THE DERIVE - NEWSLETTER #121

ISSN 1990-7079

THE BULLETIN OF THE DERIVE USER GROUP

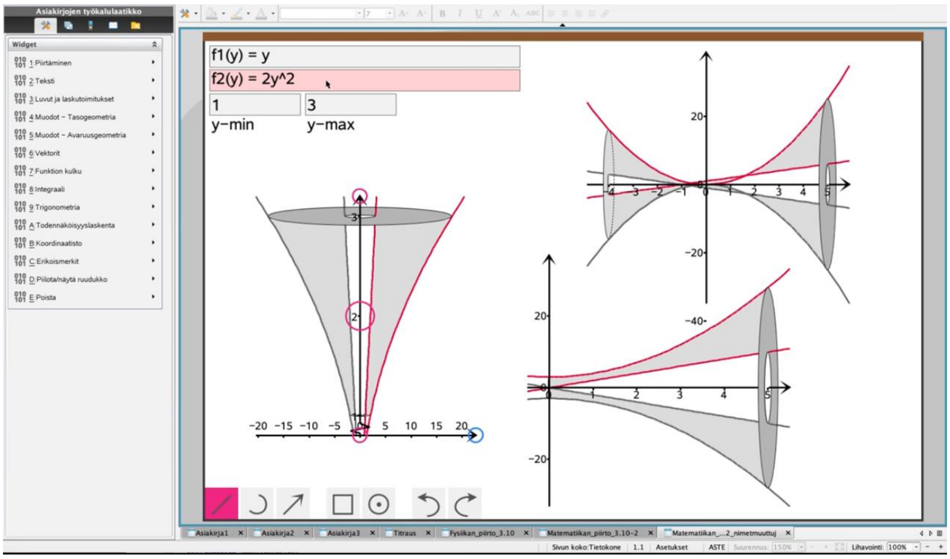
+ CAS-TI

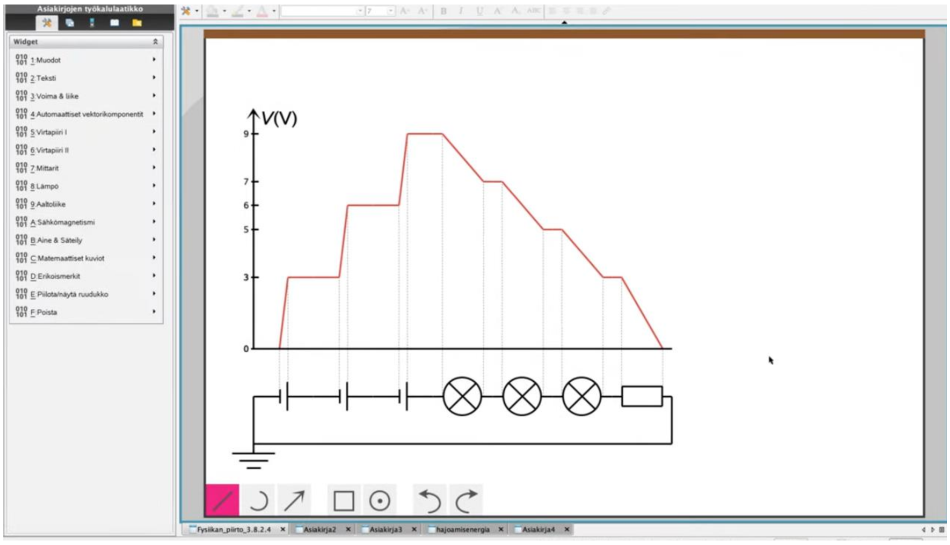
Contents:

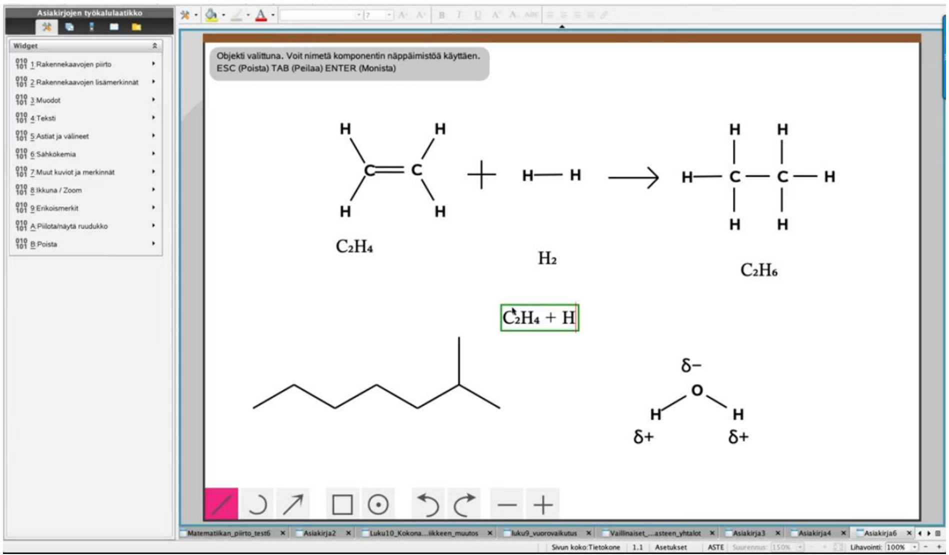
1 Letter of the Editor
2 Editorial - Preview
3 DERIVE & CAS-TI User Forum
5 Central End Examination at a German Vocational School
Last example & Solutions
Wolfgang Pröpper
11 Eine andere Sicht auf Lotteriespiele/Another View on …
Bhuvanesh Bhatt & Josef Böhm
15 Bivariate Diophantine Equations
Wolfgang Alvermann
26 The Easter Formula
Bhuvanesh Bhatt
31 Newton-Raphson & the Jacobian
Sebastian Rauh & Josef Böhm
35 Riffle Shuffle
40 Widgets for TI-Nspire - Localizations

March 2021

Olli Karkkulainen and Markku Parkkonen, two Finnish teachers developed “special applications for special needs”. They enable producing nice presentations on the Nspire screen for Mathematics, Physics and Chemistry. Have look at https://nspire.fi/widget/. See more on page 40.







About versions in other languages see page 40.

Dear DUG-Members,

many thanks for your wishes at the occasion of New Year and my last information. I enjoy very much when contacts remain upright.

I can offer DNL#121 with a short delay. The Diophantine equation contribution was equally exciting as it was time-consuming and work-intensive. Not all questions could be solved. Maybe that there are experts in our community?

Another function from Bhuvanesh Bhatt's TI-92 library was less difficult. Treating the Newton-Method using Jacobi matrices allow another view on this standard application of calculus in our schools.

It was in DNL#46 when Valeri Anisiu presented a DERIVE function for calculating the Easter date. Wolfgang Alvermann's mail fit exactly for Easter time to take up this topic again. Veit Berger provided an extra Python version. His experience enables to compare programming with TI-BASIC, DERIVE and TI-Python.

The intense communication with Wolfgang Alvermann, Sebastian Rauh, Veit Berger and Wolfgang Pröpper is the second reason for the delay of DNL#121. I cannot imagine how this had been possible in ages of snail mail – sending letters and printouts to and fro.

Wolfgang Pröpper updated his article on Lottery Games for this issue, many thanks for this.

I have to thank the other Wolfgang (Alvermann), Sebastian, Veit and Hubert Langlotz for their patience and willingness to answer my many further inquiries in detail.

Hubert sent an interesting topic which will be published in the next newsletter. He drew my attention to a website full of mathematical problems for all age groups of students.

Have a look at www.bolyaiteam.at or www.bolyaiteam.de (in German only). It's worth it. Please note the reference to the widgets!

With my best regards

Josef

Please see the important note on page 4 below.

Liebe DUG-Mitglieder,

herzlichen Dank für die vielen Grüße anlässlich des Jahreswechsels und meiner letzten Information. Es freut mich (uns) immer wieder, wenn die persönlichen Kontakte aufrecht bleiben.

Mit etwas Verspätung kann ich Euch den Newsletter#121 vorlegen. Der Beitrag zu den diophantischen Gleichungen war gleichermaßen spannend wie zeit- und arbeitsaufwändig. Nicht alle Fragen konnte ich lösen. Vielleicht finden sich in der DUG Fachleute für dieses Gebiet?

Eine weitere Funktion aus der Bibliothek von Bhuvanesh Bhatt war weniger schwierig. Die Behandlung des Newton-Verfahrens mittels Jacobi-Matrizen erlauben eine andere Sicht auf diese Standardanwendung der Differentialrechnung in der Schule.

Schon im DNL#46 – lang ist's her – gab uns Valeri Anisiu eine DERIVE-Datei zur Berechnung des Osterdatums. Wolfgang Alvermanns Schreiben passte heuer genau in die Osterzeit; so haben wir uns dieses Themas wieder angenommen, zu dem auch Veit Berger eine Python-Version beisteuerte. Veits Erfahrung lässt uns das Programmieren mit TI-BASIC, DERIVE und TI-Python vergleichen.

Die intensive Kommunikation mit Wolfgang Alvermann, Sebastian Rauh, Veit Berger und Wolfgang Pröpper ist der zweite Grund für die Verzögerung. Ich kann mir gar nicht mehr vorstellen, wie das zu Zeiten des alten Postweges – Briefe hin und her – möglich gewesen ist.

Wolfgang Pröpper hat seinen Lotteriebeitrag für den DNL überarbeitet, vielen Dank dafür. Dank auch an den anderen Wolfgang (Alvermann), Sebastian, Veit und Hubert Langlotz für ihre Geduld und Bereitschaft, meine Rückfragen immer wieder ausführlich zu beantworten.

Hubert hat mich auf eine schöne Webseite aufmerksam gemacht: www.bolyaiteam.at oder www.bolyaiteam.de. Schaut hinein, es lohnt sich. Beachtet bitte auch die Hinweise auf die Widgets!

Ich verbleibe mit besten Grüßen

Josef

Ein wichtiger Hinweis findet sich auf Seite 4 unten.

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

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

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

Next issue: June 2021

Preview: Contributions waiting to be published

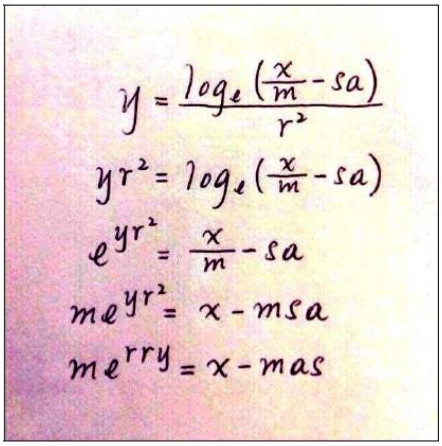
Some simulations of Random Experiments, J. Böhm, AUT, Lorenz Kopp, GER
Wonderful World of Pedal Curves, J. Böhm, AUT
Simulating a Graphing Calculator in *DERIVE*, J. Böhm, AUT
Cubics, Quartics – Interesting features, T. Koller & J. Böhm, AUT
Logos of Companies as an Inspiration for Math Teaching
Exciting Surfaces in the FAZ
BooleanPlots.mth, P. Schofield, UK
Old traditional examples for a CAS – What´s new? J. Böhm, AUT
Mandelbrot and Newton with *DERIVE*, Roman Hašek, CZ
Tutorials for the NSpireCAS, G. Herweyers, BEL
Dirac Algebra, Clifford Algebra, Vector-Matrix-Extension, D. R. Lunsford, USA
Another Approach to Taylor Series, D. Oertel, GER
Statistics of Shuffling Cards, H. Ludwig, GER
Charge in a Magnetic Field, H. Ludwig, GER
More Applications of TI-Innovator™ Hub and TI-Innovator™ Rover
Surfaces and their Duals, Cayley Symmetroid, J. Böhm, AUT
A Collection of Special Problems, W. Alvermann, GER
DERIVE Bugs? D. Welz, GER
Tweening & Morphing with TI-NspireCX-II-T, J. Böhm. AUT
The Gap between Poor and Rich, J. Böhm, AUT
More functions from M. Myers and from Bhuvanesh's Mathtools-library
Double-Die-Encrytion - Doppelwürfelverschlüsselung
QR-Code light, Problem from Bolyai-MTC 2021, Sparse Matrices
153 is another Special Number, and others

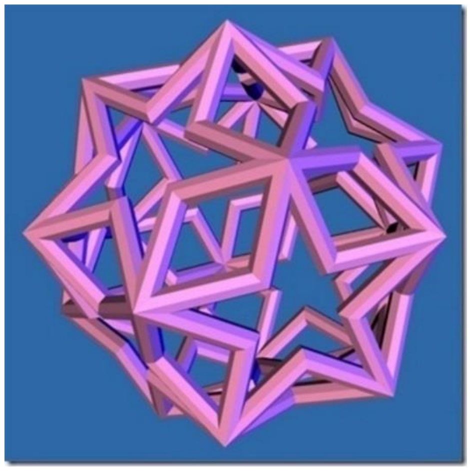
Impressum:
Medieninhaber: *DERIVE* User Group, A-3042 Würmla, D´Lust 1, AUSTRIA
Richtung: Fachzeitschrift
Herausgeber: Mag. Josef Böhm

Dave Halprin sent a very mathematical Christmas greeting.

The stars below were reactions on the Herrnhuter Star from DNL#120.

Many thanks





Unser DUG-Mitglied aus der Schweiz, Herbert Hunziker, machte mich aufmerksam, dass von ihm ein Buch erschienen ist. Er schrieb, dass „dabei natürlich DERIVE zum Einsatz gekommen ist“.

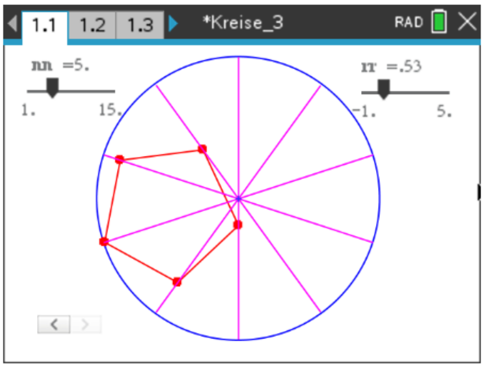
Herbert Hunziker
Bondifaktoren
Ein natürlicher Zugang zur speziellen Relativitätstheorie

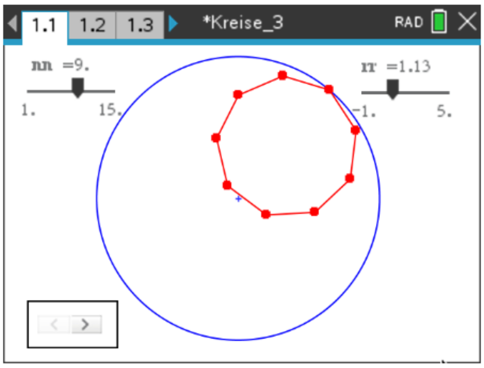
Springer Spektrum, 2021
ISBN 978-3-658-32297-7
Preis: 14.5 Euro (Softcover)/ 4.5 Euro (eBook)

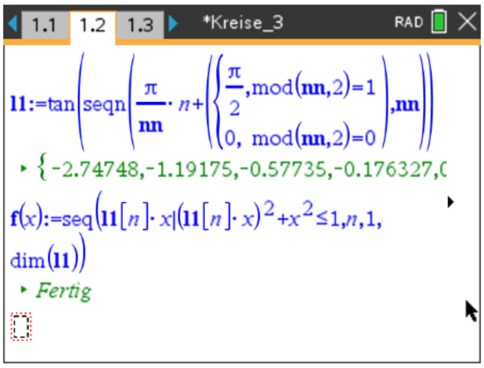
DUG Member Herbert Hunziker (Switzerland) is author of a book about Bondifactors, which is a “Natural Approach to Special Relativity”. Herbert Hunziker used DERIVE for writing his book.

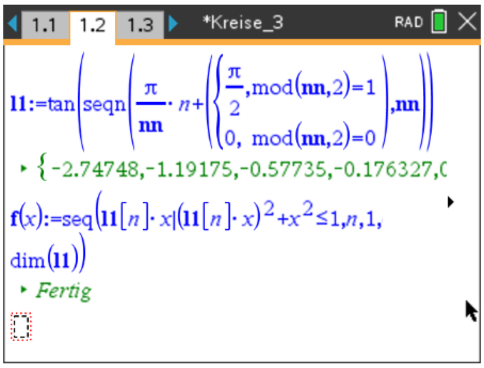
From Sebastian Rauh:

An experiment with polygons rolling in a circle. Sliders change the polygon and move it. The vertices are gliding on the diameters which can be turned on and off. The graph screens were presented in DNL120 but now I show the scenery behind Sebastian's nice animation:









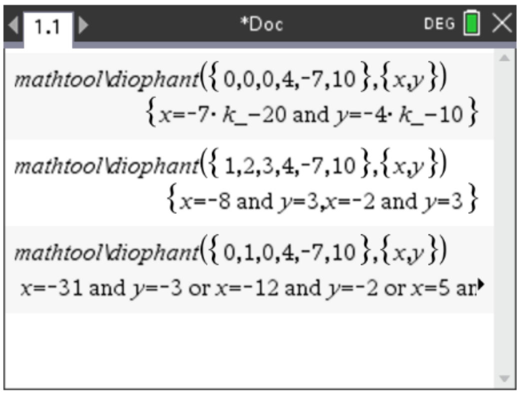
	A nnn	B aa	C xx0	D yy0	E xx	F yy
=	=seqn(	=seqn(π/nn*n,nn+1)	=sin(aa)	=cos(aa)	=xx0*sin(2*π*rr+aa)	=yy0*sin(2
1	1	0.349066	0.34202	0.939693	0.314363	
2	2	0.698132	0.642788	0.766044	0.641785	
3	3	1.0472	0.866025	0.5	0.829063	
4	4	1.39626	0.984808	0.173648	0.788566	
5	5	1.74533	0.984808	-0.1736...	0.539245	-
6	6	2.0944	0.866025	-0.5	0.197758	-
7	7	2.44346	0.642788	-0.7660...	-0.076109	
8	8	2.79253	0.34202	-0.9396...	-0.154209	
9	9	3.14159	1.E-13	-1.	-7.28969E-14	
10	10	3.49066	-0.34202	-0.9396...	0.314363	
11						
12						

H5

Important note for working with *diophant*() and *newtraph*() with TI-Nspire:

The library *mathtool* must be stored in your *mylib*-folder. The functions need some auxiliary functions which are also contained in this huge library.

