# THE DERIVE - NEWSLETTER #113

ISSN 1990-7079

THE BULLETIN OF THE

DERIVE®

USER GROUP

+ CAS-TI

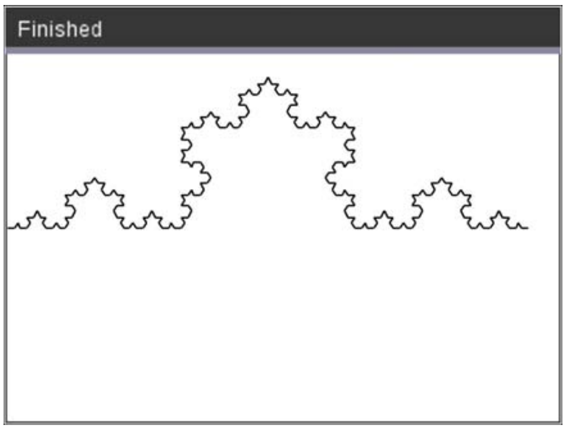
Contents:

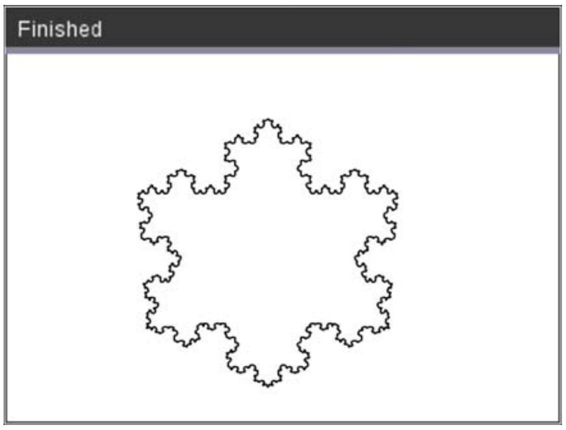
April 2019

## DUG-Mails from Germany (translated by me, Josef)

**From Hubert Langlotz, Germany**

Dear Josef,
A new Nspire-Version will available very soon. Then we will be able to reanimate our old TI-92 and V200-programs because graphing will be programable. I attach a small program of my colleague Tobias Kellner, which is similar to an old TI-92 program. But it runs only with the new software or a new calculator.





**From Julius Angres, Germany**

Dear Mr. Böhm,

I am math teacher at a vocational school in North Germany and have been working with DERIVE 6 with increasing pleasure. It was mere chance that I discovered the Derive User Group and its newsletter some weeks ago.

I‘d like to support the good cause and produced a small contribution to implement hyper operations in Derive. I attach the pdf-file together with the respective dfw-file.

If you will find it suitable, I would be pleased if you would publish it in one of the next issues of the newsletter.

I would appreciate a short feedback!

Regards

Julius. Angres

**From Wolfgang Alvermann, Germany**

Dear Josef,

*Georg Gläser* inspired me to my short article for school mathematics. You will decide if it is suitable for the DNL.

Warm regards to Austria from East Friesland, which is very rainy in these days.

Wolfgang

I answered and thanked them all, but it is not space enough to show my mails. The mentioned articles will be published in the next DNL. Josef

Dear DUG Members,

I hoped to be ready with DNL#113 in April to be not too late with it. Fortunately, I could make it. As I informed you earlier this issue was very laborious for me:

The anageo-programs needed a lot of programming and editing of the Lua-scripts. In addition, I wanted to provide an English and a German version. I am not quite sure if the programs are completely error free. So many special cases had to be considered and tested. At the other hand it is very likely that some parts of the programs could be done by a shorter code. All comments are welcome.

It is remarkable that this contribution is the result of a world wide cooperation between a German, an Australian, a Canadian and an Austrian DUG-member.

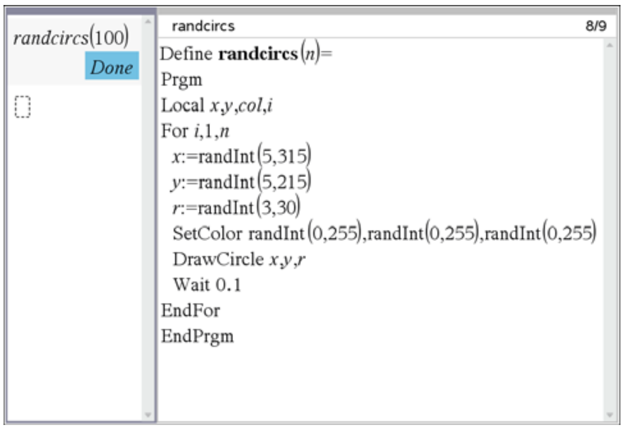
Believe it or not, even now in 2019 – many years after the official end of DERIVE – DUG is growing. And even more, new members provide contributions for our journal (see next page).

The TI-Nspire users may enjoy Don Phillips 2-stages Least Squares Regression program (the DERIVE version was published in DNL#65). Don sent a challenge for our members: the Penney Ante Game (page 35). You are invited to find out the winning strategy (without cheating by doing a web search ☺).

As you can learn from Hubert Langlotz' mail, a new version of TI-Nspire will be on the market in the next future. I had the opportunity to test it. See below a short program to produce and plot *n* random circles. You can watch how the screen fills with the randomly colored circles.

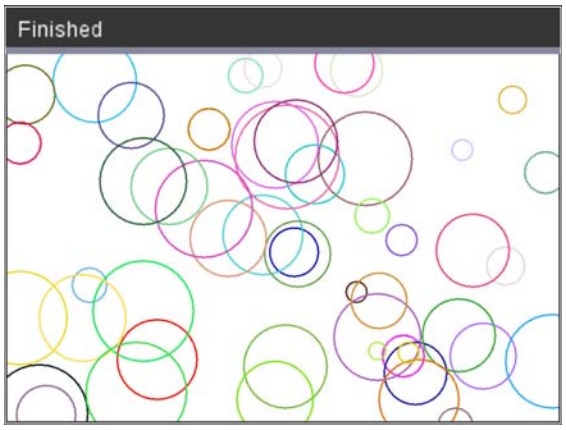
Best regards and wishes

Josef



```
randcircs(100)
                Done
```

```
randcircs                                              8/9
Define randcircs(n)=
Prgm
Local x,y,col,i
For i,1,n
  x:=randInt(5,315)
  y:=randInt(5,215)
  r:=randInt(3,30)
  SetColor randInt(0,255),randInt(0,255),randInt(0,255)
  DrawCircle x,y,r
  Wait 0.1
EndFor
EndPrgm
```



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: June 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, CZ
Tutorials for the NSpireCAS, G. Herweyers, BEL
Dirac Algebra, Clifford Algebra, Vector-Matrix-Extension, 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
The Penney-Ante Game, MacDonald Phillips, USA
Hyper operations for DERIVE, Julius Angres, Germany
Throwing range on an inclined slope, Wolfgang Alvermann, Germany
Paper & Pen vs CAS, W. Yancey, USA
Functional Derivative, Francisco M. Fernandez, ARG … and others

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

## Mail from Argentina

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

Dear Joseph,

I have found it useful to define a derivative of an expression with respect to a function in it. I am attaching a naive file with a few examples. I suspect that someone may have done it earlier and in a much more general and efficient way. In that case I would like to know about it.

Best regards,

Marcelo

#1: $\mathrm{v}(x) :=$

#2: $\mathrm{newder}(f,\ a,\ \xi) := \mathrm{SUBST}\left(\frac{d}{d\xi}\ \mathrm{SUBST}(f,\ a,\ \xi),\ \xi,\ a\right)$

#3: $\mathrm{newder}(\mathrm{SIN}(a\cdot \mathrm{v}(x)),\ \mathrm{v}(x)) = a\cdot \mathrm{COS}(a\cdot \mathrm{v}(x))$

#4: $\mathrm{newder}(\mathrm{EXP}(a\cdot \mathrm{v}(x)^2),\ \mathrm{v}(x)) = 2\cdot a\cdot e^{a\cdot \mathrm{v}(x)^2}\cdot \mathrm{v}(x)$

Dear Marcelo,

it is always great to receive a mail from Argentina, many thanks.
Your function is new for me and it will be a pleasure to post it in the next DNL.
Can you provide any application for newder()?

Best regards
Josef

Dear Josef,

I wrote it for the application of Lagrange and Hamilton equations of motion, though I am afraid I rarely do this kind of calculation. I am attaching a Derive file with one of the simplest examples: the Lagrangian equation of motion for the simple pendulum. I hope that it is sufficient for your needs; otherwise, just tell me and I will write a short note.

I did not bother to extend the definition to higher derivatives because I did not need them and because I am an extremely lazy and inefficient programmer

Best regards,

Marcelo

Dear Josef,

I think I could solve the problem with the definition of the functional derivative. It seems that *lim* works where *subst* does not. I am attaching a Derive file and I will write something later.

Best regards,

Marcelo

Simple pendulum.dfw

#1: $\text{newder}(f, a, \xi) := \text{SUBST}\left(\frac{d}{d\xi}\text{SUBST}(f, a, \xi), \xi, a\right)$

Simple pendulum of mass m, length l and amplitude ²(t), where t stands for time

#2: $\theta(t) :=$

Lagrangian function

#3: $\text{La} := \frac{m}{2}\cdot l^2\cdot\left(\frac{d}{dt}\theta(t)\right)^2 + m\cdot g\cdot l\cdot \text{COS}(\theta(t))$

Lagrangian equation of motion

#4: $\frac{d}{dt}\text{newder}(\text{La}, \theta'(t)) = \text{newder}(\text{La}, \theta(t))$

#5: $l^2\cdot m\cdot\theta''(t) = -g\cdot l\cdot m\cdot\text{SIN}(\theta(t))$

#6: $\frac{l^2\cdot m\cdot\theta''(t) = -g\cdot l\cdot m\cdot\text{SIN}(\theta(t))}{l^2\cdot m}$

#7: $\theta''(t) = -\frac{g\cdot\text{SIN}(\theta(t))}{l}$

Functional Derivative.dfw

#1: $\text{FD}(f, a, \xi) := \text{SUBST}\left(\frac{d}{d\xi}\text{SUBST}(f, a, \xi), \xi, a\right)$

#2: $[r(t) :=, \theta(t) :=, V(r) :=]$

#3: $\text{FD}(V(r(t)), r(t)) = \text{SUBST}(V'(\xi), \xi, r(t))$

#4: $\text{FD}(V(a\cdot r(t)), r(t)) = a\cdot V'(a\cdot r(t))$

#5: $\text{FD2}(f, a, \xi) := \lim_{\xi\to a}\frac{d}{d\xi}\text{SUBST}(f, a, \xi)$

#6: $\text{FD2}(V(r(t)), r(t)) = V'(r(t))$

#7: $\text{FD2}(V(a\cdot r(t)), r(t)) = a\cdot V'(a\cdot r(t))$

Marcelo sent – as promised – an article *Functional Derivative* – to be published in our next newsletter.

Many thanks to Marcelo. He is a very eager member of our community. He sends interesting questions concerning the DERIVE syntax and possible DERIVE deficiencies but also concerning DERIVE advantages, Gracias.

# Fibonacci Divisors and Fibonacci-Nim

Rüdeger Baumann, Germany

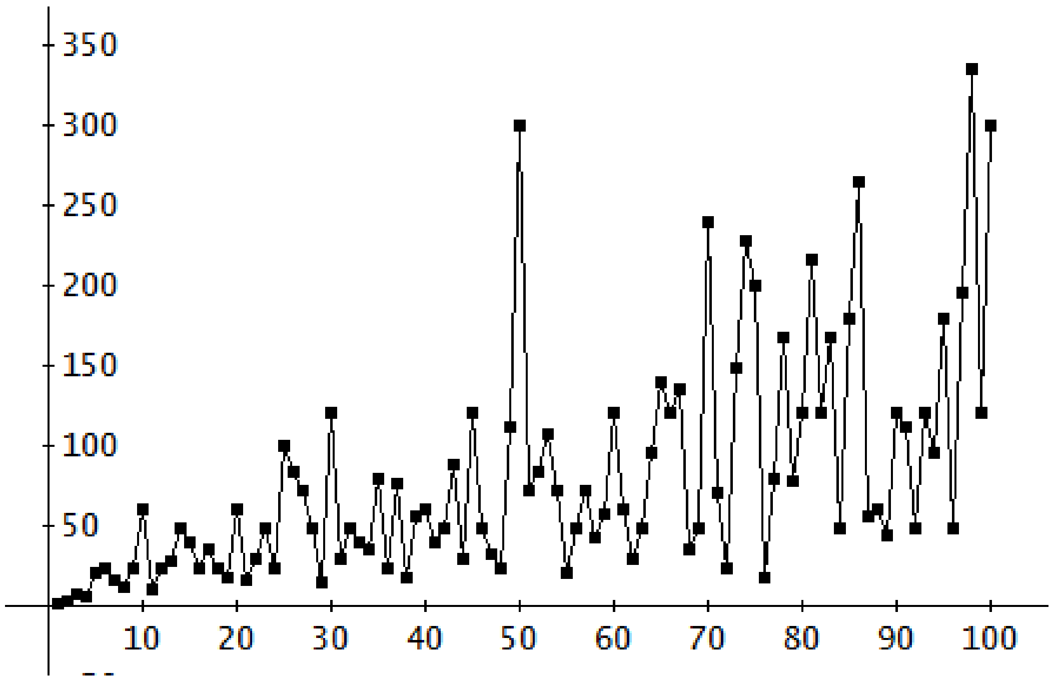
It is well known that subsequent Fibonacci numbers don't have common divisors. But inspecting pairs ($f_k$, $f_{k+1} - 1$) with $f_k$ being the $k^{th}$ Fibonacci number one can ask for common divisors. Or one asks for pairs ($f_k$, $f_{k+1}$) with a given natural number *m* as common divisor.

Take *m* = 3 then we find pair (21, 34) with *k* = 8, or *m* = 5, then (6765, 10946) with *k* = 20 and for *m* = 8 pair (144,233) with *k* = 12.

Can you find any regularity or even more, a mathematical law?

<u>Problem:</u> Write a program which finds to any given natural number *m* the least pair ($f_k$, $f_{k+1}$) of Fibonacci numbers such that *m* is divisor of $f_k$ and $f_{k+1} - 1$.

When presenting the pairs (*m*,*k*) in a coordinate system, then a random scatter diagram appears:



But when taking larger numbers, certain structures can be recognised. Special lines attract our attention.

How can they be explained?

This is what I - Josef - did:

I use DERIVE's built-in function for creating the Fibonacci numbers:

```
fib(k_) := FIBONACCI(k_)

VECTOR(fib(k), k, 1, 10)

[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

and produce a list of the first 10 pairs ($f_k$, $f_{k+1} - 1$):

```
VECTOR([fib(k), fib(k + 1) - 1], k, 1, 10)
```

```
┌  1   0 ┐
│  1   1 │
│  2   2 │
│  3   4 │
│  5   7 │
│  8  12 │
│ 13  20 │
│ 21  33 │
│ 34  54 │
└ 55  88 ┘
```

Then I undertake the first try: which pairs among the first 50 Fib-numbers have 2 as common divisor?

```
SELECT(MOD(u , 2) = 0 ∧ MOD(u , 2) = 0, u, VECTOR([k, fib(k), fib(k + 1) - 1], k, 1, 50))
            2                3
```

```
┌  3           2           2 ┐
│  6           8          12 │
│  9          34          54 │
│ 12         144         232 │
│ 15         610         986 │
│ 18        2584        4180 │
│ 21       10946       17710 │
│ 24       46368       75024 │
│ 27      196418      317810 │
│ 30      832040     1346268 │
│ 33     3524578     5702886 │
│ 36    14930352    24157816 │
│ 39    63245986   102334154 │
│ 42   267914296   433494436 │
│ 45  1134903170  1836311902 │
└ 48  4807526976  7778742048 ┘
```

9th Fib-number is 34, 10th number – 1 is 55 – 1 = 54, so we have 2 as common divisor.

We start with *k* = 3 and the respective following numbers form an arithmetic sequence with difference 3 (rule, law?).

Let's try the next natural numbers for *m*:

Replacing 2 by 3 we receive the following table:

```
┌  8          21          33 ┐
│ 16         987        1596 │
│ 24       46368       75024 │
│ 32     2178309     3524577 │
│ 40   102334155   165580140 │
└ 48  4807526976  7778742048 ┘
```

Again, the numbers of the Fib-numbers form an arithmetic sequence starting with 8 and difference 8.

For further investigations I define a function doing the job:

```
fps(m, n) := SELECT(MOD(u , m) = 0 ∧ MOD(u , m) = 0, u, VECTOR([k, fib(k),
                         2                3

  fib(k + 1) - 1], k, 1, n))
```

$$\mathrm{fps}(3,\ 30) = \begin{bmatrix} 8 & 21 & 33 \\ 16 & 987 & 1596 \\ 24 & 46368 & 75024 \end{bmatrix}$$

$$\mathrm{fps}(4,\ 20) = \begin{bmatrix} 6 & 8 & 12 \\ 12 & 144 & 232 \\ 18 & 2584 & 4180 \end{bmatrix}$$

$$\mathrm{fps}(5,\ 60) = \begin{bmatrix} 20 & 6765 & 10945 \\ 40 & 102334155 & 165580140 \\ 60 & 1548008755920 & 2504730781960 \end{bmatrix}$$

$$\mathrm{fps}(6,\ 80) = \begin{bmatrix} 24 & 46368 & 75024 \\ 48 & 4807526976 & 7778742048 \\ 72 & 498454011879264 & 806515533049392 \end{bmatrix}$$

$$\mathrm{fps}(10,\ 200) = \begin{bmatrix} 60 & 1548008755920 & 2504730781960 \\ 120 & 5358359254990966640871840 & 8670007398507948658051920 \\ 180 & 18547707689471986212190138521399707760 & 30010821454963453907530667147829489880 \end{bmatrix}$$

$$\mathrm{fps}(13,\ 150) = \begin{bmatrix} 28 & 317811 & 514228 \\ 56 & 225851433717 & 365435296161 \\ 84 & 160500643816367088 & 259695496911122584 \\ 112 & 114059301025943970552219 & 184551825793033096366332 \\ 140 & 81055900096023504197206408605 & 131151201344081895336534324865 \end{bmatrix}$$

We see that the number *k* of the Fibonacci number forming the least pair with common divisor *m* gives the next pairs with fib(2*k*),fib(3*k*), fib(4*k*), …

It is obvious that fps(6,*n*) is the intersection set of fps(2,*n*) and of fps(3,*n*).

The problem is to find the connection between a given *m* and the first *k*. It is easy to find a function giving the least pair as requested by Rüdeger Baumann:

$$\mathrm{lfp}(m,\ k) := \left[ m,\ (\mathrm{fps}(m,\ k))_{1,1} \right]$$

$$\mathrm{lfp}(13,\ 150) = [13,\ 28]$$

The first pair divisible by 13 is fib(28) and fib(29) – 1 (see above!).

$$[\mathrm{fib}(28),\ \mathrm{fib}(29) - 1] = [317811,\ 514228]$$

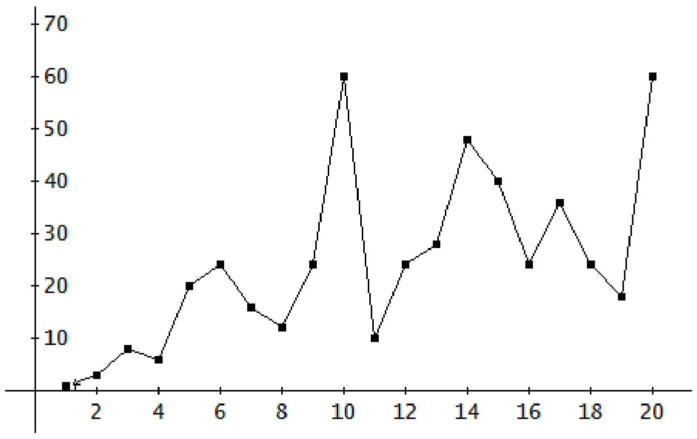
$$\frac{[317811,\ 514228]}{13} = [24447,\ 39556]$$

I must admit that I could not find a function (the mathematical law requested by Rüdeger) which gives explicitly the first pair to a given natural number *m*. It would be great if anybody in the DUG-community could provide further ideas and/or investigations in this direction.

I find the least pairs for the first 10 natural numbers and then present the first 20 pairs in a scatter diagram.

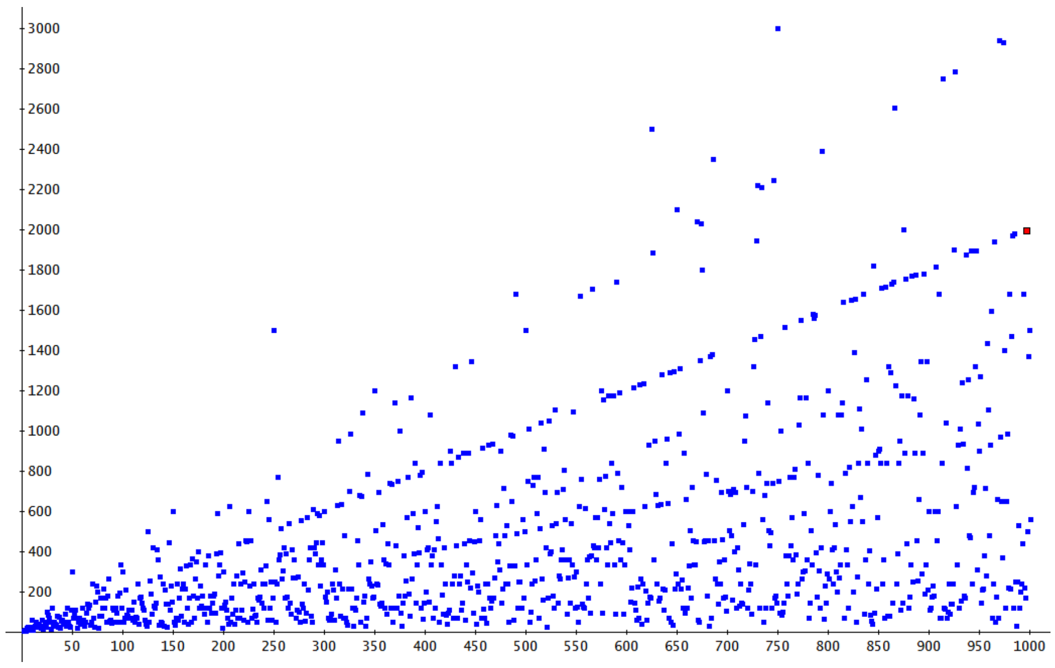
$$\text{VECTOR(lfp(m, 100), m, 10)} = \begin{bmatrix} 1 & 1 \\ 2 & 3 \\ 3 & 8 \\ 4 & 6 \\ 5 & 20 \\ 6 & 24 \\ 7 & 16 \\ 8 & 12 \\ 9 & 24 \\ 10 & 60 \end{bmatrix}$$

```
VECTOR(lfp(m, 100), m, 20)
```



I'd like to take "a little bit" larger values for *m* but then I face the problem that I don't really know the values for *k* necessary to receive the first pair. And at the other hand I don't want to produce a table of many 100 pairs in the background needing only the first one which fits. So, I wrote a small program which stops a loop when the first pair for the respective *m* has been found.

