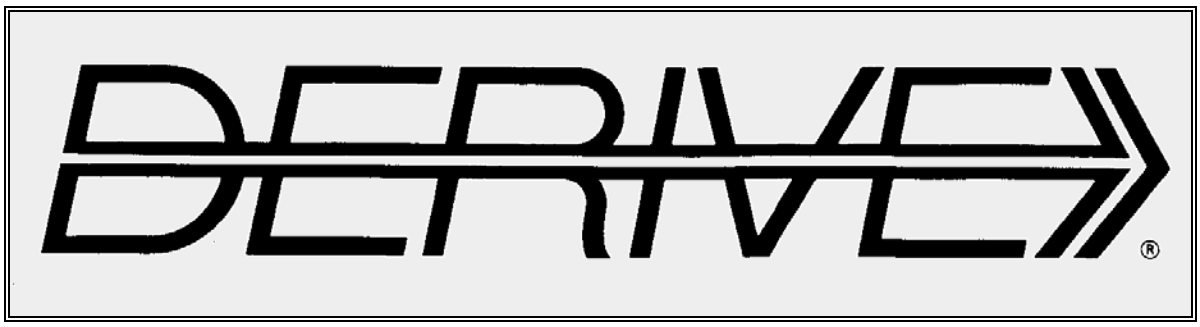
THE DERIVE - NEWSLETTER #106

ISSN 1990-7079

THE BULLETIN OF THE



USER GROUP

+ CAS-TI

Contents:

June 2017

Mail from Joel Rice, who provided a couple of links.
Many thanks to Joel on behalf of the DUG-Members,
Josef

Dear Josef

Great DUG newsletter - wish i had seen it 20 years ago.

Been reading many back issues.

my Derive 1.55 is a bit out of date

but got Macsyma - then they went out of business!

Some math links:

Edwin Woollett - Maxima by example
http://web.csulb.edu/~woollett/

Geometric Algebra / Clifford algebra packages for Matlab or Octave, and maxima
https://gacomputing.info/ga-software/

another Geometric Algebra / Clifford algebra site with links to resources
http://beyhfr.free.fr/EVA2/crbst_2.html

and yet another - with lecture notes by Anthony Lasenby
http://geometry.mrao.cam.ac.uk

Baez-Huerta-Dixon rolling ball demo exceptional group G2
http://www.math.ucr.edu/home/baez/ball/

museum of theorem provers
https://theoremprover-museum.github.io

seems like the Nspire code is more like MuSimp !

Regards
Joel Rice

Speaking of electromagnetism, long ago I saw a nice treatment in
Robert Hermann "Spinors, Clifford Algebras and Cayley numbers"
He unifies div, grad, curl - page 248 ff and Maxwell, complex quaternions and Lorentz.
Too bad it is out of print - but it is online.

https://books.google.com/books?id=IWSGwssSEkYC&pg=PA244&source=gbs_toc_r&cad=3#v=onepage&q&f=false

Regards
Joel Rice

Dear DUG Members,

Fortunately, I am almost in time with DNL#106. We have an extended User Forum containing two interesting requests from Switzerland and a contribution to an earlier article from Roland Schröder. The parameter problem forwarded by Stefanie Hunziker was a challenge for me, too, especially its 3D-solution. Many thanks for the inspiring task.

You will find an article contributed by Michel Beaudin demonstrating a new feature of TI-Nspire CX CAS v. 4.4.

Just recently I received a mail from Marcelo Fernandez from Argentina complaining that there is no function provided in the DERIVE User files for calculating the resultant and the discriminant of polynomials.

I must admit that I had no idea what the resultant might be, and the only discriminant I knew was the one appearing in the solution of a quadratic equation. Then I informed myself and found a lot in the web and other resources. This was enough for producing programs which give the expected results. Marcelo and I had a nice email-communication and finally he provided an article for the next DNL presenting an application of the resultant.

You might remember that I presented some LUA-programs for the TI in earlier DNLs. I mentioned Cahier 35 produced by the Flemish T^3-Group and Steve Arnold's wonderful LUA-Tutorials. I put all these materials together in a German LUA-paper which can now be downloaded from the TI-website (see below).

Many thanks to Michael de Villiers. His Newsletters are always a great source of information and "bonanza" for useful links.

Best regards until September

Download all *DNL*-DERIVE- and TI-files from
http://www.austromath.at/dug/

Einführung in die LUA-Programmierung
http://www.ti-unterrichtsmaterialien.net/fileadmin/DE-Materialien/Materialien/Mathematic/T3_LUA_Programmierung_2017.pdf

Michael de Villiers recommendations:

Michael de Villiers,
A Multiple Solution Task: a SA Mathematics Olympiad Problem
http://dynamicmathematicslearning.com/SAMO-2016-R1Q20.pdf

The New DQME website showcases the work of a 6 year EU project with 11 participating countries. It contains 1000s of practical materials tested and ready for use in the classroom, organised in 10 languages and by age level and mathematical content. To download materials click 'Register Now' on the webpage and Submit your details, after which materials can be accessed when you log in with email/password.
http://www.dqime.uni-dortmund.de/

The Elusive Slope
https://link.springer.com/content/pdf/10.1007%2Fs10763-017-9811-9.pdf

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 3136365
e-mail: nojo.boehm@pgv.at

Contributions:

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

Next issue: September 2017

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
Tools for 3D-Problems, P. Lüke-Rosendahl, GER
Simulating a Graphing Calculator in *DERIVE*, J. Böhm, AUT
Graphics World, Currency Change, P. Charland, CAN
Cubics, Quartics – Interesting features, T. Koller & J. Böhm, AUT
Logos of Companies as an Inspiration for Math Teaching
Exciting Surfaces in the FAZ / Pierre Charland´s Graphics Gallery
BooleanPlots.mth, P. Schofield, UK
Old traditional examples for a CAS – What´s new? J. Böhm, AUT
Mandelbrot and Newton with *DERIVE*, Roman Hašek, CZK
Tutorials for the NSpireCAS, G. Herweyers, BEL
Some Projects with Students, R. Schröder, GER
Dirac Algebra, Clifford Algebra, D. R. Lunsford, USA
A New Approach to Taylor Series, D. Oertel, GER
Henon & Co; Find your very own Strange Attractor, J. Böhm, AUT
Rational Hooks, J. Lechner, AUT
Statistics of Shuffling Cards, H. Ludwig, GER
Charge in a Magnetic Field, H. Ludwig, GER
Factoring Trinomials, D. McDougall, CAN
Selected Lectures from TIME 2016
Resultant & Discriminant for Polynomials, F.M. Fernandez, ARG & J. Böhm, AUT

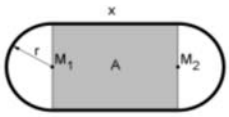
and others

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

I received some mails form a new member from Switzerland. Welcome Stefanie!

First a request came in concerning a problem connected with solving a MaxMin-problem with TI-Nspire

720. Eine ebene 400 m-Bahn soll so angelegt werden, dass sie ein Rechteck mit zwei Halbkreisen begrenzt, wobei die Halbkreise den gegenüberliegenden Rechteckseiten angesetzt sind. Wie gross muss der Kreisradius sein und wie lang ein gerades Stück zwischen den Kurven, wenn das Rechteck maximalen Flächeninhalt haben soll?



The circumference of the playground is a 400m lane. Which x and r make a maximum area of the shaded rectangle?

Given: $u:=400 \blacktriangleright 400$

Goal function: take b instead of x

Area of the rectangle (= Maximum): $a(r):=2\cdot r\cdot b$

Restriction (solved for b):

$b:=\text{right}(\text{solve}(2\cdot b+2\cdot\pi\cdot r=u,b)) \blacktriangleright 200-\pi\cdot r$

Then I receive the area as a function of r ...

$a(r) \blacktriangleright -2\cdot r\cdot(\pi\cdot r-200)$ this is ok!

$ma:=\text{right}(\text{fMax}(a(r),r)) \blacktriangleright \frac{100}{\pi}$ ma is the zero of the 1st derivative, is also ok!

What is the maximum area now?

$a(ma) \blacktriangleright \frac{-200\cdot(\pi\cdot r-200)}{\pi}$ **Why is not substituted 100/π for r?**

$a1(r):=-2\cdot r\cdot(\pi\cdot r-200) \blacktriangleright Done$ $a1(ma) \blacktriangleright \frac{20000}{\pi}$ This is the corrrect maximum area.

Here are my (Josef's) answers:

One possibility to overcome the difficulty might be:

You can help yourself using the |−operator:

$a(r)|r=\frac{100}{\pi} \blacktriangleright \frac{20000}{\pi}$

But this does not really give a satisfying answer.

Look how I am doing now (see next problem):

$u:=400 \blacktriangleright 400$

It is necessary to define b as a function of r.
If you do not, then b is a constant only and it is replaced by 200 − r π.

$b(r):=\text{right}(\text{solve}(2\cdot b\_+2\cdot\pi\cdot r=u,b\_)) \blacktriangleright \textit{Done} \qquad b(r) \blacktriangleright 200-\pi\cdot r$

You need a dummy variable b_ (don't use b(r) instead − gives an error message).

$a(r):=2\cdot r\cdot b(r) \blacktriangleright \textit{Done}$

$ma:=\text{right}(\text{fMax}(a(r),r)) \blacktriangleright \frac{100}{\pi}$

Now it works as expected!

$a(ma) \blacktriangleright \frac{20000}{\pi}$

Solutions: r = 100/π, b = x = 100, maximum area = 20 000/π

The DERIVE-treatment is following and it shows the pretty same behavior:

#1: $u := 400$

#2: $a(r) := 2\cdot r\cdot b$

#3: $(b := (\text{SOLUTIONS}(2\cdot b + 2\cdot\pi\cdot r = u,\ b))_1) = b := 200 - \pi\cdot r$

#4: $a(r) = 2\cdot r\cdot(200 - \pi\cdot r)$

#5: $\left(ma := \left(\text{SOLUTIONS}\left(\frac{d}{dr}\,a(r) = 0,\ r\right)\right)_1\right) = ma := \frac{100}{\pi}$

#6: $a(ma) = \frac{200\cdot(200 - \pi\cdot r)}{\pi}$

SUBST instead of applying the |-operator:

#7: $\text{SUBST}(a(r),\ r,\ ma) = \frac{20000}{\pi}$

Introducing b as function of r:

#8: $(b(r) := (\text{SOLUTIONS}(2\cdot b\_ + 2\cdot\pi\cdot r = u,\ b\_))_1) = b(r) := 200 - \pi\cdot r$

#9: $a(r) := 2\cdot r\cdot b(r)$

#10: $\left(ma := \left(\text{SOLUTIONS}\left(\frac{d}{dr}\,a(r) = 0,\ r\right)\right)_1\right) = ma := \frac{100}{\pi}$

#11: $a(ma) = \frac{20000}{\pi}$

Let's remember a DERIVE contribution from 1993:

LOAD(D:\DfD\DNL\DNL93\MTH12\MaxMin_ut.mth)

MaxMin.dfw

[a := 2·r·b, s := 400 − 2·r·π − 2·b]

$$\text{EXTREM}(a, s, r, b) = \begin{bmatrix} r = \frac{100}{\pi} & b = 100 \\ \text{extr} = \frac{20000}{\pi} & \text{Maximum} \\ 2\cdot r\cdot(200 - \pi\cdot r) & 4\cdot(100 - \pi\cdot r) \end{bmatrix}$$

Stefanie wrote: And there is another problem working with vectors (see attached tns-file):

Given are straight line g and points A and B

$g: \vec{r} = \begin{pmatrix}1\\1\\2\end{pmatrix} + t\begin{pmatrix}1\\2\\0\end{pmatrix}$, $A = \begin{pmatrix}7\\5\\2\end{pmatrix}$, $B = \begin{pmatrix}-2\\4\\6\end{pmatrix}$. Find points $P \in g$ with $\measuredangle APB = 40°$.

$\mathbf{g}(t) := \begin{bmatrix}1\\1\\2\end{bmatrix} + t\cdot\begin{bmatrix}1\\2\\0\end{bmatrix}$ ▸ *Done* $\quad \mathbf{a} := \begin{bmatrix}7\\5\\2\end{bmatrix}$ ▸ $\begin{bmatrix}7\\5\\2\end{bmatrix}$ $\quad \mathbf{b} := \begin{bmatrix}-2\\4\\6\end{bmatrix}$ ▸ $\begin{bmatrix}-2\\4\\6\end{bmatrix}$

$\mathbf{pa}(t) := \mathbf{a} - \mathbf{g}(t)$ ▸ *Done* $\quad \mathbf{pa}(t)$ ▸ $\begin{bmatrix}6-t\\4-2\cdot t\\0\end{bmatrix}$ $\quad \mathbf{pb}(t) := \mathbf{b} - \mathbf{g}(t)$ ▸ *Done* $\quad \mathbf{pb}(t)$ ▸ $\begin{bmatrix}-t-3\\3-2\cdot t\\4\end{bmatrix}$

$\text{solve}\left(\frac{\text{dotP}(\mathbf{pa}(t),\mathbf{pb}(t))}{\text{norm}(\mathbf{pa}(t))\cdot\text{norm}(\mathbf{pb}(t))} = \cos(40), t\right)$ ▸ $t = -3.89684$ or $t = 6.8788$ ⚠

$\text{solve}\left(\left\{\frac{\text{dotP}(\mathbf{pa}(t),\mathbf{pb}(t))}{\text{norm}(\mathbf{pa}(t))\cdot\text{norm}(\mathbf{pb}(t))} = \cos(40),\ p(t) = \mathbf{g}(t)\right\}, t, p(t)\right)$ ▸ Error: Argument must be a variable name.

I would like to find the solution points in one single step, i.e. solving the equation for t and then presenting all possible $g(t)$.

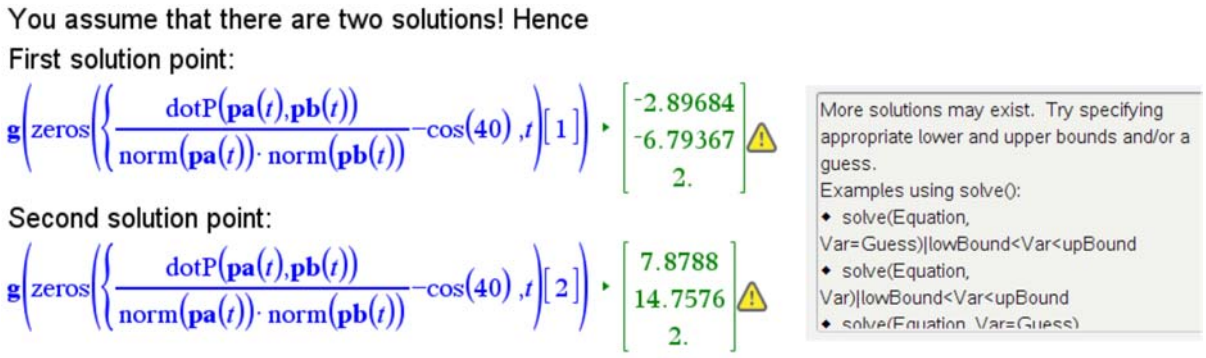
I can do it with zeros – but it needs also two steps!

$\mathbf{tt} := \text{zeros}\left(\frac{\text{dotP}(\mathbf{pa}(t),\mathbf{pb}(t))}{\text{norm}(\mathbf{pa}(t))\cdot\text{norm}(\mathbf{pb}(t))} - \cos(40), t\right)$ ▸ $\{-3.89684, 6.8788\}$ ⚠

$\mathbf{g}(\mathbf{tt}[1])$ ▸ $\begin{bmatrix}-2.89684\\-6.79367\\2.\end{bmatrix}$ $\quad \mathbf{g}(\mathbf{tt}[2])$ ▸ $\begin{bmatrix}7.8788\\14.7576\\2.\end{bmatrix}$

In the last months, I worked through a lot of parameter problems because they form an important part in our end examination. I was able to solve all respective problems (trigonometry, functions, 3D-problems, …) using one single system of equations. I'd like to apply this concept in vector problems, too, but I failed …

My (Josef's) answer:



But, you can recognize the yellow warning triangle. There might be more solutions.

I asked DERIVE:

```
#1:  CaseMode := Sensitive

#2:  [g(t) := [1, 1, 2] + t·[1, 2, 0], A := [7, 5, 2], B := [-2, 4, 6]]

#3:  [pa := A - g(t), pb := B - g(t)]

#4:  pts := VECTOR(g(t), t, SOLUTIONS(pa·pb / (|pa|·|pb|) = COS(40°), t))

#5:  pts :=
      [  3.074257854   5.148515708  2 ]
      [  2.743774516   4.487549032  2 ]
      [ -2.896836027  -6.793672054  2 ]
      [  7.878803657   14.75760731  2 ]

#6:  [P1 := pts , P2 := pts , P3 := pts , P4 := pts ]
              1          2          3          4

#7:  VECTOR(ARCCOS((A - X)·(B - X) / (|A - X|·|B - X|)), X, pts)

#8:  [140, 140, 40, 40]
```

And, indeed, four solutions are given – in one step as wanted. I doublecheck the angle and find out that two points (the two ones given by TI-Nspire) make an angle of 40°, the other two make 140°.

But, why are there four solutions given by DERIVE? Can we find them with Nspire, too?