See an example how to run the function:



Wolfgang Alvermann showed in DNL#118 his mathematics final exam from 1968. We compare this with a nowadays final exam. It is a central exam for secondary vocational school from Germany. After the first – compulsory – part, and the first blocks of the second – electoral – part I close with the last task and the solutions. Josef

Task 3B

Again and again, serious accidents occur when cyclists are overlooked in the blind spot of a truck. To avoid such accidents an inventor from Flensburg has developed a surveillance column. It scans the blind spot permanently by thermal sensors (fig 1 and fig 2). As soon as a person - a cyclist, a pedestrian or a skater - enters this zone, he/she stands out clearly from the surroundings due to his/her thermal image.

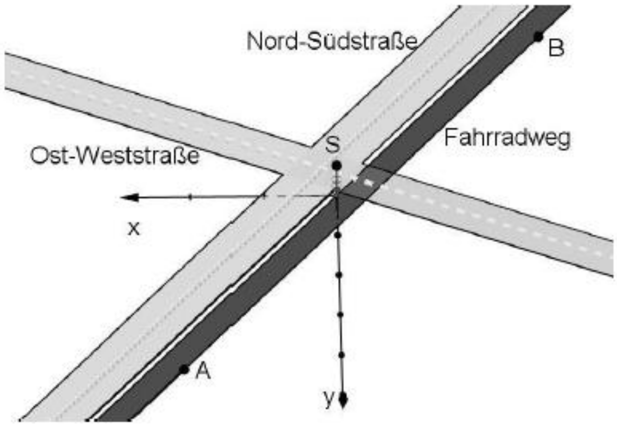


fig 1

Software of the surveillance column compares the newly captured images with the reference images and alerts the truck drivers with conspicuous flashing lights, as soon as it recognizes any change. A cycle track is running along the slope in the plane f: $x + y - 50z = 0$. Its right boundary passes points $A(21{,}54|45{,}30|1{,}34)$ and $B(-28{,}46|-41{,}30|-1{,}39)$. The left boundary of the track runs along the straight line g_L: $g_L:\ \vec{x} = r \cdot \begin{pmatrix} 50.00 \\ 86.60 \\ 2.73 \end{pmatrix}$.

The thermal sensor is fixed on a pole located in point $S(0|0|5)$. All measures are in m.

a) For exact adjustment of the thermal sensor the width of the cycle path is needed. Calculate this width.

The thermal sensors capture the track with an angle of inclination of 80° against the pole. (fig2).

In order to check the installation, the responsible authority intends to mark the beginning of the range of the sensor.

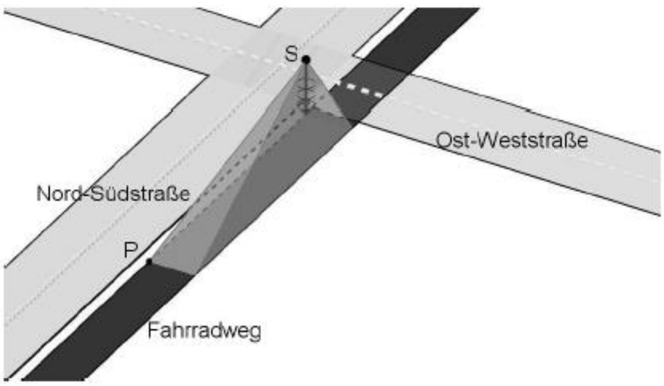


fig 2

Calculate the position of point P on the left boundary of the track where the mark is to be applied.

b) A large bird flies at a speed of 20 km/h on a straight flight path with direction described by vector $\begin{pmatrix} 10 \\ -10 \\ 1 \end{pmatrix}$ (fig.3).

To avoid false alarm due to bird flight the surveillance column reacts only when a living being is detected for longer than one second.

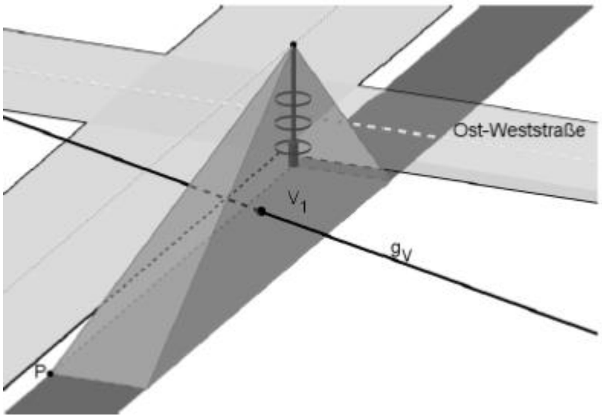


fig 3

The bird enters the monitoring area in point $V_1(2.53|9.47|2.05)$ and leaves it soon after. The plane limiting the – in flight direction – rear area can be described by h: $-1.73x + y = 0$.

Calculate the time span during which the bird is in the monitoring area and check, if the surveillance column reacts by turning on the flash light.

Solutions

(P1 – P5, 1A, 1B from DNL#119)

P1 a) $1-\frac{1}{x^2}=-3 \rightarrow x_{1,2}=\pm\frac{1}{2}$; b) $A = 4$

P2 a) $f'(x)=e^{g(x)}>0\ \forall x\in R \rightarrow$ no turning point

b) $f''(x)=g'(x)\cdot e^{g(x)} \rightarrow g'(x)=0 \rightarrow$ inflection point for x_0

P3 a) $\left(\frac{1}{5}\right)^3\cdot\frac{2}{5}=\frac{2}{625}$; b) $p(\text{sum}\geq 11)=\left(\frac{2}{5}\right)^2+2\cdot\frac{2}{5}\cdot\frac{1}{5}=\frac{8}{25}$

P4 a) e.g. (0|0|1)

b) $\begin{pmatrix} 1 \\ 1 \\ -1 \end{pmatrix}\cdot\begin{pmatrix} -1 \\ 2 \\ 1 \end{pmatrix}=0$ and $\begin{pmatrix} 2 \\ 3 \\ -4 \end{pmatrix}\cdot\begin{pmatrix} -1 \\ 2 \\ 1 \end{pmatrix}=0$; qued

c) e.g. $x + y - z = 0$

P5 a) ascending because $z_2 - z_1 = 2 - 1 > 0$

b) $\overrightarrow{P_1P_2}=\begin{pmatrix} -4 \\ 8 \\ 1 \end{pmatrix}$; $g: \begin{pmatrix} 6 \\ 2 \\ 1 \end{pmatrix}+t\cdot\begin{pmatrix} -4 \\ 8 \\ 1 \end{pmatrix}=\begin{pmatrix} x \\ y \\ 10 \end{pmatrix}$

$t = 9$, $x = -30$, $y = 74$; Position of the airplane: $P(-30|74|10)$

1A a) $kA_b(0) = 10$ (= radius) → ;diameter = 20 m (no influence of parameter b)

$b > 0$, for increasing b the turning point (minimum) moves left down, the y-intercept remains 10.

Considering condition 3: $v = \dfrac{kA_b(35)}{kA_b(x_{min})}$

$kA_b' = 0 \;\rightarrow\;$ Minimum for $x_{min} = \dfrac{\ln(b)}{b} + \dfrac{3.3161}{b}$

$$v = \frac{kA_b(35)}{kA_b\left(\dfrac{\ln(b)}{b} + \dfrac{3.3161}{b}\right)} = 3 \;\rightarrow\; b = 0.107$$

Condition 4:

$kA'(x) = 0 \;\rightarrow\; x_e \approx 10.106$

$2 \cdot kA(10.106) \approx 10.024$

All conditions are fulfilled.

b) $\pi \cdot \int_{20}^{h} kI(x)^2 \, dx = 5000 \;\rightarrow\; h \approx 34.39\,\text{m}$

$M \approx 2034.6\ \text{m}^2$; ~410 l coating are needed

c) Equation of line DW: $y = -1.742x + 75.967$;
its x-intercept ≈ 43.6 (= height of the tower)

strut length ≈ 17.3 m

1B I must apologize because I didn't give functions $g(x)$ and $h(x)$ in DNL#119. Here they are. I am very sorry:

$$g(x) = \sqrt{4 - x^2} \text{ for } x \in [-2;0] \text{ and } h(x) = 0.5(x-5)^2 - 3 \text{ for } x \in [5;9]$$

a) $t(x) = a \cdot x^5 + b \cdot x^4 + c \cdot x^3 + d \cdot x^2 + e \cdot x + f$

$t(0) = g(0)$

$t'(0) = g'(0)$

$t''(0) = g''(0)$

$t(5) = h(5)$

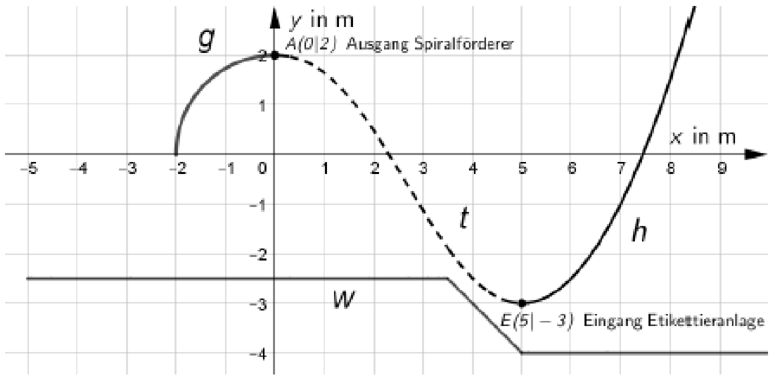
$t'(5) = h'(5)$

$t''(5) = h''(5)$

$$t(x) = -0.0036 \cdot x^5 + 0.05 \cdot x^4 - 0.15 \cdot x^3 - 0.25 \cdot x^2 + 2$$

Length of the belt: $2.1 \cdot \int_0^5 \sqrt{1 + t'(x)^2}\, dx \approx 15.46\,\text{m}$

b) Sketch of the wall:

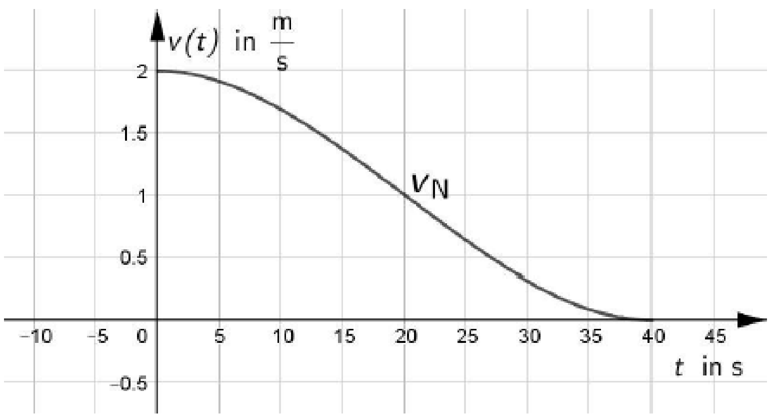


normal line : $(x - 3.755) - 2.755 = x - 6.51$

$h(x) \cap t(x) = S(4.01|-2.5)'$

distance ≈ 0.361 m

c) Sketch of v_N:



Inflection point of v_N: at $t = 20$, $|\dot{v}(t=20)| = 0.075$,

requested condition fulfilled.

Time: $v_N(t) = 0 \rightarrow (t_1 = -20)$, $t_2 = 40$

40 seconds are needed, then the way of the belt is

$$s = \int_0^{40} v_N(t)\,dt = 40; \qquad 10 = \int_a^0 v_{const}(t)\,dt = -2a \;\rightarrow\; a = -5$$

It needs 10 m transport way with regular velocity and needs 5 sec, before starting stopping time such that all bottles can be removed safely from the machine

(2A, 2B and 3A from DNL#120)

2A a) $E(X) = 2.88 \text{ Mio} \cdot 0.15 = 432\,000$ cyclists

$P(X \geq 2600) = 1 - P(X \leq 2599) = 0.24\ \%$ (binomial distribution)

$P(X \leq 1800) = F_{12000,0.15}(1800) = 50.63\%$ (binomial distribution)

$P(15000 < X < 25000) = F_{100000,0.15}(24999) - F_{100000,0.15}(15000) = 49.78\%$ (b.d.)

b) Confidence interval for “wearing helmet always”:

$h = \dfrac{900}{1500} = 0.6 \in [0.3;\, 0.7] \Rightarrow$ approximation permitted (normal distribution)

$\Phi\left(\dfrac{c}{\sigma}\right) = 0.95$ with $\sigma = \sqrt{1500 \cdot 0.6 \cdot 0.4} = \sqrt{360}$

$c \approx 31.21 \rightarrow \left[\dfrac{900 - 31.21}{1500}; \dfrac{900 + 31.21}{1500}\right] = [57.92\%; 62.08\%]$

With a security of 90% a ratio between 57.92% and 62.08% wears a helmet at all occasions. Probability has increased fourfold from 2013 until now.

Confidence interval for „wearing never a helmet”:

$h = \dfrac{375}{1500} = 0.25 \notin [0.3;\, 0.7] \Rightarrow$ approximation not permitted (normal distribution)

using ellipse or parabola

$h(p) = p \pm c \cdot \sqrt{\dfrac{p(1-p)}{n}} = p \pm 1.64 \cdot \sqrt{\dfrac{p(1-p)}{n}}$

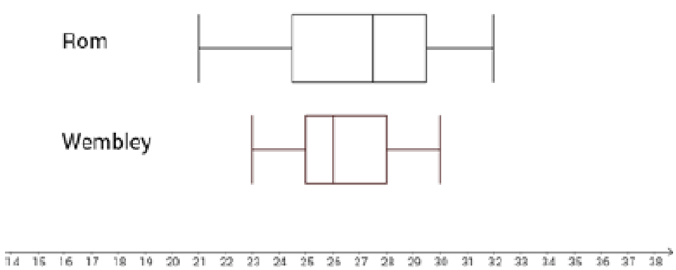
$\text{Interval}_{90\%} = [0.2321;\, 0.2688]$ (using approximation : $[0.2316;\, 0.2684]$

With a security of 90% a ratio between 23.21% and 26.88% does never wear a helmet. Probability has decreased by half from 2013 until now.

2B a) $\bar{x} \approx 26.3$ and $\sigma \approx 2.19 \rightarrow [\bar{x} - \sigma; \bar{x} + \sigma] = [24.11; 28.49]$

Boxplot:

$x_{min} = 23$, $Q_1 = 25$, $Med = 26$, $Q_3 = 28$, $x_{max} = 30$



Wembley meets better the requested “uniform growth”.

b) Binomial distribution ($p = 0.4$)

$p(X \geq 1) = 1 - p(X = 0) \geq 0.95 \rightarrow 0.6^n \leq 0.05 \rightarrow n \geq 5.86$

At least 6 measurements must be taken.

c) sample size $n = 800$, security probability $\gamma = 95\% \rightarrow c = 1.96$

confidence interval = $[p_1, p_2]$

$h = \dfrac{700}{800} = 0.875 \notin [0.3;\, 0.7] \Rightarrow$ approximation not permitted (normal distribution)

using a parabola instead:

$h\,(p - h)^2 - c^2(p - p^2) \leq 0 \rightarrow p_1 \geq 0.8503 \vee p_2 \leq 0.8961$

$\text{Interval}_{95\%} = [0.8503; 0.8961] \Rightarrow$ Wembley seems to be better

3A a) $B(3|-2|0)$, $G(-2.4|-1.6|1.2)$

Because points *E* and *F* have maximum distance from *T*, plane *EFT* has minimum inclination.

$A_{EFT} \approx 7.129$; number of tiles $n = \frac{7.129}{0.15^2} \cdot 1.1 = 348.5 \rightarrow \sim 350$ tiles needed

$V_G = 23.434 - 3.584 = 19.85$

$m = 19.85 \cdot 2000 = 39\,700 \rightarrow$ strengthening is not necessary

b) Substituting the coordinates of *C*, *D*, *G*, *H* (three of them are sufficient) gives 3.6 for all of them

or

equation of the plane: $-1.2x + 0.6z = 3.6$

Coordinates of point $S(-2.814|-1.481|0.371)$

pipe length: $|\overrightarrow{TS}| \sim 95\,\text{cm}$

3B a) Minimum distance point *A* – line *g*:

$$\vec{d} = \begin{pmatrix} 21,54 \\ 45,30 \\ 1,34 \end{pmatrix} - r \cdot \begin{pmatrix} 50,00 \\ 86,60 \\ 2,73 \end{pmatrix} = \begin{pmatrix} 21,54 - 50,00 \cdot r \\ 45,30 - 86,60 \cdot r \\ 1,34 - 2,73 \cdot r \end{pmatrix}$$

$$\vec{d} \bullet \begin{pmatrix} 50,00 \\ 86,60 \\ 2,73 \end{pmatrix} = \begin{pmatrix} 21,54 - 50,00 \cdot r \\ 45,30 - 86,60 \cdot r \\ 1,34 - 2,73 \cdot r \end{pmatrix} \bullet \begin{pmatrix} 50,00 \\ 86,60 \\ 2,73 \end{pmatrix} = 0 \quad \Rightarrow r = 0,5$$

$$\vec{d} = \begin{pmatrix} 21,54 \\ 45,30 \\ 1,34 \end{pmatrix} - 0,5 \cdot \begin{pmatrix} 50,00 \\ 86,60 \\ 2,73 \end{pmatrix} = \begin{pmatrix} -3,46 \\ 2,00 \\ -0,03 \end{pmatrix}$$

$$\left|\vec{d}\right| = \sqrt{3,46^2 + 2^2 + 0,03^2} = 3,997$$

Width of the cycle path is approximately 4 m.

