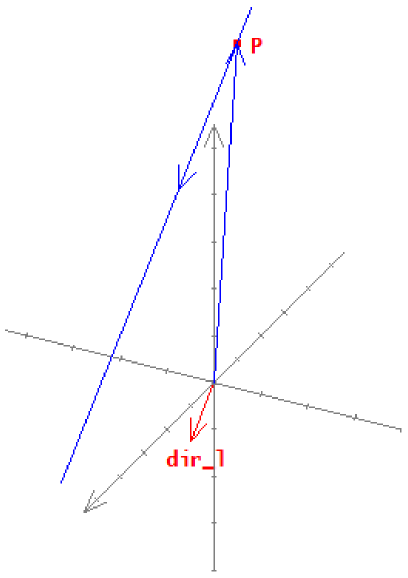
Räumliche Darstellung der durch zwei Punkte gegebenen Geraden
3D-Presentation of a straight line given by two points

```
#71:   Tr_collinear(rU, rV) := [Tr_Pt(rU + 3·(rV - rU)), Tr_Pt(rU - 3·(rV - rU))]

#72:   Tr_l_2Pt(rU, rV) := [[Tr_Pos_Vect(rU)], [Tr_Pos_Vect(rV)], [Tr_Vect(rU,
         rV)], Tr_collinear(rU, rV)]

#73:   [Tr_Pt(rA), Tr_Pt(rB)]

#74:   APPROX(Tr_l_2Pt(rA, rB))
```



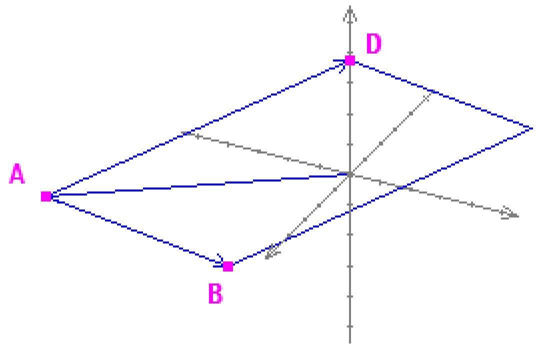
Räumliche Darstellung der durch einen Punkt und die Richtung gegebenen Geraden
3D-Presentation of a straight line given by one point and the direction vector

```
#75:   SB3_g_PR(rU, v) := [[SB3_ORTS_VEKT(rU)], [SB3_VEKT(rU, rU + v)],
         SB3_KOLLINEAR(rU, rU + v)]

#76:   Tr_l_PD(rU, v) := [[Tr_Pos_Vect(rU)], [Tr_Vect(rU, rU + v)],
         Tr_collinear(rU, rU + v)]

#77:   [Tr_Pt(rP), Tr_Pos_Vect(dir_l)]

#78:   Tr_l_PD(rP, 2.5·dir_l)
```



Räumliche Darstellung der durch drei Punkte gegebenen Ebene
3D-Presentation of a plane line given by three points

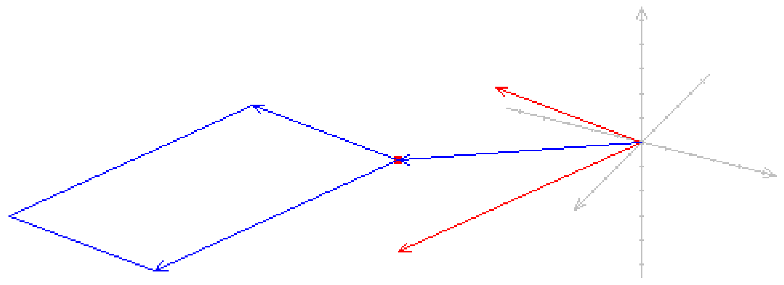
```
#79:   Tr_P_3Pts(rU, rV, rW) := [Tr_Pos_Vect(rU), Tr_Vect(rU, rV), Tr_Vect(rU,

          rW), Tr_Segm(rW, rW + rV - rU), Tr_Segm(rV, rW + rV - rU)]


#80:   Tr(-5, 5)

#81:   [Tr_Pt(rA), Tr_Pt(rB), Tr_Pt(rD)]

#82:   APPROX(Tr_P_3Pts(rA, rB, rD))
```



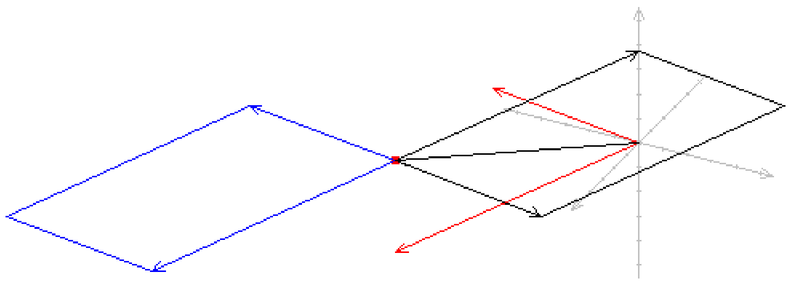
Räumliche Darstellung der durch einen Punkt und zwei Richtungsanteile gegebenen Ebene
3D-Presentation of a plane given by one point and two direction vectors

```
#83:   Tr_P_Puv(rU, u, v) := [Tr_Pos_Vect(rU), Tr_Vect(rU, rU + u), Tr_Vect(rU, rU

          + v), Tr_Segm(rU + v, (rU + v) + (rU + u) - rU), Tr_Segm(rU + u, (rU +

          v) + (rU + u) - rU)]


#84:   [dir1 := rB - rC, dir2 := rD - rC]

#85:   Tr(-5, 5)

#86:   Tr_Pt(rA)

#87:   [Tr_Pos_Vect(dir1), Tr_Pos_Vect(dir2)]

#88:   APPROX(Tr_P_Puv(rA, dir1, dir2))
```

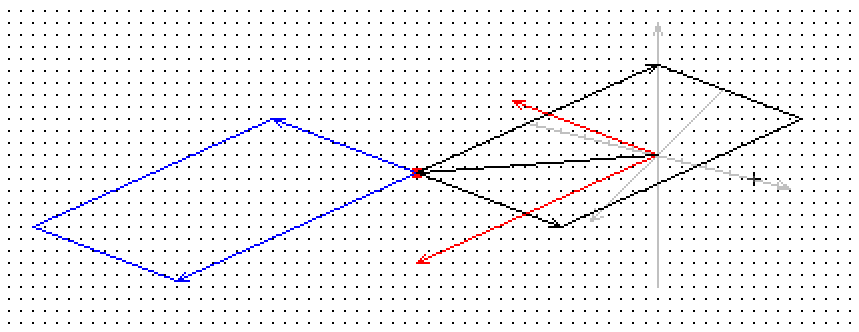
Plot #85 (the axes), #86 (position vector of point A), #87 (the two direction vectors) and finally #88 to get an impression of the plane.