Now let's do it in Exact Mode:

This leads to an order 4 equation!

$$\text{solve}\left(\frac{\text{dotP}(\mathbf{pa}(t),\mathbf{pb}(t))}{\text{norm}(\mathbf{pa}(t))\cdot\text{norm}(\mathbf{pb}(t))}=\cos(40),t\right) \blacktriangleright \frac{5\cdot t^2-17\cdot t-6}{\sqrt{(5\cdot t^2-28\cdot t+52)\cdot(5\cdot t^2-6\cdot t+34)}}=\cos(40)$$

$$\text{nSolve}\left(\frac{\text{dotP}(\mathbf{pa}(t),\mathbf{pb}(t))}{\text{norm}(\mathbf{pa}(t))\cdot\text{norm}(\mathbf{pb}(t))}=\cos(40),t\right) \blacktriangleright -3.89684$$

$$\text{nSolve}\left(\frac{\text{dotP}(\mathbf{pa}(t),\mathbf{pb}(t))}{\text{norm}(\mathbf{pa}(t))\cdot\text{norm}(\mathbf{pb}(t))}=\cos(40),t\right)\,|\,t>-3.89 \blacktriangleright 6.8788$$

We square the equation to eliminate the square root

$$\left(\frac{5\cdot t^2-17\cdot t-6}{\sqrt{(5\cdot t^2-28\cdot t+52)\cdot(5\cdot t^2-6\cdot t+34)}}\right)^2=(\cos(40))^2$$

$$\blacktriangleright \frac{(5\cdot t^2-17\cdot t-6)^2}{25\cdot t^4-170\cdot t^3+598\cdot t^2-1264\cdot t+1768}=(\cos(40))^2$$

$$25\cdot t^4-170\cdot t^3+598\cdot t^2-1264\cdot t+1768$$

We find the solutions step by step:

$$\text{nSolve}\left(\frac{(5\cdot t^2-17\cdot t-6)^2}{25\cdot t^4-170\cdot t^3+598\cdot t^2-1264\cdot t+1768}=(\cos(40))^2,t\right) \blacktriangleright 1.74377$$

$$\text{nSolve}\left(\frac{(5\cdot t^2-17\cdot t-6)^2}{25\cdot t^4-170\cdot t^3+598\cdot t^2-1264\cdot t+1768}=(\cos(40))^2,t\right)\,|\,t>1.75 \blacktriangleright 2.07426$$

$$\text{nSolve}\left(\frac{(5\cdot t^2-17\cdot t-6)^2}{25\cdot t^4-170\cdot t^3+598\cdot t^2-1264\cdot t+1768}=(\cos(40))^2,t\right)\,|\,t>2.075 \blacktriangleright 6.8788$$

$$\text{nSolve}\left(\frac{(5\cdot t^2-17\cdot t-6)^2}{25\cdot t^4-170\cdot t^3+598\cdot t^2-1264\cdot t+1768}=(\cos(40))^2,t\right)\,|\,t<1.74 \blacktriangleright -3.89684$$

The t-values give the four points. These are the missing two:

$$\mathbf{g}(1.74377) \blacktriangleright \begin{bmatrix}2.74377\\4.48754\\2.\end{bmatrix} \qquad \mathbf{g}(6.8788) \blacktriangleright \begin{bmatrix}7.8788\\14.7576\\2.\end{bmatrix}$$

This was my first answer to Stefanie Hunziker.

The equation of order four inspired me to add some investigation initialized by the question for the locus of all points from where the segment AB is seen under a fixed angle (e.g. 40°)?

It should not be too difficult to follow the TI-Nspire procedure explained in the Notes:

For easier representation of the 3D–locus I choose points A and B on the x–axis.

Given are line g [(–7,–10,–10), (–6,–9,–9)] and points A(0,0,0) and B(6,0,0).
Find those points on g from which one can see segment AB under an angle of 40°.

First of all we will investigate this problem in the xy–plane and find the locus of these points.
According to the circle theorem this is a circle containing points A2 and B2. We calculate the coordinates of its centre and its radius. Then we can plot it on the next page.

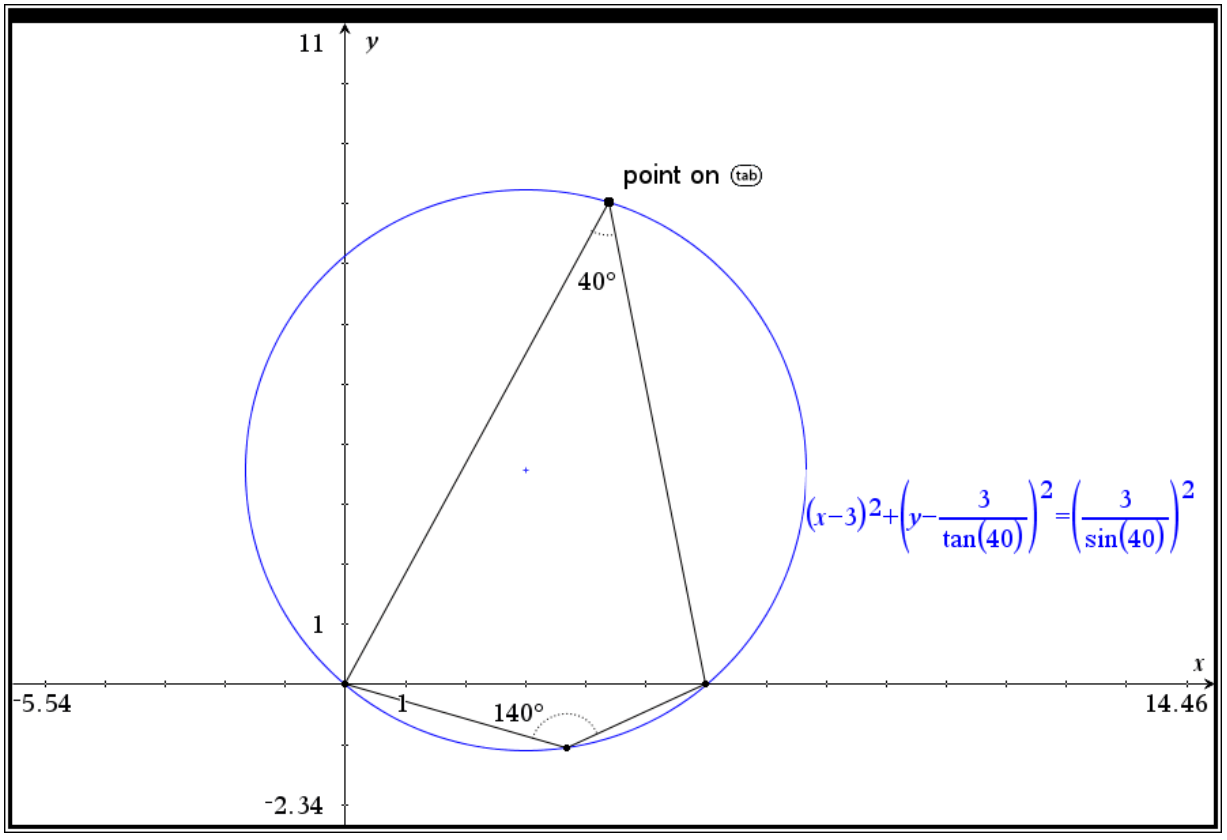
$a2:=[0 \quad 0] \blacktriangleright [0 \quad 0]$ $b2:=[6 \quad 0] \blacktriangleright [6 \quad 0]$ $\varphi:=40 \blacktriangleright 40$

$m2:=\left[3 \quad \frac{3}{\tan(\varphi)}\right] \blacktriangleright \left[3 \quad \frac{3}{\tan(40)}\right]$ $r:=\frac{3}{\sin(\varphi)} \blacktriangleright \frac{3}{\sin(40)}$

This is the parameter representation of the boundary angle circle (in German: Peripheriewinkelkreis pwk) in the plane:

$pwk3:=[m2[1,1]+r\cdot\cos(t) \quad m2[1,2]+r\cdot\sin(t)]$

Let's represent the situation on a Geometry Page:



$\mathbf{g}(t):=\begin{bmatrix}-5\\-10\\-9\end{bmatrix}+t\cdot\begin{bmatrix}1\\1\\1\end{bmatrix}$ ▸ *Done* $\quad \mathbf{a}:=\begin{bmatrix}0\\0\\0\end{bmatrix}$ ▸ $\begin{bmatrix}0\\0\\0\end{bmatrix}$ $\quad \mathbf{b}:=\begin{bmatrix}6\\0\\0\end{bmatrix}$ ▸ $\begin{bmatrix}6\\0\\0\end{bmatrix}$

$\mathbf{pa}(t):=\mathbf{a}-\mathbf{g}(t)$ ▸ *Done* $\quad \mathbf{pb}(t):=\mathbf{b}-\mathbf{g}(t)$ ▸ *Done*

We define f1(x) in order to find the solutions:

$$\mathbf{f1}(x):=\frac{\text{dotP}(\mathbf{pa}(x),\mathbf{pb}(x))}{\text{norm}(\mathbf{pa}(x))\cdot\text{norm}(\mathbf{pb}(x))}-\cos(40)$$ ▸ *Done*

$\text{solve}(\mathbf{f1}(x)=0,x)$ ▸ $x=4.67899$ or $x=12.5891$ ⚠

Again we obtain two parameter values. Let's go on in the same way as above:

$$\frac{\text{dotP}(\mathbf{pa}(x),\mathbf{pb}(x))}{\text{norm}(\mathbf{pa}(x))\cdot\text{norm}(\mathbf{pb}(x))}=\cos(40)$$ ▸ $$\frac{3\cdot x^2-54\cdot x+236}{\sqrt{(3\cdot x^2-60\cdot x+302)\cdot(3\cdot x^2-48\cdot x+206)}}=\cos(40)$$

$$\frac{\text{dotP}(\mathbf{pa}(x),\mathbf{pb}(x))}{\text{norm}(\mathbf{pa}(x))\cdot\text{norm}(\mathbf{pb}(x))}=\cos(40)$$ ▸ $$\frac{3\cdot x^2-54\cdot x+236}{\sqrt{(3\cdot x^2-60\cdot x+302)\cdot(3\cdot x^2-48\cdot x+206)}}=\cos(40)$$

$$\mathbf{f2}(x):=\left(\frac{3\cdot x^2-54\cdot x+236}{\sqrt{(3\cdot x^2-60\cdot x+302)\cdot(3\cdot x^2-48\cdot x+206)}}\right)^2-(\cos(40))^2$$ ▸ *Done*

$\text{nSolve}(\mathbf{f2}(x)=0,x)|x>4.68$ ▸ 9.02148 ⚠

$\text{nSolve}(\mathbf{f2}(x)=0,x)|x>9.02148$ ▸ 9.71045 ⚠

$\text{nSolve}(\mathbf{f2}(x)=0,x)|x>9.72$ ▸ 12.5891 ⚠

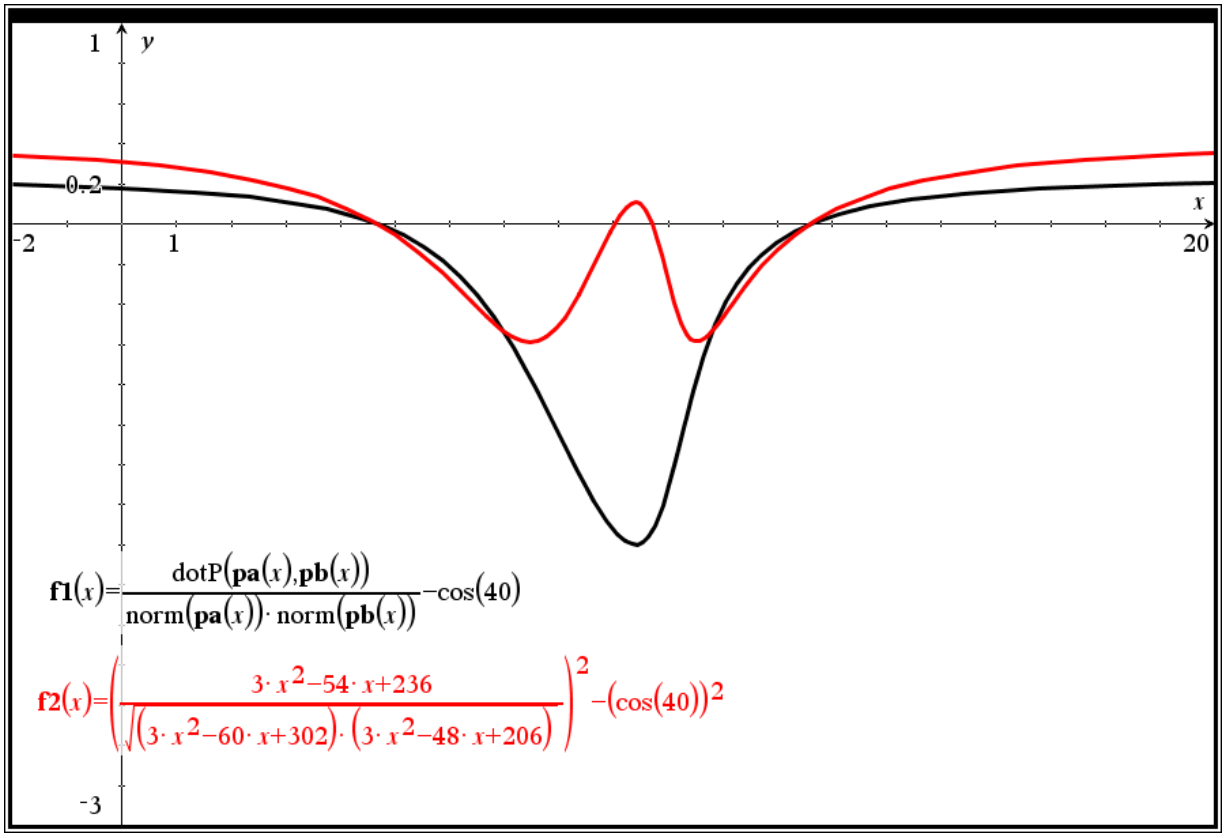
Now we have the parameter values for all possible solution points.

We plot f1(x) and f2(x) on the next page

We check which points give the angle 40°:

$$\mathbf{ang}(x):=\cos^{-1}\left(\frac{\text{dotP}(\mathbf{pa}(x),\mathbf{pb}(x))}{\text{norm}(\mathbf{pa}(x))\cdot\text{norm}(\mathbf{pb}(x))}\right)$$ ▸ *Done*

$\mathbf{ang}(4.67899)$ ▸ $40.$ $\quad \mathbf{ang}(9.0248)$ ▸ 140.092 $\quad \mathbf{ang}(9.71045)$ ▸ $140.$ $\quad \mathbf{ang}(12.5891)$ ▸ 39.9998



Now let's go into the 3rd dimension: rotating the circle around AB gives a torus.

The intersection points of the line and the torus are:

$$p1:=g(4.67899)\rightarrow\begin{bmatrix}-0.32101\\-5.32101\\-4.32101\end{bmatrix}\quad p2:=g(12.5891)\rightarrow\begin{bmatrix}7.5891\\2.5891\\3.5891\end{bmatrix}$$

$$p3:=g(9.02148)\rightarrow\begin{bmatrix}4.02148\\-0.97852\\0.02148\end{bmatrix}\quad p4:=g(9.71045)\rightarrow\begin{bmatrix}4.71045\\-0.28955\\0.71045\end{bmatrix}$$

From P1 and P2 we see AB under 40°, from P3 and P4 unter 140°.

The points are represented in the 3D Plot as small spheres:

$$ku(mi,ra):=\{ra\cdot\cos(t)\cdot\sin(u)+mi[1,1],ra\cdot\sin(t)\cdot\sin(u)+mi[2,1],ra\cdot\cos(u)+mi[3,1]\}\rightarrow Done$$

$$pta:=ku(a,0.2)\rightarrow\{0.2\cdot\cos(t)\cdot\sin(u),0.2\cdot\sin(t)\cdot\sin(u),0.2\cdot\cos(u)\}$$

$$ptb:=ku(b,0.2)\rightarrow\{0.2\cdot\cos(t)\cdot\sin(u)+6,0.2\cdot\sin(t)\cdot\sin(u),0.2\cdot\cos(u)\}$$

$$ptp1:=ku(p1,0.2)\rightarrow\{0.2\cdot\cos(t)\cdot\sin(u)-0.32101,0.2\cdot\sin(t)\cdot\sin(u)-5.32101,0.2\cdot\cos(u)-4.32101\}$$

$$ptp2:=ku(p2,0.2)\rightarrow\{0.2\cdot\cos(t)\cdot\sin(u)+7.5891,0.2\cdot\sin(t)\cdot\sin(u)+2.5891,0.2\cdot\cos(u)+3.5891\}$$

$$ptp3:=ku(p3,0.2)\rightarrow\{0.2\cdot\cos(t)\cdot\sin(u)+4.02148,0.2\cdot\sin(t)\cdot\sin(u)-0.97852,0.2\cdot\cos(u)+0.02148\}$$

$$ptp4:=ku(p4,0.2)\rightarrow\{0.2\cdot\cos(t)\cdot\sin(u)+4.71045,0.2\cdot\sin(t)\cdot\sin(u)-0.28955,0.2\cdot\cos(u)+0.71045\}$$

The 3D representation of the circle pwk is given by:

$xp2(t,u):=m2[1,1]+r\cdot\cos(t)$ ▸ *Done* $yp2(t,u):=m2[1,2]+r\cdot\sin(t)$ ▸ *Done* $zp2(t,u):=0\cdot t$ ▸ *Done*

Circle pwk rotates around the chord AB resulting in a spindle torus.