Point *P*: $P(12.3|21.3|0.67)$

b) Intersection point *T* of $g_{flight}: \vec{x} = \begin{pmatrix} 2.53 \\ 9.47 \\ 2.05 \end{pmatrix} + t \begin{pmatrix} 10 \\ -10 \\ 1 \end{pmatrix}$ with plane *h* gives point *T*:

$T(4.40|7.60|2.24)$

Flight duration between points V_1 and *T* is 0.48 seconds. The flash light will not be turned on.

Eine andere Sichtweise auf Lotteriespiele

von Wolfgang Pröpper

In Deutschland (in anderen Ländern wird es ähnlich sein) werden die Fernsehzuschauer zweimal wöchentlich mit den Ergebnissen der aktuellen Lotto-Ziehung „beglückt". Für die geldgierigen Zocker, die unterschiedlich viele Münzen eingesetzt haben, mögen die reinen Zahlen der Ziehung wichtig sein. Denn sie entscheiden über Gewinn oder Verlust bei dem Spiel.

Den feinsinnigen Mathematiker interessieren vielleicht andere Fragestellungen:

Wie groß ist die Wahrscheinlichkeit, dass nur gerade Zahlen in der Ziehung vorkommen (sehr klein, weniger als 1 %), oder dass die Ziehung nur Primzahlen aufweist (noch seltener, ca. 0,2 ‰), oder gar die Quersumme der gezogenen Zahlen höchstens 3 ist (fast gar nicht, etwa 0,006 ‰).

Deutlich häufiger erscheinen die auch sonst wertfreien Ereignisse. Nämlich, dass eine Ziehung genau einen Zwilling (d.h. 2 aufeinander folgende Zahlen wie z.B. 11 und 12 oder 37 und 38), einen Drilling, mehrere Zwillinge, bis hin zu Siebenlingen aufweist.

Dieser Frage bin ich mit Simulationen der Lotto-Ziehung und ihrer Auswertung nachgegangen. (Bei der Darstellung war mir Josef Böhm eine allzeit ansprechbare Hilfe.)

Im Dokument ***lotto.tns*** kann für eine frei wählbare Anzahl von Lotto-Ziehungen mit (fast) frei wählbarem *m* aus *n* (das m darf die Werte 5, 6 oder 7 annehmen) die Anzahl für „genau 1 Paar", „genau 2 Paare", „genau 1 Drilling" und den „Rest" ausgezählt. Das Ergebnis wird als Tabelle und mit graphischem Vergleich der (theoretischen) Wahrscheinlichkeit und der (praktischen) Frequenz für das Eintreten dieser Ereignisse dargestellt.

Im Dokument ***lotto1.tns*** werden alle möglichen Ereignisse von „genau 1 Paar" über „1 Paar und 1 Drilling" bis zu „1 Siebenling" (für $m = 7$), jedoch nur in tabellarischer Form, ausgegeben. (Dort sind für *m* sogar zusätzlich die Zahlen 3 und 4 möglich.)

Schließlich können im Dokument ***lotto2.tns*** die Ereignisse „1 Paar", „2 Paare", „1 Drilling" und „Rest" für mehrere Folgen von Ziehungen graphisch verglichen werden.

Zur Programmierung (in Nspire-Basic):

In einer Funktion **ziehung(n,m)** wird aus einem anfangs mit der Zahlen 1 bis **n** gefüllten Topf durch Zufall eine Zahl gewählt, dem gezogenen Tupel zugefügt, und aus dem Topf (dessen Umfang dann um 1 verringert wird) entnommen, bis das Ziehungs-Tupel den Umfang **m** erreicht.

Eine Alternative zur o. g. Ziehung wäre **ziehung1(n,m)**. Bei ihr wird auch der mit den Zahlen 1 .. *n* gefüllte Topf angelegt, jedoch nach der Ziehung einer Zahl diese im Topf auf 0 gesetzt. Damit kann sie, sollte die nächste (oder eine spätere) in der aktuellen Ziehung ausgewählte Zahl damit über-einstimmen, übergangen werden. Dieses Verfahren ist jedoch insofern gefährlich, als es in eine Endlos-Schleife hineinlaufen könnte. Das wäre in dem kaum vorstellbaren Fall, dass der Zufallsgenerator sich bei einer schon gezogenen Zahl „festbeißt", sie also immer wieder zieht.

(Wenn man diese Ziehungs-Variante ausprobieren will, muss man in den entsprechenden ***lotto***-Programmen die Anweisung *ziel:=ziehung(n,m)* durch *ziel:=ziehung1(n,m)* ersetzen.)

Die 3 Programme **lotto(anz,n,m)** bis **lotto2(anz,n,m)** rufen die Funktion **ziehung(n,m)** (bzw. **ziehung1(n,m)**) genau **anz**-mal auf. Nach jedem Aufruf wird das gezogene **m**-Tupel untersucht. Dabei tut die Nspire-Funktion *ΔList(liste)* gute Dienste, denn sie gibt eine Liste zurück, welche die Differenz der Elemente von liste enthält. D. h. benachbarte Zahlen erkennt man in der Ergebnisliste an einer 1, nicht benachbarte an einer Zahl größer als 1. So können die Paare, Drillinge etc. einfach erkannt und ausgezählt werden.

Die drei Nspire-Dokumente sind jeweils mit einem einleitenden Text und einem Beispiels-Datensatz versehen, sodass es leicht ist, weitere „überflüssige“ 😉 Erfahrungen mit Lotteriespielen zu sammeln.

Sehr empfehlenswert in diesem Zusammenhang ist ein Artikel von Maria Koth [1].

Another View on Lottery games

by Wolfgang Pröpper

In Germany (it will be similar in other countries), TV viewers are "graced" twice a week with the results of the current lottery draw. For the money-hungry gamblers who have staked different amounts of coins, the pure numbers of the draw may be important. Because they decide on profit or loss with the play.

The subtle mathematician may be interested in other questions:

What is the probability that only even numbers appear in the draw (very small, less than 1%), or that the draw shows only prime numbers (even rarer, about 0.2 ‰), or even that the cross sum of the drawn numbers is highest 3 (almost not at all, about 0.006 ‰).

Significantly more frequent appear the also otherwise value-free events. Namely that a draw has exactly one twin (i.e. 2 consecutive numbers like 11 and 12 or 37 and 38), one triplet, several twins, up to septuplets.

I pursued this question with simulations of the lottery drawing and their evaluation. (Josef was an always responsive help for me for representing the outcomes).

In the document ***lotto.tns***, for a freely selectable number of lotto draws with (almost) freely selectable *m* out of *n* (the m may take the values 5, 6 or 7), the number for "exactly 1 pair", "exactly 2 pairs", "exactly 1 triplet" and the "remainder" can be counted. The result is presented as a table and with graphical comparison of the (theoretical) probability and the (practical) frequency for the occurrence of these events.

In the document ***lotto1.tns*** all possible events from "exactly 1 pair" over "1 pair and 1 triplet" up to "1 septuplet" (for *m* = 7) are output, but only in tabular form. (There for m even additionally the numbers 3 and 4 are possible).

Finally, in the document ***lotto2.tns*** the events "1pair", "2pairs", "1 triplet" and "remainder" can be compared graphically for several draw sequences.

For programming (in Nspire-Basic):

In a function **ziehung(n,m)**, a number is randomly chosen from an urn initially filled with numbers 1 through **n**, added to the drawn tuple, and removed from the urn (whose content is then decreased by 1) until the target-tuple reaches the length of **m**.

An alternative to the above drawing would be **ziehung1(n,m)**. In this case, the urn filled with the numbers 1 ... *n* is also created, but after drawing a number, this number is set to 0 in the urn. Thus, if the next (or a later) number selected in the current draw coincides with it, it can be skipped. However, this procedure is dangerous in that it could run into an infinite loop. This would be in the hardly imaginable case that the random number generator "gets stuck" on a number already drawn, i.e. draws it again and again.

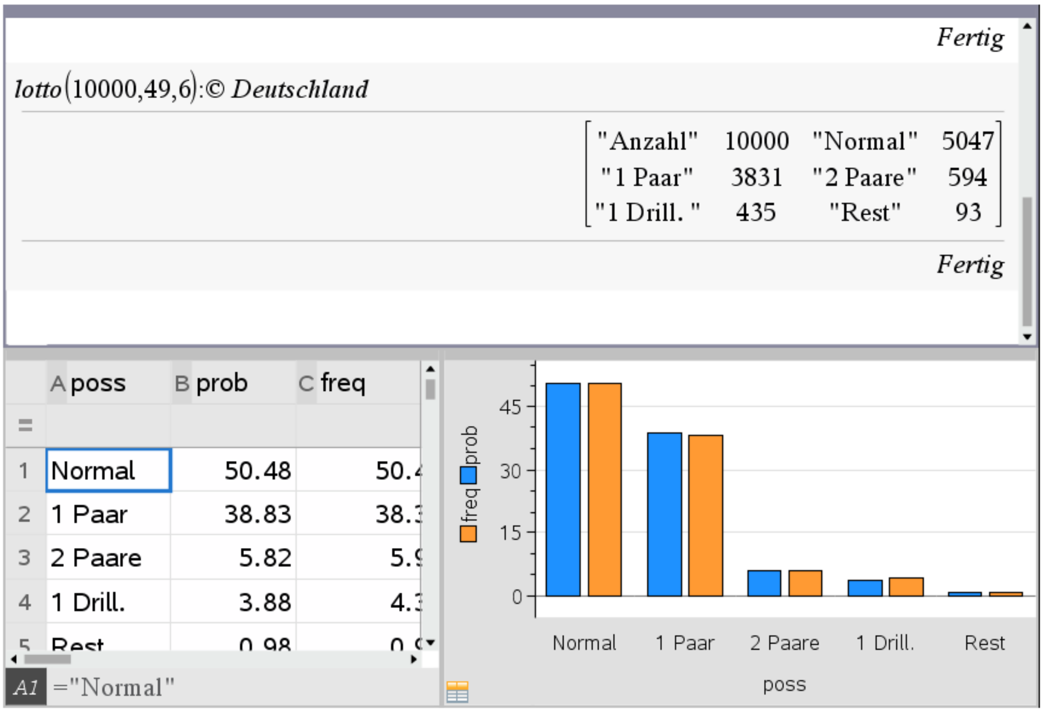
(If you want to try this drawing variant, you have to replace the instruction *ziel:=ziehung(n,m)* by *ziel:=ziehung1(n,m)* in the corresponding ***lotto*** programs).

The 3 programs **lotto(anz,n,m)** to **lotto2(anz,n,m)** call the function **ziehung(n,m)** (resp. **ziehun1g(n,m)**) exactly **anz** times. After each call, the drawn **m**-tuple is examined. Here, the Nspire function *ΔList(liste)* does a good job, because it returns a list containing the difference of the elements of *liste*. That is, adjacent numbers can be recognized in the result list by a 1, and non-neighbored numbers by a number greater than 1. Thus, the pairs, triplets, etc. can be easily recognized and counted.

The three Nspire documents are each provided with an introductory text and an example data set, so that it is easy to gain additional “superfluous” ☺ experience with lottery games.

You can find German and English versions of these file in mth121.zip.

Recommended reading: Maria Koth’s paper (in German) [1].

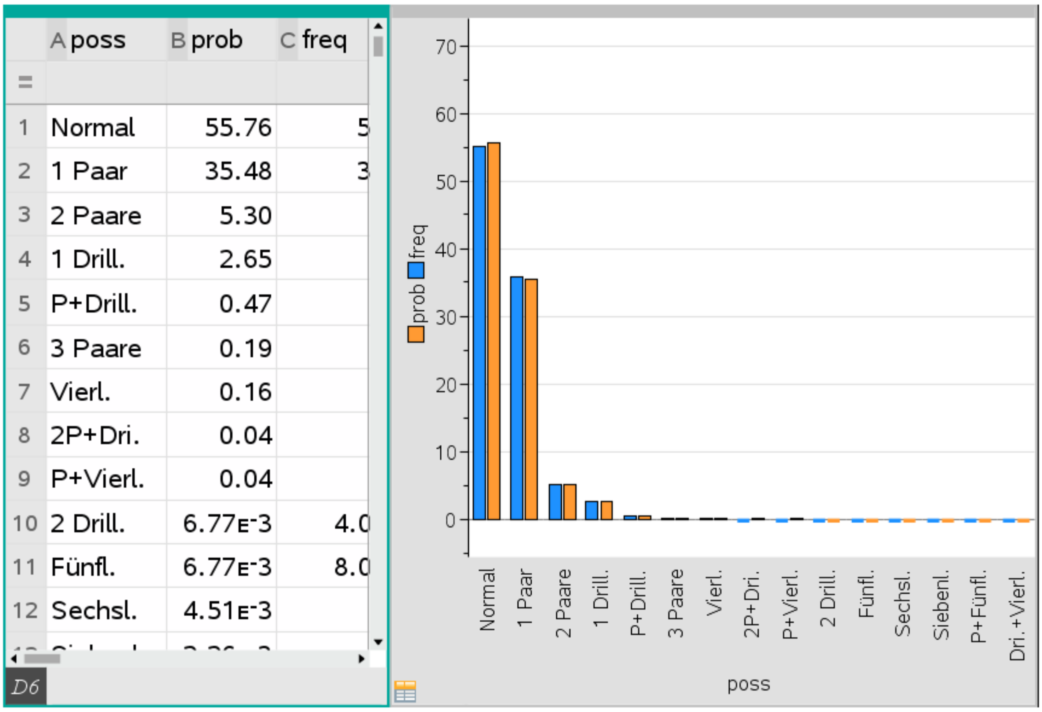


Screen shots of the German version:

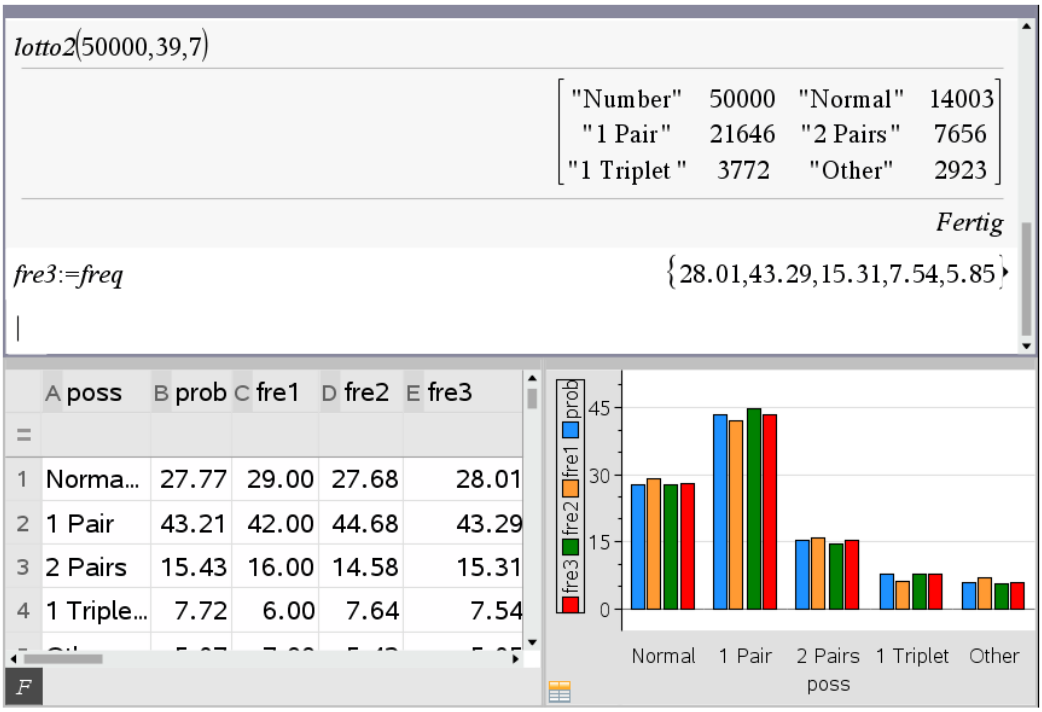
lotto1(50000,78,7):© *gibts auch nicht*

"Anzahl"	50000	"Normal"	27606
"1 Paar"	17966	"2 Paare"	2660
"3 Paare"	73	"1 Drill."	1355
"2 Drill."	2	"1 Vierl."	91
"1 Fünfl."	4	"1 Sechsl."	0
"1 Siebenl."	0	"1 Paar+Dri."	228
"2 Paare+Dri."	5	"Paar+Vierl."	10
"Paar+Fünfl."	0	"Dri.+Vierl."	0

Fertig



One screen shot of the English version:



[1] https://www.mathe-online.at/materialien/maria.koth/files/allerlei/Lotto.doc

[2] https://www.lotteryresults.co.za/tools/lotto/

[3] https://1library.net/document/y4xngm0z-lotteries-teaching-chance-course-written-chance-chance-teachers.html

A Tool for Solving Quadratic Bivariate Diophantine Equations

Bhuvanesh Bhatt (and Josef Böhm)

In DNL#120 I was busy with one of Bhuvanesh Bhatt's function contained in his huge TI-92/V200 library, *mTaylor*. Almost all of his functions work excellent. Some of them are very very extended and comprise some V200 screens of code.