It is clear that this must be the same plane as this one given by #82. We will demonstrate this by adding its plot:

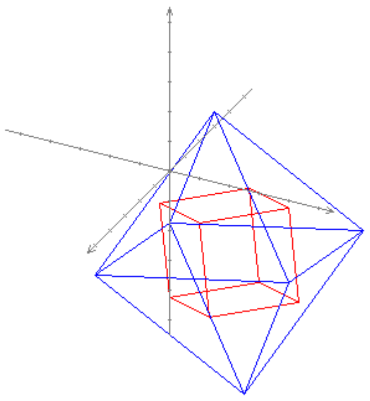


How about adding a grid, simulating a grid paper?



I skip again printing a lot of calculations producing the vertices of the solids, Josef

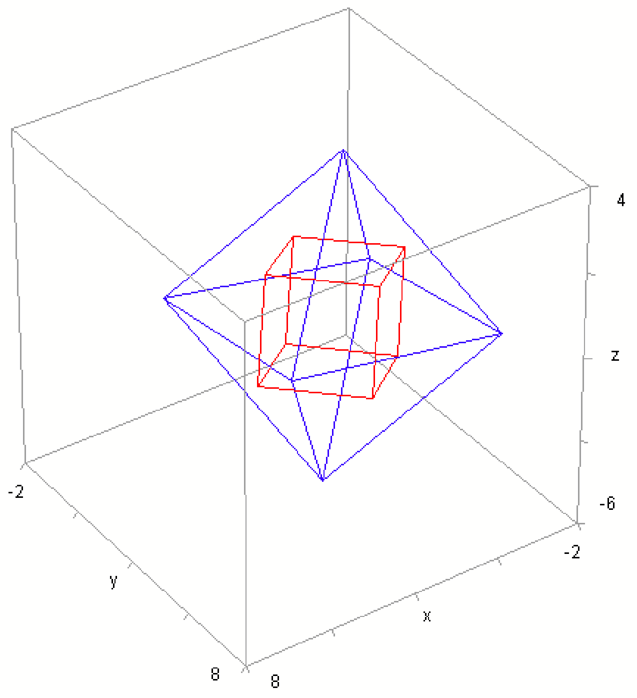
```
Oktaeder – Octahedron

        ⎡⎡ Tr_Segm(rE, rF) ⎤  ⎡ Tr_Segm(rS1, rE) ⎤  ⎡ Tr_Segm(rS2, rE) ⎤⎤
        ⎢⎢                 ⎥  ⎢                  ⎥  ⎢                  ⎥⎥
        ⎢⎢ Tr_Segm(rF, rG) ⎥  ⎢ Tr_Segm(rS1, rF) ⎥  ⎢ Tr_Segm(rS2, rF) ⎥⎥
#109:   ⎢⎢                 ⎥, ⎢                  ⎥, ⎢                  ⎥⎥
        ⎢⎢ Tr_Segm(rG, rH) ⎥  ⎢ Tr_Segm(rS1, rG) ⎥  ⎢ Tr_Segm(rS2, rG) ⎥⎥
        ⎢⎢                 ⎥  ⎢                  ⎥  ⎢                  ⎥⎥
        ⎣⎣ Tr_Segm(rH, rE) ⎦  ⎣ Tr_Segm(rS1, rH) ⎦  ⎣ Tr_Segm(rS2, rH) ⎦⎦

Innenquader – inscribed cuboid

#110: l := 0.5

#111: Tr(-5, 5)

        ⎡⎡ Tr_Segm(rM1, rM2) ⎤  ⎡ Tr_Segm(rM5, rM6) ⎤  ⎡ Tr_Segm(rM1, rM5) ⎤⎤
        ⎢⎢                   ⎥  ⎢                   ⎥  ⎢                   ⎥⎥
        ⎢⎢ Tr_Segm(rM2, rM3) ⎥  ⎢ Tr_Segm(rM6, rM7) ⎥  ⎢ Tr_Segm(rM2, rM6) ⎥⎥
#112:   ⎢⎢                   ⎥, ⎢                   ⎥, ⎢                   ⎥⎥
        ⎢⎢ Tr_Segm(rM3, rM4) ⎥  ⎢ Tr_Segm(rM7, rM8) ⎥  ⎢ Tr_Segm(rM3, rM7) ⎥⎥
        ⎢⎢                   ⎥  ⎢                   ⎥  ⎢                   ⎥⎥
        ⎣⎣ Tr_Segm(rM4, rM1) ⎦  ⎣ Tr_Segm(rM8, rM5) ⎦  ⎣ Tr_Segm(rM4, rM8) ⎦⎦
```



Now, working with DERIVE 6 we can easily plot the solids directly in the 3D-Plot-window.

$$\#113:\quad \left[\begin{bmatrix}[rE, rF]\\ [rF, rG]\\ [rG, rH]\\ [rH, rE]\end{bmatrix}, \begin{bmatrix}[rS1, rE]\\ [rS1, rF]\\ [rS1, rG]\\ [rS1, rH]\end{bmatrix}, \begin{bmatrix}[rS2, rE]\\ [rS2, rF]\\ [rS2, rG]\\ [rS2, rH]\end{bmatrix}\right]$$

$$\#114:\quad \left[\begin{bmatrix}[rM1, rM2]\\ [rM2, rM3]\\ [rM3, rM4]\\ [rM4, rM1]\end{bmatrix}, \begin{bmatrix}[rM5, rM6]\\ [rM6, rM7]\\ [rM7, rM8]\\ [rM8, rM5]\end{bmatrix}, \begin{bmatrix}[rM1, rM5]\\ [rM2, rM6]\\ [rM3, rM7]\\ [rM4, rM8]\end{bmatrix}\right]$$



