THE DERIVE - NEWSLETTER #31

ISSN 1990-7079

THE BULLETIN OF THE

DERIVE®

USER GROUP + TI 92

Contents:

revised version 2013 **September 1998**

[1] **Lineare Gleichungssysteme lösen mit *DERIVE forWindows***, B Kutzler, bk-teachware

[2] **Lineare Gleichungssysteme lösen mit dem TI-92,** B Kutzler, bk-teachware

[3] **Optimierungsaufgaben, graphisch, numerisch und analytisch mit dem *TI*-92,** J Böhm, bk-teachware

[4] **Solving Systems of Linear Equations with *DERIVE for Windows***, B Kutzler, bk-teachware

[5] **Solving Systems of Linear Equations with the *TI*-92,** B Kutzler, bk-teachware

bk-teachware, Softwarepark, A-4232 Hagenberg, Austria, email: bk.teachware@swp.co.at
Chartwell-Yorke, 114 High Street, Belmont, Bolton, UK, email: Philip.Yorke@chartwellyorke.com

----

AUTHORS SOUGHT

Authors are sought for the bk teachware series: "Support in Learning". Each text is written for the average, technology inexperienced teacher, explaining how to use one of the following tools for teaching: TI-92, TI-89, CBL, CBR, DERIVE, CABRI GEOMETRE II.

We offer:
- quick publication, professional production, attractive sales prices
- 15% of net income as royalty
- publication with ISBN number
- experienced editors: Brian Denton (Editor-in-Chief) Terence Etchells (Assistant Editor) Philip Yorke (Desk Editor)
- marketing through distributors in the UK, USA, Scandinavia, and Benelux
- option for translating and marketing in other language regions (German, French, Italian, Japanese, ...)

The booklets of the bk teachware series have a format of 17 x 24 cm and comprise 32-64 pages. A diskette can be attached to the text. They are priced between 4 and 10 Pounds.

One booklet should treat one topic, ideally should be self-contained, and should describe a lesson in all details. The following booklets can serve as models for your own:-

SL-03: Solving Linear Equations with the TI-92
SL-04: Solving Systems of Linear Equations with the TI-92
SL-05: Solving Linear Equations with Derive for Windows
SL-06: Solving Systems of Linear Equations with Derive for Windows

We would be pleased to send you sample pages from one of these booklets. Please contact the desk editor:

Philip Yorke: Philip.Yorke@ChartwellYorke.com

Please send manuscripts to the Editor-in-Chief:-

Brian Denton: B.H.Denton@livjm.ac.uk

Yours sincerely

Bernhard Kutzler
Publisher

Interesting Web Sites: (only the red ones are still valid!)

http://cs.gettyburg.edu/~leinbach/DRVTI_Conf/ConfPrg.html	Three keynotes from the Gettysburg Conference
http://hutchinson.belmont.ma.us/tth/tth.html	A tool to translate TEX into HTML (free!!)
http://www.imaxx.net/~gdorner/	G Dorner´s pdf-files concerning a thrilling TI-92 book
http://www.brunel.ac.uk/~mastmmg/ssguide/sshome.htm	An Online Study Skills guide
http://www.bham.ac.uk/ctimath	Always interesting
http://www.msri.org/realvideo/	100 lectures (recommended by cti Birmingham)
http://www.ti.com/calc/docs/t3ww	The T^3 site (Teachers Teaching with Technology)

Liebe DUG Mitglieder,

In der Nummer 1 nach Gettysburg möchte ich mich vorerst auch an dieser Stelle im Namen aller Teilnehmer bei den Organisatoren der diesjährigen Konferenz für ihren Einsatz recht herz-lich bedanken. Pat und Carl Leinbach haben es verstanden, die Tage vom 15.- 17.Juli zu einem Fest für die DERIVE & TI-92 Familie zu machen. Beth und ChristyAnn haben versucht, jeden Wunsch zu erfüllen (sogar den absurden Wunsch nach 7 Tennisschlägern!! 1 ≠ 7. "You crazy Europeans with your 1", Carl L.). Leider konnte Bert Waits nicht bis zum Ende bleiben - er versäumte ein großartiges Picknick auf Old Mac Leinbachs Farm. Wir haben ihn einige Zeit später in seinem "Paradies" auf Seabrook Island besuchen dürfen - noch vor dem Wirbelsturm Bonnie. (Bert, dein Sanddollar ist noch immer ganz.)

In dieser Nummer ist noch ein Beitrag vom 1997 Symposium in Särö zu finden: Terence Etchells zieht auf spitzfindige Weise aus numerischen Daten Schlüsse auf deren logischen Zusammenhang. Ich habe ganz bewusst den Abschluss von D. Halprins Glove Osculants für die nächste Ausgabe aufgehoben, denn ich möchte, dass Sie seinen Beitrag bequem anhand der files auf der Diskette des Jahres 1998 nachvollziehen können. Aus dem gleichen Grund habe ich einen geplanten Artikel von S. Abraham Ibrahim über "Modified Bessel Functions" in den DNL#32 verschoben. Es wird auch noch einen Beitrag von S. Welke geben, der zeigt, wie sich DERIVE-Objekte auch ohne ACROSPIN animieren lassen.

Ich möchte Sie hier auf die Möglichkeit hinweisen, eine preiswerte CD-ROM mit den Vorträgen und Workshops der 3.Internationalen DERIVE und TI-92 Konferenz bei Jerry Glynns MathWare zu beziehen (Seite 8).

Viele liebe Grüße

Josef Böhm

Dear DERIVE and TI-friends,

This is issue #1 after Gettysburg, and on behalf of all delegates I want to express my thanks to the organizers for this tremendous event. Thanks to Pat and Carl Leinbach, the days from 15 - 17 July became a festival for the *DERIVE* & *TI 92* family. Beth and ChristyAnn actually fullfilled each and every wish (even the absurd request for 7 tennis rackets!! 1 ≠ 7. "You crazy Europeans with your 1", Carl L.). Unfortunately Bert Waits could not stay until the end - he missed a great Picknick at Old Mc Leinbach´s Farm. We visited him and his wife in their "Paradise" on Seabrook Island - long before Bonnie, the hurricane. (Bert, your sanddollar is still unhurt.)

In this issue you will find the final contribution from the 1997 Symposium in Särö. Terence Etchells draws logical conclusions from numerical data in a very inspiring and tricky way. I leave the last part of D. Halprin´s Glove Osculants for the next issue, because I think having the Diskette of the Year 1998 in your hands - or drive -, it will be much easier for you to reproduce his ideas. For the same reason I have shifted a planned article from S. Abraham Ibrahim about "Modified Bessel Functions" to DNL#32. There will also be a nice contribution from S.Welke, showing how to animate *DERIVE* objects even without ACROSPIN.

I´d like to point out that there is opportunity to purchase an inexpensive CD-ROM containing the lectures and workshops of the 3rd International *DERIVE* and *TI*-92 Conference from Jerry Glynn´s MathWare (page 8).

With best regards

Josef Böhm

The *DERIVE-NEWSLETTER* is the Bulletin of the *DERIVE User Group*. It is published at least four times a year with a contents of 44 pages minimum. The goals of the *DNL* are to enable the exchange of experiences made with *DERIVE* as well as to create a group to discuss the possibilities of new methodical and didactical manners in teaching mathematics.
We have established a section dealing with the use of the *TI*-92 and we try to combine these modern technologies.

Editor: Mag. Josef Böhm
A-3042 Würmla
D´Lust 1
Austria
Phone: 43-(0)660 3136 365
email: nojo.boehm@pgv.at

email: josef.boehm@blackboard.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 *DNL*. The more contributions you will send, the more lively and richer in contents the *DERIVE Newsletter* will be.

Next issue: December 1998
Deadline: 15 November 1998

Preview: Contributions for the next issues

3D-Geometry, Reichel, AUT
Linear Programming, Various Projections, Word Problems, Böhm, AUT
A Utility file for complex dynamic systems, Lechner, AUT
Examples for Statistics, Roeloffs, NL
Fractals and other Graphics, Koth, AUT
Implicit Multivalue Bivariate Function 3D Plots, Biryukov, RUS
Riemann, a package for the TI-92, Böhm & Pröpper, AUT/GER
Quaternion Algebra, Sirota, RUS
Parametric 3D-Plots with DERIVE and 3DV, Welke, GER
Various Training Programs for Students on the TI-92, Böhm, AUT
A Critical Comment on the "Delayed Assignation" := =, Kümmel, GER
Sand Dunes, River Meander and Elastica, The lighter side ...., Halprin, AUS
Type checking, finite continued fractions, ..., Welke, GER
Evaluation of the Modified Bessel Functions, Ibrahim, ESP
On the Resolution of the Linear Differential Equation ......, Candel, ESP
TI-92 Statistics with *DERIVE*, Böhm, AUT

and
Setif, FRA; Vermeylen, BEL; Leinbach, USA; Weth, GER; Kirmse, GER
Aue, GER; Koller, AUT; Mitic, UK; Schorn, GER; Santonja, ESP;
Dorner, USA, Pröpper, GER and and and ....

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

CONCENTRIC CURVE SHADING

G P Speck, Wanganui, New Zealand

In the article "Broken Lines and Shaded Areas" in DNL#17 on pp 16-18, Josef Böhm gave a very simple elegant `SHADE( )` function which can be used to shade a region with line segments parallel to the *y*-axis, where the region is bounded by curves of the form $y = F(x)$ and $y = G(x)$. Similarly, a trivial modification of the `SHADE( )` function gives a function which can be used to shade a region with line segments parallel to the *x*-axis, where the region is bounded by curves of the form $x = F(y)$ and $x = G(y)$.

Of course, we can make the transparent observation that if a few marks (such as x's) in a region are an adequate pointer to the region for some particular purpose, then these few marks can be placed in the region easily by devices such as the (Ctrl) + (A) annotation device of *DERIVE for Windows* (*DfW*).

In this article, shading regions bounded by curves given parametrically with concentric curves is considered both as a way of creating a rather artistic pointer to a region, as well as a topic of interest in and of itself. Of course, curves given parametrically are, in general, much more complex than curves given in the form $y = F(x)$ or $x = G(y)$. And even though it may be theoretically possible (in some cases) to reduce a given parametrization to a finite number of curves of the form $y = F(x)$ or $y = G(x)$, it may be EXTREMELY tedious to try to actually produce such a reduction, even with the aid of software such as *DERIVE*.

Herein we will consider five fairly elementary parametrizations in SHADING1.MTH through SHADING5.MTH. These MTH file names will serve both as our names for our parametric curves, which will be given first, as well as the names for the files which show how the curves with shading were constructed, which we will give together with the graphs.

The regions bounded by parametric curves in SHADING1.MTH, SHADING2.MTH and SHADING3.MTH are convex sets, for which our method of concentric curve shading will be seen to be appropriate since any ray from an interior point of a convex set intercepts the boundary in one and only one point. Thus, though the regions which we wish to shade in SHADING4.MTH and SHADING5.MTH are not convex, our method of concentric shading will still be seen to be appropriate since, in each case, there exists a particular point interior to the non-convex region such that each ray from the point intercepts the boundary in one and only one point.

Also, since my original version of *DfW* will not PLOT curves given via `IF( )` statements (a real shortcoming!), I have defined characteristic functions which enable me to define piecewise parametrizations without the use of `IF( )` statements. (Incidentally, I have found these characteristic functions to be very useful elsewhere - e.g., in shoring up *DERIVE* shortcomings in limit evaluations.)

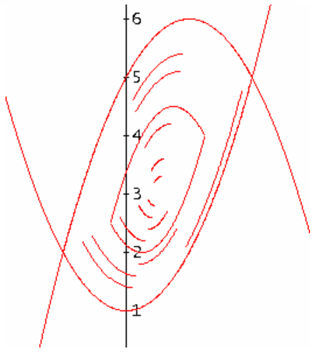
My self-explanatory (hopefully) examples follow.

```
#1:   SHAD_UT.MTH

      [ Method via Characteristic Functions for my version of DfW ]
#2:   [                                                           ]
      [  which will not plot functions given via IF() statements  ]

#3:   CHARACTERISTIC FUNCTIONS

#4:   α(x) := FLOOR(x + 1)·χ(0, x + 1, ∞) - FLOOR(x)·χ(0, x, ∞)

#5:   CHPOINT(x, a) := α(x - a)·α(a - x)

#6:   CHGRTEQ(x, a) := α(x - a)

#7:   CHGRT(x, a) := α(x - a) - CHPOINT(x, a)

#8:   CHLESS(x, a) := 1 - α(x - a)

#9:   CHUNEQ(x, a) := CHLESS(x, a) + CHGRT(x, a)

#10:  CHLESSEQ(x, a) := 1 - α(x - a) + CHPOINT(x, a)

#11:  CH_01_CL(x) := α(x)·α(1 - x)

                                  ⎛   1           ⎞
#12:  CHCLOSED(x, a, b) := CH_01_CL⎜ ───────·(x - a)⎟
                                  ⎝ b - a         ⎠

#13:  CHOPENRT(x, a, b) := CHCLOSED(x, a, b) - CHPOINT(x, b)

#14:  CHOPENLF(x, a, b) := CHCLOSED(x, a, b) - CHPOINT(x, a)

#15:  CHOPEN(x, a, b) := CHCLOSED(x, a, b) - CHPOINT(x, b) - CHPOINT(x, a)
```

I try applying G P Speck's shading utility from 1998 with DERIVE 6.1:

```
#1:   SHADING1.MTH - first preload SHAD_UT.MTH

      [ Concentric curve shading of a region bounded by parametric curve ]
#2:   [                                                                  ]
      [     made up of portions of y = 1 + x^2 and y = - x^2 + 2x + 5     ]

      [      X11(t) := ¿·CHLESS(t, -1)       ]
      [                                      ]
      [    X12(t) := t·CHCLOSED(t, -1, 2)    ]
#3:   [                                      ]
      [ X13(t) := (-t + 4)·CHOPENLF(t, 2, 5) ]
      [                                      ]
      [       X14(t) := ¿·CHGRT(t, 5)        ]

      [                     Y11(t) := ¿·CHLESS(t, -1)                      ]
      [                                                                    ]
      [                              2                                     ]
      [              Y12(t) := (1 + t )·CHCLOSED(t, -1, 2)                 ]
#4:   [                                                                    ]
      [                       2                                            ]
      [ Y13(t) := (- (-t + 4)  + 2·(-t + 4) + 5)·CHOPENLF(t, 2, 5)         ]
      [                                                                    ]
      [                      Y14(t) := ¿·CHGRT(t, 5)                       ]
```

```
#5:  [ X1(t) := X11(t) + X12(t) + X13(t) + X14(t) ]
     [ Y1(t) := Y11(t) + Y12(t) + Y13(t) + Y14(t) ]

#6:  [S(t) := 6·t - 1, u1 := 1 + x^2, v1 := - x^2 + 2·x + 5, p := 0.5, q := 3]

#7:  APPEND([ u1 ; v1 ], APPEND([[X1(S(t)), Y1(S(t))]], VECTOR([a·(X1(S(t)) - p) +
       p, a·(Y1(S(t)) - q) + q], a, 0.1, 0.9, 0.1)))
```



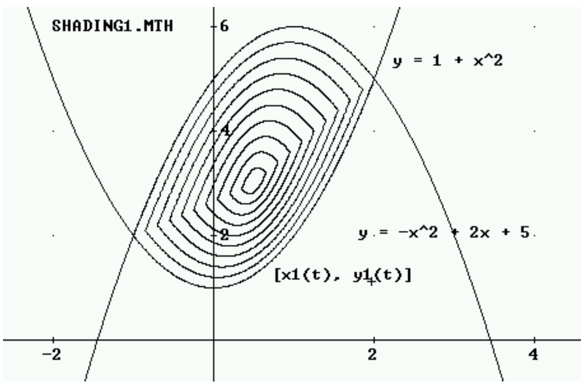
As you can see the plot is not really satisfying because of many gaps – which I cannot explain..

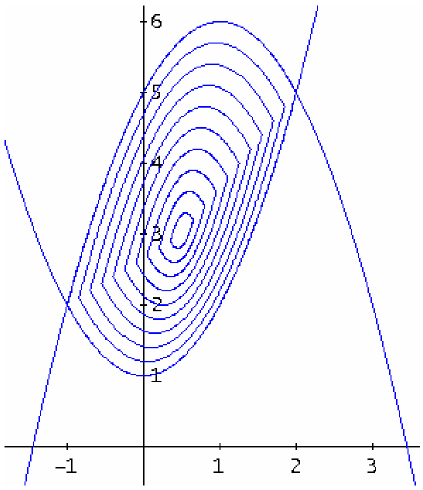
So I take the “alternative” IF-method – which caused problems in various earlier versions.

Unfortunately I cannot present an explanation for details like S(t) = 6t –1, (-t + 4) replaces x, ...

```
#1:  [u1 := 1 + x^2, v1 := - x^2 + 2·x + 5]

     X2(t) :=
       If -1 ≤ t ≤ 2
          t
          If 2 < t ≤ 5
             -t + 4
#2:

     Y2(t) :=
       If -1 ≤ t ≤ 2
          1 + t^2
          If 2 < t ≤ 5
             - (-t + 4)^2 + 2·(-t + 4) + 5
#3:

#4:  [S(t) := 6·t - 1, u := X2(S(t)), v := Y2(S(t)), p := 0.5, q := 3]

#5:  APPEND([ u1 ; v1 ], APPEND([[u, v]], VECTOR([a·(u - p) + p, a·(v - q) + q], a, 0.1, 0.9, 0.1)))
```

Original plot from 1998 (below) and
DERIVE 6.10 plot (right). $0 \le t \le 1$ for all plots





Comments from 1998:

I tried to reproduce Mr Speck´s graphs and found that the various DfW versions show different behaviours:
In version 4.0 the "Char"-functions don´t work (Sorry, the highlighted expression cannot be plotted), and the "If" shows only the bottom half of the shading.
Version 4.05 reacts on "Char" with "No finite real values!", "If" works.
Version 4.10 similar to 4.05. Very strange behaviour with "Char" - after a few attempts mixes up the algebra window with the plot window, hangs up, ...

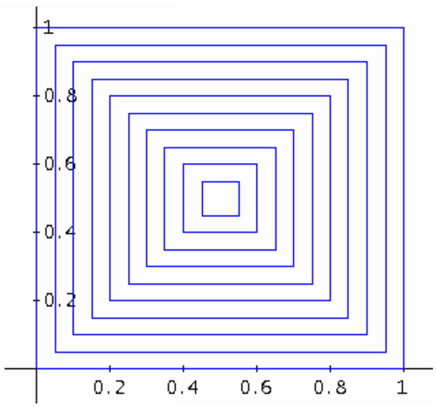
I had no problems with DOS versions 3.10 to 4.x.   Josef

Al Rich´s (SWHH) answer: *Apparently the problem with the "Char"-functions is due to the huge expression that is generated which in turn causes a memory overflow. We will investigate. ......*
Since the IF( ) statement version of the shading program is faster and shorter, why publish the "Char" function version?

Good question. Because it is an interesting example of "programming" with DERIVE.