```
fibdiv(n, m, k, divs) :=
  Prog
    divs := []
    m := 1
    Loop
      WRITE(m)
      If m > n exit
      k := 1
      Loop
        If MOD(fib(k), m) = 0 ∧ MOD(fib(k + 1) - 1, m) = 0 exit
        k :+ 1
      divs := APPEND(divs, [[m, k]])
      m :+ 1
    divs

fibdiv(1000)
```



Now I can recognize the “structures” Rüdeger has mentioned above: many points of the scatter diagram seem to lie on or close to special lines. You can see one red point on the right border. Its coordinates are (997, 1996) – which tells us, that 997 is a divisor of `fib(1996)` and `fib(1997)-1`. Check it?

```
GCD(fib(1996), fib(1997) - 1)
─────────────────────────────
             997
```

The GCD is an integer with 206 digits. fib(1996) appears immediately on the screen and shows 417 digits!!

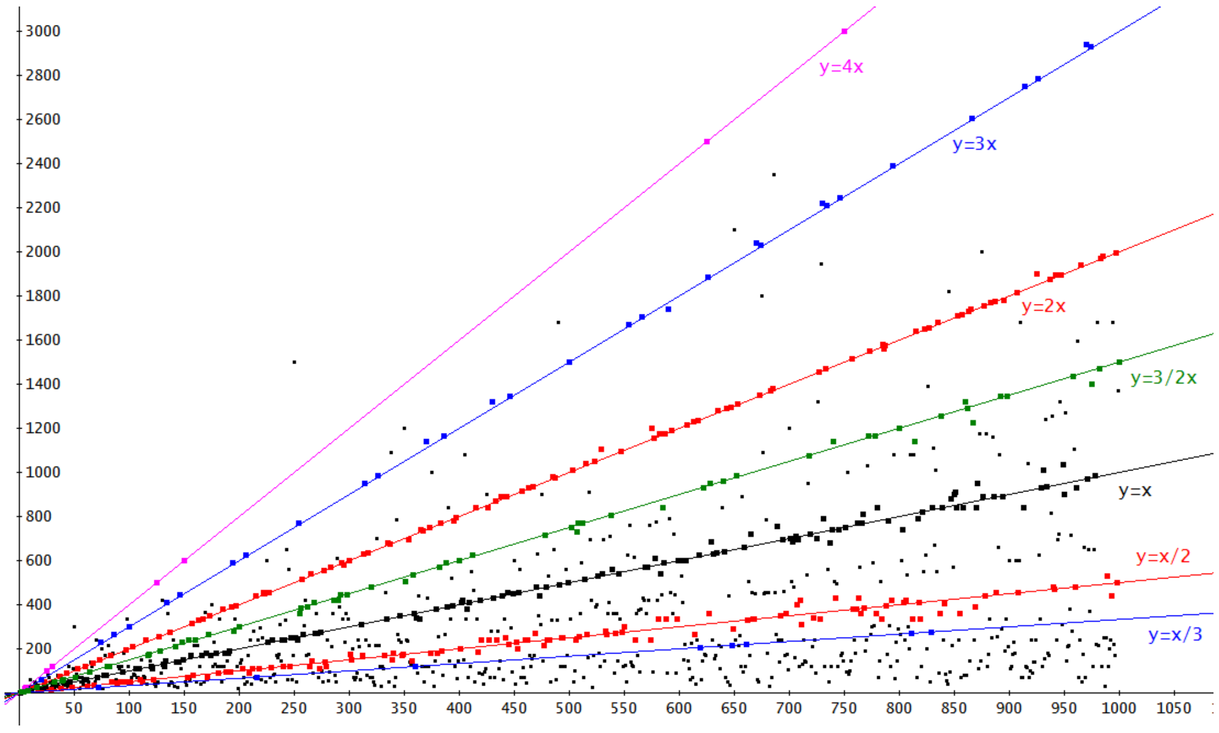
(Thanks Lisp-based DERIVE we can easily handle such long numbers!)

As 1996/997 ~ 2.002006018 I add the plot of $y = 2x$ (red line). Is this one of the lines addressed by Rüdeger?

In order to organize my personal research for the “structures” I change my program from above to present only these points which give a quotient *m*/*k* close to a qiven *q* (integer or rational number). Close is within $q \pm \varepsilon$ ($\varepsilon = 0.1$ by default, but can be changed).

```
fibdiv_quot(n, q, eps := 0.1, i, k, divs) :=
  Prog
    divs := []
    i := 1
    Loop
      WRITE(i)
      If i > n exit
      k := 1
      Loop
        If MOD(fib(k), i) = 0 ∧ MOD(fib(k + 1) - 1, i) = 0 exit
        k :+ 1
      divs := IF(ABS(k/i - q) ≤ eps, APPEND(divs, [[i, k]]), divs)
      i :+ 1
    divs
```

I plot `fibdiv_quot(1000,1)`, `fibdiv_quot(1000,2)`, …, `fibdiv_quot(1000,1/3)`) in different colors and add the respective lines in same colors:



But which pattern is hidden?

There are three points with show a kind of pattern in their coordinates (*m*,*k*).

Remember: (12,24) says that fib(24) and fib(25)–1 is the least pair with common divisor 12.

```
fps(12, 100)
```

$$\begin{bmatrix} 24 & 46368 & 75024 \\ 48 & 4807526976 & 7778742048 \\ 72 & 498454011879264 & 806515533049392 \\ 96 & 51680708854858323072 & 83621143489848422976 \end{bmatrix}$$

```
fibdiv_quot(1000, 2, 0.05)

93 points in 0.05-neighborhood of 2

fibdiv_quot(1000, 2, 0.01)

54 points in 0.05-neighborhood of 2
```

$$\text{fibdiv\_quot}(1000,\ 2,\ 0) = \begin{bmatrix} 12 & 24 \\ 60 & 120 \\ 300 & 600 \end{bmatrix}$$

I add some other interesting results ($\varepsilon = 0$) which point to more lines forming the “structure”:

$$\text{fibdiv\_quot}(1000,\ 1,\ 0) = \begin{bmatrix} 1 & 1 \\ 24 & 24 \\ 120 & 120 \\ 600 & 600 \end{bmatrix}$$

$$\text{fibdiv\_quot}(1000,\ 3,\ 0) = \begin{bmatrix} 20 & 60 \\ 100 & 300 \\ 500 & 1500 \end{bmatrix}$$

$$\text{fibdiv\_quot}(1000,\ 4,\ 0) = \begin{bmatrix} 5 & 20 \\ 6 & 24 \\ 25 & 100 \\ 30 & 120 \\ 125 & 500 \\ 150 & 600 \\ 625 & 2500 \\ 750 & 3000 \end{bmatrix}$$

$$\text{fibdiv\_quot}\left(1000,\ \frac{1}{2},\ 0\right) = \begin{bmatrix} 48 & 24 \\ 96 & 48 \\ 192 & 96 \\ 240 & 120 \\ 384 & 192 \\ 480 & 240 \\ 768 & 384 \\ 960 & 480 \end{bmatrix}$$

Yes, there is a pattern, but unfortunately I am not able to form it into any formula or rule: I can neither find a connection between the given quotient *q* and the first pair nor between *q* and the factor from one pair to the next one – sometimes it is 5, then 2, or there are even two factors like below with 3 and 5 or there are two starting numbers as above (5 and 6)?

$$\text{fibdiv\_quot}\left(1000,\ \frac{1}{3},\ 0\right) = \begin{bmatrix} 72 & 24 \\ 216 & 72 \\ 360 & 120 \\ 648 & 216 \end{bmatrix}$$

$$\text{fibdiv\_quot}\left(200,\ \frac{3}{2},\ 0\right) = \begin{bmatrix} 2 & 3 \\ 4 & 6 \\ 8 & 12 \\ 16 & 24 \\ 32 & 48 \\ 40 & 60 \\ 64 & 96 \\ 80 & 120 \\ 128 & 192 \\ 160 & 240 \\ 200 & 300 \end{bmatrix}$$

I am quite sure – or, I hope so, - that there are DUGers who see more than I. Maybe that I don't "see the wood because of so many trees" – German saying: "Ich sehe den Wald vor lauter Bäumen nicht".

Again Rüdeger Baumann: You are invited to investigate pairs ($f_k$, $f_{k+1}$ + 1). Our DERIVE-colleagues are called to pose respective questions and to make their own discoveries.

Title from Rüdeger Baumann's second origin paper from 200? – long ago ..

## Fibonacci-Nim

After Richard Schorn described the game *Wyhoff-Nim* (DNL#45) and Johann Wiesenbauer showed the connections to the Fibonacci sequence (DNL#46) I'd like to present the DUG-colleagues a strategy game, which is also connected with the Fibonacci numbers.

*n* counters are on the table. Both players remove alternating at least one counter. The player who cannot remove a counter – because there are no more – is loser. There are restrictions: it is not allowed to remove all counters in the first move; and it is also not allowed to remove more than twice as many counters which have been removed by the opponent.

An example game: 15 counters are on the table, player A starts:

15 counters remaining, player A removes 2 counters
13 counters remaining, player B removes 1 counter
12 counters remaining, player A removes 2 counters
10 counters remaining, player B removes 2 counters
8 counters remaining, player A removes 2 counters
6 counters remaining, player B removes 1 counter
5 counters remaining, player A removes 2 counters
3 counters remaining, player B removes 3 counters
0 counters remaining, B wins the game

A could have won the game, if he had not made a mistake in the third move: instead of two counters he should have removed only one, then 11 counters had remained for B and now B had no chance to fight against 8 counters on his next turn – because this is a loser position.

For analysing the game, the famous Fibonacci sequence 1, 1, 2, 3, 5, 8, 13, 34, 55, … is of importance because – you can recognise this in our sample game – numbers 3, 5, 8, 13 appear at essential positions.

This sequence has – among others – the interesting property that every natural number can be presented as sum of not neighboured Fibonacci numbers (Theorem of Zeckendorf [1,2]. Examples: 15 = 13 + 2, 12 = 8 + 3 + 1, 10 = 8 + 2, 6 = 5 + 1.

[1] https://en.wikipedia.org/wiki/Zeckendorf%27s_theorem
[2] http://www.ijon.de/mathe/fibonacci/node5.html

If the game starts with a Fib-number of counters on the table, then player B will win – otherwise player A must reduce the numbers of counters to a Fibonacci number and he/she will win. This can be achieved as follows:

(1) If it is A's turn to move at a non-Fib-number (at the begin of a game or during the game), he should do this: he removes – if possible – all counters, otherwise he presents the number as sum of Fib-numbers and removes the smallest summand.

(2) If A is starting with a Fib-number and leaves a Fib-number for B, then B can remove all counters and wins the game. If A does not leave a Fib-number, then B can proceed as described in (1) above.

Problem: Write a program which can recognize whether a given natural number is a Fibonacci number or not.

I'd like to show two solutions one without and one with DERVE's built-in `FIBONACCI`-function:

```
fib(k_) := FIBONACCI(k_)

fib?(n, f1, a1, a2, k, nf) :=
  Prog
    a1 := 1
    a2 := 1
    k := 2
    Loop
      k :+ 1
      nf := a1 + a2
      If nf = n ∨ nf > n exit
      a1 := a2
      a2 := nf
    If nf = n
       APPEND("Fibonacci Number, Nr. ", STRING(k))
       APPEND("no Fibonacci Number – last FN: ", STRING(nf – a1))

fib?(139583862450) = no Fibonacci Number – last FN: 139583862445

fib?(139583862445) = Fibonacci Number, Nr. 55

fib(55) = 139583862445

fib?(222232244629420445529739893461909967206666939096499764990979000)

no Fibonacci Number – last FN:

  137347080577163115432025771710279131845700275212767467264610201


fib?(137347080577163115432025771710279131845700275212767467264610201)

  = Fibonacci Number, Nr. 299
```

And here is the version with the built-in function!

```
fib2?(n, k) :=
  Prog
    k := 0
    Loop
      k :+ 1
      If fib(k) = n ∨ fib(k) > n exit
    If fib(k) = n
       APPEND("Fibonacci Number, Nr. ", STRING(k))
       APPEND("no Fibonacci Number – last FN: ", STRING(fib(k – 1)))
```

```
fib2?(222232244629420445529739893461909967206666939096499764990979000)

no Fibonacci Number - last FN:

  137347080577163115432025771710279131845700275212767467264610201


fib2?(137347080577163115432025771710279131845700275212767467264610201~
  ) = Fibonacci Number, Nr. 299


fib(299) =

  137347080577163115432025771710279131845700275212767467264610201
```

Both versions work, but the first one works much faster.

Rüdeger: It is possible to define a Fibonacci-number system like our decimal system. Only digits 0 and 1 are permitted. As an example, 17 can be composed as $17 = 1 \cdot 13 + 0 \cdot 8 +$ $+ 0 \cdot 5 + 1 \cdot 3 + 0 \cdot 2 + 1 \cdot 1$, so we can denote the decimal 17 as a Fib-number 100101. Because of $F_2 = F_1 = 1$ and $F_0 = 0$ we don't need $F_1$ and $F_0$ for representing the numbers.

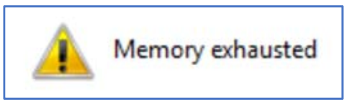
Problem: Write a program which gives to any given natural number its representation in the Fibonacci-number system.

Partial solution:

```
FibSeq := [0, 1, 2, 3, 5, 8, 13, 21, 34]

fib_Coded(n, i := 0, word := []) :=
  Prog
    Loop
      i := i + 1
      If FibSeq↓i > n exit
    Loop
      i := i - 1
      If i ≤ 1
         RETURN REVERSE(word)
      If n < FibSeq↓i
         word := ADJOIN(0, word)
         Prog
           n := n - FibSeq↓i
           word := ADJOIN(1, word)
```

As you can see, Rüdeger's partial solution works, but why is it only a partial solution (Josef)? Try

```
fib_Coded(34)
```


Memory exhausted

It works only then properly if you have predefined `FibSeq` in sufficient length.

```
VECTOR([n, fib_Coded(n)], n, 20)

┌ 1          [1]           ┐
  2        [1, 0]
  3       [1, 0, 0]
  4       [1, 0, 1]
  5     [1, 0, 0, 0]
  6     [1, 0, 0, 1]
  7     [1, 0, 1, 0]
  8   [1, 0, 0, 0, 0]
  9   [1, 0, 0, 0, 1]
 10   [1, 0, 0, 1, 0]
 11   [1, 0, 1, 0, 0]
 12   [1, 0, 1, 0, 1]
 13 [1, 0, 0, 0, 0, 0]
 14 [1, 0, 0, 0, 0, 1]
 15 [1, 0, 0, 0, 1, 0]
 16 [1, 0, 0, 1, 0, 0]
 17 [1, 0, 0, 1, 0, 1]
 18 [1, 0, 1, 0, 0, 0]
 19 [1, 0, 1, 0, 0, 1]
└20 [1, 0, 1, 0, 1, 0]     ┘
```

In my version I start searching how many elements of the Fib-sequence (1, 2, 3, 5, 8, …) are needed (then stored in list `fs`).

```
fib_C(n, a1, a2, a3, fs, fc) :=
  Prog
    If n = 1
       RETURN [1]
    [a1 := 1, a2 := 2]
    [fs := [a2, a1], fc := []]
    Loop
      a3 := a1 + a2
      fs := APPEND([a3], fs)
      If a3 > n exit
      [a1 := a2, a2 := a3]
    Loop
      fs := REST(fs)
      If fs = [] exit
      If n < fs↓1
         fc := APPEND(fc, [0])
      If n ≥ fs↓1
         Prog
           n := n - fs↓1
           fc := APPEND(fc, [1])
    RETURN fc
```

```
VECTOR([n, fib_C(n)], n, 20)

  1        [1]
  2       [1, 0]
  3      [1, 0, 0]
  4      [1, 0, 1]
```

```
VECTOR([n, fib_C(n)], n, 80, 95)

 80    [1, 0, 1, 0, 0, 0, 1, 0, 1]
 81    [1, 0, 1, 0, 0, 1, 0, 0, 0]
 82    [1, 0, 1, 0, 0, 1, 0, 0, 1]
 83    [1, 0, 1, 0, 0, 1, 0, 1, 0]
 84    [1, 0, 1, 0, 1, 0, 0, 0, 0]
 85    [1, 0, 1, 0, 1, 0, 0, 0, 1]
 86    [1, 0, 1, 0, 1, 0, 0, 1, 0]
 87    [1, 0, 1, 0, 1, 0, 1, 0, 0]
 88    [1, 0, 1, 0, 1, 0, 1, 0, 1]
 89  [1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
 90  [1, 0, 0, 0, 0, 0, 0, 0, 0, 1]
 91  [1, 0, 0, 0, 0, 0, 0, 0, 1, 0]
 92  [1, 0, 0, 0, 0, 0, 0, 1, 0, 0]
 93  [1, 0, 0, 0, 0, 0, 0, 1, 0, 1]
 94  [1, 0, 0, 0, 0, 0, 1, 0, 0, 0]
 95  [1, 0, 0, 0, 0, 0, 1, 0, 0, 1]
```

95 = 89 + 5 + 1

Back to Rüdeger's last problem presented in his Fibonacci−Nim paper:

We could also take 1 and 2 as digits because every natural number *n* can be written in the form

$n = a_1 \cdot F_1 + a_2 \cdot F_2 + \ldots + a_k \cdot F_k$ (or with decreasing place value in reverse order).

with $a_i = 1$ or $a_i = 2$. Example: Because of 17 = 2·1+2·1+1·2+2·3+1·5 we can represent 17 as 22121 or 12122 (reversed). This has been proved just recently by an Austria Greek (or: Greek Austrian with French name) Gert-Elias Lampakis-Morçeau (private communication from 1. 1. 2003).

Problem: Develop a program which represents any given natural number in "austro-graeco" form.

This was a nice problem to face and it took me some time to produce correct running programs. I write plural form "programs" because I had two approaches for both sequences, the regular Fibonacci sequence and the reduced one (doing without the leading 1).

My first solution is a bit tricky on the one hand but very slow for larger numbers on the other hand: I remembered a function to generate the combinations of *n* elements in groups of *k* elements with replacement: I try all products of possible lists of 1s and 2s,starting with [1], [2], [1,1],[1,2], [2,1], [2,2], [1,1,1], [1,1,2], [1,2,1],[1,2,2], … and the lists of Fib-numbers with respective length [1], [1,1],[1,1,2], [1,1,2,3], [1,1,2,3,5], … until their dot product equates the number which I want to represent.

The function producing the combinations mentioned above is given at the end of the DERIVE-part of this contribution, Josef.

The program and the first try (represent 17):

```
f(k) := VECTOR(FIBONACCI(k_), k_, k)

autgre(n, k, k_, kk, m_ := 0, n_, t_) :=
  Prog
    k := 1
    Loop
      Loop
        If m_ = 2^k exit
        k_ := k
        n_ := m_
        t_ := []
        Loop
          t_ := ADJOIN([1, 2]↓(MOD(n_, 2) + 1), t_)
          n_ := FLOOR(n_, 2)
          If f(k)·t_ = n
             Prog
               t_ := REVERSE(t_)
               RETURN (CODES_TO_NAME(VECTOR(NAME_TO_CODES(c), c, t_)'))↓1
          k_ :- 1
          If k_ = 0 exit
        m_ :+ 1
      k :+ 1

autgre(17) = 12122
```

It seems to work. As an extra a didn't want to return the list of coefficients only, but a number with decreasing place values (needs some work with the list elements – see in the RETURN-line).

```
                                 ┌ 1     1 ┐
                                 │ 2     2 │
                                 │ 3    12 │
                                 │ 4    22 │
                                 │ 5   112 │
VECTOR([k, autgre(k)], k, 1, 10) = │ 6   122 │
                                 │ 7   212 │
                                 │ 8   222 │
                                 │ 9  1122 │
                                 └ 10 1212 ┘
```

Take a larger number:

```
autgre_(2000) = 12111112112121

needs 200 sec

REVERSE(f_(14)) = [610, 377, 233, 144, 89, 55, 34, 21, 13, 8, 5, 3, 2, 1]

REVERSE(f_(14))·[1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 2, 1] = 2000
```

Unfortunately, the representation of natural numbers in this Fib-system is not unique: 17 ~ 12211. But we have also 17 ~ 21111:

```
[2, 1, 1, 1, 1]·[5, 3, 2, 1, 1] = 17
```

What about this problem using the reduced Fib-sequence? Program `autgre_(n)` is in the file.