$xp3(t,u):=xp2(t,u)$ ▸ *Done* $yp3(t,u):=yp2(t,u)\cdot\cos(u)$ ▸ *Done* $zp3(t,u):=yp2(t,u)\cdot\sin(u)$ ▸ *Done*

And these are the rays of the angles (40° red, 140° blue); with 0 ≤ t ≤ 1

$s1:=mat\blacktriangleright list(a+t\cdot(p1-a))$ ▸ $\{-0.32101\cdot t,-5.32101\cdot t,-4.32101\cdot t\}$

$s2:=mat\blacktriangleright list(b+t\cdot(p1-b))$ ▸ $\{6-6.32101\cdot t,-5.32101\cdot t,-4.32101\cdot t\}$

$s3:=mat\blacktriangleright list(a+t\cdot(p2-a))$ ▸ $\{7.5891\cdot t,2.5891\cdot t,3.5891\cdot t\}$

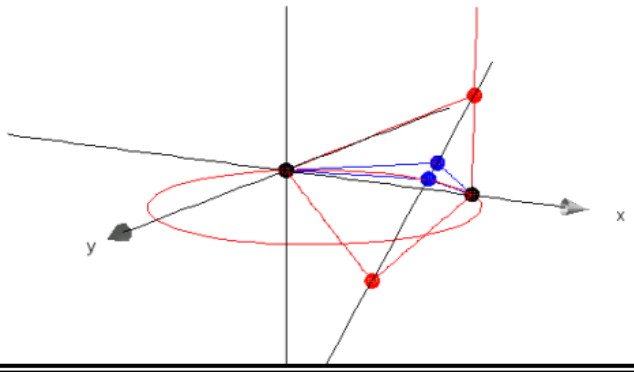
$s4:=mat\blacktriangleright list(b+t\cdot(p2-b))$ ▸ $\{1.5891\cdot t+6,2.5891\cdot t,3.5891\cdot t\}$

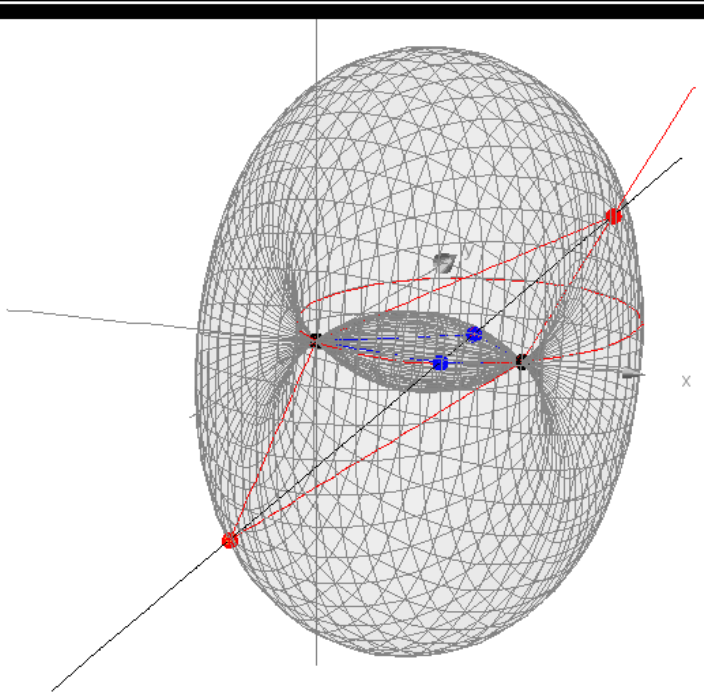
$s5:=mat\blacktriangleright list(a+t\cdot(p3-a))$ ▸ $\{4.02148\cdot t,-0.97852\cdot t,0.02148\cdot t\}$

$s6:=mat\blacktriangleright list(b+t\cdot(p3-b))$ ▸ $\{6-1.97852\cdot t,-0.97852\cdot t,0.02148\cdot t\}$

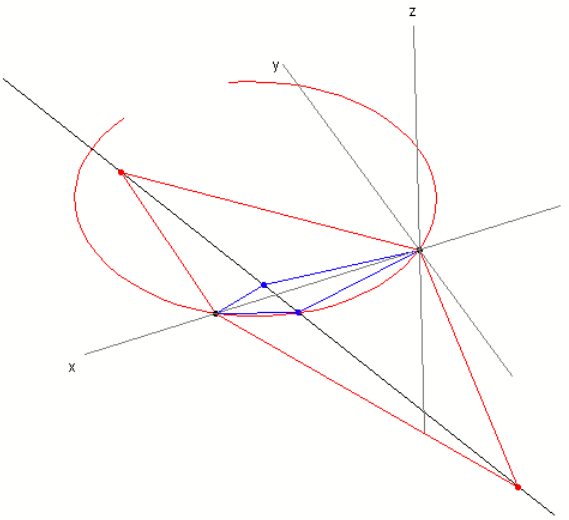
$s7:=mat\blacktriangleright list(a+t\cdot(p4-a))$ ▸ $\{4.71045\cdot t,-0.28955\cdot t,0.71045\cdot t\}$

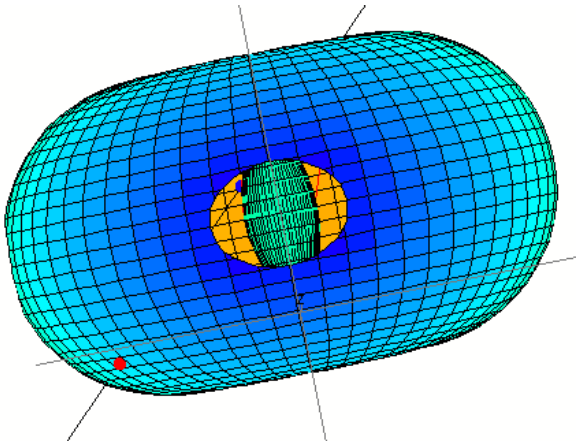
$s8:=mat\blacktriangleright list(b+t\cdot(p4-b))$ ▸ $\{6-1.28955\cdot t,-0.28955\cdot t,0.71045\cdot t\}$

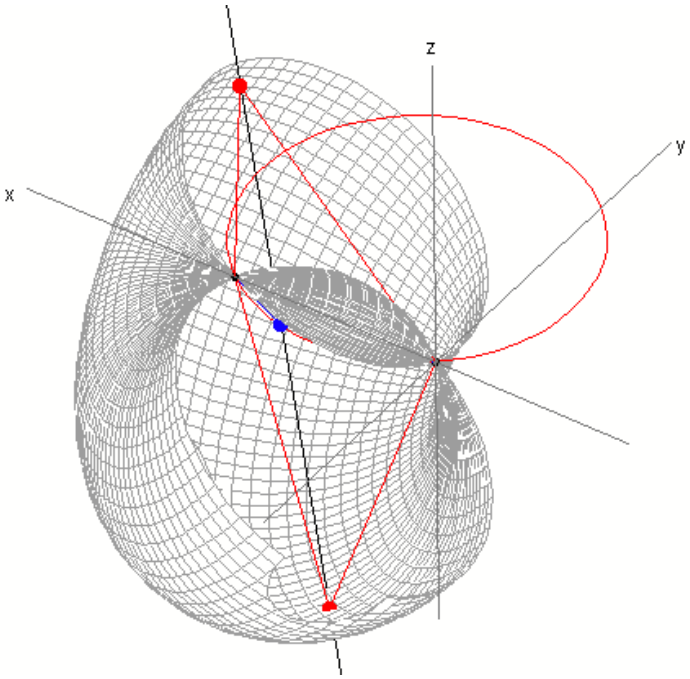




These are the respective DERIVE plots:







Mail from Nicolas Rosillo , nicolasrosillo@gmail.com

Addition to DNL 104 Schröder contribution

Hi Joseph

If I is defined as (f2(a),f1(a)), then ABDEFGHI are in the same circle $(x - 0.5)^2 + (y - 0.5)^2 = 0.5$

AGDH and BEIF form a unit square

AEDF and GBHI form rectangles with sides parallel to the axes

GEHF and BAID form rectangles with sides parallel to y = x and y = -x

I attach you a Derive file with some proofs.

Greetings from Spain

#1: $\left[f1 := \frac{x^2 + x}{x^2 + 1},\ f2 := \frac{x^2 - x}{x^2 + 1},\ f3 := \frac{1 + x}{x^2 + 1},\ f4 := \frac{1 - x}{x^2 + 1} \right]$

#2: [A := [f1, f2], B := [f3, f4], D := [f4, f3], E := [f1, f3]]

#3: [F := [f4, f2], G := [f2, f4], H := [f3, f1], I := [f2, f1]]

ABDEFGHI are on a circle

#4: $OnCircle(M) := (M_1 - 0.5)^2 + (M_2 - 0.5)^2 = 0.5$

#5: [OnCircle(A), OnCircle(B), OnCircle(D), OnCircle(E)] = [true, true, true, true]

#6: [OnCircle(F), OnCircle(G), OnCircle(H), OnCircle(I)] = [true, true, true, true]

GEHF is a rectangle with edges paralel to y=x and y=-x

#7: [(E - G)·(H - E), (H - E)·(F - H), (F - H)·(G - F), (G - F)·(E - G)] = [0, 0, 0, 0]

#8: $[|E - G|, |H - F|, |F - G|, |H - E|] = \left[\frac{2 \cdot \sqrt{2} \cdot |x|}{x^2 + 1}, \frac{2 \cdot \sqrt{2} \cdot |x|}{x^2 + 1}, \frac{\sqrt{2} \cdot |x^2 - 1|}{x^2 + 1}, \frac{\sqrt{2} \cdot |x^2 - 1|}{x^2 + 1} \right]$

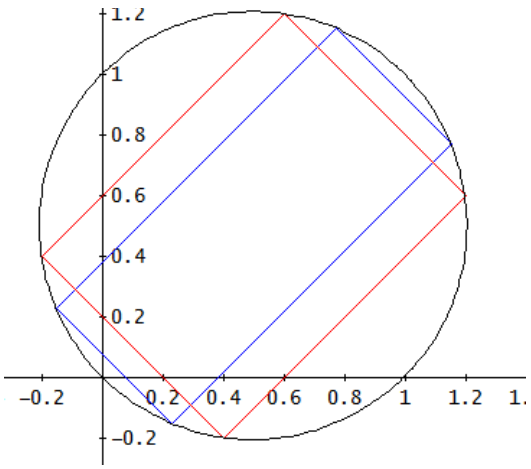
#9: $\left[\frac{E_2 - G_2}{E_1 - G_1}, \frac{H_2 - F_2}{H_1 - F_1}, \frac{F_2 - G_2}{F_1 - G_1}, \frac{H_2 - E_2}{H_1 - E_1} \right] = [1, 1, -1, -1]$

We plot two rectangles and the circle:

#10: $(x - 0.5)^2 + (y - 0.5)^2 = 0.5$

#11: SUBST([G, E, H, F, G], x, 1.5)

#12: SUBST([G, E, H, F, G], x, 0.5)



Roots of Unity and Polynomial Equations Plotting

Let n be a positive integer and let w be a given fixed complex number. How do we solve the equation $z^n = w$. It is easy: we use the polar form and write $w = R\cdot e^{i\cdot\varphi}$ where R is the module of w and where φ is the argument of w, arg(w). The convention is $-\pi < \arg(w) \le \pi$. In Nspire, arg is angle: for example

$|-3+2\cdot i| \blacktriangleright \sqrt{13}$ $\quad$ $\text{angle}(3-3\cdot i) \blacktriangleright \frac{-\pi}{4}$

Now we set $z = r\cdot e^{i\cdot\theta}$. Then, according to De Moivre formula, we have

$r^n\cdot e^{i\cdot n\cdot\theta} = R\cdot e^{i\cdot\Phi}$ and it follows

$r = \sqrt[n]{R}$ and $\theta = \frac{\varphi + 2\cdot k\cdot\pi}{n}$, $n = 0, 1, 2, \ldots, n-1$.

Now, let's take $w = 1$. The the (n complex) solutions of the equation $z^n = 1$ are called the roots of unity, a trivial solution being 1. They can be easily found using de De Moivre formula and it is well known they are given by $e^{2\cdot k\cdot\pi\cdot i/n}$ with $k = 0, 1, 2, \ldots, n-1$.

See page 2 where the csolve command of Nspire is used. This command seems to use the above procedure.

Example: the 3 cubic roots of unity can easily be found by hand because we simply need to divide z^3-1 by $z-1$ and the 2 complex conjugates root will be found.

$\frac{z^3-1}{z-1} \blacktriangleright z^2+z+1$

$\text{cZeros}(z^2+z+1,z) \blacktriangleright \left\{\frac{-1}{2}+\frac{\sqrt{3}}{2}\cdot i, \frac{-1}{2}-\frac{\sqrt{3}}{2}\cdot i\right\}$

The csolve command yieds directly the result:

$\text{cSolve}(z^3=1,z) \blacktriangleright z=\frac{-1}{2}+\frac{\sqrt{3}}{2}\cdot i$ or $z=\frac{-1}{2}-\frac{\sqrt{3}}{2}\cdot i$ or $z=1$

$(\text{cSolve}(z^3=1,z))\blacktriangleright\text{Decimal} \blacktriangleright z=-0.5+0.866025\cdot i$ or $z=-0.5-0.866025\cdot i$ or $z=1.$

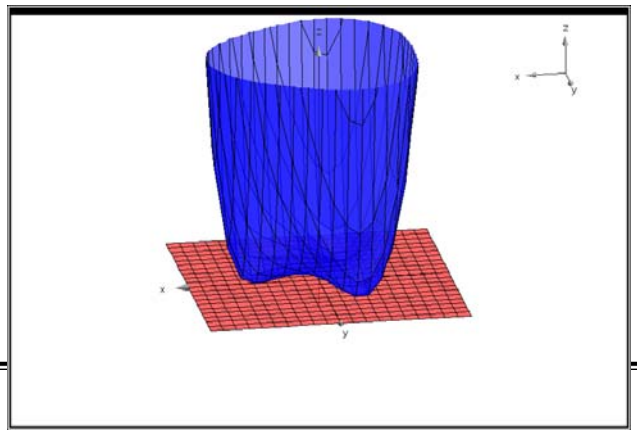
Of course, we can plot these 3 solutions in a plane and see they form the vertices of a equilateral triangle, each point being at a distance of 1 unity from the origin.

Looking at the Roots of Unity by Using a 3D Graph

Students have problems with complex solutions of an equation because they don't see the roots on the real line. Suppose you want to see the solution of the (polynomial) equation $p(z)=0$. Because, for a complex number w we have $w = 0 \Leftrightarrow |w| = 0 \Leftrightarrow |w|^2=0$, one possibility is to plot the graph of the surface $|p(z)|^2$ and to look at those places where the surface is touching the plane $z = 0$. This has always been possible with Nspire CX CAS because 3D plotting is available. See page 4 for the example of the equation $p(z) = 0$ where $p(z) = z^3-1$.

In this case, we take the function defined by

$|(x+i\cdot y)^3-1|^2 \blacktriangleright x^6+3\cdot x^4\cdot y^2-2\cdot x^3+3\cdot x^2\cdot y^4+6\cdot x\cdot y^2+y^6+1$



Looking at the Roots of Unity by Using 2D graphs

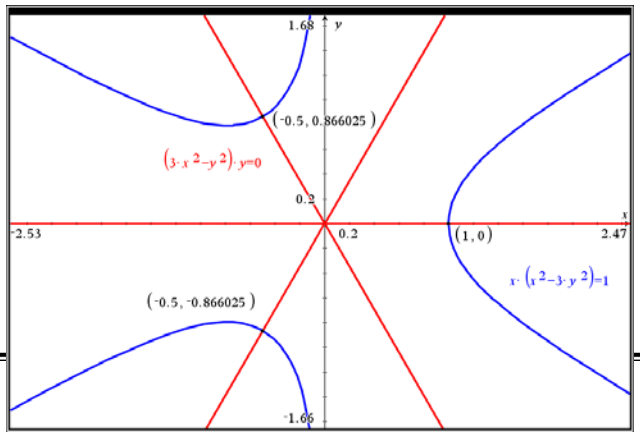
With OS4.4, we can now plot any **polynomial** equation in two variables x, y of any degree. Going back to our equation $z^3 = 1$, we simply need to take real and imaginary parts of both sides:

$\text{real}\left((x+i\cdot y)^3\right)=1 \blacktriangleright x\cdot\left(x^2-3\cdot y^2\right)=1$

$\text{imag}\left((x+i\cdot y)^3\right)=0 \blacktriangleright \left(3\cdot x^2-y^2\right)\cdot y=0$

We open a 2D plot window and choose the RelationGraph Entry/Edit mode. Where the two curves intersect are the 3 solutions. See page 6:

Note it would have been possible to plot the 2 curves of page 6 without OS4.4 because the equations can be solved for y or x.



Solving $z^5 = 1$

Exemple: now, let's take a look at the equation $z^5=1$, that is the 5 fith roots of unity.

The solutions are

$\text{cSolve}(z^5=1,z)$

$z=\frac{-(\sqrt{5}+1)}{4}+\frac{\sqrt{-2\cdot(\sqrt{5}-5)}}{4}\cdot i$ or $z=\frac{-(\sqrt{5}+1)}{4}-\frac{\sqrt{-2\cdot(\sqrt{5}-5)}}{4}\cdot i$ or $z=\frac{\sqrt{5}}{4}-\frac{1}{4}+\frac{\sqrt{2\cdot(\sqrt{5}+5)}}{4}\cdot i$ or $z=\frac{\sqrt{5}}{4}-\frac{1}{4}-\frac{\sqrt{2\cdot(\sqrt{5}+5)}}{4}\cdot i$ or $z=1$

BTW, we see Nspire knows what are the exact values of sines and cosines of angles of the form $\frac{k\cdot\pi}{5}$ for integers k.