Concentric shading of the unit square given parametrically (IF-method):

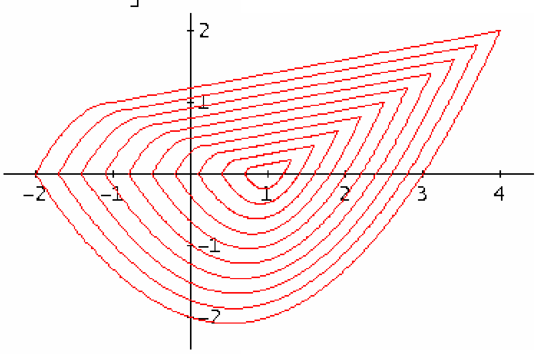
```
[G(t) := F(t - 0.25), F(t) := 0.5 + 0.5·COS(4·π·t)]

⎡X2(t) :=                       , Y2(t) :=                   ⎤
⎢  If 0 ≤ t ≤ 0.25                If 0 ≤ t ≤ 0.25            ⎥
⎢     0                              G(t)                    ⎥
⎢     If 0.25 < t ≤ 0.5              If 0.25 < t ≤ 0.5       ⎥
⎢        F(t)                           1                    ⎥
⎢        If 0.5 < t ≤ 0.75              If 0.5 < t ≤ 0.75    ⎥
⎢           1                              F(t)              ⎥
⎢           If 0.75 < t ≤ 1                If 0.75 < t ≤ 1   ⎥
⎣              G(t)                           0              ⎦

[S(t) := t, p := 0.5, q := 0.5]

[u := X2(S(t)), v := Y2(S(t))]

APPEND([[u, v]], VECTOR([a·(u - p) + p, a·(v - q) + q], a, 0.1, 0.9, 0.1))
```

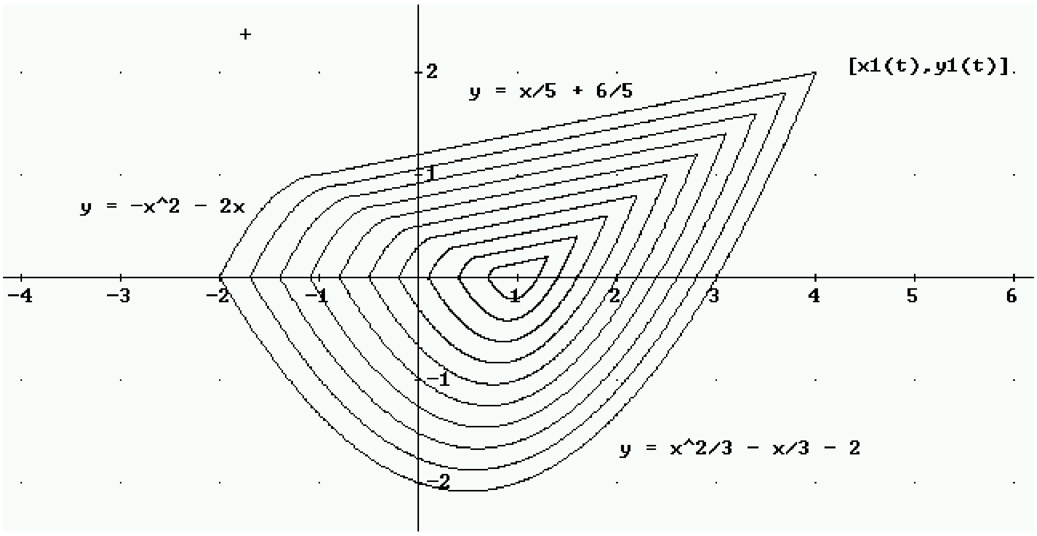


Concentric shading of region made up of portions of bounding (curves F1, F2 and F3):

$$\left[F1(x) := \frac{1}{3}\cdot x^2 - \frac{1}{3}\cdot x - 2,\ F2(x) := \frac{x}{5} + \frac{6}{5},\ F3(x) := -x^2 - 2\cdot x\right]$$

```
[Y21(t) := F1(t), Y22(t) := F2(8 - t), Y23(t) := F3(8 - t)]

⎡X2(t) :=             , Y2(t) :=            ⎤
⎢  If -2 ≤ t ≤ 4          If -2 ≤ t ≤ 4     ⎥
⎢     t                      Y21(t)         ⎥
⎢     If 4 < t ≤ 10          If 4 < t ≤ 9   ⎥
⎢        8 - t                  Y22(t)      ⎥
⎢                               If 9 < t ≤ 10⎥
⎣                                  Y23(t)   ⎦

[S(t) := 12·t - 2, p := 1, q := 0]

[u := X2(S(t)), v := Y2(S(t))]

APPEND([[u, v]], VECTOR([a·(u - p) + p, a·(v - q) + q], a, 0.1, 0.9, 0.1))
```



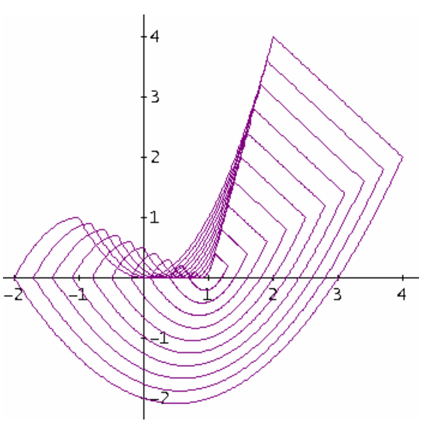


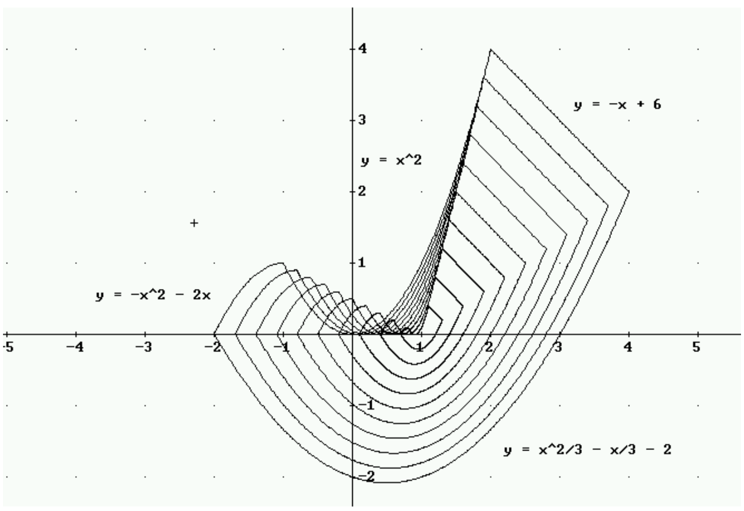
Concentric shading of a region made up of portions of bounding (curves F1, F2, F3 and F4):

$$\left[F1(x) := \frac{1}{3}\cdot x^2 - \frac{1}{3}\cdot x - 2,\ F2(x) := 6 - x,\ F3(x) := x^2,\ F4(x) := -x^2 - 2\cdot x\right]$$

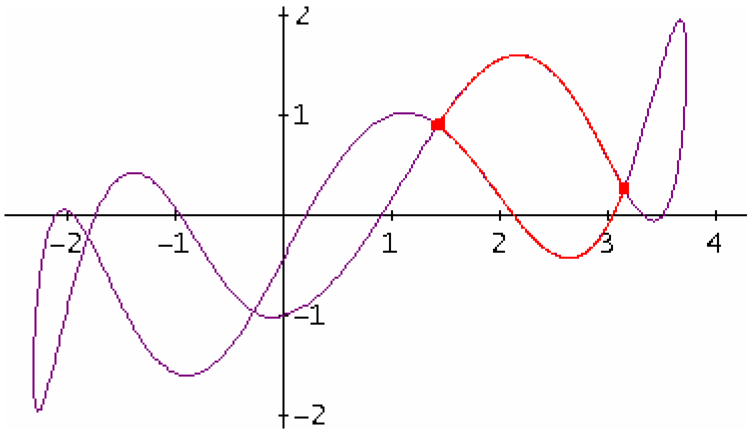
```
[Y21(t) := F1(t), Y22(t) := F2(8 - t), Y23(t) := F3(8 - t), Y24(t) := F4(8 - t)]

[X2(t) :=                , Y2(t) :=
   If -2 ≤ t ≤ 4            If -2 ≤ t ≤ 4
      t                        Y21(t)
      If 4 < t ≤ 10            If 4 < t ≤ 6
         8 - t                    Y22(t)
                                  If 6 < t ≤ 9
                                     Y23(t)
                                     If 9 < t ≤ 10
                                        Y24(t)                  ]

[S(t) := 12·t - 2, p := 1, q := 0]

[u := X2(S(t)), v := Y2(S(t))]

APPEND([[u, v]], VECTOR([a·(u - p) + p, a·(v - q) + q], a, 0.1, 0.9, 0.1))
```

DERIVE 6.10 (below) vs DfW4 (right):

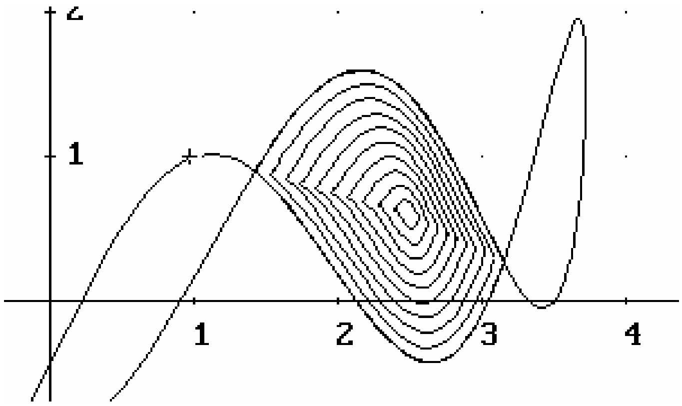




Shading of one particular loop of [3 sin(t)+cos(0.5t), cos(5t)+sin(t)] with $-\pi \le t \le +\pi$.
We fill the loop given by $0.14 \le t \le 0.84$ and $2.07 \le t \le 2.72$.



Unfortunately I cannot reproduce G P Speck's shaded figure because – shame on me – I cannot define the right IF-constructs. You can see the figure from 1998. I hope that George will explain the procedure more detailed and then we can try again. It might be a challenge for you, too:



Memories of Gettyburg

Carl Leinbach produced a CD-ROM containing the lectures and workshops of the 3rd International *DERIVE* and *TI-92* Conference. You can purchase this inexpensive CD-ROM at Jerry Glynn´s company **MathWare.**

MathWare, P.O.Box 3025 Urbana, IL 61803, email: info@mathware.com

Three keynotes can be downloaded from Carl´s home page!

As time went by this home page is not valid in 2013.

One keynote is among the files collected in mth31.zip.

You can download the Proceedings from:

http://rfdz.ph-noe.ac.at/acdca/konferenzen/gettysburg-1998.html

Cyclone 98

Now you are able to investigate graphs of surfaces which have simple equations which would be normally hard (or impossible) to graph like abs(x ∗ y ∗ z) = 1 or sin(2x) + sin(2y) = sin(2z). You can graph 3D inequalities and you also can generate lists of surfaces. The objects can be animated. *Cyclone 98* runs on Win95 or Windows NT.

I googled for Cyclone 98 – and what did I find? There were al lot of advertisements for two products:

A paintball equipment -

and a model of a famous race car.

But there is no soft ware!! Go for **http://www.dpgraph.com/**

Philip Yorke asked me to print the following paragraph:

Chartwell-Yorke have added full details of a wide range of English language books using *DERIVE* to their website. You will find full descriptions, cover photos and contents lists at http://ChartwellYorke.com/Booksl.html. There are a great many new books which you may not be familiar with on the site! They would like to add your independent comments about the books to your site, so you can help each other to choose which suits your needs. If you are family with any of the books covered, please email your comments on their usefulness to Philip.Yorke@ChartwellYorke.com and he will add your feedback to the site.

From "nested IF expressions" to a "Macro for Solving Equations"

From: HeinzRainer.Geyer@t-online.de 31-01-1998

Subject: nested IF expressions

Hi all, what is wrong with the following definition:

```
TRANSF(equation,operator,value):=IF(operator = "+",equation+value,
IF(operator = "-", equation-value,"?"))
```

Test `TRANSF(10x + 6=0,"-",6)` returns the unsimplified expression,

with `TRANSF(10x + 6=0,"+",-6)` you get what one is expecting: `10x = 6`

I want to define a function to automate equivalence transformations of equations, of course with more than "+","-".

I use DfW 4.06 German and DfW 4.0 English.

Thanks for hints. Helpful would also be an other form than the nested if structure.

Rainer Geyer

From: math@rhombus.be 31-01-1998

Dear Heinz,
I can´t solve your nested IF´s problem, but:
please see the attached LIN_EQ.MTH file which automates and illustrates the consecutive equivalence transformations on linear equations.
Best regards, Jan Vermeylen

LIN_EQ.MTH - Jan Vermeylen, version 4:

#1: EQUIV0(eq) := EXPAND(eq)

#2: EQUIV1(eq) := EXPAND(eq) - (TERMS(LHS(EXPAND(eq))))$_2$ - (TERMS(RHS(EXPAND(eq))))$_1$

#3: $\text{EQUIV2(eq)} := \dfrac{\text{EXPAND(eq)}}{\text{(FACTORS(LHS(EXPAND(eq))))}_2}$

#4: example

#5: $2\cdot(x - 1) + \dfrac{x}{3} = 4\cdot x - 9 + \dfrac{x - 2}{3}$

#6: $\text{EQUIV0}\left(2\cdot(x - 1) + \dfrac{x}{3} = 4\cdot x - 9 + \dfrac{x - 2}{3}\right) = \left(\dfrac{7\cdot x}{3} - 2 = \dfrac{13\cdot x}{3} - \dfrac{29}{3}\right)$

#7: $\text{EQUIV1}\left(\dfrac{7\cdot x}{3} - 2 = \dfrac{13\cdot x}{3} - \dfrac{29}{3}\right) = \left(- 2\cdot x = - \dfrac{23}{3}\right)$

#8: $\text{EQUIV2}\left(- 2\cdot x = - \dfrac{23}{3}\right) = \left(x = \dfrac{23}{6}\right)$

DERIVE 6.10's answers:

#9: $\text{EQUIV0}\left(2\cdot(x-1)+\frac{x}{3}=4\cdot x-9+\frac{x-2}{3}\right)$

#10: $x=\frac{13\cdot x}{7}-\frac{23}{7}$

Expand does much more than only expanding both sides!!

#11: $\text{EQUIV1}\left(x=\frac{13\cdot x}{7}-\frac{23}{7}\right)=\left(x=\frac{23}{6}\right)$

From: Spwelke@aol.com 01-02-1998

Hallo Mr. Geyer, hallo all DERIVE users,

Your problem is that DERIVE does not evaluate string expressions like "Hallo" = "hallo" as false, but DERIVE evaluates "Hallo" = "Hallo" as true:

```
#1:  IF("Hallo" = "Hallo", 1, 2) = 1
#2:  IF("Hallo" = "hallo", 1, 2) = IF("Hallo" = "hallo", 1, 2)
```

doesn´t evaluate as 2 as one would expect. A simple way around your problem is to assign numerical values to the operations in mind, for example:

DERIVE 6 works as expected: `#2:  IF("Hallo" = "hallo", 1, 2) = 2`

But we meet again the same problem as before, DERIVE does too much in this case.

DERIVE immediately solves the equation for the variable and does not consider the given task.

```
transf(eq, op, val) :=
  If op = "+"
     eq + val
     If op = "-"
        eq - val
        eq
```

$\text{transf}(4\cdot x+3=8,\ -,\ 3)=\left(x=\frac{5}{4}\right)$

$\text{transf}(4\cdot x+3=8,\ +,\ 300)=\left(x=\frac{5}{4}\right)$

There is a much simpler way to deal with your particular problem. If I understood your intentions well, the problem is to handle equations as we all have learned at school. Please see an example:

#6: $2^{2-x}+3=9$

#7: $(2^{2-x}+3=9)-3=(2^{2-x}=6)$

#8: $\text{LOG}(2^{2-x}=6)=((2-x)\cdot\text{LN}(2)=\text{LN}(6))$

#9: $\frac{(2-x)\cdot\text{LN}(2)=\text{LN}(6)}{\text{LN}(2)}=\left(2-x=\frac{\text{LN}(6)}{\text{LN}(2)}\right)$

#10: $\left(2-x=\frac{\text{LN}(6)}{\text{LN}(2)}\right)-2=\left(-x=\frac{\text{LN}(6)}{\text{LN}(2)}-2\right)$

#11: $\left(-x=\frac{\text{LN}(6)}{\text{LN}(2)}-2\right)\cdot(-1)=\left(x=2-\frac{\text{LN}(6)}{\text{LN}(2)}\right)$

We can see that arithmetic operations and functions apply to both sides of an equation separately.

From: GNSStepan@aol.com 02-02-1998

Hallo Mr Welke, hallo all DERIVE users,

your simple way handling equations was very interesting to me. At school it is important that simplification occurs step by step. Executing your example with DfW 4.05 equation #10 simplifies to

$$-x = \frac{LN(3)}{LN(2)} - 1 \quad \text{instead of} \quad -x = \frac{LN(6)}{LN(2)} - 2$$

What can I do to avoid this additional simplification step? Thanks for a tip. Georg Stepan

(A short additional comment: You might give the problem a new quality: ask your students for DERIVE´s hidden calculation, Josef)

From: Spwelke@aol.com 02-02-1998

Dear Mr. Geyer, dear Mr. Stepan, dear DERIVE users,

(1) At first I respond to Mr. Geyer´s request concerning a CASE-Function:

A sequence of nested IF-expressions calls for a CASE-function, which allows the user to select between more than two alternatives. A simple way to achieve this functionality is the following CASE-function:

#1: $\text{CASE}(a\_, v\_) := (\text{SELECT}(a\_ = k\_{\_1}, k\_, v\_))_{1,2}$

#2: $\text{test} := \begin{bmatrix} 3 & 4 \\ 4 & 5 \\ 5 & t \\ 7 & z \end{bmatrix}$

#3: $\text{CASE}(5, \text{test}) = t$

#4: [plus := 1, minus := 2, times := 3, divided_by := 4]

#5: $\text{TRANSE}(eq, op, val) := \text{CASE}\left(op, \begin{bmatrix} 1 & eq + val \\ 2 & eq - val \\ 3 & eq \cdot val \\ 4 & \dfrac{eq}{val} \end{bmatrix}\right)$

DfW 4:

#7: $\text{TRANSE}\left(\frac{2\cdot x}{3} - 7 = \frac{2}{11}, \text{plus}, 7\right) = \left(\frac{2\cdot x}{3} = \frac{79}{11}\right)$

#8: $\text{TRANSE}\left(\frac{2\cdot x}{3} = \frac{79}{11}, \text{times}, 3\right) = \left(2\cdot x = \frac{237}{11}\right)$

#9: $\text{TRANSE}\left(2\cdot x = \frac{237}{11}, \text{divided\_by}, 2\right) = \left(x = \frac{237}{22}\right)$

but DERIVE 6 solves - unasked - the equation:

#10: $\text{TRANSE}\left(\frac{2\cdot x}{3} - 7 = \frac{2}{11}, \text{plus}, 7\right) = \left(x = \frac{237}{22}\right)$

Another use of the CASE-function is to assign the value `true` or `false` to the first argument of the CASE-function. Then the first element of each line v_ must be a logical expression, which can be evaluated to `true` or `false`. Please see the next example:

#11: $F(x\_) := \text{CASE}\left(\text{true}, \begin{bmatrix} x\_ < 1 & \text{too small} \\ x\_ > 3 & \text{too big} \\ x\_ \ge 1 \wedge x\_ \le 3 & \text{ok} \end{bmatrix}\right)$

#12: $[F(-3), F(e), F(2\cdot\pi)] = [\text{too small}, \text{ok}, \text{too big}]$

Works also in the recent versions, Josef

(2) Now I respond to Mr. Stepan´s question
There is no way to manipulate the evaluation of expressions in *DERIVE*. In *MATHEMATICA*, for example, there is an extensive set of commands to control, prevent and to alter the way *MATHEMATICA* evaluates expressions. I cannot see any simple trick to prevent the evaluation of LOG(6)/LOG(2) as LOG(3)/LOG(2) – 1. Sorry! Maybe someone else in the *DERIVE* community has a trick?

So long for this time, Stefan Welke

The next part of this long way to the final Macro is of historic interest. I reprint the evolution process of this equation solving macro designed and improved by Johann Wiesenbauer.
As the DERIVE versions changed his MACRO has to be changed, too. Commands which worked in earlier versions did not work any longer... You must consider that in 1998 we could not write complex programs as we can do now. There were “only” mor or less complicated functions which called other functions which called ...

The final version which works even with DERIVE 6.10 was published in DNL#53 (March 2004). I don’t intend to let you waiting until DNL#53 will be revised ☺. I include the latest version in this DNL#31 rev. But you may try providing a similar MACRO for TI-Nspire, Josef

From: J.Wiesenbauer@tuwien.ac.at 03-02-1998
Subject: Once again: Nested If expressions

Hi all,
Yes it is true, the idea of solving an equation step by step is great, but I have not yet seen a really satisfying solution. What I think is the best approach makes use of the following tiny auxiliary function:

```
#1:  DO(u) := (eq := u)
```

In the following examples you can see it at work:

Example 1 (Jan Vermeylen´s equation)

#2: $eq := 2\cdot(x - 1) + \frac{x}{3} = 4\cdot x - 9 + \frac{x - 2}{3}$

#3: $DO(EXPAND(eq)) = \left[\frac{7\cdot x}{3} - 2 = \frac{13\cdot x}{3} - \frac{29}{3}\right]$

#4: $DO\left[eq - \frac{13\cdot x}{3} + 2\right] = \left[- 2\cdot x = - \frac{23}{3}\right]$

#5: $DO\left[\frac{eq}{-2}\right] = \left[x = \frac{23}{6}\right]$

#6: "or:"

#7: $eq := 2\cdot(x - 1) + \frac{x}{3} = 4\cdot x - 9 + \frac{x - 2}{3}$

#8: $DO(EXPAND(eq)) = \left[\frac{7\cdot x}{3} - 2 = \frac{13\cdot x}{3} - \frac{29}{3}\right]$

#9: $DO(eq\cdot 3) = (7\cdot x - 6 = 13\cdot x - 29)$

#10: $DO(eq - 13\cdot x + 6) = (- 6\cdot x = -23)$

#11: $DO\left[\frac{eq}{-6}\right] = \left[x = \frac{23}{6}\right]$

Assets: (1) My method is not restricted to linear equations (cf. Ex. 2)
(2) There is no need to input equarions (except for #2)!