```
autgre_(17) = 2112
```

I don't know if now uniqueness is given. At least I didn't find a counter example.

I was not really satisfied with my solution because of two reasons:

(1) too slow for large numbers and

(2) the use of the built-in FIBONACCI.function.

My version for the reduced Fib-sequence is very fast and no built-in function is used:

```
autgre2(n, a1, a2, a3, fs, sf, co) :=
  Prog
    [a1 := 1, a2 := 2]
    [fs := [a1], sf := a1, fc := []]
    Loop
      fs := APPEND([a2], fs)
      sf := sf + a2
      If sf > n exit
      a3 := a2 + a1
      a1 := a2
      a2 := a3
    fs := REST(fs)
    sf := sf - a2
    Loop
      If fs = [] exit
      If n - fs↓1 ≥ sf
         co := 2
         co := 1
      fc := APPEND(fc, [co])
      n := n - co·fs↓1
      fs := REST(fs)
      sf := Σ(fs)
    RETURN (CODES_TO_NAME(VECTOR(NAME_TO_CODES(c), c, fc)'))↓1
```

```
autgre2(17) = 2112

autgre2(2000) = 12111112112121

in zero time

[1, 2, 1, 1, 1, 1, 1, 2, 1, 1, 2, 1, 2, 1]·REVERSE(f_(14)) = 2000

REVERSE(f_(14)) = [610, 377, 233, 144, 89, 55, 34, 21, 13, 8, 5, 3, 2, 1]

autgre2(20000) = 1112121111212121111

in 0.000 sec
```

I don't know if Lampakis' proof mentioned by Rüdeger Baumann is also valid for the representation using the reduced Fibonacci sequence.

My final comment to the DERIVE programs: I am quite sure that there are more elegant ways to perform the conversions. I would be glad to receive one from our members.

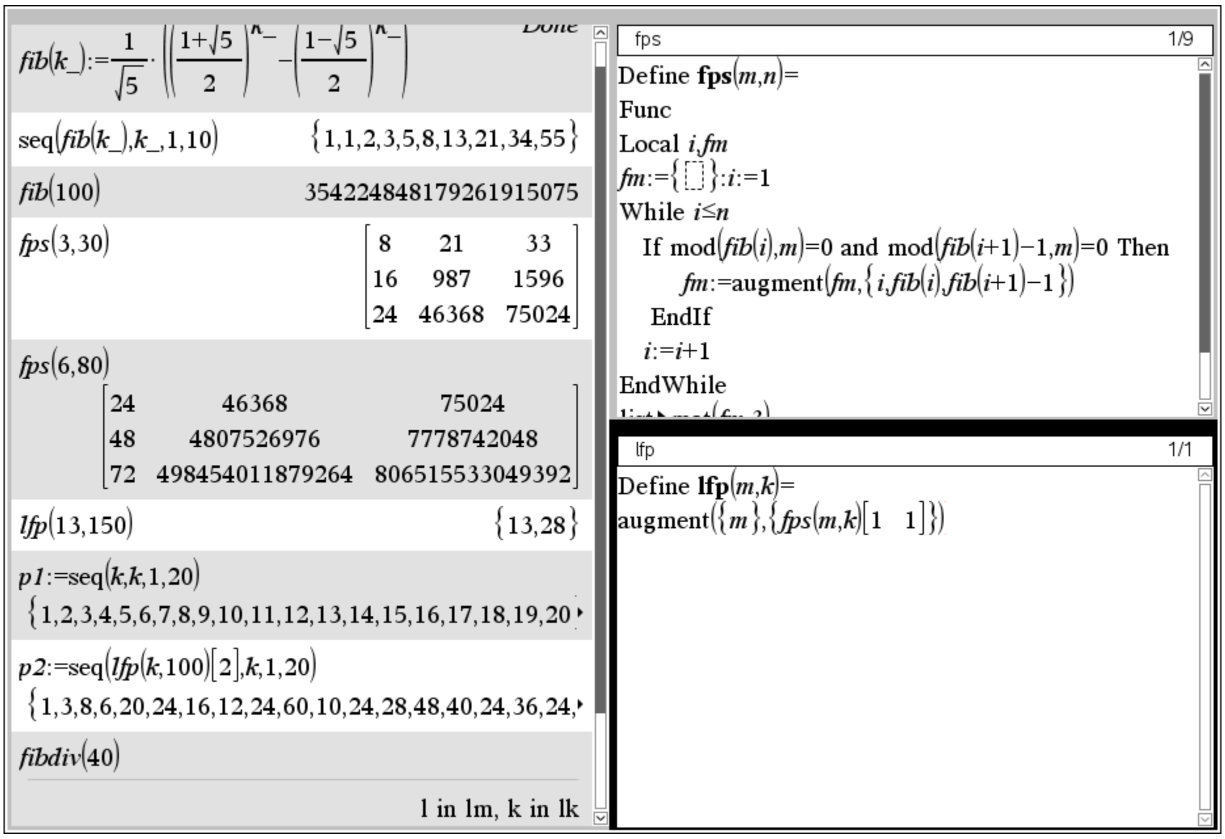
And here is the function producing the combinations mentioned above:

```
vars(v, k, b, k_, m_ := 0, n_, s_ := [], t_) :=
  Loop
    b := DIM(v)
    If m_ = b^k
       RETURN REVERSE(s_)
    k_ := k
    n_ := m_
    t_ := []
    Loop
      t_ := ADJOIN(v↓(MOD(n_, b) + 1), t_)
      n_ := FLOOR(n_, b)
      k_ :- 1
      If k_ = 0 exit
    s_ := ADJOIN(t_, s_)
    m_ :+ 1
```

$$\text{vars([1, 2], 3)} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 1 \\ 1 & 2 & 2 \\ 2 & 1 & 1 \\ 2 & 1 & 2 \\ 2 & 2 & 1 \\ 2 & 2 & 2 \end{bmatrix}$$

I didn't want to discount the TI-NspireCAS treatment of Rüdeger's challenging problems – and it was a nice programming work, indeed.

Everything is working – but with restrictions because TI-Nspire cannot handle as large numbers as DERIVE. *fibdiv*(40) works, but *fibdiv*(50) does not. Therefore I didn't find a way to visualize the "structures" which Rüdeger wanted to be discovered. *fibdiv_quot* was not programmed.



$fib(k\_):=\frac{1}{\sqrt{5}}\cdot\left(\left(\frac{1+\sqrt{5}}{2}\right)^{k\_}-\left(\frac{1-\sqrt{5}}{2}\right)^{k\_}\right)$ Done

seq(*fib*(*k_*),*k_*,1,10) {1,1,2,3,5,8,13,21,34,55}

*fib*(100) 354224848179261915075

*fps*(3,30)

$$\begin{bmatrix} 8 & 21 & 33 \\ 16 & 987 & 1596 \\ 24 & 46368 & 75024 \end{bmatrix}$$

*fps*(6,80)

$$\begin{bmatrix} 24 & 46368 & 75024 \\ 48 & 4807526976 & 7778742048 \\ 72 & 498454011879264 & 806515533049392 \end{bmatrix}$$

*lfp*(13,150) {13,28}

*p1*:=seq(*k*,*k*,1,20)
{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20▸

*p2*:=seq(*lfp*(*k*,100)[2],*k*,1,20)
{1,3,8,6,20,24,16,12,24,60,10,24,28,48,40,24,36,24,▸

*fibdiv*(40)

1 in lm, k in lk

fps 1/9

```
Define fps(m,n)=
Func
Local i,fm
fm:={□}:i:=1
While i≤n
  If mod(fib(i),m)=0 and mod(fib(i+1)−1,m)=0 Then
     fm:=augment(fm,{i,fib(i),fib(i+1)−1})
   EndIf
  i:=i+1
EndWhile
```

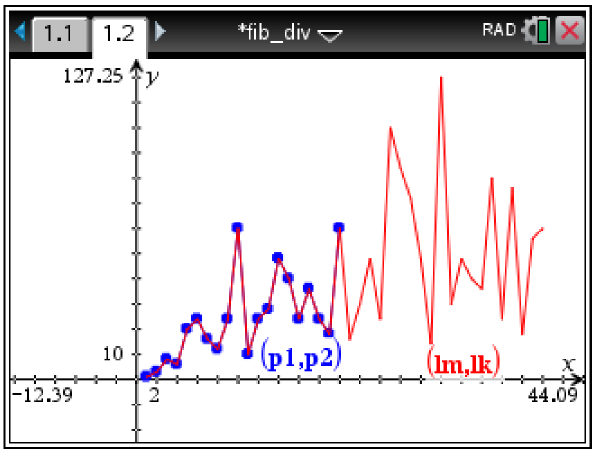
lfp 1/1

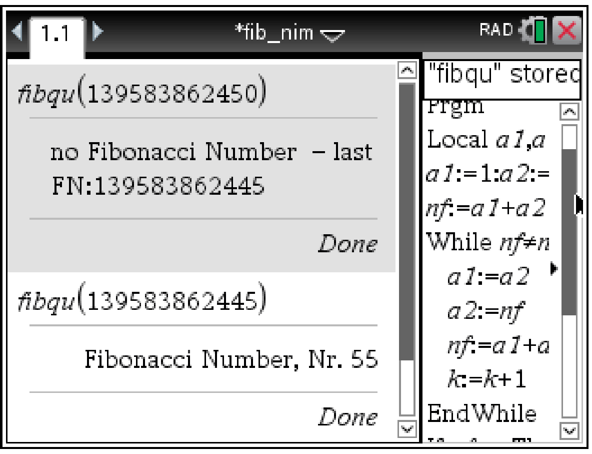
```
Define lfp(m,k)=
augment({m},{fps(m,k)[1  1]})
```

The DERIVE programs were more or less "translated" into TI-Basic of TI-NspireCAS. Some things must be changed:

The DERIVE-loops are replaced by While-EndWhile-loops, VECTORs and VECTOR-operations are replaced by lists ({ }) and respective operations.

Take augment instead of APPEND, take care for the correct application of the IF-constructs, …

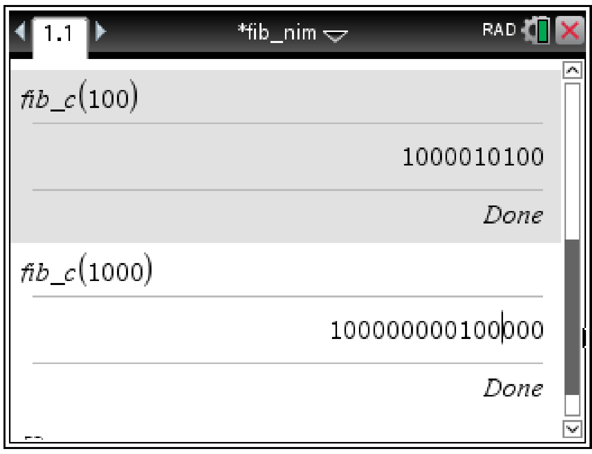




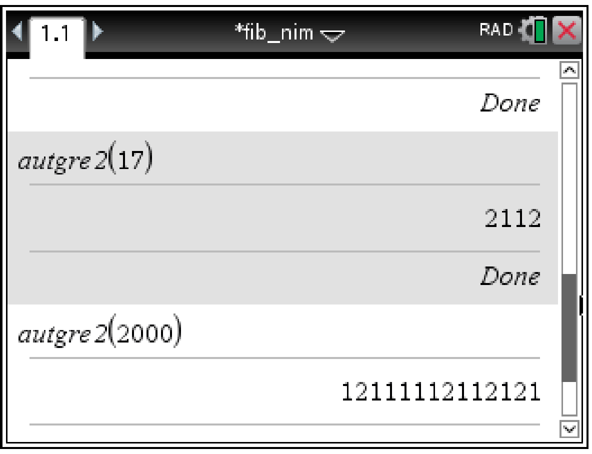
```
fibqu(139583862450)
  no Fibonacci Number  – last
  FN:139583862445
                          Done
fibqu(139583862445)
      Fibonacci Number, Nr. 55
                          Done
```

```
"fibqu" stored
Prgm
Local a1,a
a1:=1:a2:=
nf:=a1+a2
While nf≠n
  a1:=a2
  a2:=nf
  nf:=a1+a
  k:=k+1
EndWhile
```

The DERIVE-programs contain the useful REST-function. It can be successfully substituted by right(list,dim(list)−1).



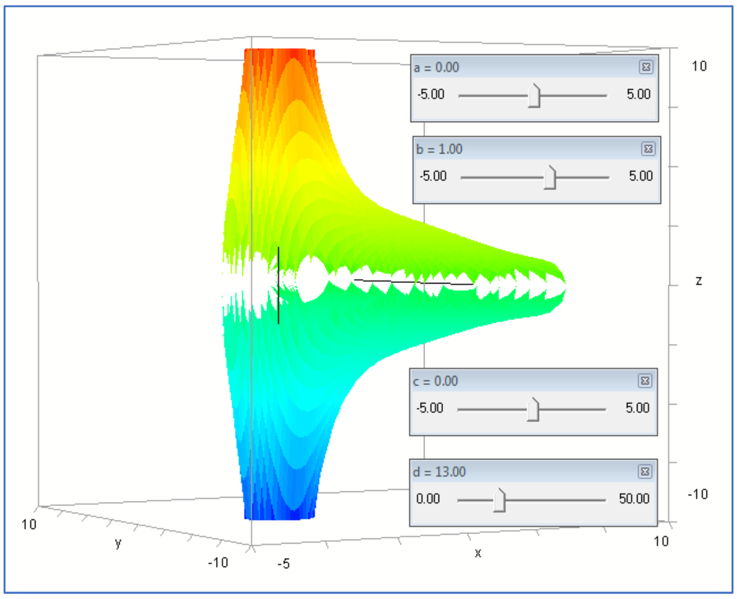
```
fib_c(100)
                   1000010100
                         Done
fib_c(1000)
             100000000100000
                         Done
```



```
                          Done
autgre2(17)
                          2112
                          Done
autgre2(2000)
               12111112112121
```

The TI-Nspire program codes are not printed. If you are interested you are invited to load fib_div.tns and fib_nim.tns. Both files are contained in MTH113.zip.

This screen shot belongs to the contribution on page 42.



# Anageo

## From a Voyage 200 program via TI-NspireCAS to TI-NspireCAS+Lua

Jens Staacke, Borna, Germany and Josef Böhm (supported by
Steve Arnold, Australia and Geneviève Savard, Canada))

Some time ago I found a mail from a German DUG-member in my mailbox
(I summarize in English):

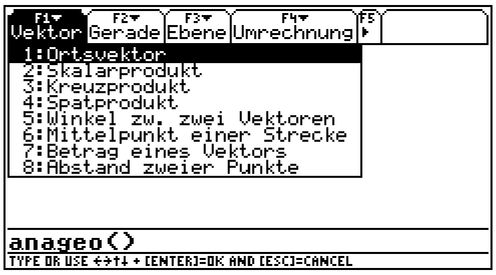
… I found some of your publications with regard to TI-NspireCAS in the web. I am Math teacher at gymnasium “Am Breiten Teich” in Borna, Saxonia.

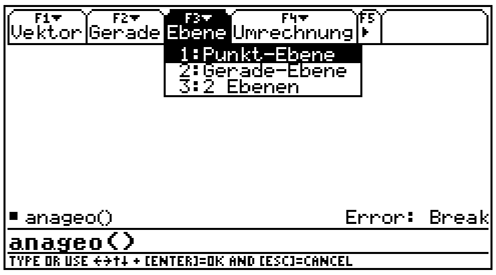
Inspired by my suggestions my students wanted to rewrite Voyage 200 programs for TI-Nspire. For this reason, they have acquired some knowledge in LUA. They – and I, too – would like to know if there is a possibility to program customized menus on the top of the screen. I attach program anageo (a tool for analytic geometry) as an example.

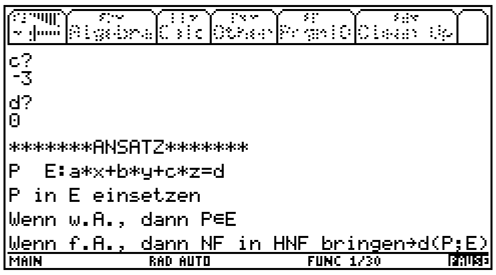
Can we create similar menus supported by LUA, too?

Best regards, Jens Staacke

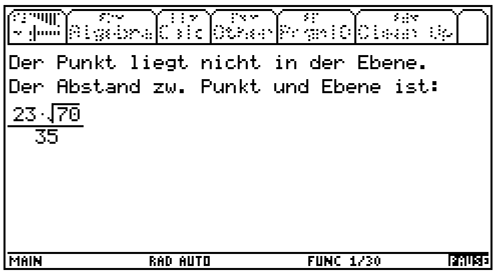
I connected my good old V200 with the PC, loaded TI Connect, transferred the program and produced some screen shots:







```
c?
-3
d?
0
*******ANSATZ*******
P  E:a*x+b*y+c*z=d
P in E einsetzen
Wenn w.A., dann P∈E
Wenn f.A., dann NF in HNF bringen→d(P;E)
```



```
Der Punkt liegt nicht in der Ebene.
Der Abstand zw. Punkt und Ebene ist:
```

$$\frac{23\cdot\sqrt{70}}{35}$$

Menu F3 Plain offers 1:Point-Plain which treats the relationship between a point and a plain given in R3. After a short explication of the procedure, the user is asked for the coordinates of the point and the coefficients of the equation of the plane. Then the result is displayed. In this case: The point does not lie in the plain and its distance from the plain is 23*√70/35.

As I had no – good – idea how to create such menus and because my LUA-knowledge is also very restricted, I decided to ask my LUA-Guru from Down Under, Steve Arnold. I must admit that I feared that – if at all possible – according to my LUA-experience it would be hard programming work.

I wrote to Steve and in the meanwhile I tried to find another way to substitute the V200-menus.

Dear Steve,

Just a short question:
is it possible to call (and to run) a program (contained in a library) from the LUA-environment via a menu created in LUA?
Many thanks
Josef

This was Steve's immediate answer

Hi Josef

The short answer is "No". Lua can only step outside it's sandbox via variables, and while, strictly speaking, a TI-BASIC program IS a variable, I have been unable to find any way to RUN these from Lua.

I HAVE been able to find ways for Lua to store, edit and define programs by storing them as strings.

This allows a Lua Script to carry multiple programs and define these in a new document – works well with the concept of Widgets.

So, if you regularly have use in a document of multiple TI-BASIC programs, then you can send a widget to the document which will automatically define these, and the user can then run them in the usual way.

http://compasstech.com.au/TNS_Authoring/Scripting/script_tut40.html

On your previous question, programs can only run within Calculator, properly, and Lua has no way to reach a Calculator window.

However, FUNCTIONS are much more flexible, running well in Notes and Spreadsheet – it may be possible for a Lua script to create and run a function?

Steve

Dear LUA-Guru,

Ooops, so quick to receive an answer …

But I was also busy in between: I studied your sampler script and built in a short function for calculating the direction vector between two points.

The reason for my request was a mail from a German Nspire user who wants to have a menu guided package of programs for geometry (in the plane and in space as well). He expects to have similar menus like the menus created with V200 (menu bar on the top with dropdown-menus to choose the various functions).

This is impossible with the Nspire. I proposed to split the many programs into groups and use libraries. This works very fine.

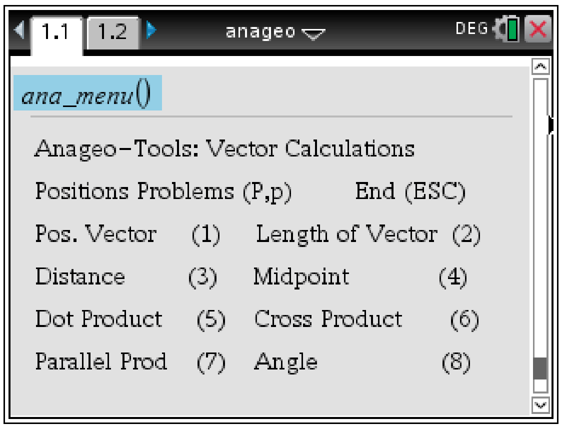
Then the question came up how to do with LUA? See how I started to change your Innovator-package.

Best regards and many thanks (I am also busy with your script_tut40 …)
Josef

(See also my further communication with Steve on page 36.)

At first, I'd like to show my proposal for the Non-LUA-version.

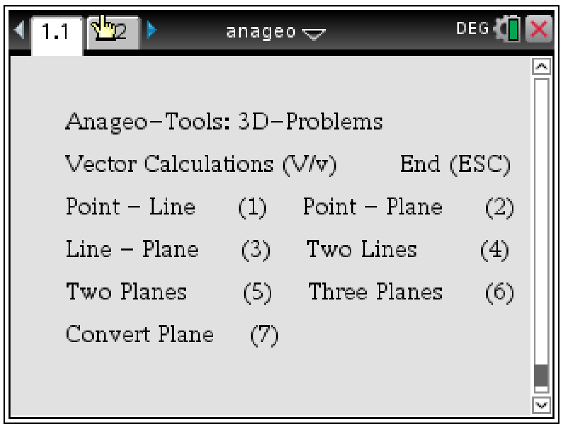
Load anageo.tns. *ana_menu*() opens the English version and *anageo_menu*() the German one:



*ana_menu*()

Anageo−Tools: Vector Calculations

| | | | |
|---|---|---|---|
| Positions Problems (P,p) | | End (ESC) | |
| Pos. Vector | (1) | Length of Vector | (2) |
| Distance | (3) | Midpoint | (4) |
| Dot Product | (5) | Cross Product | (6) |
| Parallel Prod | (7) | Angle | (8) |



Anageo−Tools: 3D−Problems

| | | | |
|---|---|---|---|
| Vector Calculations (V/v) | | End (ESC) | |
| Point − Line | (1) | Point − Plane | (2) |
| Line − Plane | (3) | Two Lines | (4) |
| Two Planes | (5) | Three Planes | (6) |
| Convert Plane | (7) | | |

You can switch between 3D-problems (Lagenaufgaben) and Vector Calculations (Vektoren).

I did not consider the Conics (*Kegelschnittsaufgaben*) which were also intended in the TI-92 version. There were more or less only the menu headers given and I didn't know which calculations were demanded. At the other hand it might not be too difficult to produce a menu-controlled program for the conics with a structure similar to anageo.tns – if you like this, of course.

I sent the first ideas of this way to "re-invent" the Voyage200 menu to Jens and he wrote back, that his students faced the challenge and accomplished the tool – split in two programs for vectors and 3D-problems. It is a pleasure to congratulate Jens to his motivated students.

anageo.tns is my final version. Some of the 3D-programs (e.g. *three_planes*()) are pretty large and I don't want to bore you with the printed program codes. I will demonstrate the functionality of selected programs on the next pages (and in the next DNL).

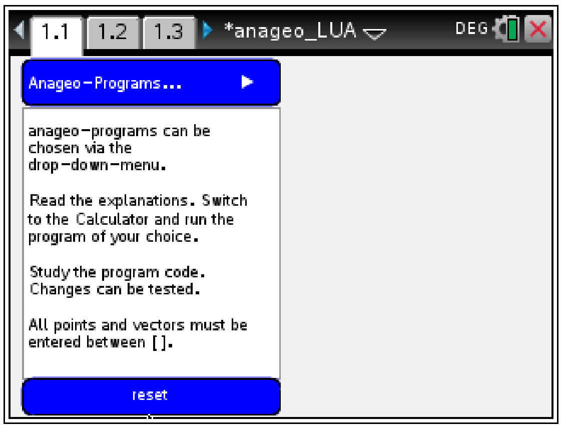
Although I tried to check al special cases for planes and lines, I am not quite sure, if I there is not one or the other bug still well-hidden in the programs. It is also very likely that some program parts could be more elegant. But to quote "The Voice": *I did it my Way*.

A nice feature of Jens' Voyage200 version was that some of the 3D-problems started showing the paper and pen procedure to solve the problem. Two examples are shown in examples to follow.

As I mentioned in earlier articles, programming in LUA, i.e. producing scripts is not easy. I am very grateful that I could use the backbone of Steve's script. As I didn't want – and didn't dare – to omit lines of code, Steve's script comprises more that 1500 lines of code. What I did was to exchange the Innovator programs by my anageo programs – and it works.

As a special extra I added visualizations of some 3D-problems on a TI-Nspire 3D-graph page using Geneviève Savard's wonderful geo3d.tns library which she presented at TIME 2014 in Krems [1]. Geo3d.tns must be stored in your MyLib-folder.

Load anageo_LUA.tns (English version) or anageo_LUA_d.tns (German version). A mouse click on the header opens the list of available programs. You can browse the list and receive short descriptions of the programs.

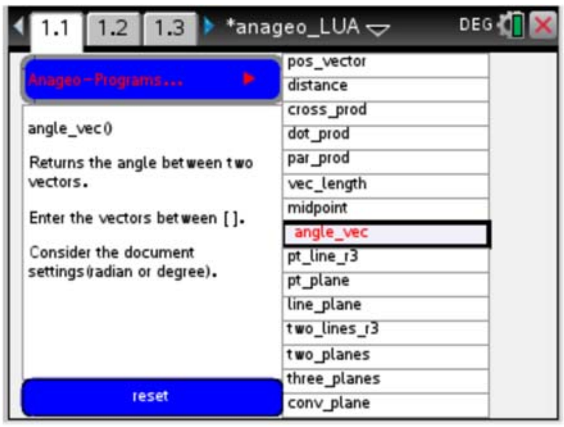


anageo-programs can be chosen via the drop-down-menu.

Read the explanations. Switch to the Calculator and run the program of your choice.

Study the program code. Changes can be tested.

All points and vectors must be entered between [].

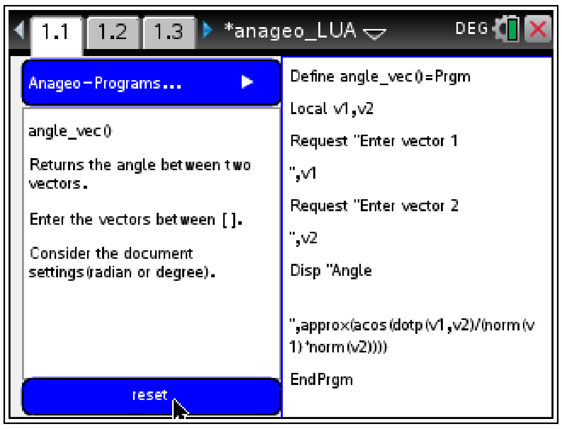


angle_vec()

Returns the angle between two vectors.

Enter the vectors between [].

Consider the document settings (radian or degree).



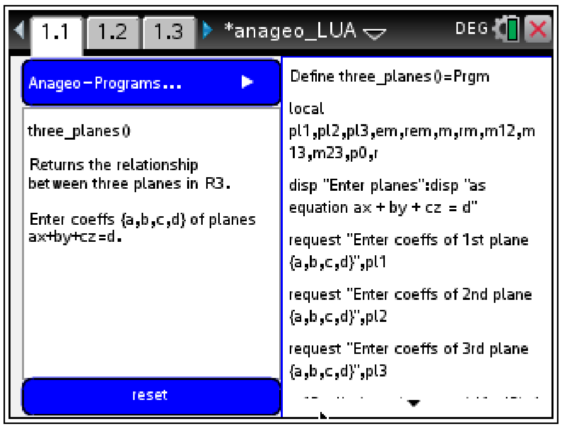
angle_vec()

Returns the angle between two vectors.

Enter the vectors between [].

Consider the document settings (radian or degree).

