THE DERIVE - NEWSLETTER #96

ISSN 1990-7079

THE BULLETIN OF THE

DERIVE®

USER GROUP

+ CAS-TI

Contents:

1 Letter of the Editor

2 Editorial - Preview

3 User Forum
Küderli: Comment on (recursive) graphics programming with Nspire,
Langlotz: Problems with Equations

M. Beaudin, G. Picard, Ch. Trottier & G. Savard
7 Using TI-NspireCAS Technology in Teaching Engineering Mathematics: Ordinary DEs

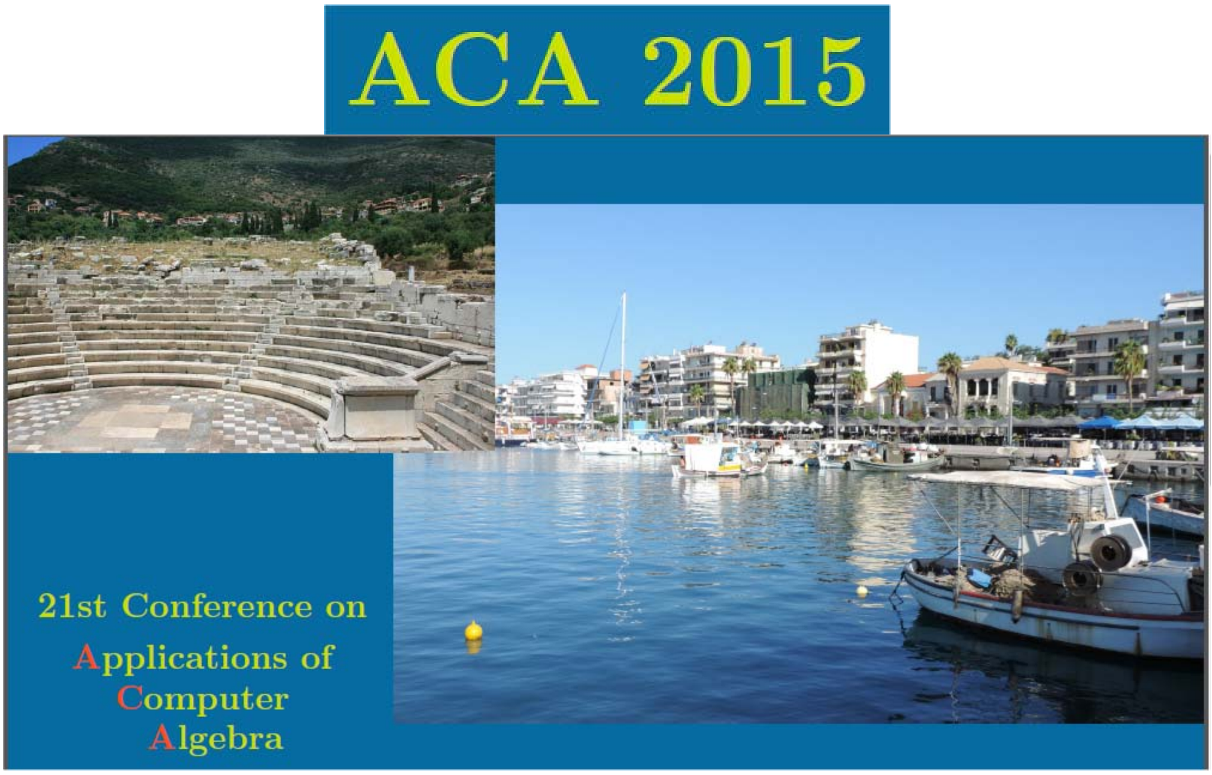
Günter Dreeßen-Meyer & Josef Böhm
20 Spirals of Polygons

Benno Grabinger
28 Stochastik und Experiment & Simulating a Tennis Match

Heinrich Ludwig
31 Complex Numbers on Norwegian Sweaters

Benno Grabinger, Guido Herweyers & Josef Böhm
42 Histograms of discrete probability distributions

January 2015



Computer Algebra in quantum computing and quantum information theory

Human-Computer Algebra Interaction

Computer Algebra in Education

Organizers: Michael Wester, Alkis Akritas, Elena Varbanova, José Luis Galán García and Michel Beaudin

Computer Algebra in Coding Theory and Cryptography

Computational differential and difference algebra

Algebraic and Algorithmic Differential and Integral Operator

Symbolic summation and integration: algorithms, complexity, and applications

Algebraic Graph Theory and its Applications

more sessions are to be announced

July 20 - 23, 2015
Elite City Resort
Kalamata, Greece

http://www.singacom.uva.es/ACA2015/

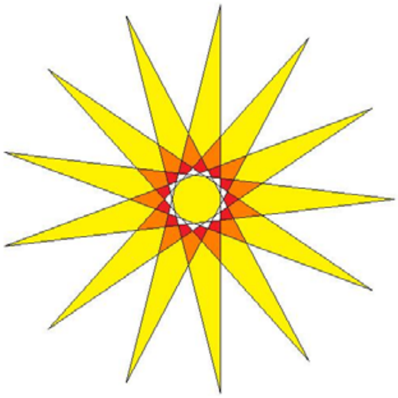
February 27 : Deadline for special session proposals
May 15 : Deadline for talk submissions
June 1 : Notification of acceptance

Dear DUG Members,

Late, very late indeed, I present DNL#96. Due to many reasons, too many to explain, I was not able to finish this DNL earlier. Please apologize and enjoy this issue.

I'd like to express my gratefulness to so many wishes for Christmas and New Year. It was impossible to answer to all mails, many thanks, we were really moved.

What concerns DNL#96, I invite you to notice the announcement of ACA2015, held in Greece this summer. This is a great conference series focussing on Computer Algebra Applications. It is a long tradition to have an Educational Session among other sessions presenting CA applications in various fields. I attended ACA several times and it was always exciting to learn what can be done with CAS – not only in Teaching Mathematics and in Pure Mathematics – not to mention the high quality lectures in the Educational-session.



Take the chance to combine visiting one of the cradles of European Culture (Mykene, Sparta, Epidaurus, Olympia, …) and learning about modern mathematics applications.

We have more Nspire contributions in this DNL, some of them based on earlier DERIVE applications. I received a couple of mails complaining: "Oh, how easy was this with our good old DERIVE!"

Thanks to Heinrich Ludwig we can offer also a pure DERIVE article. His stitch patterns based on complex numbers invite for further investigations and discoveries. It would need another LUA-program to transfer his patterns on the Nspire machine. Please notice his bundle of comments and questions at the end of his article. Heinrich produced his paper some time ago assuming that DERIVE's MOD- and MODS-function don't support complex division. It seems to be that later versions of DERIVE are able to apply MOD and MODS as well for complex numbers.

I finish with my best regards until next DNL. Btw, we should reach DNL#100 this year.

Always Yours

Josef

The late Christmas Star was created by Roland Schröder for the DUG-Community.

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: March 2015

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
Hill-Encryption, J. Böhm, AUT
Simulating a Graphing Calculator in *DERIVE*, J. Böhm, AUT
Do you know this? Cabri & CAS on PC and Handheld, W. Wegscheider, AUT
An Interesting Problem with a Triangle, Steiner Point, P. Lüke-Rosendahl, GER
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
Truth Tables on the TI, M. R. Phillips, USA
Where oh Where is It? (GPS with CAS), C. & P. Leinbach, USA
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
Simulation of Dynamic Systems with various Tools, J. Böhm, AUT
Technical Problems solved with Secondary Maths, W. Alvermann, GER
Pickover's Mygalomorphs and Spiders, A. Roulier & J. Böhm, SUI/AUT
Statistics of Shuffling Cards, Charge in a Magnetic Field, H. Ludwig, GER
What if General Lee Had a Graphing Calculator at Gettysburg? R. Hawkes, USA

and others

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

Invitation from Jean-Jacques Dahan

Enjoy these 3 videos

[22/01/15 17:07:21] DAHAN Jean-Jacques: http://youtu.be/cszoiCBUl3c
[22/01/15 17:07:49] DAHAN Jean-Jacques: http://youtu.be/OiUHp1MRSTk
[22/01/15 17:08:13] DAHAN Jean-Jacques: http://youtu.be/_WW62xi_TIQ

Amitiés
Jean-Jacques

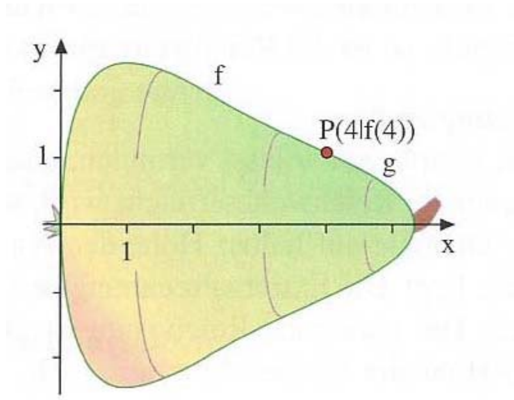
Hubert Langlotz, Germany

Problems with equations

Dear Josef,

Working on one of our examination problems (Modeling with a pear-formed body) we came across a problem with TI-Nspire.

Partial problem f) reads:
$f(x) = 4\cdot\sqrt{x}\cdot e^{-0.5x}$ and $g(x) = \sqrt{a\cdot x + b}$ shall be connected in point *P* so that we have a smooth transition (= differentiable). Find *a* and *b*.



The procedure is well known and leads to a nonlinear system of two equations with two unknowns, *a* and *b*.

$f(x):=4\cdot\sqrt{x}\cdot e^{-0.5\cdot x}:g(x):=\sqrt{a\cdot x+b}$	*Done*
$f1(x):=\frac{d}{dx}(f(x)):g1(x):=\frac{d}{dx}(g(x))$	*Done*
$\{f1(x),g1(x)\}$	$\left\{\left(\frac{2.}{\sqrt{x}}-2.\cdot\sqrt{x}\right)\cdot(0.606531)^x,\frac{a}{2\cdot\sqrt{a\cdot x+b}}\right\}$
⚠ $\text{solve}\left(\begin{cases}f(4)=g(4)\\ f1(4)=g1(4)\end{cases},\{a,b\}\right)$	$\sqrt{4\cdot a+b}=8\cdot e^{-2}$ and $\frac{a}{\sqrt{4\cdot a+b}}=-6\cdot e^{-2}$
⚠ $\text{solve}\left(\begin{cases}f(4)=g(4)\\ f1(4)=g1(4)\end{cases},\{a,b\}\right)$	$\sqrt{a+0.25\cdot b}=0.541341$ and $\frac{a}{\sqrt{a+0.25\cdot b}}=-1.62402$

TI-Nspire seems to be unable to solve this system. How easy was this with good old DERIVE:

```
#1:   f(x) := 4·√x·EXP(- 0.5·x)

#2:   g(x) := √(a·x + b)

#3:   SOLVE(f(4) = g(4) ∧ f'(4) = g'(4), [a, b])

                    -4              -4
#4:   a = - 48·e       ∧ b = 256·e

#5:   a = -0.8791506666 ∧ b = 4.688803555
```

Do you have any idea how to overcome this deficiency? Regards Hubert

Hallo Hubert,

I played a little bit around with your pear. It works – but Nspire needs some support:

It is very interesting that we get the exact result solving the system for {x,b} – but not for {x,a}!! Don't ask me why? Is x here something like a catalyst?

I recommend to add a suitable estimation value for one of the unknowns.

I must agree with you: this does not replace the functionality of DERIVE.

$$\text{solve}\left(\begin{cases} f(4)=g(4) \\ f1(4)=g1(4)\end{cases},\{a,x\}\right)$$

$$x=c5 \text{ and } a=6\cdot\left(\sqrt{b\cdot e^{4}+144}+12\right)\cdot e^{-4} \text{ and } \sqrt{24\cdot\sqrt{b\cdot e^{4}+144}+b\cdot e^{4}+288}=8 \text{ or } x=c5 \text{ and}$$

$$\text{solve}\left(\begin{cases} f(4)=g(4) \\ f1(4)=g1(4)\end{cases},\{x,b\}\right) \qquad x=c6 \text{ and } a=^{-}48\cdot e^{-4} \text{ and } b=256\cdot e^{-4}$$

$$\text{solve}\left(\begin{cases} f(4)=g(4) \\ f1(4)=g1(4)\end{cases},\{x,b\}\right) \qquad x=c7 \text{ and } a=^{-}0.879151 \text{ and } b=4.6888$$

$$\text{solve}\left(\begin{cases} f(4)=g(4) \\ f1(4)=g1(4)\end{cases},\{a,b=1\}\right) \qquad a=^{-}0.879151 \text{ and } b=4.6888$$

$$\text{solve}\left(\begin{cases} f(4)=g(4) \\ f1(4)=g1(4)\end{cases},\{a=1,b\}\right) \qquad a=^{-}0.879151 \text{ and } b=4.6888$$

Hubert sent a second example where TI-Nspire didn't behave as expected. It is a 7th order equation. The real bad fact is not that Nspire cannot solve the equation but that we receive two different solutions. Students will ask: which one is the correct one?

$$f(x):=(9-5\cdot x)^{7} \qquad \textit{Fertig}$$

$$\text{solve}(f(x)-2=0,x) \qquad x=2.05738$$

$$\text{solve}(f(x)=2,x) \qquad x=\frac{^{-}\left(2^{\frac{1}{7}}-9\right)}{5}$$

$$\text{solve}(f(x)=2,x) \qquad x=1.57918$$

$$\text{nSolve}(f(x)-2=0,x) \qquad 1.57918$$

We have the choice!!

DNL: I tried with V200, too. nSolve gives the correct solution immediately. In Exact Mode we get no solution with solve (that's clear) but solve approximated gives 1.8919 – which is a third possible solution?

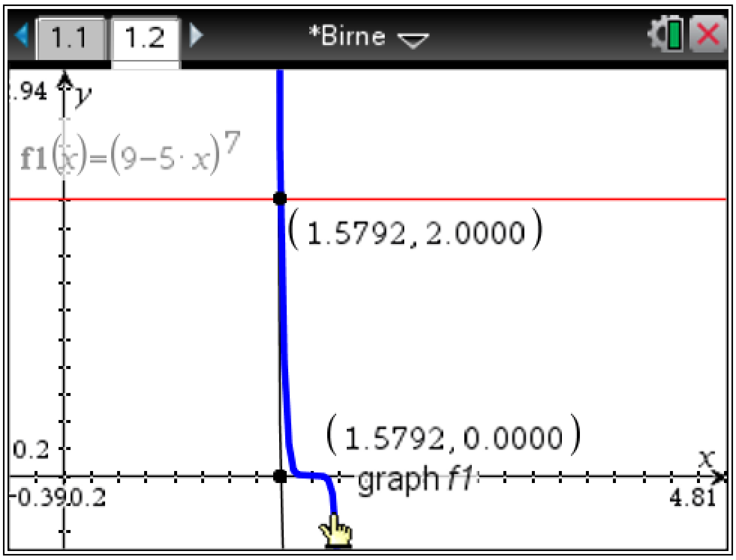
I recommend either applying nSolve (because we know that there is no exact solution) or working with solve and similar to the problem above add an estimation for the solution.