So, I came across his function for solving quadratic Diophantine equations. Its code is given on 13 screens full with very tight code in very small characters. This was my challenge, to rewrite *Diophant* at least for TI-Nspire – and if possible, for DERIVE, too.

It was really hard work and took me many hours, evenings and even days. The story of the Diophantine Equations – its successes and failures – are told in this contribution.

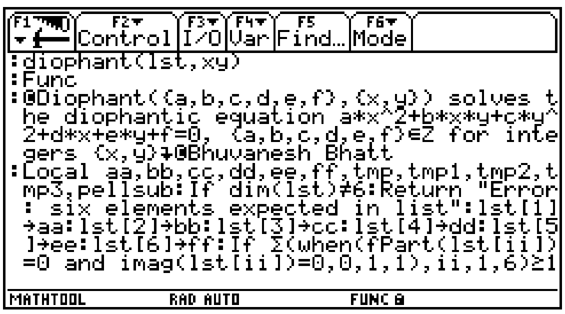
I added my name under the title (in parentheses) because I came across some severe bugs in the original code during the transcription process, which I tried to correct. I learned a lot about this type of equations finding very interesting resources in the web. Most helpful was Alpertron website [6] given at the end of the article.

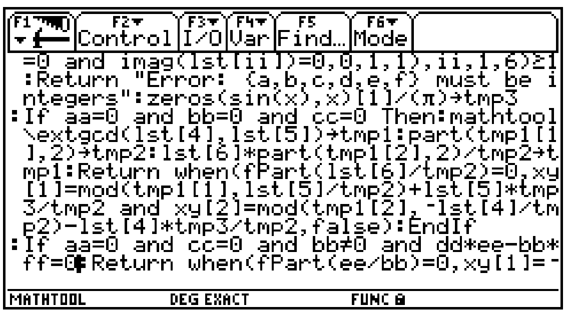
I start with Bhuvanesh's description of the function together with the V200-screen shots.

Diophant({a,b,c,d,e,f},{x,y}) solves the quadratic bivariate diophantine equation
$a\cdot x^2+b\cdot x\cdot y+c\cdot y^2+d\cdot x+e\cdot y+f = 0$, $\{a,b,c,d,e,f\} \in \mathbb{Z}$, for integers {x,y}

Needs: CFracExp, ContFrc1, Divisors, ExtGCD, ListSwap, m_or

Examples:
Diophant({42,8,15,23,17,-4915},{x,y}) ⇒ x= -11 and y= -1,
exp►list(Diophant({0,2,0,5,56,7},{x,y}),{x,y}) ⇒
[[105,-2][-9,1][-21,7][-27,64][-29,-69][-35,-12][-47,-6][-161,-3]],
Diophant({0,0,0,5,22,18},{x,y}) ⇒ x=22·@n1+8 and y=-5·@n1-1,
Diophant({8,-24,18,5,7,16},{x,y}) ⇒ x=41·@n2–174·@n2^2–4 and y=37·@n2–116·@n2^2–4 or
x=17·@n2–174·@n2^2–2 and y=21·@n2–116·@n2^2–2,
Diophant({1,0,4,0,0,-1},{x,y}) ⇒ x=1 and y=0 or x=-1 and y=0,
Diophant({1,0,3,0,0,1},{x,y}) ⇒ false,
Diophant({1,0,-2,0,0,-1},{x,y}) ⇒ x=cosh(@n3·ln(…)) and y=sinh(@n3·ln(…))·√(2)/2
and @n3 ≥ 0 or …
Note: Diophant returns false if there are no solutions for {x,y} in the integers.
Diophant can currently solve any solvable linear, elliptic, or parabolic equation, as well as many hyperbolic (including some Pell-type) equations.





```
 ff=0:Return when(fPart(ee/bb)=0,xy[1]=-
 ee/bb and xy[2]=tmp3,false) or when(fPa
 rt(dd/bb)=0,xy[1]=tmp3 and xy[2]=-dd/bb
 ,false)
:If aa=0 and cc=0 and bb≠0 and dd*ee-bb*
 ff≠0 Then:mathtool\divisors(dd*ee-bb*ff
 )→tmp:augment(tmp,-tmp)→tmp:Return math
 tool\mor(seq(when(fPart((tmp[ii]-ee)/bb
 )=0,xy[1]=(tmp[ii]-ee)/bb and xy[2]=((d
 d*ee-bb*ff)/(tmp[ii])-dd)/bb,false),ii,
 1,dim(tmp))):EndIf
:If bb^2-4*aa*cc<0 Then
```

```
:If bb^2-4*aa*cc<0 Then
:cZeros((bb^2-4*aa*cc)*qq^2+2*(bb*ee-2*c
 c*dd)*qq+ee^2-4*cc*ff,qq)→tmp:If when(i
 mag(tmp[1])=0 and imag(tmp[2])=0,false,
 true,true):Return false:If tmp[1]>tmp[2
 ]:mathtool\listswap(tmp,1,2)→tmp:{ceili
 ng(tmp[1]),floor(tmp[2])}→tmp1:If tmp1[
 1]>tmp1[2]:Return false:Return mathtool
 \mor(seq(when(fPart((-(bb*ii+ee)+√((bb*
 ii+ee)^2-4*cc*(aa*ii^2+dd*ii+ff)))/(2*c
 c))=0,xy[1]=ii and xy[2]=(-(bb*ii+ee)+√
 ((bb*ii+ee)^2-4*cc*(aa*ii^2+dd*ii+ff)))
```

```
 ((bb*ii+ee)^2-4*cc*(aa*ii^2+dd*ii+ff)))
 /(2*cc),false) or when(fPart((-(bb*ii+e
 e)-√((bb*ii+ee)^2-4*cc*(aa*ii^2+dd*ii+f
 f)))/(2*cc))=0,xy[1]=ii and xy[2]=(-(bb
 *ii+ee)-√((bb*ii+ee)^2-4*cc*(aa*ii^2+dd
 *ii+ff)))/(2*cc),false),ii,tmp1[1],tmp1
 [2]))
:EndIf
:If bb^2-4*aa*cc=0 Then:gcd(aa,cc)→tmp1:
 If sign(tmp1)≠sign(aa):-tmp1→tmp1:{aa,b
 b,cc}/tmp1→tmp2:√(tmp2[1])→tmp2[1]:sign
 (bb/aa)*√(tmp2[3])→tmp2[3]:If tmp2[3]*d
```

```
 (bb/aa)*√(tmp2[3])→tmp2[3]:If tmp2[3]*d
 d-tmp2[1]*ee=0 Then:cZeros(tmp2[1]*tmp1
 *qq^2+dd*qq+tmp2[1]*ff,qq)→tmp:Return m
 athtool\diophant({0,0,0,tmp2[1],tmp2[3]
 ,-tmp[1]},xy) or mathtool\diophant({0,0
 ,0,tmp2[1],tmp2[3],-tmp[2]},xy):Else:ma
 thtool\mor(seq(when(mod(tmp2[1]*tmp1*ii
 ^2+dd*ii+tmp2[1]*ff,tmp2[3]*dd-tmp2[1]*
 ee)=0,qq=ii,false),ii,0,ceiling(abs(tmp
 2[3]*dd-tmp2[1]*ee)-1)))→tmp:exp▸list(t
 mp,qq)→tmp:Return mathtool\mor(seq(xy[1
 ]=tmp2[3]*tmp1*(tmp2[1]*ee-tmp2[3]*dd)*
```

```
 ]=tmp2[3]*tmp1*(tmp2[1]*ee-tmp2[3]*dd)*
 tmp3^2-(ee+2*tmp2[3]*tmp1*tmp[ii])*tmp3
 -(tmp2[3]*tmp1*tmp[ii]^2+ee*tmp[ii]+tmp
 2[3]*ff)/(tmp2[3]*dd-tmp2[1]*ee) and xy
 [2]=tmp2[1]*tmp1*(tmp2[3]*dd-tmp2[1]*ee
 )*tmp3^2+(dd+2*tmp2[1]*tmp1*tmp[ii])*tm
 p3+(tmp2[1]*tmp1*tmp[ii]^2+dd*tmp[ii]+t
 mp2[1]*ff)/(tmp2[3]*dd-tmp2[1]*ee),ii,1
 ,dim(tmp))):EndIf:EndIf
:©Solutions of the homogeneous equation
 a*x^2+b*x*y+c*y^2+f=0
:If dd=0 and ee=0 and bb^2-4*aa*cc>0 and
```

```
:If dd=0 and ee=0 and bb^2-4*aa*cc>0 and
  fPart(√(bb^2-4*aa*cc))=0 Then:If ff=0:
 Return xy[1]=0 and xy[2]=0 or mathtool\
 diophant({0,0,0,2*aa,bb+√(bb^2-4*aa*cc)
 ,0},xy) or mathtool\diophant({0,0,0,2*a
 a,bb-√(bb^2-4*aa*cc),0},xy):√(bb^2-4*aa
 *cc)→tmp:mathtool\divisors(-4*aa*ff)→tm
 p1:augment(tmp1,-tmp1)→tmp1:Return math
 tool\mor(seq(when(fPart(tmp1[ii]-(bb+tm
 p)*(tmp1[ii]+4*aa*ff/(tmp1[ii]))/(2*tmp
 ))=0 and fPart((tmp1[ii]+4*aa*ff/(tmp1[
 ii]))/(2*tmp))=0,xy[1]=tmp1[ii]-(bb+tmp
```

```
 ii]))/(2*tmp))=0,xy[1]=tmp1[ii]-(bb+tmp
 )*(tmp1[ii]+4*aa*ff/(tmp1[ii]))/(2*tmp)
  and xy[2]=(tmp1[ii]+4*aa*ff/(tmp1[ii])
 )/(2*tmp),false),ii,1,dim(tmp1)))
:EndIf
:If dd=0 and ee=0 and bb^2-4*aa*cc>0 The
 n
:If ff=0:Return xy[1]=0 and xy[2]=0:gcd(
 gcd(aa,bb),cc)→tmp:If mod(ff,tmp)≠0:Ret
 urn false:aa/tmp→aa:bb/tmp→bb:cc/tmp→cc
 :ff/tmp→ff:©Continued fraction expansio
 ns…
```

```
 :ff/tmp→ff:©Continued fraction expansio
 ns…
:EndIf
:If aa=1 and bb=0 and floor(√(abs(cc)))=
 √(abs(cc)) and (abs(ff)=1 or cc=0) Then

:If cc=0:Return when(floor(√(-ff))=√(-ff
 ),xy[1]=√(-ff),false,false)
:If ff=1:Return when(cc<0,cc=-1 and xy[1
 ]=0 and (xy[2]=1 or xy[2]=-1),false,und
 ef)
:If ff=-1:Return xy[2]=0 and (xy[1]=1 or
```

```
 ef)
:If ff=-1:Return xy[2]=0 and (xy[1]=1 or
  xy[1]=-1) or when(cc<0 or cc=1,false,x
 y[1]=0 and (xy[2]=1 or xy[2]=-1),undef)

:EndIf
:Define pellsub(nn)=Func:Local tmp:If fl
 oor(√(nn))=√(nn):Return unimpl:dim(part
 (mathtool\cfracexp(√(nn),∞)[2],2))→tmp:
 If mod(tmp,2)=1:2*tmp→tmp:mathtool\cont
 frc1(mathtool\cfracexp(√(nn),tmp))→tmp:
 {getNum(tmp),getDenom(tmp)}:EndFunc
```

```
 {getNum(tmp),getDenom(tmp)}:EndFunc
:If aa=1 and bb=0 and cc<0 and floor(√(-
 cc))≠√(-cc) and ff=-1 Then
:pellsub(-cc)→tmp:{tmp[1]-√(-cc)*tmp[2],
 tmp[1]+√(-cc)*tmp[2]}^tmp3→tmp:Return t
 mp3≥0 and (xy[1]=(-tmp[1]-tmp[2])/2 and
  (xy[2]=-(tmp[1]-tmp[2])/(2*√(-cc)) or
 xy[2]=(tmp[1]-tmp[2])/(2*√(-cc))) or xy
 [1]=(tmp[1]+tmp[2])/2 and (xy[2]=-(tmp[
 1]-tmp[2])/(2*√(-cc)) or xy[2]=(tmp[1]-
 tmp[2])/(2*√(-cc))))
:EndIf
```

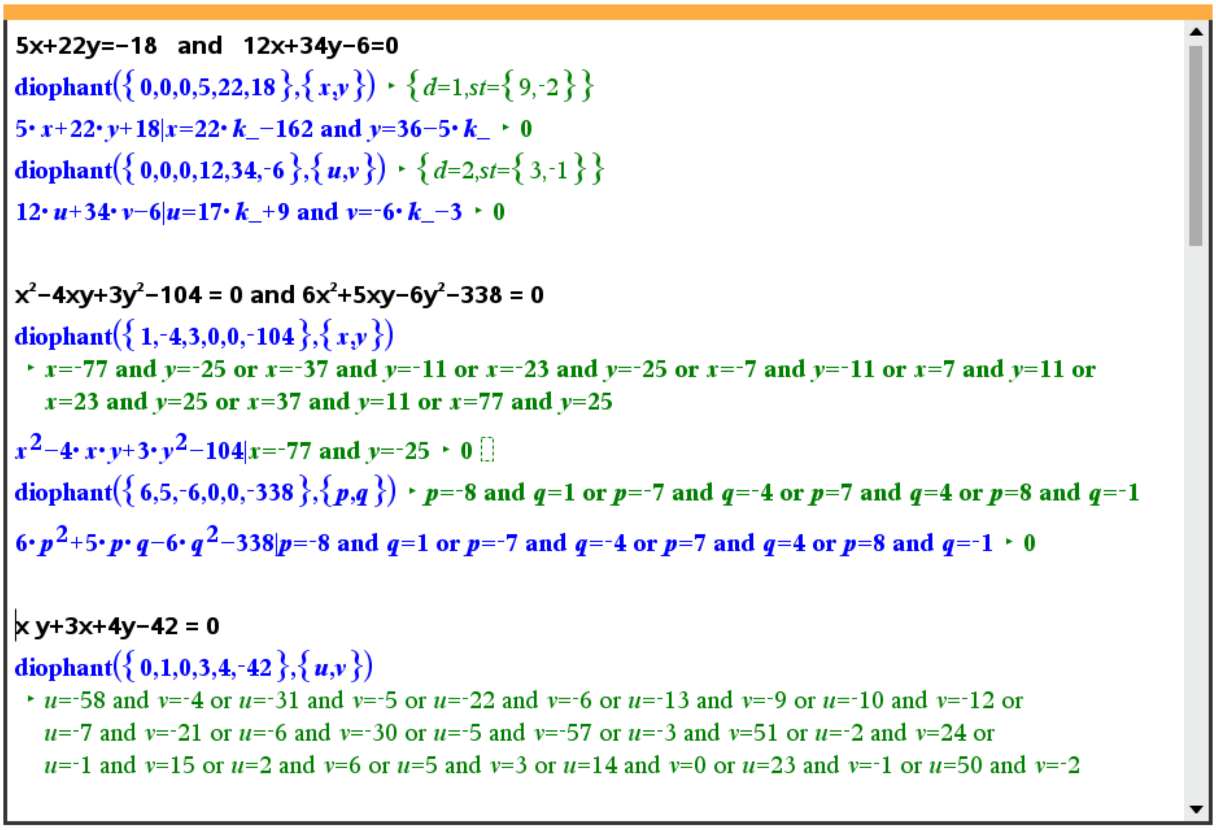
```
 cc))≠√(-cc) and ff=-1 Then
:pellsub(-cc)→tmp:{tmp[1]-√(-cc)*tmp[2],
 tmp[1]+√(-cc)*tmp[2]}^tmp3→tmp:Return t
 mp3≥0 and (xy[1]=(-tmp[1]-tmp[2])/2 and
  (xy[2]=-(tmp[1]-tmp[2])/(2*√(-cc)) or
 xy[2]=(tmp[1]-tmp[2])/(2*√(-cc))) or xy
 [1]=(tmp[1]+tmp[2])/2 and (xy[2]=-(tmp[
 1]-tmp[2])/(2*√(-cc)) or xy[2]=(tmp[1]-
 tmp[2])/(2*√(-cc))))
:EndIf
:unimpl
:EndFunc
```

Can you imagine the hard work?

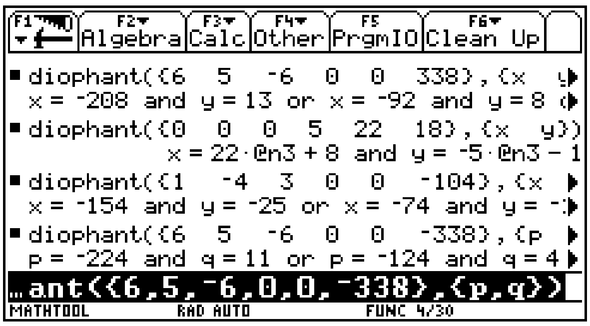
Before I started writing the Nspire-version I checked Bhuvanesh's examples – and I found that some of them are not correct. So, the solution of example 3 does not fit. I was surprised that even the linear Diophantine equations gave wrong results. See the result of example 3. Another type with $a, b, c, f, \neq 0$ and $d = e = 0$ gave also wrong results. (screenshot follows below).

Unfortunately, I have little knowledge – to be honest, very little knowledge about D. E. (except the linear ones). So, I had to start an intense internet research and I asked our DUG-expert Johann Wiesenbauer for assistance. Supported by his advice and a couple of useful websites I could correct these deficiencies:

The V200-TI-screen shows the original results for four Diophantine equations. The Notes-App gives the correct answers.