Remark: In contrast to Jan´s method the pupil has to find the appropriate transformations by himself. In my eyes this is actually an advantage (after all, we know that DERIVE can do it!), but you may view this as a draw-back

Example 2 (Stefan Welke´s equation)

#12: $eq := 2^{2 - x} + 3 = 9$

#13: $DO(eq - 3) = (2^{2 - x} = 6)$

#14: $DO(LOG(eq, 2)) = \left[2 - x = \frac{LN(3)}{LN(2)} + 1\right]$

#15: $DO(eq - 2) = \left[-x = \frac{LN(3)}{LN(2)} - 1\right]$

#16: $DO\left[\frac{eq}{-1}\right] = \left[x = 1 - \frac{LN(3)}{LN(2)}\right]$

Remark: At first glance this is very close to Stefan´s solution, but unlike him I don´t have to care about equation numbers and, above all, what looks nice in the input line at first turns immediately into a very ugly term after pressing ENTER! What you see (in his e-mail) is not what you get (on the screen)!

(In his e-mail Stefan used the expression numbers as an abbreviation. Johann closed his message with another function to give a concluding report containing all steps performed. I´ll skip that part because he overtrumped himself in the message which arrived 10 days later. Josef)

Don’t try now, because the “tiny auxiliary” function DO(u) does not work!!

From: J.Wiesenbauer@tuwien.ac.at 13-02-1998
Subject: A macro for solving equations

In the attached file you can see for the first time what one might a call a *DERIVE*-macro! I have used it to record the step-by-step solution of an equation, but advanced programmers should have no difficulty in exploiting the underlying ideas for other purposes! (Good luck!)
Cheers, Johann

```
#1:  "A macro for solving equations ((c) Johann Wiesenbauer,Vienna)"
#2:  "The following six functions must be loaded:"
#3:  NEWO(u):=(o:=u)
#4:  NEWEQ(u):=(eq:=u)
#5:  DO(u):=NEWEQ(LIM(u,@,eq))+0*SUM(NEWO(APPEND(o,[u])))
#6:  UNDO(u):=NEWEQ(LIM(ITERATE(u,@,@,-1),@,eq))+0*SUM(NEWO
     (DELETE_ELEMENT(o,DIMENSION(o))))
#7:  RECORD(u):=(eq:=(e:=u))+SUM(o:=[])
#8:  play:=DELETE_ELEMENT(ITERATES([LIM(a_ SUB 1,@,e_),IF
     (DIMENSION(o_)=0,o_,[o_ SUB 1]),DELETE_ELEMENT(o_)],[e_,a_,o_],
     [e,[o SUB 1],DELETE_ELEMENT(o)],DIMENSION(o))`,3)`
#9:  "Study the following examples carefully to see how it works!"
```

#10: RECORD($3^{2·x + 1}$ - 7 = 20)   User

#11: $3^{2·x + 1} - 7 = 20$ Simp(#10)

#12: DO(@ + 7)   User

#13: $3^{2·x + 1} = 27$ Simp(#12)

#14: DO(LOG(@, 3))

#15: $2·x + 1 = 3$

#16: DO$\left[\frac{@}{2}\right]$

#17: $\frac{2·x + 1}{2} = \frac{3}{2}$

#18: "Oops!"

#19: UNDO$\left[\frac{@}{2}\right]$

#20: $2·x + 1 = 3$

#21: DO(@ - 1)

#22: $2·x = 2$

#23: DO$\left[\frac{@}{2}\right]$

#24: $x = 1$

#25: play

#26: $\begin{bmatrix} 3^{2·x + 1} - 7 = 20 & [@ + 7] \\ 3^{2·x + 1} = 27 & \left[\frac{LN(@)}{LN(3)}\right] \\ 2·x + 1 = 3 & [@ - 1] \\ 2·x = 2 & \left[\frac{@}{2}\right] \\ x = 1 & [\,] \end{bmatrix}$

I skip the 2nd example, because this version does not work in recent versions. Josef

From: J.Wiesenbauer@tuwien.ac.at 14-02-1998

Subject: Troubleshooting as regards my macro

Since I got e-mails of users who have troubles with my macro two comments might come in handy:

- All evocations of my functions must be simplified subsequently, in particular `RECORD(...)`. Sorry that I was mistaken in taking that for granted. (After all, you can see those simplifications in the annotations of my example!!!)
- If you still have troubles, this might be due to an older version of DfW. I can only guarantee that it works in DfW 4.08!) Just in case you have an older English version DfW 4.0x, there is an easy remedy: You should use a free (!) update on the *DERIVE*-homepage!!!
- Cheers, Johann

(I tried to use Johann´s macro with DOS versions 3.x. It works if you replace all the @s in `DO`, `UNDO` *and* `PLAY` *by any unused variable, e.g. take g_ . I wish much luck, much fun and much successworking with this macro. Josef)*

This is Johann's the final MACRO - version (together with small additions. Johann wrote: *"On the other hand, I have added a new command sqr(@) to compute square roots. In a similar way, other useful commands (e.g. for expanding and factoring polynomials) could be added, if there is a need for it."*

Examples: do(@,"e"), do(@,"f") and do(@,"r") for expanding, factoring and drawing the root.

```
     RECORD(u) :=
       Prog
         o := []
         @1 := LHS(u)
#1:      @2 := RHS(u)
         l := [@1]
         r := [@2]
         u

     sqr(u, u_) :=
       Prog
         If NUMBER?(u)
#2:         RETURN ± √u
         u_ := POLY_GCD(u, ∂(u, x))
         If NUMBER?(u/u_^2)
            √(u/u_^2)·u_

     UNDO(u) :=
       Prog
         o := REST(o)
         l := REST(l)
#4:      r := REST(r)
         @1 := FIRST(l)
         @2 := FIRST(r)
         ["", "↓", ""; @1, "=", @2]
```

```
     do(u, t := 0, t_) :=
       Prog
         t_ := (NAME_TO_CODES(t))↓1
         If t_ = 101
            Prog
              @1 := EXPAND(@1, "Radical")
              @2 := EXPAND(@2, "Radical")
              o := ADJOIN(["expand(@)"], o)
         If t_ = 102
            Prog
              @1 := FACTOR(@1, "Radical")
              @2 := FACTOR(@2, "Radical")
              o := ADJOIN(["factor(@)"], o)
#3:      If t_ = 114
            Prog
              @1 := sqr(@1)
              @2 := sqr(@2)
              o := ADJOIN(["±√@"], o)
         If t_ = 48
            Prog
              @1 := SUBST(u, @, @1)
              @2 := SUBST(u, @, @2)
              o := ADJOIN([u], o)
         l := ADJOIN(@1, l)
         r := ADJOIN(@2, r)
         ["", "↓", ""; @1, "=", @2]
```

```
#5:   play := [REVERSE(l), VECTOR(=, k_, DIM(l)), REVERSE(r), APPEND(REVERSE(o), [])]'
```

A few examples:

#6: $\text{RECORD}('(3^{2\cdot x+1}-7=20))$

#7: $3^{2\cdot x+1}-7=20$

#8: $\text{do}(@+7)=\left[\begin{matrix}&\downarrow&\\3^{2\cdot x+1}&=&27\end{matrix}\right]$

#9: $\text{do}(\text{LOG}(@,3))=\left[\begin{matrix}&\downarrow&\\2\cdot x+1&=&3\end{matrix}\right]$

#10: $\text{do}\left(\frac{@}{2}\right)=\left[\begin{matrix}&\downarrow&\\\frac{2\cdot x+1}{2}&=&\frac{3}{2}\end{matrix}\right]$

#11: Oops, sorry!

#12: $\text{UNDO}()=\left[\begin{matrix}&\downarrow&\\2\cdot x+1&=&3\end{matrix}\right]$

#13: $\text{do}(@-1)=\left[\begin{matrix}&\downarrow&\\2\cdot x&=&2\end{matrix}\right]$

#14: $\text{do}\left(\frac{@}{2}\right)=\left[\begin{matrix}&\downarrow&\\x&=&1\end{matrix}\right]$

#15: $\text{play}=\left[\begin{matrix}3^{2\cdot x+1}-7&=&20&[@+7]\\3^{2\cdot x+1}&=&27&\left[\frac{\text{LN}(@)}{\text{LN}(3)}\right]\\2\cdot x+1&=&3&[@-1]\\2\cdot x&=&2&\left[\frac{@}{2}\right]\\x&=&1&\end{matrix}\right]$

#25: $\text{RECORD}('(x^2-4\cdot x+7=10))$

#26: $x^2-4\cdot x+7=10$

#27: $\text{do}(@-3)=\left[\begin{matrix}&\downarrow&\\x^2-4\cdot x+4&=&7\end{matrix}\right]$

#28: $\text{do}(\sqrt{@})=\left[\begin{matrix}&\downarrow&\\|x-2|&=&\sqrt{7}\end{matrix}\right]$

This is the standard √, let's try the parameter **"r"**:

#29: $\text{do}(@,r)=\left[\begin{matrix}&\downarrow&\\x-2&=&\pm\sqrt{7}\end{matrix}\right]$

#30: $\text{do}(@+2)=\left[\begin{matrix}&\downarrow&\\x&=&\pm\sqrt{7}+2\end{matrix}\right]$

#31: $\text{play}=\left[\begin{matrix}x^2-4\cdot x+7&=&10&[@-3]\\x^2-4\cdot x+4&=&7&[\pm\sqrt{@}]\\x-2&=&\pm\sqrt{7}&[@+2]\\x&=&\pm\sqrt{7}+2&\end{matrix}\right]$

A more complex equation:

#16: $\text{RECORD}\left('\left(\frac{b+x}{b-x}-\frac{4\cdot b\cdot(3\cdot b-x)}{b^2-2\cdot b\cdot x+x^2}=\frac{b-x}{b+x}\right)\right)$

#17: $\frac{b+x}{b-x}-\frac{4\cdot b\cdot(3\cdot b-x)}{b^2-2\cdot b\cdot x+x^2}=\frac{b-x}{b+x}$

#18: $\text{do}(@\cdot(b-x)^2\cdot(b+x))=\left[\begin{matrix}&\downarrow&\\-(x+b)\cdot(x^2-4\cdot b\cdot x+11\cdot b^2)&=&(b-x)^3\end{matrix}\right]$

#19: $\text{do}(@,e)=\left[\begin{matrix}&\downarrow&\\-x^3+3\cdot b\cdot x^2-7\cdot b^2\cdot x-11\cdot b^3&=&-x^3+3\cdot b\cdot x^2-3\cdot b^2\cdot x+b^3\end{matrix}\right]$

#20: $\text{do}(@+x^3-3\cdot b\cdot x^2+7\cdot b^2\cdot x)=\left[\begin{matrix}&\downarrow&\\-11\cdot b^3&=&4\cdot b^2\cdot x+b^3\end{matrix}\right]$

#21: $do(@ - b^3) = \left[\begin{array}{c}\downarrow\\ -12\cdot b^3 = 4\cdot b^2\cdot x\end{array}\right]$

#22: $do\left(\dfrac{@}{4\cdot b^2}\right) = \left[\begin{array}{c}\downarrow\\ -3\cdot b = x\end{array}\right]$

#23: play

#24: $$\left[\begin{array}{ccc} -\dfrac{x^2 - 4\cdot b\cdot x + 11\cdot b^2}{(x-b)^2} = \dfrac{b-x}{x+b} & \left[@\cdot(x+b)\cdot(x-b)^2\right] \\ -(x+b)\cdot(x^2 - 4\cdot b\cdot x + 11\cdot b^2) = (b-x)^3 & [expand(@)] \\ -x^3 + 3\cdot b\cdot x^2 - 7\cdot b^2\cdot x - 11\cdot b^3 = -x^3 + 3\cdot b\cdot x^2 - 3\cdot b^2\cdot x + b^3 & \left[x^3 - 3\cdot b\cdot x^2 + 7\cdot b^2\cdot x + @\right] \\ -11\cdot b^3 = 4\cdot b^2\cdot x + b^3 & \left[@ - b^3\right] \\ -12\cdot b^3 = 4\cdot b^2\cdot x & \left[\dfrac{@}{4\cdot b^2}\right] \\ -3\cdot b = x & \end{array}\right]$$

Jan Vermeylen's equation:

#32: $RECORD\left('\left(2\cdot(x-1) + \dfrac{x}{3} = 4\cdot x - 9 + \dfrac{x-2}{3}\right)\right)$

#33: $2\cdot(x-1) + \dfrac{x}{3} = 4\cdot x - 9 + \dfrac{x-2}{3}$

#34: $do(@,\ e) = \left[\begin{array}{c}\downarrow\\ \dfrac{7\cdot x - 6}{3} = \dfrac{13\cdot x - 29}{3}\end{array}\right]$

#35: $do(@\cdot 3) = \left[\begin{array}{c}\downarrow\\ 7\cdot x - 6 = 13\cdot x - 29\end{array}\right]$

#36: $do(@ - 7\cdot x + 29) = \left[\begin{array}{c}\downarrow\\ 23 = 6\cdot x\end{array}\right]$

#37: $do\left(\dfrac{@}{6}\right) = \left[\begin{array}{c}\downarrow\\ \dfrac{23}{6} = x\end{array}\right]$

#38: $$play = \left[\begin{array}{cc} \dfrac{7\cdot x - 6}{3} = \dfrac{13\cdot x - 29}{3} & [expand(@)] \\ \dfrac{7\cdot x - 6}{3} = \dfrac{13\cdot x - 29}{3} & [3\cdot @] \\ 7\cdot x - 6 = 13\cdot x - 29 & [-7\cdot x + @ + 29] \\ 23 = 6\cdot x & \left[\dfrac{@}{6}\right] \\ \dfrac{23}{6} = x & \end{array}\right]$$

This is the last lecture of the 1997 International DERIVE and TI-92 symposium held in Särö, Sweden, to be published in the DNL.

An Algorithm for Extracting Logic Propositions from Numerical Data and its Implementation with DERIVE™.

Terence Etchells
School of Computing and Mathematical Sciences
Liverpool John Moores University, UK.

Abstract

This article is based on the work of Hiroshi Tsukimoto of the Systems and Software Engineering Laboratory, Toshiba Corporation [1]. The aim of this work is to review the work of Tsukimoto and implement his algorithms in **Derive for Windows**.

If a regression analysis has been performed on some data set, then the resulting expression is an algebraic one involving variables *x*, *y*, *z*, ... etc. depending on the number of variables. For example, imagine that we are given a set of data of three varaibles *x*, *y* and *z*. A linear regression plane of the form $z = ax + by + c$ can be readily obtained. What does this regression line tell us about the underlying structure of the data? What we require is a Logic proposition that describes the hidden structure that generates the data. For example such a proposition could be $z = (x \wedge \neg y)$, which would be interpreted as the property ***z* is true if** the property ***x* is true *and*** the property ***y* is *not* true** (i.e. false). The essence of this paper is to describe and implement an algorithm for extracting such propositions from given regression hyper planes or data.

Backgound

Logic Vectors

Propositions in Boolean logic can be represented as vectors, theses vectors are called **Logic Vectors**. We will spend some time looking at the construction of Logic vectors and their building blocks **atoms**. As it was thought for some time that atoms were the building blocks of matter, atoms in the Boolean sense are the building blocks of Boolean functions.

Let us take the most simple Boolean function *x*. The function *x* can take only two values 0 or 1 (i.e. elements from the set {0,1}) as it is Boolean, and also consider its negation $\neg x$ which can also only take values $\{0,1\}$. The truth table of *x* and $\neg x$ is

x	$\neg x$
1	0
0	1

Figure 1
Truth table for $\neg x$

which tells us some thing very important, that all Boolean functions of a single variable will either be a contradiction (e.g. $x \wedge \neg x$), a tautology (e.g. $x \vee \neg x$) , x or $\neg x$. This is because a truth table of a Boolean function of one variable can only contain the values [1,1], [1,0],[0,1] or [0,0]. Here we have written the vertical column of the truth table as a horizontal vector, the idea of the truth values of a function being represented as a vector is paramount to the ensuing work. To illustrate this further we will study the truth tables of some Boolean functions of a single variable, namely $\neg((x \wedge \neg x) \vee \neg(x \wedge x))$; $\neg x \vee \neg(x \wedge x)$; $x \wedge x \vee x$.

As DERIVE is used extensively in this work we will use it from the outset to produce truth tables.

```
TRUTH_TABLE(x, ¬ ((x ∧ ¬ x) ∨ ¬ (x ∧ x)), ¬ x ∨ ¬ (x ∧ x), (x ∧ x) ∨ x)

┌  x     ¬ ((x ∧ ¬ x) ∨ ¬ (x ∧ x))   ¬ x ∨ ¬ (x ∧ x)   (x ∧ x) ∨ x ┐
│ true            true                    false            true     │
└ false           false                   true             false    ┘
```

As it is desirable to have values of 0 or 1 instead of the variables `true` and `false`, the functions `TEST(arg)` and `TRUTH(tt)` are introduced.

```
TEST(arg) :=
  If arg = true
     1
     0
     arg

TRUTH(tt) := APPEND(┌tt ┐, VECTOR(VECTOR(TEST(tt   ), m, 1, DIMENSION(tt')), r, 2, DIMENSION(tt)))
                    └ 1 ┘                      r,m

      ┌  x     ¬ ((x ∧ ¬ x) ∨ ¬ (x ∧ x))   ¬ x ∨ ¬ (x ∧ x)   (x ∧ x) ∨ x ┐
TRUTH │ true            true                    false            true     │
      └ false           false                   true             false    ┘

┌ x  x  ¬ x  x ┐
│ 1  1   0   1 │
└ 0  0   1   0 ┘
```

As we can see the application of the `TRUTH()` function simplifies the Boolean expressions on the top row of the truth table to their minimised (reduced) form. DERIVE's ability to reduce Boolean expressions to a minimum form automatically is utilised in later work. So any Boolean function of a single variable can be written in terms of the 'atoms' x and $\neg x$. Indeed, they may reduce to a tautology [1,1] or a contradiction [0,0], but these are not atoms because an atom must be orthogonal to all other atoms and be of unit size. We should think of these 'atoms' as orthogonal unit vectors in a 2 dimensional Euclidean space, and as we do not think of matter comprising of nothing, or indeed everything, the Logic vectors [0,0] and [1,1] are not atoms.

We now extend these ideas to Boolean functions of more than one variable. The atoms of single variable functions arose very naturally from the truth tables, so the truth tables of Boolean functions of x and y should help the development of the 2 variable atoms.

```
                            ┌ x  1  1  0  0 ┐
TRUTH(TRUTH_TABLE(x, y))' = │               │
                            └ y  1  0  1  0 ┘
```

As it can be seen from the truth table above, for Boolean functions of two variables, the vector that represent their truth values have dimension $2^2 = 4$. We can see that the truth vector of x = [1,1,0,0] and that of y = [1,0,1,0], so neither x nor y are atoms in a 2 variable Boolean expression as they are not of unit length. Hidden in the truth table above is the structure of the atoms! Reading a 1 as a proposition and a 0 as its negation the truth table above can be read as;

$$\left[x \wedge y, x \wedge \neg y, \neg x \wedge y, \neg x \wedge \neg y\right]$$

We will now verify that $x \wedge y, x \wedge \neg y, \neg x \wedge y$ and $\neg x \wedge \neg y$ are indeed the atoms of Boolean functions of two variables x and y.

```
TRUTH(TRUTH_TABLE(x, y, x ∧ y, x ∧ ¬ y, ¬ x ∧ y, ¬ x ∧ ¬ y))
```

x	y	x ∧ y	x ∧ ¬ y	¬ x ∧ y	¬ x ∧ ¬ y
1	1	1	0	0	0
1	0	0	1	0	0
0	1	0	0	1	0
0	0	0	0	0	1

As we can see the Boolean expressions $x \wedge y$ and $x \wedge \neg y$ have the 'truth' vectors [1,0,0,0] and [0,1,0,0], respectively. We will now call theses 'truth' vectors, **Logic Vectors** to agree with the relevant literature.

We will now adopt the convention of omitting $\wedge$ operators in Boolean expressions (as product is implied, so is consistent with scalar algebra), so that large expressions do not appear as bulky. Also, we will adopt the $\bar{x}$ notation to represent the negation of x (i.e. $\neg x$), purely to save space.

The atoms for 3 variable Boolean expressions are;

$$\left[xyz, xy\bar{z}, x\bar{y}z, x\bar{y}\,\bar{z}, \bar{x}yz, \bar{x}y\bar{z}, \bar{x}\,\bar{y}z, \bar{x}\,\bar{y}\,\bar{z}\right]$$

and the number of atoms is $2^3 = 8$. So, in general the number of atoms for an n variable Boolean expression is 2^n.

T logic

A scalar algebraic model for classical logic

We seek an elementary algebraic model for classical logic. In other words a set of definitions and functions that allows 'normal' algebra to act in a Boolean fashion. The first problem that can be identified occurs with the simple $\wedge$ operator. We consider the $\wedge$ operator a product operator and $x \wedge x = x$, with elementary algebra then product of x and x is x^2. However, if we linearise (i.e. chop the exponent off) we can mimic the $\wedge$ operation.

This linearisation is defined by the function τ_x, which maps the polynomial $f(x)$ to the linear polynomial $r(x)$.

Any polynomial $f(x)$ can be written in the form

$$f(x) = x(1-x)q(x) + r(x) \tag{1.1}$$

where $q(x)$ and $r(x)$ are the quotient and remainder of $\dfrac{f(x)}{x(1-x)}$, respectively.