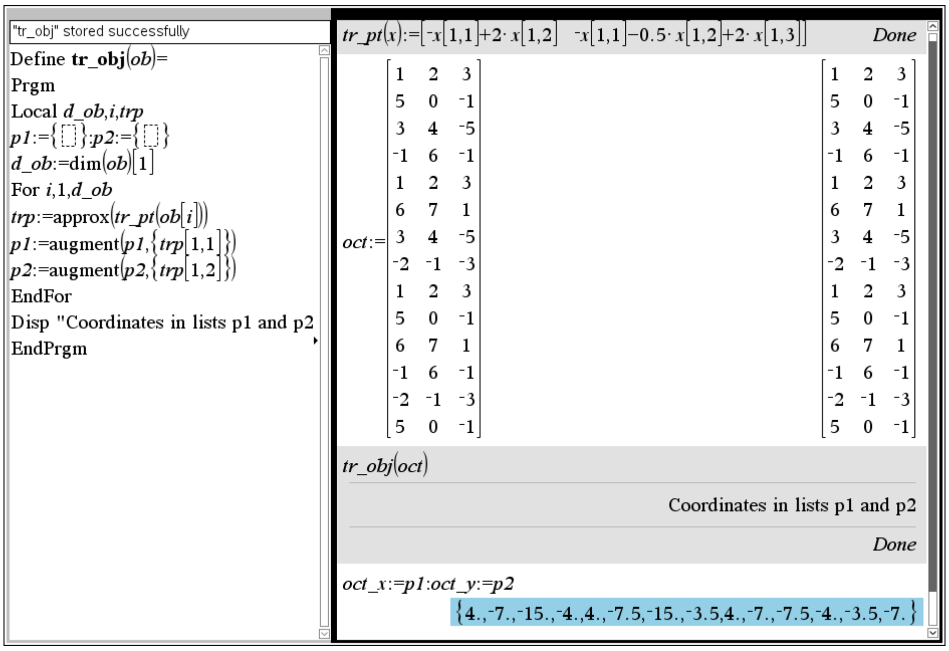
Unfortunately, we cannot present discrete points or segments or solids built of several edges in TI-Nspire's 3D-graphing window. I don't know why the Nspire-programmers have not added this feature until know? Does anybody of you know a trick how to overcome this deficiency? Or, do I miss something in the manual? Maybe that there will be a version 5?

Anyhow, in this case we must do like Peter does – or did: produce a projection of the 3D-object onto the 2D-plane in order to plot the mapping of the object using a scatter diagram with lists for the *x*- and *y*-coordinates.

You can follow the short programs given in the screen shots on the next page.

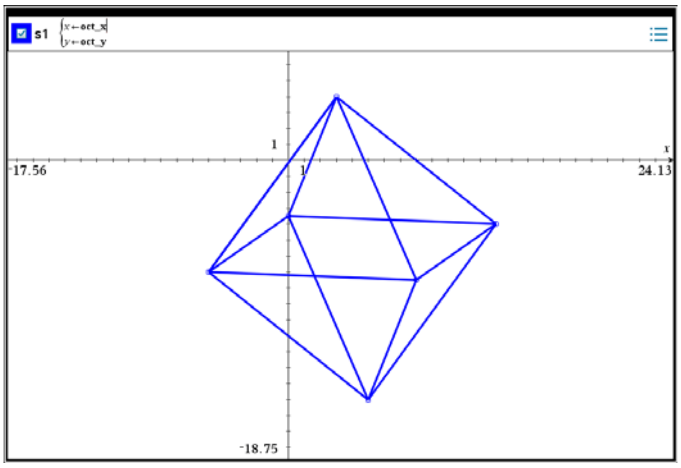
I define the octahedron from above as a list of its vertices and store them in a matrix:

E-F-G-H-E-S1-G-S2-E-F-S1-H-S2-F. Here we go … (Josef)

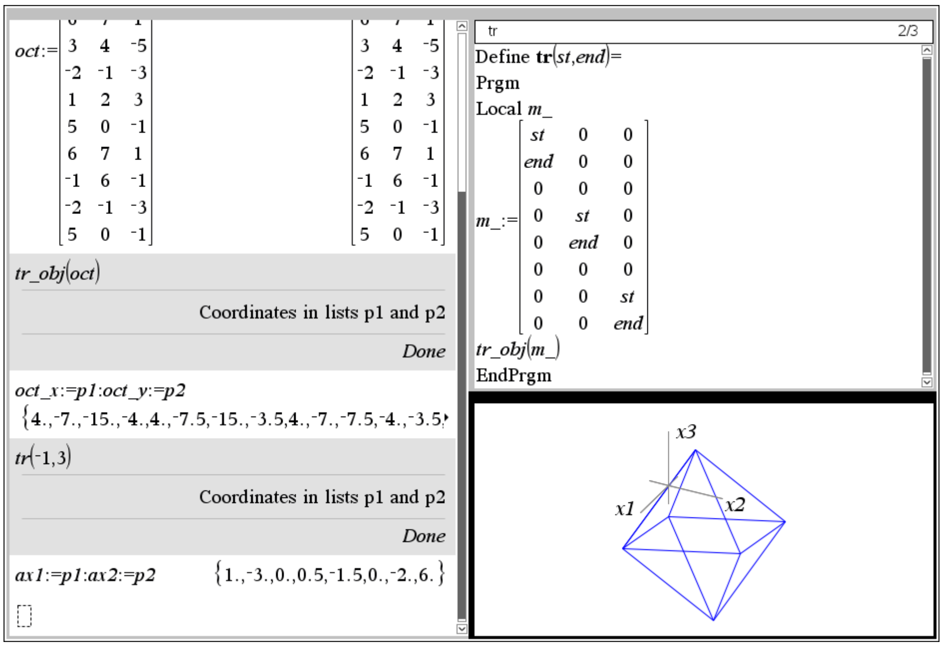


```
"tr_obj" stored successfully
Define tr_obj(ob)=
Prgm
Local d_ob,i,trp
p1:={⬚}:p2:={⬚}
d_ob:=dim(ob)[1]
For i,1,d_ob
trp:=approx(tr_pt(ob[i]))
p1:=augment(p1,{trp[1,1]})
p2:=augment(p2,{trp[1,2]})
EndFor
Disp "Coordinates in lists p1 and p2
EndPrgm
```

```
tr_pt(x):=[⁻x[1,1]+2·x[1,2]   ⁻x[1,1]−0.5·x[1,2]+2·x[1,3]]     Done

        [ 1   2   3]                       [ 1   2   3]
        [ 5   0  ⁻1]                       [ 5   0  ⁻1]
        [ 3   4  ⁻5]                       [ 3   4  ⁻5]
        [⁻1   6  ⁻1]                       [⁻1   6  ⁻1]
        [ 1   2   3]                       [ 1   2   3]
        [ 6   7   1]                       [ 6   7   1]
oct:=   [ 3   4  ⁻5]                       [ 3   4  ⁻5]
        [⁻2  ⁻1  ⁻3]                       [⁻2  ⁻1  ⁻3]
        [ 1   2   3]                       [ 1   2   3]
        [ 5   0  ⁻1]                       [ 5   0  ⁻1]
        [ 6   7   1]                       [ 6   7   1]
        [⁻1   6  ⁻1]                       [⁻1   6  ⁻1]
        [⁻2  ⁻1  ⁻3]                       [⁻2  ⁻1  ⁻3]
        [ 5   0  ⁻1]                       [ 5   0  ⁻1]

tr_obj(oct)
                                 Coordinates in lists p1 and p2
                                                           Done
oct_x:=p1:oct_y:=p2
      {4.,⁻7.,⁻15.,⁻4.,4.,⁻7.5,⁻15.,⁻3.5,4.,⁻7.,⁻7.5,⁻4.,⁻3.5,⁻7.}
```