These 5 solutions in approximade mode:

$(\text{cSolve}(z^5=1,z))\blacktriangleright\text{Decimal}$

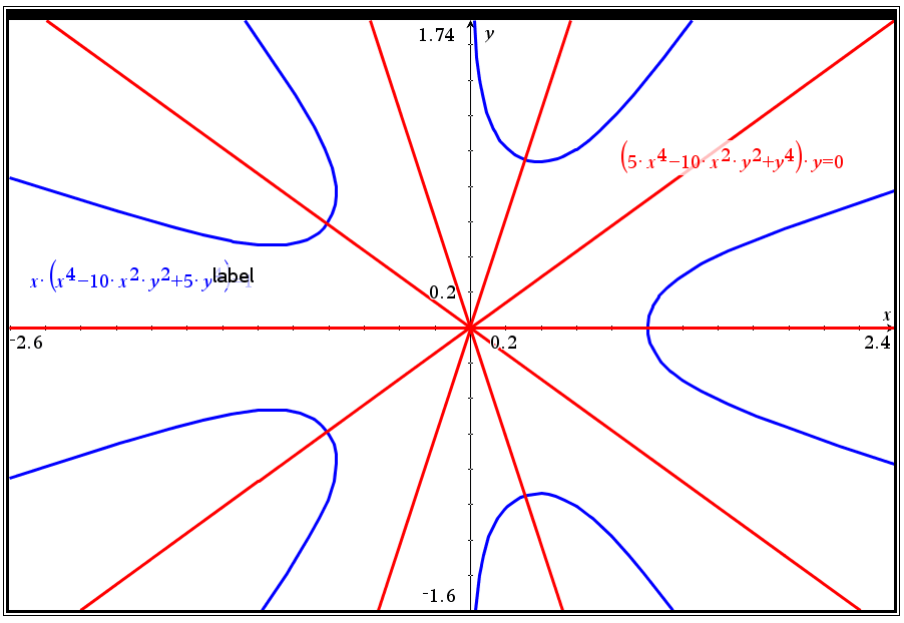
$z=0.309017+0.951057\cdot i$ or $z=0.309017-0.951057\cdot i$ or $z=-0.809017+0.587785\cdot i$ or $z=-0.809017-0.587785\cdot i$ or $z=1.$

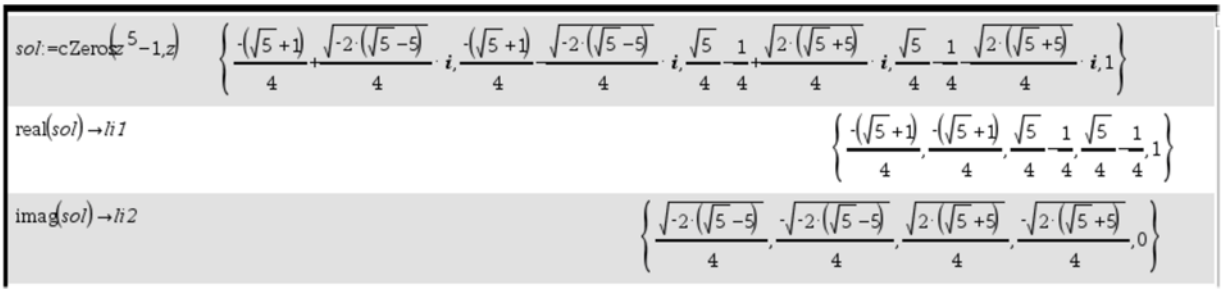
With OS4.4, we are now able to see these 5 points in the plane by using the intersection of 2 curves, namely

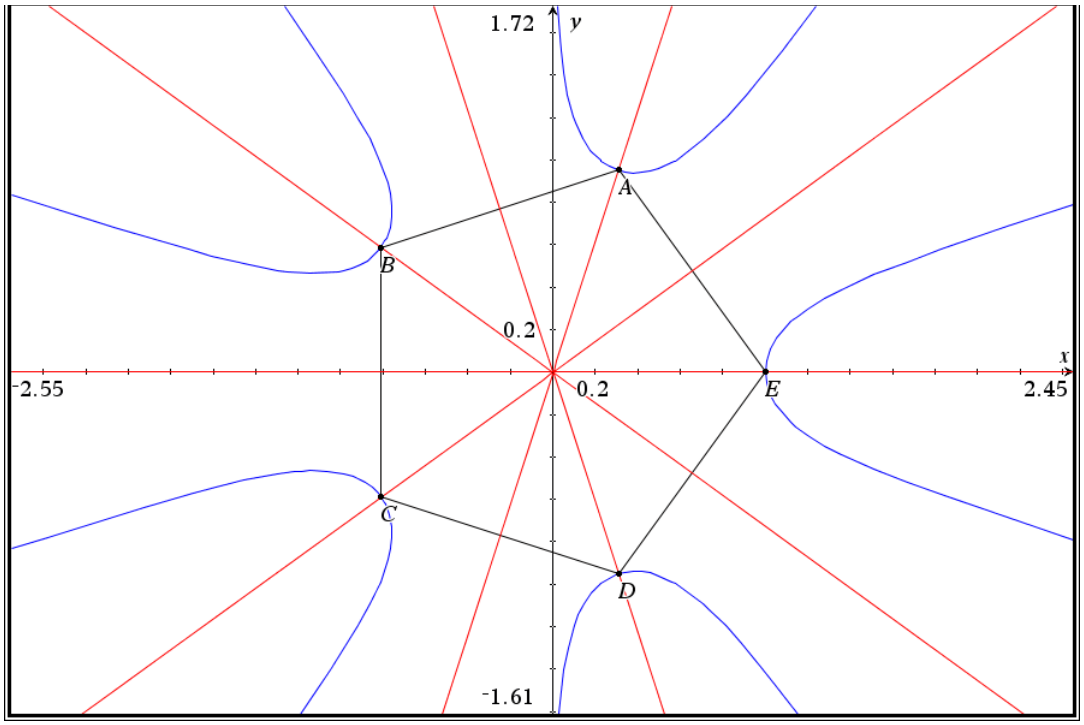
$\text{real}((x+i\cdot y)^5)=1 \blacktriangleright x\cdot(x^4-10\cdot x^2\cdot y^2+5\cdot y^4)=1$

$\text{imag}((x+i\cdot y)^5)=0 \blacktriangleright (5\cdot x^4-10\cdot x^2\cdot y^2+y^4)\cdot y=0$

See page 2. Page 3 is a calculator page where we have defined the list of the real and imaginary parts of the solutions. Page 4 shows the pentagon whose vertices are the 5 solutions. Uncheck the 2 relations in order to hide the curves.

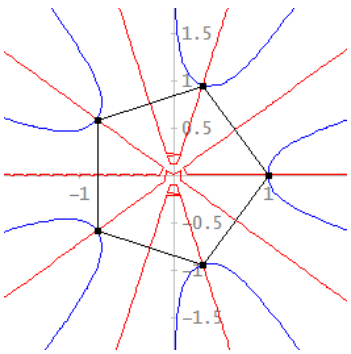


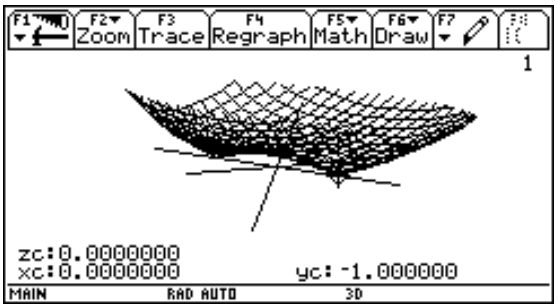




I cannot resist to add the respective DERIVE “Root-Pentagon” and a screen shot from an earlier DNL: It was Bert Wait’s favorite example demonstrating the TI-92’s 3D-plotting feature: making complex roots visible (DNL#55, The Modulo Surface).

```
#1:   RE((x + i·y)^5) = 1

#2:   IM((x + i·y)^5) = 0

#3:   pts := [RE(SOLUTIONS(z^5 - 1 = 0, z)), IM(SOLUTIONS(z^5 - 1 = 0, z))]'

#4:   VECTOR(pts_k, k, [1, 2, 4, 5, 3, 1])
```





Attracted by (STRANGE) Attractors

An illustrated guided tour from well-known to unknown attractors

Your Tour Guide: Josef Böhm

Abstract

When investigating dynamic systems, fractals and chaotic behavior it is inevitable to come across "attractors". An attractor is a set towards which a dynamical system evolves over time. This can be one ore more points, a curve, or a kind of fractal structure in the phase space. The latter are called "strange attractors". Investigating these special sets opens a huge box of surprising shapes which might form a linkage between mathematics and arts.

We will use the excellent mathematical capabilities of DERIVE (and other tools) wrapped in programs to present well known and not so well-known attractors which can be found in "CHAOS-Literature", as there are: Lorenz-, Hénon-, Rössler-, Ikeda-, Gumowski-Mira-attractors and others.

This is a revised and extended summary of my PowerPoint presentation given at TIME 2012 together with some additional notes. In particular, I added the treatment by TI-NspireCAS, too – as far as possible.

1 Introduction and well-known basics

The *Strange Attractors* can be counted to recreational mathematics and to creational mathematics as well. This field of mathematics offers an affective (emotional) approach to interesting insights rather than a cognitive (rational) one. There is also a lot of pure and hard mathematics behind all these nice patterns which I will present in the following. I served more than 30 years as school teacher and was always glad offering "pretty" mathematical objects to catch the interest of the students which were not so happy with the hard facts of maths. One can really see and experience the BEAUTY of mathematics approaching the secrets of STRANGE ATTRACTORS.

An attractor is a set towards a dynamical system evolves over time. This can be

- one or more points
- a kind of fractal structure
- a curve

A key word for all what is following is **ITERATION**:

Iteration of one variable: $x_{n+1} = f(x_n); x_0 = \text{given}$

Iteration of two (or more variables (dynamic system)): $\begin{matrix} x_{n+1} = f_1(x_n, y_n) \\ y_{n+1} = f_2(x_n, y_n) \end{matrix}$; $x_0, y_0 = \text{given}$

The first example of a one variable iteration is the *Logistic Parabola*. Its iteration formula is given by

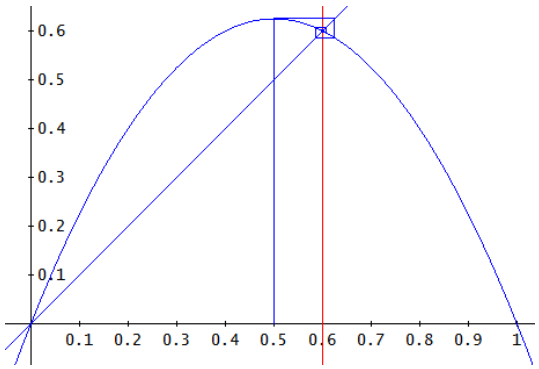
$$x_{n+1} = r \cdot x_n \cdot (1 - x_n).$$

The question is: What happens when plotting the graph of x_{n+1} versus x_n?

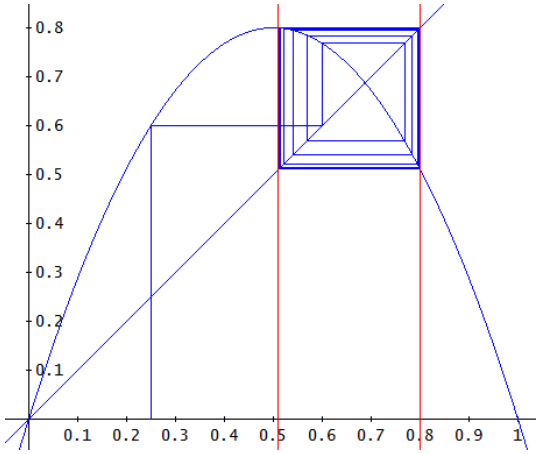
First, we investigate how the sequence develops. We apply DERIVE's versatile ITERATES-command and show the last *k* of the first *n* elements of the sequence.

```
#1:   [Precision := Approximate, Notation := Decimal, NotationDigits := 4]

      last_k(r, x0, n, k) := VECTOR((ITERATES(r·x·(1 - x), x, x0, n))  , i, n - k + 1, n)
#2:                                                                  i

#3:   last_k(2.5, 0.3, 100, 10) = [0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6, 0.6]

#4:   last_k(3.2, 0.5, 100, 10)

#5:   [0.5130, 0.7994, 0.5130, 0.7994, 0.5130, 0.7994, 0.5130, 0.7994, 0.5130, 0.7994]

#6:   last_k(3.5, 0.75, 100, 10)

#7:   [0.8749, 0.3828, 0.8269, 0.5008, 0.8749, 0.3828, 0.8269, 0.5008, 0.8749, 0.3828]

#8:   last_k(3.8, 0.2, 100, 10)

#9:   [0.4438, 0.9380, 0.2209, 0.6540, 0.8598, 0.4578, 0.9432, 0.2034, 0.6157, 0.8991]
```

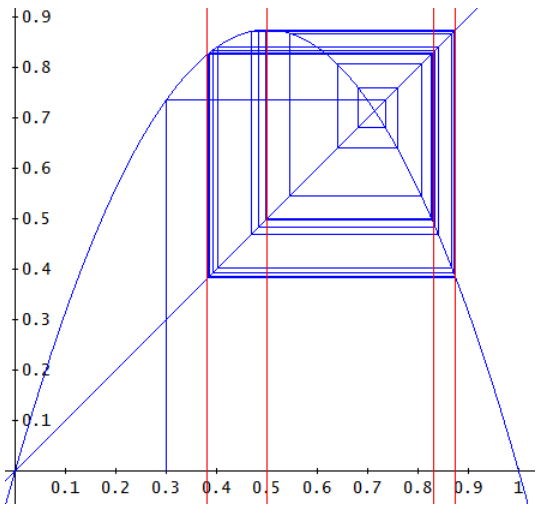
We observe the different behavior of the iteration process for different values for *r*. Let's illustrate this quite better by plotting the graph of x_{n+1} versus x_n producing a cobweb diagram:



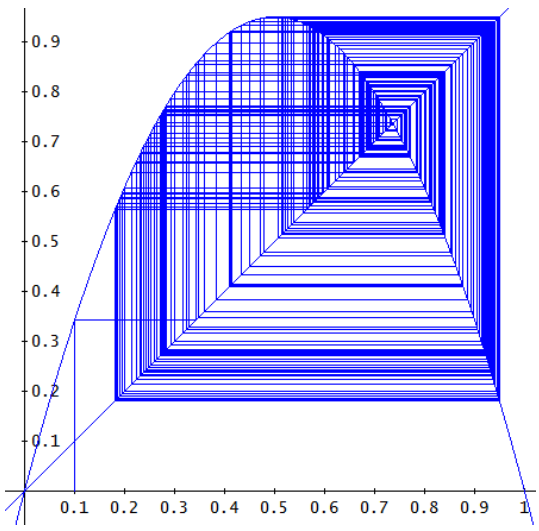
r = 2.5



r = 3.2



r = 3.5



r = 3.8

The figures given above illustrate the iteration process for different values for r plotting the cobweb diagrams. All iterations are starting with $x_0 = 0.3$. We can observe one, two, four point attractors, and chaotic behavior, respectively. The end behavior does not depend on the initial value.

I present two ways to produce the logistic parabola together with the cobweb diagram and the straight line $y = x$.

You can take a program. #11 gives the first figure.

The purists among the *DERIVIANs* will miss the very special ITERATES-command for describing this iteration process. (Let's remember the huge and impressive ITERATEs-constructs composed by Hannes Wiesenbauer!).