Clearly, the substitution of 0 or 1 into $x(1-x)$ will result in 0, hence at 0 and 1 $f(x) = r(x)$, which is linear in x. Hence, we have an efficient (i.e. fast) method of linearising any polynomial, with respect to a particular variable; divide the function by $x(1-x)$ and find the remainder.

Fortunately, DERIVE has an internal function `REMAINDER(u,v,x)`, which will evaluate the remainder of the quotient u/v with respect to the variable x.

The next few lines of DERIVE commands show the linearisation $\tau\left(x^3y^3 + x^2y^2 + xy + x + y + 1\right) = 3xy + x + y + 1$ in action.

```
          3  3    2  2
REMAINDER(x ·y  + x ·y  + x·y + x + y + 1, x·(1 - x), x)

    3    2
x·(y  + y  + y + 1) + y + 1

             3    2
REMAINDER(x·(y  + y  + y + 1) + y + 1, y·(1 - y), y)

x·(3·y + 1) + y + 1
```

The same expression in a slightly different form. We will now automate this process with user defined DERIVE functions. An extremely useful DERIVE function is `VARIABLES(u)`, which returns a vector of all the free variables in the expressions u, e.g. `VARIABLES(3xy+2x+3z)` returns `[x,y,z]` when evaluated. The next two functions automate the linearisation procedure.

```
t_AUX(w1w, v1v) := (ITERATE([r1r + 1, REMAINDER(u1u, v1v   ·(1 - v1v   ), v1v   )],
                                                     r1r          r1r       r1r

  [r1r, u1u], [1, w1w], DIMENSION(v1v)))
                                                   2

t(w1w) := t_AUX(w1w, VARIABLES(w1w))
```

Programming notes:

`r1r`, `w1w`, `v1v` are obscure variables to avoid clashes with functions that may be defined later and call this particular function. It is not wise practice to use the same variables in a functions that call one another. The `V SUB r` function extracts the r^{th} element of the vector `V`. The use of auxillary functions avoids the repetition of calculations, by passing evaluated results into other functions.

The τ function in action:

```
   3  3    2  2
t(x ·y  + x ·y  + x·y + x + y + 1) = x·(3·y + 1) + y + 1
```

We now have a function that will automatically linearise a polynomial of any order and any number of variables. Functions of this type, with two or more variables are called multi linear functions.

The τ function can be used to simplify a sequence of $\wedge$ operations on many variables, for example the Boolean expression $((x \wedge x \wedge y) \wedge (x \wedge y \wedge z)) \wedge (x \wedge z)$ converts to the elementary algebraic expression $((xxy)(xyz))(xz) = x^4y^2z^2$ and $\tau(x^4y^2z^2) = x\,y\,z$, which converts back to $x \wedge y \wedge z$.

More formally we have

$$x \wedge y \Leftrightarrow \tau(xy).$$

The $\vee$ operation is the of the form $\tau(x+y-xy)$. As a test we know that $y \vee y = y$, so using the definition for $\vee$ we get

$$\tau\left(y+y-y^2\right)=\tau\left(2y-y^2\right)=2y-y=y$$

so

$$x \vee y \Leftrightarrow \tau(x+y-xy).$$

The last definition is that of the negation operator and we have

$$\neg x \Leftrightarrow \tau(1-x).$$

As an example the Boolean expression $x \vee (\neg x \wedge y)$ simplifies to $x \vee y$. Using the above definitions this translates to

$$\begin{aligned}\tau(x+(1-x)y-x(1-x)y) &= \tau(x+y-xy-xy+x^2y)\\ &= x+y-xy\end{aligned}$$

which is of course $x \vee y$.

Orthonormal Expansions

In the ensuing work, it will be important to be able to express continuos algebraic expressions in their orthonormal form. For example, the expression $x+y-xy$ can be written as $xy+x(1-y)+(1-x)y+0(1-x)(1-y)$. As we can see each of the terms is the continuos equivalents of the Boolean atoms, e.g. $(1-x)y \equiv \neg x \wedge y$. Logic vectors naturally arise from expressions which are written in this orthonormal form. In our continuos Boolean logic, the expression $x \wedge y$ is equivalent to $x+y-xy$ and its Logic vector is [1, 1, 1, 0]. We can see from the identity

$$x+y-xy \equiv 1xy+1x(1-y)+1(1-x)y+0(1-x)(1-y)$$

the logic vector appears naturally from the coefficients of each of the terms.

More formally, the orthonormal system is defined as

$$\phi_i=\prod_{j=1}^{n} e\left(x_j\right) \quad \left(i=1,\cdots,2^n,\ j=1,\cdots,n\right)$$

where $e\left(x_j\right)=1-x_j$ or x_j. In other words the ϕ_i 's are Boolean atoms.

For example, if $n = 3$ then $\phi_2=e\left(x_1\right)e\left(x_2\right)e\left(x_3\right)=x_1x_2\left(1-x_3\right)$. The definition above is not precise enough to help us determine when $e\left(x_j\right)=1-x_j$ or x_j, as we could see in the example we chose $e\left(x_3\right)=1-x_3$ but $e\left(x_1\right)=x_1$ and $e\left(x_2\right)=x_2$. If we are to successfully automate this process a more precise definition is called for.

We define $e(x_j,p)=\begin{cases} x_j & \text{if } p=0 \\ 1-x_j & \text{if } p=1 \end{cases}$

and $BD(n,r)$ as the r^{th} digit (from the right) of n in binary form. For example, $BD(5,3)=1$ as $5=101_2$ and the third digit from the right is 1. Then we can now define ϕ_i as

$$\phi_i = \prod_{j=1}^{n} e\left(x_j, BD(i-1, n+1-j)\right) \quad \left(i = 1, \cdots, 2^n, j = 1, \cdots, n\right)$$

we will elaborate on how this definition works.

In the example above, we used $n = 3$ variables and this generates the orthonormal expressions

orthonormal expressions		p values
$x_1x_2x_3$		000
$x_1x_2(1-x_3)$		001
$x_1(1-x_2)x_3$	$\rightarrow$	010
$x_1(1-x_2)(1-x_3)$		011
$\vdots$		$\vdots$

we can see that the switch to decide whether p=0 or 1 forms a binary sequence starting at 0 and finishing at $2^n - 1$.

So that this definition may be coded we need a function that will convert a decimal integer in to its binary equivalent. To make use of the binary digits we will need to produce a vector of the binary digits, e.g. convert 5 to the vector [1,0,1]. The function `BINARY_AUX(m2m,n2n)` produces such a vector of the first `m2m` binary digits of the decimal integer `n2n`. This function relies on the division by 2 algorithm that is taught (or used to be) in most high school mathematics curricula.

```
BINARY_AUX(m2m, n2n) := REVERSE_VECTOR(ITERATES([(p2p - MOD(p2p, 2))/2, MOD(p2p, 2)],
   [p2p, r], [(n2n - MOD(n2n, 2))/2, MOD(n2n, 2)], m2m - 1)`
                                                             2)
```

We now define the functions

```
EE1(x1x, p1p) :=
  If p1p = 0
     x1x
     1 - x1x
     1 - x1x


                    n1n
φ(vars, i1i, n1n) := ∏  EE1(vars , (BINARY_AUX(n1n, i1i)) )
                    j=1        j                         j
```

to automate the above definitions.

To see these functions in action, we will find the orthonormal expressions for the variables $[x, y, z]$.

```
VECTOR(φ([x, y, z], i, 3), i, 0, 7)

[x·y·z, x·y·(1 - z), x·z·(1 - y), x·(y - 1)·(z - 1), y·z·(1 - x),
  y·(x - 1)·(z - 1), z·(x - 1)·(y - 1), (1 - z)·(x - 1)·(y - 1)]
```

To automate the process of producing a vector of the orthonormal expressions we define the functions:

```
                                                                        num
ORTHONORM_AUX(exps, vars, num) := VECTOR(φ(vars, i, num), i, 0, 2    - 1)

ORTHONORM(exps) := ORTHONORM_AUX(exps, VARIABLES(exps),

  DIMENSION(VARIABLES(exps)))
```

Our task is to convert an expression, say $xy - x + 1$, into its orthonormal form. We can do this by writing, for example,

$$xy - x + 1 = axy + bx(1-y) + c(1-x)y + d(1-x)(1-y)$$

Clearly, we can find the coefficient *a* by substituting x = 1 and y = 1, (1,1) into the above expression. We get

$$1\cdot 1 - 1 + 1 = a\cdot 1\cdot 1 + b\cdot 1\cdot 0 + c\cdot 0\cdot 1 + d\cdot 0\cdot 0 \;\Rightarrow\; a = 1$$

repeating with (1,0) ,(0,1) and (0,0) will give *b*, *c* and *d* respectively. Notice that we are subsitituting the coordinates of a square. In general, for functions of n variables we substitute the coordinates of an n dimensional hyper-cube into the function to find the elements of the logic vector.

The function `HYPER_CUBE(n3n)` produces a matrix of coordinates of the `n3n` unit hyper-cube.

```
                                                           n3n
HYPER_CUBE(n3n) := VECTOR(BINARY_AUX(n3n, w3w), w3w, 0, 2      - 1)
```

The following functions calcuates the Logic vector of a Boolean function in T logic form.

```
LOGIC_VECTOR_AUX(u2u, vars, vect) := VECTOR(      lim       u2u, r2r,
                                            vars→vect↓r2r

  DIMENSION(vect), 1, -1)


LOGIC_VECTOR_AUX2(u2u, vars) := LOGIC_VECTOR_AUX(u2u, vars,

  HYPER_CUBE(DIMENSION(vars)))


LOGIC_VECTOR(u2u) := LOGIC_VECTOR_AUX2(u2u, VARIABLES(u2u))
```

Using this function on $x\,y - x + 1$ we have;

```
LOGIC_VECTOR(x•y - x + 1) = [1, 0, 1, 1]
```

Logic Propositions from Regression Functions

Until the work of Tsukimoto and Morita there have not been any methods for extracting logical pro-posistions from regression planes. For example, the function

$$z = 0.71x - 0.99y + 0.25$$

is obtained from a least squares fit of a set of data points. What is the underlying Boolean structure of the data. Looking at the equation, if we consider in the region [0,1] near 0 is not true and near 1 is true. If we apply our method for finding Logic vectors to this regression line we will get a vector whose elements are in the interval [0,1] instead of the set $\{0,1\}$.

```
LOGIC_VECTOR(0.71·x - 0.99·y + 0.25) = [-0.03, 0.96, -0.74, 0.25]
```

We now need to find the descrete Boolean Logic vector that is closest to this continuous 'Logic vector'. If $\underline{C}$ is the continous Logic vector and $\underline{D}$ is the discrete Boolean Logic vector, $\underline{C}_i$ is the *i* th element of the continuous Logic vector and $\underline{D}_i$ is the *i* th element of the discrete Boolean Logic vector. Note that $\underline{D}_i = 0$ or 1. The nearest Boolean vector minimises $\sum\left(\underline{C}_i - \underline{D}_i\right)^2$. Minimising each term independently, we have that $\underline{D}_i = 1$ if $\underline{C}_i \geq 0.5$ and $\underline{D}_i = 0$ other wise. This approximation method is regarded as pseudo maximum likely hood method using the principle of indifference.

Applying this algorithm to the example above gives us the logic vector `[0, 1, 0, 0]` which happily is an atom and easily recognisable as $x \wedge \neg y$. So according to this method of approximation the nearest Logic proposition that describes the underlying Logic structure of the data is $z = x \wedge \neg y$.

The rest of this section describes the code for automating the extraction of the Logic propositions from multilinear regression hyper planes.

The maximum likely hood approximation is coded as

```
MAX_LIKELY_HOOD_AUX(vect1) := VECTOR(IF(vect1    < 0.5, 0, 1), rrr, 1,
                                             rrr

  DIMENSION(vect1))


MAX_LIKELY_HOOD(ff) := MAX_LIKELY_HOOD_AUX(LOGIC_VECTOR(ff))
```

The maximum likely hood expression is automated with:

```
BOOLEAN_LIKELY_HOOD(ff) := MAX_LIKELY_HOOD(ff)·ORTHONORM(ff)
```

e.g. `BOOLEAN_LIKELY_HOOD(0.71·x - 0.99·y + 0.25) = x·(1 - y)`.

Now the task is to convert this algebraic expression into its Boolean logic equivalent. This requires some inventive programming within the DERIVE environment. As space for this article is short listings and their purpose are given, the detail is left to the reader.

```
NEGATE(xx, rr) :=
  If rr = 1
     xx
     ¬ xx
```

Makes a variable `xx` to `¬xx` if `rr = 1`, leaves as `xx` else.

```
                                                   nn
PROP_BINARY_AUX(nn) := VECTOR(BINARY_AUX(nn, xx), xx, 2   - 1, 0, -1)

PROP_BINARY(ff) := PROP_BINARY_AUX(DIMENSION(VARIABLES(ff)))
```

Produces a hyper cube for a given expression of dimension

```
NEGATE_MAT_AUX(www, ddd) := VECTOR(VECTOR(NEGATE(www   ,
                                                     rr

                                                 ddd
  (PROP_BINARY_AUX(ddd))      ), rr, 1, ddd), ss, 1, 2   )
                          ss,rr
```

```
NEGATE_MAT_AUX_1(www) := NEGATE_MAT_AUX(www, DIMENSION(www))

NEGATE_MAT(ff) := NEGATE_MAT_AUX_1(VARIABLES(ff))
```

Produces an $n \times 2$ matrix with the n variables of `ff` in the first column and their Boolean negation in the second column.

```
VECT_AND_AUX(vars, nnn) := (ITERATES([rrr + 1, vvv ∧ vars        ],
                                                       rrr + 1

  [rrr, vvv], [1, vars ], nnn - 1))
                      1
                                  nnn,2

VECT_AND(vect) := VECT_AND_AUX(vect, DIMENSION(vect))
```

Converts a vector $[x, y, z, ...]$ into the logical expression $x \wedge y \wedge z \wedge \cdots$

```
VECT_OR_AUX(vars, nnn) := (ITERATES([rrr + 1, vvv ∨ vars        ],
                                                      rrr + 1

  [rrr, vvv], [1, vars ], nnn - 1))
                      1
                                  nnn,2

VECT_OR(vect) := VECT_OR_AUX(vect, DIMENSION(vect))
```

Converts a vector $[x, y, z, ...]$ into the logical expression $x \vee y \vee z \vee \cdots$

```
ZERO_FALSE(vect) := VECTOR(IF(vect  = 0, false, vect , vect ), r, 1,
                                  r                r      r

  DIMENSION(vect))
```

Converts any 0' in a vector to the statement false.

```
PROPOSITION_AND_AUX(ff, nmat) := VECTOR(VECT_AND(nmat  ), r_, 1,
                                                     r_

   DIMENSION(VARIABLES(ff)))
  2

PROPOSITION_AND(ff) := PROPOSITION_AND_AUX(ff, NEGATE_MAT(ff))
```

Creates a vector of all atoms for `ff`, e.g if `ff=xy+x` then this function produces the vector $[x \wedge y, x \wedge \neg y, \neg x \wedge y, \neg x \wedge \neg y]$

```
ELEMENT_PROD(vect1, vect2) := VECTOR(vect1   ·vect2   , r__, 1,
                                          r__      r__

  DIMENSION(vect1))
```

Produces an element by element vector product (Logic And).

```
EXTRACT_PROP(ff) := VECT_OR(ZERO_FALSE(ELEMENT_PROD(MAX_LIKELY_HOOD(ff),
  PROPOSITION_AND(ff))))
```

This final function takes any multilinear function and produces a pseudo-maximum likelyhood Boolean function that best describes the underlying Boolean structure.

Experiment

These functions will generate 'noisy' data that has an underlying Boolean structure of $x \wedge \neg y \wedge z$.

```
⎡F3(x) :=     , G3(y) :=     , H3(z) :=  ⎤
⎢  If x < 0.5    If y < 0.6    If z < 0.4⎥
⎢     0             0             0      ⎥
⎣     1             1             1      ⎦

PP(t1t) :=
  If t1t = 0
     RANDOM(1)/2
     RANDOM(1)/2 + 1/2
     RANDOM(1)/2 + 1/2

APPEND(APPEND(VECTOR(VECTOR(VECTOR([x, y, z, PP(F3(x) ∨ (¬ G3(y) ∧
  H3(z)))], x, 0, 1, 0.25), y, 0, 1, 0.25), z, 0, 1, 0.25)))
```

Using DERIVE's FIT() function we can find a multilinear regression curve of the form $w = a\,x\,y\,z + b\,x\,y + c\,x\,z + d\,y\,z + e\,x + f\,y + g\,z + h$. In this experiment it turns out to be

```
0.932032·x·y·z - 0.0892345·x·y - 0.817753·x·z + 0.629476·x - 0.807328·y·z +
+ 0.103246·y + 0.724527·z + 0.188913
```

We now use our function to exctract the underlying Boolean proposition

```
EXTRACT_PROP(0.932032·x·y·z - 0.0892345·x·y - 0.817753·x·z + 0.629476·x -
  0.807328·y·z + 0.103246·y + 0.724527·z + 0.188913)
```

```
x ∨ (¬ y ∧ z)
```

Success!!

References

[1] The Discovery of Logical Propositions from Noisy Data, Hiroshi Tsukimoto, AAAI Workshop on Knowledge Discovery in Databases, pp 205-216.

Comment for DERIVE 6.10: The latest version of DERIVE confirms all intermediate results. An exception is the final outcome because of the leading zeros in the logic expression:

```
0 ∨ 0 ∨ 0 ∨ x ∨ (¬ y ∧ z)
x ∨ (¬ y ∧ z) ∨ 0
```

Hi Terence, can you explain and adjust the respective function(s)?

Dear Editor,
We are Ukrainian followers of Information Technologies (in particular *DERIVE* and *CABRI*, unfortunately not *TI-92* due to financial circumstances) in Mathematics education for many years. We are now subscribers of your remarkable journal. In this matter we send you our article which maybe interesting for publishing in your journal.
With best regards to you and your journal,
sincerely yours,
Dr. Rakov S.A. and Dr. Gorokh V.P.

Information Technologies in Geometry

(An example of the generalization of a well-known problem about squares)

Rakov S.A. & Gorokh V.P.

Kharkov State Pedagogical University named after G.S.Skovorada
tel/fax: (+380) (0572) 676-604
e.mail: Rakov@pu.ac.Kharkov.UA

Introduction

Information technologies have changed all kinds of human activities. Mathematics is not the exception - it becomes technologically dependent. The most important changes take place in the process of doing mathematics - discovering new facts and their proof. Special mathematical packages offer the user a suitable environment for arranging computer experiments in the problem field with the aim of finding mathematical regularities on the first step of exploration and then support the process of proof with the powerful opportunities of computer algebra. No doubt that the future of mathematics is a symbiosis of man and computer. Using mathematical packages becomes the inalienable component of mathematical culture.

Innovative trends in mathematical education are lying in the framework of constructive approach - involving students in the process of constructing their own mathematical system which consists of mathematical knowledge and mathematical beliefs. One of the most effective ways of the realization of a constructive approach is the method of learnig explorations when students explore open-ended problems on their own. Solving open-ended problems can be regarded as the model of the professional mathematical work. Therefore it is naturally to use information technologies in mathematical education just in the same manner: computer experiments as the source of powerful ideas, computer algebra as a tool of deductive method. Using information technologies in arranging learning explorations and carrying out the proofs cannot only do this work more effective but acquaint students with modern technologies of mathematics.

The authors have had the pleasure of playing with Cabri-Geométre in one well-known problem [1] and as a result have conjectured its generalization which then they have proved by the help of *DERIVE*. They were highly impressed with these computer games and decided to share their experience with their students. Thereby the 2-week computer practice with undergraduate students at the Mathematical Teachers Department of Kharkov State Pedagogical University in 1997 was organized in the following form: 1st week - acquainting with packages Cabri-Geométre and *DERIVE*, 2nd week playing with the above mentioned problem with the aim of its generalization. The students have coped with the task posed successfully without any help. The current article outlines activities which were used in the process of generalization of this problem (it is rather common for authors and students).

Problem

The problem area is:

Two sqares $A_1QC_1D_1$ and $A_2B_2C_2Q$ with centers O_1 and O_2 and common vertex Q with identical orientation are given. Let E and F the midpoints of the segments A_1A_2 and C_1C_2. Then the quadrangle O_1EO_2F is a square as well.
How can the problem mentioned above be generalized?