Want to show the axes, too? No Problem ...



```
        [ 3   4  ⁻5]                       [ 3   4  ⁻5]
oct:=   [⁻2  ⁻1  ⁻3]                       [⁻2  ⁻1  ⁻3]
        [ 1   2   3]                       [ 1   2   3]
        [ 5   0  ⁻1]                       [ 5   0  ⁻1]
        [ 6   7   1]                       [ 6   7   1]
        [⁻1   6  ⁻1]                       [⁻1   6  ⁻1]
        [⁻2  ⁻1  ⁻3]                       [⁻2  ⁻1  ⁻3]
        [ 5   0  ⁻1]                       [ 5   0  ⁻1]

tr_obj(oct)
                                 Coordinates in lists p1 and p2
                                                           Done
oct_x:=p1:oct_y:=p2
 {4.,⁻7.,⁻15.,⁻4.,4.,⁻7.5,⁻15.,⁻3.5,4.,⁻7.,⁻7.5,⁻4.,⁻3.5▸
tr(⁻1,3)
                                 Coordinates in lists p1 and p2
                                                           Done
ax1:=p1:ax2:=p2          {1.,⁻3.,0.,0.5,⁻1.5,0.,⁻2.,6.}
⬚
```

```
tr                                                        2/3
Define tr(st,end)=
Prgm
Local m_
      [ st    0    0 ]
      [ end   0    0 ]
      [ 0     0    0 ]
m_:=  [ 0     st   0 ]
      [ 0     end  0 ]
      [ 0     0    0 ]
      [ 0     0    st]
      [ 0     0    end]
tr_obj(m_)
EndPrgm
```

Lagrange Multipliers

Don Phillips, USA

Lagrange multipliers can be used to find the extrema of a function of several variables when the variables are restricted in some manner. The function lagrangemulti(function,constraint(s))can be used to find the extrema. The function is entered as an expression; the constraint(s) can be entered as an equation, if there is only one constraint, or in a list or a (n by 1) matrix if 2 or more constraints. The output is a matrix of the values of the variables and the function at those values. See below for examples.

Example 1: Find the extrema of $f(x,y) = x\cdot y$ if (x,y) is restricted to the ellipse $4x^2+y^2-4=0$.

$$\text{lagrangemulti}\left(x\cdot y,4\cdot x^2+y^2-4=0\right) \blacktriangleright \begin{bmatrix} \text{"Answers"} & \text{"Value"} \\ x=\frac{-\sqrt{2}}{2} \text{ and } y=-\sqrt{2} \text{ and } \lambda=\frac{1}{4} & 1 \\ x=\frac{-\sqrt{2}}{2} \text{ and } y=\sqrt{2} \text{ and } \lambda=\frac{-1}{4} & -1 \\ x=\frac{\sqrt{2}}{2} \text{ and } y=-\sqrt{2} \text{ and } \lambda=\frac{-1}{4} & -1 \\ x=\frac{\sqrt{2}}{2} \text{ and } y=\sqrt{2} \text{ and } \lambda=\frac{1}{4} & 1 \end{bmatrix}$$

The equation of the ellipse could also be entered as $4x^2+y^2=4$ or as $x^2+\frac{y^2}{4}=1$. Either way will get the same answers.

Example 2: Let C denote the first-octant arc of the curve in which the paraboloid $x^2+y^2+2z=16$ and the plane $x+y=4$ intersect. Find the points on C that are closest to and farthest from the origin. Find the minimum and maximum distances from the origin to C. The largest and smallest values of distance $d(0,P) = \sqrt{x^2+y^2+z^2}$ must be found.

$$\text{lagrangemulti}\left(\sqrt{x^2+y^2+z^2},\begin{bmatrix} x^2+y^2+2\cdot z=16 \\ x+y=4 \end{bmatrix}\right)$$

$$\blacktriangleright \begin{bmatrix} \text{"Answers"} & \text{"Value"} \\ x=-(\sqrt{3}-2) \text{ and } y=\sqrt{3}+2 \text{ and } z=1 \text{ and } \lambda 1=\frac{\sqrt{15}}{30} \text{ and } \lambda 2=0 & \sqrt{15} \\ x=2 \text{ and } y=2 \text{ and } z=4 \text{ and } \lambda 1=\frac{\sqrt{6}}{6} \text{ and } \lambda 2=\frac{-\sqrt{6}}{2} & 2\cdot\sqrt{6} \\ x=\sqrt{3}+2 \text{ and } y=-(\sqrt{3}-2) \text{ and } z=1 \text{ and } \lambda 1=\frac{\sqrt{15}}{30} \text{ and } \lambda 2=0 & \sqrt{15} \end{bmatrix}$$

The minimum value, $\sqrt{15}$, occurs at two points and the maximum value of$2\cdot\sqrt{6}$ at one point.