Compare the above results with the results given by the TI-92/Voyage 200:



Calculation needs much more calculation time on the TI-92/V200, though.

I tried to do my best and finally, I got a version which could solve all examples given by Bhuvanesh.

```
diophant({42,8,15,23,17,-4915},{x,y})
                                        {x=-11 and y=-1}
exp▸list(diophant({0,2,0,5,56,7},{u,v}),{u,v})
    [ 105   -2 ]
    [  -9    1 ]
    [ -21    7 ]
    [ -27   64 ]
    [ -29  -69 ]
    [ -35  -12 ]
    [ -47   -6 ]
    [-161   -3 ]
diophant({8,-24,18,5,7,16},{x,y})
 x=-174·k_^2+41·k_-4 and y=-116·k_^2+37·k
diophant({1,0,4,0,0,-1},{x,y})
                          {x=-1 and y=0,x=1 and y=0}
diophant({1,0,3,0,0,1},{x,y})                    false
diophant({1,0,-2,0,0,-1},{x,y})
```

```
diophant                                                  101/103
Define LibPub diophant(lst,xy)=
Func
© Bhuvanesh Bhatt
Local aa,bb,cc,dd,ee,ff,tmp,tmp1,tmp2,tmp3,pellsı
Local tmp3
If dim(lst)≠6:Return "Error: six coefficients expe
aa:=lst[1]:bb:=lst[2]:cc:=lst[3]:dd:=lst[4]:ee:=lst|
If Σ(ii=1,6) (when(fPart(lst[ii])=0 and imag(lst[ii])=C
© linear diophantine equation
 If aa=0 and bb=0 and cc=0:  Return mathtool\
© mixed case
If aa=0 and cc=0 and bb≠0 and dd·ee-bb·ff=0
 Return when(fPart(ee/bb)=0,xy[1]=-ee/bb and xy[2]=
©
 If aa=0 and cc=0 and bb≠0 and dd·ee-bb·ff≠0
 tmp:=mathtool\divisors(dd·ee-bb·ff):tmp:=augı
```

I was surprised to find a solution – which should consist of integers only – presenting cosh and sinh, logarithm and square roots …

```
rr:=diophant({1,0,-2,0,0,-1},{x,y})
 ▸ x=cosh(k_·ln(-(2·√2-3))) and y=sinh(k_·ln(-(2·√2-3)))·√2/2 and k_≥0 or
   x=cosh(k_·ln(-(2·√2-3))) and y=-sinh(k_·ln(-(2·√2-3)))·√2/2 and k_≥0 or
   x=-cosh(k_·ln(-(2·√2-3))) and y=sinh(k_·ln(-(2·√2-3)))·√2/2 and k_≥0 or
   x=-cosh(k_·ln(-(2·√2-3))) and y=-sinh(k_·ln(-(2·√2-3)))·√2/2 and k_≥0
x^2-2·y^2|rr ▸ 1
seq(rr,k_,0,2)
 ▸ {x=1 and y=0 or x=1 and y=0,x=-3 and y=-2 or x=-3 and y=2 or x=3 and y=-2 or x=3 and y=2,x=-17 and y=
x^2-2·y^2|x=1 and y=0 or x=3 and y=2 or x=17 and y=12 or x=99 and y=70 or x=577 and y=408 ▸ 1
seq(x=-cosh(k_·ln(-(2·√2-3))) and y=-sinh(k_·ln(-(2·√2-3)))·√2/2 and k_≥0,k_,0,4)
 ▸ {x=-1 and y=0,x=-3 and y=2,x=-17 and y=12,x=-99 and y=70,x=-577 and y=408}
x^2-2·y^2|x=-1 and y=0 or x=-3 and y=2 or x=-17 and y=12 or x=-99 and y=70 or x=-577 and y=408 ▸ 1
```

Substitution for the parameter leads to integer numbers which prove to be solutions of the equation. This is a solution of the so-called *Pell's equation*, which is a special case of the Diophantine equation.

In [6] I found this solution (together with an extended explanation demonstrating stepwise the solving procedure):

$x^2 - 2y^2 - 1 = 0$

$x = 1$
$y = 0$

and also:

$x = -1$
$y = 0$

Recursive solutions:

$x_{n+1} = 3x_n + 4y_n$
$y_{n+1} = 2x_n + 3y_n$

and also:

$x_{n+1} = 3x_n - 4y_n$
$y_{n+1} = -2x_n + 3y_n$

In the spreadsheet App we can produce the recursively defined solutions of this equation.

Compare with the elements presented above in the sequences (in the Notes).

1	0	-1	0	1	0	-1	0	1
3	2	-3	-2	3	-2	-3	2	1
17	12	-17	-12	17	-12	-17	12	1
99	70	-99	-70	99	-70	-99	70	1
577	408	-577	-408	577	-408	-577	408	1
3363	2378	-3363	-2378	3363	-2378	-3363	2378	1
19601	13860	-19601	-13860	19601	-13860	-19601	13860	1
114243	80782	-114243	-80782	114243	-80782	-114243	80782	1

Theory of Diophantine equations is very extended, because there are so many cases to be considered. A lot of number theory knowledge is necessary. One has to know how to use continued fractions, and and and…

Some more examples are following:

```
diophant({6,-3,10,-15,61,-887},{x,y}) ▸ {x=-7 and y=4,x=1 and y=7,x=5 and y=7}
6·x^2-3·x·y+10·y^2-15·x+61·y-887|x=-7 and y=4 or x=1 and y=7 or x=5 and y=7 ▸ 0
diophant({0,1,0,3,4,-42},{u,v})
 ▸ u=-58 and v=-4 or u=-31 and v=-5 or u=-22 and v=-6 or u=-13 and v=-9 or u=-10 and v=-12 or u=-7 and v=-21 or
   u=-6 and v=-30 or u=-5 and v=-57 or u=-3 and v=51 or u=-2 and v=24 or u=-1 and v=15 or u=2 and v=6 or
   u=5 and v=3 or u=14 and v=0 or u=23 and v=-1 or u=50 and v=-2
u·v+3·u+4·v-42|u=-31 and v=-5 ▸ 0

diophant({3,-7,11,0,0,-1947},{x,y})
 ▸ {x=-32 and y=-9,x=-11 and y=9,x=-11 and y=-16,x=11 and y=16,x=11 and y=-9,x=32 and y=9}

diophant({1,-4,3,0,0,-104},{p,q})
 ▸ p=-77 and q=-25 or p=-37 and q=-11 or p=-23 and q=-25 or p=-7 and q=-11 or p=7 and q=11 or p=23 and q=25 or
   p=37 and q=11 or p=77 and q=25
p^2-4·p·q+3·q^2-104|p=-77 and q=-25 or p=-37 and q=-11 or p=-23 and q=-25 ▸ 0

diophant({0,2,0,2,4,-42},{x,y})
 ▸ x=-48 and y=-3/2 or x=-25 and y=-2 or x=-4 and y=-25/2 or x=-3 and y=-24 or x=-1 and y=22 or x=0 and y=21/2 or
   x=21 and y=0 or x=44 and y=-1/2
2·x·y+2·x+4·y-42|x=-25 and y=-2 ▸ 0  2·x·y+2·x+4·y-42|x=-3 and y=-24 ▸ 0
2·x·y+2·x+4·y-42|x=-4 and y=-25/2 ▸ 0
```

As you can see in the last example above, that there also fractions can appear as solutions. So, pick out the integer ones but the rational solutions match also.

parabolic case: b^2 - 4a*c = 0

diophant({8,-24,18,-6,-9,-209},{x,y})

$x=216\cdot k\_^2+417\cdot k\_+\frac{735}{4}$ and $y=144\cdot k\_^2+266\cdot k\_+\frac{667}{6}$ or

$x=216\cdot k\_^2+381\cdot k\_+\frac{301}{2}$ and $y=144\cdot k\_^2+242\cdot k\_+90$ or $x=216\cdot k\_^2+357\cdot k\_+130$ and $y=144\cdot k\_^2+226\cdot k\_+77$ or

$x=216\cdot k\_^2+321\cdot k\_+\frac{407}{4}$ and $y=144\cdot k\_^2+202\cdot k\_+\frac{355}{6}$ or

$x=216\cdot k\_^2+201\cdot k\_+\frac{117}{4}$ and $y=144\cdot k\_^2+122\cdot k\_+\frac{85}{6}$ or $x=216\cdot k\_^2+165\cdot k\_+14$ and $y=144\cdot k\_^2+98\cdot k\_+5$ or

$x=216\cdot k\_^2+141\cdot k\_+\frac{11}{2}$ and $y=144\cdot k\_^2+82\cdot k\_$ or $x=216\cdot k\_^2+105\cdot k\_-\frac{19}{4}$ and $y=144\cdot k\_^2+58\cdot k\_-\frac{35}{6}$

Take the solutions with integer coefficients

$x=216\cdot k\_^2+357\cdot k\_+130$ and $y=144\cdot k\_^2+226\cdot k\_+77|k\_=\{0,1\}$ ▸ $x=\{130,703\}$ and $y=\{77,447\}$

$x=216\cdot k\_^2+165\cdot k\_+14$ and $y=144\cdot k\_^2+98\cdot k\_+5|k\_=\{-2,-1,0\}$ ▸ $x=\{548,65,14\}$ and $y=\{385,51,5\}$

$x=216\cdot k\_^2+417\cdot k\_+\frac{735}{4}$ and $y=144\cdot k\_^2+266\cdot k\_+\frac{667}{6}|k\_=\{0,1\}$ ▸ $x=\left\{\frac{735}{4},\frac{3267}{4}\right\}$ and $y=\left\{\frac{667}{6},\frac{3127}{6}\right\}$

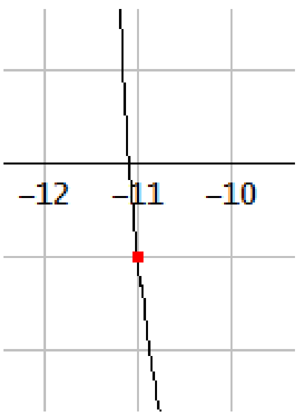
$8\cdot x^2-24\cdot x\cdot y+18\cdot y^2-6\cdot x-9\cdot y-209|x=\frac{735}{4}$ and $y=\frac{667}{6}$ ▸ 0

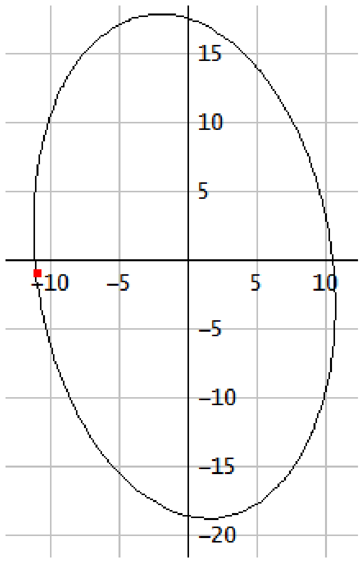
The rational solutions do also fulfull the equation!!

I can imagine to add a “filter” in the function, which selects the integer solutions only. Maybe that this is left to one of the interested readers. Another “filter” is to follow later.

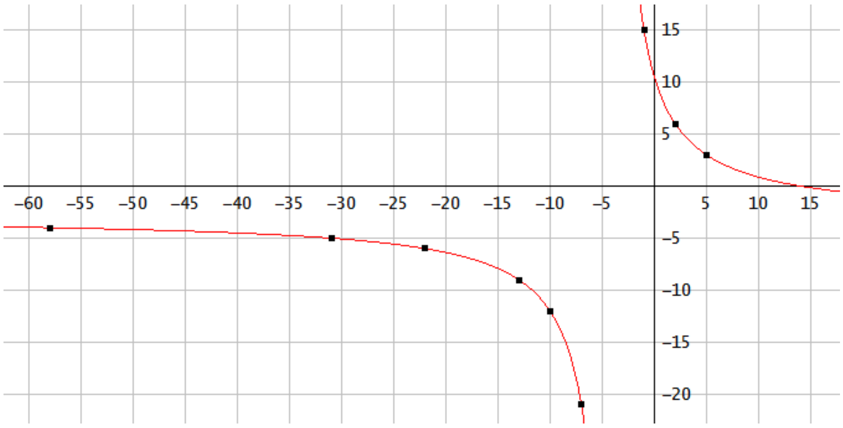
I’d like to show an obvious graphic representation of one equation together with its solution. The equation describe conics – and this is the reason, why it is distinguished between hyperbolic, elliptic and parabolic cases (depends on the sign of $b^2 - 4ac$). The solutions are points on the conic with integer coordinates. See two examples:

Diophant({42,8,15,23,17,-4915},{x,y}) ⇒ x= -11 and y= -1
(elliptic case)





$xy + 3x + 4y - 42 = 0$ together with some solution points (hyperbolic case):



Reference [6] provides four families of solutions for the parabolic case given in the Nspire screenshot above:

8 x² - 24 xy + 18 y² - 6 x - 9 y - 209 = 0

$x = 864\,k^2 + 1578\,k + 703$
$y = 576\,k^2 + 1028\,k + 447$

and also:

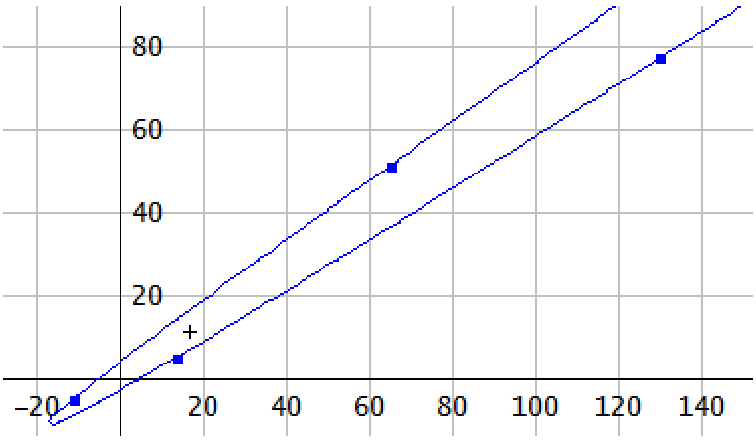
$x = 864\,k^2 + 1194\,k + 395$
$y = 576\,k^2 + 772\,k + 247$

and also:

$x = 864\,k^2 + 714\,k + 130$
$y = 576\,k^2 + 452\,k + 77$

and also:

$x = 864\,k^2 + 330\,k + 14$
$y = 576\,k^2 + 196\,k + 5$



Written by Dario Alpern. Last updated on 18 January 2021.

I calculated some solution points using the formulae given above. Most points are far out of the screen. Only three of them are within plottable range: (-11,-5), (14,5), (130,77) and (65,51), all of them are points on the parabola!

$x^2 + 2xy - 4y^2 - 20 = 0$ cannot be solved by diophant() (Answer: "unimplemented")

Reference [7] gives some solutions with (6,4) and (4,1) among them together with recursive solutions:

Recursive solutions:

$x_{n+1} = 5\,x_n + 16\,y_n$
$y_{n+1} = 4\,x_n + 13\,y_n$

and also:

$x_{n+1} = 13\,x_n - 16\,y_n$
$y_{n+1} = -4\,x_n + 5\,y_n$

I use DERIVE's ITERATES-construct to calculate some more solutions and can immediately check if they are correct.

```
#1:   f(x, y) := x^2 + 2·x·y - 4·y^2 - 20

#2:   ITERATES([5·v1 + 16·v2, 4·v1 + 13·v2], v, [4, 1], 4) =
        [     4        1 ]
        [    36       29 ]
        [   644      521 ]
        [ 11556     9349 ]
        [207364   167761 ]

#3:   ITERATES([5·v1 + 16·v2, 4·v1 + 13·v2], v, [6, 4], 4) =
        [     6        4 ]
        [    94       76 ]
        [  1686     1364 ]
        [ 30254    24476 ]
        [542886   439204 ]

#4:   VECTOR(f(v1, v2), v, [[4, 1], [36, 29], [644, 521], [11556, 9349], [207364, 167761]]) = [0, 0, 0, 0, 0]

#5:   VECTOR(f(v1, v2), v, [[6, 4], [94, 76], [1686, 1364], [30254, 24476], [542886, 439204]]) = [0, 0, 0, 0, 0]
```

Now I will remain working with DERIVE. Not only to verify solutions but to solve quadratic Diophantine equations.

This was not an easy task, too. Everybody who ever wrote larger DERIVE programs/functions knows that editing larger functions is not so comfortable and it needs some workaround because we miss the for-endfor, while-endwhile loops. Bhuvanesh prefers working with nested if, if-then and when-constructions as you can see in the TI-92 screen shots above.

I helped myself by splitting up the huge function in one main function (diophant) and a few subfunctions – and this made programmer's life much easier.