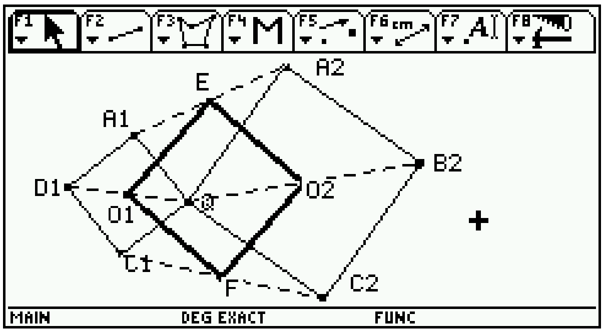
Computer experiments in solving the problem

Experiment 1

At first we construct a computer model of a given problem in the Cabri-Geométre environment. For that purpose we first construct a macro-construction *quad* which draws a square using two given points at the endpoint of the diagonal.
(The authors presented Cabri constructions. I reproduced the constructions with the TI-92´s Geometry Tool which is a Cabri implementation on the TI-92. You can find the .92a files on the Diskette of the Year. Josef)

Produce a computer **Model 1** of the problem:

(1) Construct a macro *quad* which draws a square on two given points at the ends of the diagonal.
(2) Define three points D_1, Q and B_2 and draw the squares $A_1QC_1D_1$ and $A_2B_2C_2Q$ using pairs of points D_1, Q and Q, B_2 as endpoints of the diagonals to apply the macro *quad*.
(3) Draw the segments A_1A_2, QB_2, C_1C_2 and D_1Q.
(4) Define the points E, O_2, F and O_2 as the midpoints of the segments constructed in the previous step (3).

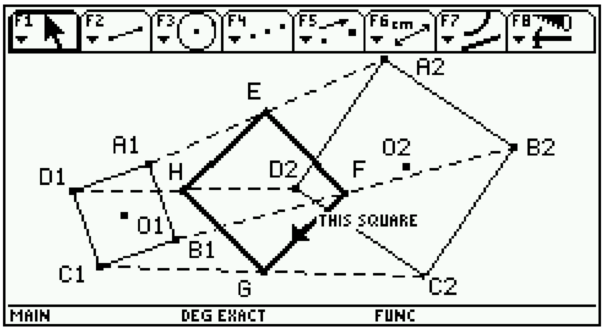


Now the dynamic model is ready and we can play with it. Really our model is dynamic ("alive") in the sense we can dynamically change its parameters. All initial points D_1, Q and B_2 are moveable - they can be moved with the little hand , (with the mouse in the PC-Cabri). Playing with this model we really see that the third - depending - quadrangle is a square (we can convince in it by inspection or by measuring the parameters of a figure with the Cabri Geométre instruments and/or by applying again the macro on E,F or O_1,O_2). Of course our experimental assurance needs the deductive proof, it is only the hypothesis, but this hypothesis is of a great confidentiality. We put aside now the attempts to prove the hypothesis and proceed our computer experiments. (*File exp1.92a*)

Experiment 2 (disjointing the vertices)

Is it really important that the initial squares have a vertex in common?

Unfortunately, the previous model cannot be modified for these purposes. Therefore we must repeat the algorithm discribed above with a little change in the second step - the initial points will be four independent points B_1, D_1, B_2 and D_2, the diagonal points of the future quadrates.

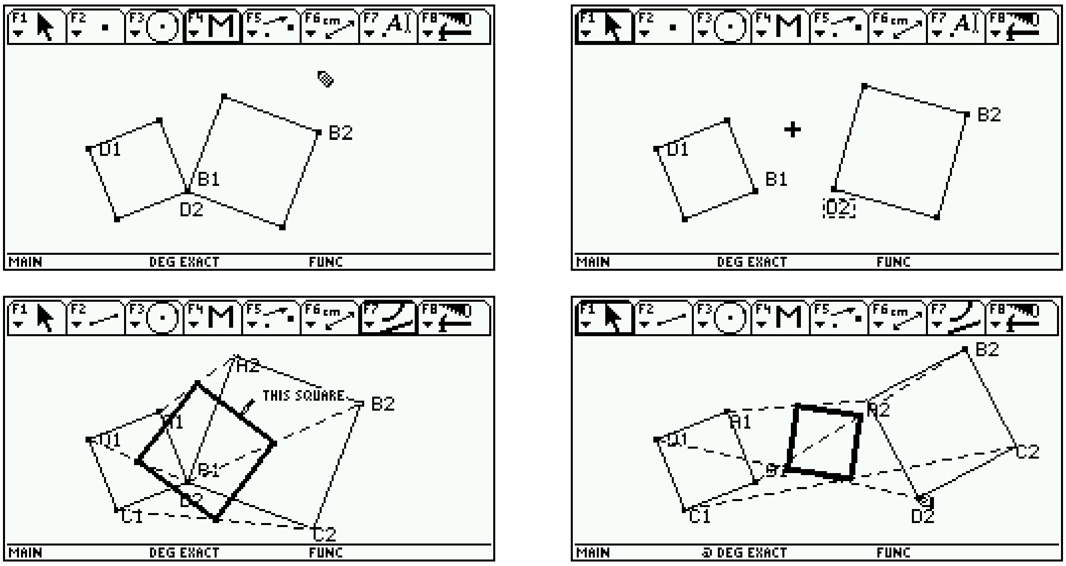


By playing with a new **Model 2** we can convince ourselves that in this case the resulting quadrangle is a square as well. As a result of the experiments described we can formulate the next hypothesis:

Generalization 1

Two squares $A_1B_1C_1D_1$ and $A_2B_2C_2D_2$ with centers O_1 and O_2 of identical orientation are given. Let E, F, G and H the midpoints of the segments A_1A_2, B_1B_2, C_1C_2 and D_1D_2. Then the quadrangle $EFGH$ is a square as well.

*I tried to generalize model 1 on the TI-92 and was lucky enough to be successful. You only have to define two points B_1 and D_2 on the same spot and then produce the initial squares. Whenever you are pointing to that "double point" you are asked "Which object?". You have only to choose one of the two answers proposed. The four TI-92 screen shots are showing first only the two squares - first connected and then disconnected - and in the second row you can see **model 1** followed by **model 2.** Josef*



(files exp2.92a and exp22.92a on the diskette)

Experiment 3 (moving the middle points)

Why must the vertices of the resulting figure be the midpoints?

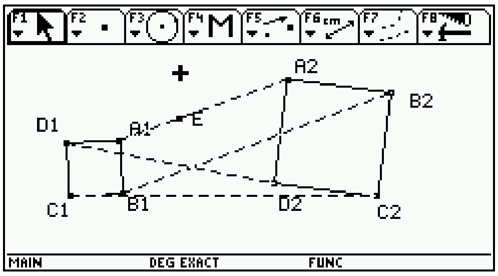
Maybe the result remains valid in the case of arbitrary points which divide the segments joining correspondent points in a given proportion?

Modify our previous model. For that purpose a new macroconstruction *divide* would be needed, which divides a segment in a ratio given by the segment and a point on it. Such a macro can be defined with the ideas of the *Fale´*s theorem for example. Then we produce **Model 3** using this macro by picking an arbitrary point E on the segment A_1A_2 and then define all the other points F, G and H applying *divide* as points which divide the correspondent segments in the same ratio.

Now again with the mouse (*or with the* ,) we can change the sizes of the squares and their mutual positions and the position of the division points as well. As in the previous case we can easily convince ourselves that the resulting figure is a square again. Thus we obtain the next generalization.

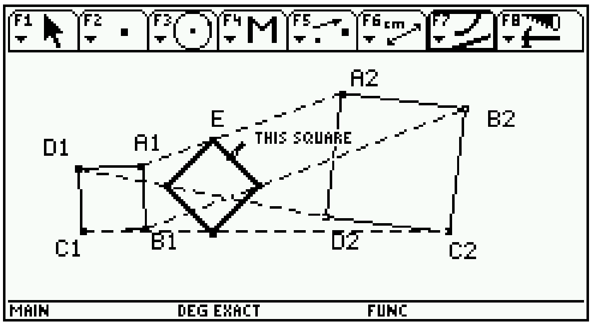
Generalization 2

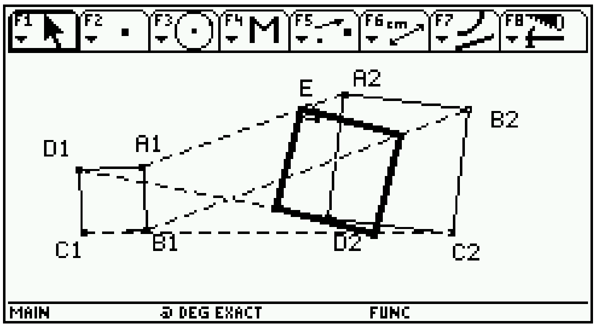
Two squares $A_1B_1C_1D_1$ and $A_2B_2C_2D_2$ with centers O_1 and O_2 of identical orientation are given. Let *E, F, G* and *H* points of the segments A_1A_2, B_1B_2, C_1C_2 and D_1D_2 respectively which divide them in the same ratio. Then the quadrangle *EFGH* is a square as well.



You can find the TI-92 realization on the Diskette of the Year 1998 saved as exp3.92a. The macro divide can be applied pointing first to the given segment (eg A_1A_2), then to the given dividing point (E) and then to the segment to be divided in the given ratio $A_1E : EA_2$. Josef

(*File exp3.92a*)





Proof in the DERIVE environment

```
#1:  "SQUARES.MTH, Rakov & Gorokh"

#2:  CaseMode := Sensitive
```

Let $O_1(a,b)$ and $O_2(c,d)$ the centers of the squares $A_1B_1C_1D_1$ and $A_2B_2C_2D_2$. n_1 and n_2 are the vectors O_1A_1 and O_2A_2.

```
#3:  [O1 := [a, b], O2 := [c, d], n1 := [p, q], m1 := [-q, p], n2 := [r, s],
     m2 := [-s, r]]

#4:  [A1 := O1 + n1, B1 := O1 + m1, C1 := O1 - n1, D1 := O1 - m1]      User

#5:  [A2 := O2 + n2, B2 := O2 + m2, C2 := O2 - n2, D2 := O2 - m2]      User
```

Define the function `divide`:

```
                          x + t·y
#6:  divide(x, y, t) :=  ─────────                                      User
                           1 + t
```

Declare the coordinates of the points *E, F, G* and *H* which divide the segments A_1A_2, B_1B_2, C_1C_2 and D_1D_2 in the ration $t : 1$.

```
#7:  [E := divide(A1, A2, t), F := divide(B1, B2, t), G := divide(C1, C2, t),
     H := divide(D1, D2, t)]
```

Find the difference of the vectors *HE* and *GF*. Simplifying this expression we obtain `[0,0]`. Consequently the quadrangle *EFGH* is a parallelogram.

```
#8:  (E - H) - (F - G) = [0, 0]                          User=Simp(User)
```

Evaluate now the scalar product of the vectors *GF* and *FE*, which simplifies to `0`. Thus we have proved that the quadrangle *EFGH* is a rectangle.

```
#9:  (F - G)·(E - F) = 0                                  User=Simp(User)
```

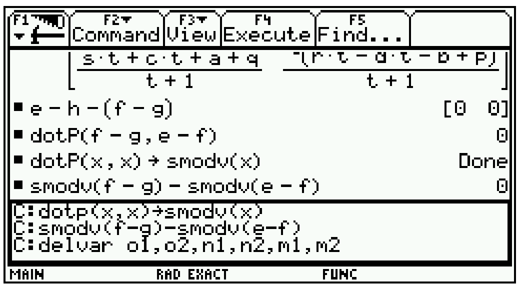
Compare the lengths of the sides *GF* and *FE* of this rectangle. For this purpose declare a new function `SModV`(**S**quare of the **Mod**ule of the **V**ector). Simplification of expression #11 gives `0`, consequently the rectangle is a square. Note that the square *EFGH* can degenerate in a point.

```
#10: SModV(x) := x·x                                      User
#11: SModV(F - G) - SModV(E - F) = 0                      User=Simp(User)
```

(*If you are able to realize Cabri and the CAS of DERIVE on a TI-92 you will proof the conjecture on the TI-92. I add the script* proof. *Loading it in the TextEditor you can execute all lines pressing* † . *Please note that I had to use the name* `divid` *for the function*, *because* `divide` *is in use (for the macro) and that I am not allowed to use the reserved system variables* `c1` *and* `c2`. *Josef*)

```
C:[a,b]»o1:[c,d]»o2:[p,q]»n1:[ªq,p]»m1
C:[r,s]»n2:[ªs,r]»m2
C:o1+n1»a1:o1+m1»b1:o1-n1»cc1:o1-m1»d1
C:o2+n2»a2:o2+m2»b2:o2-n2»cc2:o2-m2»d2
C:(x+t*y)/(1+t)»divid(x,y,t)
C:divid(a1,a2,t)»e:divid(b1,b2,t)»f
C:divid(cc1,cc2,t)»g:divid(d1,d2,t)»h
```

```
C:(e-h)-(f-g)
C:dotp((f-g),(e-f))
C:dotp(x,x)»smodv(x)
C:smodv(f-g)-smodv(e-f)
C:delvar o1,o2,n1,n2,m1,m2
C:delvar a1,a2,b1,b2,cc1,cc2,d1,d2
C:delvar divid,smodv
```



Further generalization

Experiments with arbitrary similar quadrangles allow one to formulate the following generalization:

Generalization 3

Let F_1 and F_2 two similar figures in the plain of the same orientation (it means that there exists a similarity f of the first class, which maps the figure F_1 onto the figure F_2. For each point X_1 of figure F_1, and a point $X_2 = f(X_1)$ of figure F_2 define a point X, which divides the segment X_1X_2 in a ratio t with $t = X_1X : XX_2$. Then figure F, which consists of all such points X is similar to the two original figures or degenerates to a single point.

Proof this conjecture!

The radius-vector of a point M will be denoted as $\overline{M}$ in the following.

Let X_1 and Y_1 two arbitrary points of figure F_1. Let $X_2 = f(X_1)$ and $Y_2 = f(Y_1)$. X and Y are the points dividing the segments X_1X_2 and Y_1Y_2 in a ratio t.

Then: $$\overline{X} = \frac{\overline{X_1} + t\,\overline{X_2}}{1+t}, \quad \overline{Y} = \frac{\overline{Y_1} + t\,\overline{Y_2}}{1+t}.$$

As a consequence we obtain: $$\overline{XY} = \frac{\overline{X_1Y_1} + t\,\overline{X_2Y_2}}{1+t}$$

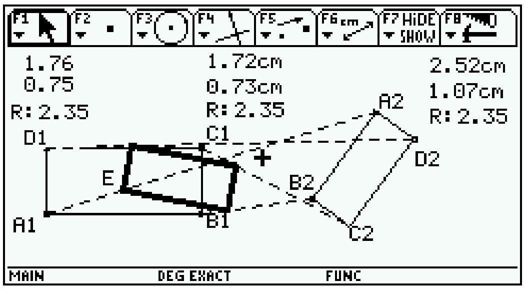
Therefore we have: $$\left|\overline{XY}\right|^2 = \frac{\overline{X_1Y_1}^2 + 2t\,\overline{X_1Y_1}\cdot\overline{X_2Y_2} + t^2\,\overline{X_2Y_2}^2}{(1+t)^2}$$

Denote the angle beween the ray X_1Y_1 and its image under the similarity f as φ. This angle φ is constant because the mapping f is a similarity of the first class. With respect to the relation $|X_2Y_2| = k|X_1Y_1|$, where k is the coefficient of the similarity we finally obtain:

$$|XY|^2 = \frac{1 + 2t\cos\varphi + t^2k^2}{(1+t)^2}|X_1Y_1|^2$$

Consequently $|XY| = C\,|X_1Y_1|$, where C is a konstant.

Thus figure F is similar to figure F_1 if $k > 0$. In the case $k = 0$ figure F is a point.



I realized Generalization 3 on the TI-92 working with two similar rectangles $A_1B_1C_1D_1$ and $A_2B_2C_2D_2$ and the dividing point E. As you can see the ratio of the lengths of the sides is the same for the three figures (= 2.35). Josef

(*File exp4.92a*)

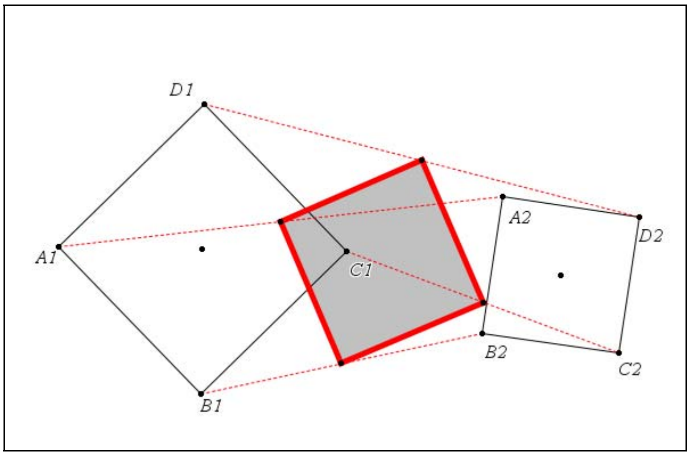
Remark that in the most general case (**Generalization 3**) the proof was done without support of the computer and was more simple and natural. It is a rather general situation because from the general point of view the unimportant details are disappearing and the matter of fact becomes more clear and obvious. Nevertheless the computer experiments have played the substantial role in the process of generalization. By the way, the proof discussed above could be done on a computer as well but all the analytic calculations were so simple that computer support was unnecessary.

Last remarks

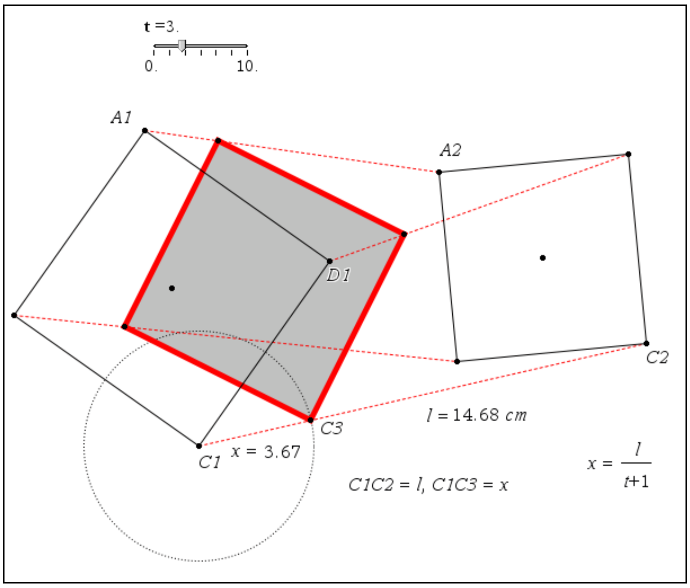
As it can be clearly seen from the proof given above the fact stays valid in the case of an arbitrary dimension (not only in the case of 2D, but 3D, 4D etc. as well).
The successfully proved theorem can be used in computer animation for modeling continuous transformations of figures in plane or space.
The authors of the article welcome all the enthusiasts of mathematics and computing to continue with computer explorations in the discussed problem field in particular in searching for further generalizations and applications of the obtained results.

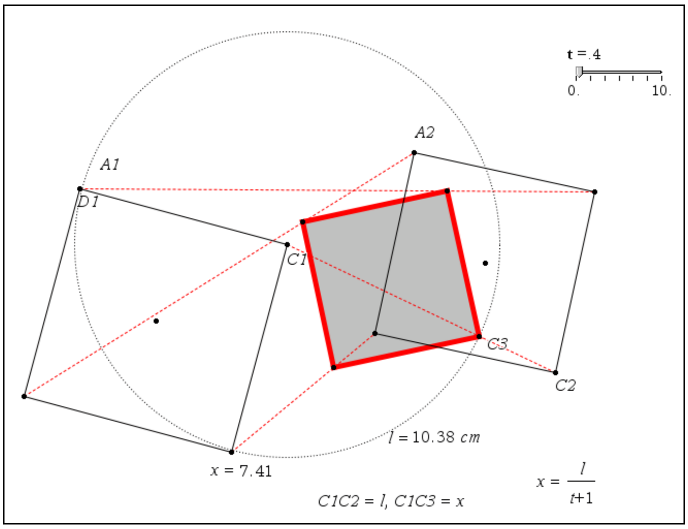
Literature: [1] Boltyansky V. G. , About one parquet, Mathematics in School, 1984, Nr 1, p65 - 66

The squares and TI-NspireCAS (by Josef Böhm)



This is generalization 2 performed on the Geometry page of TI-Nspire:





The ratio can be changed by the slider. Distance *x* is calculated according the given formula. It is the distance between the vertices of the first square to the division points.

There are no scripts available with TI-Nspire. But instead of this we have the Notes Application which can be used in a similar way.