⚠ $\text{solve}\left((9-5\cdot x)^7-2=0,x=1\right)$	$x=1.57918$
⚠ $\text{solve}\left((9-5\cdot x)^7-2=0,x=0\right)$	$x=1.57918$
$\text{nSolve}\left((9-5\cdot x)^7=2,x\right)$	1.57918
$\text{nSolve}\left((9-5\cdot x)^7-2=0,x\right)$	1.57918

See again the difference:

$\text{solve}\left((9-5\cdot x)^7-2=0,x\right)$	$x=2.05738$
⚠ $\text{solve}\left((9-5\cdot x)^7-2=0,x=1\right)$	$x=1.57918$

And what is always i good idea: check the solution – or at least find an estimation value – in the graphic screen:



Christoph Küderli from Switzerland sent an answer on my request posed in the revised DNL#33 how to produce the Snowflake Curve in a recursive way:

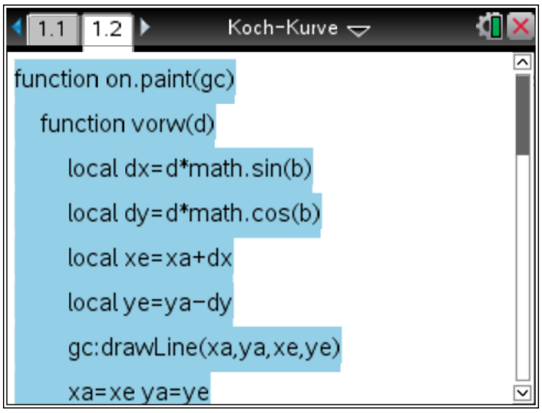
„I wrote a Koch curve program which does not work in the Graphs-coordinate system but in a system with the origin left above and the y-axis directed downwards. I applied turtle graphics integrating two turtle commands for forwards and clockwise rotating (*vorw* and *rechts*).“

The next page demonstrates Christoph Küderli's program and procedure.

Comments on (recursive) graphics programming with TI-Nspire

Christoph Küderli, Switzerland

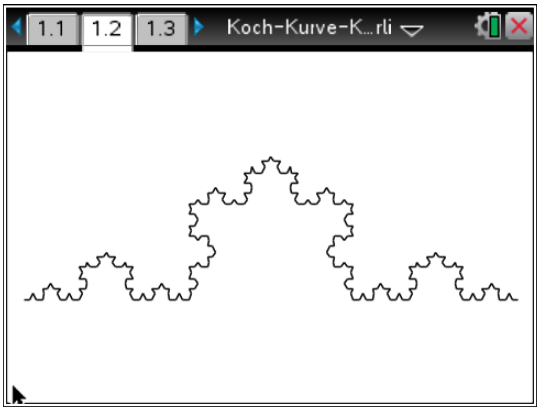
You need the language Lua. There is an editor for LUA which is running on Nspire: Oclua (= on calculator LUA). Run Nspire and open oclua.tns and insert a new Notes page. Here you can write the (LUA-) program. Copy and paste the full text into the first page of oclua.tns (Ctrl C and Ctrl V). The output of the program – the curve – will appear immediately!




```
function on.paint(gc)
  function vorw(d)
    local dx=d*math.sin(b)
    local dy=d*math.cos(b)
    local xe=xa+dx
    local ye=ya-dy
    gc:drawLine(xa,ya,xe,ye)
    xa=xe ya=ye
  end
  function rechts(winkel)
    phi=phi+winkel
    b=math.pi*phi/180 --Bogenmass
  end
  function zeichne(l)
    if l<4 then
      vorw(l)
    else
      l=l/3
```

(continued in the right column)

```
      zeichne(l) rechts(-60)
      zeichne(l) rechts(120)
      zeichne(l) rechts(-60) zeichne(l)
    end
  end
  xa=10 ya=150
  phi=0 b=0
  rechts(90) zeichne(300)
end
```



The oclua-Editor can be downloaded from the TI-website:

http://www.ticalc.org/archives/files/fileinfo/440/44075.html

Using TI-Nspire CAS Technology in Teaching Engineering Mathematics: ODEs

Gilles Picard, Chantal Trottier, Michel Beaudin, Geneviève Savard

Service des enseignements généraux
École de technologie supérieure
Canada
michel.beaudin@etsmtl.ca, gilles.picard@etsmtl.ca, genevieve.savard@etsmtl.ca, chantal.trottier@etsmtl.ca

ABSTRACT

In September 2011, our engineering school has adopted the TI-Nspire CAS CX calculator (and software) for all new students entering bachelor level. With both platforms and integrated facilities, Nspire CAS is becoming a more complete solution than Voyage 200. Although we already started to use this new device in the classroom, we will have a transition period where both the TI Voyage 200 and the new CX handheld calculators can be found on the students' desks. This talk will demonstrate examples of how this technology is used in classrooms when teaching differential equations.

Nspire CX handheld with its faster CPU and extended graphing capabilities gives us even more options for doing math in our classes. Starting with first order equations, the use of an Nspire-CAS calculator makes it easy for students to explore slope fields, numerical solutions with Euler's method or even Picard's approximations for well behaved equations. With Nspire, we can now see graphically how a numerical solution fits when compared to an exact solution. Consider applications of first order ODE's, Newton's law of cooling for example. With Nspire, students can hook up a temperature probe to their calculator and gather data which will then be compared to the theoretical solution.

Looking at second order equations, a CAS calculator can help in computing part of the algebraic solution. In the method of undetermined coefficients, we still teach the classical approach but let students use their handheld to substitute the candidate in the equation and solve the resulting system of equations. When teaching the method of variation of parameters, we show them step by step solution (with the help of their calculator) instead of giving them a unique formula for the solution.

For teachers, having this technology can be very useful for grading exams. One can construct a small program that will yield step by step solution for the variation of parameters solution. If a student makes an error at the beginning (wrong solutions for the homogeneous equation), the teacher could easily obtain what should be the remaining solution for this student. We will demonstrate how simple it is to create such a program.

Our students must learn skills in regards to technology: this is now part of the curriculum and will be presented. In exams or in take-home work, some problems will require the use of Nspire CAS (or Voyage 200) technology. Students should also learn to validate manual results using these tools and understand unexpected results obtained by CAS systems. We will present examples where the benefit of using this approach is obvious. Moreover, the majority of students will spend a lot of energy exploring these math concepts if these tools are used. Everything is more visual and they love to play with technology, as long as they are told which direction to follow.

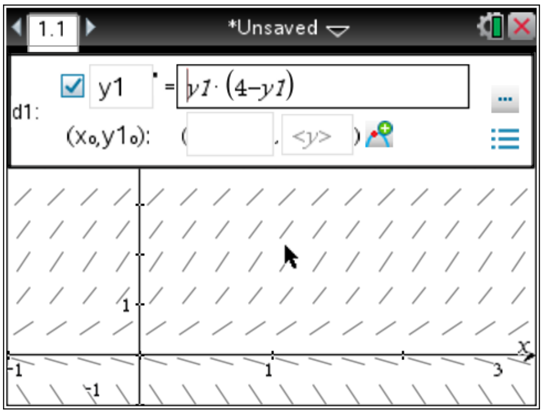
1. First order equations

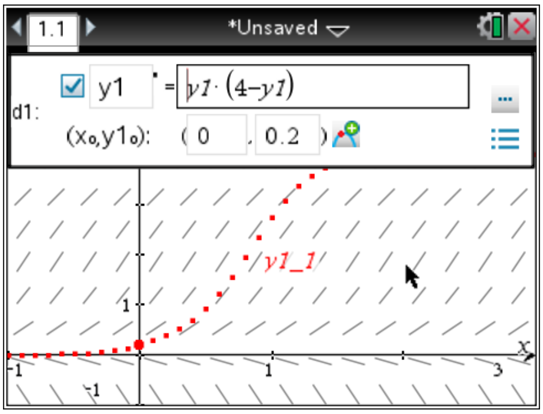
Consider the differential equation $\dfrac{dy}{dx} = y \cdot (4 - y)$, with the initial condition $y(0) = 0.2$.

We want to find the value of y when $x = 1$.

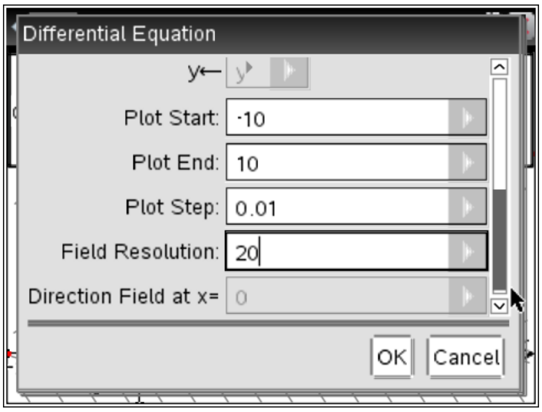
We will graph the slope field of this equation, the numerical solution (Euler method) and the exact solution, all in the same window. Also we will approximate the solution with a polynomial, using Picard's iterations; we will graph Picard's 3rd iteration.
We will also show the required value $y(0)$ estimated by Euler method (in a table of values) and compare it with the exact value.

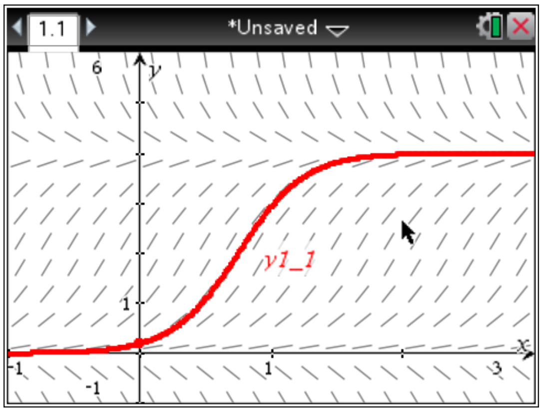
The left screen[1] gives the slope field, and the right screen gives the slope field with the numerical solution, by Euler, using a plot step of 0.1:





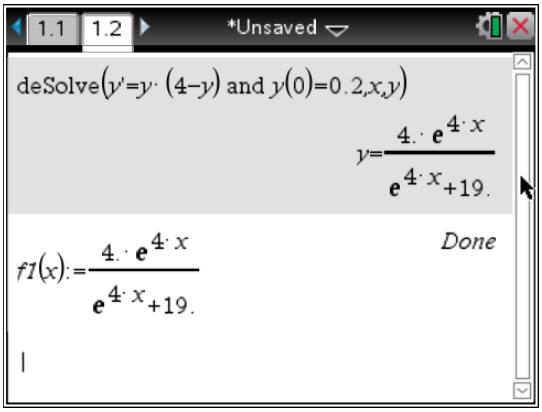
Again, the slope field and the numerical solution, but with a plot step of 0.01:

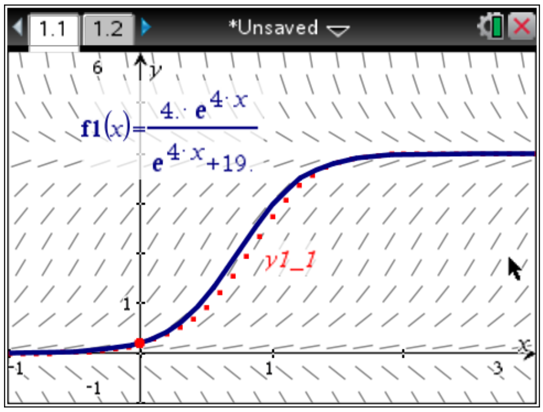




Let's now compute the exact solution; this is possible because the differential equation is a simple (separable) equation and the calculator has a command for finding exact solution of well-behaved 1st and 2nd order equations.

We graph this exact solution, comparing it with the Euler solution. We come back with a plot step of 0.1 for the numerical solution since it will help us see clearly the difference between the 2 solutions:



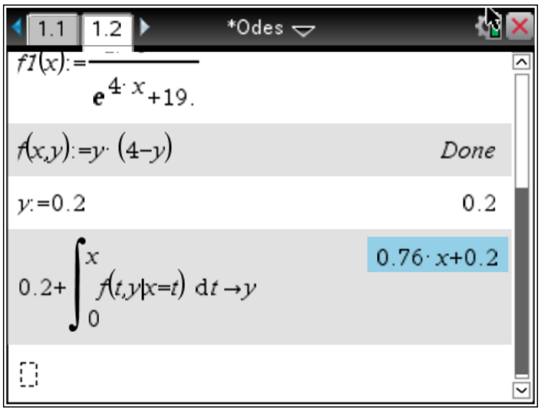


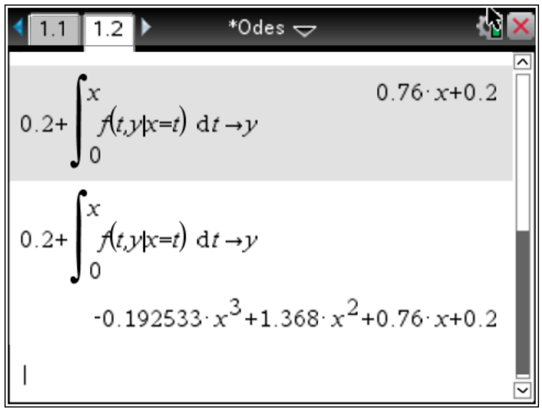
Picard's iterations

Let's recall that given the differential equation $\frac{dy}{dx} = f(x,y)$ subject to the initial condition $y(x_0) = y_0$, the formula for Picard's iterations is $y_{n+1}(x) = y_0 + \int_{x_0}^{x} f(t, y_n(t))dt$.

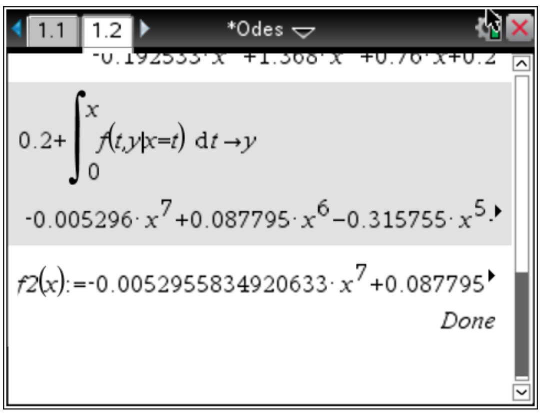
In our example we have $f(x,y) = y\cdot(4-y),\ x_0 = 0$ and $y_0 = 0.2$.

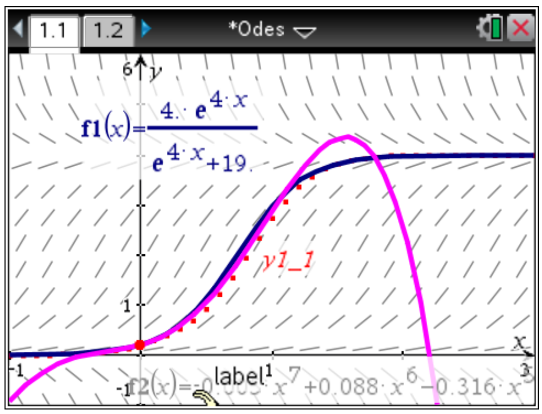
On the left screen, we first store the function $f(x,y)$, and then we prepare to get the iterations. On the right screen we press ENTER several times to iterate this formula.





Our goal is to graph an approximate polynomial solution (see the right screen below). We chose Picard's 3rd iteration which is a 7th degree polynomial.





Looking at this last graph, we see that, as expected, the polynomial solution does a good job near $x = 0$ but is much in error when the x-value grows greater that 1.