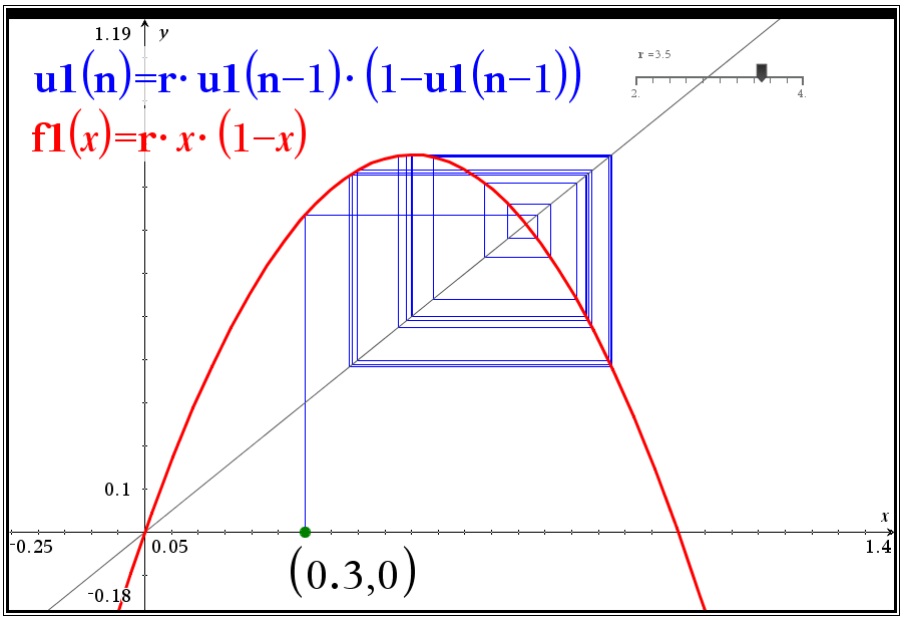
```
#10:  log_parab(r, x0, n, i, f, web) :=
        Prog
          web := [[x0, 0]]
          i := 1
          Loop
            If i > n exit
            f := r·x0·(1 – x0)
            web := APPEND(web, [x0, f; f, f])
            x0 := f
            i :+ 1
          [web, y = x, r·x·(1 – x)]

#11:  log_parab(2.5, 0.5, 100)
```

In DNL#13 you can find a short note of Jerry Glynn providing a respective routine. Here is a similar one:

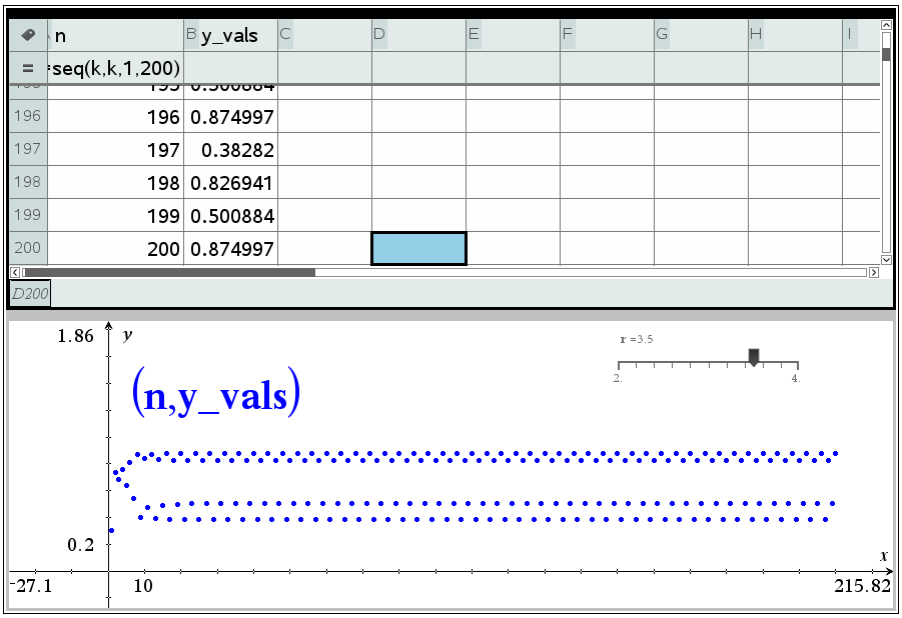
```
f(r, x) := r·x·(1 – x)

cobw(r, x0, n) := [f(r, x), y = x, [x0  0; x0  x0], ITERATES([v3,1  v3,1; v3,1  f(r, v3,1); f(r, v3,1)  f(r, v3,1)], v, [x0  x0; x0  f(r, x0); f(r, x0)  f(r, x0)], n)]

cobw(3.5, 0.3, 200)
```

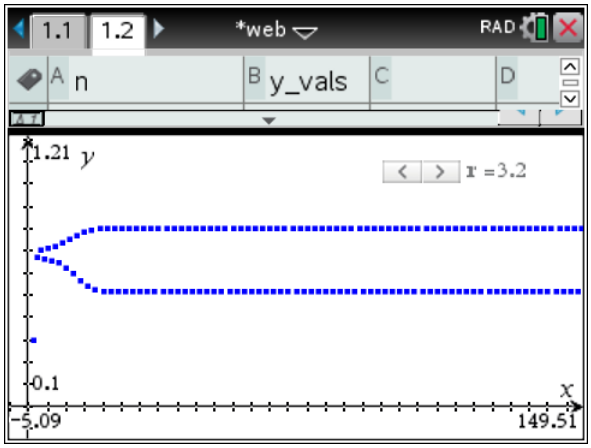
We can observe one, two or four point attractors, and chaotic behavior, respectively.

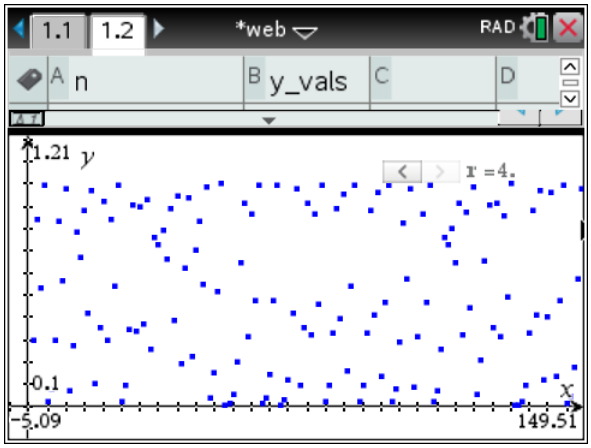
TI-NspireCX CAS allows comfortable plotting without programming. We define the iteration in the sequence entry and install a slider for r. We set a point on the x-axis (green) and link its x-coordinate to a variable, say $x0$, which is used as initial value for the sequence. Then we can move the initial point and vary the r-value as well.



Two columns in the spreadsheet and their representation as a scatter diagram demonstrate the appearance of the attractors in combination with the slider in another attractive way







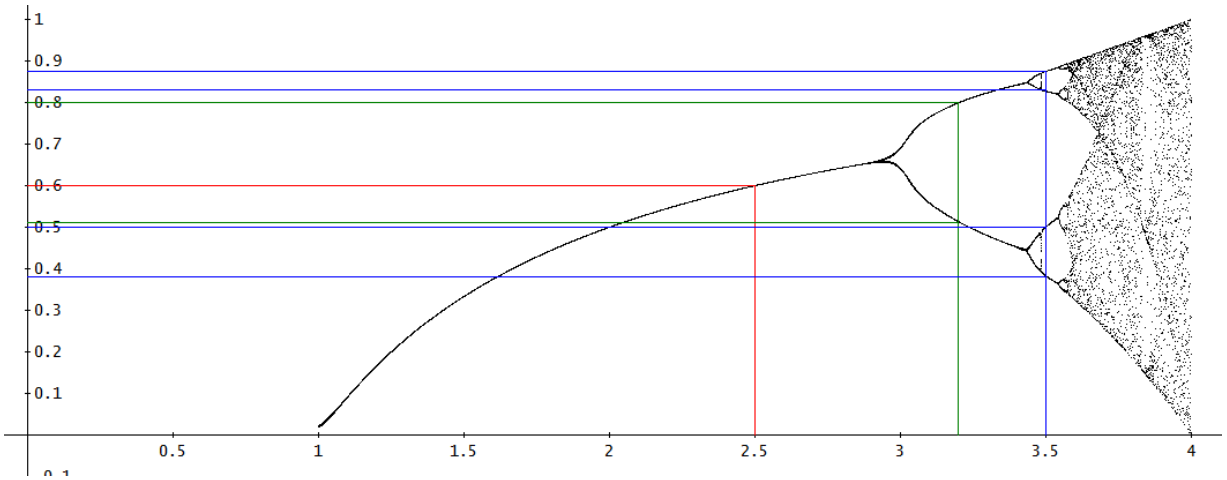
The so called *period doubling* and its transition to *chaos* can be presented in another famous form.

The key word is **Bifurcation** (Feigenbaum diagram):

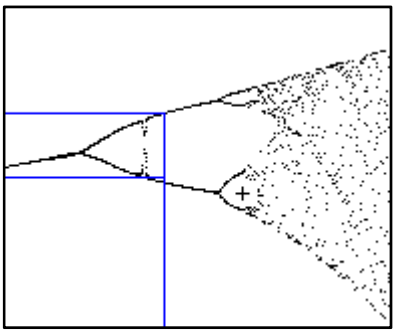
We calculate for all values r with $1 < r < r\_e$ with step size *inc* the first n elements of the iteration and we plot the last k values against r. So we can observe the attractors.

#20 generates the k points belonging to one given r and initial value $x0$ and #21 composes all vertical sets of points to the Feigenbaum diagram. The plot of #22 is presented on the next page. Calculation does not need too long time. You might find out how many points are plotted.

```
#20:  bifurc(r, x0, n, k) := VECTOR([r, (last_k(r, x0, n, k))[i]], i, k)

#21:  feigb(x0, n, k, r_e, inc) := VECTOR(bifurc(r_, x0, n, k), r_, 1, r_e, inc)

#22:  feigb(0.3, 50, 10, 4, 0.001)
```

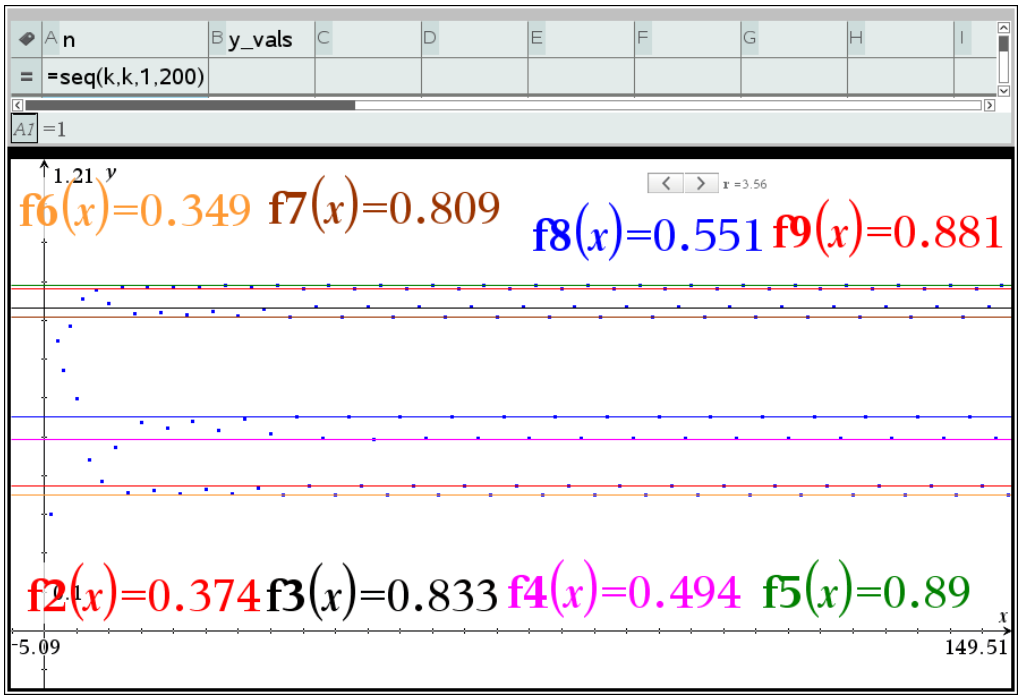


The endpoint behavior for r 2.5, 3.2 and 3.5 is marked in red, green and blue, respectively. Inspecting this scatter plot we might get the impression that there appears at least one more period doubling. Right of the blue line x = 3.5 we detect four "bubbles" lying in a vertical line.

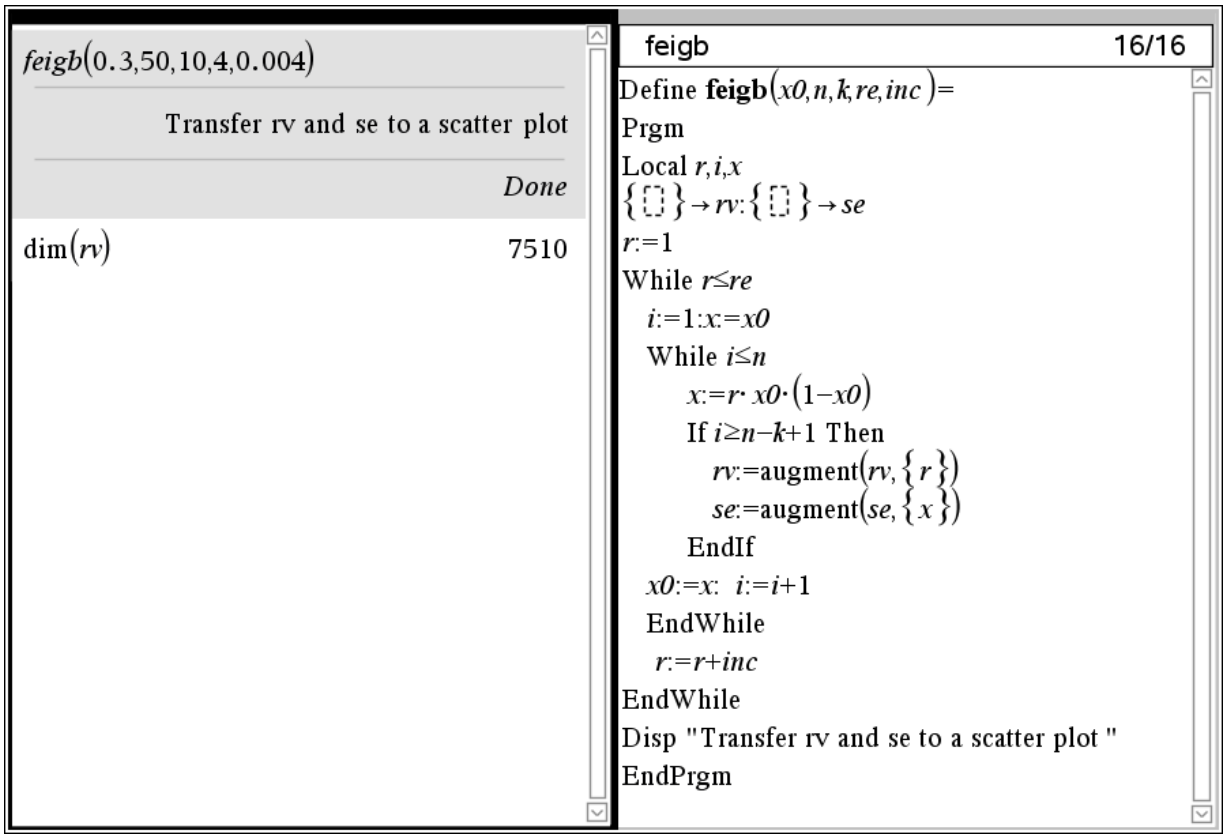


We zoom in and read off the x-coordinate of the cross ≈ 3.56. When we calculate the last 20 elements of the first 200 ones we really can confirm that there is a period of eight elements.

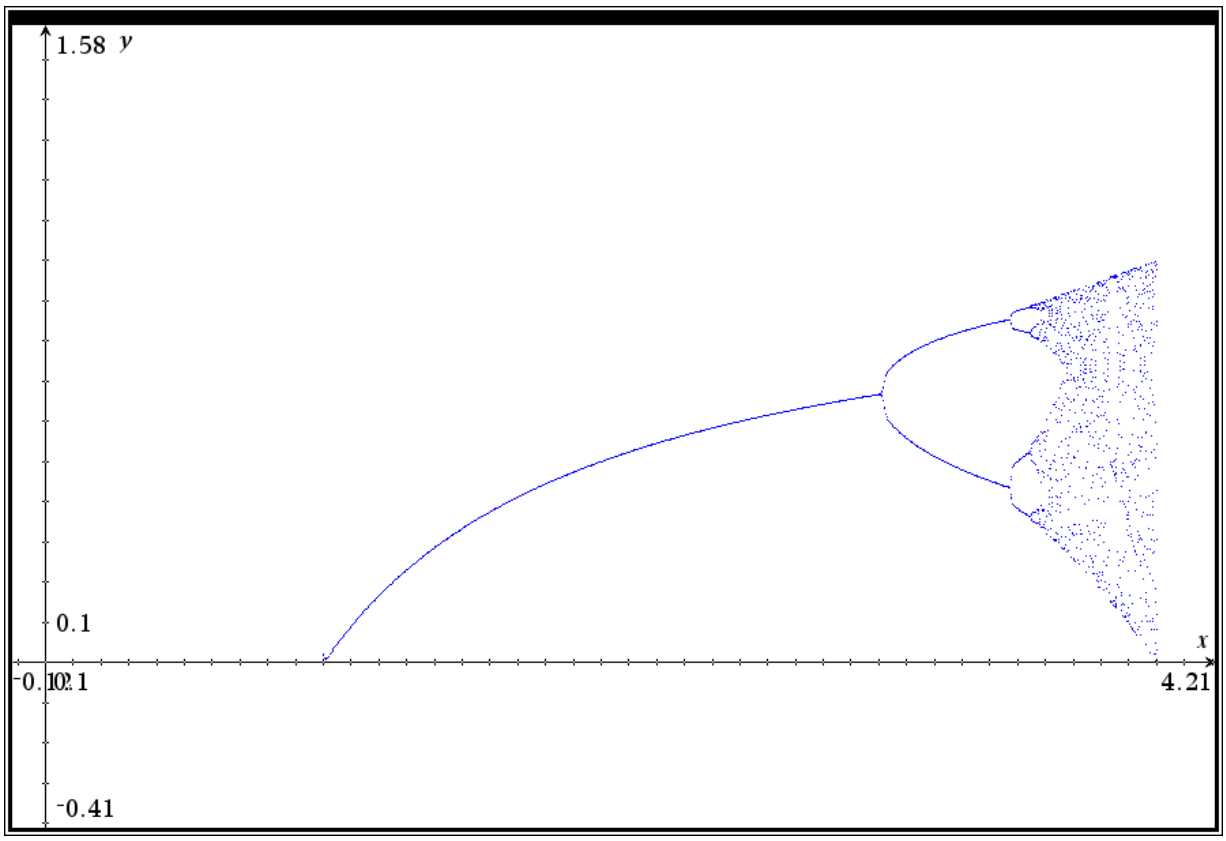
```
#23:  last_k(3.56, 0.4, 200, 20)

#24:  [0.3738, 0.8333, 0.4944, 0.8898, 0.3488, 0.8086, 0.5508, 0.8807, 0.3738, 0.8333,
       0.4944, 0.8898, 0.3488, 0.8086, 0.5508, 0.8807, 0.3738, 0.8333, 0.4944, 0.8898]
```



The question is: Can we produce the Feigenbaum diagram on the TI-NspireCAS screen, too. To answer with Barack Obama: “Yes, we can!”.