Left screen:

$\begin{bmatrix} s \end{bmatrix}$

$m2:=\begin{bmatrix} -s \\ r \end{bmatrix}$

a1:=o1+n1

b1:=o1+m1

c1:=o1−n1

d1:=o1−m1

a2:=o2+n2

b2:=o2+m2

c2:=o2−n2

d2:=o2−m2

$divide(xx,yy,tt):=\dfrac{xx+tt\cdot yy}{1+tt}$

e:=divide(a1,a2,tt)

f:=divide(b1,b2,tt)

g:=divide(c1,c2,tt)

h:=divide(d1,d2,tt)

e−h−(f−g)

dotP(f−g,e−f)

smodv(xx):=dotP(xx,xx)

smodv(f−g)−smodv(e−f)

Right screen:

$divide(xx,yy,tt):=\dfrac{xx+tt\cdot yy}{1+tt}$ ▸ *Done*

$e:=divide(a1,a2,tt)$ ▸ $\begin{bmatrix} \dfrac{tt\cdot r+a+c\cdot tt+p}{tt+1} \\ \dfrac{tt\cdot s+b+d\cdot tt+q}{tt+1} \end{bmatrix}$

$f:=divide(b1,b2,tt)$ ▸ $\begin{bmatrix} \dfrac{-(tt\cdot s-a-c\cdot tt+q)}{tt+1} \\ \dfrac{tt\cdot r+b+d\cdot tt+p}{tt+1} \end{bmatrix}$

$g:=divide(c1,c2,tt)$ ▸ $\begin{bmatrix} \dfrac{-(tt\cdot r-a-c\cdot tt+p)}{tt+1} \\ \dfrac{-(tt\cdot s-b-d\cdot tt+q)}{tt+1} \end{bmatrix}$

$h:=divide(d1,d2,tt)$ ▸ $\begin{bmatrix} \dfrac{tt\cdot s+a+c\cdot tt+q}{tt+1} \\ \dfrac{-(tt\cdot r-b-d\cdot tt+p)}{tt+1} \end{bmatrix}$

$e-h-(f-g)$ ▸ $\begin{bmatrix} 0 \\ 0 \end{bmatrix}$ ⚠

dotP(f−g,e−f) ▸ 0 ⚠

smodv(xx):=dotP(xx,xx) ▸ *Done*

smodv(f−g)−smodv(e−f) ▸ 0 ⚠

What is to be executed on TI-92 and V200 is here written in so called *Math Boxes*.

In the left screen shot all evaluations are suppressed. Now you have the choice evaluating one box after the other and following the process step by step

or

you can evaluate the “script” from the first row to the last one as a whole.

I didn’t use variable *t* because it is used as slider variable in the Geometry App. Same for variable *x*. The “traffic sign” is a warning: *Domain of the result might be larger than the domain of the input.*

In spring 1997 a Teachers´ Conference took place in Münster, Germany, to demonstrate applications of modern technologies in teaching natural sciences. Among many interesting lectures I attended Helmut Wunderling´s DERIVE supported investigations of Parallel Curves. I asked him to publish his lecture in the DNL and some weeks later I received a summary of his ideas, which I tried to translate. Josef

Parallel Curves

Helmut Wunderling, Berlin, hwunder@berlin-snafu.de

Starting point for the following activity working with parallel curves was the following question: "What is the location of the center of gravity of a lamp-shade with exponential cross section. We did neither have in mind a full body nor a virtual shade with wall thickness 0. The shade should have a constant thickness *a*. Its inner face is follows an exponential function, so we have to solve the problem to find a function for the form of the outer face.

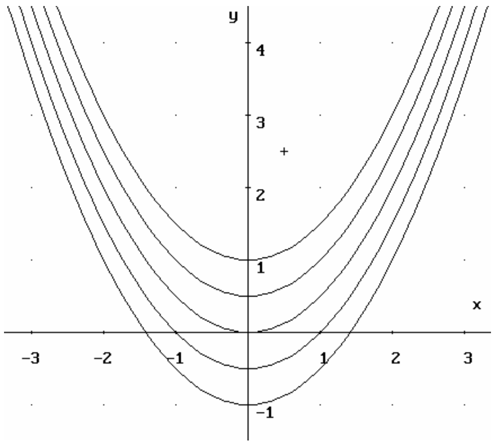
Generalizing this very special problem we pose the following task:

Given is a funktion *f* with its graph *Gf* and a distance *a*. Find a function *g* with graph *Gg* parallel to *Gf* in a constant distance *a*.

To treat that problem very early we replaced the exponential form by a wine glass with a "nice" parabola as its outer face. So $f(x) = c\,x$^2. (You can choose your own *c*. Let´s take $c = 1/2$)

Now the teacher asks the students to find function *g* for a meaningful *a*.

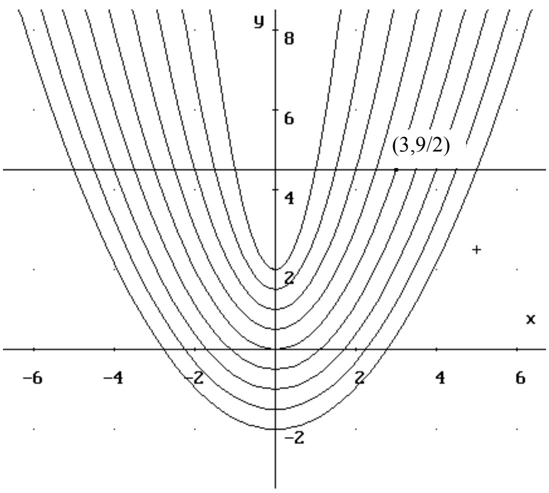
1st Solution:



Shifting the parabola in vertical direction by *a*. Hence $g(x) = f(x) + a$.
DERIVE shows that there are no parallel curves. Astonshing, but at last easy to understand.

Theorem: Parallel shifting of curves does not result in parallel curves, unlike doing this with straight lines

2nd Solution:



We spread out the wanted parabola in such a way that we have the (horizontal) distance *a* at the upper rim of the glass, too.

So $g(x)= C*x^2 + a$, where $C=\dfrac{c\,x_r^{\,2}-a}{a^2+2a\,x_r^{\,2}+x_r^{\,2}}$ and

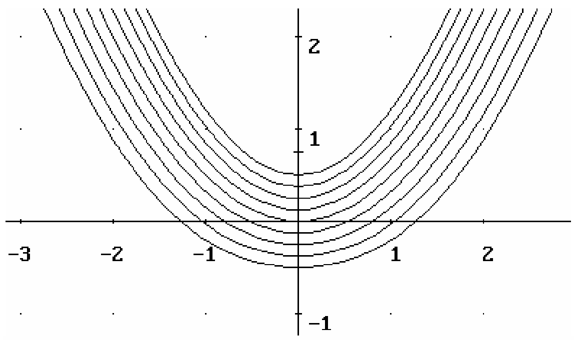
(x_r,y_r) are the coordinates of a point of the rim. *DERIVE* shows again that only small distances *a* are resulting in approximately parallel curves.

(In a Basic Course this could be sufficient, but in this case we have to be satisfied with an approximation!)
(point on the rim: (3,4.5), $a = \pm\, 0.5\, k$)

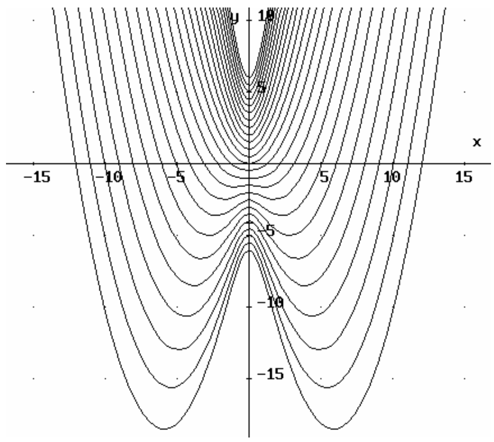
3rd Solution:

Now we will consider the slope of the cross section, i.e. the direction of the tangent and then we will calculate an appropriate correction $k(x,a)$ using the perpendicular distance a:
$g(x)=f(x)+k(x,a)$, with $k(x,a)=a$*SQRT($f'(x)$^2 +1).
Experimenting with various distances a (positive and negative) *DERIVE* presents terrible surprises (outside of the origin parabola).

```
F(x) := x^2/2
```



$$\text{VECTOR}\left(F(x) + a\cdot\sqrt{1+\left(\frac{d}{dx}F(x)\right)^2},\ a,\ -\frac{1}{2},\ \frac{1}{2},\ \frac{1}{8}\right)$$



$$\text{VECTOR}\left(F(x) + a\cdot\sqrt{1+\left(\frac{d}{dx}F(x)\right)^2},\ a,\ -6,\ 6,\ 0.5\right)$$

4th Solution:

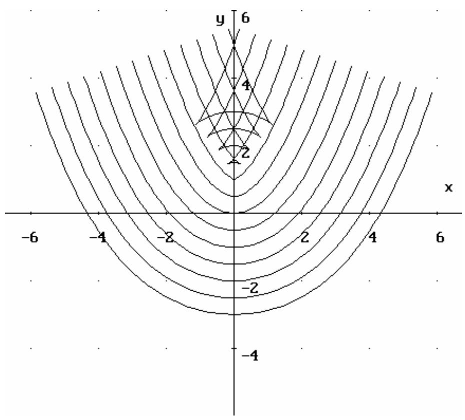
Let´s consider now the fact that we want to plot simultaneously to each point P of the original curve the point Q of the parallel curve → points $P(t)$ and $Q(t)$. Following this idea we find the perfect solution of our problem. (only in the Advanced Course). Here we include calculation with vectors. And this is my urgent demand:

> Don't teach Calculus and Linear Algebra separated from each other in upper secondary level but use a connected approach where- and whenever it is possible.

Let $vp(t) = [xp(t),\ yp(t)]$ the position vector of $P(t)$. Then $vq(a,t) = vp(t) + a\cdot vne(t)$ is the position vector of $Q(t)$ where $vne(t)$ is the perpendicular unity vector.

In our special case we have:

$$\text{VECTOR}\left(\left[t,\ \frac{t^2}{2}\right] + \frac{a}{\sqrt{(t^2+1)}}\cdot[-t,\ 1],\ a,\ -3,\ 3,\ 0.5\right)$$



DERIVE can show that there are really parallel curves.

Really for each real a? No, there are surprises again - and now inside of the given parabola. But in this case we find an obvious reason for that misbehaviour: inside of a narrow parabola there is no space for a second one in a distance a. A new point of discussion is born:

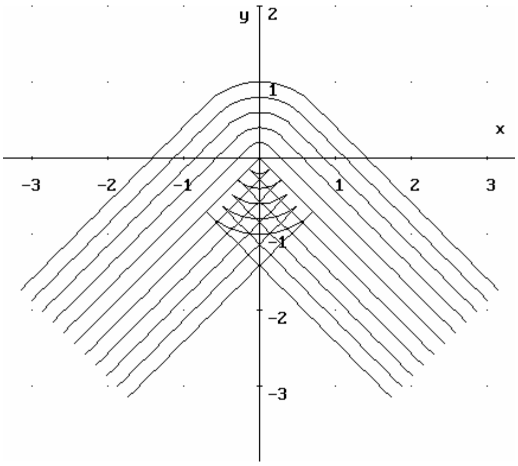
What is the maximum thickness of the glass? (from now we will only consider parallel curves inside of convex curves).

In generalized form we have the following definitions:

```
F(t) :=

VP(t) := [t, F(t)]

         d
VR(t) := ── VP(t)
         dt

VN(t) := [- (VR(t))  , (VR(t))  ]
                   2           1

                   a
AVP(a, t) := ─────────·VN(t)
              |VN(t)|

VQ(a, t) := VP(t) + AVP(a, t)
```

Using a piecewise defined curve - lines and a quarter circle - (instead of the parabola) we can observe, that the parallel curve with maximum distance is marking time, even when curve plotting goes on. ($a = -1$) Therefore we use the "plot velocity" of the parallel curve, which is the derivative of $vq(a,t)$. If $vq'(a,t)$ is the zero vector we can determine the maximum distance $b(t)$ for each t.

```
F(t) :=
  If t < - √2/2
     t + √2
     If t < √(1/2)
        √(1 - t^2)
        √2 - t

VECTOR(VQ(a, t), a, -2, 0, 0.2)
```



We return to the parabola from above:

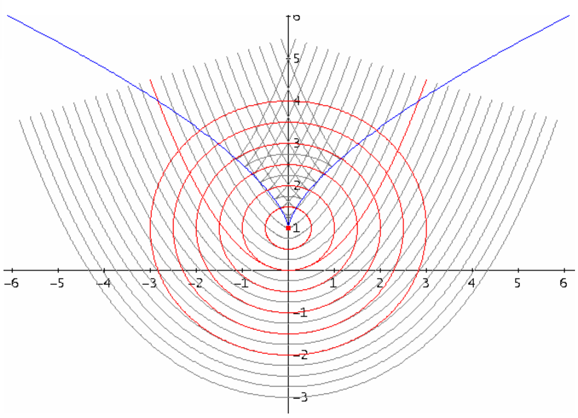
$$F(t) := \frac{t^2}{2}$$

$$\text{SOLVE}\left(\frac{d}{dt}\,\text{VQ}(a,\ t) = [0,\ 0],\ a\right) = \left(a = (t^2 + 1)^{3/2}\right)$$

So for $t = 0$ the maximum distance is $a = 1$ (which is the radius of curvature).

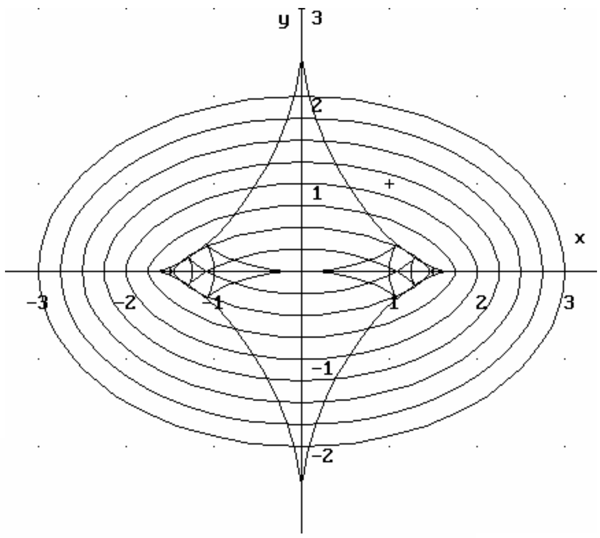
$$\text{VQ}\left((t^2 + 1)^{3/2},\ t\right)$$

$$\left[-t^3,\ \frac{3\cdot t^2}{2} + 1\right]$$



By the way, the points on the blue curve (evolute) are the centres of curvature for all parallel curves.

For example, you obtain a general view of the maximum wall thicknesses of a dragée - an ellipsoid of revolution with contour *vp*(*t*) = [3 sin(*t*), 5 cos(*t*)] if you plot *vq*(*b*(*t*),*t*). This leads us immediately to the well-known evolute - an astroide. (The locus of the "maximum distance points".)



$$\mathrm{SOLVE}\left(\frac{d}{dt}\,\mathrm{VQ}(a,\ t) = [0,\ 0],\ a\right) = \left(a = (t^2 + 1)^{3/2}\right)$$

$$\mathrm{VP}(t) := [3\cdot\mathrm{SIN}(t),\ 2\cdot\mathrm{COS}(t)]$$

$$\mathrm{VECTOR}(\mathrm{VQ}(a,\ t),\ a,\ -2,\ 0,\ 0.25)$$

$$\mathrm{SOLVE}\left(\frac{d}{dt}\,\mathrm{VQ}(a,\ t) = [0,\ 0],\ a\right) = \left[\left(\mathrm{COS}(t) = 0 \wedge \mathrm{SIN}(t) = 0\right) \vee a = -\frac{(5\cdot\mathrm{COS}(t)^2 + 4)^{3/2}}{6}\right]$$

$$\mathrm{VQ}\left(-\frac{(5\cdot\mathrm{COS}(t)^2 + 4)^{3/2}}{6},\ t\right) = \left[\frac{5\cdot\mathrm{SIN}(t)^3}{3},\ -\frac{5\cdot\mathrm{COS}(t)^3}{2}\right]$$

DERIVE allows to perform the algebraic transformations in generalized form:

$$\mathrm{SOLVE}\left(\frac{d}{dt}\,\mathrm{VQ}(a,\ t) = [0,\ 0],\ a\right) = \left[a = \frac{(F'(t)^2 + 1)^{3/2}}{F''(t)}\right]$$

$$B(t) := \frac{(F'(t)^2 + 1)^{3/2}}{F''(t)}$$

$$F(t) := \frac{t^2}{2}$$

$$\mathrm{VQ}(B(t),\ t) = \left[-t^3,\ \frac{3\cdot t^2}{2} + 1\right]$$

Among the DNL#31-files you will find a Demo-file including all the approaches shown above using some other functions. It should inspire for own investigations. Let me close with a question:

I know:
The parallel curve of a line is a line,

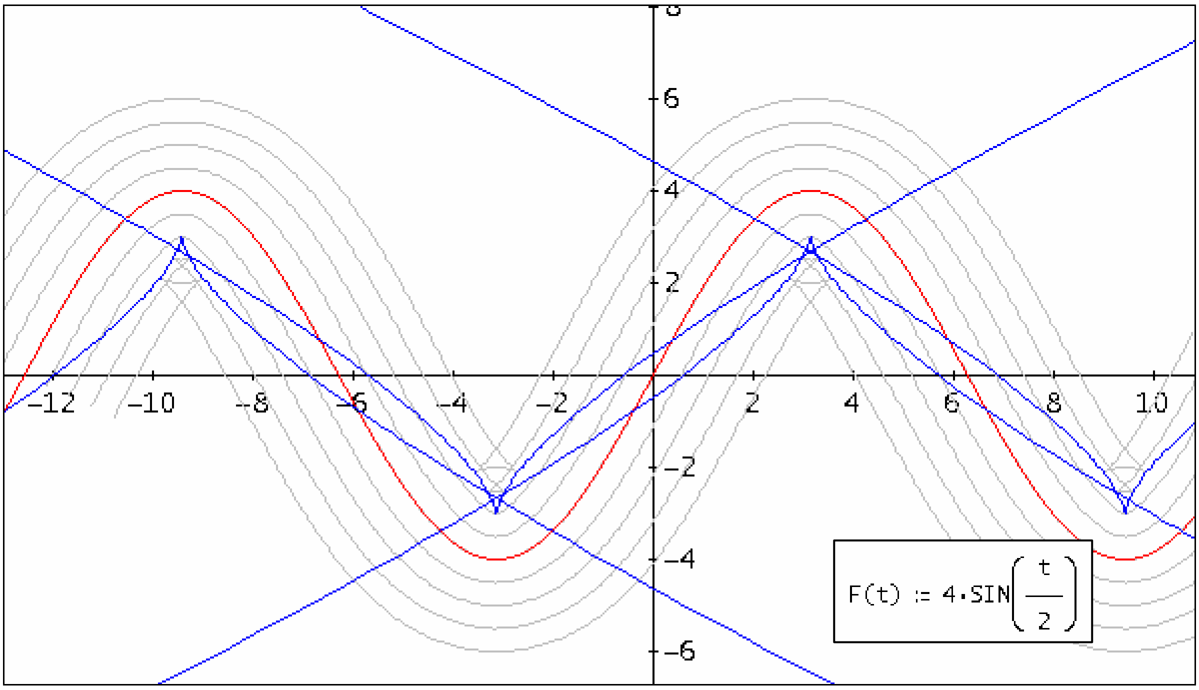
The parallel curve of a circle is a (concentric) circle.

I would like to know:
Which type of curves allows an equivalent rule?

Helmut Wunderling

You can experiment with other types of curves.

See below a trig function followed by an exponential function – which was the starting popint of the discussion.