Table of values

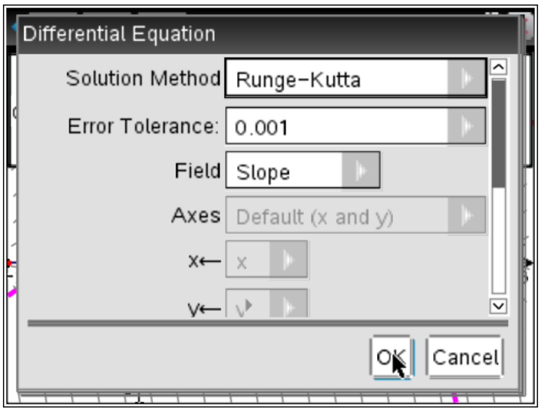
We now want to compare the numerical values obtained by these different methods. That's why we use a table of values:

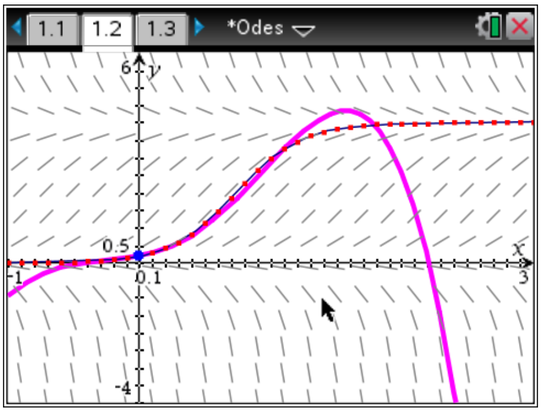
x	f1(x):= 4.*e^(4*x)/(..	f2(x):= -0.00529...	DE1.y1... interpolate..
0.	0.2	0.2	0.2
0.25	0.500644	0.495147	0.442041
0.5	1.12002	1.05128	0.925589
0.75	2.05555	1.87151	1.73505
1.	2.96737	2.85069	2.71611

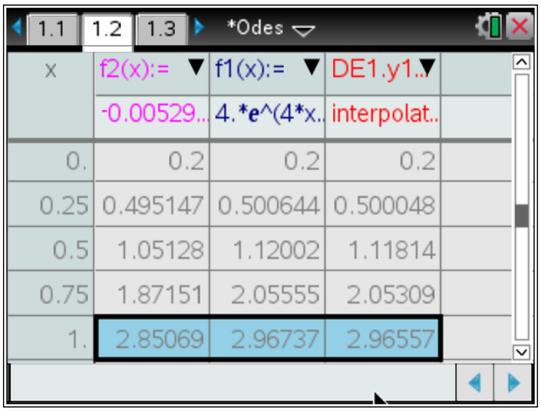
2.0555467320468

The first column gives the exact (6 digits) values of the solution, so that $y(1) = 2.96737$; the second column is the Picard's 3rd iteration : 2.85069, and the third column gives the value obtained by Euler, using a step size of 0.1 : 2.71611.

There is another numerical method we can use, which is far better than Euler, and this is the special version of the Runge-Kutta method implemented in TI-Nspire (a 3rd order adaptive step version). As we can see below, if we change the solution method to "Runge-Kutta" in the "Differential Equation Settings" window, the values in the 3rd column of our table will change. We see below that the estimated value for $y(1)$, by Runge-Kutta, is the same as the exact solution when values are rounded up to two decimals. When rounding up to three decimals, the difference between the two values is only 0.001. Thus we observe a good improvement in the numerical solution and we notice that the graph obtained with this method is very close to the one of the exact solution!







x	f2(x):= -0.00529..	f1(x):= 4.*e^(4*x..	DE1.y1.. interpolat..
0.	0.2	0.2	0.2
0.25	0.495147	0.500644	0.500048
0.5	1.05128	1.12002	1.11814
0.75	1.87151	2.05555	2.05309
1.	2.85069	2.96737	2.96557

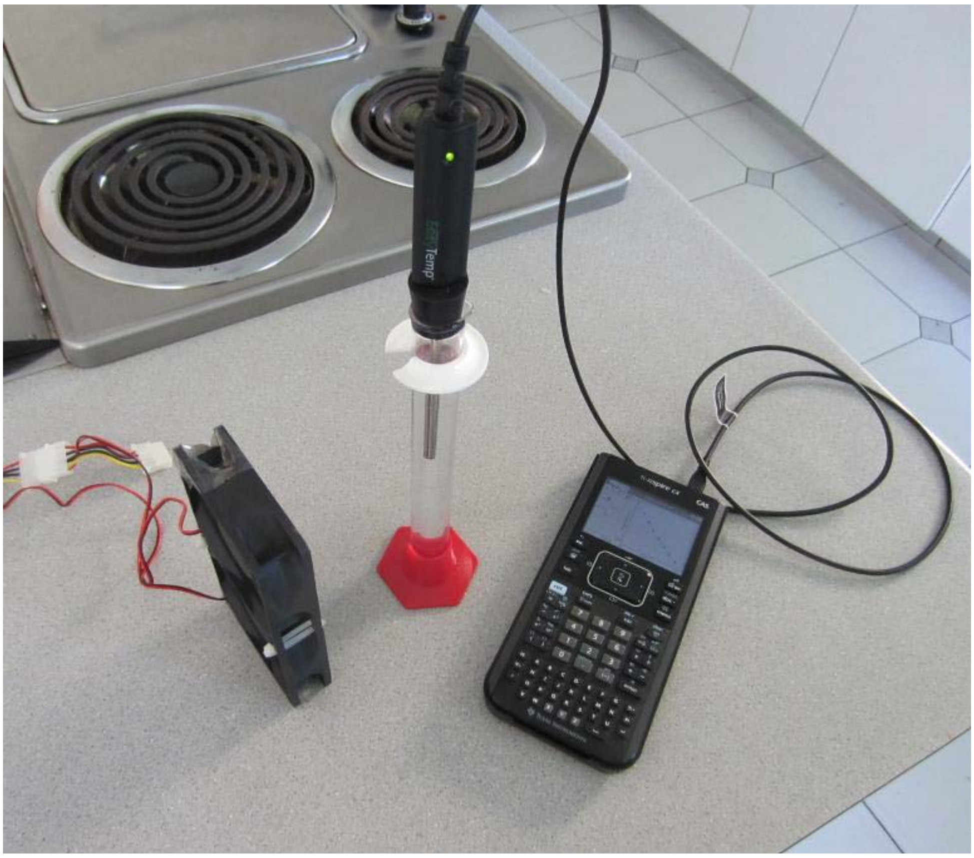
2. Newton's law of cooling, using a temperature probe

Here is a classic example of an application of 1st order differential equations in real life; we want to analyze the temperature of a liquid that is cooling down.

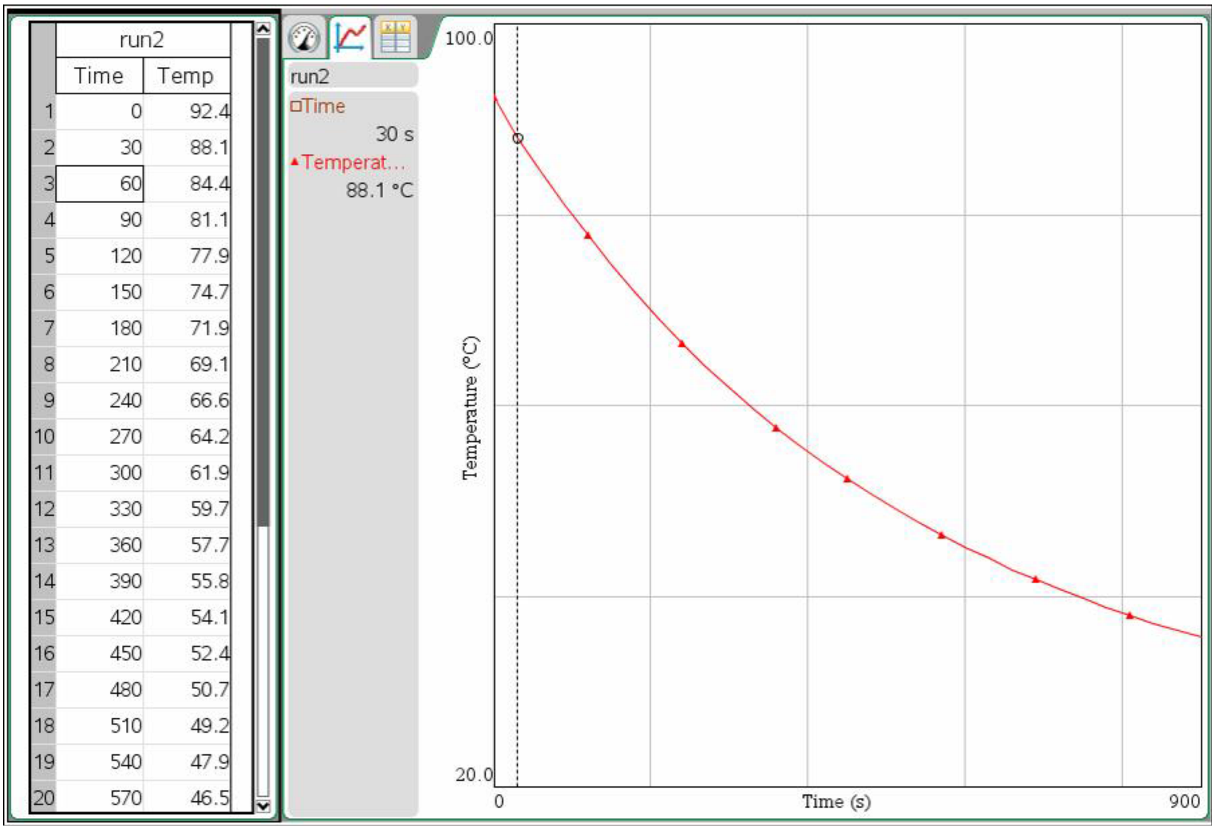
Newton's law states that this situation will be modelled by the equation $\frac{dT}{dt} = k \cdot (T - T_E)$, where T is the temperature of the liquid, T_E is the ambient temperature (supposed constant) and k is a parameter depending on the environment of the experiment. The solution is $T(t) = T_E + C \cdot e^{kt}$ where C is the difference between initial temperature and ambient temperature.

With the new TI-Nspire CX CAS, we can get real experimental data in the calculator using a temperature-probe. Knowing the ambient temperature, we will use two of the observations to get the theoretical solution and see how it compares to the data collected.

Here is the experiment: We took temperatures of water cooling, from 100ºC. You can see on the picture the test tube filled with hot water, the probe connected to a calculator and a fan to keep ambient temperature constant. We set the calculator to take a temperature reading every 30 seconds, for 15 minutes.



The results are shown below. As we can see, we started a little late; water had already cooled below 95ºC. The ambient temperature was 24.0 ºC.

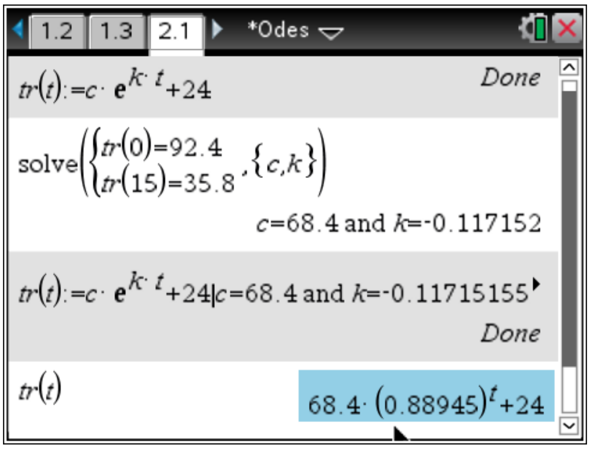


	Time	Temp
1	0	92.4
2	30	88.1
3	60	84.4
4	90	81.1
5	120	77.9
6	150	74.7
7	180	71.9
8	210	69.1
9	240	66.6
10	270	64.2
11	300	61.9
12	330	59.7
13	360	57.7
14	390	55.8
15	420	54.1
16	450	52.4
17	480	50.7
18	510	49.2
19	540	47.9
20	570	46.5

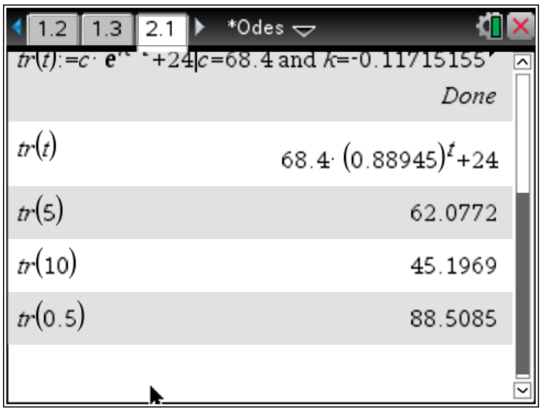
On the left we have the temperature every 30 seconds. On the right, the points are some of these values (about 1 over 4), and the software automatically connects them.

To find these values, we solve a system of 2 equations. Let's take data at $t = 0$ and 900 seconds. This is an arbitrary choice, we could choose other points which would give us similar, but not necessarily the exact same values for C and k.

We work with time in minutes instead of seconds, so that k is not too small. We have, taking tr for temperature: $tr(0) = 92.4$ and $tr(15) = 35.8$.



Thus we find that $tr(t) = 24 + 68.4{\cdot}e^{-0.117152t}$. This is the solution according to the theoretical model. We use this function to check whether it gives the results we had obtained by experiment at 300, 600 and 30 seconds.



Let's recall the temperatures obtained by experiment:

$T(300) = 61.9$

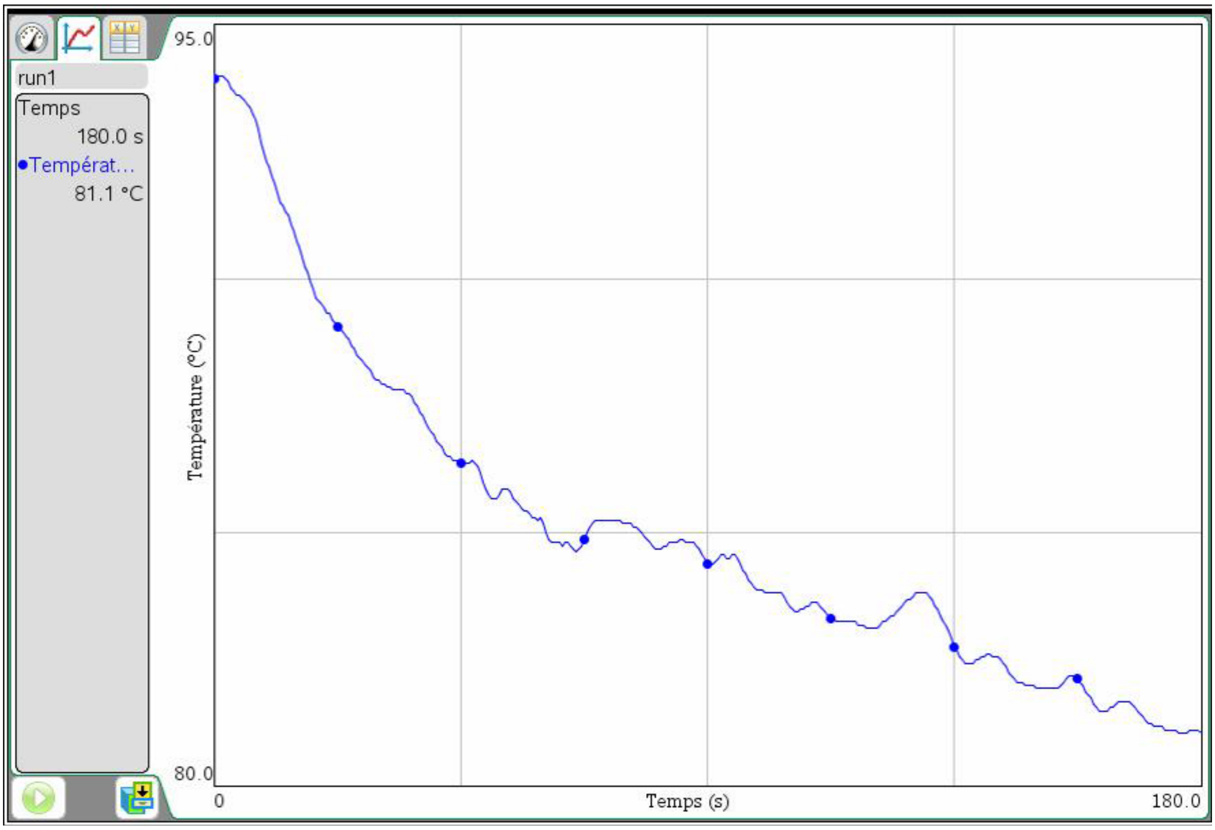
$T(600) = 45.2$

and

$T(30) = 88.1$

We see that the results are quite satisfying, the theoretical solution giving values near what was observed.