```
Define angle_vec()=Prgm
Local v1,v2
Request "Enter vector 1
",v1
Request "Enter vector 2
",v2
Disp "Angle

",approx(acos(dotp(v1,v2)/(norm(v
1)*norm(v2))))
EndPrgm
```



three_planes()

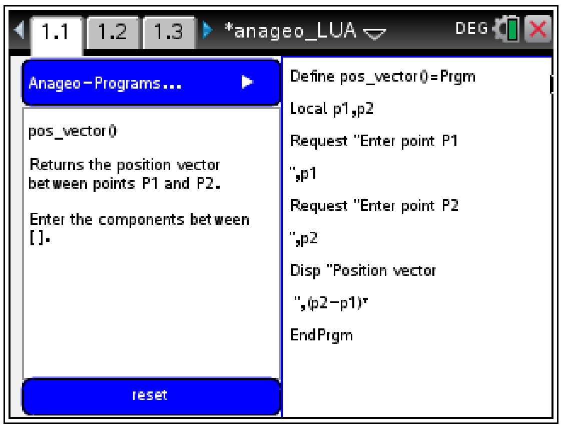
Returns the relationship between three planes in R3.

Enter coeffs {a,b,c,d} of planes ax+by+cz=d.

```
Define three_planes()=Prgm
local
pl1,pl2,pl3,em,rem,m,rm,m12,m
13,m23,p0,r
disp "Enter planes":disp "as
equation ax + by + cz = d"
request "Enter coeffs of 1st plane
{a,b,c,d}",pl1
request "Enter coeffs of 2nd plane
{a,b,c,d}",pl2
request "Enter coeffs of 3rd plane
{a,b,c,d}",pl3
```

Let me start our tour d'horizon with the first program and I will demonstrate another great feature of Steve's LUA-script.

We find the position vector to a vector given by two points. All programs must be run from a calculator page.

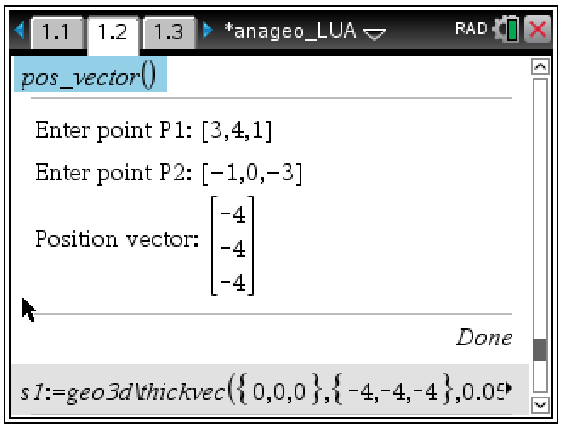


pos_vector()

Returns the position vector between points P1 and P2.

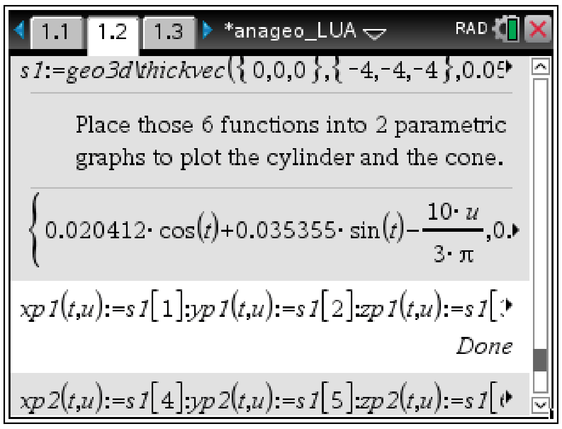
Enter the components between [].

```
Define pos_vector()=Prgm
Local p1,p2
Request "Enter point P1
",p1
Request "Enter point P2
",p2
Disp "Position vector
",(p2-p1)^T
EndPrgm
```

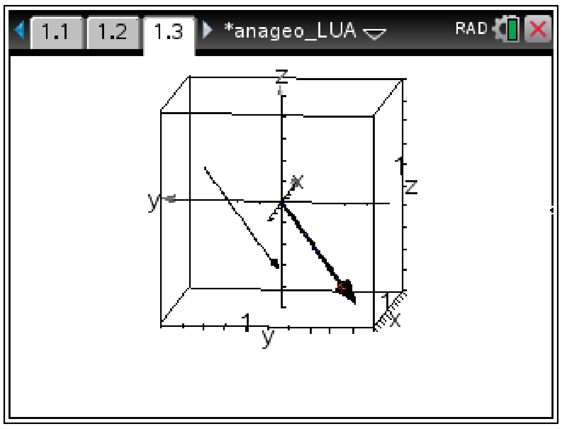


```
pos_vector()
Enter point P1: [3,4,1]
Enter point P2: [-1,0,-3]
Position vector: [-4]
                 [-4]
                 [-4]
                                   Done
s1:=geo3d\thickvec({0,0,0},{-4,-4,-4},0.05
```

In the Entry line you can see my call for *thickvec*(*start, end, radius*) from geo3d.tns. This gives the data for plotting the "thick vector" as a cylinder with an attached cone to represent the position vector.



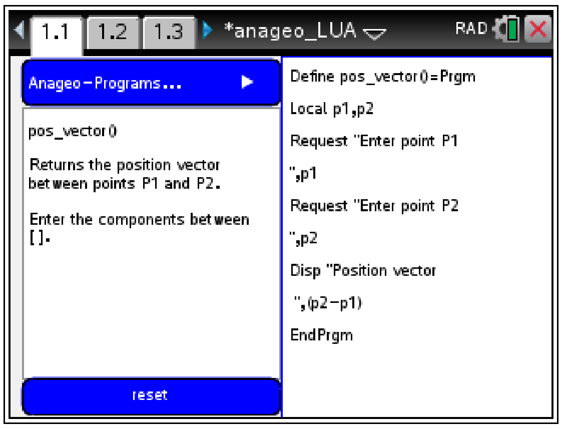
```
s1:=geo3d\thickvec({0,0,0},{-4,-4,-4},0.05
Place those 6 functions into 2 parametric
graphs to plot the cylinder and the cone.
{0.020412· cos(t)+0.035355· sin(t)−10· u/(3· π),0.
xp1(t,u):=s1[1]:yp1(t,u):=s1[2]:zp1(t,u):=s1[
                                          Done
xp2(t,u):=s1[4]:yp2(t,u):=s1[5]:zp2(t,u):=s1[
```



*vec*(*start,end*) gives the functions to plot the given vector as an "ordinary" arrow. Take care to enter the coordinates between { } in Geneviève's functions.

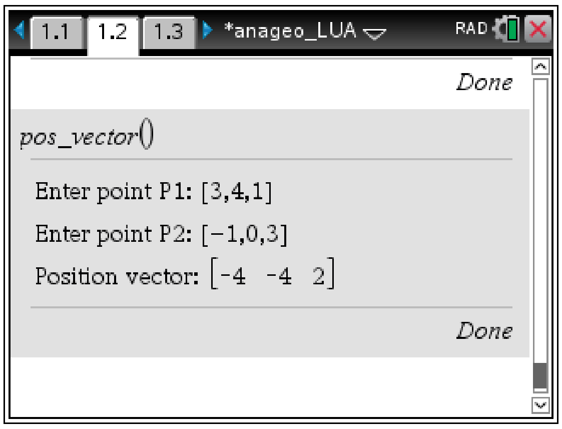
Steve's LUA script offers another very nice – and as I believe useful – feature. It is possible to change the program code and experiment without any danger to remove parts or to destroy the code.

The resulting position vector is represented as a column vector. As all vectors must be entered as row vectors it should be consistent to have also row vectors as results! What to do for the students? They can inspect the function code on the right half of the screen (see above) and try to change it.



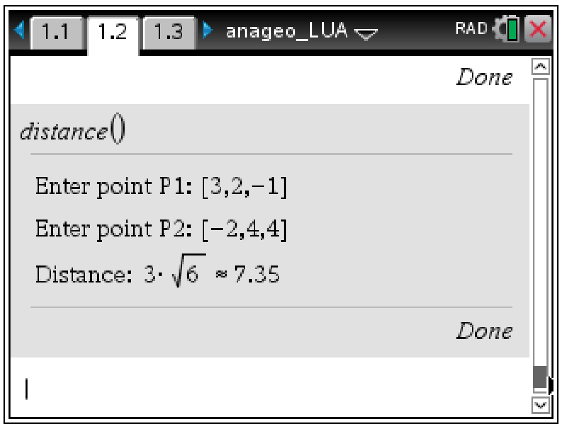
```
pos_vector()

Returns the position vector
between points P1 and P2.

Enter the components between
[].
```

```
Define pos_vector()=Prgm
Local p1,p2
Request "Enter point P1
",p1
Request "Enter point P2
",p2
Disp "Position vector
 ",(p2−p1)
EndPrgm
```



```
                                   Done
pos_vector()
 Enter point P1: [3,4,1]
 Enter point P2: [−1,0,3]
 Position vector: [-4  -4  2]
                                   Done
```

They remove the transpose symbol and run the function again … they found the right way. However, this change is not permanent. For saving the change you have to edit the script and store the program.

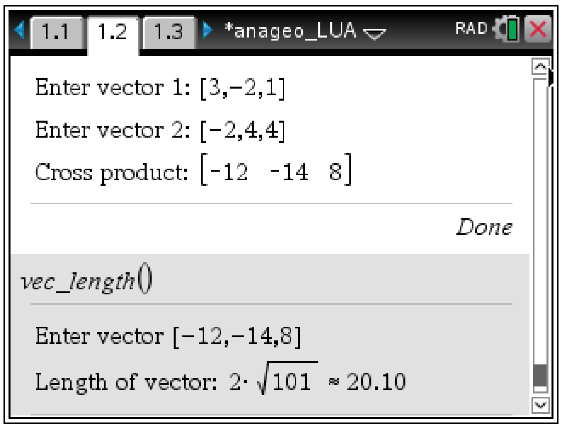


```
                                   Done
distance()
 Enter point P1: [3,2,−1]
 Enter point P2: [−2,4,4]
 Distance: 3· √6  ≈ 7.35
                                   Done
```

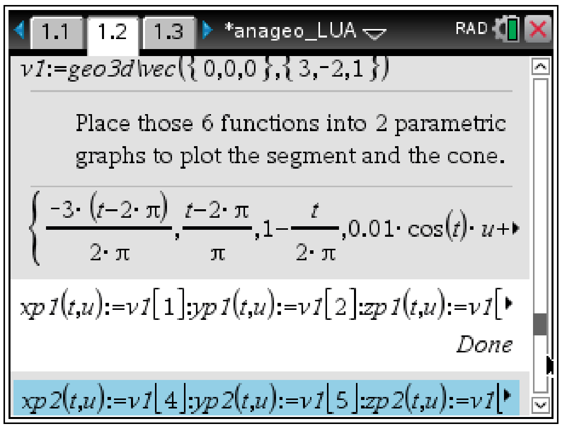
Next menu item gives the distance between two points. Accuracy depends on the Document Settings.

It's clear that it is not necessary defining functions for tasks which can be executed using built-in functions. But it might be useful for students to start defining functions and programs with easy examples.

I proceed with cross product (*cross_prod*()) and its visualization.

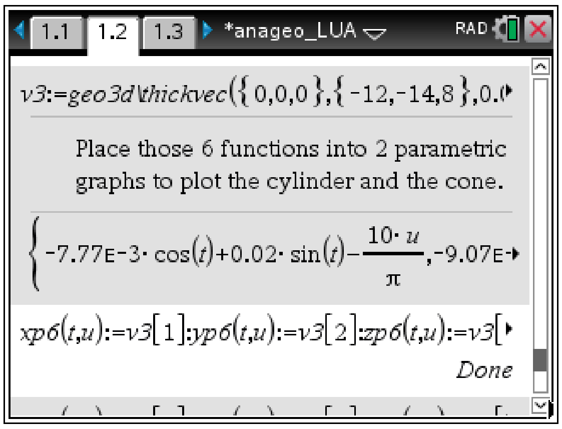


```
Enter vector 1: [3,−2,1]
Enter vector 2: [−2,4,4]
Cross product: [−12  −14  8]
                                  Done
vec_length()
Enter vector [−12,−14,8]
Length of vector: 2·√101  ≈ 20.10
```

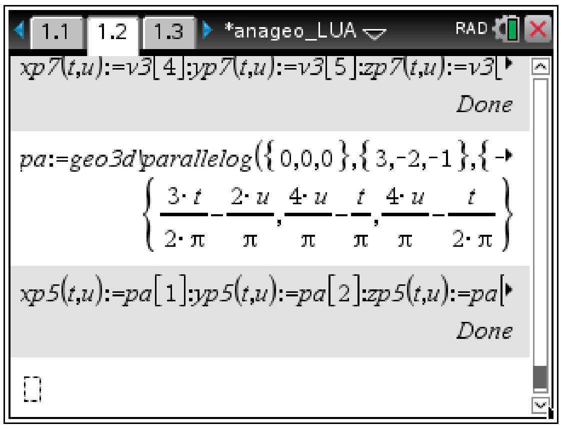


```
v1:=geo3d\vec({0,0,0},{3,−2,1})
   Place those 6 functions into 2 parametric
   graphs to plot the segment and the cone.
{ −3·(t−2·π)/(2·π), (t−2·π)/π, 1−t/(2·π), 0.01·cos(t)·u+▸
xp1(t,u):=v1[1]:yp1(t,u):=v1[2]:zp1(t,u):=v1[▸
                                  Done
xp2(t,u):=v1[4]:yp2(t,u):=v1[5]:zp2(t,u):=v1[▸
```

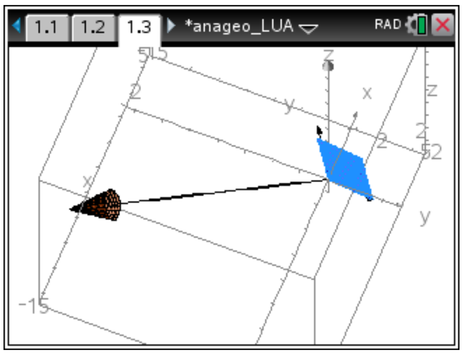
I plot vectors v1 and v2, then the cross-product vector as *thickvector*, and finally the parallelogram with vertex [0,0,0] and *v1* and *v2*.