F(t) := 0.5·EXP(0.3·t)

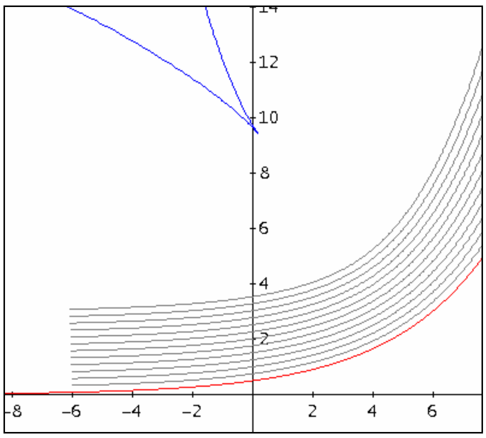
VECTOR(VQ(a, t), a, 0.25, 3, 0.25)

$$\text{SOLVE}\left(\frac{d}{dt}\,\text{VQ}(a, t) = [0, 0], a\right)$$

$$a = \frac{e^{-3\cdot t/10}\cdot(9\cdot e^{3\cdot t/5} + 400)^{3/2}}{360}$$

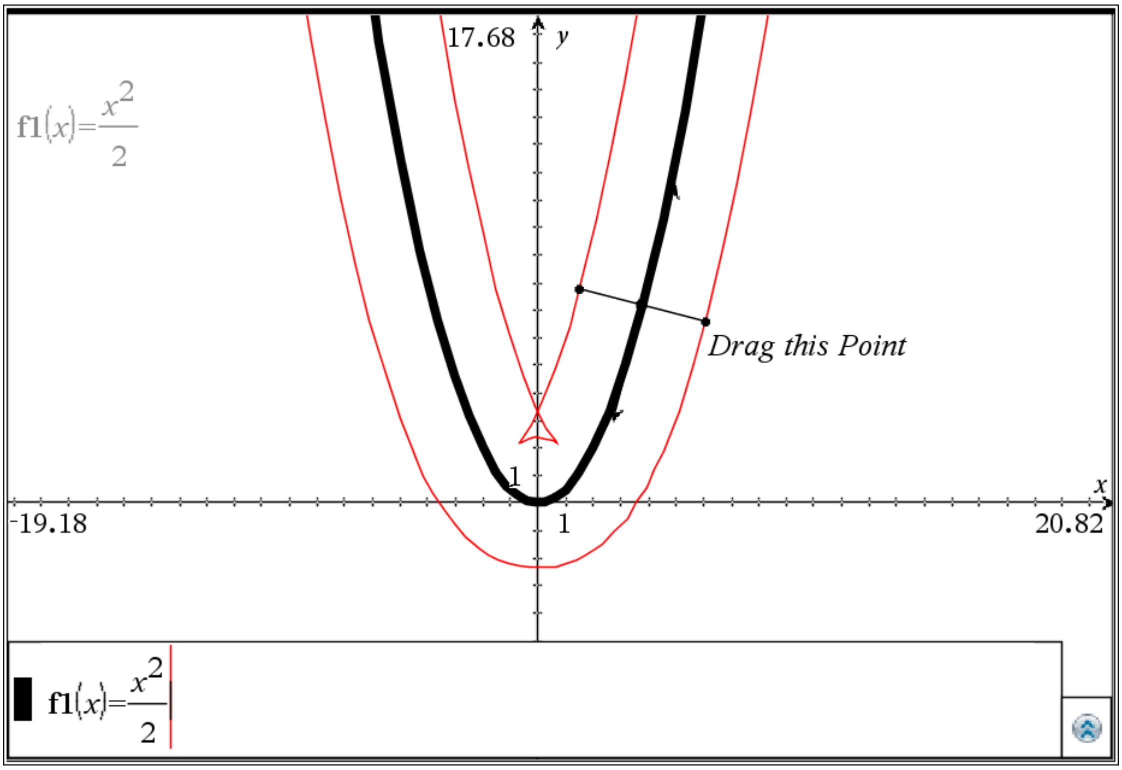
$$\text{VQ}\left(\frac{e^{-3\cdot t/10}\cdot(9\cdot e^{3\cdot t/5} + 400)^{3/2}}{360}, t\right)$$

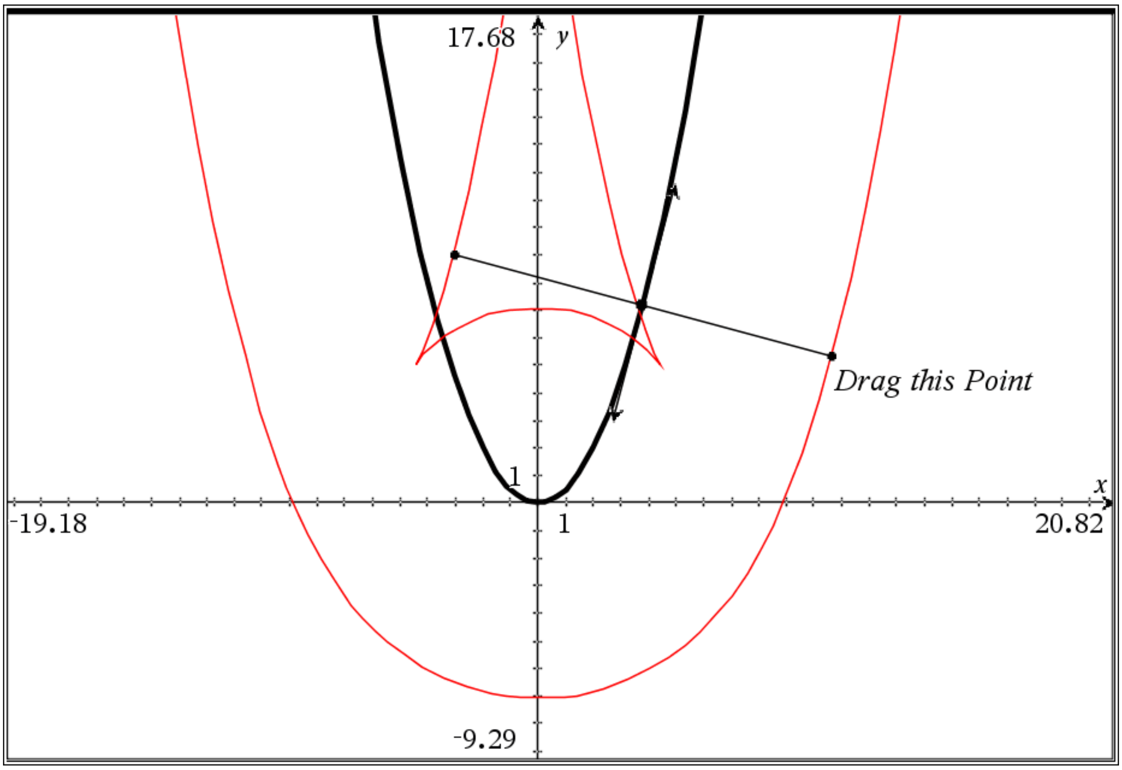
$$\left[-\frac{3\cdot e^{3\cdot t/5}}{40} + t - \frac{10}{3},\ e^{3\cdot t/10} + \frac{200\cdot e^{-3\cdot t/10}}{9}\right]$$



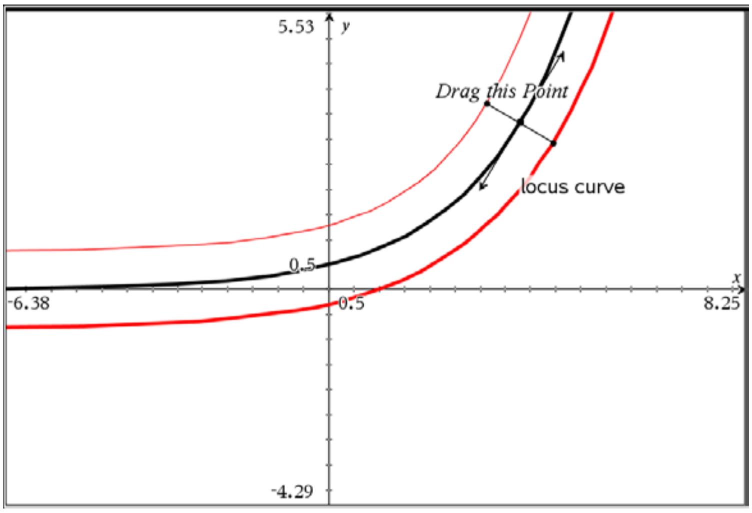
The next two pages show the 2013 approach plotting the parallel curves in the Graph Application of TI-NspireCAS. The curves are produced as loci of a point and then as enveloppe of a line parallel to the tangent of the given function graph.

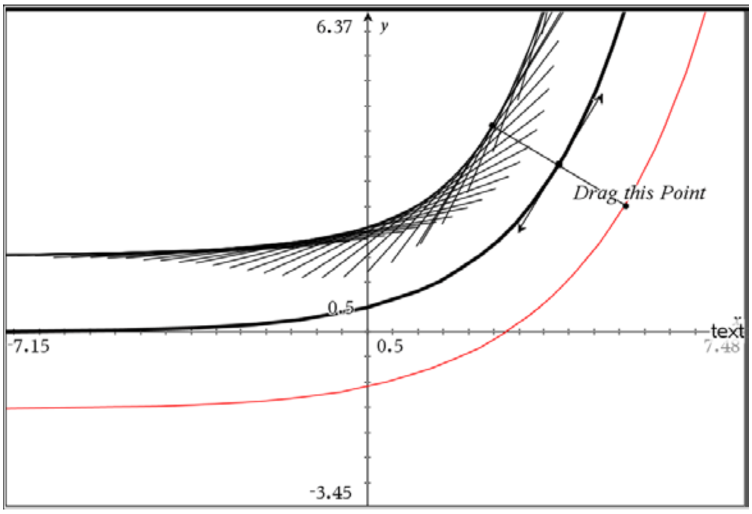
The parabola together with the loci of the moveable point.



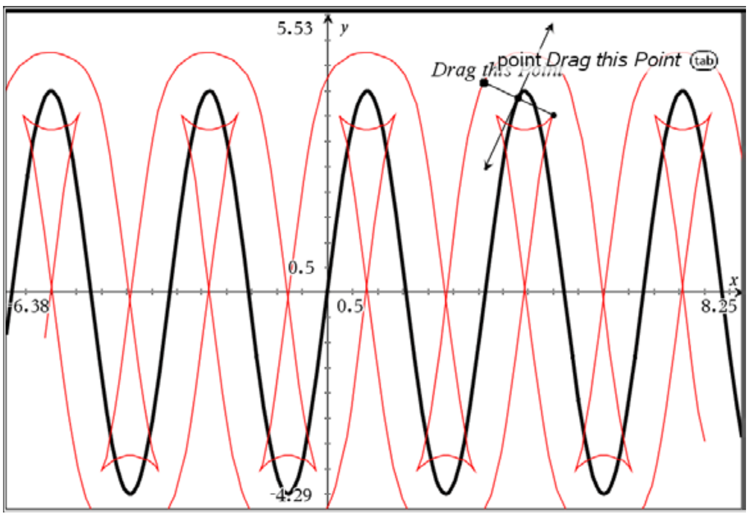


Next page gives two screen shots of the exponential “mother curve” and its parallel curves produced as loci of a point and as envelope of the parallel to the tangent.





The trig function together with its parallel curves produced with TI-NspireCAS.



The first contributions of the User Forum starting next page are a 1998 discussion about SORT-routines. Now these discussions are obsolete because an excellent working SORT procedure is built in. Nevertheless the discussion might still be of interest. Josef

Reinhard Schaeck, Berlin, Germany schaeck@ibm.net

I have a serious (though trivial) problem with Derive 4.09 for Windows. I want to sort a vector. I hope you can reproduce the mysterious result with the following steps:
1) We load the utility-file NUMBER.MTH (which provides the SORT-function). Test it with

```
SORT([3,2,5,1,7]) = [1,2,3,5,7]
```

which is o.k.

2) We enter:

```
a:=[374,358,341,355,342,334,353,346,355,344,349,330,352,328,336,359,
    361,345,324,386,335,371,358,328,353,352,366,354,378,324]
```

3) We test the dimension:

```
DIMENSION(a) = 30
```

which is o.k.

4) Now we enter SORT(a)

5) and simplify:

```
[648,656,330,334,335,336,341,342,344,345,346,349,704,706,354,710,
 716,359,361,366,371,374,378,386]
```

which is NOT the sorted vector and has only 24 elements.

Any idea? Some other vectors with 30 elements work. Sorting should be so simple...

Jim FitzSimons cherry@neta.com

That sort utility does not sort duplicate values. 358, 355, 353, 352, 324, 328 are duplicates. The sort adds the duplicates 2*324=648, 2*328=656, 2*352=704, 2*353=706, 2*355=710, 2*358=716.

DERIVE needs a better sort.

Johann Wiesenbauer, Vienna, Austria J.Wiesenbauer@tuwien.ac.at

Hi Reinhard,
You are just ... wait a minute ... the 1274th DERIVE-user who was trapped into thinking that the sort-routine in NUMBER.MTH is an all-purpose sort-routine. I assure you (straight from the horse's mouth, as it were, since I wrote it) that it isn't one and it was never intended to be one.
The truth is that it works for distinct nonzero rational numbers only, and in fact, this will cover more than 80% of all cases in practice, in particular when it comes to number theory. (Yes, it is true, they should have pointed out that restriction in the online help!)

Rumours are that there will be a built-in SORT-routine in the next release of DERIVE. In the meantime, this is what I can offer you:

(Rumours were right at this time. In the meanwhile a built-in SORT-routine is available in the next versions.)

```
SORT0(v) := LIM(TERMS(v╖VECTOR(x_^(-k_), k_, v)), x_, 1)
```

(Sorts any list of distinct nonzero rational numbers.)

```
SORT1(v) := VECTOR(vSUBk_, k_, LIM(DIF(TERMS(EXPAND(VECTOR(x_^(-v_),
v_,
v)·VECTOR(y_^k_, k_, 1, DIMENSION(v)))), y_), [x_, y_], [1, 1]))
```

(Sorts any list of nonzero rational numbers.)

```
SORT2(v):=APPEND(SORT1(SELECT(v_<=0,v_,v)),SORT1(SELECT(v_>0,v_,v)))
```

(Sorts any list of rational numbers.)

```
SORT3(v) := VECTOR(vSUBk_, k_, LIM(DIF(TERMS(EXPAND(VECTOR(
            x_^IF(NUMBER(v_), -v_, - APPROX(v_)), v_, v)·
            VECTOR(y_^k_, k_, 1, DIMENSION(v)))), y_), [x_, y_],
            [1, 1]))
```

(Sorts any list of nonzero real numbers.)

```
SORT4(v) := APPEND(SORT3(SELECT(v_ < 0, v_, v)), SELECT(v_ = 0, v_,
            v), SORT3(SELECT(v_ > 0, v_, v)))
```

(Sorts any list of real numbers.)

SORT3(v) and SORT4(v) are probabilistic only, i.e. there is a very small likelihood of error. (It must be said though that winning three times in a row in a lottery is much more likely than a failure due to this built-in possibility of error!) There are also deterministic sort-routines available (including a very fast one by David Stegenga based on quick-sort) but you have to pay a rather high price in terms of speed for that extra security.

If you want to be on the safe side, you could also check the result of SORT3 or SORT4, respectively, by means of

```
SORTED(v) := IF(PRODUCT(LIM(DELETE_ELEMENT(v, DIMENSION(v)) <=
DELETE_ELEMENT(v), [true, false], [1, 0])) = 1, true, false)
```

Of course, this isn't necessary in your case, where SORT1(a) will already do the trick. Sorting should be so simple? Well, it depends...

Cheers, Johann

<u>Reinhard Schaeck, Berlin, Germany</u> **schaeck@ibm.net**

Hello and thank you for the quick answer. After I examined it a little bit, I found your solution (SORT1) rather tricky. exploiting the fact that *DERIVE* obviously sorts some results internally (when using EXPAND and/or TERMS-fuction).
So it is not really a sorting algorithm, but does the job perfectly. Just in the case I have to work with "Precision = Approximate" it fails, because no internal sorting is necessary). I admit my current background has nothing to do with number theory (where approxinate numbers are rarely used) but with statistics, where I need sorted vectors AND approximate numbers. So let´s wait for a true sorting algorithm in *DERIVE x.xx*.

```
SORT(-1.21, 3.14, -0.5, ê, -2.4, -1.21, 3.14, 10, 3.14)
[-2.4, -1.21, -1.21, -0.5, 2.71828, 3.14, 3.14, 3.14, 10]
```

<u>David Stenenga, Honolulu, Hawaii</u> **dave@math.hawaii.edu**

I wrote a sort function based on the quick sort algorithm so that it´s computation time is approximately N*LOG(N), where N is the size of the list. It takes a little longer than Johann´s version which is based on *DERIVE*´s internal sort routine but it is nevertheless very fast.
In addition I will sort a list of vectors in lexicographic order (actually any order can be used by modifying the PRED-function).

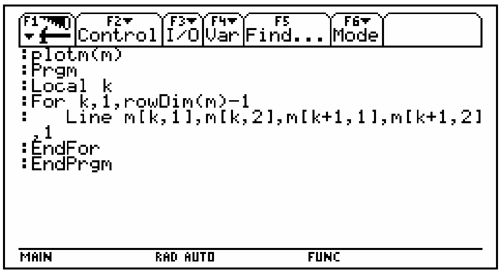
Examples:

```
SORT([3, #e, pi, -1, 0, 0, 2)]) = [-1, 0, 0, 2, #e, 3, pi]
QSORT([[1,4],[2,3],[0,5],[1,2]] = [[0,5],[1,2],[1,4],[2,3]]
```

Aloha, David

Sebastiano Cappuccio, Forli scappucc@spfo.unibo.it

Connected Mode with the TI-92

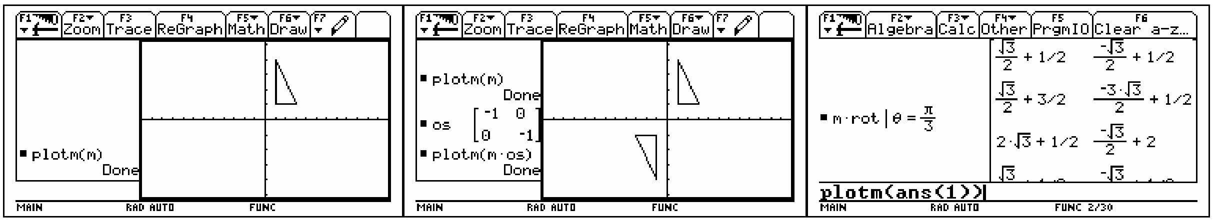


With *DERIVE* we can plot a $n \times 2$ matrix to get the graphical representation of a set of n points.
A very useful feature is available for that: the Connected Mode, which allows to plot a polygon by plotting the matrix of its vertices.

DERIVE's brother, the TI-92 calculator, has not such a feature, but with its powerful programming language, we can create it easily.

Of course, if you put a matrix 3×2, e.g. [[1,1],[3,1],[1,4]], as argument of the program PLOTM, you don't get a triangle; in order to have a triangle you have to "close" the polygon by putting a 4th row in the matrix m with the coordinates of the the first point again: [1,1], or you can modify the program.

Now we can define some new matrix and use matrix multiplication to get, for example, a symmetric triangle with respect to the origin or to get a 60° clockwise rotated triangle:



Günter Scheu, Pfinztal, Germany

Dear Josef,
The disussion on page 46 (revised DNL#30) has surprised me very much because "apples are compared with pears". How is probability defined there? There is no Laplace experiment and the number of outcomes is not finite.

You know:
(1) All primes - with exception 2 and 3 - show the form 6n-1 or 6n+1 (proof is elementar). (Numbers of the form (6n-1, 6n+1) are called prime twins, e.g. n=1, 2.) It is not known if there is an infinite number of prime twins.
(2) Asymptotic estimation of the prime numbers´ distribution.

Jan Vermeylen´s question was if the relative frequency tends towards 0.5. Johann Wiesenbauer proved this. T. Etchells´ remark makes no sense because it refers to a limit. The axioms of probability theory (Kolmogoroff) say that

$$p_1 + p_2 + \ldots\ldots\ldots + p_n = 1 \text{ which leads in our case to } p(\{2,3\}) + p(Z_{6n-1}) + p(Z_{6n+1}) = 1.$$

If $p(Z_{6n-1}) = 0.5$ then the number of primes of the form 6n-1 would exceed the others of the form 6n+1. Such results cannot be proved, because there is no formula to construct primes. (and following Goldbach´s Conjecture there will never be one!).

Summary: Be careful to change from finite sets to infinite sets. CAS is at most able to handle finite sets. CAS can lead us to conjectures but unfortunately cannot prove them. Best regards to you all, Günter

DNL: *I don´t think that Terence´s intention was to "prove" the conjecture. In Günter´s sense Terence´s DERIVE code is affirmating a conjecture. See his comment: "Pure maths wins the day!"*

Peter Witthinrich, Luebeck, Germany **PWitthinrich@t-online.de**

A comment to "6174 is a Special Number" from *DNL*#30

According [1], a collection of witty brain-twisters, a certain D R Kaprekar from Delavi (India) should have discovered 6174 thirty years ago. In [1] you can find the constants from 2 through 7 digits as follows:

k2	=	81
k3	=	495
k4	=	6174
k5	=	75933
k6	=	642654
k7	=	7509843

I have used those "Kaprekar´s Numbers" several times in teaching Information Technology, because it performs as a nice not too large programming problem for Pascal. The natural numbers are represented as strings or as arrays and the basic calculations are simulated by appropriate algorithms - like in elementary school. In *DERIVE* it runs much quicker, a fine thing.

[1] Otto Botsch, In der Werkstatt der Hirnverzwirner, Aulis-Verlag, Köln 1979, Problem Nr. 79 (page 115)

Steven Schonefeld, Auburn, IN, USA **schonefelds@alpha.tristate.edu**

Hi Josef,

It was nice seeing you in Gettysburg. I enjoyed your talk on *Dimensional Analysis*. I learned about this subject in college chemistry class.

I was impressed by Terence Etchells´ Gettysburg talk. One of the things he talked about was the implementation of termination tests using *DERIVE*. It is about time for the procedures in my book [1] to be re-written with built-in termination tests. Terence´s talk inspired me to do so. Te results of my effort are on the enclosed disk.

Let me say a few words about these procedures. A typical iterative method starts with an ***initial guess***, x_0, and creates a sequence of ***iterates***, $\{x_0, x_1, x_2, \ldots\}$, which are computed from the initial guess by a function, $G(x)$: $x_1 = G(x_0)$, $x_2 = G(x_1)$, etc.

My feeling is that the calculation of new iterates should be stopped by a two-part ***termination test***.

Part one stops the iteration process if two successive x´s are close. That is, the iteration stops with x_n if

$$ABS(x_n - x_{n-1}) < \delta\_tol$$

- where δ_tol is a small, positive number.

Part two stops the iteration process after a given number of iterations have been performed. That is, the iteration stops with x_n if

$$n > n\_max$$

- where *n_max* is a large, positive number.