It is important, while we do the experiment, not to move or touch the recipient and we need to have a fan to insure a constant temperature for the surrounding environment. Look at the curve below. It represents the results of another experiment we had done earlier where we had moved the probe, and touched the sides of the recipient with it while having no fan.



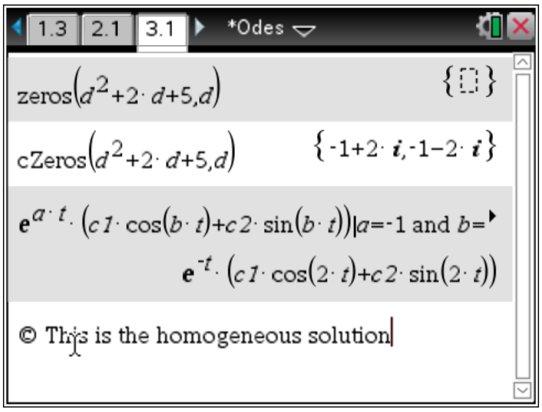
3. Second order equations

A – Undetermined coefficients

Let's consider the equation $\frac{d^2y}{dt^2}+2\frac{dy}{dt}+5y=61\cdot\sin(9t)$.

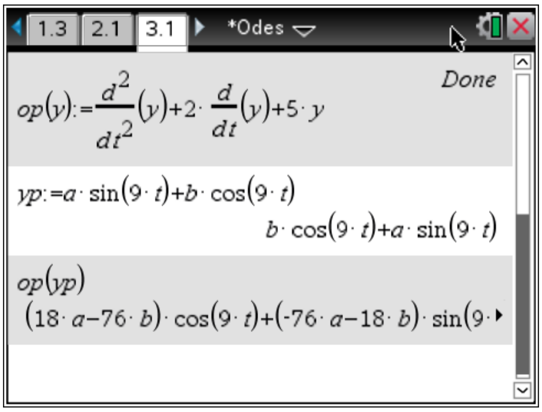
We want our students to solve this equation by the method of undetermined coefficients. We show them how their CAS calculators can help in doing some parts of the solution.

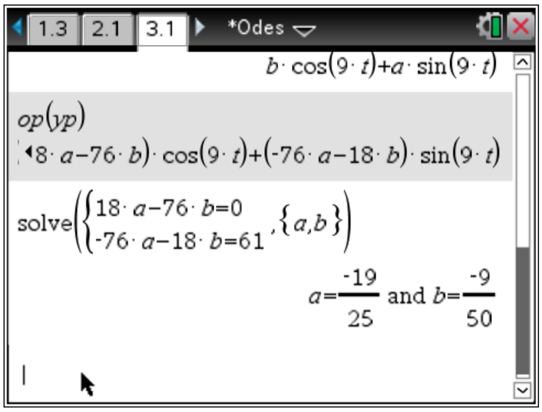
First they have find the roots of the characteristic polynomial and write the homogeneous solution.

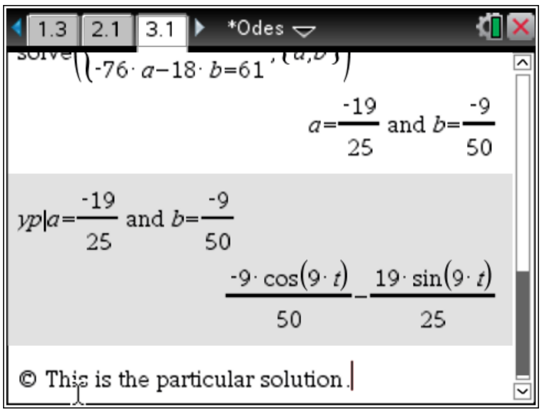


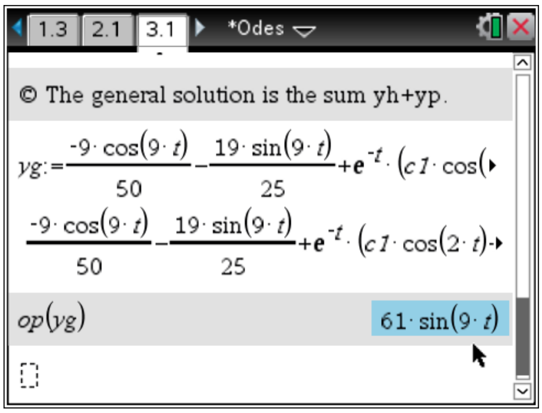
They then write the candidate for the particular solution, this candidate is A sin (9t) + B cos(9t). Now, the problem is to find the values of A and B so that this candidate gives a solution when substituted in the differential equation.

They can substitute it in the equation using the operator `op( )` we have shown them to automate this substitution, and then solve the resulting system of equations.





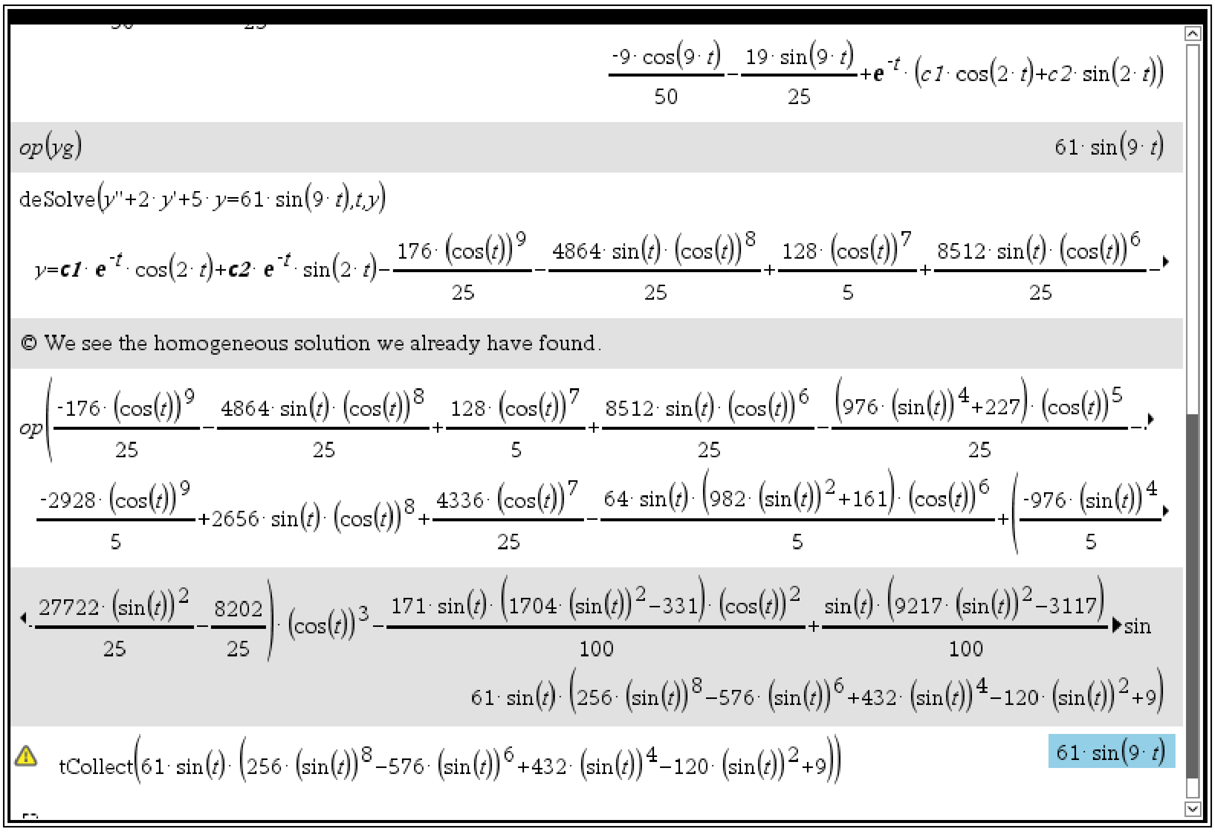




Thus they obtain the general solution $y_g = \frac{-9}{50}\cos(9t) - \frac{19}{25}\sin(9t) + e^{-t}\cdot\left(c_1\cdot\cos(2t) + c_2\cdot\sin(2t)\right)$, and they even can verify that the solution is good by substituting it in the differential equation, using op( ).

As we saw earlier, we could use a command of the calculator to get the solution in one step. The next screen shows this but, notice the huge difference between our previous solution and the general solution obtained with the DeSolve command.

Using the op( ) operator, we can verify that this long solution is indeed a solution of the differential equation, with a *little* use of trigonometric identities :



(Comment: The DERIVE users will remember: ►sin (end of the sixth row) works similar to Trig Powers: Sines in the Mode settings, Josef).

B – Variation of parameters (Lagrange method)

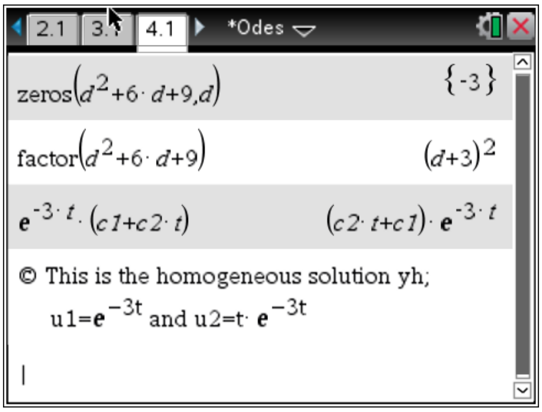
We ask the students to solve the differential equation $\frac{d^2y}{dt^2} + 6\frac{dy}{dt} + 9y = 45t^4 \cdot e^{-3t}$, using the method of variation of parameters.

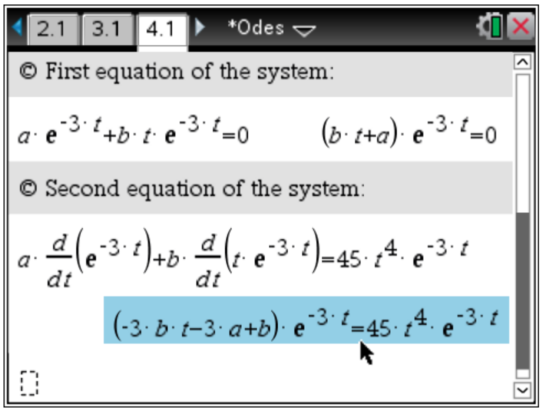
First they have to find the roots of the characteristic polynomial and write the homogeneous solution: $y_h = c_1 u_1 + c_2 u_2$. According to this method, a particular solution will be of form $y_p = L_1 u_1 + L_2 u_2$, where L_1 and L_2 can be variable (functional) coefficients. They have to find L_1 and L_2.

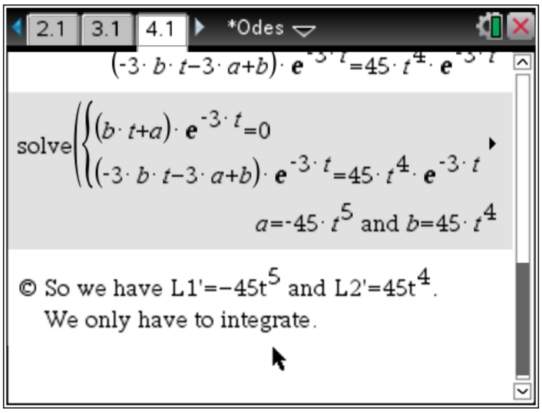
The system of equations to solve is $\begin{cases} L_1' u_1 + L_2' u_2 = 0 \\ L_1' u_1' + L_2' u_2' = 45t^4 \cdot e^{-3t} \end{cases}$.

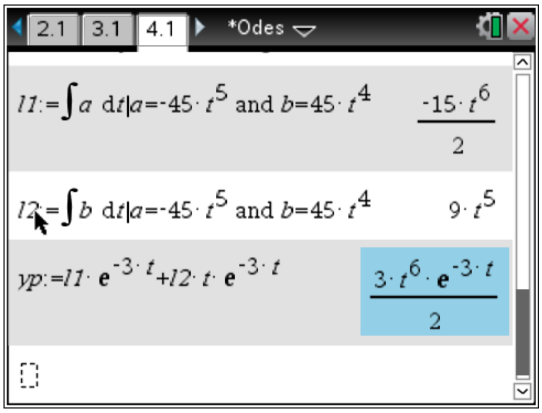
They obtain L_1' and L_2', integrate to find L_1 and L_2, and give the desired solution.

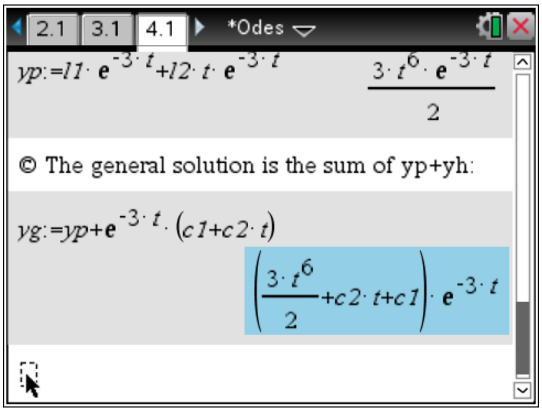
We could ask them to do all this by manual calculation (classic textbook approach) but let's get a little help from the CAS calculator.

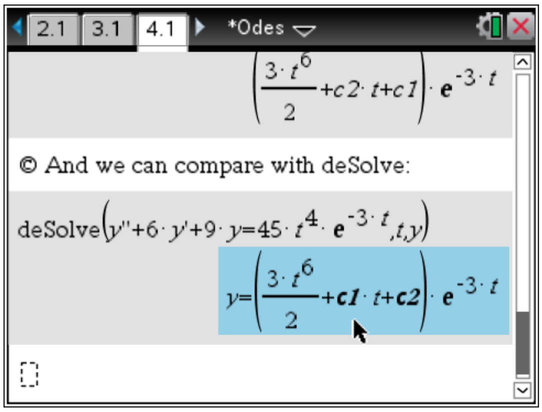












We could give them a direct formula to get this result but we prefer a step by step approach.