Example 3: Find the volume of the largest rectangular box with faces parallel to the coordinate planes that can be inscribes in the ellipsoid $16x^2+4y^2+9z^2=144$. In the first octant the volume will be $x\cdot y\cdot z$. In the 8 octants of the ellipsoid the volume will therefore be $8\cdot x\cdot y\cdot z$.

$$\textbf{lagrangemulti}\left(8\cdot x\cdot y\cdot z, 16\cdot x^2+4\cdot y^2+9\cdot z^2=144\right)$$

$$\begin{bmatrix}
\text{"Answers"} & \text{"Value"} \\
x=-3 \text{ and } y=0 \text{ and } z=0 \text{ and } \lambda=0 & 0 \\
x=-\sqrt{3} \text{ and } y=-2\cdot\sqrt{3} \text{ and } z=\frac{-4\cdot\sqrt{3}}{3} \text{ and } \lambda=\frac{-2\cdot\sqrt{3}}{3} & -64\cdot\sqrt{3} \\
x=-\sqrt{3} \text{ and } y=-2\cdot\sqrt{3} \text{ and } z=\frac{4\cdot\sqrt{3}}{3} \text{ and } \lambda=\frac{2\cdot\sqrt{3}}{3} & 64\cdot\sqrt{3} \\
x=-\sqrt{3} \text{ and } y=2\cdot\sqrt{3} \text{ and } z=\frac{-4\cdot\sqrt{3}}{3} \text{ and } \lambda=\frac{2\cdot\sqrt{3}}{3} & 64\cdot\sqrt{3} \\
x=-\sqrt{3} \text{ and } y=2\cdot\sqrt{3} \text{ and } z=\frac{4\cdot\sqrt{3}}{3} \text{ and } \lambda=\frac{-2\cdot\sqrt{3}}{3} & -64\cdot\sqrt{3} \\
x=0 \text{ and } y=-6 \text{ and } z=0 \text{ and } \lambda=0 & 0 \\
x=0 \text{ and } y=0 \text{ and } z=-4 \text{ and } \lambda=0 & 0 \\
x=0 \text{ and } y=0 \text{ and } z=4 \text{ and } \lambda=0 & 0 \\
x=0 \text{ and } y=6 \text{ and } z=0 \text{ and } \lambda=0 & 0 \\
x=\sqrt{3} \text{ and } y=-2\cdot\sqrt{3} \text{ and } z=\frac{-4\cdot\sqrt{3}}{3} \text{ and } \lambda=\frac{2\cdot\sqrt{3}}{3} & 64\cdot\sqrt{3} \\
x=\sqrt{3} \text{ and } y=-2\cdot\sqrt{3} \text{ and } z=\frac{4\cdot\sqrt{3}}{3} \text{ and } \lambda=\frac{-2\cdot\sqrt{3}}{3} & -64\cdot\sqrt{3} \\
x=\sqrt{3} \text{ and } y=2\cdot\sqrt{3} \text{ and } z=\frac{-4\cdot\sqrt{3}}{3} \text{ and } \lambda=\frac{-2\cdot\sqrt{3}}{3} & -64\cdot\sqrt{3} \\
x=\sqrt{3} \text{ and } y=2\cdot\sqrt{3} \text{ and } z=\frac{4\cdot\sqrt{3}}{3} \text{ and } \lambda=\frac{2\cdot\sqrt{3}}{3} & 64\cdot\sqrt{3} \\
x=3 \text{ and } y=0 \text{ and } z=0 \text{ and } \lambda=0 & 0
\end{bmatrix}$$

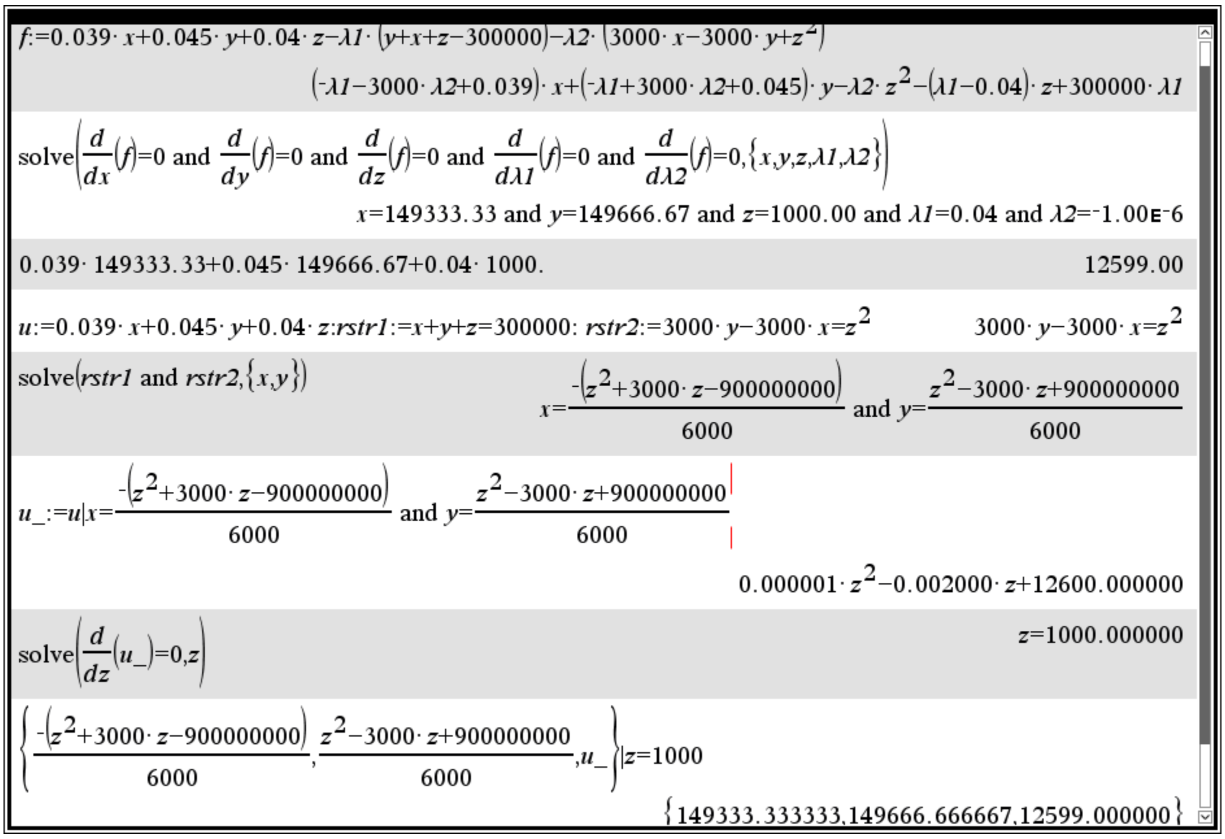
I - Josef - tried another example (from *Larson & Edwards, Calculus*):

An investor intends to invest € 300000 in three different stocks with different annual dividends: x is expected to earn 3.9%, y 4.5% and z 4.0%.
The amount invested in z must follow the condition: $3000x - 3000y + z^2 = 0$.
How much should be invested in each stock to yield a maximum of dividends?

$$\textbf{lagrangemulti}\left(0.039\cdot x+0.045\cdot y+0.04\cdot z, \begin{bmatrix} y+x+z=300000 \\ 3000\cdot x-3000\cdot y+z^2=0 \end{bmatrix}\right)$$

$$\begin{bmatrix}
\text{"Answers"} & \text{"Value"} \\
x=149333. \text{ and } y=149667. \text{ and } z=1000. \text{ and } \lambda 1=0.042 \text{ and } \lambda 2=-0.000001 & 12599.
\end{bmatrix}$$