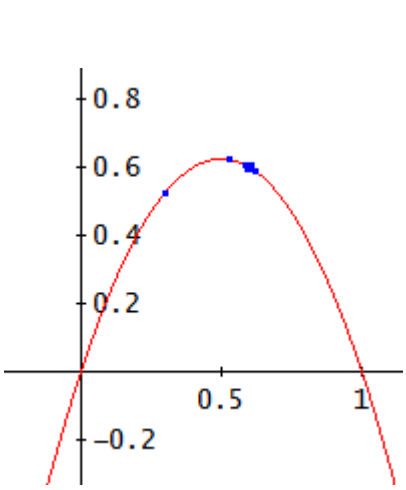


Program feigb with the same syntax as DERIVE `feigb` generates lists rv (r-values) and se (sequence elements) which can be placed into the Scatter Plot Graph Entry. Gives a pretty plot!

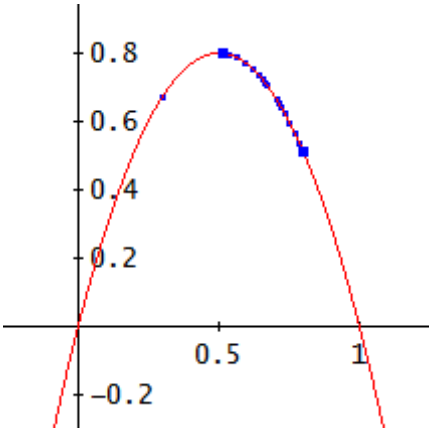


Before proceeding to the next chapter I would like to describe one more possibility visualizing the iteration process together with the attractors.

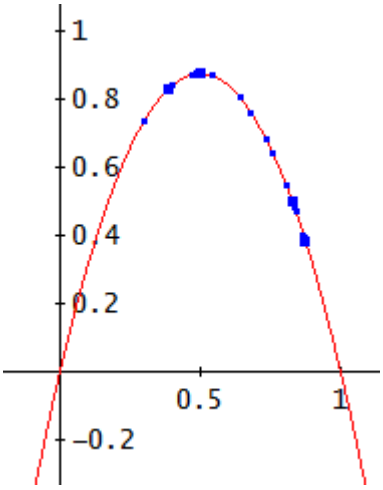
```
lp(r, x0, n) := ITERATES([v , f(r, v )], v, [x0, f(r, x0)], n)
                          [ 2        2 ]
```



r = 2.5, 1000 iterated points



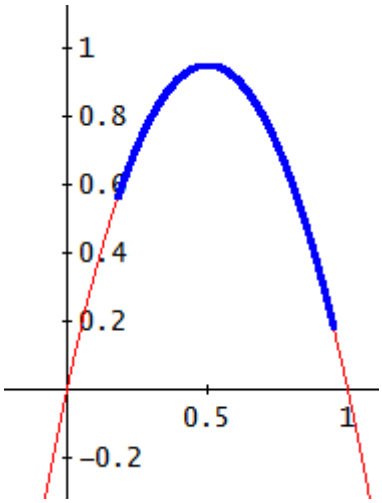
r = 3.2, 1000 points



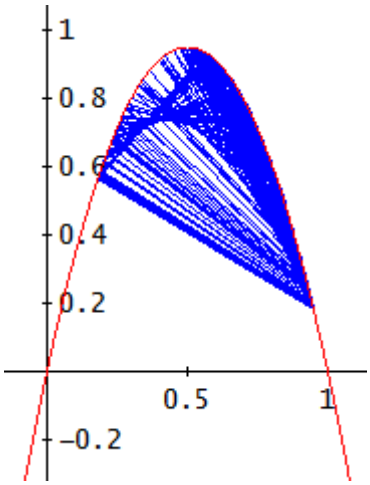
r = 3.5, 1000 points

I plot the logistic parabola (red) and the first 1000 or 10 000 points (x_i, x_{i+1}) in blue (medium size). We see that all points are lying on the parabola. The attractor points are also in blue but in large size. There are no attractors for *r* = 3.8.

Once I forgot to deactivate the Points >Connect>No box and found the family of chords plotted. I wonder if there is an eneveloppe of the chords – and how to find it.



r = 3.8, 10 000 points

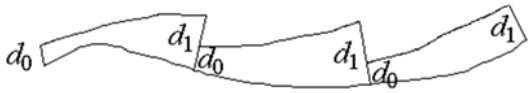


200 points, Connected

2 The Lyapunov exponent

As we are approaching chaos now we need a measure to test chaotic behavior. Chaotic processes are extremely sensitive to initial conditions. Reading this you might remember the famous “Butterfly Effect”.

The general idea is to follow two nearby orbits and to calculate their average logarithmic rate of separation. The Lyapunov exponent λ is the power of 2 by which the difference between two nearly equal *x*-values changes on average for each iteration i.e. the difference changes by an average of 2^{λ} for each iteration. (This is a definition for a one-dimensional iteration. It is adapted in case of more dimensional systems.)



Start with any point on the attractor and choose a nearby point separated by d_0. Iterate both values and find their distance d_1. Now calculate $\log_2\left|\frac{d_1}{d_0}\right|$. Readjust one orbit so its separation is d_0 in the same direction as d_1 and repeat this procedure many times and then calculate the average of the logarithms [3].

> If $\lambda < 0$ then the solutions approach another → we receive a point attractor
> but
> if $\lambda > 0$ then the process is sensitive to initial conditions → we can expect chaotic behavior → we receive a **strange attractor**.

This is another interpretation: The Lyapunov exponent λ (LE) is the rate at which information is lost when a map is iterated.

In case of one dimensional iterations we can use means of calculus to find the LE.

$$x_{i+1} = f(x_i)$$

x_i and neighbour $\tilde{x}_i = x_i + \Delta_i$

$$\lim_{\Delta\to 0}\frac{\Delta_{i+1}}{\Delta_i} = \lim_{\Delta\to 0}\frac{\tilde{x}_{i+1} - x_{i+1}}{\Delta_i} = \frac{f(x_i+\Delta_i) - f(x_i)}{\Delta_i} =$$

$$= \left|f'(x_i)\right|$$

$$\lambda = \lim_{N\to\infty}\frac{1}{N}\sum_{i=1}^{N}\ln\left|f'(x_i)\right|$$

Applying this formula on the logistic equation we receive:

$$x_{i+1} = r\,x_i(1-x_i) = r\,x_i - r\,x_i^2 = f(x_i)$$

$$\left|f'(x_i)\right| = \left|r - 2r\,x_i\right|$$

$$\lambda = \frac{1}{N}\sum_{i=1}^{N}\log_2\left|r - 2r\,x_i\right|$$

I wrote a short DERIVE program for testing the Lyapunov exponent:

```
lyaptest(cf, its, t_, xn, x0, o, f, f1, i, l) :=
  Prog
    o := DIM(cf) - 1
    f := cf·VECTOR(x^k, k, 0, o)
    f1 := ∂(f, x)
    [i := 1, l := 0, x0 := 0.05]
    x0 := ITERATE(f, x, x0, 14 - 2·o)
    Loop
      If ABS(x0) > 100 exit
      If i > its
         RETURN l/its
      xn := LIM(f, x, x0)
      l := l + LOG(ABS(LIM(f1, x, xn)), 2)
      x0 := xn
      i :+ 1
```

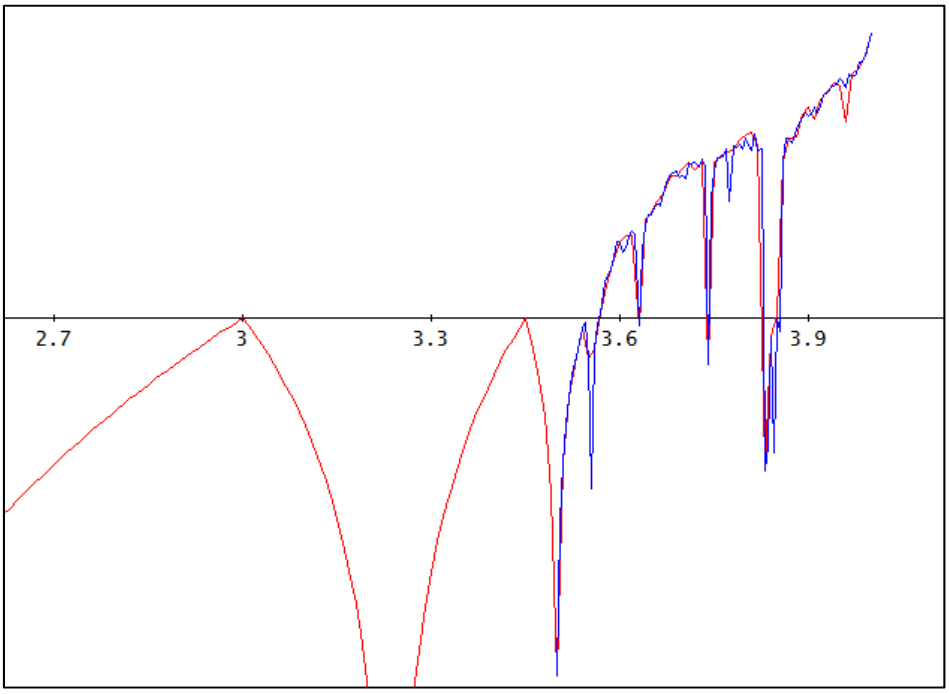
Let us check the LEs of the logistic parabolas from above.

You can see that for $r = 3.8$ the LE turns out to be positive.

`cf` are the coefficients of the parabola in ascending order, e.g. [0, 3.2, -3.2] for $3.2x\cdot(1-x) = 3.2x - 3.2x^2$. `its` is the number of iterations – which is N in the formulas given above. We will use this program for polynomials of higher order, too.#

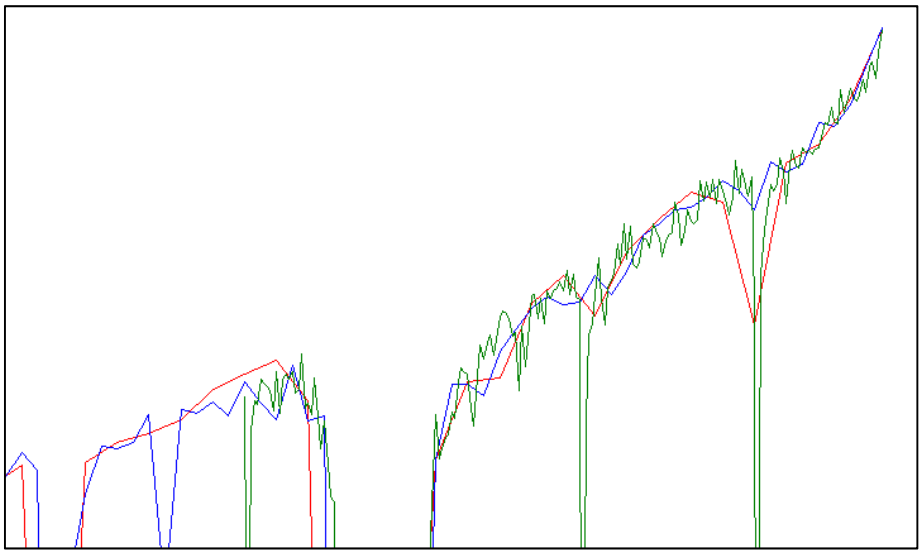
```
lyaptest([0, 2.5, -2.5], 2000) = -1.0000
lyaptest([0, 3.2, -3.2], 2000) = -1.3219
lyaptest([0, 3.5, -3.5], 2000) = -1.2587
lyaptest([0, 3.8, -3.8], 2000) = 0.61736
```

In [3] I found the graph of the LE depending on r for the logistic parabolas with $2 \le r \le 4$. Applying the basics of the program from above we can produce this interesting graph using program `lyape_f(iterations, r_start, r_end, r_inc)`.



`lyape_f(1000,2.5,4,0.01)` (red) and `lyape_f(1000,3.5,4,0.005)` (blue)

The zeros of the graph indicate the position of the bifurcations. Zooming into the right upper corner shows the fractal structure of this graphic representation (`lyape_f(1000,3.8,4,0.001)` in green).



3 The first investigation

We will try to extend the logistic parabola to ordinary parabolas and further to parabolas (polynomials) of higher order. At the moment, we have the only chance to find quadratic parabolas with a positive LE by try and error:

My first try:

$$f(x) := 3.2 \cdot x^2 + 1.2 \cdot x - 0.1$$

lyaptest([−0.1, 1.2, 3.2], 1000) = −2.7474

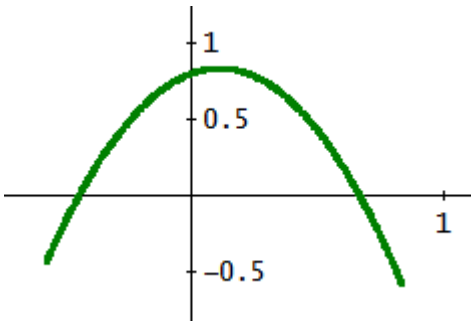
$$\text{ITERATES}\left(\begin{bmatrix} v_2 & f(v_2) \end{bmatrix}, v, [0.1, f(0.1)], 5\right) = \begin{bmatrix} 0.1 & 0.052 \\ 0.052 & -0.028947 \\ -0.028947 & -0.13205 \\ -0.13205 & -0.20266 \\ -0.20266 & -0.21176 \\ -0.21176 & -0.21061 \end{bmatrix}$$

This was not very successful. We have a point attractor, which is not surprising because the LE is negative. Let's try another one:

$$g(x) := -2.7 \cdot x^2 + 0.6 \cdot x + 0.8$$

lyaptest([0.8, 0.6, −2.7], 1000) = 0.86524

$$\text{ITERATES}\left(\begin{bmatrix} v_2 & g(v_2) \end{bmatrix}, v, [0.1, g(0.1)], 1000\right)$$



The positive LE is promising. We plot x_{n+1} vs x_n.

It is boring and frustrating finding "attractive" parabolas and polynomials just by trying sets of coefficients. Very soon we will provide a tool for organizing this search which will guarantee success.

How to obtain "attractive" polynomials of higher order? Once found a parabola like $g(x)$ this is an easy job. We plot x_{n+k} (k steps later iteration) versus x_n with $k \geq 1$.

The next program gives this plot with `lits` = k with `lits := 1` by default.

```
pol_map(cf, its, x0, lits := 1, f, xn, xn_, vals, i) :=
  Prog
    vals := []
    i := 1
    f := cf•VECTOR(x^k, k, 0, DIM(cf) − 1)
    Loop
      If i > its
         RETURN vals
      xn := LIM(f, x, x0)
      xn_ := ITERATE(f, x, x0, lits)
      vals := APPEND(vals, [[x0, xn_]])
      x0 := xn
      i :+ 1
```

The reason for entering the parabola (and later the polynomials) in form of a list of the coefficients - and not as an expression - will be explained in the next paragraph.

We will compare the first generated points changing the last parameter of this function from 1 (by default) to 2 and further to 3.

```
pol_map([0.8, 0.6, -2.7], 5, 0.1)
```

0.1	0.833
0.833	−0.57370
−0.57370	−0.43287
−0.43287	0.034342
0.034342	0.81742

```
pol_map([0.8, 0.6, -2.7], 5, 0.1, 2)
```

0.1	−0.57370
0.833	−0.43287
−0.57370	0.034342
−0.43287	0.81742
0.034342	−0.51362

```
pol_map([0.8, 0.6, -2.7], 5, 0.1, 3)
```

0.1	−0.43287
0.833	0.034342
−0.57370	0.81742
−0.43287	−0.51362
0.034342	−0.22046

```
EXPAND(ITERATE(g(v), v, x, 2))
```

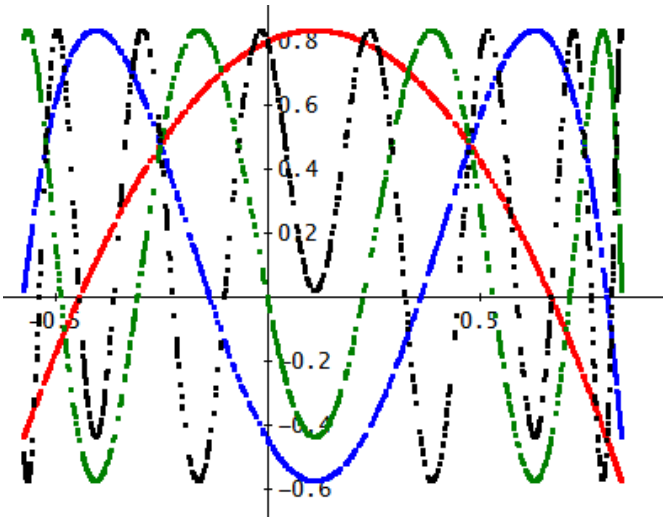
$$-19.683\cdot x^4 + 8.748\cdot x^3 + 9.072\cdot x^2 - 2.232\cdot x - 0.448$$

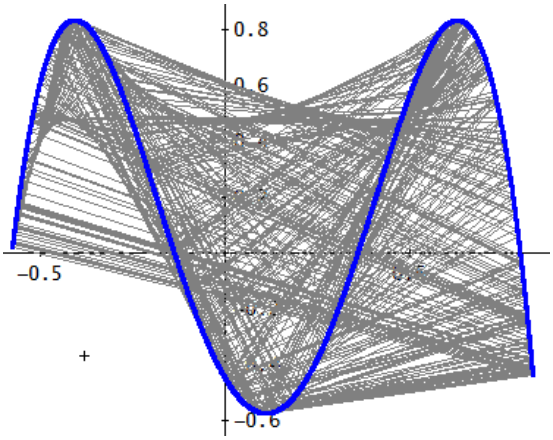
0.1	−0.57370
−0.57370	0.034342
0.034342	−0.51362
−0.51362	0.53648
0.53648	0.68591

Can you find respective points in this table and in the table above?