We made a program with NspireCAS to test these results, even if students make a mistake, for example if they take the wrong homogeneous solution. This program outputs the intermediate steps we want to see in the students' solutions.

deSolve(y''+6·y'+9·y=45·t^4·e^(-3·t),t,y)

$$y=\left(\frac{3\cdot t^{6}}{2}+c1\cdot t+c2\right)\cdot e^{-3\cdot t}$$

$vap\left(e^{-3\cdot t},t\cdot e^{-3\cdot t},45\cdot t^{4}\cdot e^{-3\cdot t},t\right)$

$$\text{Wronskian} = \begin{bmatrix} e^{-3\cdot t} & t\cdot e^{-3\cdot t} \\ -3\cdot e^{-3\cdot t} & (1-3\cdot t)\cdot e^{-3\cdot t}\end{bmatrix}$$

$$\text{L1' and L2'} = \begin{bmatrix} -45\cdot t^{5} \\ 45\cdot t^{4}\end{bmatrix}$$

$$\text{L1 and L2} = \begin{bmatrix} \frac{-15\cdot t^{6}}{2} \\ 9\cdot t^{5}\end{bmatrix}$$

$$\text{yp} = \frac{3\cdot t^{6}\cdot e^{-3\cdot t}}{2}$$

$$\text{yg} = \left(\frac{3\cdot t^{6}}{2}+c2\cdot t+c1\right)\cdot e^{-3\cdot t}$$

Done

vap 9/12

```
Define LibPub vap(u,v,f,x)=
Prgm
© Order of parameters: u1, u2, f(x), x
Local a,s,w
w:=[u  v ; d/dx(u)  d/dx(v)]
Disp "Wronskian = ",w
s:=propFrac(w^-1·[0;f])
Disp "L1' and L2' = ",s
propFrac(∫s dx)→a
Disp "L1 and L2 =",a
yp:=propFrac(a[1,1]·u+a[2,1]·v)
Disp "yp = ",yp
yg:=propFrac(yp+c1·u+c2·v)
Disp "yg = ",yg
EndPrgm
```

If they make a mistake in finding the homogeneous solution, for example if they forget the middle term $6\frac{dy}{dt}$, they will have $y_h = c_1\cos(3t)+c_2\sin(3t)$ and their particular solution will not be good either, but we still can grade their solutions.

Done

$vap\left(\cos(3\cdot t),\sin(3\cdot t),45\cdot t^{4}\cdot e^{-3\cdot t},t\right)$

$$\text{Wronskian} = \begin{bmatrix} \cos(3\cdot t) & \sin(3\cdot t) \\ -3\cdot\sin(3\cdot t) & 3\cdot\cos(3\cdot t)\end{bmatrix}$$

$$\text{L1' and L2'} = \begin{bmatrix} -15\cdot t^{4}\cdot e^{-3\cdot t}\cdot\sin(3\cdot t) \\ 15\cdot t^{4}\cdot e^{-3\cdot t}\cdot\cos(3\cdot t)\end{bmatrix}$$

$$\text{L1 and L2} = \begin{bmatrix} \left(\frac{5\cdot t^{4}}{2}+\frac{10\cdot t^{3}}{3}+\frac{5\cdot t^{2}}{3}-\frac{5}{27}\right)\cdot e^{-3\cdot t}\cdot\cos(3\cdot t)+\left(\frac{5\cdot t^{4}}{2}-\frac{5\cdot t^{2}}{3}-\frac{10\cdot t}{9}-\frac{5}{27}\right)\cdot e^{-3\cdot t}\cdot\sin(3\cdot t) \\ \left(\frac{-5\cdot t^{4}}{2}+\frac{5\cdot t^{2}}{3}+\frac{10\cdot t}{9}+\frac{5}{27}\right)\cdot e^{-3\cdot t}\cdot\cos(3\cdot t)+\left(\frac{5\cdot t^{4}}{2}+\frac{10\cdot t^{3}}{3}+\frac{5\cdot t^{2}}{3}-\frac{5}{27}\right)\cdot e^{-3\cdot t}\cdot\sin(3\cdot t)\end{bmatrix}$$

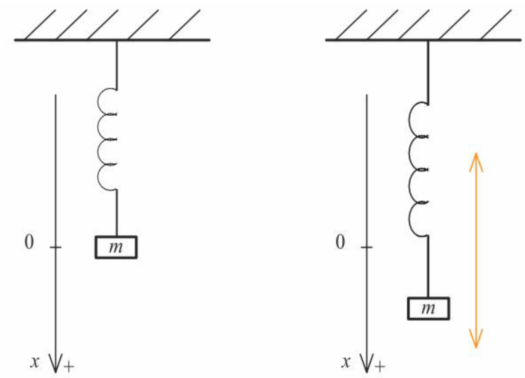
$$\text{yp} = \left(\frac{5\cdot t^{4}}{2}+\frac{10\cdot t^{3}}{3}+\frac{5\cdot t^{2}}{3}-\frac{5}{27}\right)\cdot e^{-3\cdot t}$$

$$\text{yg} = c1\cdot\cos(3\cdot t)+c2\cdot\sin(3\cdot t)+\left(\frac{5\cdot t^{4}}{2}+\frac{10\cdot t^{3}}{3}+\frac{5\cdot t^{2}}{3}-\frac{5}{27}\right)\cdot e^{-3\cdot t}$$

Done

4. Mass-spring system

The harmonic motion is often modelled with a mass-spring system, as we see in this picture:



An object of mass m is hanging at the end of a spring.

We set an x-axis such that $x = 0$ at rest, and x is positive when the spring is stretched.

Let's take a mass-spring system that leads to the equation

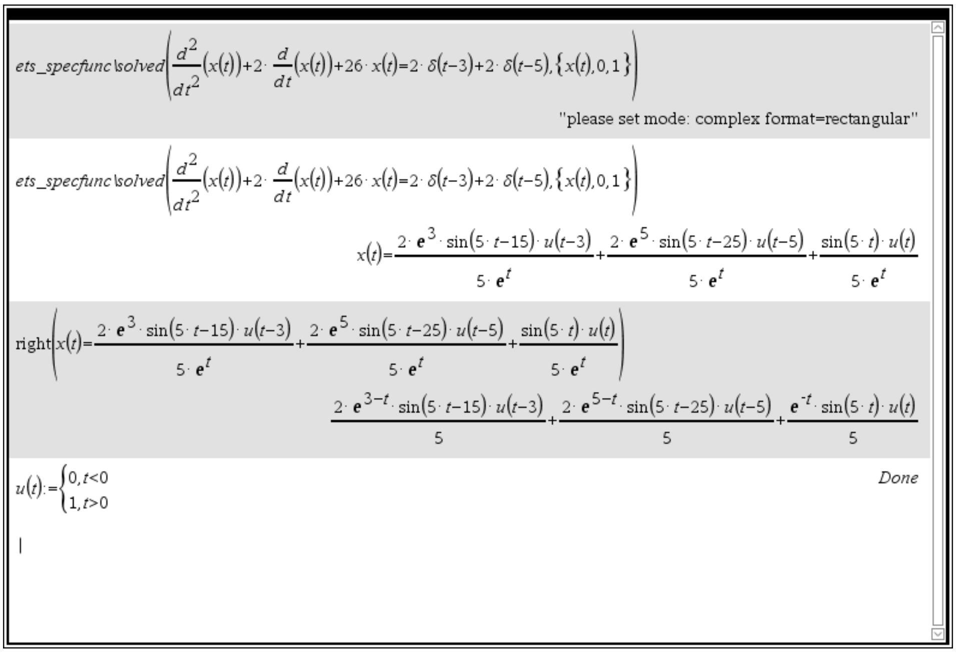
$$x'' + 2x' + 26x = 2\delta(t-3) + 2\delta(t-5), \quad x(0) = 0,\, x'(0) = 1$$

As we saw, x is the position of the mass so that x' is its velocity and x'' is its acceleration. And the Dirac Delta function $\delta(t-a)$ could be visualized as hammer blow given on the object at $t = a$, so that we have two hammer blows, the first-one after three seconds, and the second-one after five seconds.

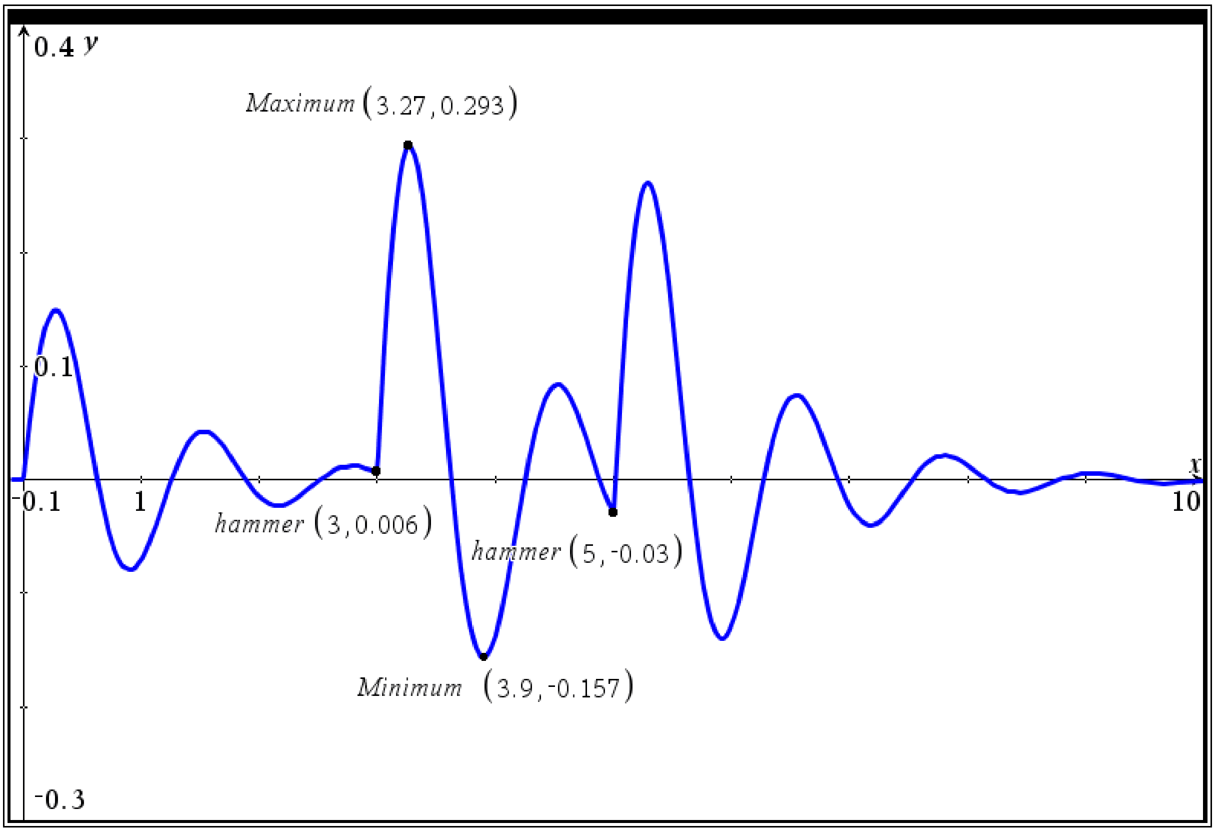
We ask our students to solve this differential equation, with Laplace transforms. They are allowed to use a program they installed on their calculators. This program from Lars Fredericksen was originally done for the TI-89 and it was named « Laplace ». It has been modified for TI-Nspire by Philippe Fortin (Lycée Louis Barthou – Pau)[2], and re-named « specfunc ». Finally the comments have been translated in French by Chantal Trottier, and the program received the name of « ETS-specfunc ».

Once they get the solution, we ask them to graph it. This presents some difficulties.

They have to get the solution, then define the function $u(t)$, which is the unit-step function (Heaviside function); its value is 0 if $t < 0$, and 1 if $t > 0$. They don't want this function to be understood by the calculator when they use this program, so they will have to delete it after use. Or they could create a permanent Heaviside function in *step*(t) and replace in the output solution all occurrences of the function $u(t)$ by *step*(t).



It remains only to graph it, and to answer some questions to demonstrate their comprehension, such as « Is the object going down or up when the first hammer blow occurs? », « And the second? », « What is the greatest distance from equilibrium, when does the object reach it, and is the spring stretched or compressed at that point? »



These were some examples of how we can benefit from the use of CAS technology when teaching a basic course in differential equations. We still teach the classic topics using a paper and pencil approach for simple problem. We then show how technology can be used to aid in long and tedious computations. This gives us time to explore even more mathematics with students.

[1] All solutions and screens were done in the original paper using TI-NspireCA CX, with O.S. 3.1. I updated all screens using the latest Nspire version which is O.S. 3.4, Josef

[2] Philippe is a very good contributor to this website in France: http://www.universs-ti-nspire.com/

You can download two respective pdf-files how to use the library (in french):

http://seg-apps.etsmtl.ca/nspire/documents/transf%20Laplace%20prog.pdf

https://cours.etsmtl.ca/seg/mbeaudin/MAT265/BlocLabosTI/Mat%20235%20et%20TI-Ex.supp.-Fonctions.pdf

The libraries specfunc and ets_specfunc as well are among the zipped MTH96 files.

Spirals of Polygons

Günter Dreeßen-Meyer & Josef Böhm

Last September I received a mail from Germany asking how to produce spirals of polygons. Günter Dreeßen-Meyer from Berlin complained that this was very easy done with DERIVE applying one single VECTOR-command.

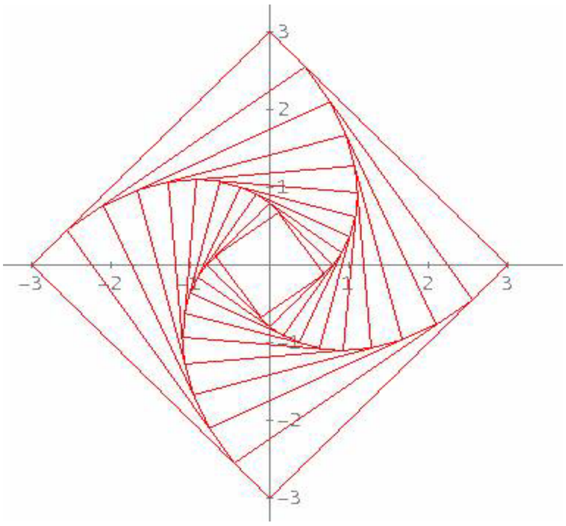
Students could apply matrix calculations first and then individually proceed with the so generated graphs (colours, …). But unfortunately he and hiscolleagues were unable to create the spirals with TI-NspireCAS.

His mail ended with:

Do you have any idea or advice how to achieve the spirals with TI-Nspire?

Best regards,

Günter Dreeßen-Meyer, Berlin



A Spiral of (DERIVE-) Squares

I wrote back that I would try to find a solution. I must admit that I didn't remember at this time that Maria Koth had treated these spirals in DNL#33. So I started from the very beginning and my first solution might be interesting but it is not very elegant at all. The problem I encountered was – again – missing the plotting commands of Voyage 200. The only one way plotting segments is via one – or even more – scatter diagrams …

I sent file Poly_Spiralen.tns: parameters must be entered on a Notes page.

© Rotation of point st by angle θ in positive direction − stretching or compressing by factor fa

$rot(\theta,st,fa):=st\cdot fa\cdot\begin{bmatrix}\cos(\theta) & \sin(\theta)\\ -\sin(\theta) & \cos(\theta)\end{bmatrix}$ *Done*

© Generating the coordinates of a regular n−gon with initial vertex st

$poly\_x(st,n):=\mathrm{seq}\left(\cos\left(\frac{2\cdot\pi\cdot k}{n}\right)\cdot st[1,1]-\sin\left(\frac{2\cdot\pi\cdot k}{n}\right)\cdot st[1,2],k,0,n\right)$ *Done*

$poly\_y(st,n):=\mathrm{seq}\left(\sin\left(\frac{2\cdot\pi\cdot k}{n}\right)\cdot st[1,1]+\cos\left(\frac{2\cdot\pi\cdot k}{n}\right)\cdot st[1,2],k,0,n\right)$ *Done*

© Rule for finding the next point coordinates (f = q(n,a) = fa in the Notes)

$q(n,\alpha):=\dfrac{\cos\left(\frac{\pi}{n}\right)}{\cos\left(\frac{\pi}{n}-\alpha\right)}$ *Done*

© Rule for finding the next point coordinates (f = q(n,a), see fa in the Notes)

Enter the parameters on this Notes page:

starting point ***pt***, number of vertices of the regular polygon ***n*** and angle of rotation ***α***

$\mathbf{pt}:=[0\ \ 2] \blacktriangleright [0\ \ 2],\ \mathbf{n} \blacktriangleright 5.,\ \boldsymbol{\alpha} \blacktriangleright 0.314159$

We can install sliders for n and for α.