One of the subroutines treats linear Diophantine equations:

```
diophl(lst, xy, tmp, tmp1) :=
  Prog
    If MOD(lst↓3, GCD(lst↓2, lst↓1)) ≠ 0
       RETURN "no solution"
    tmp1 := EXTENDED_GCD(lst↓1, lst↓2)
    tmp := [lst↓2, - lst↓1]·k_/GCD(lst↓2, lst↓1) - lst↓3/GCD(lst↓2,
              lst↓1)·[tmp1↓2↓1, tmp1↓2↓2]
    RETURN [xy↓1 = tmp↓1 ∧ xy↓2 = tmp↓2]

diophl([12, 34, -6], [p, q]) = [p = 17·k_ + 9 ∧ q = - 6·k_ - 3]
```

dioph_m1 and dioph_m2 solve two forms of the "mixed case":

```
dioph_m1(lst, xy, b, d, e, f, tmp, i, sol) :=
  Prog
    [b := lst↓1, d := lst↓2, e := lst↓3, f := lst↓4]
    sol := []
    tmp := DIVISORS(d·e - b·f)
    If d·e - b·f > 0
       tmp := APPEND(tmp, -tmp)
    i := 1
    Loop
      If i > DIM(tmp)
         RETURN sol
      If INTEGER?((tmp↓i - e)/b)
         sol := APPEND(sol, [xy↓1 = (tmp↓i - e)/b ∧ xy↓2 =
               ((d·e - b·f)/tmp↓i - d)/b])
      i :+ 1
```

```
10xy - 4x + 2y + 12 = 0

dioph_m1([10, -4, 2, 12], [x, y])
```

$$\left[x = -13 \wedge y = \frac{1}{2},\ x = -1 \wedge y = 2,\ x = 0 \wedge y = -6,\ x = 3 \wedge y = 0\right]$$

$$\text{VECTOR}\left(\text{SUBST}(10\cdot x\cdot y - 4\cdot x + 2\cdot y + 12,\ [x,\ y],\ k),\ k,\ \begin{bmatrix} -13 & \frac{1}{2} \\ -1 & 2 \\ 0 & -6 \\ 3 & 0 \end{bmatrix}\right) = [0,\ 0,\ 0,\ 0]$$

The hardest part for me was working with the continued fraction expansion. I have read in the resources that some cases of D.E. are needing it. First of all I had to produce Bhuvanesh's auxiliary functions to create a continued fraction expansion and then to come back to the approximating fraction. See the example for treating √123:

```
cfrac_ex(nn, oo := ∞, tmp) :=
  Prog
    tmp := CONTINUED_FRACTION(nn)
    If oo < ∞
       CONTINUED_FRACTION(nn, oo)
       [tmp↓1, tmp↓[2, ..., DIM(tmp) - 1]]

contfrc1(lst) :=
  If DIM(lst) = 1
     lst↓1
     lst↓1 + 1/contfrc1(REST(lst))

cfrac_ex(√123, 10) = [11, 11, 22, 11, 22, 11, 22, 11, 22, 11, 22]
```

$$\text{contfrc1}([11,\ 11,\ 22,\ 11,\ 22,\ 11,\ 22,\ 11,\ 22,\ 11,\ 22]) = \frac{9551881487599}{861264149131}$$

$$\left[\sqrt{123},\ \frac{9551881487599}{861264149131}\right]$$

[11.09053650, 11.09053650]

Pell's equation $x^2 - c\,y^2 - 1 = 0$ is a special case of D.Es. (see [10])

```
diophant([1, 0, -123, 0, 0, -1], [x, y])
```

$$\left(x = -\frac{(122 - 11\cdot\sqrt{123})^{k\_}}{2} - \frac{(11\cdot\sqrt{123} + 122)^{k\_}}{2} \wedge y = \frac{\sqrt{123}\cdot(11\cdot\sqrt{123} + 122)^{k\_}}{246}\right.$$

$$\frac{(11\cdot\sqrt{123} + 122)^{k\_}}{2} \wedge y = \frac{\sqrt{123}\cdot(122 - 11\cdot\sqrt{123})^{k\_}}{246} - \frac{\sqrt{123}\cdot(11\cdot\sqrt{123} + 122)^{k\_}}{246}$$

$$\left.\frac{\sqrt{123}\cdot(11\cdot\sqrt{123} + 122)^{k\_}}{246} - \frac{\sqrt{123}\cdot(122 - 11\cdot\sqrt{123})^{k\_}}{246}\right) \vee \left(x = \frac{(122 - 11\cdot\sqrt{123})^{k\_}}{2}\right.$$

$$\left.\frac{\sqrt{123}\cdot(11\cdot\sqrt{123} + 122)^{k\_}}{246}\right)$$

subst k_=4

$$(x = -1772148577 \vee x = 1772148577) \wedge (y = -159789256 \vee y = 159789256)$$

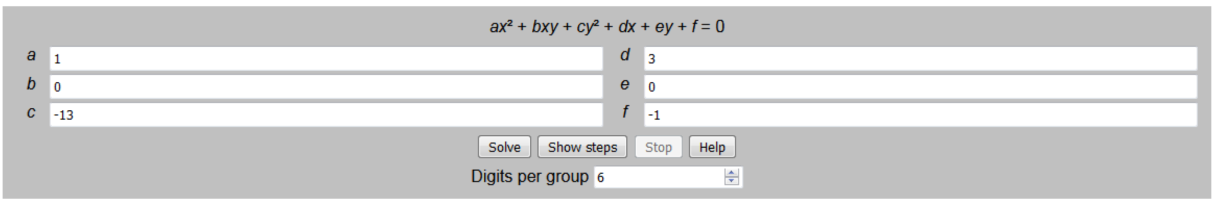
$$(-1772148577)^2 - 123\cdot(-159789256)^2 - 1 = 0$$

$$(x = -1772148577 \wedge y = -159789256) \vee (x = -1772148577 \wedge y = 159789256) \vee (x = 1$$

This gives four pairs of solutions. Every *k_* gives another four pairs. It works for this special equation.

Then I tried $x^2 - 13y^2 + 3x - 1 = 0$ – and received a solution, but it did not match the equation. Same happened with $x^2 - 13y^2 + 3y - 1 = 0$. The solutions were the same as for $x^2 - 13y^2 - 1 = 0$!! I inspected the program and found out that the original program had a special last part reserved for Pell's equation asking for $a = 1$, $b = 0$, $c < 0$, $f = 1$ – and it assumes that $d = e = 0$. Neither d nor e are considered in this part of the function.

Asking Alpertron [6] for the solution(s) of $x^2 - 13y^2 + 3x - 1 = 0$, I received solutions:



x = 31, y = 9

Recursive solutions:

$x_{n+1} = 649\ x_n + 2340\ y_n + 972$
$y_{n+1} = 180\ x_n + 649\ y_n + 270$

x = 31, y = -9

and also:

$x_{n+1} = 649\ x_n - 2340\ y_n + 972$
$y_{n+1} = -\ 180\ x_n + 649\ y_n - 270$

What to do? I was not satisfied to have a function which gives wrong results, then better no results. As Bhuvanesh left for all cases which cannot be solved with his program the answer "not implemented" I built in a "filter" which ignores the wrong solutions. Now it does not give a misleading answer:

```
diophant([1, 0, -13, 3, 0, -1], [x, y])

not implemented
```

Same for the Nspire on the next page:

$diophant(\{1,0,-13,0,0,-1\},\{x,y\})$

$x=\cosh(k\_\cdot \ln(-(180\cdot \sqrt{13}-649)))$ ar▸

$diophant(\{1,0,-13,3,0,-1\},\{x,y\})$ "unimplemented""

$diophant(\{1,0,-13,0,1,-1\},\{x,y\})$ "unimplemented""

```
{getNum(tmp),getDenom(tmp)}
EndFunc
If aa=1 and bb=0 and cc<0 and floor(√-cc)≠√-cc and ff=-1 and dd≠
If aa=1 and bb=0 and cc<0 and floor(√-cc)≠√-cc and ff=-1 Then
 tmp:=mathtool\pellsub(-cc)
 tmp:={tmp[1]-√-cc·tmp[2],tmp[1]+√-cc·tmp[2]}^k_
 Return k_≥0 and |xy[1]=(-tmp[1]-tmp[2])/2 and |xy[2]=-(tmp[1]-tmp[2...)/(2·√-cc)
```

Alpertron [6] provides the stepwise solution of this type of D.Es. This is some PC-screens long and addresses again application of continued fraction expansion. I must admit that this was too difficult for me and I didn't find any resource explaining the solving algorithm.

Would be great, if any of our members could either provide a description how to solve them or – even much better, update my program – either Nspire or DERIVE ☺.

Some space is left on this page: Let's check Alpertron's solution from above:

$$\text{ITERATES}([649\cdot v_1 + 2340\cdot v_2 + 972,\ 180\cdot v_1 + 649\cdot v_2 + 270],\ v,\ [31,\ 9],\ 5)$$

$$\begin{bmatrix} 31 & 9 \\ 42151 & 11691 \\ 54713911 & 15174909 \\ 71018616271 & 19697020191 \\ 92182109207791 & 25566717033009 \\ 119652306733098391 & 33185579011825491 \end{bmatrix}$$

$$\text{VECTOR}\left(\text{SUBST}(x^2 - 13\cdot y^2 + 3\cdot x - 1,\ [x,\ y],\ k),\ k,\ \begin{bmatrix} 31 & 9 \\ 42151 & 11691 \\ 54713911 & 15174909 \\ 71018616271 & 19697020191 \\ 92182109207791 & 25566717033009 \\ 119652306733098391 & 33185579011825491 \end{bmatrix}\right)$$

[0, 0, 0, 0, 0, 0]

[1] https://mathworld.wolfram.com/DiophantineEquation2ndPowers.html

[2] https://arxiv.org/ftp/math/papers/0405/0405206.pdf

[3] https://www.researchgate.net/publication/331088172_Positive_integer_solutions_of_some_second-order_Diophantine_equations/link/5c651cc845851582c3e6fa4e/download

[4] http://www.numbertheory.org/php/main_pell.html

[5] http://www.numbertheory.org/php/

[6] https://www.alpertron.com.ar/QUAD.HTM

[7] https://unipub.uni-graz.at/obvugrhs/content/titleinfo/2679661/full.pdf

[8] http://www.mathe2.uni-bayreuth.de/stoll/teaching/DiophGl-WS2018/Skript-DiophGl-pub-print.pdf

[9] https://mathematikalpha.de/pellsche-gleichung

[10] https://mathshistory.st-andrews.ac.uk/HistTopics/Pell/

The Easter Formula

Wolfgang Alvermann, Germany

Prof. Manfred Oswalden, Klosterneuburg-Kierling NÖ, gave a talk 1998 in the *Zeiss Planetarium of the City of Vienna* [1] with among others topics how to calculate the Easter date, called:

Gregorianian Easter Formula without Exceptions

Dividend	*Divisor*	*Integer Quotient*	*Remainder*	
J	*19*	-	*a*	
J	*100*	*b*	*c*	
b	*4*	*d*	*e*	
b+8	*25*	*f*	-	
b-f+1	*3*	*g*	-	
19a+b-d-g+15	*30*	-	*h*	
c	*4*	*i*	*k*	
32+2e+2i-h-k	*7*	-	*l*	
a+11h+22l	*451*	*m*	-	
h+l-7m+114	*31*	*n*	*p*	

Easter **→** ***Day:*** ***p + 1***

→ ***Month:*** ***n***

Which date was Easter Sunday in your year of birth?

Which date was Whit Monday in this year?

1949 : 19 = 102		rem 11	a = 11
1949 : 100 = 19	b = 19	rem 49	c = 49
19 : 4 = 4	d = 4	rem 3	e = 3
27 : 25	f = 1	rem 2	
19 : 3 = 6	g = 6	rem 1	
233 : 30 = 7		rem 23	h = 23
49 : 4 = 12	i = 12	rem 1	k = 1
38 : 7 = 5		rem 3	l = 3
330 : 451 = 0	m = 0	rem 30/41	
140 : 31 = 4	n = 4	rem 16	p = 16

Easter Sunday 1949 was 17 April; Whit Monday was 6 June

The respective Excel-file was attached:

	A	B	C	D	E	F	G
1	**Gregorianische Osterformel ohne Ausnahmen**						
2							
3	**Ihr Geburtsjahr J:**		**1949**				
4							
5		**Dividend**	**Divisor**	**Ganzzahl-Quotient**		**Rest**	
6	**J**	1949	19	102	**-**	11	**a**
7	**J**	1949	100	19	**b**	49	**c**
8	**b**	19	4	4	**d**	3	**e**
9	**b+8**	27	25	1	**f**	2	**-**
10	**b-f+1**	19	3	6	**g**	1	**-**
11	**19a+b-d-g+15**	233	30	7	**-**	23	**h**
12	**c**	49	4	12	**i**	1	**k**
13	**32+2e+2i-h-k**	38	7	5	**-**	3	**l**
14	**a+11h+22i**	330	451	0	**m**	330	**-**
15	**h+l-7m+114**	140	31	4	***n***	16	***p***
16							
17			**Ostersonntag = p + 1:**	**17**			
18			**Monat = n**	**4**			
19			**50 Tage später**				
20			**Pfingstmontag**	**6**			
21			**Monat**	**6**			

In https://exceltricks.blog/feiertagsberechnung-ostersonntag-in-excel-berechnen/ a "0ne-expression-formula" is given: 7*RUNDEN((4&-A1)/7+REST(19*REST(A1;19)-7;30)*0,14;)-6

C1 =7*RUNDEN((4&-A1)/7+REST(19*REST(A1;19)-7;30)*0,14;)-6

	A	B	C	D	E	F
1	1949		17.04.1949			

But, in Excel a date-function is implemented, which makes calculation a little bit easier.
The Nspire has no data function implemented; this is the spread sheet following Wolfgang's recipe and Excel table:

	A year	B divisoı	C dividend	D int_quot	E remainder	F months
=						
1	2021	_	_	_	_	_
2		2021	19	_	7	_
3		2021	100	20	21	March
4		20	4	5	0	April
5		28	25	1	3	May
6		19	3	6	_	June
7		157	30	_	7	_
8		21	4	5	1	_
9		34	7	_	6	_
10		216	451	0	_	_
11		127	31	April	4	Easter/Ostern
12				May	24	Whit Monday/Pfingsmontag

A1 2021

Happy Easter

easter(2021) ▸ {2021, "Easter: 4 April , Whit Monday: 24 May "}

easter(1949) ▸ {1949, "Easter: 17 April , Whit Monday: 6 June "}

seq(easter(k),k,2015,2022) ▸

2015	"Easter: 5 April , Whit Monday: 25 May "
2016	"Easter: 27 March , Whit Monday: 16 May "
2017	"Easter: 16 April , Whit Monday: 5 June "
2018	"Easter: 1 April , Whit Monday: 21 May "
2019	"Easter: 21 April , Whit Monday: 10 June "
2020	"Easter: 12 April , Whit Monday: 1 June "
2021	"Easter: 4 April , Whit Monday: 24 May "
2022	"Easter: 17 April , Whit Monday: 6 June "

Another algorithm found in [3] realized in DERIVE (for years 1583 to 4099):

```
easter(year, firstdig, remain19, d, m, ta, tb, tc, td, te, temp, case1, case2) ≔
  Prog
    case1 ≔ [21, 24, 25, 27, 28, 29, 30, 31, 32, 34, 35, 38]
    case2 ≔ [33, 36, 37, 39, 40]
    firstdig ≔ FLOOR(year, 100)
    remain19 ≔ MOD(year, 19)
    temp ≔ FLOOR((firstdig - 15)/2) + 202 - 11·remain19
    If DIM(SELECT(firstdig = case1↓k, k, DIM(case1))) = 1
       temp ≔ temp - 1
    If DIM(SELECT(firstdig = case2↓k, k, DIM(case1))) = 1
       temp ≔ temp - 2
    temp ≔ MOD(temp, 30)
    ta ≔ temp + 21
    If temp = 29
       ta ≔ ta - 1
    If temp = 28 ∧ remain19 > 10
       ta ≔ ta - 1
    tb ≔ MOD(ta - 19, 7)
    tc ≔ MOD(40 - firstdig, 4)
    If tc = 3
       tc ≔ tc + 1
    If tc > 1
       tc ≔ tc + 1
    temp ≔ MOD(year, 100)
    td ≔ MOD(temp + FLOOR(temp/4), 7)
    te ≔ MOD(20 - tb - tc - td, 7) + 1
    d ≔ ta + te
    If d > 31
       Prog
         d ≔ d - 31
         m ≔ 4
       m ≔ 3
    RETURN [year, d, m]
```

VECTOR(easter(y), y, 2000, 2010)

2000	23	4
2001	15	4
2002	31	3
2003	20	4
2004	11	4
2005	27	3
2006	16	4
2007	8	4
2008	23	3
2009	12	4
2010	4	4

This year we have Easter Sunday on 4 April; which years from 1900 until now had the same date?

$\text{SELECT}(k_{[2,\ 3]} = [4,\ 4],\ k,\ \text{VECTOR}(\text{easter}(y),\ y,\ 1900,\ 2021))$

1915	4	4
1920	4	4
1926	4	4
1999	4	4
2010	4	4
2021	4	4

In which years from 1950 to 2000 was Easter in the last four days of March?

$\text{SELECT}(k_3 = 3 \wedge k_2 \ge 28,\ k,\ \text{VECTOR}(\text{easter}(y),\ y,\ 1950,\ 2000))$

1959	29	3
1964	29	3
1970	29	3
1975	30	3
1986	30	3
1991	31	3
1997	30	3

This is my first attempt with Python, supported by Veit Berger:

```
easterdate.py                                                        1/48
from math import *

def month(x):
    return ["Jan","Feb","March","April","May","June","July","Aug","Sept","Oct","Nov","Dec"][x-1]

def easter(yr):
    a=fmod(yr,19)
    b=floor(yr/100)
    c=fmod(yr,100)
    d=floor(b/4)
    e=fmod(b,4)
    f=floor((b+8)/25)
    g=floor((b-f+1)/3)
    h=fmod(19*a+b-d-g+15,30)
    i=floor(c/4)
    k=fmod(c,4)
    l=fmod(32+2*e+2*i-h-k,7)
    m=floor((a+11*h+22*l)/451)
    n=floor((h+l-7*m+114)/31)
    p=fmod(h+l-7*m+114,31)
    return [floor(n),floor(p+1)]

yr = int(input("Which year?"))
date=easter(yr)
print(month(date[0]),",",date[1])

min = 31; max = 1
minyr = 0; maxyr = 0
```

```
easterdate.py                                                        48/48
    return [floor(n),floor(p+1)]

yr = int(input("Which year?"))
date=easter(yr)
print(month(date[0]),",",date[1])

min = 31; max = 1
minyr = 0; maxyr = 0

for i in range(1800, 2051):
    date = easter(i)
    if month(date[0]) == "March" and date[1] < min:
        min = date[1]
    if month(date[0]) == "April" and date[1] > max:
        max = date[1]

print("Earliest easter date:")
for i in range(1800, 2051):
    date = easter(i)
    if month(date[0]) == "March" and date[1] == min:
        print( "March -", min, "-", i)

print("Latest easter date:")
for i in range(1800, 2051):
    date = easter(i)
    if month(date[0]) == "April" and date[1] == max:
        print("April -", max, "-", i)
```

```
1.1  1.2    easter_pyth                RAD
Python-Shell                             11/11
>>>#Running easterdate.py
>>>from easterdate import *
Which year?2021
April , 4
Earliest easter date:
March - 22 - 1818
Latest easter date:
April - 25 - 1886
April - 25 - 1943
April - 25 - 2038
>>>
```

The tables given in [4] were used to check the results.