We plot $g(x)$ together with the first 1000 points of x_{i+1} vs x_i (red), x_{i+2} vs x_i (blue), x_{i+3} vs x_i (green) and x_{i+4} vs x_i (black) which is a polynomial of order 16.

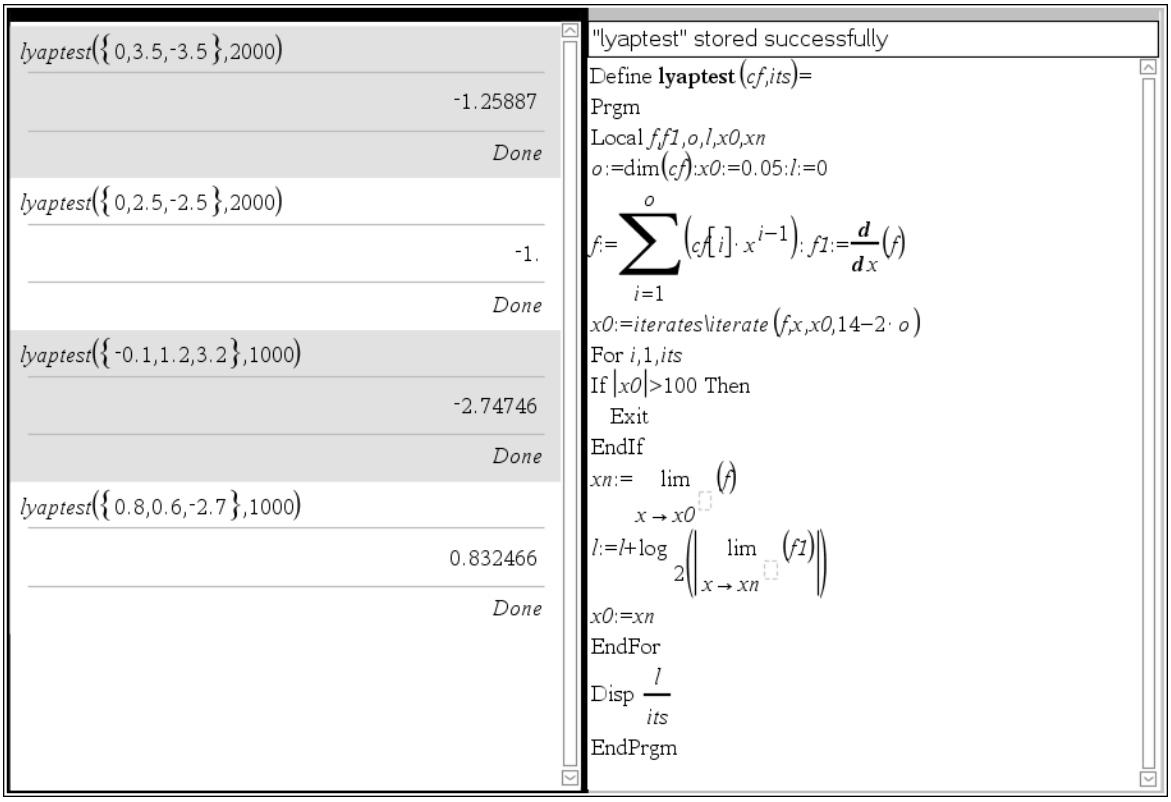
```
pol_map([0.8, 0.6, -2.7], 1000, 0.1)
pol_map([0.8, 0.6, -2.7], 1000, 0.1, 2)
pol_map([0.8, 0.6, -2.7], 1000, 0.1, 3)
pol_map([0.8, 0.6, -2.7], 1000, 0.1, 4)
```



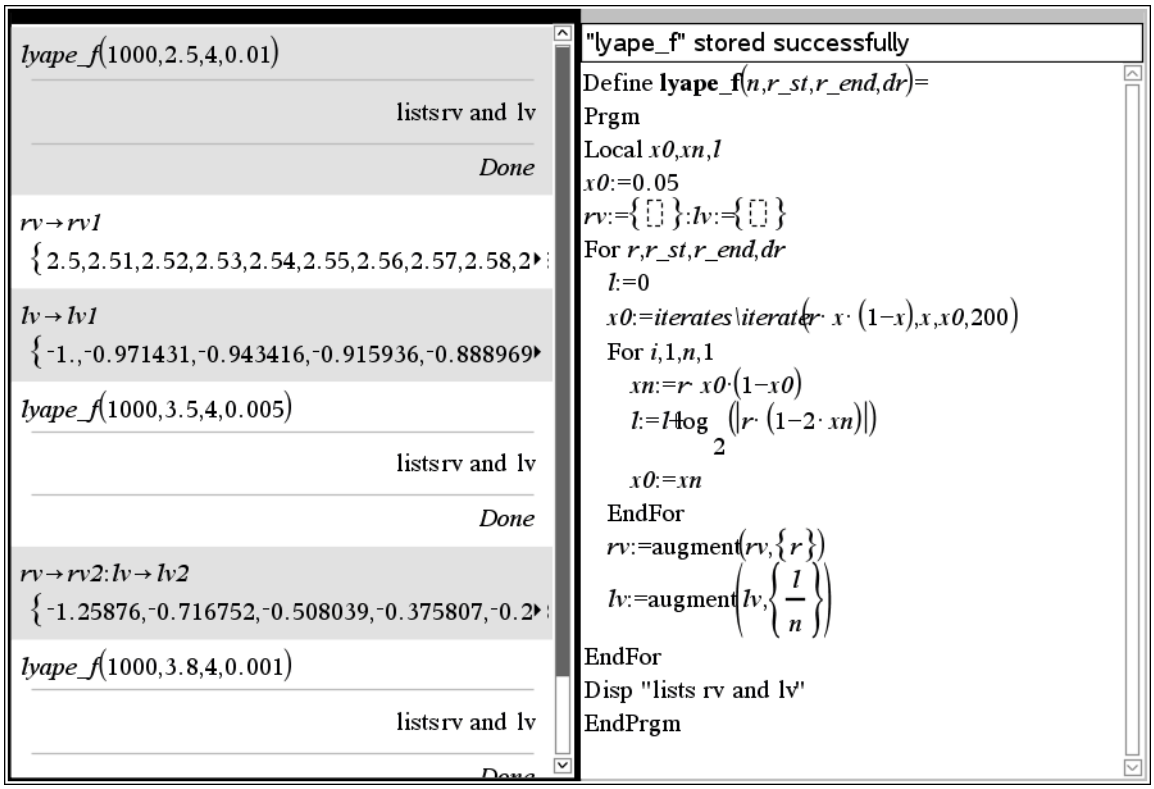


Plotting the points connected gives the family of chords and we again get the impression of a possible envelope.

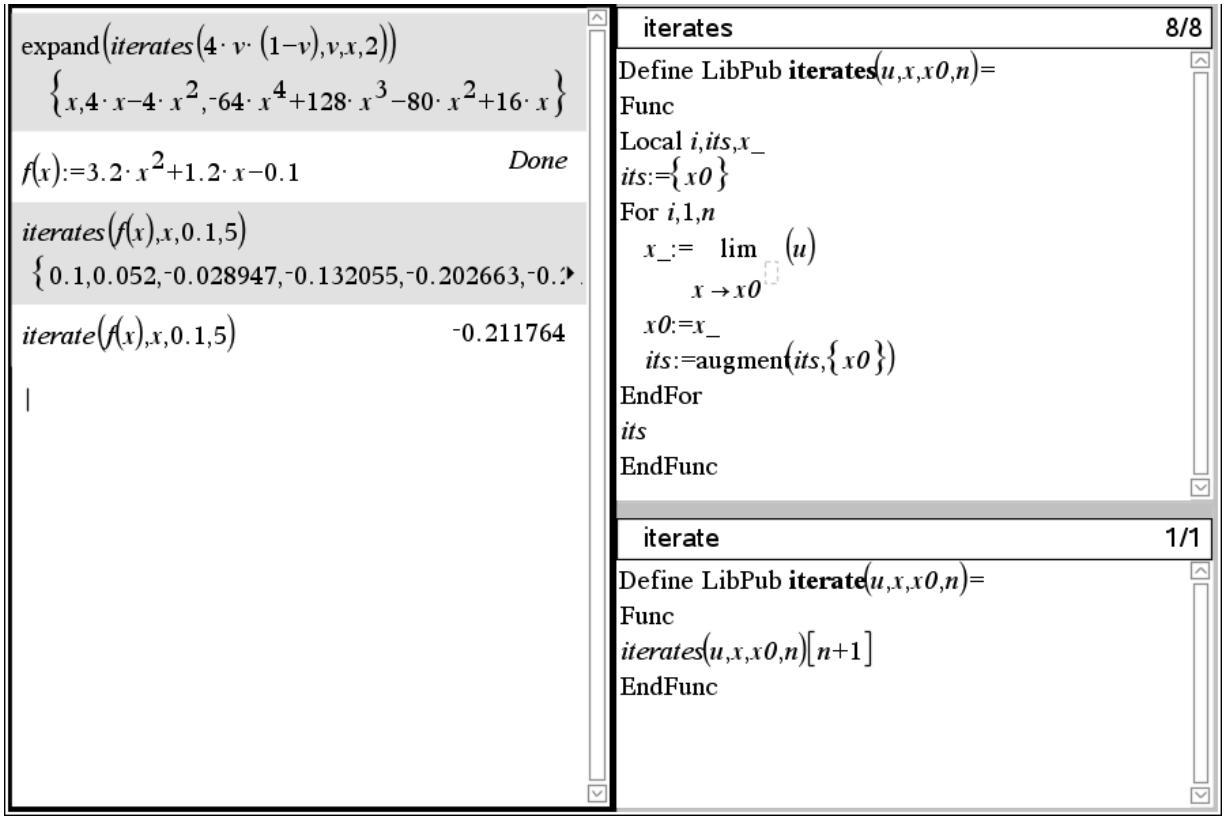
Now let's turn on the TI-NspireCAS again. Calculation of the Lyapunov exponent follows the DERIVE program from above. VECTOR must be replaced by a sequence and – more important – we have the common for-next loop available.



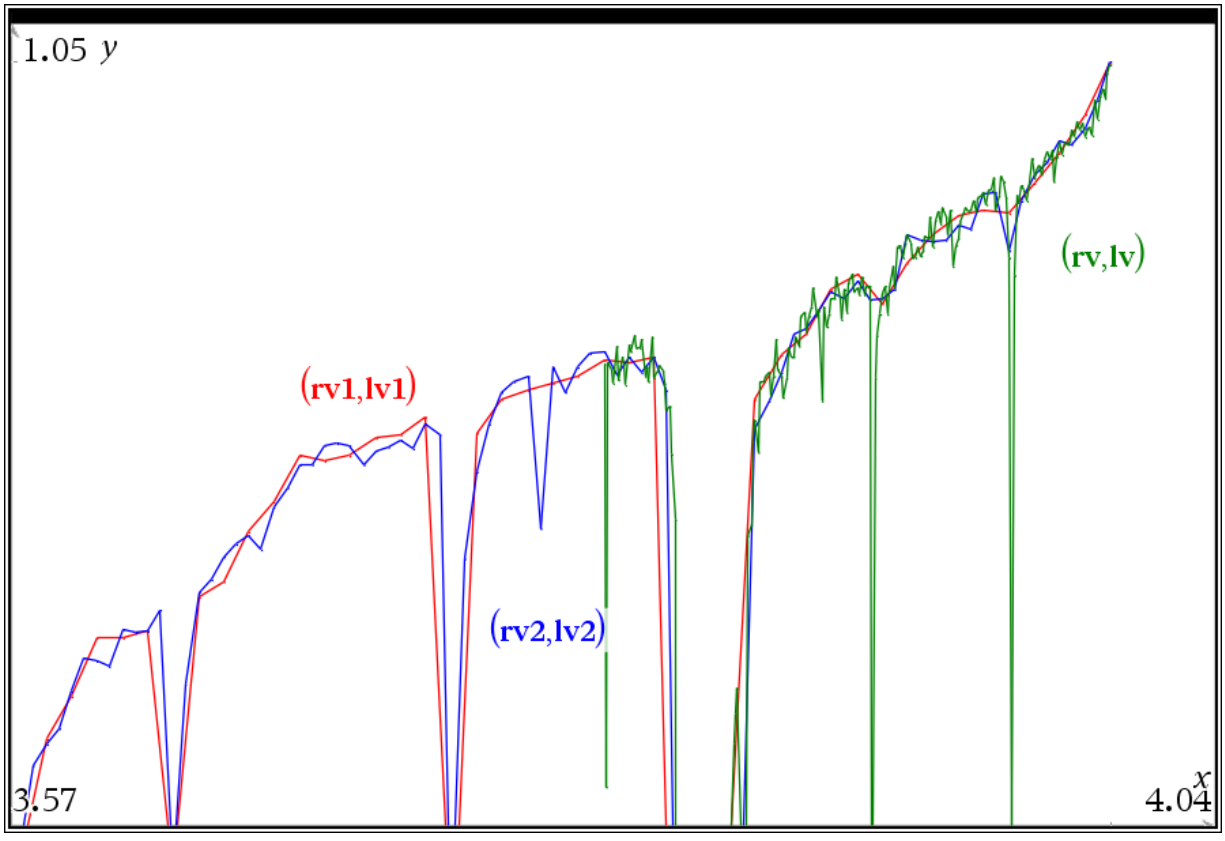
You will notice *iterate* for TI-NspireCAS. It is not implemented, so we define our own *iterate*-function and *iterates* as well. *Michel Beaudin* provides a more powerful *iterate* (and *iterates* as well) in his TI-Nspire toolbox but this one is sufficient enough for our purpose.



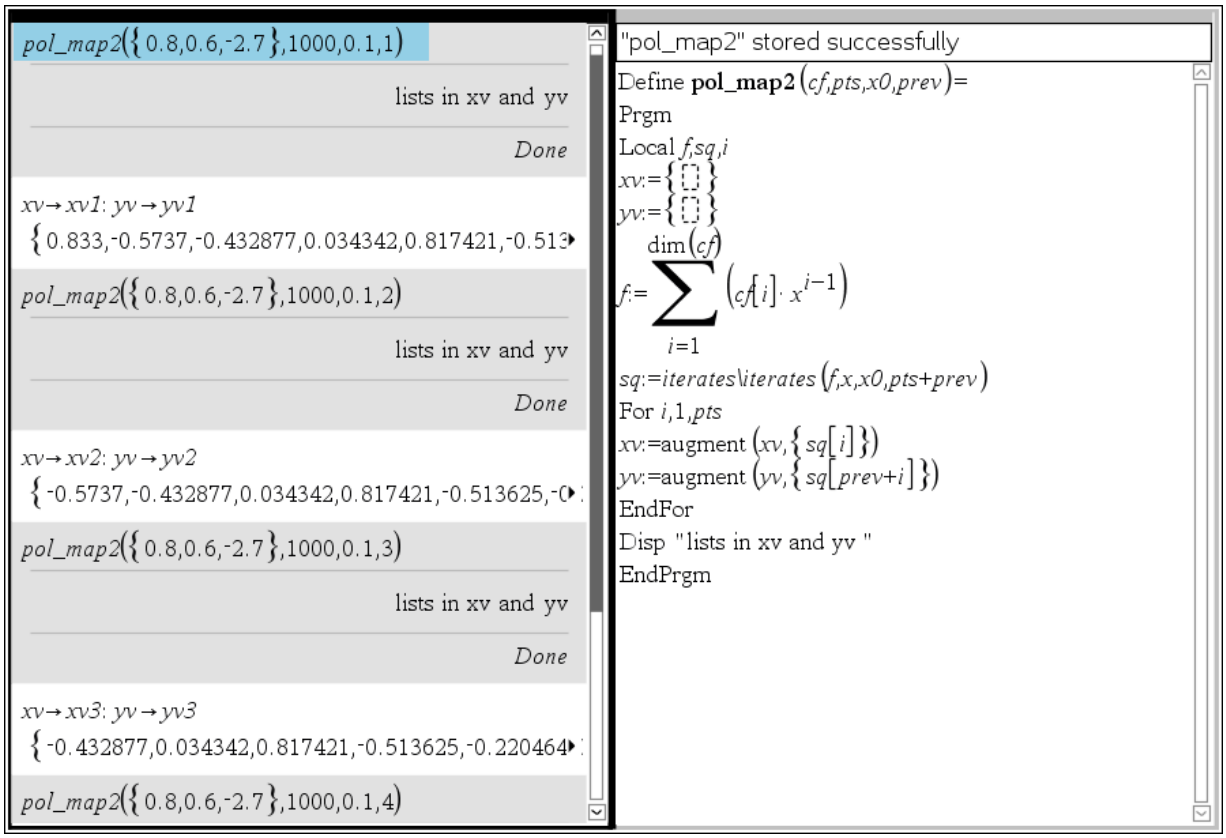
This is our *iterate* for TI-Nspire. (I put both function in a public library in order to have it available at any occasion.)