```
v3:=geo3d\thickvec({0,0,0},{−12,−14,8},0.▸
   Place those 6 functions into 2 parametric
   graphs to plot the cylinder and the cone.
{ −7.77E−3·cos(t)+0.02·sin(t)−10·u/π, −9.07E▸
xp6(t,u):=v3[1]:yp6(t,u):=v3[2]:zp6(t,u):=v3[▸
                                  Done
```



```
xp7(t,u):=v3[4]:yp7(t,u):=v3[5]:zp7(t,u):=v3[▸
                                  Done
pa:=geo3d\parallelog({0,0,0},{3,−2,−1},{−▸
   { 3·t/(2·π) − 2·u/π, 4·u/π − t/π, 4·u/π − t/(2·π) }
xp5(t,u):=pa[1]:yp5(t,u):=pa[2]:zp5(t,u):=pa[▸
                                  Done
```



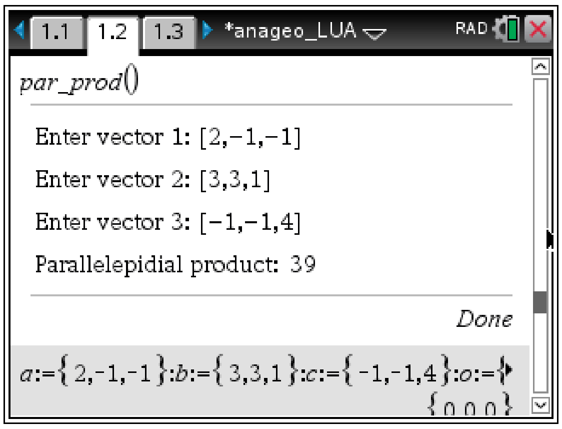
The given vectors form a parallelogram with vertex in origin. The cross product is a vector orthogonal to the plane of the quadrilateral and its length is the value of area of the parallelogram.

The 3D-plot looks much nicer on the PC-screen.

The parallelepidial product (in German: *Spatprodukt*) yields the volume of the parallelepiped spanned by the three vectors.

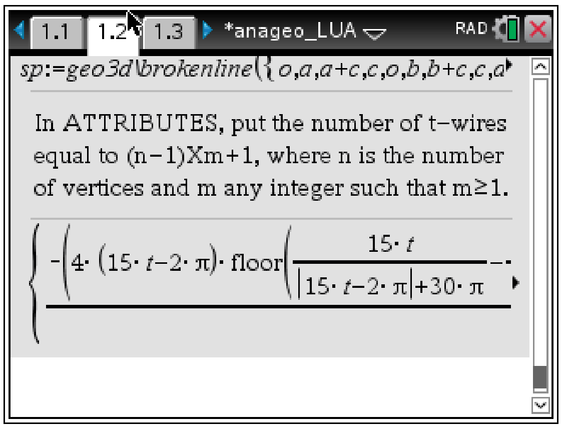
I define the points as lists of coordinates. Then it will be more comfortable to define the parallelepiped as a *geo3d/brokenline*(…).

But first of all, I apply once more *geo3d/vec*(…) to plot the given vectors as position vectors.



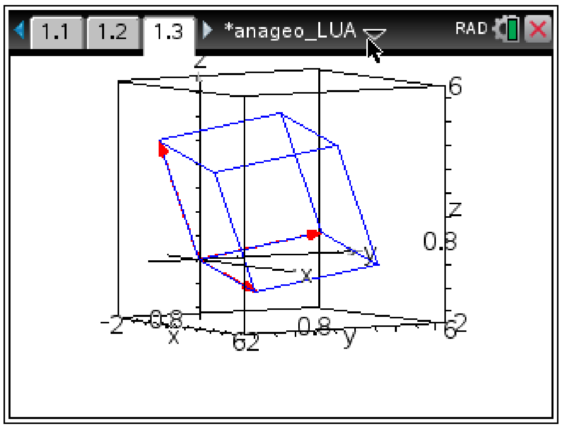
```
par_prod()
Enter vector 1: [2,−1,−1]
Enter vector 2: [3,3,1]
Enter vector 3: [−1,−1,4]
Parallelepidial product: 39
                                  Done
a:={2,−1,−1}:b:={3,3,1}:c:={−1,−1,4}:o:={▸
                                  {0,0,0}
```

The solid is given as a wire frame – one *broken line* connects all vertices:



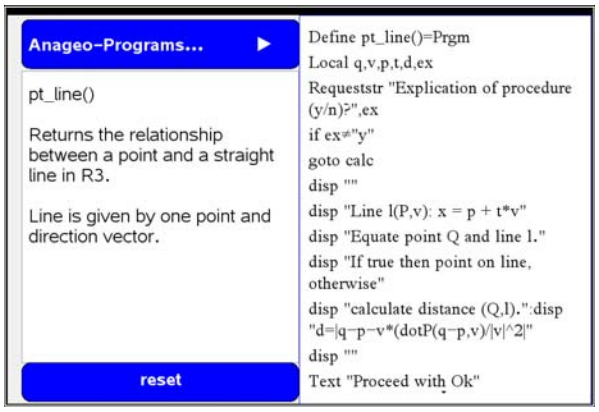
```
sp:=geo3d\brokenline({o,a,a+c,c,o,b,b+c,c,a

In ATTRIBUTES, put the number of t−wires
equal to (n−1)Xm+1, where n is the number
of vertices and m any integer such that m≥1.
```

$$\left\{ -\left(4\cdot(15\cdot t-2\cdot\pi)\cdot \text{floor}\left(\frac{15\cdot t}{|15\cdot t-2\cdot\pi|+30\cdot\pi}\right.\right.\right.$$



Take care to delete variables *o, a, b, c*.

Next progam is *pt_line*(). You can see the given data on the right screen below. The program starts presenting the calculation procedure. This explication can be skipped.

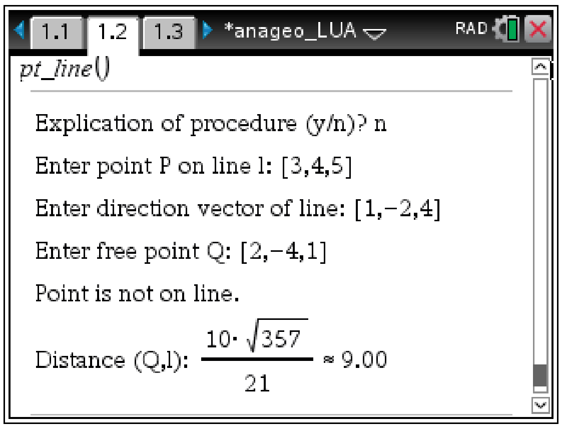


pt_line()

Returns the relationship between a point and a straight line in R3.

Line is given by one point and direction vector.

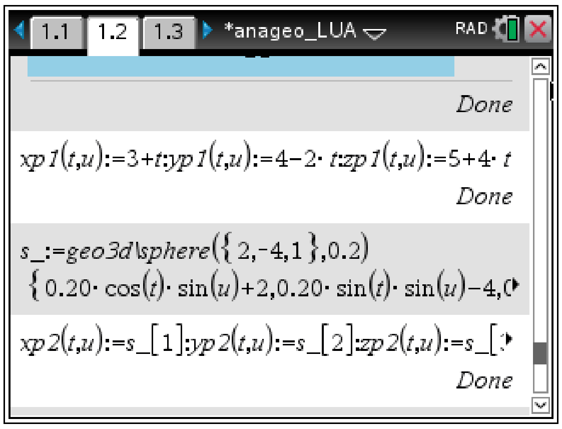
```
Define pt_line()=Prgm
Local q,v,p,t,d,ex
Requeststr "Explication of procedure
(y/n)?",ex
if ex≠"y"
goto calc
disp ""
disp "Line l(P,v): x = p + t*v"
disp "Equate point Q and line l."
disp "If true then point on line,
otherwise"
disp "calculate distance (Q,l).":disp
"d=|q−p−v*(dotP(q−p,v)/|v|^2|"
disp ""
Text "Proceed with Ok"
```



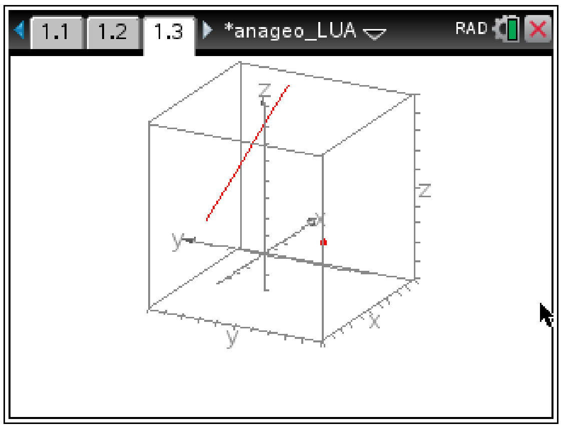
```
pt_line()

Explication of procedure (y/n)? n
Enter point P on line l: [3,4,5]
Enter direction vector of line: [1,−2,4]
Enter free point Q: [2,−4,1]
Point is not on line.
```

$$\text{Distance (Q,l): } \frac{10\cdot\sqrt{357}}{21} \approx 9.00$$

geo3d.tns allows representing single points as small spheres. (The radius of the spheres must be adapted to the size of the box). I plot the line together with point Q.

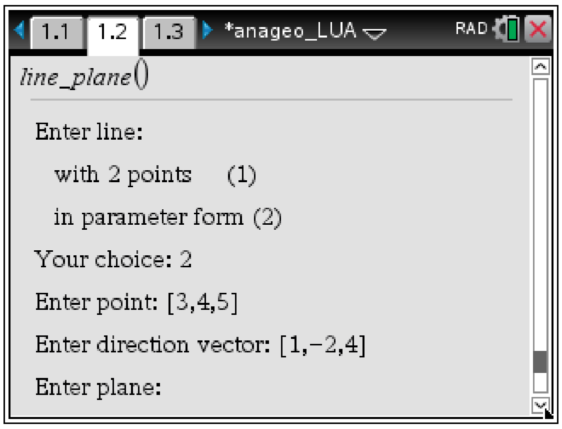


```
Done
xp1(t,u):=3+t:yp1(t,u):=4−2·t:zp1(t,u):=5+4·t
Done
s_:=geo3d\sphere({2,-4,1},0.2)
{0.20·cos(t)·sin(u)+2,0.20·sin(t)·sin(u)−4,0
xp2(t,u):=s_[1]:yp2(t,u):=s_[2]:zp2(t,u):=s_[
Done
```



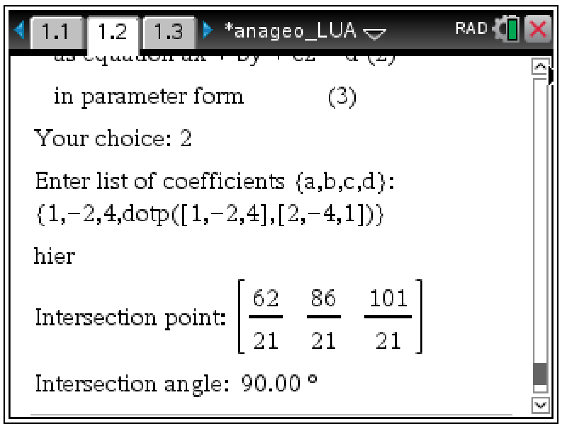
It needs some considerations to set an appropriate range for the box in order to achieve a fine view of the situation given.

I want to visualize the distance between line and point. This gives the opportunity to demonstrate the next program *line_plane*(): the intersection point of the plane orthogonal to the line through Q with line l must be connected with Q.



*line_plane*()

Enter line:
with 2 points (1)
in parameter form (2)
Your choice: 2
Enter point: [3,4,5]
Enter direction vector: [1,−2,4]
Enter plane:

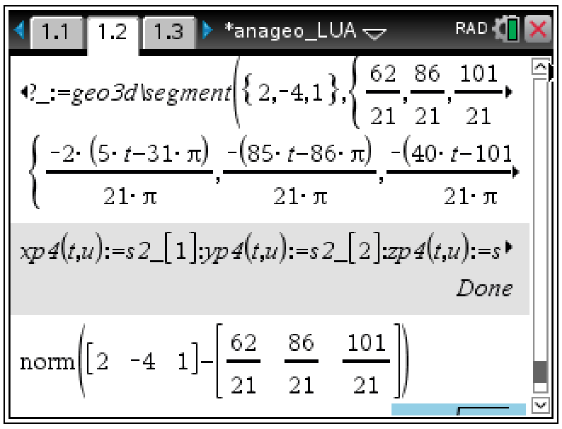


in parameter form (3)
Your choice: 2
Enter list of coefficients {a,b,c,d}:
{1,−2,4,dotp([1,−2,4],[2,−4,1])}
hier

Intersection point: $\begin{bmatrix}\frac{62}{21} & \frac{86}{21} & \frac{101}{21}\end{bmatrix}$

Intersection angle: 90.00 °

I add the intersection point as a second – little bit smaller – sphere and the segment connecting the two points (spheres). Finally, I could calculate the length of this segment using built-in *norm* or anageo’s *distance*() to confirm the result from above.



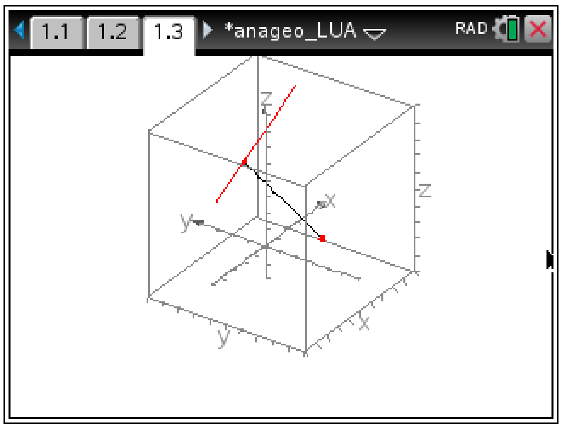
$s2\_:=geo3d\backslash segment\left(\{2,-4,1\},\left\{\frac{62}{21},\frac{86}{21},\frac{101}{21}\right\}\right)$

$\left\{\frac{-2\cdot(5\cdot t-31\cdot\pi)}{21\cdot\pi},\frac{-(85\cdot t-86\cdot\pi)}{21\cdot\pi},\frac{-(40\cdot t-101}{21\cdot\pi}\right.$

$xp4(t,u):=s2\_[1]:yp4(t,u):=s2\_[2]:zp4(t,u):=s$

*Done*

$\text{norm}\left(\begin{bmatrix}2 & -4 & 1\end{bmatrix}-\begin{bmatrix}\frac{62}{21} & \frac{86}{21} & \frac{101}{21}\end{bmatrix}\right)$



I didn’t forget the DERIVians: pt_line “translated”: no dialogue but comfortable plotting!

```
#1:   [CaseMode := Sensitive, Notation := Mixed]

      pt_line(P, v, Q, t, d) :=
        Prog
          DISPLAY("Line l (P,v): x = p + t·v")
          DISPLAY("Equate point Q and line l.")
          DISPLAY("If true then Qεl, otherwise")
          DISPLAY("calculate distance (Q,l):")
#2:       DISPLAY("d = |q-p-v·DOTPRODUCT(q-p,v)/|v|^2|")
          t := (Q↓1 - P↓1)/v↓1
          If Q↓2 = P↓2 + t·v↓2 ∧ Q↓3 = P↓3 + t·v↓3
             RETURN "Point is on line."
          d := ABS(Q - P - v·((Q - P) • v)/ABS(v)^2)
          RETURN ["Distance(Q,l): d =", d, APPROX(d)]

#3:   pt_line([3, 4, 5], [1, -2, 4], [2, -4, 1])

Line l (P,v): x = p + t·v

Equate point Q and line l.

If true then Qεl, otherwise

calculate distance (Q,l):

d = |q-p-v·DOTPRODUCT(q-p,v)/|v|^2|
```

#4: $\left[\text{Distance(Q,l): d =}, \frac{10\cdot\sqrt{357}}{21}, 8.997354108\right]$

As we cannot ask for input data, I must fix the input *line_plane* (point on line, direction vector of the line, coefficients of the plane equation – no braces!!)

```
line_plane([3, 4, 5], [1, -2, 4], [1, -2, 4, 14])
```

$$\begin{bmatrix} \text{Intersection point:} & \begin{bmatrix} \frac{62}{21}, & \frac{86}{21}, & \frac{101}{21} \end{bmatrix} \\ \text{Intersection angle (in °):} & 90 \end{bmatrix}$$

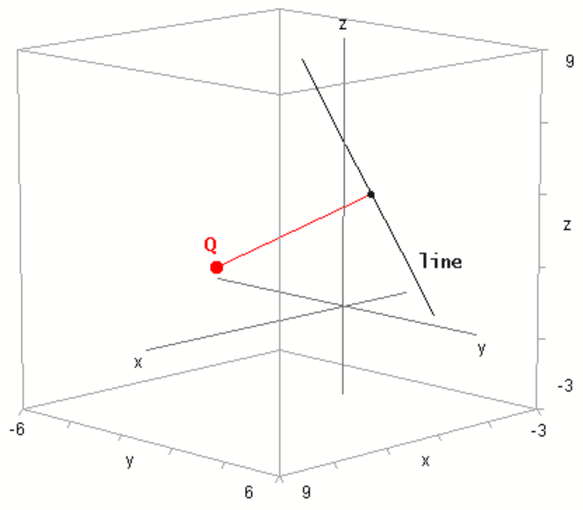
Plotting is easy and we can add labels – are not fixed on the objects.

$[3 + t, 4 - 2\cdot t, 5 + 4\cdot t]$

$[2, -4, 1]$

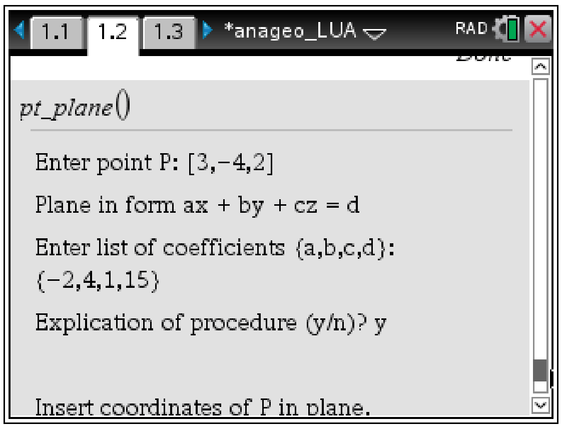
$$\begin{bmatrix} \frac{62}{21}, & \frac{86}{21}, & \frac{101}{21} \end{bmatrix}$$

$$\begin{bmatrix} 2 & -4 & 1 \\ \frac{62}{21} & \frac{86}{21} & \frac{101}{21} \end{bmatrix}$$



*pt_plane*() is the last program of anageo.tns presented in this DNL. In the next issue I will treat the remaining ones – they are not so easy because a lot of special cases must be considered – which is no problem with paper and pen, but needs bulky programming (at least in the way, I did it). *Three planes in space – and how they can intersect* is one topic among the contributions to be published (page 2). *three_planes*() will provide answers.

*pt_plane*() investigates the relationship of point and a plane in R3.



*pt_plane*()

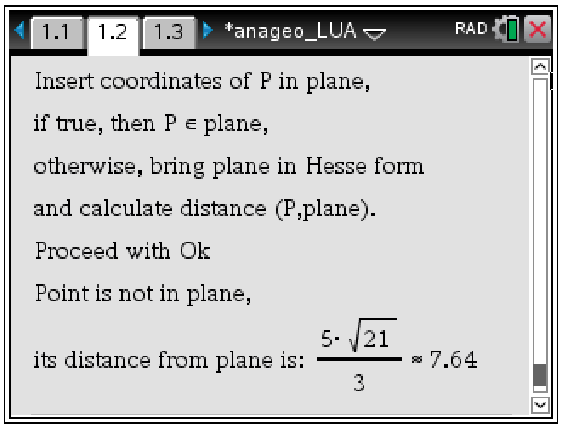
Enter point P: [3,−4,2]

Plane in form ax + by + cz = d

Enter list of coefficients {a,b,c,d}: {−2,4,1,15}

Explication of procedure (y/n)? y

Insert coordinates of P in plane.



Insert coordinates of P in plane,

if true, then P ∊ plane,

otherwise, bring plane in Hesse form

and calculate distance (P,plane).

Proceed with Ok

Point is not in plane,

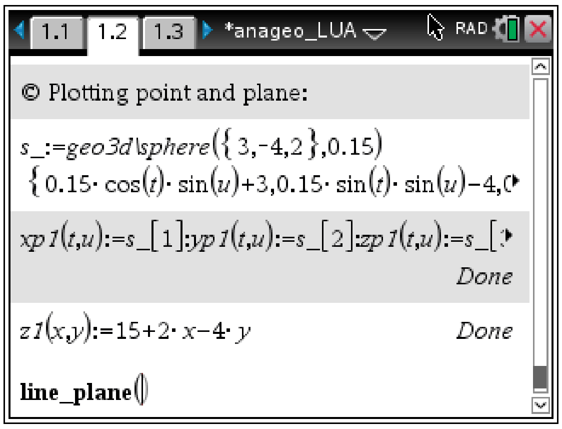
its distance from plane is: $\frac{5\cdot\sqrt{21}}{3} \approx 7.64$

If the plane is not given as equation, then you could use *conv_plane*() in order to convert from the parameter form or from given three points to its equation.

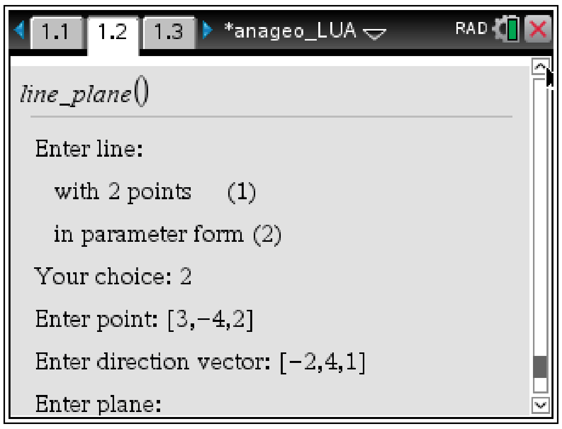
How to find the intersection in a constructive way?

Intersect the line through the point and being orthogonal to the plane with the plane. Then the distance between point and intersection point is the requested distance.

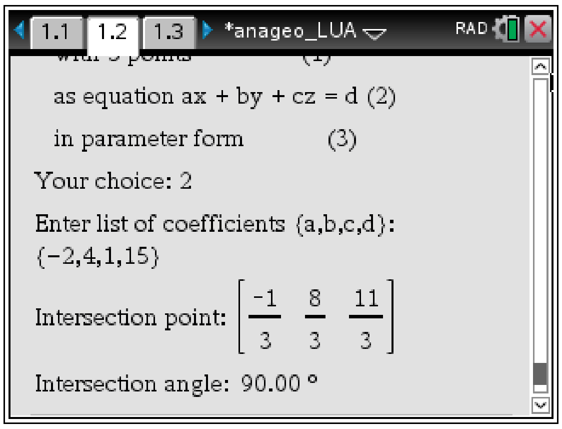
The first three numbers in the parameter list of the plane {-2,4,1} are the components of the normal vector of the plane which is the direction vector of the orthogonal line. We intersect this line with the given plane using in the meanwhile well-known *line_plane*():



```
© Plotting point and plane:
s_:=geo3d\sphere({3,-4,2},0.15)
 {0.15·cos(t)·sin(u)+3,0.15·sin(t)·sin(u)-4,0▸
xp1(t,u):=s_[1]:yp1(t,u):=s_[2]:zp1(t,u):=s_[▸
                                        Done
z1(x,y):=15+2·x-4·y                     Done
line_plane()
```



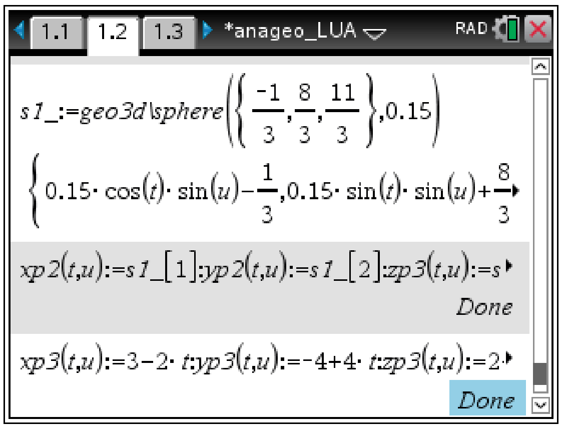
```
line_plane()
  Enter line:
    with 2 points     (1)
    in parameter form (2)
  Your choice: 2
  Enter point: [3,-4,2]
  Enter direction vector: [-2,4,1]
  Enter plane:
```