[1] https://de.zxc.wiki/wiki/Zeiss_Planetarium_der_Stadt_Wien

[2] https://www.timeanddate.com/calendar/determining-easter-date.html

[3] https://www.assa.org.au/edm

[4] http://www.maa.clell.de/StarDate/feiertage.html

Newton-Raphson and the Jacobian

(from Bhuvanesh Bhatt's TI-92 functions collection)

Bhuvanesh's description:

NewtRaph(f,vars,start,tol,intermresults) returns a solution for f = 0

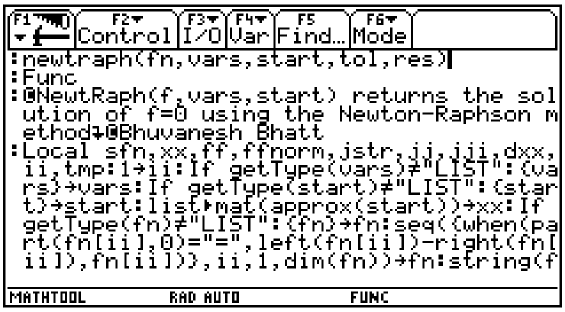
Needs: Jacobian, list2eqn

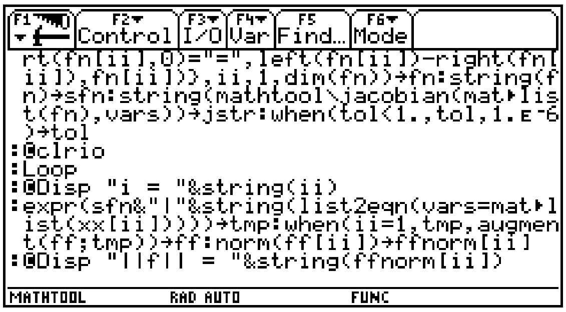
Example: nSolve($x \cdot e^x$ = 2, x = 0) ⇒ 0.852605502014,

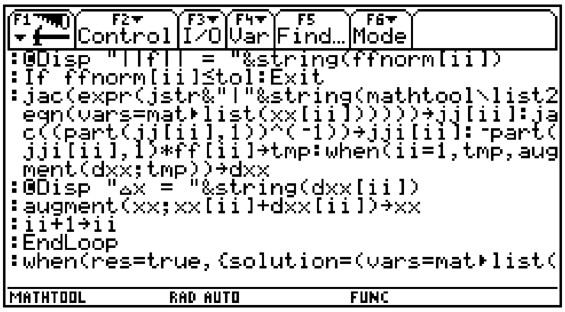
NewtRaph($x \cdot e^x$ = 2, x, 0, 1E-10, false) ⇒ {x=0.852605502014},

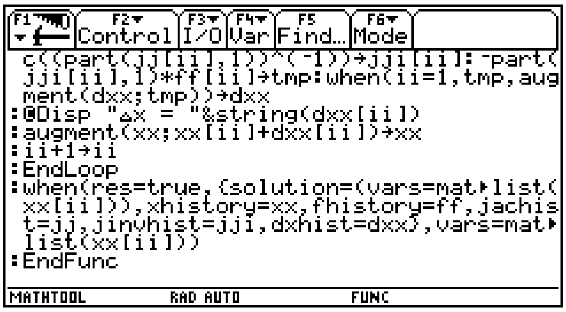
NewtRaph($x \cdot e^x$ = 2, x, 0, 1E-10, true) ⇒ {solution={x=…}, …}

Notes: NewtRaph uses Newton-Raphson iteration. The 'intermresults' argument should be true or false; it specifies whether or not intermediate results are given.



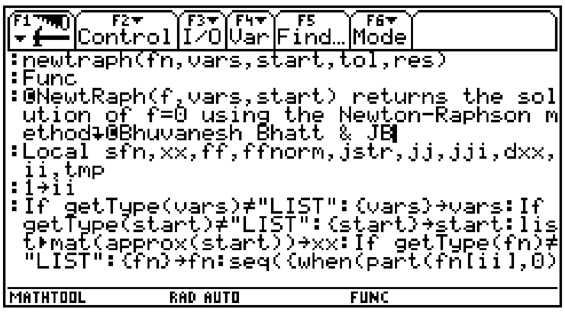


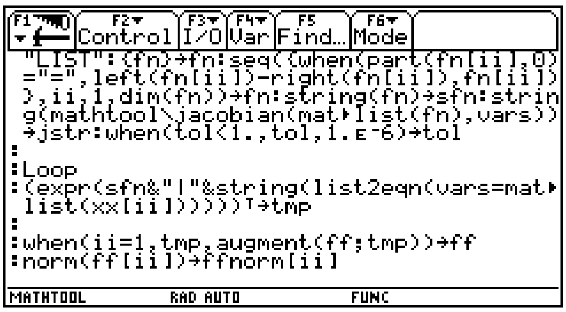


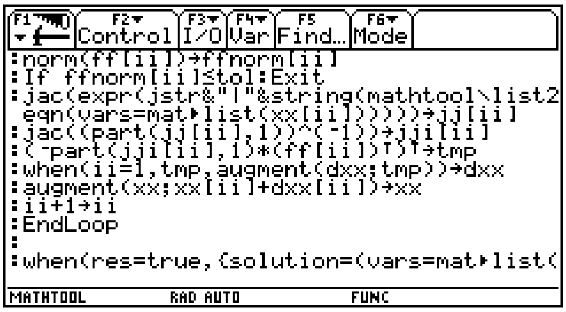


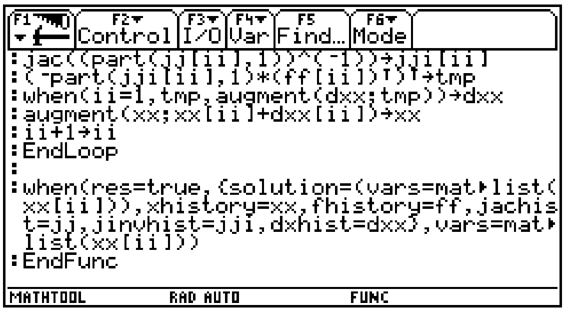
When I tested this function, I found out that it does only work - how intended - for one variable equations. It took some time to discover the problems – most of the errors were dimension errors. The comments included seem to be from earlier attempts to write a program and not a function because Disp does not work within functions.

This is the correct version, which works for systems of equations, too which is demonstrated below.



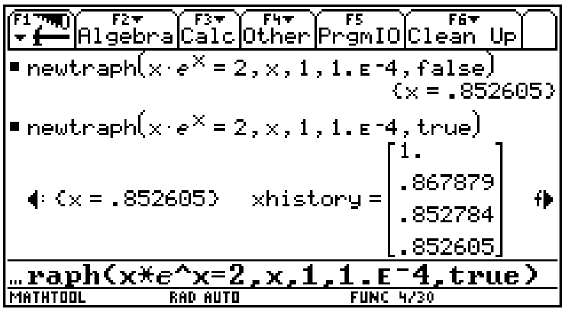


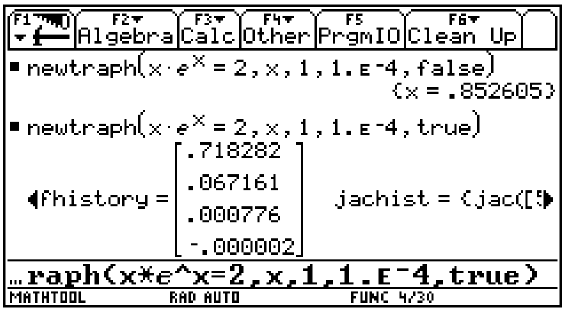


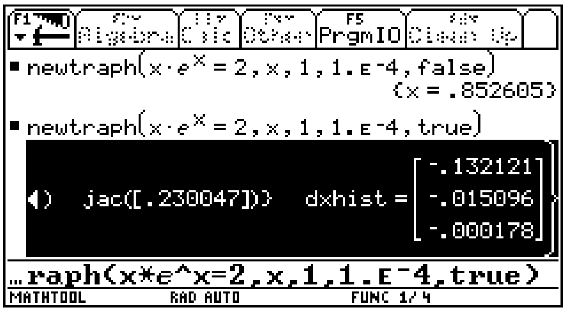


Bhuvanesh used the technique applying the Jacobian matrix.

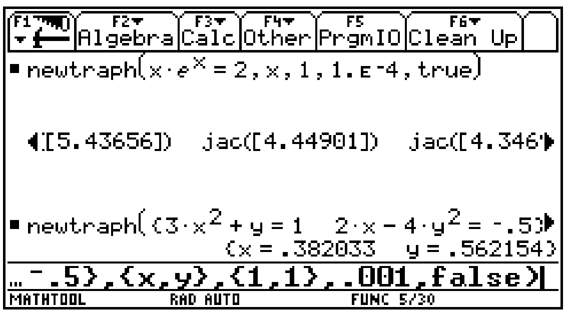
The iteration is given by: $X_{n+1} = X_n - J_F(X_n)^{-1} \cdot F(X_n)$

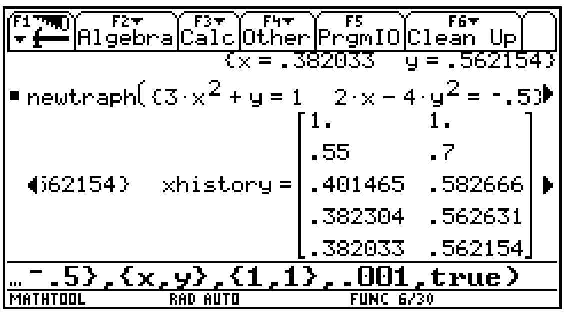


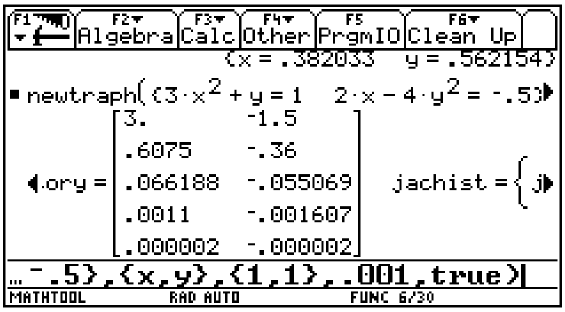


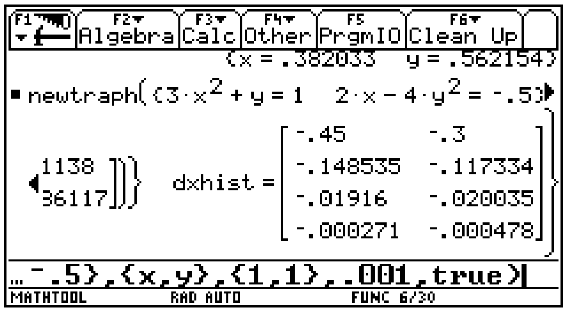


The "intermediate results" are: the iterated values of the unknown, the iterations of the function value (should tend to zero), the iterations of the Jacobian and its inverse and finally the iterated values of the differences between subsequent approximations.



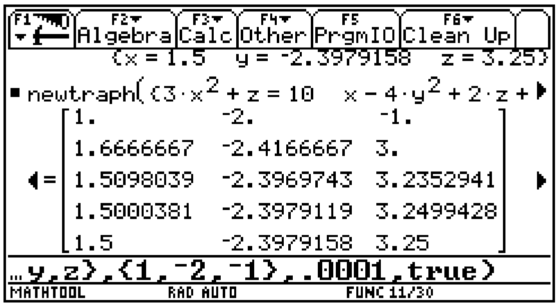






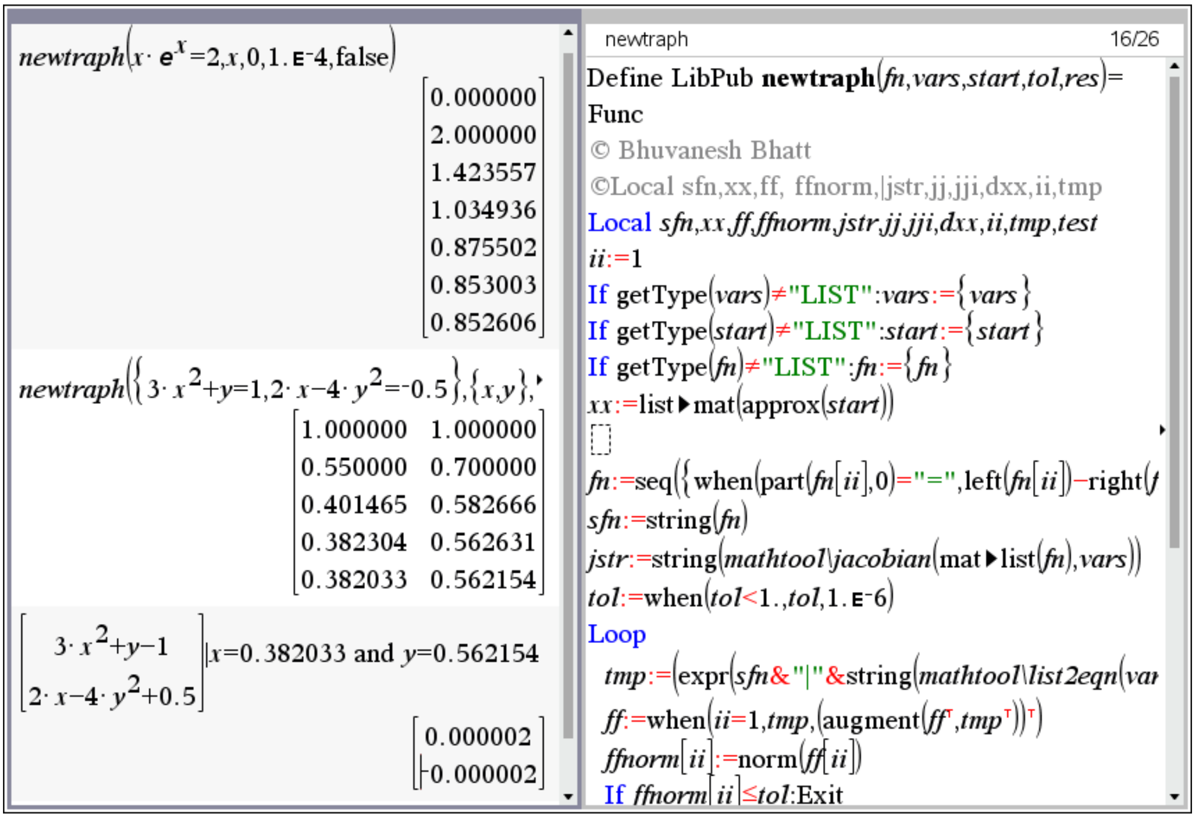
$$\text{NEWTONS}([3\cdot x^2 + z - 10,\ x - 4\cdot y^2 + 2\cdot z + 15,\ x^2 + y^2 - 8],\ [x,\ y,\ z],\ [1,\ -2,\ -1])$$

$$\begin{bmatrix} 1 & -2 & -1 \\ 1.666666666 & -2.416666666 & 3 \\ 1.509803921 & -2.396974306 & 3.235294117 \\ 1.500038147 & -2.397911967 & 3.249942778 \\ 1.5 & -2.397915761 & 3.25 \\ 1.5 & -2.397915761 & 3.25 \end{bmatrix}$$



DERIVE provides an implemented function NEWTONS() which gives the same intermediate values as this function.