That's all, don't enter any other data, calculation will be done in the background!

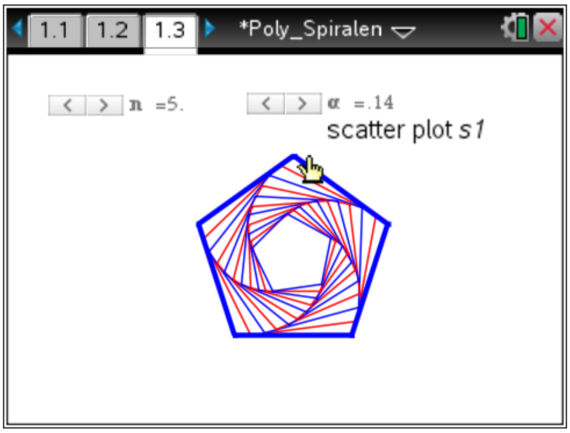
$\mathbf{xw}:=\mathbf{poly\_x}(\mathbf{pt},\mathbf{n}) \blacktriangleright \{0.,^-1.90211,^-1.17557,1.17557,1.90211,0.\}$

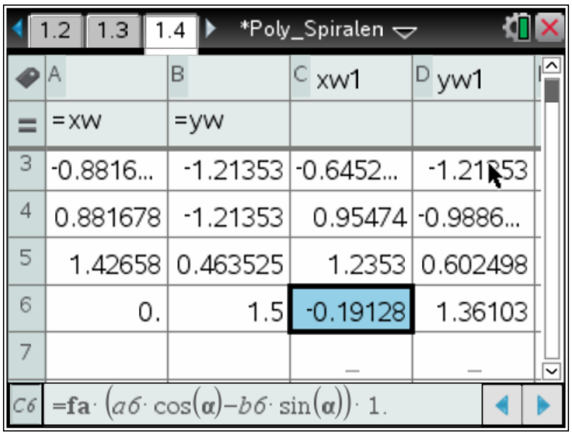
$\mathbf{yw}:=\mathbf{poly\_y}(\mathbf{pt},\mathbf{n}) \blacktriangleright \{2.,0.618034,^-1.61803,^-1.61803,0.618034,2.\}$

$\mathbf{fa}:=\mathbf{q}(\mathbf{n},\boldsymbol{\alpha}) \blacktriangleright 0.850651$

xw and yw are the vertices of the first polygon. The coordinates of the next 10 generations are given in the – flexible – spreadsheet page – up to a 20-gon (xw1,yw1) to (xw10,yw10). Then 11 scatter diagrams are defined – one for each generation.

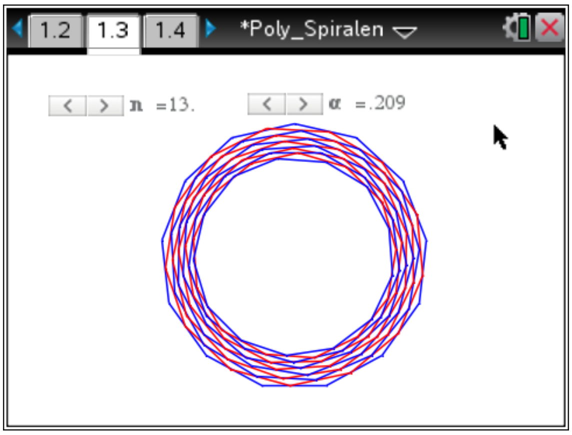
I print the handheld view to save space.

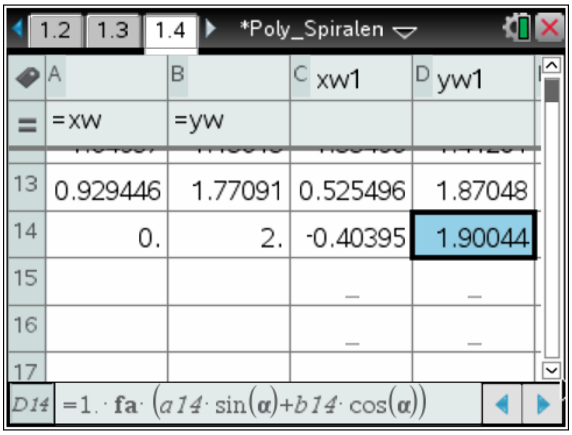




	A	B	C xw1	D yw1
=	=xw	=yw		
3	-0.8816...	-1.21353	-0.6452...	-1.21353
4	0.881678	-1.21353	0.95474	-0.9886...
5	1.42658	0.463525	1.2353	0.602498
6	0.	1.5	-0.19128	1.36103
7				

C6 $=\mathbf{fa}\cdot(a6\cdot\cos(\alpha)-b6\cdot\sin(\alpha))\cdot 1.$





	A	B	C xw1	D yw1
=	=xw	=yw		
13	0.929446	1.77091	0.525496	1.87048
14	0.	2.	-0.40395	1.90044
15				
16				
17				

D14 $=1.\cdot\mathbf{fa}\cdot(a14\cdot\sin(\alpha)+b14\cdot\cos(\alpha))$

This is in fact not very comfortable – compared with the DERIVE-VECTOR procedure.

I will try a LUA-script – but this will last a little while.

From Berlin:

Your proposal looks quite interesting, in particular storing the vertices in the spreadsheet.
I will inspect it more accurately.

Many thanks and best regards

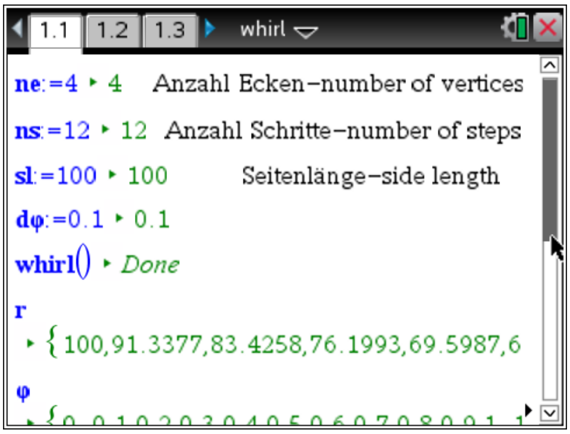
Günter Dreeßen-Meyer

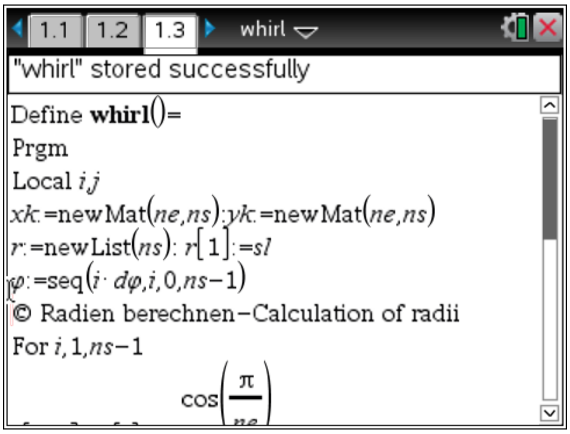
Dear Mr. Dreeßen-Meyer

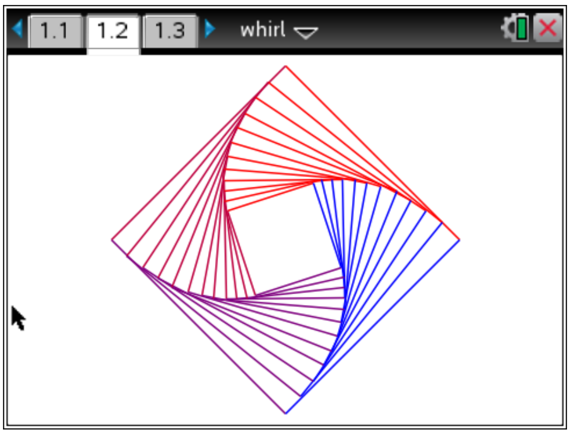
This is the first solution of your problem worked out with the script language LUA. It was sent by a Swiss DUG-member, Alfred Roulier. You can see how excellent cooperation is working in the DUG-community, btw, many thanks Alfred.

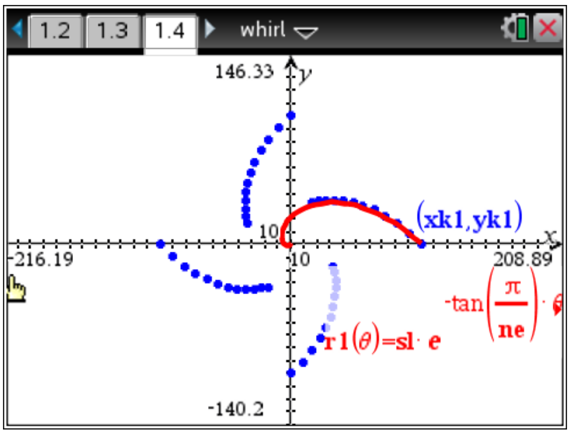
I am working on a LUA-script calculating and plotting the single polygons stepwise instead of generating the whole point matrix – which was Alfred's way.

This is Alfred Roulier's file whirl.tns (handheld screen shots). The important part of the LUA-script is given on the next page. Program whirl() generates the numeric values which are then used by LUA.









The vertices are lying on a logarithmic spiral with the centre of the polygon as pole.
(from A. Roulier, whirl.doxc inspired by an article in Scientific American, 1965)

```
…
function on.paint(gc)
ne=(var.recall("ne") or  1)  -- import number of vertices
ns=(var.recall("ns") or  1)  -- import number of steps
--Punktmatrizen definieren
 xk = {} ; yk = {}
   for j=1,ne do
     xk[j] = {} ; yk[j] = {}  -- create a new row
     for i=1,ns do
       xk[j][i] = 0 ; yk[j][i] = 0
     end
   end
-- import point coordinates
   xk=(var.recall("xk" or 2)) ; yk=(var.recall("yk" or 2))

-- plot polygons
  for i=1,ns do
     for j=1,ne-1 do
       gc:setColorRGB(255*(1-(j-1)/ne),0,255*(j-1)/ne)
       gc:drawLine(w/2+xk[j][i],h/2-yk[j][i],w/2+xk[j+1][i],h/2-yk[j+1][i])
     end
     gc:setColorRGB(0,0,255)
     gc:drawLine(w/2+xk[ne][i],h/2-yk[ne][i],w/2+xk[1][i],h/2-yk[1][i])
  end
 end
...
```

Again a mail from Berlin:

Dear Mr Böhm,

many thanks for the mail and the files.

How easy was it with DERIVE! And the issue mappings and matrices was reproducible and investigable by students. It's a pity that these fine properties have not been taken over to NSpire!

I should start programming. Many things seem to become possible. I see that you have been grasped by programming, too.

Regards from the North Sea,

Yours G. D.-M.

Dear Mr. Dreeßen-Meyer,

finally it worked. I was able to transfer my iterative algorithm into a LUA-program (see the attached file Poly_Spirals_LUA.tns)

Two nice problems for the students could be offered.

How to generalize the mapping that

(1) the direction of the spirals are reversed, and

(2) the spirals are not inscribed, but circumscribed? (possible answers see page 27)

Best regards, Josef B.

Die Eingabe der Parameter *n*, *α* und *its* erfolgt über die Schieberegler auf page 3

Enter the parameters *n*, *α* and *its* on page 3 using the sliders

Startpunkt – initial point

pt:=[0 3] ▸ [0 3]

n ▸ 8.

its ▸ 6.

α ▸ 47.

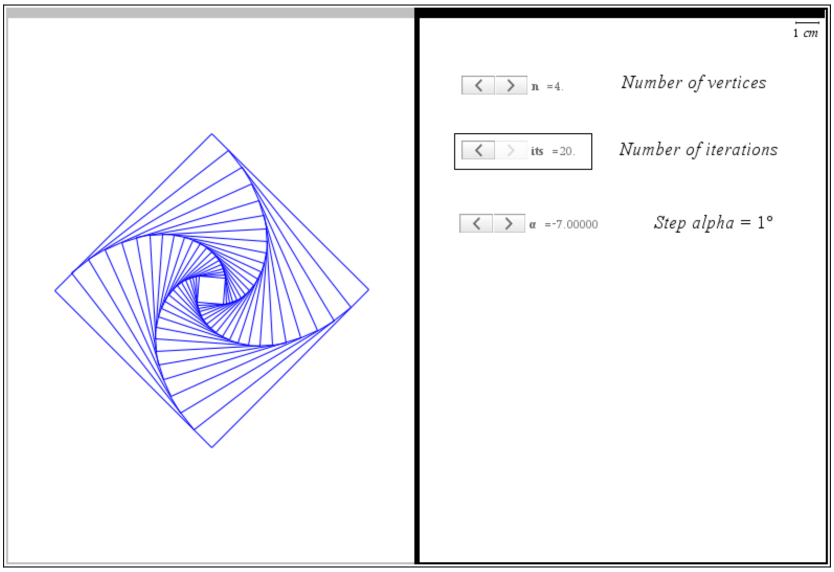
Nichts mehr eingeben – Nothing to enter!

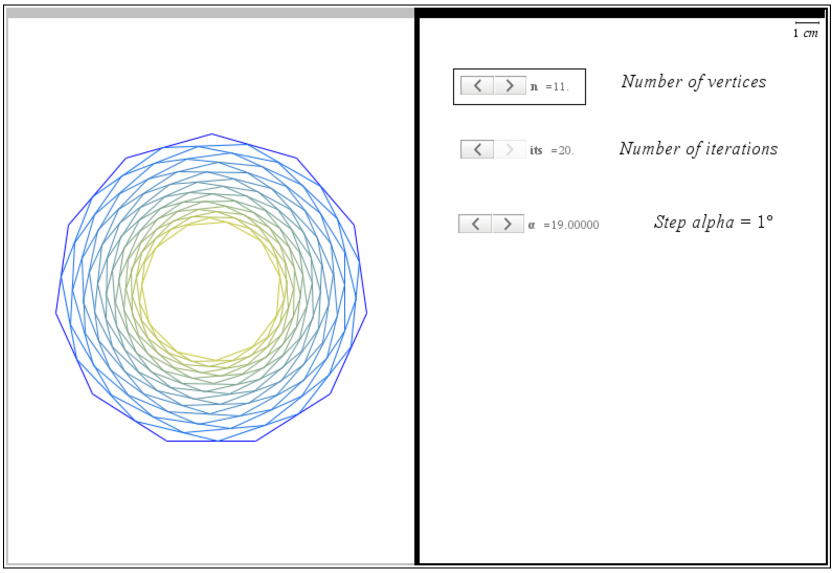
xw:=round(1000·(50·poly_x(pt,n)),3)/1000 ▸ {0.,-106.066,-150.,-106.066,0.,106.066,150.,106.066,0.}

yw:=round(1000·(50·poly_y(pt,n)),3)/1000 ▸ {150.,106.066,0.,-106.066,-150.,-106.066,0.,106.066,150.}

Daten für das Script (nichts eingeben) – Data for the script, nothing to enter:

a:=approx$\left(\frac{\alpha\cdot\pi}{180}\right)$ ▸ 0.820305, fa:=approx(q(n,a)) ▸ 1.0153





This is the complete LUA-script:

```
platform.apilevel='2.2'
function on.construction()
  h=platform.window:height(); w=platform.window:width()
end
function on.resize(width,height)
  h=height; w=width
end
function on.enterKey()
  platform.window:invalidate()
end
function on.paint(gc)
  n=var.recall("n");a=var.recall("a")
  xw = {}; yw={}
    for i=1,n+1 do
       xw[i]=0;yw[i]=0
    end
  xw=var.recall("xw"); yw=var.recall("yw")
  for i=1,n do
  gc: setColorRGB(0,0,255)
  gc: drawLine(w/2+xw[i],h/2-yw[i],w/2+xw[i+1],h/2-yw[i+1])
  end
  its=var.recall("its");fa=var.recall("fa")
  xn={};yn={}
  for i=1,n do
     xn[i]=0;yn[i]=0
  end

-- the iterative algorithm
  for j=1,its do
   for i=1,n+1 do
     xn[i]=fa*(xw[i]*math.cos(a)-yw[i]*math.sin(a))
     yn[i]=fa*(xw[i]*math.sin(a)+yw[i]*math.cos(a))
   end
   for i=1,n do
      gc: setColorRGB(0,0,255)
      gc: drawLine(w/2+xn[i],h/2-yn[i],w/2+xn[i+1],h/2-yn[i+1])
      xw[i]=xn[i];yw[i]=yn[i]
   end
   xw[n+1]=xn[n+1];yw[n+1]=yn[n+1]
  end
-- end of iteration
end
```

Exchanging the gc:setColorRGB(0,0,255) by gc: setColorRGB(10*j,100+5*j,255-10*j) results in a nice shading of the nested polygons.