Using ITERATES(NEW_V(v),v,v$_0$):

With this method, *v* is a vector containing an iterate x_n. As far as I could tell, an iteration counter, *n*, must also be included in the vector *v*. For example, in my implementation of the Newton-Raphson method (2_3_NR.MTH), the function $G(x)$ is

$$G(x) := x - \frac{F(x)}{F'(x)}$$

and v is of the form $[x_n, \delta_n, F(x_n), n]$

where $x_n = x_{n-1} + \delta_n$ and $\delta_n = -F(x_{n-1}) / F'(x_{n-1})$. The good news is that the termination test is easy to implement - since $x_n - x_{n-1} = \delta_n$ and *n* are contained in the current vector. The bad news is that when the termination test is satisfied, the last vector *v* is repeated in the resulting matrix *m*. In order to avoid confusion, I had to **T**ake **O**ff the **L**ast row of *m*. Hence TOL( ) is the name of the function which does this task.

Using ITERATE(NEW_M(m),m,m$_0$):

With this method, *m* is a matrix. Each row of matrix *m* contains an iterate x_n, (usually plus more information.) For example, in my implementation of repeated substitution (2_2_REP.MTH), a typical row of the matrix, *m*, is of the form

$$[x_n, n] \qquad \text{with } x_n = G(x_{n-1}).$$

The good news is that we may determine the iteration counter, *n*, from DIMENSION(*m*). (I included *n* in the typical row of *m* for pedagogical reasons.) The bad news is that the initial matrix, m_0, must contain at least two iterates for the first part of the termination test

$$ABS(m_{n,1} - m_{n-1,1}) < \delta\_tol$$

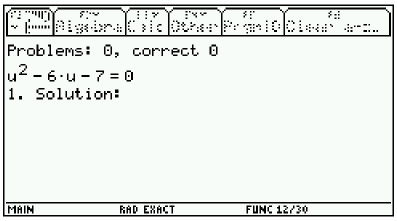
to make sense. Also when using ITERATE(, ,), the function, NEW_M(*m*), may be quite complicated.

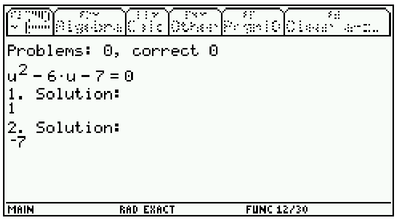
[1] Numerical Analysis via DERIVE, Steven Schonefeld, 1994, MathWare

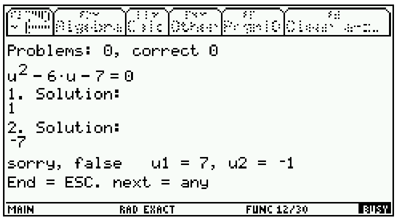
Steven offers an update of his DERIVE-files containing self terminating procedures for his book to book owners. If you are interested in the files then please contact Steven or me. Thanks a lot for that offer. Josef

In revised DNL#20, p 54 you could find "Vieta by Chance", a *DERIVE* Utility to train solving quadratics with integer solutions mentally. In my opinion some skills should survive even in times of *DERIVE* and the *TI*-92, *Voyage 200* - and *TI-Nspire* as well. So I wrote a little *TI*-program and believe it or not, the pupils liked to compete their skill in solving as many quadratics as possible. Josef

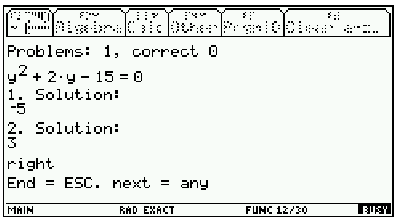
```
vieta()
Prgm
Local ˆb,x1,x2,k1,k2,v1,v2,bz,rz,key
0»bz:0»rz
Lbl start
ClrIO
Disp "Problems: "&string(bz)&", correct "&string(rz)
char(rand(26)+96)»ˆb
Lbl test
rand(24)-12»x1:rand(34)-17»x2
If x1*x2=0 or x1=ªx2
Goto test
ªx1-x2»k1:x1*x2»k2
If k1<0 Then
   " - "»v1
   Else
   " + "»v1
EndIf
If k2<0 Then
   " - "»v2
   Else
   " + "»v2
EndIf
If k1=1 Then
    ""»k1
   Else
    string(abs(k1))»k1
EndIf
string(abs(k2))»k2
Disp  expr(ˆb&"^2"&v1&k1&ˆb&v2&k2&"  =
0")
Lbl chk1
Input "1. Solution:",l1
Input "2. Solution:",l2
If l1+l2=x1+x2 and l1*l2=x1*x2 Then
   Disp "right"
   rz+1»rz
Else
  Disp "sorry, false   "&ˆb&"1 = "&string(x1)&", "&ˆb&"2 = "&string(x2)
EndIf
```

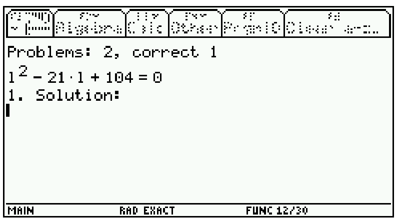






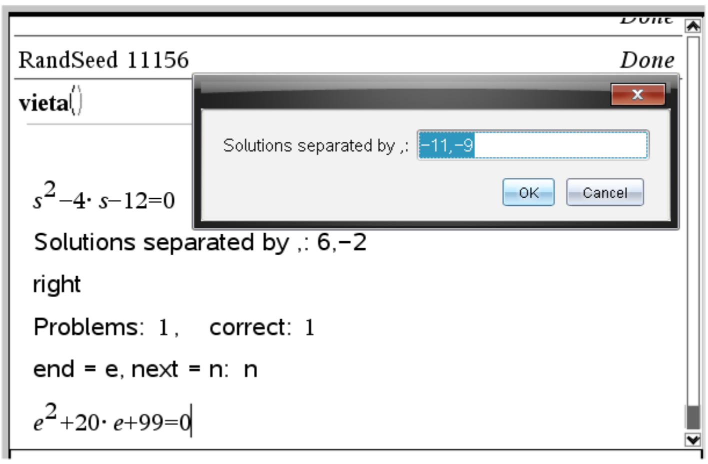

```
bz+1»bz
Disp "End = ESC. next = any"
Loop
getKey()»key
If key=264 Then
   Goto ende
ElseIf key 0 Then
   Goto start
EndIf
EndLoop
Lbl ende
EndPrgm
```





It is not too difficult to transfer the program to TI-NspireCAS considering the changed form of input and output:

```
Define vieta()=
Prgm
:Local θb,xx1,xx2,kk1,kk2,vv1,vv2,ss,bz,rz,key,equ
:bz:=0:rz:=0
:Lbl start
:ss:=""
:θ  b:=char(randInt(97,122))
:Lbl vals
:xx1:=randInt(1,24)−12:xx2:=randInt(1,34)−17
:If xx1*xx2=0 or xx1+xx2=0: Goto vals
:kk1:=⁻  xx1−xx2:kk2:=xx1*xx2
:vv1:=when(kk1<0," − "," + ")
:vv2:=when(kk2<0," − "," + ")
:kk1:=when(kk1=1,"",abs(kk1))
:kk2:=abs(kk2)
:equ:= θb&"^2"&vv1&string(kk1)&θ  b&vv2&string(kk2)&" = 0"
:Disp expr(equ)
:RequestStr "Solutions separated by ,:",ss
:ss:=expr("{"&ss&"}")
:If ss[1]+ss[2]=xx1+xx2 and ss[1]*ss[2]=xx1*xx2 Then
:Disp "right"
:rz:=rz+1
:Else
:Disp "Sorry, wrong:  "&θb&"1 = "&string(xx1)&", "&θb&"2 = "&string(xx2)
:EndIf
:bz:=bz+1
:Disp "Problems: ",bz,",    correct: ",rz
:RequestStr "end = e, next = n: ",key
:If ord(key)=110:Goto start
:EndPrgm
```



The left column shows a VIETA-session performed with the software.

The right column is a collection of handheld screen shots.

vieta()

$s^2-4\cdot s-12=0$

Solutions separated by ,: 6,−2

right

Problems: 1 , correct: 1

end = e, next = n: n

$e^2+20\cdot e+99=0$

Solutions separated by ,: −11,−9

right

Problems: 2 , correct: 2

end = e, next = n: n

$l^2-12\cdot l+32=0$

Solutions separated by ,: 8,4

right

Problems: 3 , correct: 3

end = e, next = n: n

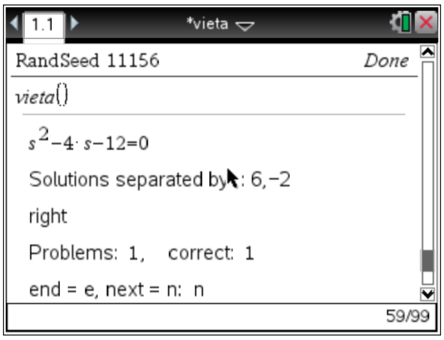
$k^2-17\cdot k+30=0$

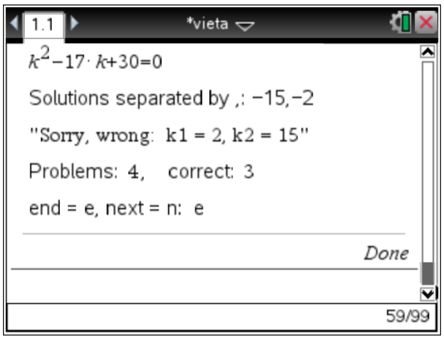
Solutions separated by ,: −15,−2

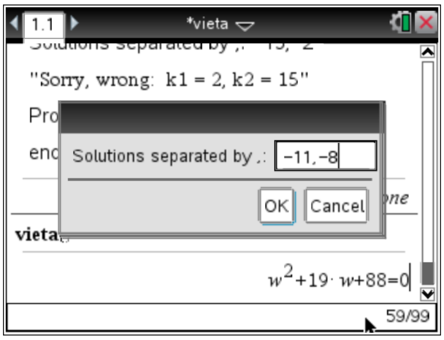
"Sorry, wrong: k1 = 2, k2 = 15 "

Problems: 4 , correct: 3

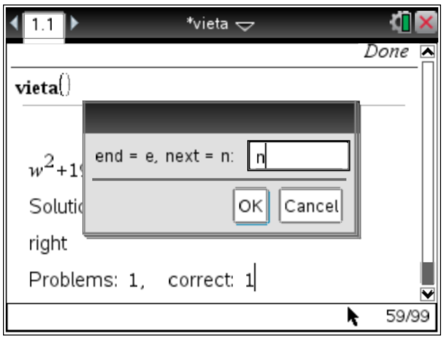
end = e, next = n: e

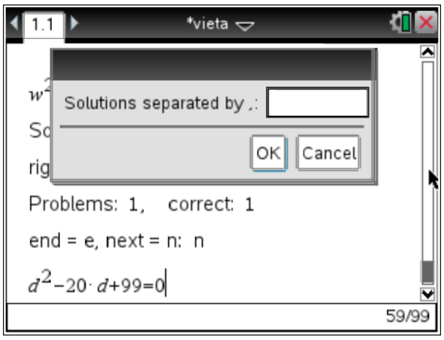






Another run:





Karl-Heinz Keunecke, Kiel, Germany
kh@keukiel.netzservice.de

Acceleration due to Gravity ≈8.14 m sec^{-2} ?

1 Introduction

Data collection by means of computers is well known and even in schools a used technology. The CBL[1] and the CBR[2] are used in connection with graphic or symbolic calculators. These handheld systems are easier to handle than PC-based systems. But the most important advantage lies in the fact that the computer algebra of the *TI*-92 or the *TI*-89 can be applied to the data. That gives the teacher the chance to process the data in such a way that matches the curriculum, the capability or the interests of his students, his own ideas, etc.

As an example in this paper the gravity acceleration is determined from the observation of a jumping ball with an ultrasonic range meter. The unusual result is explained as an effect of the buoancy of the ball in the air.

2 Observation of a jumping ball with the CBR

For our experiments we use a big gymnastics ball with a radius of 34 cm. It jumps regularly with a small damping rate and it reflects the sound signal always clearly back to the CBR. The ultrasonic ranger was mounted at a height of 2.2 m. The ball is held at a distance of 50 cm from the CBR. As soon as the measurement starts it is released and falls down to the ground.

The distance from ground is shown in figure 1.

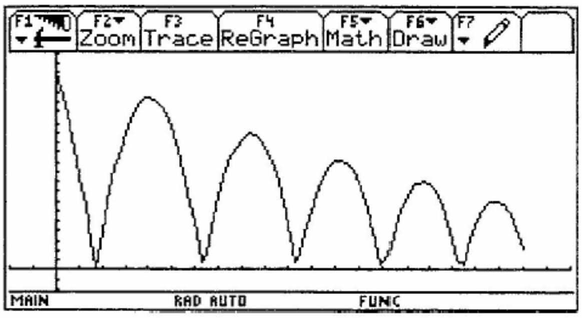


Figure 1: *Height of a jumping ball*

5 rebounds at the ground can be recognized.

In order to inspect the physical process more detailed, a section of figure 1 is considered in figure 2.

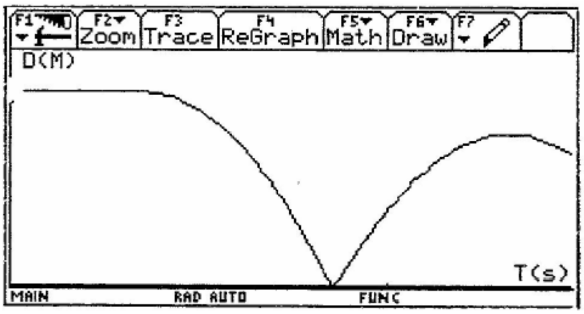


Figure 2: *Balls height as a function of time*

The calculated velocity and acceleration are displayed in figures 3 and 4.

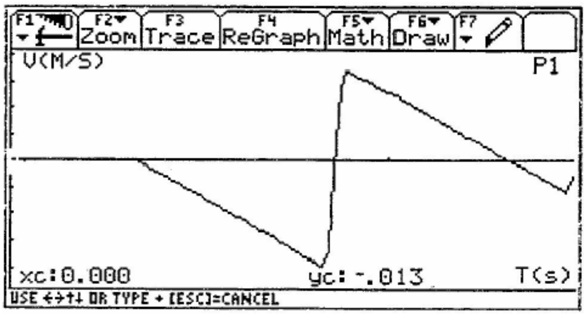


Figure 3: *Velocity of the jumping ball*

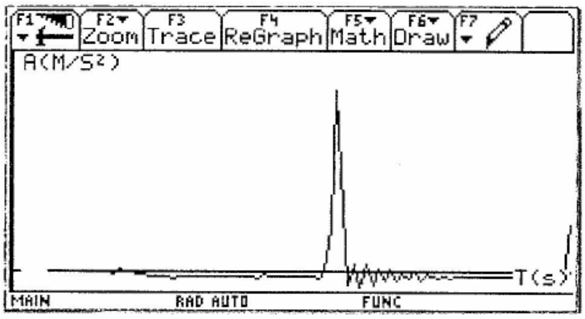


Figure 4: *Acceleration of the jumping ball*

From figure 3 it can be seen that the absolute value of the velocity increases linearly to the ground. There the ball is stopped within a short period and is accelerated then in opposite direction. The ball moves up again with linear velocity until it arrives at maximum height where the velocity is zero. From there the described ball drop begins a second time. The acceleration in figure 4 increases to about 20g during the ball is rebounded from the ground. It can also be seen that the rebounce produces oscillations of the ball for about half a second.

3 Data processing with the TI-92

After the observations have been discussed by the students, they dealt with the question: *What functions describe the height and the velocity of the ball?*

These questions can easily be solved by using the tools of statistical calculations of the TI-92. For that purpose the lists l1 to l4 are read into a table by means of the Data/Matrix Editor. Then a regression function can be calculated. Choosing a quadratic regression the calculated function is shown in figure 5. The coefficients *b* and *c* of the quadratic term are negligible and therefore the describing function is approximately:

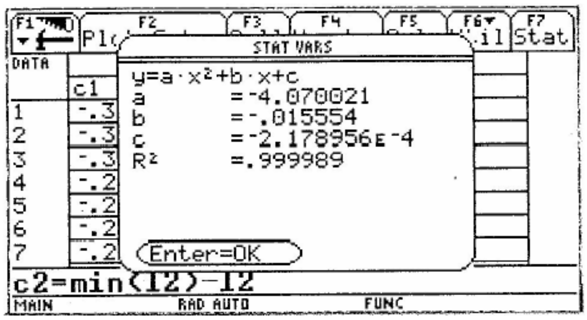


Figure 5: *Quadratic regression function*

$$y \cong -4.07x^2 . \qquad (1)$$

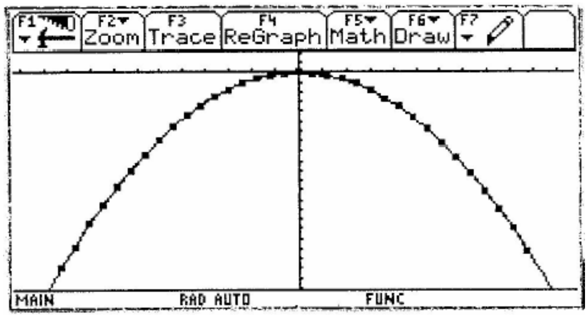


Figure 6: *Distance-time diagram (squares) and quadratic regression function*

Using the same procedure to find a linear regression for the velocity the equation

$$v \cong 8.14\, t . \qquad (2)$$

is yielded. The data and the regression function are plotted in figures 6 and 7.

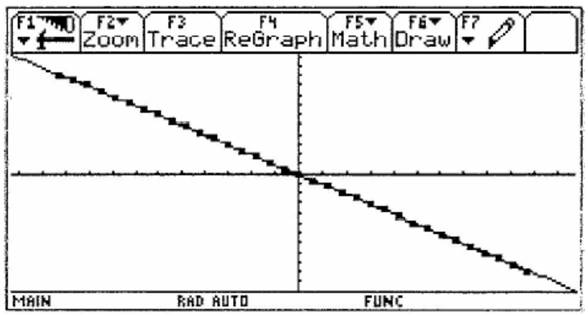


Figure 7: *Velocity-time diagram (squares) and the regression line*

The regression function for the distance and for the velocity as well yields an acceleration of 8.14 m sec^{-2}.

Now the students are faced to the question: *Why is the gravity acceleration clearly smaller than 9.81 m sec^{-2}?*

Friction cannot be the reason. The influence of friction increases when the velocity goes up. Therefore it would reduce the acceleration at higher speed. This cannot be seen from the data for the velocity increases linearly through the period of observation.

But the buoyant force of the ball in the air reduces the gravity force to a certain amount which can be determined by using the principle of Archimedes.

The reduced gravity force of the ball in air G' equals the difference of the gravity force in vacuum G and the buoyancy of the ball A:

$$G' = G - A \tag{3}$$

The ball is buoyed up with a force A equal to the gravity force of the displaced air

$$A = m_{dis}\, g \tag{4}$$

We can write $G = m\, g$ and $G' = m\, g'$ where m is the mass of the ball, which is the sum of the mass m_{env} of the envelope of the ball and the mass m_{air} of the air included in the ball:

$$m = m_{env} + m_{air} \tag{5}$$

Then equation (3) yields:

$$(m_{env} + m_{air})\, g' = m_{env}\, g + m_{air}\, g - m_{dis}\, g \tag{6}$$

The pressure in the ball is not much above the atmospheric pressure. Therefore it can be assumed that the mass of the air in the ball equals nearly the mass of the air which is displaced by the ball,

$$m_{air} \cong m_{dis}\,. \tag{7}$$

Now the reduced gravity acceleration g' can be determined from equation (6):

$$g' = g\,\frac{m_{env}}{m_{env} + m_{air}}\,. \tag{8}$$

The mass m_{air} can be calculated from the radius r of the ball and the density ρ_{air} of the air,

$$m_{air} \cong m_{dis} = \rho_{air}\, V = \rho_{air}\,\frac{4\pi}{3} r^3,$$

$$r = 34\text{cm},\ \rho_{air} = 1.29\,\frac{\text{g}}{\text{dm}^3} \Rightarrow m_{air} \cong 0.212\,\text{kg}$$

The mass of the ball determined by a pair of scales, equals under the made assumptions the mass of the envelope,

$$m_{env} = 1.158\ \text{kg}.$$

With these results the reduced gravity acceleration can be calculated from equation (8):

$$g' = 9.81\cdot\frac{\text{m}}{\text{sec}^2}\left(\frac{1.158\,\text{kg}}{1.158\,\text{kg} + 0.212\,\text{kg}}\right) =$$

$$= 8.29\,\frac{\text{m}}{\text{sec}^2}$$

The calculated acceleration does not agree totally with the observed acceleration. The reason probably is the assumption of equation (4). The mass of the air included in the ball is, because of the pressure in the ball, greater than the air being displaced by the ball.

---

[1] **C**alculator **B**ased **L**aboratory, Texas Instruments

[2] **C**alculator **B**ased **R**anger, Texas Instruments