I solve the problem on a Calculus page in two traditional forms (with Lagrange multipliers and without):



$$f:=0.039\cdot x+0.045\cdot y+0.04\cdot z-\lambda 1\cdot(y+x+z-300000)-\lambda 2\cdot(3000\cdot x-3000\cdot y+z^2)$$

$$(-\lambda 1-3000\cdot\lambda 2+0.039)\cdot x+(-\lambda 1+3000\cdot\lambda 2+0.045)\cdot y-\lambda 2\cdot z^2-(\lambda 1-0.04)\cdot z+300000\cdot\lambda 1$$

$$\text{solve}\left(\frac{d}{dx}(f)=0 \text{ and } \frac{d}{dy}(f)=0 \text{ and } \frac{d}{dz}(f)=0 \text{ and } \frac{d}{d\lambda 1}(f)=0 \text{ and } \frac{d}{d\lambda 2}(f)=0,\{x,y,z,\lambda 1,\lambda 2\}\right)$$

$$x=149333.33 \text{ and } y=149666.67 \text{ and } z=1000.00 \text{ and } \lambda 1=0.04 \text{ and } \lambda 2=^-1.00\text{E}^-6$$

$$0.039\cdot 149333.33+0.045\cdot 149666.67+0.04\cdot 1000. \qquad 12599.00$$

$$u:=0.039\cdot x+0.045\cdot y+0.04\cdot z:rstr1:=x+y+z=300000: rstr2:=3000\cdot y-3000\cdot x=z^2 \qquad 3000\cdot y-3000\cdot x=z^2$$

$$\text{solve}(rstr1 \text{ and } rstr2,\{x,y\}) \qquad x=\frac{-(z^2+3000\cdot z-900000000)}{6000} \text{ and } y=\frac{z^2-3000\cdot z+900000000}{6000}$$

$$u\_:=u|x=\frac{-(z^2+3000\cdot z-900000000)}{6000} \text{ and } y=\frac{z^2-3000\cdot z+900000000}{6000}$$

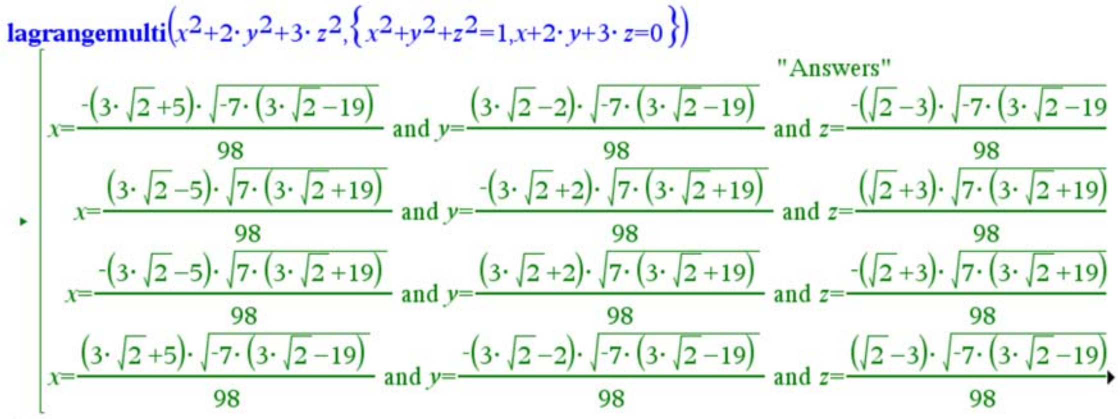
$$0.000001\cdot z^2-0.002000\cdot z+12600.000000$$

$$\text{solve}\left(\frac{d}{dz}(u\_)=0,z\right) \qquad z=1000.000000$$

$$\left\{\frac{-(z^2+3000\cdot z-900000000)}{6000},\frac{z^2-3000\cdot z+900000000}{6000},u\_\right\}|z=1000$$

$$\{149333.333333,149666.666667,12599.000000\}$$

Just for fun I solved a problem from Günter/Kusmin: *Aufgabensammlung zur Höheren Mathematik*.



$$\text{lagrangemulti}(x^2+2\cdot y^2+3\cdot z^2,\{x^2+y^2+z^2=1,x+2\cdot y+3\cdot z=0\})$$

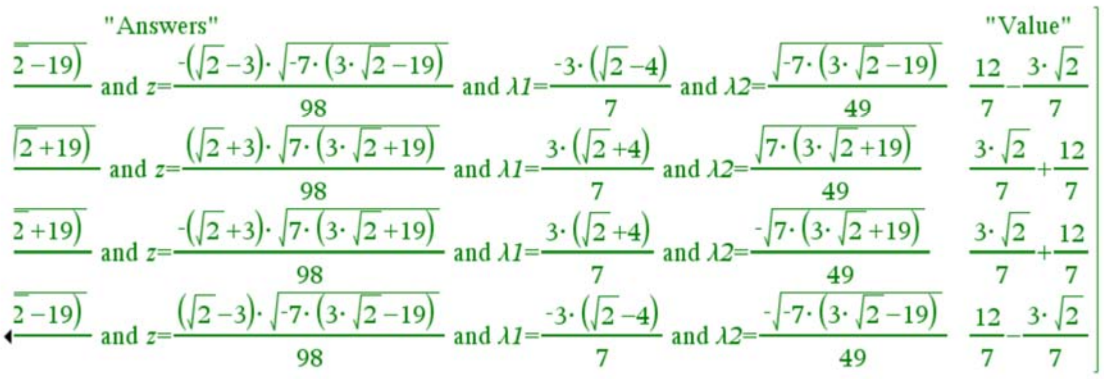
"Answers"

$$x=\frac{-(3\cdot\sqrt{2}+5)\cdot\sqrt{-7\cdot(3\cdot\sqrt{2}-19)}}{98} \text{ and } y=\frac{(3\cdot\sqrt{2}-2)\cdot\sqrt{-7\cdot(3\cdot\sqrt{2}-19)}}{98} \text{ and } z=\frac{-(\sqrt{2}-3)\cdot\sqrt{-7\cdot(3\cdot\sqrt{2}-19}}{98}$$