This is the Nspire version:



One must try to find suitable initial values for the procedure. It can happen that the function hangs up or that the result reads “Memory Exhausted”

$$newtraph\left(\left\{3\cdot x^2+y=1,2\cdot x-4\cdot y^2=-0.5\right\},\{x,y\},\{1,1\},1.E\text{-}5,true\right)$$

$$\left\{sol=\{x=0.382033,y=0.562154\},xhist=\begin{bmatrix}1.000000 & 1.000000\\ 0.550000 & 0.700000\\ 0.401465 & 0.582666\\ 0.382304 & 0.562631\\ 0.382033 & 0.562154\end{bmatrix},fhist=\begin{bmatrix}3.000000 & -1.500000\\ 0.607500 & -0.360000\\ 0.066188 & -0.055069\\ 0.001100 & -0.001607\\ 0.000002 & -0.000002\end{bmatrix},jachist,\ldots\right.$$

$$newtraph\left(\left\{3\cdot x^2+y=1,2\cdot x-4\cdot y^2=-0.5\right\},\{x,y\},\{1,1\},1.E\text{-}5,true\right)$$

$$\ldots\begin{matrix}48828\\ 161133\end{matrix}\Bigg]\Bigg),jac\left(\begin{bmatrix}0.352379 & 0.075596\\ 0.151193 & -0.182096\end{bmatrix}\right),jac\left(\begin{bmatrix}0.365208 & 0.081138\\ 0.162277 & -0.186117\end{bmatrix}\right)\Bigg\},dxhist=\begin{bmatrix}-0.450000 & -0.300000\\ -0.148535 & -0.117334\\ -0.019160 & -0.020035\\ -0.000271 & -0.000478\end{bmatrix}\Bigg\}$$

$$newtraph\left(\left\{3\cdot x^2+z=10,x-4\cdot y^2+2\cdot z+15=0,x^2+y^2=8\right\},\{x,y,z\},\{1,-2,-1\},1.E\text{-}5,false\right)$$

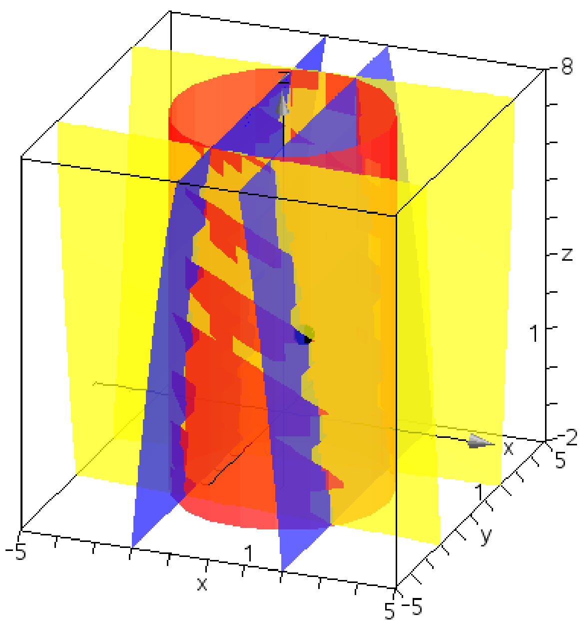
$$\{x=1.500000,y=-2.397916,z=3.250000\}$$

$$newtraph\left(\left\{3\cdot x^2+z=10,x-4\cdot y^2+2\cdot z+15=0,x^2+y^2=8\right\},\{x,y,z\},\{0,-2,-1\},1.E\text{-}5,false\right)$$

$$\{x=-1.000000,y=-2.645751,z=7.000000\}$$

As you can see above, a slight change in the initial values leads to quite another solution.

Finally, I had the idea to plot the three surfaces given in the system of equations above together with one solution (realized as the black sphere).



Mail from Sebastian Rauh

Ein anderer Schüler ist auf ein interessantes Kartenproblem gestoßen: Er teilt einen geordneten Kartenstapel (z.B. 1,2,3,4,5,6,7,8) in zwei Hälften und zieht abwechselnd von beiden Hälften und macht damit einen neuen Stapel (also 1,5,2,6,3,7,4,8) Wenn man den Vorgang *n* mal wiederholt kommt man wieder bei der ursprünglichen Sortierung an. Seine Frage: Gibt es dafür eine Formel? Ich habe es dann programmiert und es gibt die Folge tatsächlich in OEIS.

Another student came across an interesting problem with playing cards: He divides an ordered stack of cards (e.g. 1,2,3,4,5,6,7,8 – must be an even number) in two halves and then draws alternatively one card from both stacks forming a new stack, which gives now 1,5,2,6,3,7,4,8. Repeating this procedure *n* times one reaches again the initial sorting. His question: is there a formula? I wrote a program and the sequence exists indeed in OEIS.

Good question!

The respective article – very extended - can be found at https://oeis.org/A002326 :

Multiplicative order of 2 mod 2n+1

... In other words, least m > 0 such that 2n+1 divides 2^m-1.

Number of riffle shuffles of 2n+2 cards required to return a deck to initial state. A riffle shuffle replaces a list s(1), s(2), ..., s(m) with s(1), s((i/2)+1), s(2), s((i/2)+2), ... a(1) = 2 because a riffle shuffle of [1, 2, 3, 4] requires 2 iterations [1, 2, 3, 4] -> [1, 3, 2, 4] -> [1, 2, 3, 4] to restore the original order.

The first program simulates shuffling a deck of cards with different numbers of cards. It counts the required number of times to rearrange the cards in the original order.

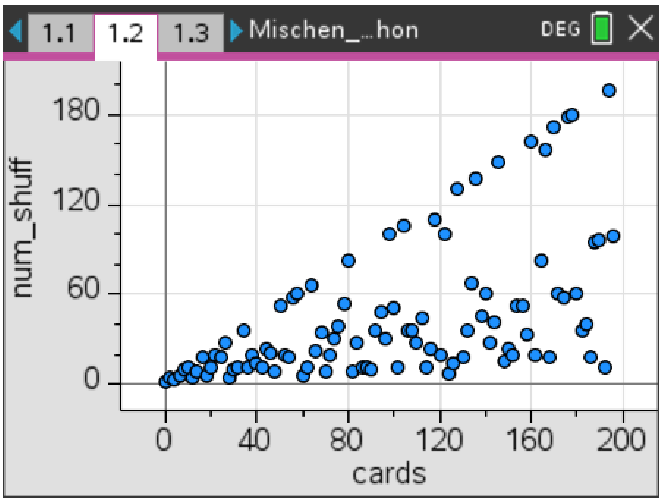
The number of shuffles for 2n and 2n+1 are always equal, therefore only the even cases are considered.

$\text{seqn}(\textbf{no\_shuffles\_exp}(2\cdot n),12) \blacktriangleright \{2,4,3,6,10,12,4,8,18,6,11,20\}$

The second program calculates the needed shuffles directly.

(OEIS, A002326)

$\text{seqn}(\textbf{num\_o\_shuffles}(n),12) \blacktriangleright \{2,4,3,6,10,12,4,8,18,6,11,20\}$



These are the functions needed (TI-Nspire CAS):

```
Define no_shuffles_exp(nn)=
Func
:©Counts the number of required shuffles to restore the original order
:©nn= number of cards in deck
:Local deck,deck_shuff,erg
:erg:=1
:deck:=seqn(n,nn)
:deck_shuff:=shuffle(deck)
:While istungleich(deck,deck_shuff)
:  deck_shuff:=shuffle(deck_shuff)
:  erg:=erg+1
:  If erg=100 Then
:    Return erg
:  EndIf
:EndWhile
:Return erg
:EndFunc
```

```
Define istungleich(l1,l2)=
Func
:©Checks, if two lists are identical
:Local i
:For i,1,dim(l2)
:  If l1[i]≠l2[i] Then
:    Return true
:  EndIf
:EndFor
:Return false
:EndFunc
```

```
Define shuffle(ls)=
Func
:©shuffles a given list ls
:Local erg,i
:erg:={}
:For i,1,floor(((dim(ls))/(2)))
:  erg:=augment(erg,{ls[floor(((dim(ls))/(2)))+i],ls[i]})
:EndFor
:If mod(dim(ls),2)=1 Then
:  erg:=augment(erg,{ls[dim(ls)]})
:EndIf
:Return erg
:EndFunc
```

```
Define num_o_shuffles(n)=
Func
:©From OEIS #A002326; with n=number of cards in deck
:©Least m > 0 such that 2n+1 divides 2^(m)-1
:Local m
:m:=1
:While mod(2^(m)-1,2*n+1)≠0
:  m:=m+1
:EndWhile
:Return m
:EndFunc
```

Sebastian provided a Python-solution, too: Just set the number of cards (Max_Deck_Größe) and switch to Python-Shell pressing Ctrl+R:

```
shu_py.py                                                                16/20
# Math Calculations
#================================
from math import *
#================================
def shuffle(ls):
  return [ls[i] for i in range(len(ls)) if i%2==1]+[ls[i] for i in range(len(ls)) if i%2==0]

def anzahl_der_mischungen(deckgröße):
  startdeck = [i for i in range(deckgröße)]
  aktuelles_deck = shuffle(startdeck)
  z = 1
  while  aktuelles_deck != startdeck:
    aktuelles_deck = shuffle(aktuelles_deck)
    z += 1
  return z

Max_Deck_Größe = 50

print([anzahl_der_mischungen(2*i) for i in range(Max_Deck_Größe)])
```

```
Python-Shell                                                              8/8
>>>#Running shu_py.py
>>>from shu_py import *
[1, 2, 4, 3, 6, 10, 12, 4, 8, 18, 6, 11, 20, 18, 28, 5, 10, 12, 36, 12]
>>>#Running shu_py.py
>>>from shu_py import *
[1, 2, 4, 3, 6, 10, 12, 4, 8, 18, 6, 11, 20, 18, 28, 5, 10, 12, 36, 12, 20, 14, 12, 23, 21, 8, 52, 20, 18, 58, 6
66, 22, 35, 9, 20, 30, 39, 54, 82, 8, 28, 11, 12, 10, 36, 48, 30]
>>>|
```

Another link led to distinguish between **in-shuff**le and **out-shuffle**:
https://mathworld.wolfram.com/RiffleShuffle.html

… The top half of the deck is placed in the left hand, and cards are then alternatively interleaved from the left and right hands (an in-shuffle) or from the right and left hands (an out-shuffle).

This is the DERIVE treatment (this is my turn, Josef):

Inshuffle

```
      Inshuffle(n, cds, newcds, oldcds, j, k) :=
        Prog
          cds := VECTOR(i, i, 1, n)
          "DISPLAY(cds)"
          k := 0
          oldcds := cds
          Loop
            If newcds = cds
               RETURN k
#1:         j := 1
            newcds := []
            Loop
              If j > n/2 exit
              newcds := APPEND(newcds, oldcds↓[j, n/2 + j])
              j :+ 1
            "DISPLAY(newcds)"
            oldcds := newcds
            k :+ 1
```

```
#2:   Inshuffle(10)

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

#3:   6

VECTOR(Inshuffle_o(k), k, 2, 50, 2)

[1, 2, 4, 3, 6, 10, 12, 4, 8, 18, 6, 11, 20, 18, 28, 5, 10, 12, 36, 12, 20, 14, 12, 23, 21]
```

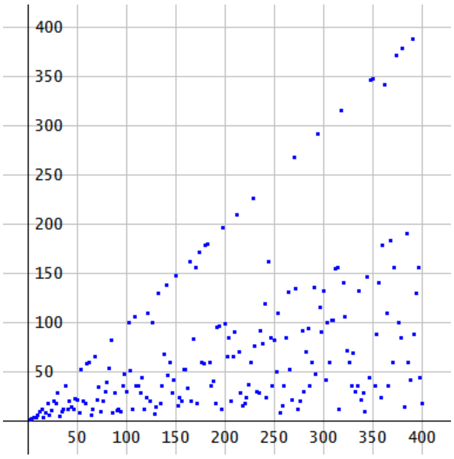
```
Outshuffle_o(n, cds, newcds, oldcds, j, k) :=
  Prog
    cds := VECTOR(i, i, 1, n)
    k := 0
    oldcds := cds
    Loop
      If newcds = cds
         RETURN k
      j := 1
      newcds := []
      Loop
        If j > n/2 exit
        newcds := APPEND(newcds, oldcds↓[n/2 + j, j])
        j :+ 1
      oldcds := newcds
      k :+ 1
```

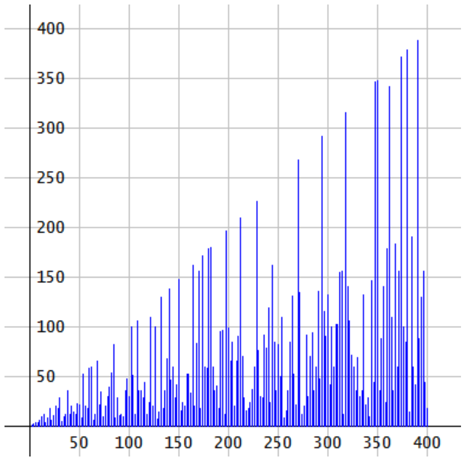
```
Outshuffle(52) = 52

#44:  Outshuffle(10)

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

#45:  10

VECTOR([k, Outshuffle(k)], k, 2, 100, 2)

[2, 4, 3, 6, 10, 12, 4, 8, 18, 6, 11, 20, 18, 28, 5, 10, 12, 36, 12, 20, 14, 12, 23, 21, 8]
```

```
VECTOR([k, Inshuffle(k)], k, 2, 400, 2)
```

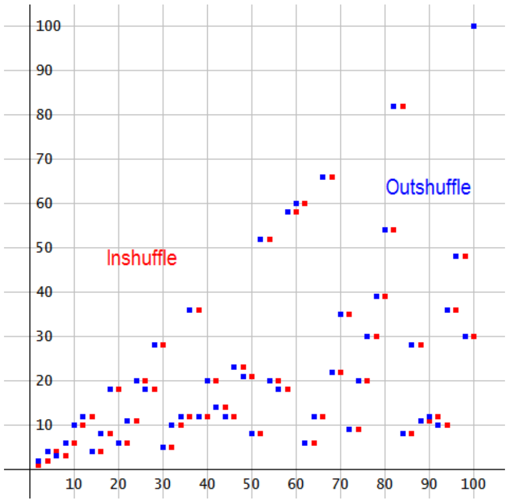
$$\text{VECTOR}\left(\begin{bmatrix} k & 0 \\ k & \text{Inshuffle}(k) \end{bmatrix}, k, 2, 400, 2\right)$$

Two forms of plotting the numbers of necessary in-shuffles (depending on the number of cards):





In-shuffles and Out-shuffles together



And finally, the formula for the sequence of the shuffle numbers (following the OEIS recipe):

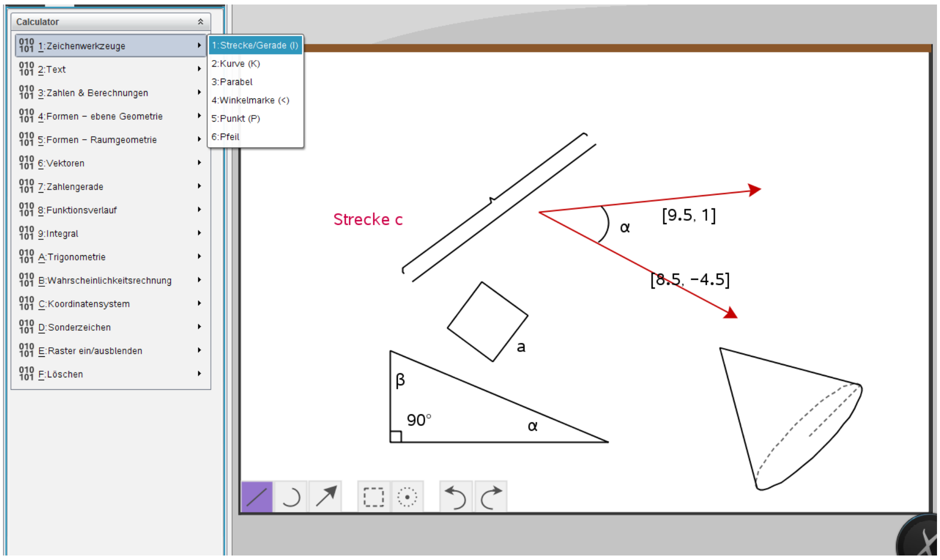
```
inshuffle_f(n, m) :=
  Prog
    m := 1
    Loop
      If FLOOR((2^m - 1)/(2·n - 1)) = (2^m - 1)/(2·n - 1)
         RETURN m
      m :+ 1
```

elements #10 and #20 of the sequence:

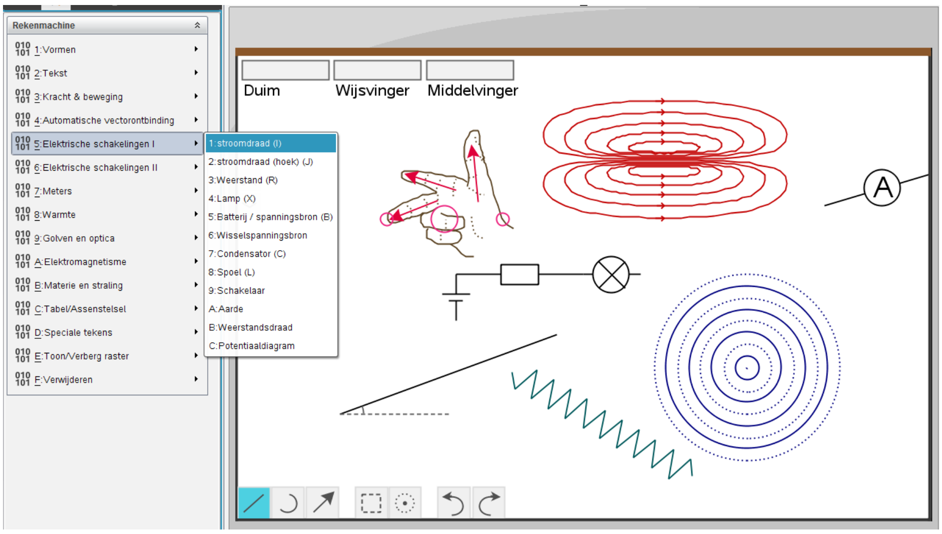
```
inshuffle_f(10) = 18

inshuffle_f(20) = 12

outshuffle_f(n) := inshuffle_f(n + 1)

VECTOR(inshuffle_f(k), k, 25)

[1, 2, 4, 3, 6, 10, 12, 4, 8, 18, 6, 11, 20, 18, 28, 5, 10, 12, 36, 12, 20, 14, 12, 23, 21]

VECTOR(outshuffle_f(k), k, 25)

[2, 4, 3, 6, 10, 12, 4, 8, 18, 6, 11, 20, 18, 28, 5, 10, 12, 36, 12, 20, 14, 12, 23, 21, 8]
```

A team is working on translations in other languages. The localizations should be ready shortly (German, Dutch, French, Danish and Portuguese).

Deutsch



Nederlands



Francais