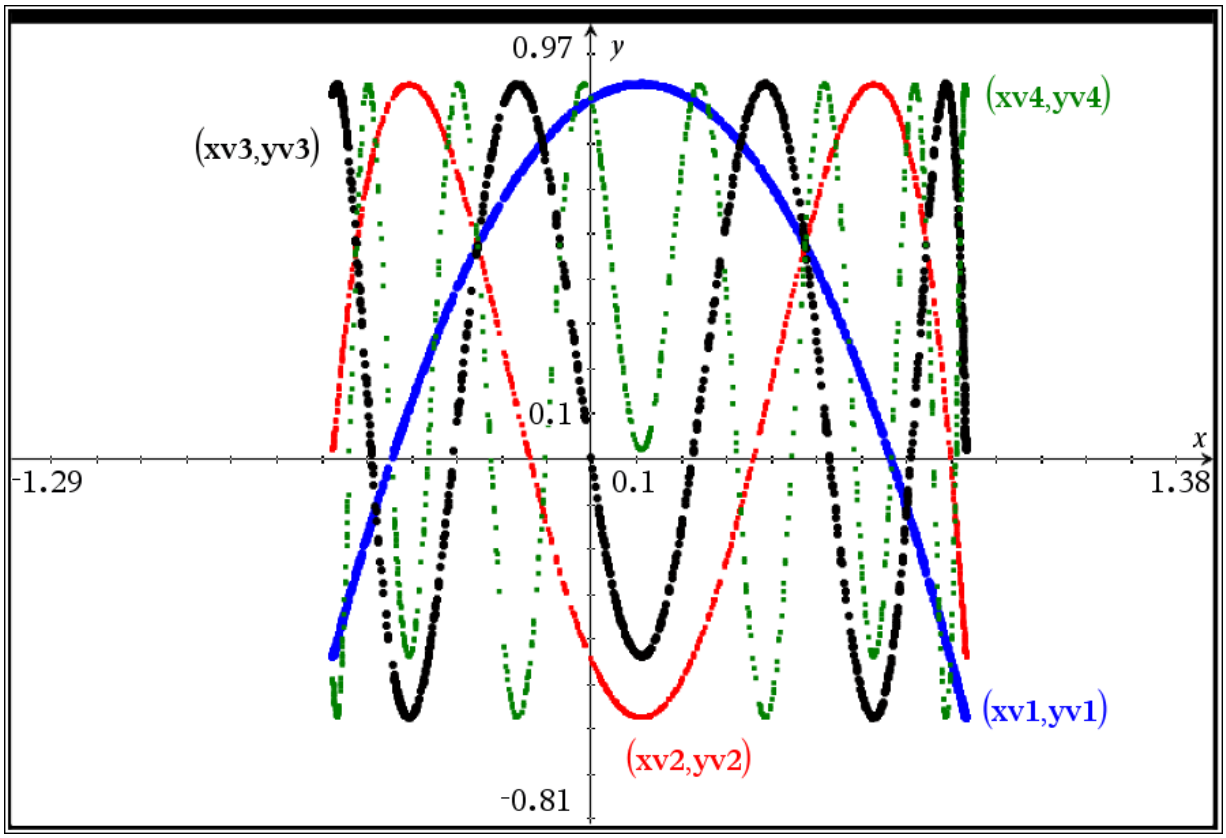


You are invited to compare the plot of the Lyapunov exponent function.



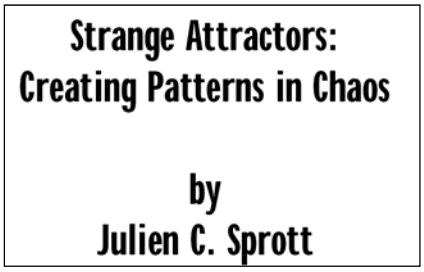
I produced a program *polmap* as a “translation” from DERIVE-syntax to TI-Nspire syntax. Here I show another way to plot the attractor, generating the whole sequence of iterations first followed by gathering the elements for the two lists which are needed for the scatter plot, compare with `polmap` from above.





4 Motivation and Inspiration as well

My motivation and inspiration as well to deal with strange attractors was a book written by Julien C. Sprott:



and his incredible search for strange attractors [3]. This book can be downloaded as a pdf-file free of charge together with program SA.EXE (**S**earch **A**ttractors). This program is a compiled BASIC program (10 pages code). Among the downloadable files are numerous examples for wonderful strange attractors. Sprott's search machine is based on a random choice of coefficients for the polynomials needed for one and more dimensional systems. He encodes the coefficients from -4.5 to 5.0 (with increment = 0.1) with characters (from " " (ASCII 32) to "_" (ASCII 127)). Together with a leading character associated with the type of iteration each attractor can be described in a unique way. This is one example:

$$\text{decode(cd)} := \text{VECTOR}\left(\frac{c - 77}{10},\ c,\ \text{REST(NAME\_TO\_CODES(cd))}\right)$$

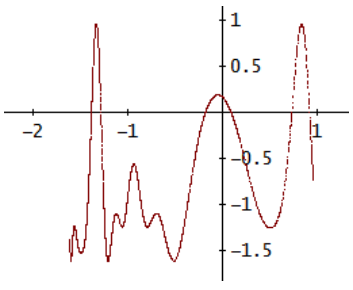
```
decode(CBLCTX) = [-1.1, -0.1, -1, 0.7, 1.1]

decode(EWM?MPMMWMMMM) = [1, 0, -1.4, 0, 0.3, 0, 0, 1, 0, 0, 0, 0]
```

"CBLCTX" describes a one-dimensional quartic map.

```
decode("CBLCTX") = [-1.1, -0.1, -1, 0.7, 1.1]
```

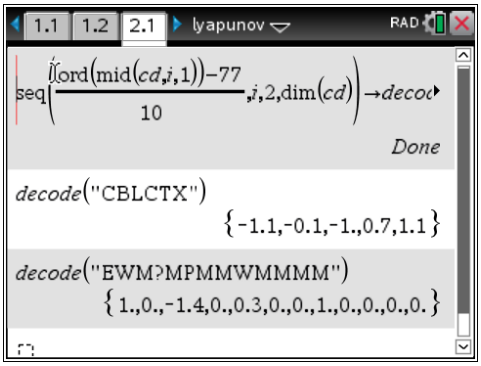
C is the code for a quartic, which reads as $f(x) = -1.1 - 0.1x - x^2 + 0.7x^3 + 1.1x^4$.

```
pol_map([-1.1, -0.1, -1, 0.7, 1.1], 10000, 0.05, 3)
```



will give the plot of the of x_{i+3} vs x_i plot of this attractor.

The second one is for a two-dimensional quadratic mapping (will be presented very soon).



Sprott's codes for the coefficients of the iteration functions go up to 540 characters!!

The "decoding"-function for the TI-NspireCAS is given on the handheld screen.

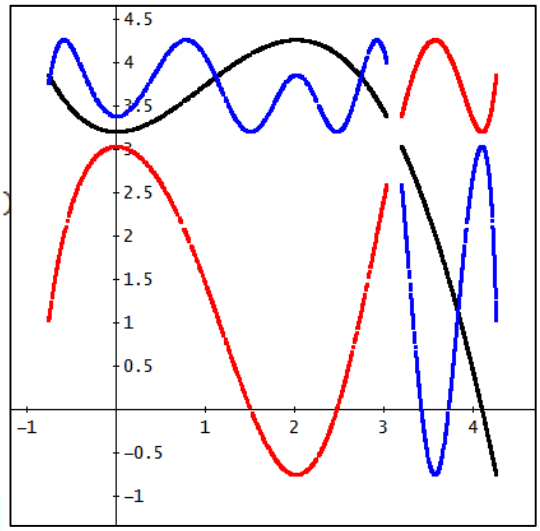
We could be satisfied to reproduce the numerous wonderful plots depicted in Sprott's book together with so many others which are given by their codes only. `pol_map` and some other similar programs would do this up to strange attractors in 3D and 4D space.

This is not sufficient for us and was not sufficient for Julien C. Sprott. He had the idea – and this is the core message of his book – to find own "self-made" strange attractors which had never been seen before.

5 Search for Polynomial Attractors

The trick is to create for polynomials of order *n* sets of randomly chosen coefficients between -5 and +5, compose the respective polynomials, interpret them as iteration function and then calculate their Lyapunov exponent. All sets of coefficients which give a positive LE are connected with an attractor.

```
sap(o, n, tr, t_, xn, x0, f, f1, cf, vals, i, s, dummy, l) :=
  Prog
    [dummy := RANDOM(0), tr := 1, vals := []]
    Loop
      WRITE([tr, DIM(vals)])
      If tr > n exit
      cf := VECTOR(ROUND(100·(9.5·RANDOM(1) - 4.5))/100, k, o + 1)
      f := cf·VECTOR(x^k, k, 0, o)
      f1 := ∂(f, x)
      [i := 1, l := 0]
      x0 := ITERATE(f, x, 0.05, 14 - 2·o)
      Loop
        If ABS(x0) > 100 exit
        If i > 2000 ∧ l > 0
          vals := APPEND(vals, [[l/2000, cf]])
        If i > 2000 exit
        xn := LIM(f, x, x0)
        l := l + LOG(ABS(LIM(f1, x, xn)), 2)
        x0 := xn
        i :+ 1
      tr :+ 1
    RETURN vals
sap(4, 200)
```

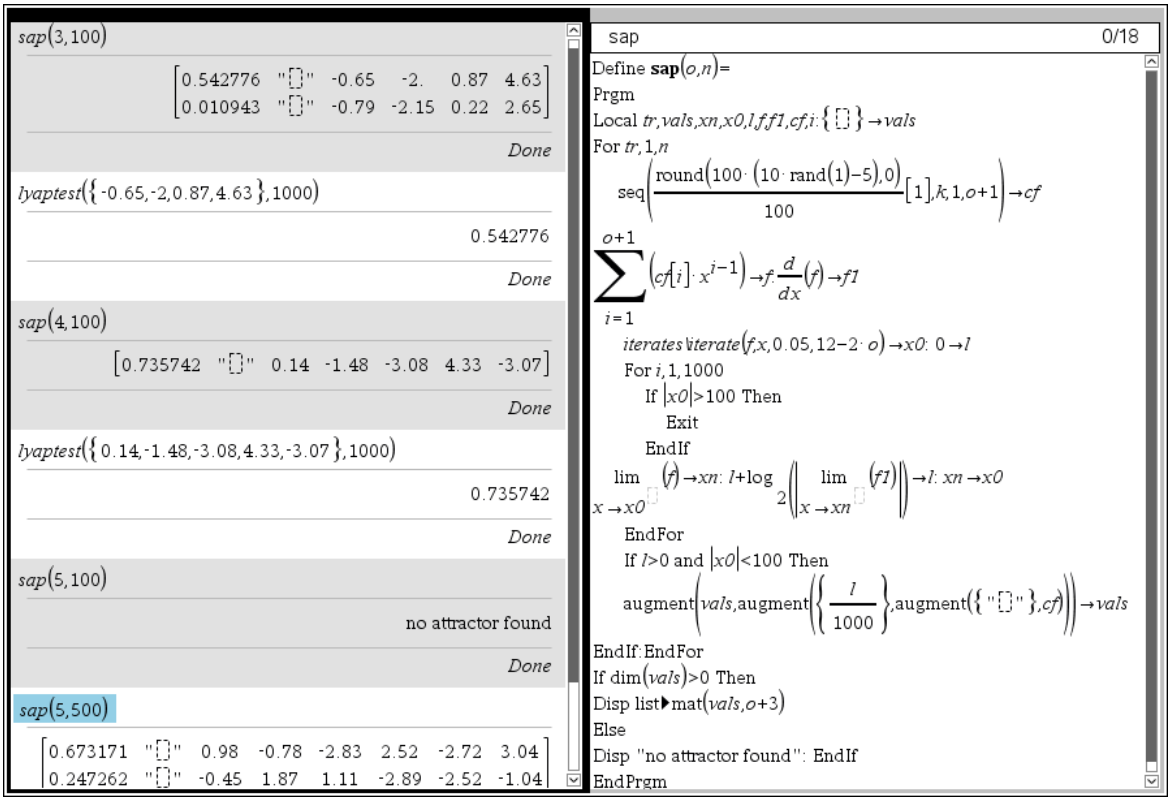



```
⎡ 0.59115   [3.2, -0.01, 0.87, -0.34, 0.02]  ⎤
⎢  0.341   [-0.58, 1.6, 3.94, -3.79, -1.29]  ⎥
⎢  1.137    [-0.5, 1.91, 2.16, -1.79, 0.14]  ⎥
⎢ 0.7173     [0.92, -2, -1.17, 0.97, 0.47]   ⎥
⎢ 0.53857  [0.35, 1.51, -1.97, -3.85, 0.37]  ⎥
⎣ 0.83091  [-1.34, -2.95, -0.49, 0.91, 0.23] ⎦
```

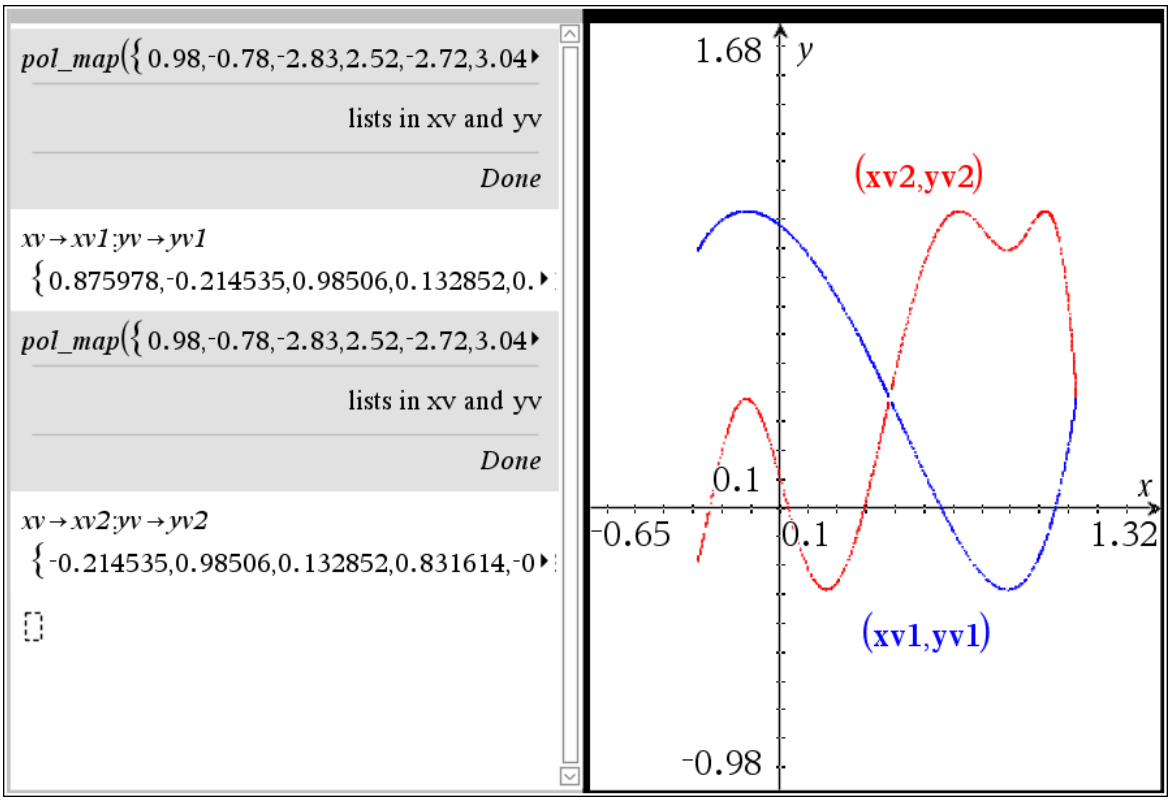
```
pol_map([3.2, -0.01, 0.87, -0.34, 0.02], 2000, 0.05)

pol_map([3.2, -0.01, 0.87, -0.34, 0.02], 2000, 0.05, 2)

pol_map([3.2, -0.01, 0.87, -0.34, 0.02], 2000, 0.05, 3)
```

We start iteration with $x_0 = 0.05$ and perform 2000 iterations. Only coefficient sets with a positive LE are listed. Please notice the `write`-command. You can follow the calculation progress in the status line at left bottom.

Let's finish this first part of our tour by searching for polynomial attractors with TI-Nspire CAS.



It needs some – not too difficult – tricks to obtain an output similar to the DERIVE output. Finally, we convert the resulting list into a matrix with the Les in the first column. The rest of the rows present the coefficients in the same order as before.



.

The real exciting adventure will start in the next part of the tour. Promised!!