```
    as equation ax + by + cz = d (2)
    in parameter form            (3)
  Your choice: 2
  Enter list of coefficients {a,b,c,d}:
  {-2,4,1,15}
```

Intersection point: $\left[\frac{-1}{3}\ \frac{8}{3}\ \frac{11}{3}\right]$

Intersection angle: 90.00 °

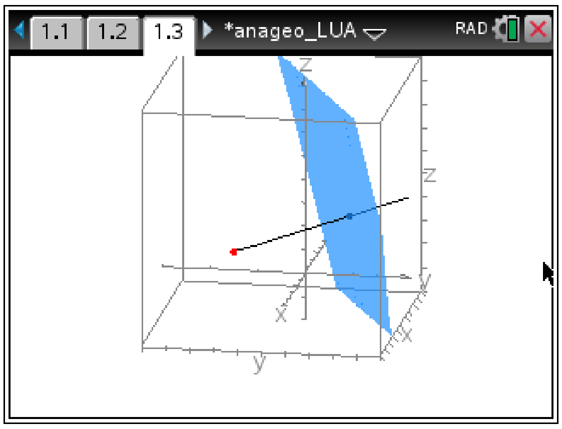


$s1\_:=geo3d\backslash sphere\left(\left\{\frac{-1}{3},\frac{8}{3},\frac{11}{3}\right\},0.15\right)$

$\left\{0.15\cdot\cos(t)\cdot\sin(u)-\frac{1}{3},0.15\cdot\sin(t)\cdot\sin(u)+\frac{8}{3}\right.$▸

```
xp2(t,u):=s1_[1]:yp2(t,u):=s1_[2]:zp3(t,u):=s▸
                                        Done
xp3(t,u):=3-2·t:yp3(t,u):=-4+4·t:zp3(t,u):=2▸
                                        Done
```

Now I can accomplish the 3D plot (plane in blue, given point in red, intersection point and normal line in black.



Distance between intersection point and given point P is $\frac{5\sqrt{21}}{3}$.

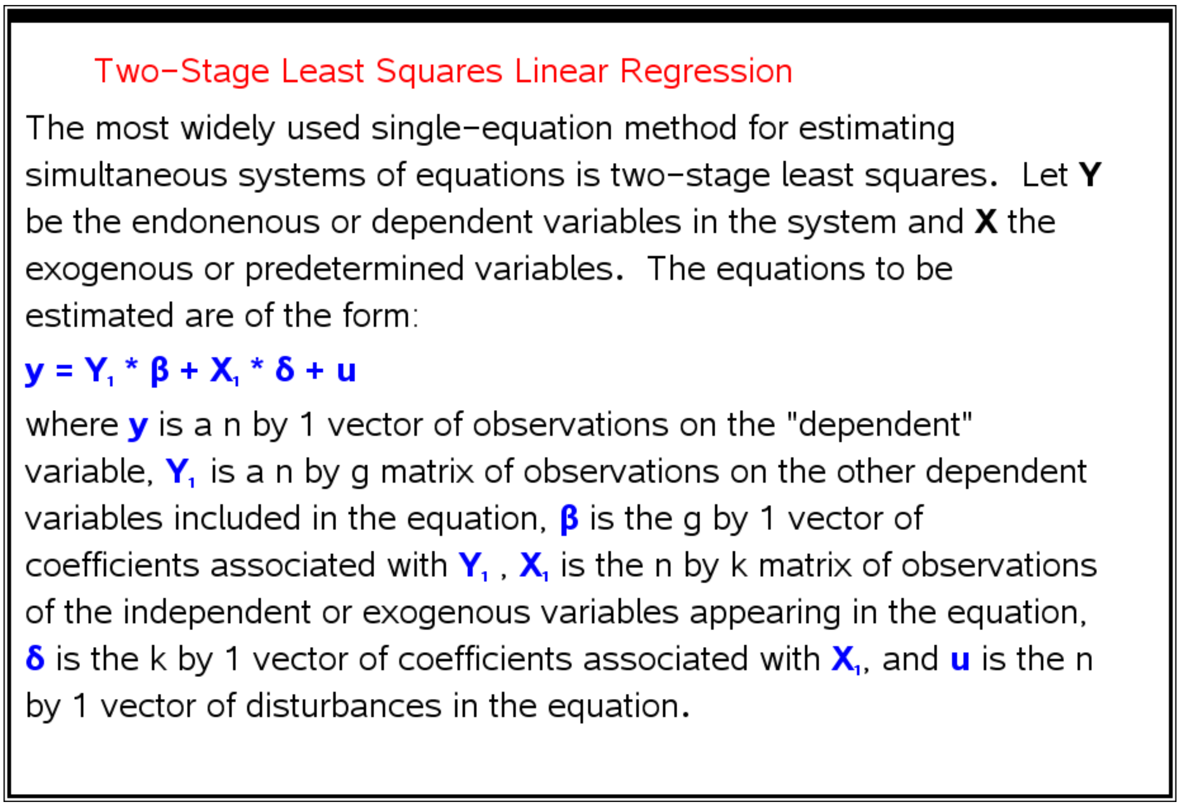
Functions *two_lines*(), *two_planes*(), *three_planes*() *and conv_plane*() will be discussed in DNL#114.

Die deutsche Version anaeo_LUA_d.tns ist in MTH113.zip enthalten.

[1] Geneviéve Savard's lecture: https://journal.ph-noe.ac.at/index.php/resource/issue/view/7

Don Phillips <don.phillips@gmail.com>

Josef,
Attached is my Two-Stage Least Squares program I had in DERIVE now ported to the TI-nSpire CX CAS calculator. Enjoy!

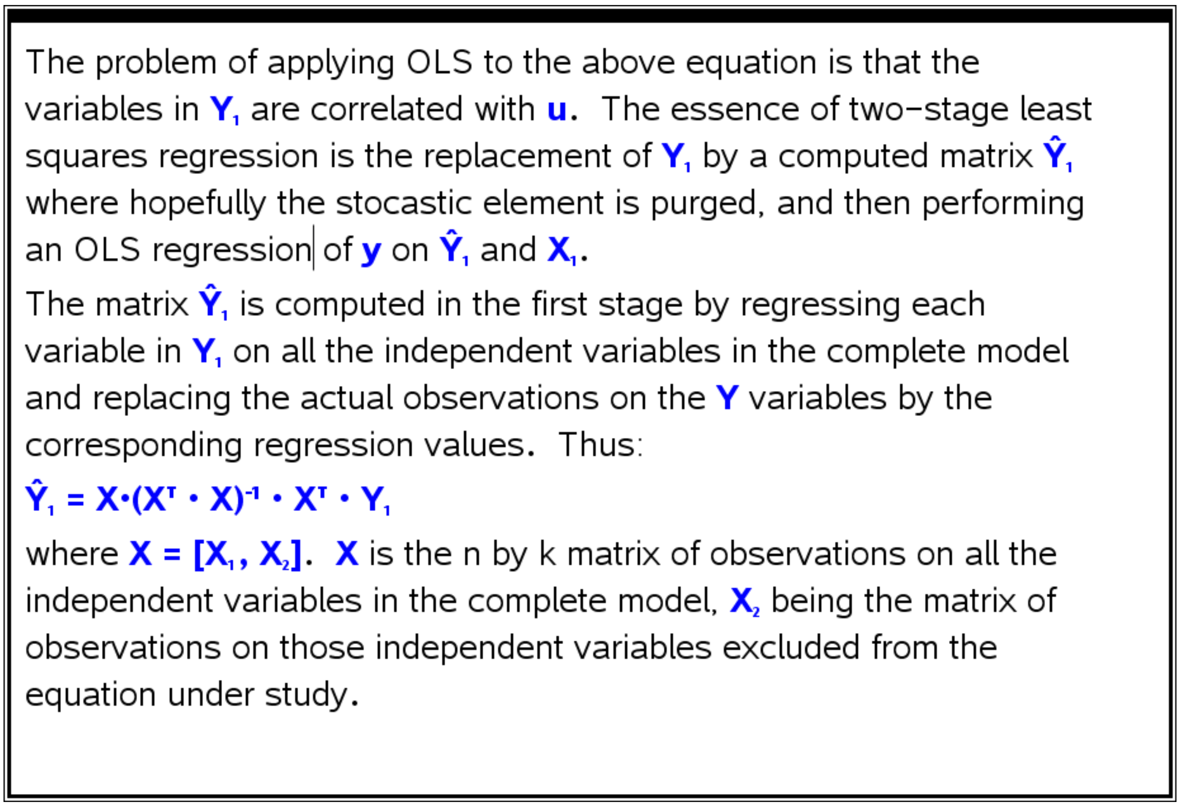


Two-Stage Least Squares Linear Regression

The most widely used single-equation method for estimating simultaneous systems of equations is two-stage least squares. Let **Y** be the endonenous or dependent variables in the system and **X** the exogenous or predetermined variables. The equations to be estimated are of the form:

$$\mathbf{y} = \mathbf{Y_1} * \boldsymbol{\beta} + \mathbf{X_1} * \boldsymbol{\delta} + \mathbf{u}$$

where **y** is a n by 1 vector of observations on the "dependent" variable, $\mathbf{Y_1}$ is a n by g matrix of observations on the other dependent variables included in the equation, $\boldsymbol{\beta}$ is the g by 1 vector of coefficients associated with $\mathbf{Y_1}$ , $\mathbf{X_1}$ is the n by k matrix of observations of the independent or exogenous variables appearing in the equation, $\boldsymbol{\delta}$ is the k by 1 vector of coefficients associated with $\mathbf{X_1}$, and **u** is the n by 1 vector of disturbances in the equation.

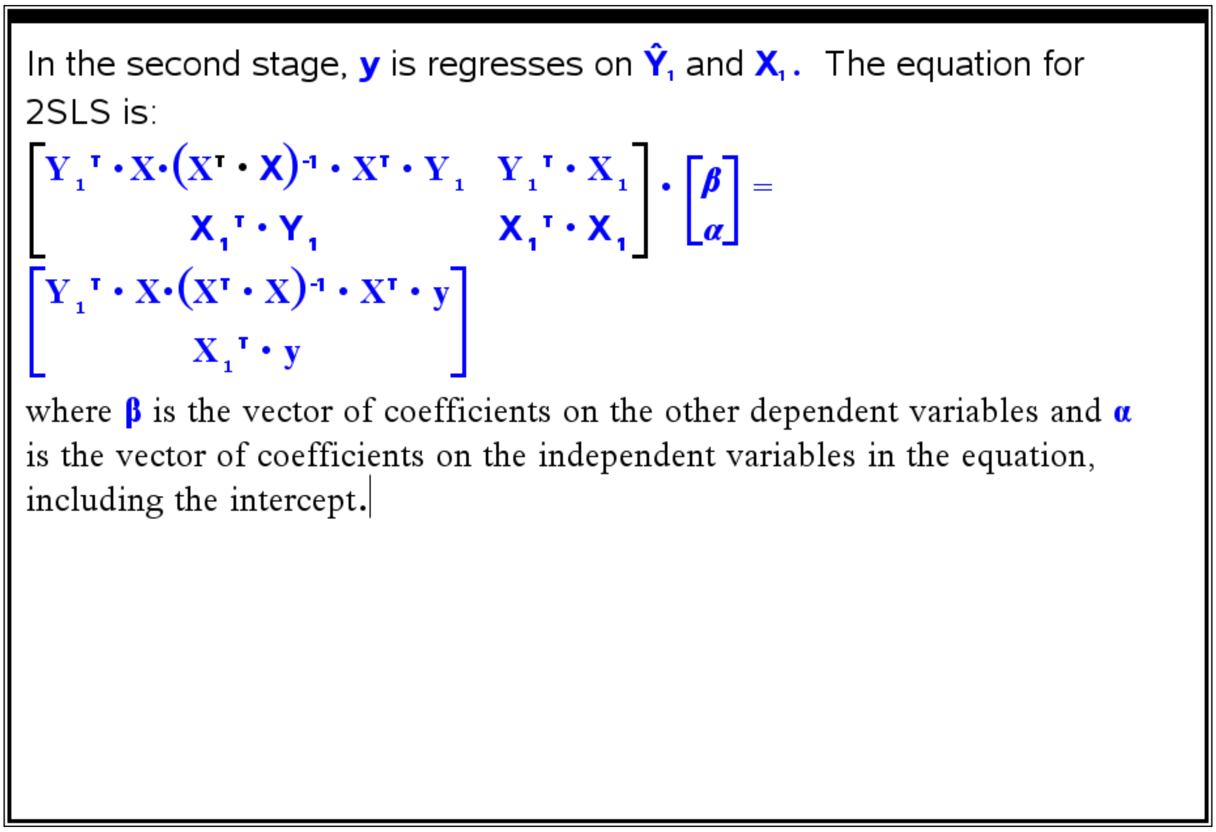


The problem of applying OLS to the above equation is that the variables in $\mathbf{Y_1}$ are correlated with **u**. The essence of two-stage least squares regression is the replacement of $\mathbf{Y_1}$ by a computed matrix $\mathbf{\hat{Y}_1}$ where hopefully the stocastic element is purged, and then performing an OLS regression of **y** on $\mathbf{\hat{Y}_1}$ and $\mathbf{X_1}$.

The matrix $\mathbf{\hat{Y}_1}$ is computed in the first stage by regressing each variable in $\mathbf{Y_1}$ on all the independent variables in the complete model and replacing the actual observations on the **Y** variables by the corresponding regression values. Thus:

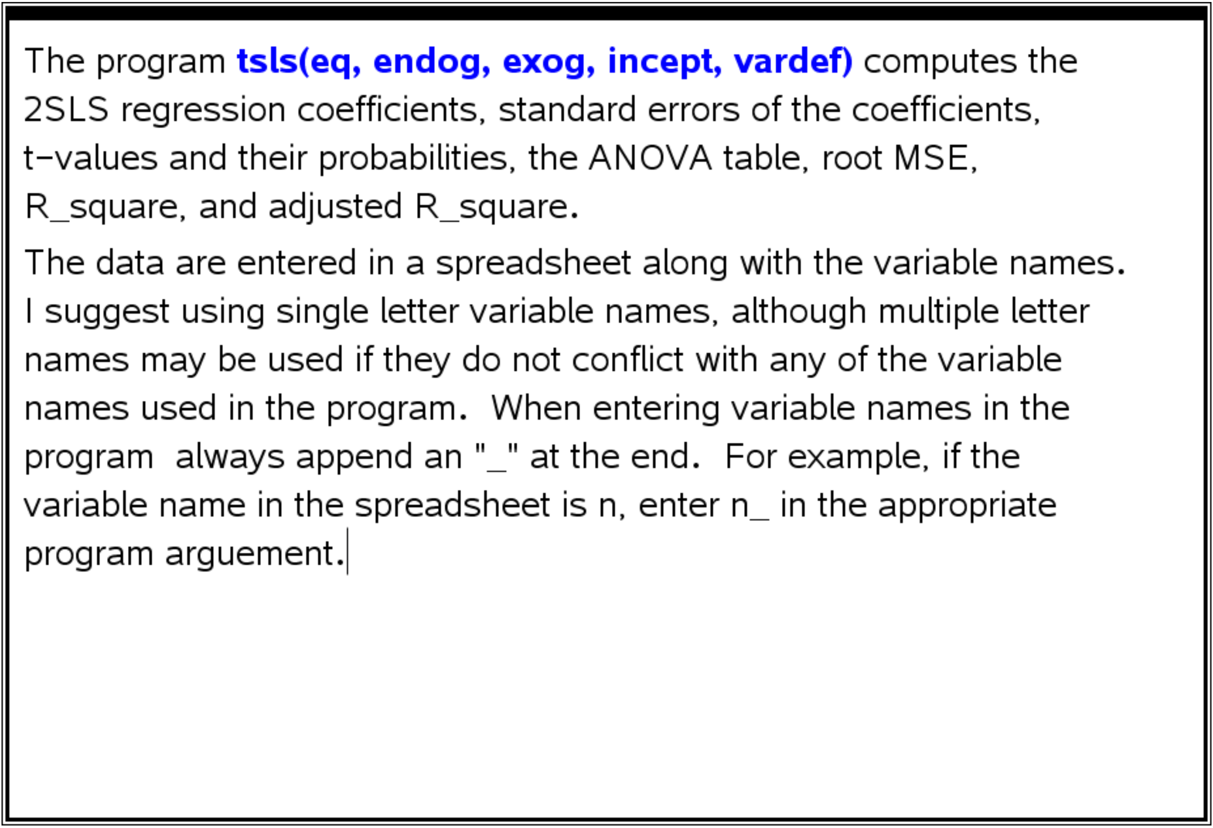
$$\mathbf{\hat{Y}_1} = \mathbf{X} \cdot (\mathbf{X}^T \cdot \mathbf{X})^{-1} \cdot \mathbf{X}^T \cdot \mathbf{Y_1}$$

where $\mathbf{X} = [\mathbf{X_1}, \mathbf{X_2}]$. **X** is the n by k matrix of observations on all the independent variables in the complete model, $\mathbf{X_2}$ being the matrix of observations on those independent variables excluded from the equation under study.

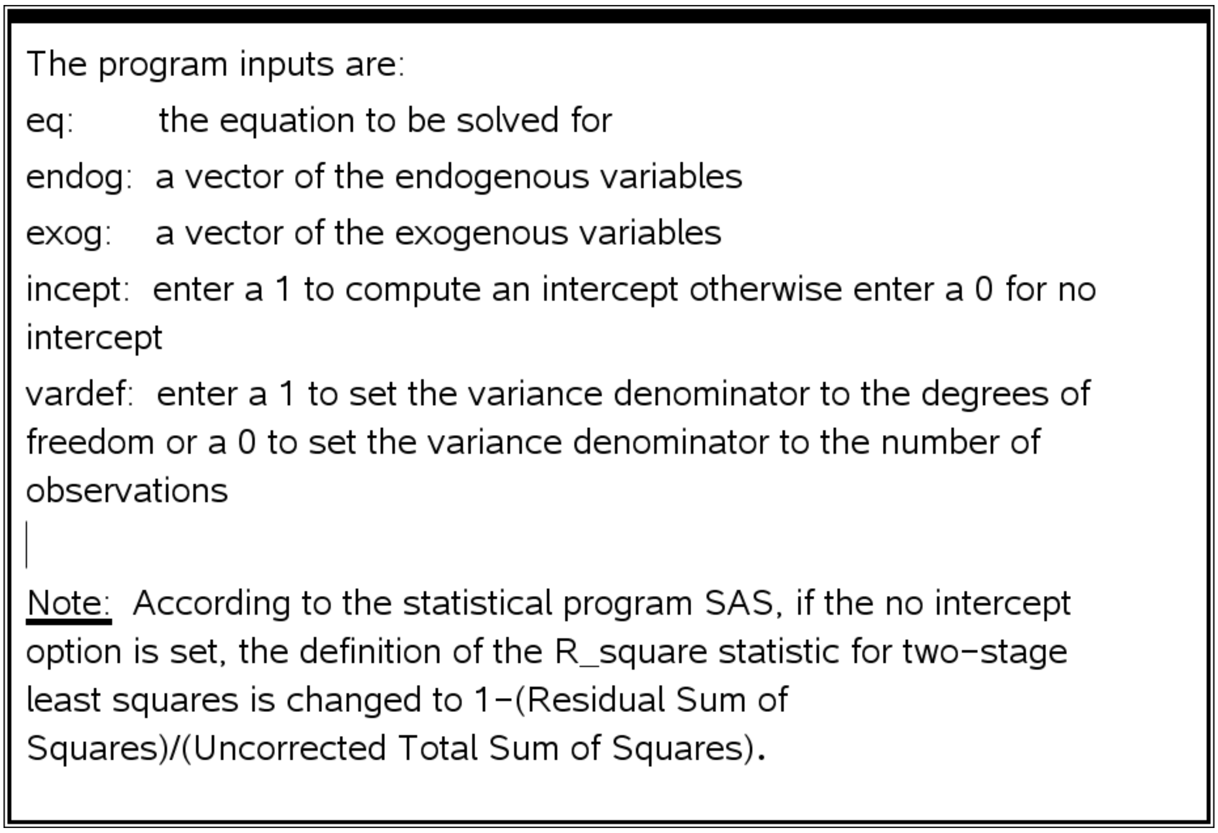

In the second stage, **y** is regresses on $\hat{\mathbf{Y}}_1$ and $\mathbf{X}_1$. The equation for 2SLS is:

$$\begin{bmatrix} Y_1^T \cdot X \cdot (X^T \cdot X)^{-1} \cdot X^T \cdot Y_1 & Y_1^T \cdot X_1 \\ X_1^T \cdot Y_1 & X_1^T \cdot X_1 \end{bmatrix} \cdot \begin{bmatrix} \beta \\ \alpha \end{bmatrix} = \begin{bmatrix} Y_1^T \cdot X \cdot (X^T \cdot X)^{-1} \cdot X^T \cdot y \\ X_1^T \cdot y \end{bmatrix}$$

where **β** is the vector of coefficients on the other dependent variables and **α** is the vector of coefficients on the independent variables in the equation, including the intercept.


The program **tsls(eq, endog, exog, incept, vardef)** computes the 2SLS regression coefficients, standard errors of the coefficients, t−values and their probabilities, the ANOVA table, root MSE, R_square, and adjusted R_square.

The data are entered in a spreadsheet along with the variable names. I suggest using single letter variable names, although multiple letter names may be used if they do not conflict with any of the variable names used in the program. When entering variable names in the program always append an "_" at the end. For example, if the variable name in the spreadsheet is n, enter n_ in the appropriate program arguement.



The program inputs are:

eq: the equation to be solved for

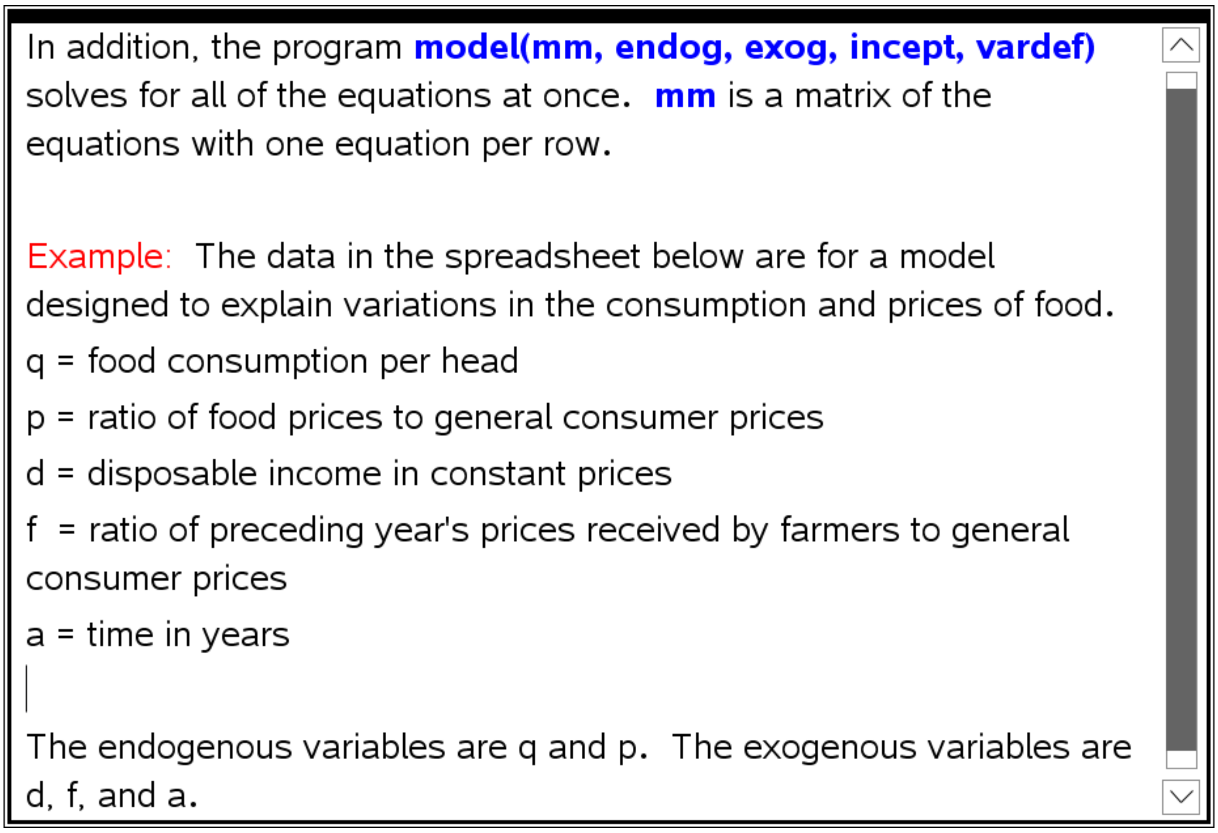
endog: a vector of the endogenous variables

exog: a vector of the exogenous variables

incept: enter a 1 to compute an intercept otherwise enter a 0 for no intercept

vardef: enter a 1 to set the variance denominator to the degrees of freedom or a 0 to set the variance denominator to the number of observations

Note: According to the statistical program SAS, if the no intercept option is set, the definition of the R_square statistic for two-stage least squares is changed to 1−(Residual Sum of Squares)/(Uncorrected Total Sum of Squares).



In addition, the program **model(mm, endog, exog, incept, vardef)** solves for all of the equations at once. **mm** is a matrix of the equations with one equation per row.

Example: The data in the spreadsheet below are for a model designed to explain variations in the consumption and prices of food.

q = food consumption per head

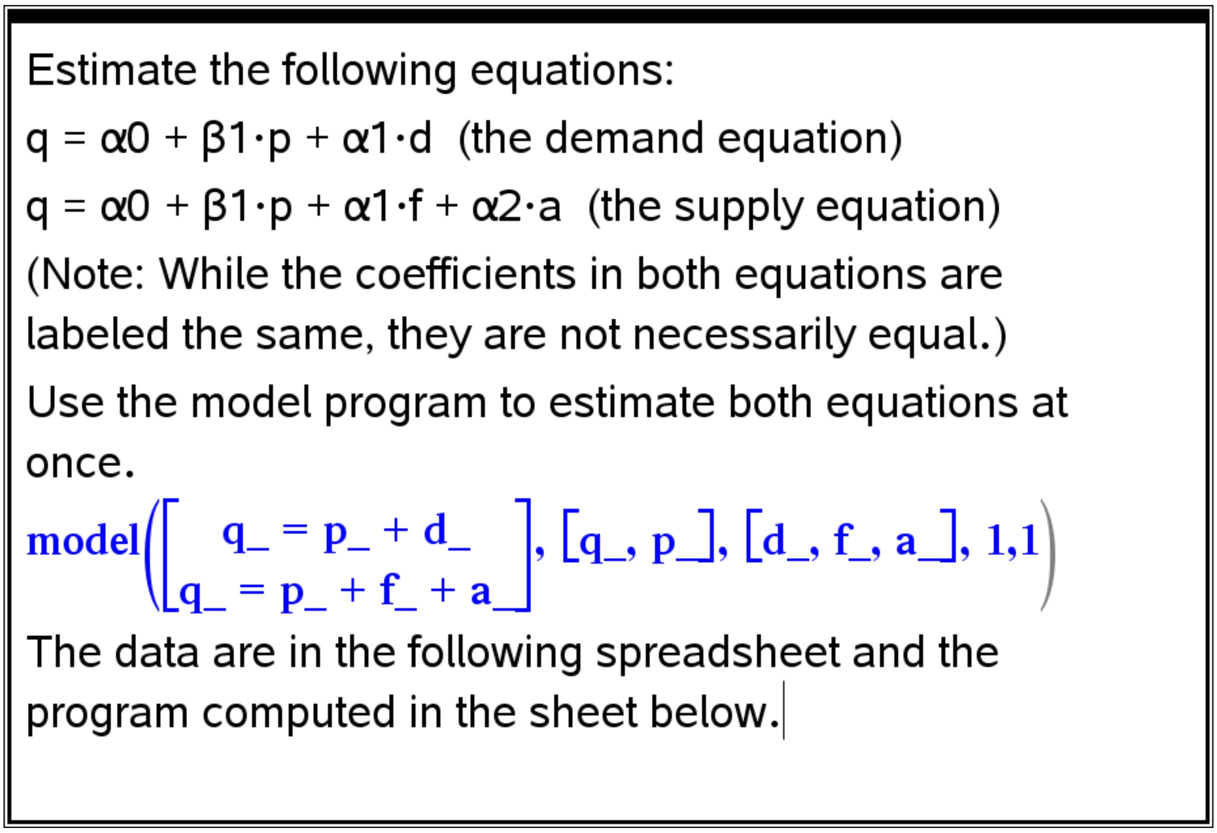
p = ratio of food prices to general consumer prices

d = disposable income in constant prices

f = ratio of preceding year's prices received by farmers to general consumer prices

a = time in years

The endogenous variables are q and p. The exogenous variables are d, f, and a.



Estimate the following equations:

q = α0 + β1·p + α1·d  (the demand equation)

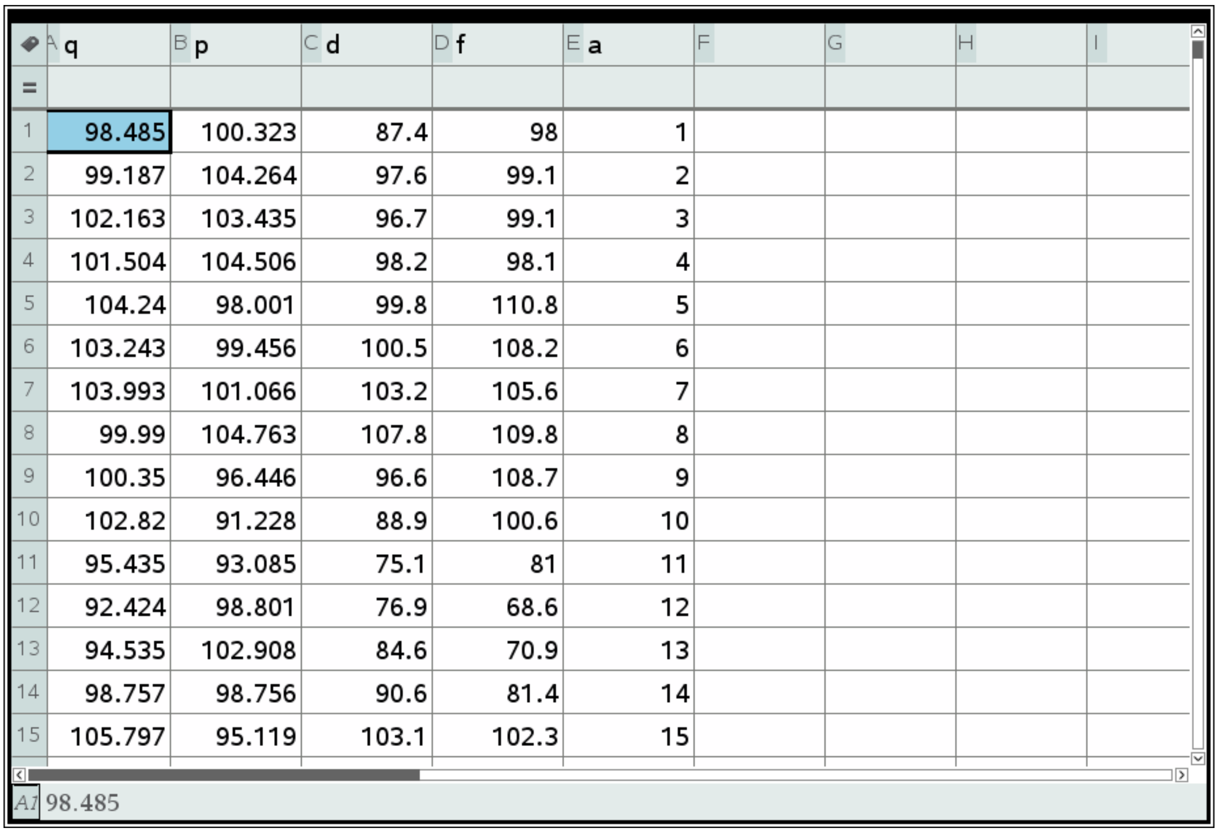
q = α0 + β1·p + α1·f + α2·a  (the supply equation)

(Note: While the coefficients in both equations are labeled the same, they are not necessarily equal.)

Use the model program to estimate both equations at once.

$$\text{model}\left(\begin{bmatrix} q\_ = p\_ + d\_ \\ q\_ = p\_ + f\_ + a\_ \end{bmatrix}, [q\_, p\_], [d\_, f\_, a\_], 1,1\right)$$

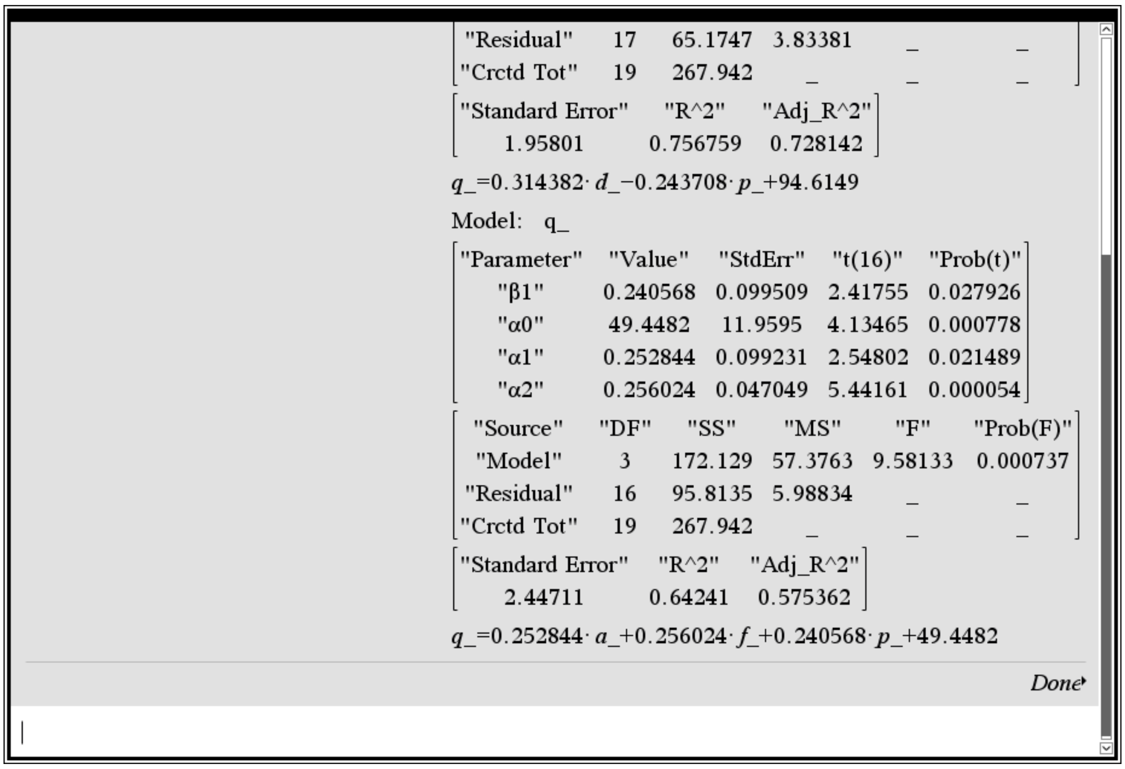
The data are in the following spreadsheet and the program computed in the sheet below.



| | A q | B p | C d | D f | E a | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| = | | | | | | | | | |
| 1 | 98.485 | 100.323 | 87.4 | 98 | 1 | | | | |
| 2 | 99.187 | 104.264 | 97.6 | 99.1 | 2 | | | | |
| 3 | 102.163 | 103.435 | 96.7 | 99.1 | 3 | | | | |
| 4 | 101.504 | 104.506 | 98.2 | 98.1 | 4 | | | | |
| 5 | 104.24 | 98.001 | 99.8 | 110.8 | 5 | | | | |
| 6 | 103.243 | 99.456 | 100.5 | 108.2 | 6 | | | | |
| 7 | 103.993 | 101.066 | 103.2 | 105.6 | 7 | | | | |
| 8 | 99.99 | 104.763 | 107.8 | 109.8 | 8 | | | | |
| 9 | 100.35 | 96.446 | 96.6 | 108.7 | 9 | | | | |
| 10 | 102.82 | 91.228 | 88.9 | 100.6 | 10 | | | | |
| 11 | 95.435 | 93.085 | 75.1 | 81 | 11 | | | | |
| 12 | 92.424 | 98.801 | 76.9 | 68.6 | 12 | | | | |
| 13 | 94.535 | 102.908 | 84.6 | 70.9 | 13 | | | | |
| 14 | 98.757 | 98.756 | 90.6 | 81.4 | 14 | | | | |
| 15 | 105.797 | 95.119 | 103.1 | 102.3 | 15 | | | | |

A1 98.485

The data and models are from Elements of Econometrics, Jan Kmenta, 1971, pp. 563-65.