The final solution – without LUA came in from Berlin:

… many thanks for your files and your intellectual efforts. I will study the program code.

and some days later

Hello Mr. Böhm and Karl-Heinz Keunecke,

I have solved our problem (matrices and spirals) in another way.

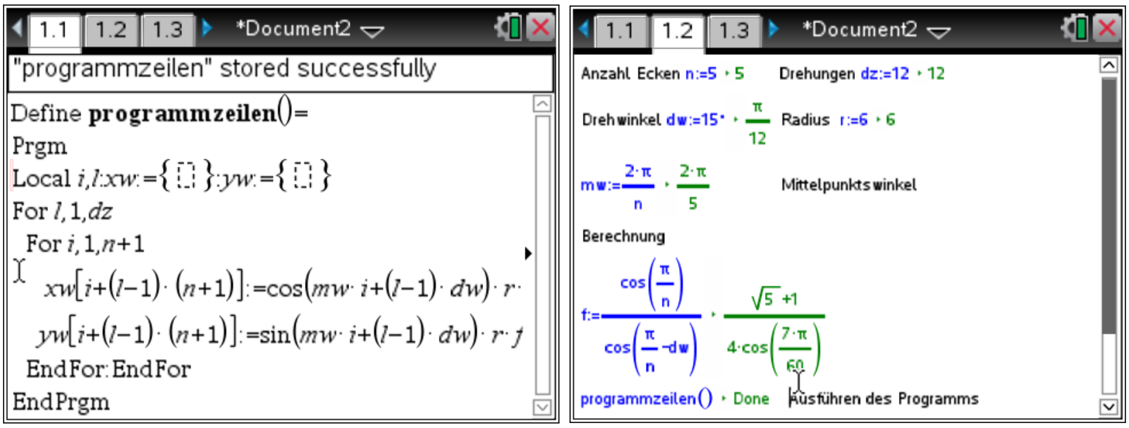
The vertices of a polygon are defined by polar coordinates. The coordinates of the initial polygon are stored together with the coordinates of the rotated figures in **one** list for the x-values and a second one for the y-values.

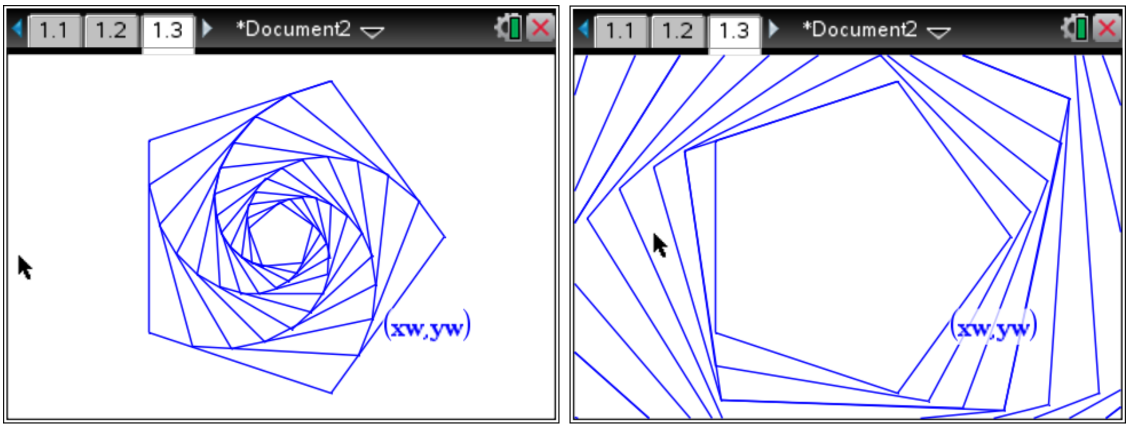
The scatter plot of the lists gives the plot. The attributes must be adopted. Calculation of both lists is performed within the Nspire environment (not in a LUA-program).

Best regards from Berlin

Günter D.

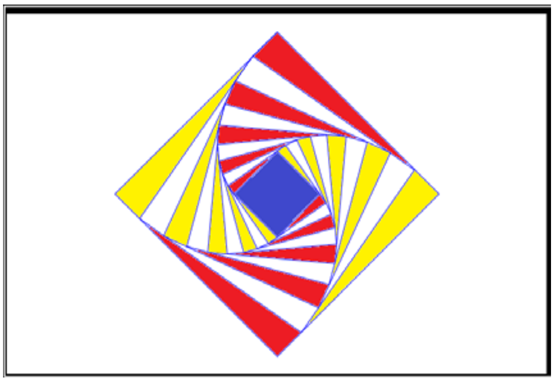
I intend to write a short article for the TI-News.





This is the family of pentagons with dw = 80°.

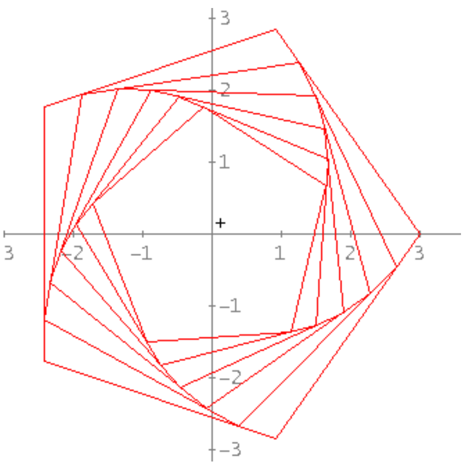
Günter D.-M. derives the formula for the factor *f* in the same way as Maria Koth did (DNL#33). He shows a pretty picture accomplished by one of his students adding a nice pattern to a spiral of squares.



As this is still the *DERIVE* newsletter I'd like to add some *DERIVE* made plots, too. On page 23 I posed possible problems for the students.

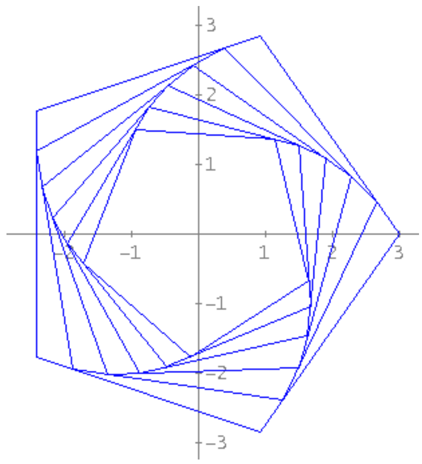
$$\mathrm{POL\_ROT}(r,\ n,\ \phi,\ j,\ d := 1) := \mathrm{ITERATES}(\mathrm{S}(n,\ \phi)^{d}\cdot\mathrm{ROT}(\phi,\ p),\ p,\ \mathrm{POLY}(r,\ n),\ j)$$

$$\mathrm{POL\_ROT}\left(3,\ 5,\ \frac{\pi}{18},\ 5\right)$$



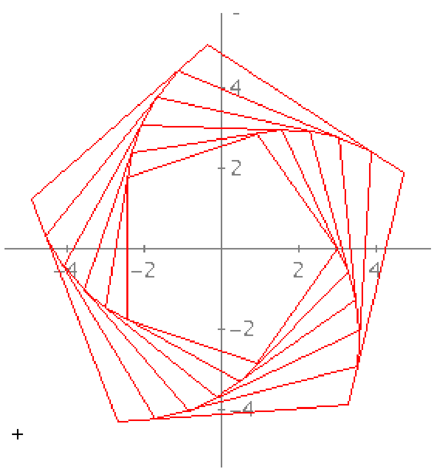
inscribed

$$\mathrm{POL\_ROT}\left(3,\ 5,\ -\frac{\pi}{18},\ 5\right)$$



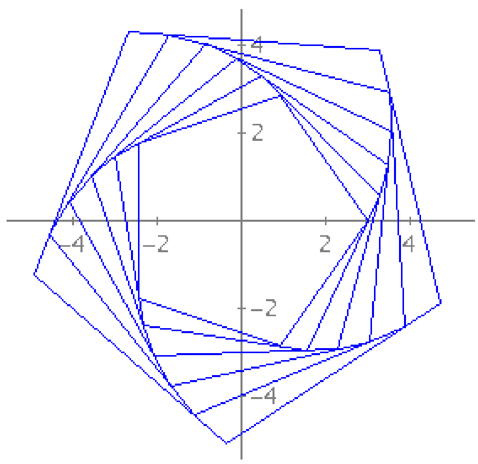
inscribed

$$\mathrm{POL\_ROT}\left(3,\ 5,\ \frac{\pi}{18},\ 5,\ -1\right)$$



circumscribed

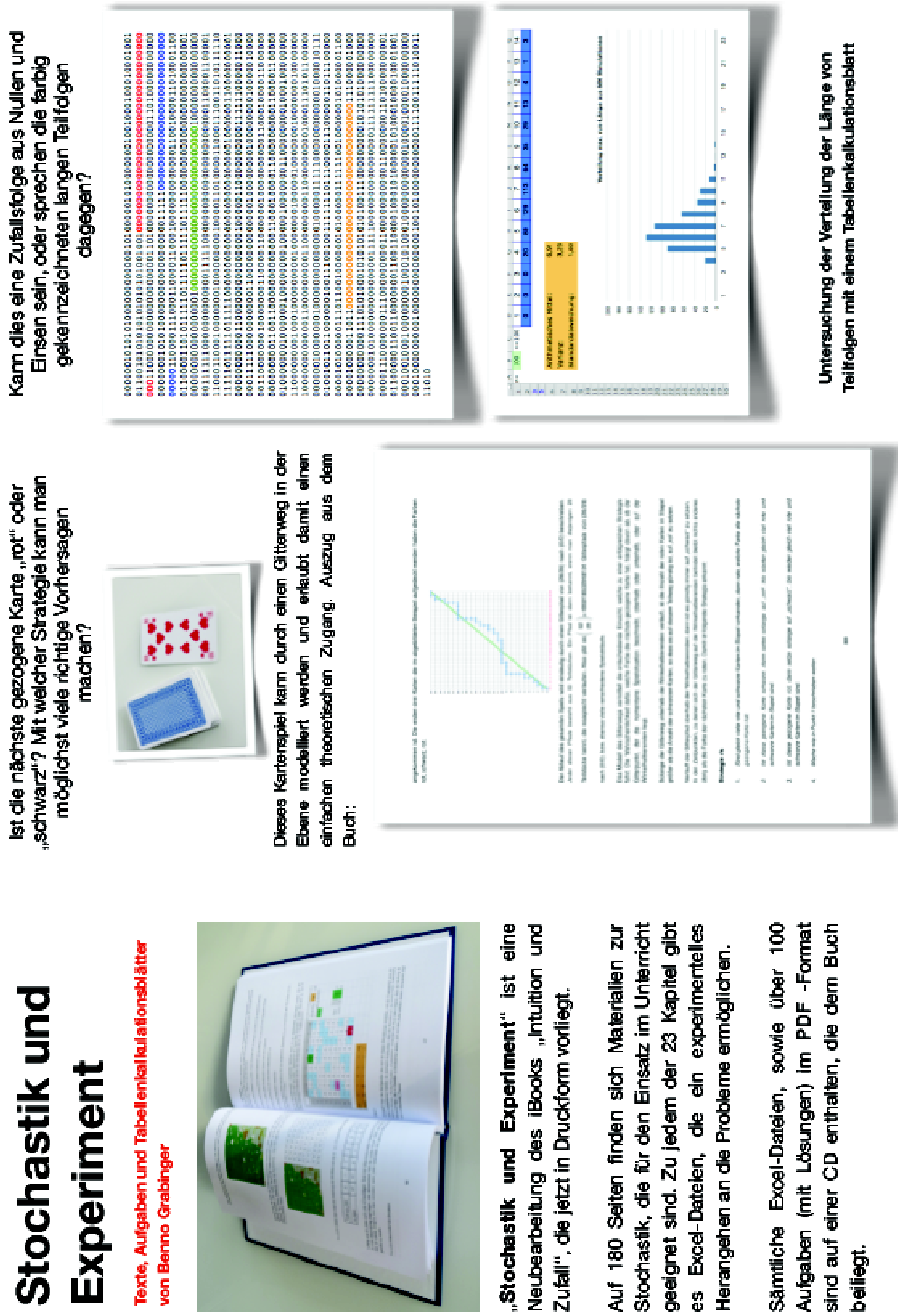
$$\mathrm{POL\_ROT}\left(3,\ 5,\ -\frac{\pi}{18},\ 5,\ -1\right)$$



circumscribed

In the last DNL I recommended Benno's wonderful book – in German only and until yesterday for MS-Excel. I incited Benno to rewrite this great collection of computer simulations for TI-Nspire – and just today a mail came in:

"I finished your challenge just now. There are 92 Nspire files – many of the including LUA-scripts. I am sending the preliminary text as a pdf-file."



Bestellmöglichkeiten:

Variante 1
Buch im DIN A4 Format, Hard-Cover Einband mit über 200 farbigen Abbildungen), CD mit Excel-Dateien, Aufgaben und PDF-Datei des Buchtextes
incl. 19% Mehrwertsteuer............23,45€

zzgl. Versand..............................3,30€

Variante 2
CD mit Excel-Dateien, Aufgaben und PDF-Datei des Buchtextes
incl. 19% Mehrwertsteuer ....12,60€

zzgl. Versand..............................3,30€

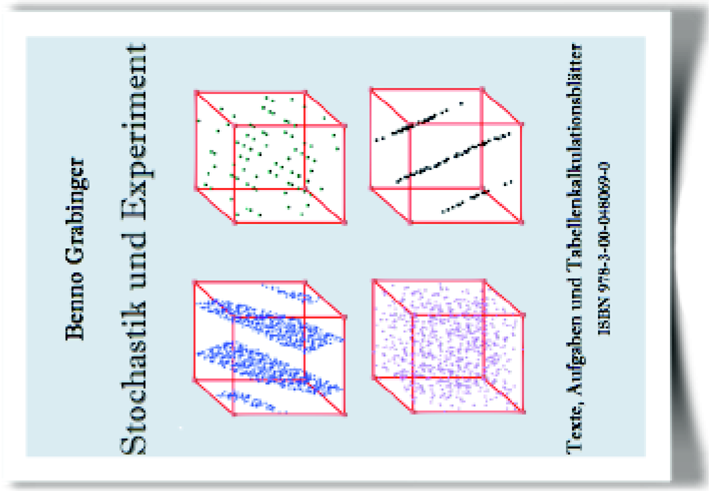
Bestellungen mit Angabe der Lieferadresse senden an:

bennograbinger@iCloud.com

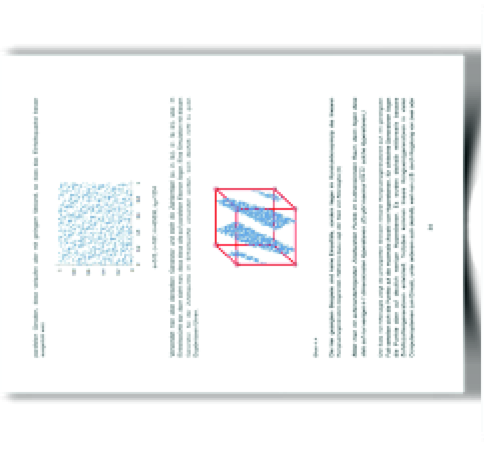
Verlag Grabinger
Am Häuselberg 1, 67434 Neustadt
Tel. 06321 32208

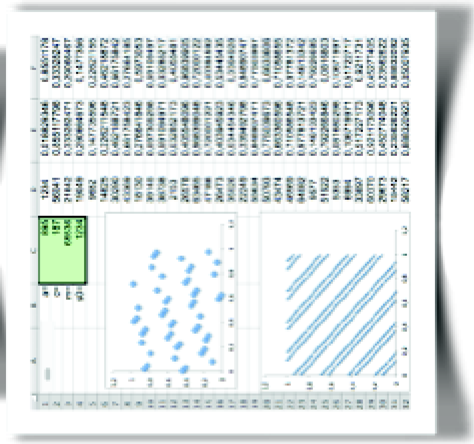
Weitere Informationen zu den einzelnen Kapiteln findet man unter

www.bennograbinger.de



Wie kann man Zufallszahlen erzeugen und wie „gut" sind diese?