$$x=\frac{(3\cdot\sqrt{2}-5)\cdot\sqrt{7\cdot(3\cdot\sqrt{2}+19)}}{98} \text{ and } y=\frac{-(3\cdot\sqrt{2}+2)\cdot\sqrt{7\cdot(3\cdot\sqrt{2}+19)}}{98} \text{ and } z=\frac{(\sqrt{2}+3)\cdot\sqrt{7\cdot(3\cdot\sqrt{2}+19)}}{98}$$

$$x=\frac{-(3\cdot\sqrt{2}-5)\cdot\sqrt{7\cdot(3\cdot\sqrt{2}+19)}}{98} \text{ and } y=\frac{(3\cdot\sqrt{2}+2)\cdot\sqrt{7\cdot(3\cdot\sqrt{2}+19)}}{98} \text{ and } z=\frac{-(\sqrt{2}+3)\cdot\sqrt{7\cdot(3\cdot\sqrt{2}+19)}}{98}$$

$$x=\frac{(3\cdot\sqrt{2}+5)\cdot\sqrt{-7\cdot(3\cdot\sqrt{2}-19)}}{98} \text{ and } y=\frac{-(3\cdot\sqrt{2}-2)\cdot\sqrt{-7\cdot(3\cdot\sqrt{2}-19)}}{98} \text{ and } z=\frac{(\sqrt{2}-3)\cdot\sqrt{-7\cdot(3\cdot\sqrt{2}-19)}}{98}$$



"Answers" "Value"

$$2-19) \text{ and } z=\frac{-(\sqrt{2}-3)\cdot\sqrt{-7\cdot(3\cdot\sqrt{2}-19)}}{98} \text{ and } \lambda 1=\frac{-3\cdot(\sqrt{2}-4)}{7} \text{ and } \lambda 2=\frac{\sqrt{-7\cdot(3\cdot\sqrt{2}-19)}}{49} \qquad \frac{12}{7}-\frac{3\cdot\sqrt{2}}{7}$$

$$2+19) \text{ and } z=\frac{(\sqrt{2}+3)\cdot\sqrt{7\cdot(3\cdot\sqrt{2}+19)}}{98} \text{ and } \lambda 1=\frac{3\cdot(\sqrt{2}+4)}{7} \text{ and } \lambda 2=\frac{\sqrt{7\cdot(3\cdot\sqrt{2}+19)}}{49} \qquad \frac{3\cdot\sqrt{2}}{7}+\frac{12}{7}$$

$$2+19) \text{ and } z=\frac{-(\sqrt{2}+3)\cdot\sqrt{7\cdot(3\cdot\sqrt{2}+19)}}{98} \text{ and } \lambda 1=\frac{3\cdot(\sqrt{2}+4)}{7} \text{ and } \lambda 2=\frac{-\sqrt{7\cdot(3\cdot\sqrt{2}+19)}}{49} \qquad \frac{3\cdot\sqrt{2}}{7}+\frac{12}{7}$$

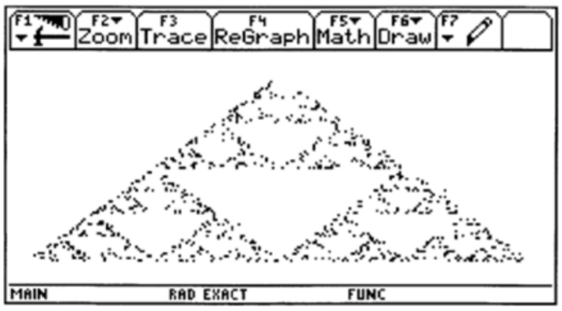
$$2-19) \text{ and } z=\frac{(\sqrt{2}-3)\cdot\sqrt{-7\cdot(3\cdot\sqrt{2}-19)}}{98} \text{ and } \lambda 1=\frac{-3\cdot(\sqrt{2}-4)}{7} \text{ and } \lambda 2=\frac{-\sqrt{-7\cdot(3\cdot\sqrt{2}-19)}}{49} \qquad \frac{12}{7}-\frac{3\cdot\sqrt{2}}{7}$$

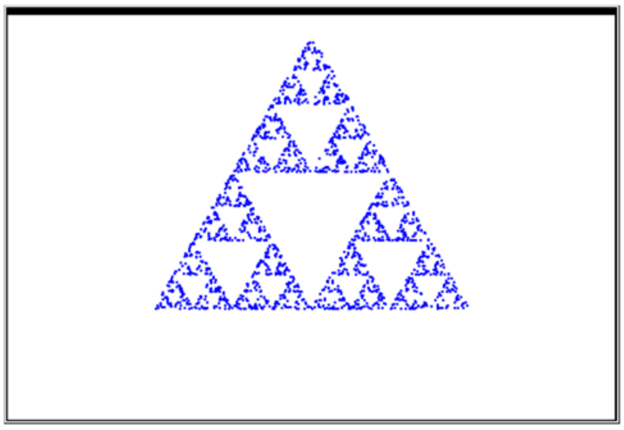
The DERIVE output is muchmore compact. See LagrangeMulti.dfw, Josef.

CHAOS-Game in Space

Josef Böhm

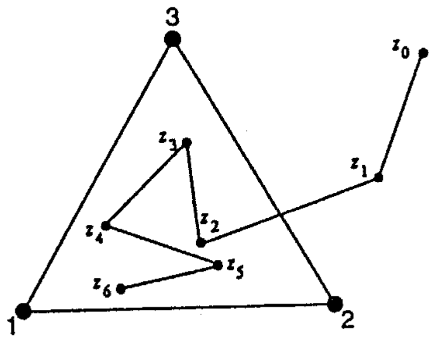
When you see the pictures below you might possibly remember earlier DNLs: in DNL #22 I presented the CHAOS-Game as a program for TI-92 (Larry Gilligan). The revised version of DNL#34 provides a TI-Nspire version of it together with Maria Koth's CHAOS polygons.





Reminder:

Start with an equilateral triangle with vertices v1, v2 and v3. Then pick any arbitrary point z0 anywhere in the plane. Now, randomly select one of the three vertices, say v2. The midpoint of the segment (v2,z0) gives the next point z1. Select - again randomly - one of the vertices, say v1 $\rightarrow$ midpoint of segment (v1,z1) gives z2, etc.

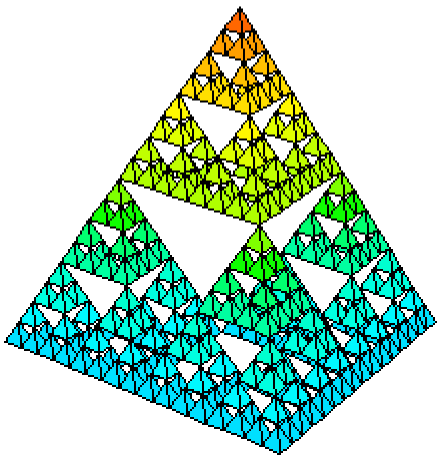


Heinz Rainer Geyer and his students plotted a three-dimensional Sierpinski-tetrahedron.