```
 "Residual"    17   65.1747  3.83381    _        _
"Crctd Tot"    19   267.942     _       _        _

"Standard Error"   "R^2"     "Adj_R^2"
     1.95801      0.756759   0.728142

q_=0.314382·d_−0.243708·p_+94.6149
Model:  q_

"Parameter"  "Value"    "StdErr"   "t(16)"   "Prob(t)"
   "β1"      0.240568   0.099509   2.41755   0.027926
   "α0"      49.4482    11.9595    4.13465   0.000778
   "α1"      0.252844   0.099231   2.54802   0.021489
   "α2"      0.256024   0.047049   5.44161   0.000054

 "Source"   "DF"   "SS"      "MS"      "F"       "Prob(F)"
 "Model"     3    172.129   57.3763   9.58133   0.000737
 "Residual"  16   95.8135   5.98834     _          _
"Crctd Tot"  19   267.942     _         _          _

"Standard Error"  "R^2"     "Adj_R^2"
    2.44711      0.64241   0.575362

q_=0.252844·a_+0.256024·f_+0.240568·p_+49.4482

                                                   Done
```

In you have any questions about the program, I can be reached at don.phillips@gmail.com.

## 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.

Regards, DRL

**<u>David Sjöstrand, Onsala, Sweden</u>**

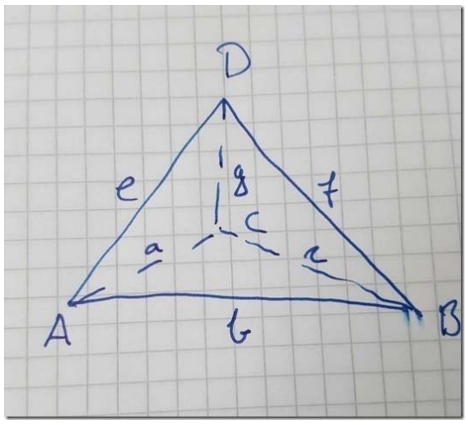
Hi Josef,

I hope that you and Noor are still going strong.

Yvonne and I are still quite OK.

I really like the Heron formula for the area of and I tried to find a nice corresponding formula for the volume of a tetrahedron with given side lengths.

I derived a formula with Derive for the volume tetrahedron (see sketch on the right).



```
V(a, b, c, e, f, g) :=

         4  2      2   2   2     2    2      2   2     2     2   2     2     4     2  2     4  2 ~
   √(- a ·f  - a ·(b ·(c  - f  - g ) - c ·(e  + f ) + e ·(g  - f ) + f  - f ·g ) - b ·g  ~
  ──────────────────────────────────────────────────────────────────────────────────────────~
                                                                                           ~

      2    2    2     2      2   2     2      2  2     4      2    2  2     4     2   2     2      2  2~
  + b ·(c ·(e  + g ) + e ·(g  - f ) + f ·g  - g ) - c ·(c ·e  + e  - e ·(f  + g ) + f ·g ~
  ──────────────────────────────────────────────────────────────────────────────────────────~
   12                                                                                      ~

  ))
  ──
```

```
                      2                          2
V(a, b, c, e, f, g)  - V(g, e, a, f, b, c)  = 0
```

If you permute the side lengths you in general receive different volumes. I have found out that if six positive numbers are given there are at most 30 different tetrahedrons with these side lengths. This is much more complicated than the corresponding proposition for triangles; if three positive numbers are given there is at most one triangle modulo congruent ones. It is quite fun to play around and find equalities like the one given above

Now I have started to consider this problem. Let six positive numbers be given. Is there a simple way to decide if there exists a tetrahedron having these side lengths?

I attach a dfw-file with a formula for the volume of the tetrahedron that you will find in this file.

Give my regards to Noor.
Greetings from Yvonne.
Best regards,
David

Dear David,

great to hear from you and also great to learn that you are still busy with mathematics.
I know some of my former colleagues who – since being retired – are not very interested in doing mathematics.

Your formula is really "great". I wonder if it might be possible to bring it in a "nice" form (rearranging the voluminous expressions, …).
Would you like to produce an article for the DNL – including how you derived the formula and the still open questions?

Why do you not use "d" for one side length?
One proposal: base sides a, b, c and the other edges d – opposite of a, e – opposite of b and f – opposite of c (for the given Tetrahedron).
This will change in case of permutation of the edges.

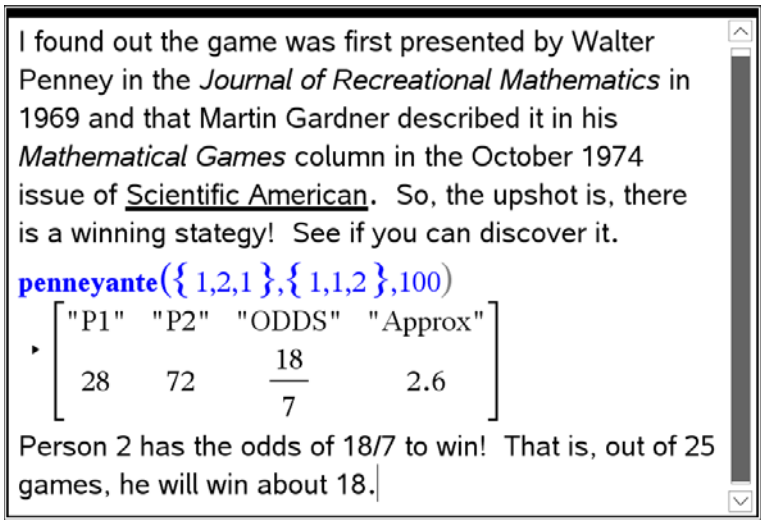
Dear Josef,

thank you for your answer.

I will spend more time with this and come back to you.

Best regards,
David

**Don Phillips sent:**

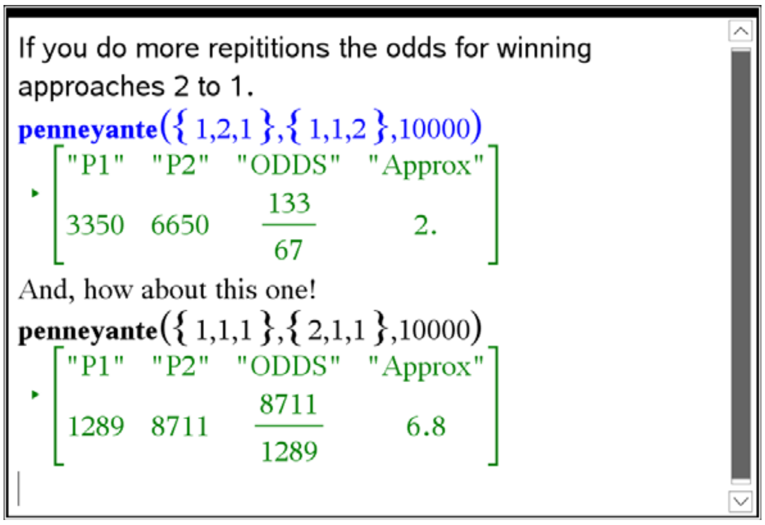


I found out the game was first presented by Walter Penney in the *Journal of Recreational Mathematics* in 1969 and that Martin Gardner described it in his *Mathematical Games* column in the October 1974 issue of Scientific American. So, the upshot is, there is a winning stategy! See if you can discover it.

**penneyante**$(\{1,2,1\},\{1,1,2\},100)$

$$\begin{bmatrix} \text{"P1"} & \text{"P2"} & \text{"ODDS"} & \text{"Approx"} \\ 28 & 72 & \frac{18}{7} & 2.6 \end{bmatrix}$$

Person 2 has the odds of 18/7 to win! That is, out of 25 games, he will win about 18.



If you do more repititions the odds for winning approaches 2 to 1.

**penneyante**$(\{1,2,1\},\{1,1,2\},10000)$

$$\begin{bmatrix} \text{"P1"} & \text{"P2"} & \text{"ODDS"} & \text{"Approx"} \\ 3350 & 6650 & \frac{133}{67} & 2. \end{bmatrix}$$

And, how about this one!

**penneyante**$(\{1,1,1\},\{2,1,1\},10000)$

$$\begin{bmatrix} \text{"P1"} & \text{"P2"} & \text{"ODDS"} & \text{"Approx"} \\ 1289 & 8711 & \frac{8711}{1289} & 6.8 \end{bmatrix}$$

**Mail from Steve Arnold, Australia**

Hey Josef - not sure if I mentioned my latest hobby to you - I am teaching myself JavaScript (or brushing up on the little I learned many years ago) and having a lovely time seeing what can be done for free in a browser on any platform! In particular, exploring the many ways in which GeoGebra can be adapted and used for learning and exploration within a web page.

My main project is called Live Mathematics and STEM on the Web:

https://compasstech.com.au/LIVE/

A key element are tutorials (as I did with Lua) which trace my own learning and hopefully may be helpful to others (teachers and even students) who may want to learn to develop their own "live" web pages.

Recent efforts have been

https://compasstech.com.au/ggb/

which accompany lessons 4 and 5:

https://compasstech.com.au/learn5/

Currently having fun with some musical explorations...

https://compasstech.com.au/piano/

which accompanies lesson 6:

https://compasstech.com.au/learn6/

So much can be done freely - to me, this lies at the heart of democratisation of technology in education - I love what TI offers, but great tools should not just be for those that can afford expensive devices..

I was amazed to discover that I could access BLE controls, internal device sensors (especially accelerometer, gyroscope and magnetometer/compass), GPS, and more.

So much to learn and to play with! I would be interested to hear your ideas...

With best wishes,
Steve

Dear Steve,
Many thanks for your mail and the very inspiring links to your "latest hobby" – or even better "your hobbies".
I would like to put your mail in the next DNL because of these links. In my opinion they should be shared within our community.
Josef

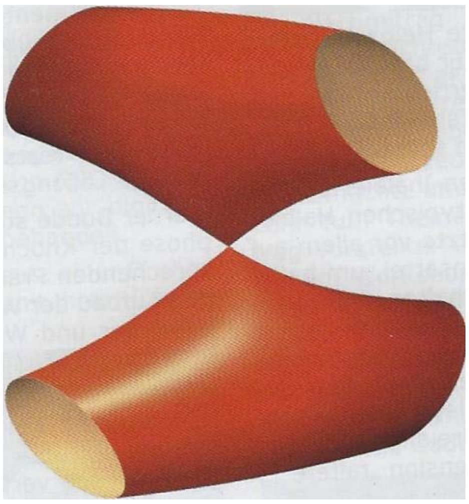
I would be very happy for you to include my links for your community. I want nothing more than for some to find something useful in my efforts!

With best wishes,
Steve

## An Interesting 3D-Locus – and how to plot it

Josef Böhm, Austria

At the bottom of the editorial of the latest issue of the IBDG (Informationsblätter der Geometrie) is the image of an interesting surface:



This 4th order surface is the locus of all points with constant product of the distances to two skew lines.

It is funny, but this picture fits wonderful to the Anageo-contribution. I wanted to reproduce the surface. This could be a nice problem for students and it was a nice problem for me to produce satisfying plots.

I start with *DERIVE* remembering the formula for the distance between a point and a straight line in space (page 25, third screen shot) and pour this formula into a function:

$$\text{dist}(p,\ v,\ q) := \left| q - p - \frac{v\cdot((q - p)\ \bullet\ v)}{|v|^2} \right|$$

$$\text{dist}([3,\ 4,\ 5],\ [1,\ -2,\ 4],\ [2,\ -4,\ 1]) = \frac{10\cdot\sqrt{357}}{21}$$

Given are the lines $l1: x = \begin{pmatrix}3\\5\\6\end{pmatrix} + t\cdot\begin{pmatrix}-1\\2\\0\end{pmatrix}$ and $l2: x = \begin{pmatrix}-1\\2\\4\end{pmatrix} + t\cdot\begin{pmatrix}3\\6\\1\end{pmatrix}$. What is the locus of all points with constant product of the distances to both lines?

To keep it not too hard for the students we will start with an easy task:

What is the locus of all points with a constant distance to line *l1?*

The answer shouldn't be too difficult.

```
Q := [x, y, z]

dist([3, 5, 6], [-1, 2, 0], Q)
```

$$\frac{\sqrt{5}\cdot\sqrt{(4\cdot x^2 + 4\cdot x\cdot(y - 11) + y^2 - 22\cdot y + 5\cdot z^2 - 60\cdot z + 301)}}{5} = C$$

$$\text{SOLVE}\left(\frac{\sqrt{5}\cdot\sqrt{(4\cdot x^2 + 4\cdot x\cdot(y - 11) + y^2 - 22\cdot y + 5\cdot z^2 - 60\cdot z + 301)}}{5} = C,\ z\right)$$

$$z = \text{IF}\left(C > 0,\ \frac{\sqrt{5}\cdot(6\cdot\sqrt{5} - \sqrt{(- 4\cdot x^2 + 4\cdot x\cdot(11 - y) - y^2 + 22\cdot y + 5\cdot C^2 - 121)})}{5}\right) \vee z = \text{IF}\left(C > 0,\ \frac{\sqrt{5}\cdot(\sqrt{(- 4\cdot x^2 + 4\cdot x\cdot(11 - y) - y^2 + 22\cdot y + 5\cdot C^2 - 121)} + 6\cdot\sqrt{5})}{5}\right)$$

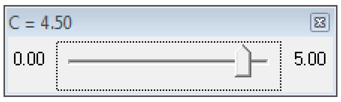
```
[3 - t, 5 + 2·t, 6]
```

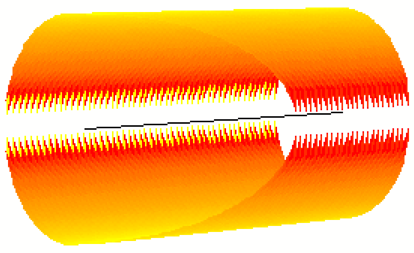
The locus of point Q (with variable distance *C* from the line) is given in its implicit form in the third expression. Unfortunately, we cannot perform implicit plots with DERIVE, so we try to find explicit expression(s) for z.

The two expressions give upper and lower half of the expected cylinder. Problems of representations occur in vertical regions of the surface. I add the graph of the line. The slider changes the radius of the cylinder.

Now they should be ready to find the requested locus!

One single line solves the problem (supported by a CAS).





One single line solves the problem (supported by a CAS).

```
dist([3, 5, 6], [-1, 2, 0], Q)·dist([-2, 1, -4], [3, 6, 1], Q) = C
```

Inspecting the result, we can be happy to have a CAS at our disposal.

$$\frac{\sqrt{230}\cdot\sqrt{(4\cdot x^2 + 4\cdot x\cdot(y - 11) + y^2 - 22\cdot y + 5\cdot z^2 - 60\cdot z + 301)}\cdot\sqrt{(37\cdot x^2 - 2\cdot x\sim}{230\ \sim}$$

$$\frac{\cdot(18\cdot y + 3\cdot z - 80) + 10\cdot y^2 - 4\cdot y\cdot(3\cdot z + 35) + 5\cdot(9\cdot z^2 + 72\cdot z + 190))}{} = C$$

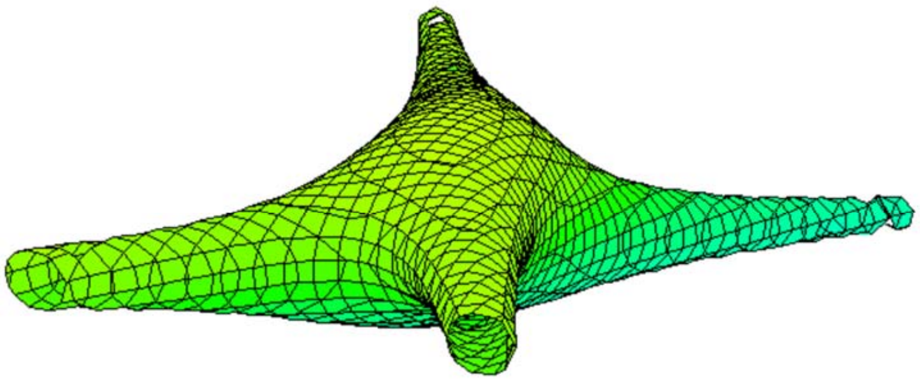
We can square the whole expression and we see, that we receive a polynomial of order 4 which gives us no hope to achieve any explicit form. What to do?

One idea is – if you have DPGraph installed – clicking on the DPGraph button in the menu bar (Window > Customize > Commands > DPGRaph).

It needs s little bit of editing in DPGraph :

```
graph3d.minimumx:=-60
graph3d.maximumx:=60
graph3d.minimumy:=-60
graph3d.maximumy:=60
graph3d.minimumz:=-60
graph3d.maximumz:=60
graph3d.resolution:=50
graph3d.box:=false
c.minimum:=10
c.maximum:=200
graph3d((4*x^2+4*x*(y-11)+y^2-22*y+5*z^2-60*z+301)*(37*x^2-
2*x*(18*y+3*z-80)+10*y^2-4*y*(3*z+35)+5*(9*z^2+72*z+190))/230=C^2)
```

The most important command – the last one - has been transferred from DERIVE to DPGrasp. The others must be added:



Later we will find another way for plotting the surface – using a wonderful free software, be patient for a little while.

The picture is quite pretty, but looks different to graph shown above!

I try with two other lines taking special positions: $l1: x = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} + t \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$ and $l2: x = \begin{pmatrix} 5 \\ 0 \\ o \end{pmatrix} + t \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$ and hope to get expressions which can easier be handled.

$$\mathrm{dist}([0,\ 0,\ 0],\ [0,\ 0,\ 1],\ Q)\cdot\mathrm{dist}([5,\ 0,\ 0],\ [0,\ 1,\ 0],\ Q) = C$$

$$\sqrt{(x^2 + y^2)}\cdot\sqrt{(x^2 - 10\cdot x + z^2 + 25)} = C$$

$$(\sqrt{(x^2 + y^2)}\cdot\sqrt{(x^2 - 10\cdot x + z^2 + 25)} = C)^2$$

$$(x^2 + y^2)\cdot(x^2 - 10\cdot x + z^2 + 25) = C^2$$

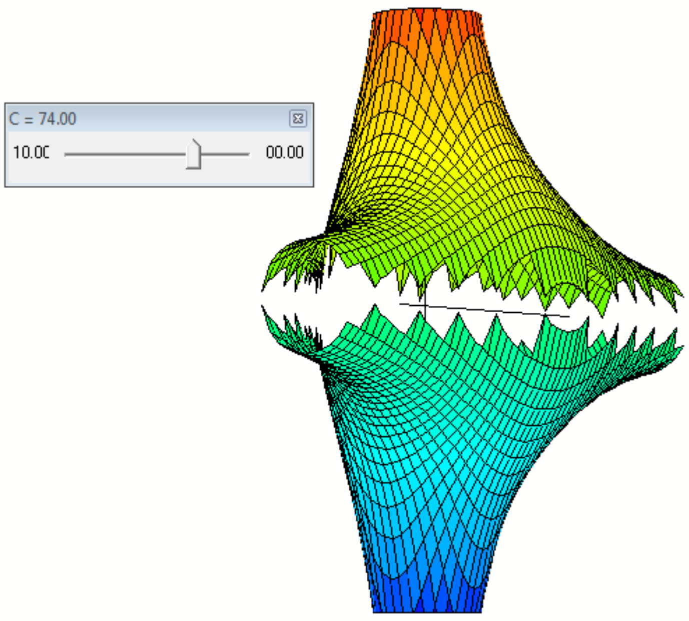
Heureka, this equation can be solved for *z*: We do it, plot the two halves of the surface together with the graphs of the two lines. A slider for constant product *C* gives an impression how the solid can change.

$$\mathrm{SOLUTIONS}((x^2 + y^2)\cdot(x^2 - 10\cdot x + z^2 + 25) = C^2,\ z)$$

$$\left[\frac{\sqrt{(-x^4 + 10\cdot x^3 - x^2\cdot(y^2 + 25) + 10\cdot x\cdot y^2 - 25\cdot y^2 + C^2)}}{\sqrt{(x^2 + y^2)}},\ -\frac{\sqrt{(-x^4 + 10\cdot x^3 - x^2\cdot(y^2 + 25) + 10\cdot x\cdot y^2 - 25\cdot y^2 + C^2)}}{\sqrt{(x^2 + y^2)}}\right]$$

$$[0,\ 0,\ t]$$

$$[5,\ t,\ 0]$$



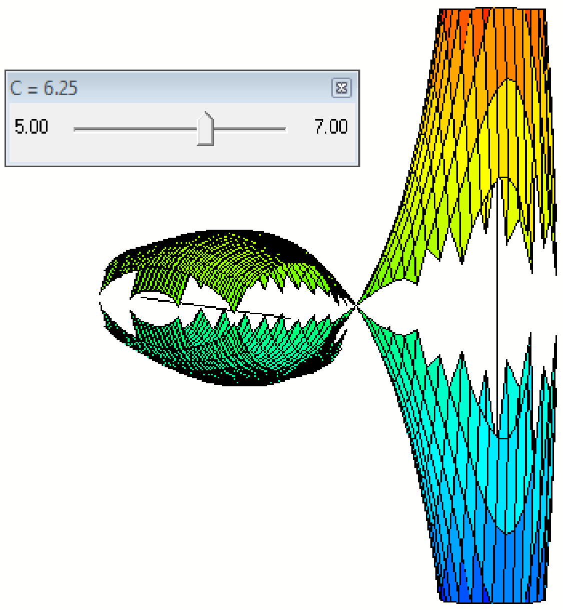
Ok, nice again, but it is not the surface with the singularity – the point, where the two tubes are touching each other.

Can it be, that it needs a special value for *C*?

Let the students reason and discuss! (Same what I did, discussing with myself!)

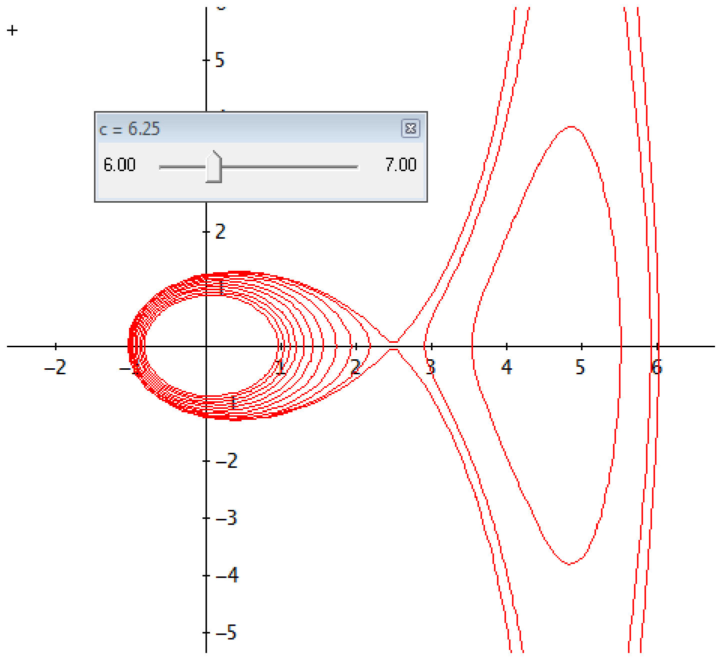
I am sure that you got it? *C* is half of the shortest distance between the two lines.

This is 2.5 for our lines in special positions. So, $C = 2.5^2 = 6.25$.



Now we can be satisfied, because we can see the singularity very clear. Before producing a better graph – not with DPGraph, we can insert a short exercise: Let's plot the level curves of the surface with variable values for *C*:

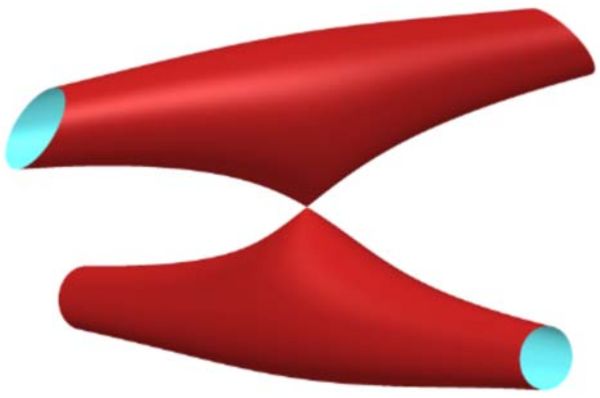
```
            2     2     2            2          2
VECTOR((x  + y ) · (x  − 10 · x + z  + 25) = C , z, 0, 5, 0.5)
```



Finally, I come back to a wonderful piece of software – Surfer - , which can be downloaded free of charge:

https://imaginary.org/ and https://imaginary.org/de

*Surfer* is the miracle worker:

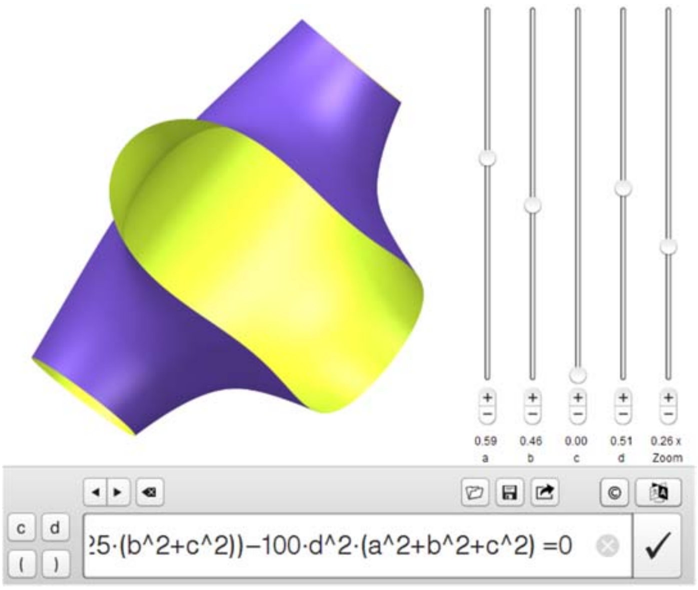


Hello, that's very similar to the picture from IBDG. It seems to be that the author is also a *Surfer*.

I define the direction vector of the second line as [*a*,*b*,*c*] and the constant product as *d*. Then I am free to *surf* with *Surfer* through four parameters and enjoy the metamorphoses (value of sliders only between 0 and 1 possible → 100*d*).

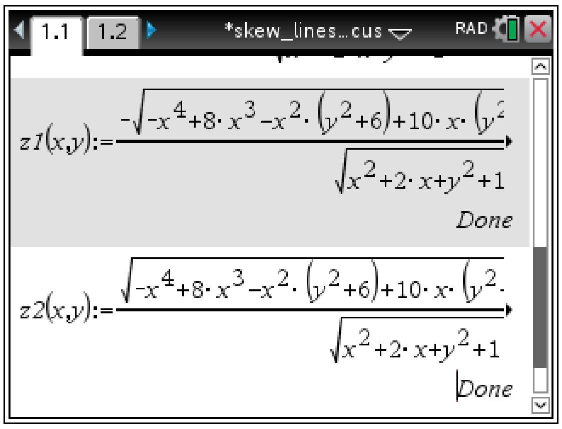
This is in my opinion a great example of meaningful use of a CAS: we know what and how to do, but we hesitate or we waive the nasty and boring calculation.

The *DERIVE* plot including sliders is shown on page 18.



²5·(b^2+c^2))−100·d^2·(a^2+b^2+c^2) =0

Working with TI-NspireCAS offers the same advantage (CAS) and shortcoming (implicit plotting and splitting in two halves with problems in the regions near vertical parts of the surface as *DERIVE*), which in fact is not really a surprise.

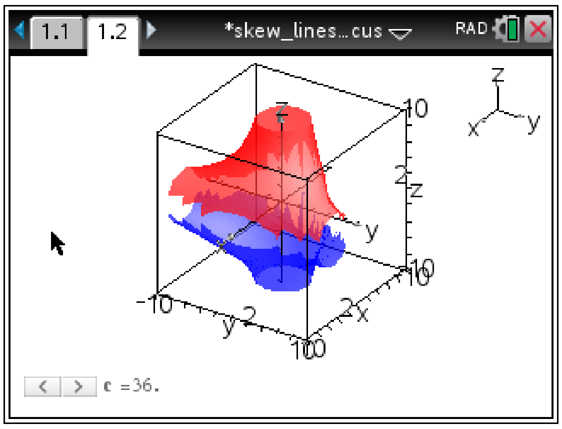


$$z1(x,y):=\frac{-\sqrt{-x^4+8\cdot x^3-x^2\cdot\left(y^2+6\right)+10\cdot x\cdot\left(y^2\right.}}{\sqrt{x^2+2\cdot x+y^2+1}}$$

*Done*

$$z2(x,y):=\frac{\sqrt{-x^4+8\cdot x^3-x^2\cdot\left(y^2+6\right)+10\cdot x\cdot\left(y^2\right.}}{\sqrt{x^2+2\cdot x+y^2+1}}$$

*Done*


1.1
1.2
*skew_lines...cus
RAD
z
x
y
10
2
z
y
10
-10
y
2
2
x
10
c =36.