You can find it in DNL#34.

This is not a "chaotic" figure. All vertices are well determined.

(File is contained in MTH112.zip.)

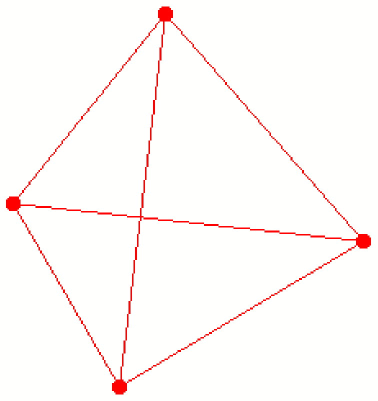


I had the idea to transfer the CHAOS-Game into the third dimension: Take a regular tetrahedron instead of the triangle and then perform the same procedure.

So, let's go on!

The vertices of the tetrahedron are given in a matrix:

$$T := \begin{bmatrix} -\frac{3}{2} & -\frac{3}{6}\cdot\sqrt{3} & 0 \\ \frac{3}{2} & -\frac{3}{6}\cdot\sqrt{3} & 0 \\ 0 & \sqrt{3} & 0 \\ 0 & 0 & \sqrt{6} \end{bmatrix}$$

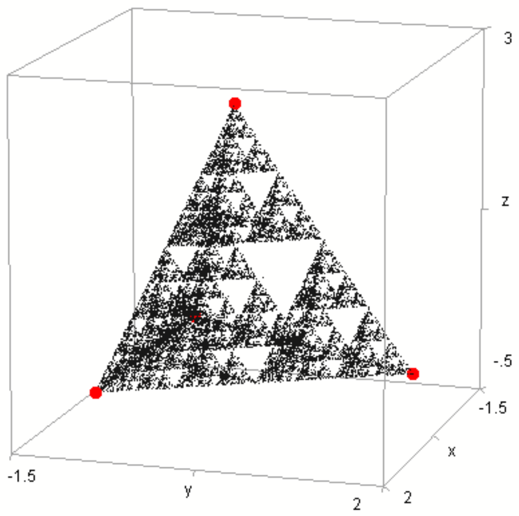


The *DERIVE*-program:

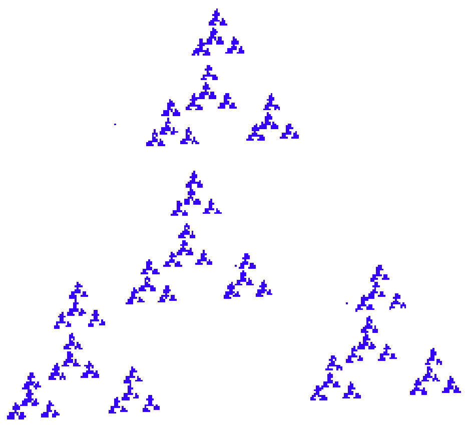
```
s_t(n, pr := 0, k := 0.5, dummy, i, xn, r, plist) :=
  Prog
    dummy := RANDOM(0)
    xn := [4·RANDOM(1) - 2, 4·RANDOM(1) - 2, 4·RANDOM(1) - 2]
    plist := IF(pr = 0, [[xn, xn]], [[xn]])
    i := 1
    Loop
      If i > n exit
      r := RANDOM(4) + 1
      xn := xn + k·(T↓r - xn)
      plist := IF(pr = 0, APPEND(plist, [[[xn, xn]]]), APPEND(plist, [[xn]]))
      i :+ 1
    plist
```

The standard call `s_t(n)` gives a plot of *n* points with *k* = 0.5 which is the division ratio to obtain a new point x_{n+1} on the segment between old point x_n and randomly chosen vertex. `s_t(n,1)` gives also *n* points but you can choose the size of the points via Insert > Plot in the 3D-Plot Window. The last parameter changes the division ratio. Let's show some examples:

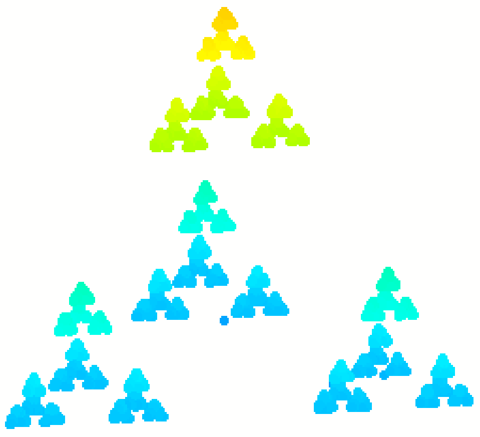
`s_t(20000)`

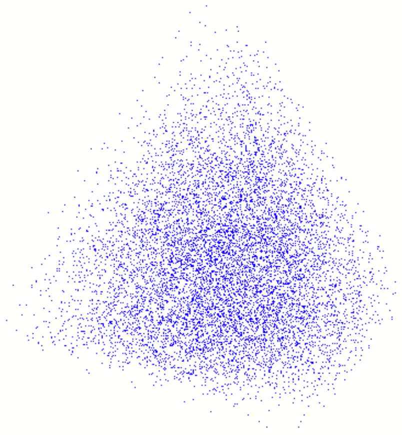


$\text{s\_t}\left(20000,\ 0,\ \frac{2}{3}\right)$



$s\_t\left(3000,\ 1,\ \dfrac{2}{3}\right)$ $s\_t\left(10000,\ 0,\ \dfrac{1}{4}\right)$





Then I asked myself: Why not try with a cube? (C = the matrix consisting of the eight vertices of the cube.)

```
s_c(n, k := 0.5, dummy, i, xn, r, plist) :=
  Prog
    dummy := RANDOM(0)
    xn := [4·RANDOM(1) - 2, 4·RANDOM(1) - 2, 4·RANDOM(1) - 2]
    plist := [[xn, xn]]
    i := 1
    Loop
      If i > n exit
      r := RANDOM(8) + 1
      xn := xn + k·(C↓r - xn)
      plist := APPEND(plist, [[[xn, xn]]])
      i :+ 1
    plist
```

After some experiments I found an appropriate value for *k*. Try to find other interesting *k*s!

s_c(10000)

$s\_c\left(20000,\ 0,\ \dfrac{1}{\sqrt{2}}\right)$