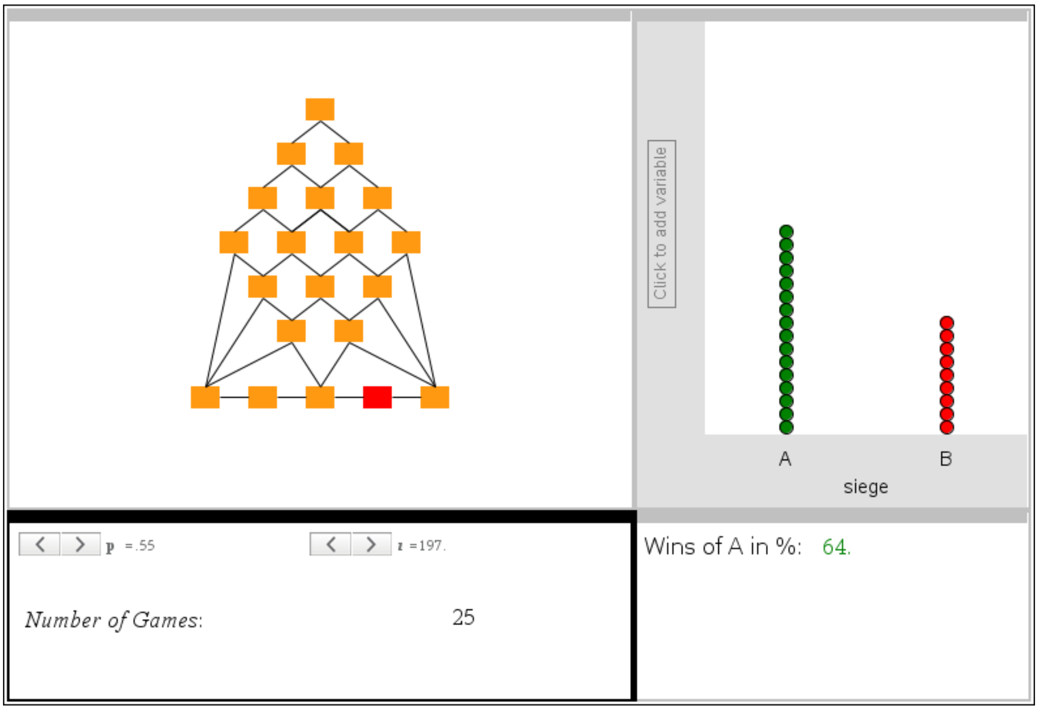


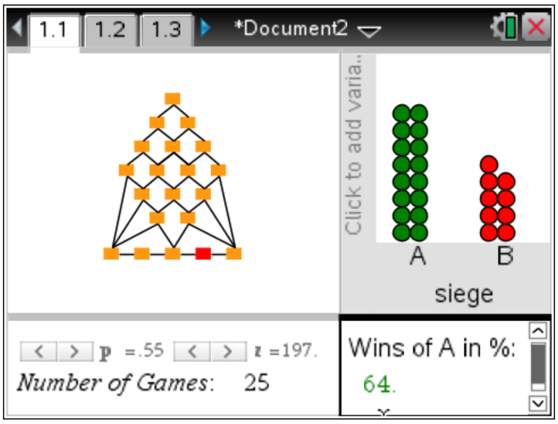


Zugehöriges Tabellenkalkulationsblatt

This is the flyer for the Excel-edition. The next page shows some graphs

I present some screen shots from Chapter 18 “Tennis”. There are 24 chapters on nearly 200 pages.



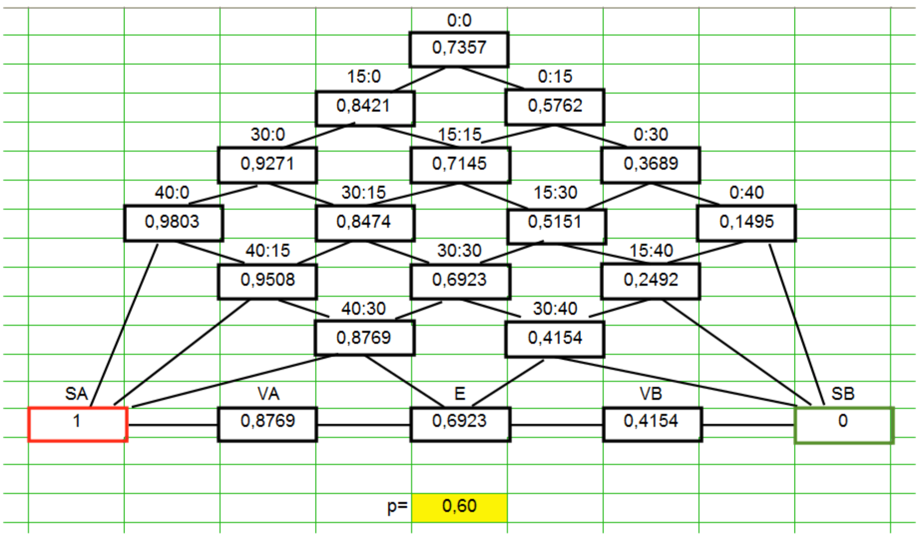


You can see the probability tree for the run of one game.

The last row shows the outcome, either on the left (SA = A wins the game) or on the right (SB = B wins the game).

The positions in the last row are:
Win A, Advantage A, Deuce, Advantage B and Win B.

The respective Excel-Graph offers more space for annotating the knots in the graph.

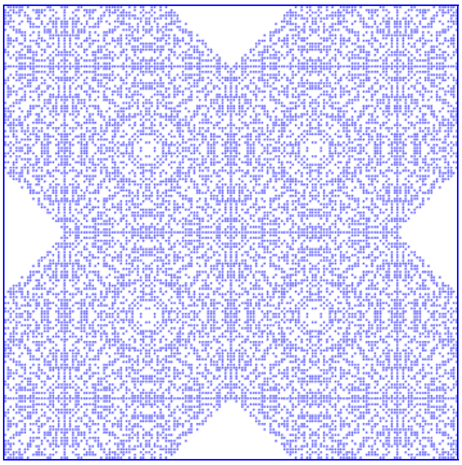


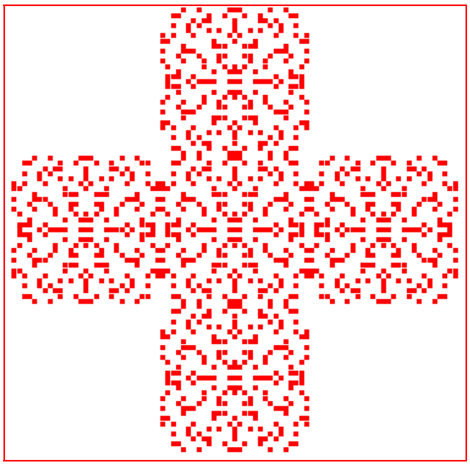
Reste und ihre Muster
Complex Numbers on Norwegian Sweaters

Heinrich Ludwig, Germany

Die graphische Darstellung mathematischer Prozesse und Strukturen führt oft zu ästhetisch ansprechenden Bildern. Fraktale Mengen wie Julia- oder Mandelbrotmengen beispielsweise fanden ihren Weg in die populärwissenschaftliche Literatur hauptsächlich wegen ihrer reizvollen Abbildungen. Aber auch die elementare Zahlentheorie kann interessante Grafiken erzeugen. Der folgende Text möchte das mit einem Beispiel belegen. Einen kleinen Vorgeschmack geben die abgebildeten Grafiken.

Graphic representation of mathematical processes and structures often leads to aesthetic appealing pictures. Fractal sets like Julia- or Mandelbrot sets found their way into popular literature primarily because of its attractive pictures. But even elementary number theory is able to produce interesting graphics. The following article shall demonstrate this. Take the pictures below as a foretaste.





Reste in Z und Z[i], Remainders in Z and Z[i]

Die Menge der ganzen Zahlen **Z** ist ein euklidischer Ring. Damit ist u.a. gesagt, dass es in **Z** eine Division mit Rest gibt. Legt man einen bestimmten Divisor (= Modul) m fest, dann ist f(x) = MODS(x,m) eine Funktion, die jeder ganzen Zahl ihren Divisionsrest zuordnet. Dabei ist der Betrag des Funktionswerts, d.h. des Restes, kleiner als der Betrag des Moduls m. Diese Eigenschaft ist für das Weitere wichtig, denn sie stellt sicher, dass f(x) nur endlich viele Funktionswerte hat, obwohl f für unendlich viele x-Werte definiert ist.

Auch die Menge aller ganzen komplexen Zahlen **Z**[i] ist ein euklidischer Ring, d.h. die von den ganzen Zahlen **Z** her bekannte Division mit Rest gibt es auch in **Z**[i]. Leider arbeitet die von DERIVE bereitgestellte MODS-Funktion nicht mit komplexen Zahlen. Das macht nichts; eine Funktion, die den komplexen Rest berechnet, kann man mit DERIVE programmieren.

The set of the integer numbers is an Euclidean ring. Among other properties exists a division with remainder in **Z**. Fixing a certain divisor (= modulus) m, then f(x) = MODS(x,m) is a function assigning every integer x its remainder after division by m. The absolute value of the remainder (function value of MODS) is smaller than the absolute value of the modulus. This is important for the following because it makes sure that f(x) has only a finite number of function values although f is defined for infinite many x-values.

The set of all integer complex numbers **Z**[i] form an Euclidean ring, too, i.e. the division with remainder is also existing. Unfortunately function MODS provided by DERIVE does not support complex numbers. Doesn't matter, a function which calculates the complex remainder can be defined by DERIVE.

```
MODC(a, m, c, d) :=
  Prog
    If m = 0
       RETURN a
    c := a/m
    d := FLOOR(RE(c)) + i·FLOOR(IM(c))
    d := a - m·d
    If RE(d) = 0
       d := IM(d)·i
    If IM(d) = 0
       d := RE(d)
    RETURN d
```

```
MODCS(a, m, c, d) :=
  Prog
    If m = 0
       RETURN a
    c := a/m
    d := FLOOR(RE(c) + 1/2) + i·FLOOR(IM(c) + 1/2)
    d := a - m·d
    If RE(d) = 0
       d := IM(d)·i
    If IM(d) = 0
       d := RE(d)
    RETURN d
```

Die beiden IF-Anweisungen am Ende von MODC bzw. MODCS sind aus mathematischer Sicht überflüssig. Sie verhindern nur die Auswirkung eines internen Fehlers von DERIVE 6.

Both IF-commands at the end of MODC and MODCS are not necessary from the mathematical point of view. They prevent an internal error of DERIVE.

MODCS(a,m) und MODC(a,m) sind die komplexen Analoga zu MODS und MOD, wobei a der Dividend und m der Modul ist. Ein Beispiel:

MODCS(a,m) and MODC(a,m) are the complex analogues of MODS and MOD, where a is the dividend and m the modulus. An example:

```
MODC(5 + 3·i, 4 + i) = 1 + 2·i

MODCS(5 + 3·i, 4 + i) = 1 + 2·i

MOD(5 + 3·i, 4 + i) = 1 + 2·i

MODS(5 + 3·i, 4 + i) = 2 - 2·i
```

Comment: See the definitions of MOD and MODS given in the DERIVE Online-Help:

MOD(m, n) simplifies to m modulo n (i.e. the nonnegative remainder of m/n).
MODS(m, n) simplifies to the symmetric m modulo n which will be in the half-open interval [-n/2,n/2).

Ein Modul m aus **Z** \ {0} sei festlegt. Wenn man die Funktion MODS mit allen möglichen ganzen Zahlen füttert, dann bekommt man als Wertemenge alle ganzen Zahlen des Intervalls (- m/2, m/2]. Beispielsweise bekommt man für m = 13 folgende Restemenge: {-6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6}

A modulus from **Z** \{0} be fixed. Feeding function MODS with all possible integers then one receives all integers ∈ (- m/2, m/2]. E.g. for m = 13 the set of remainders is given by {-6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6}.

```
VECTOR(MODS(k, 13), k, 100, 120)

[-4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3]

VECTOR(MOD(k, 13), k, 100, 120)

[9, 10, 11, 12, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 0, 1, 2, 3]
```

Was erhält man bei analogem Vorgehen im Ring **Z**[i]? Ein Modul m aus **Z**[i] \ {0} sei festgelegt. Füttert man die Funktion MODCS(x, m) mit allen möglichen komplexen ganzen Zahlen x, so treten auch hier nur endlich viele Reste auf. Der Grund dafür ist derselbe wie in **Z**.
Für m = 4+i und eine Auswahl von Dividenden x = a+b·i mit -5 ≤ a ≤ 5, -5 ≤ b ≤ 5 sind die Ergebnisse in folgenden Tabelle (Matrix) zusammengefasst.

What is happening in ring **Z**[i]? A modulus m ∈ **Z**[i] \ {0} is fixed. Feeding function MODCS with all possible complex integer numbers we receive again a finite number of remainders. The reason is the same like in **Z**. See m = 4 + i and a selection of dividends x = a + b·i with -5 ≤ a ≤ 5, -5 ≤ b ≤ 5. Then the result can be given in form of a table of remainders:

```
VECTOR(VECTOR(MODCS(a + b·i, 4 + i), a, -5, 5), b, 5, -5, -1)
```

$$\begin{bmatrix} 2i & 1+2i & -2+i & -1+i & i & 1+i & -2 & -1 & 0 & 1 & 2 \\ i & 1+i & -2 & -1 & 0 & 1 & 2 & -1-i & -i & 1-i & 2-i \\ 0 & 1 & 2 & -1-i & -i & 1-i & 2-i & -1-2i & -2i & 2i & 1+2i \\ -i & 1-i & 2-i & -1-2i & -2i & 2i & 1+2i & -2+i & -1+i & i & 1+i \\ -2i & 2i & 1+2i & -2+i & -1+i & i & 1+i & -2 & -1 & 0 & 1 \\ -1+i & i & 1+i & -2 & -1 & 0 & 1 & 2 & -1-i & -i & 1-i \\ -1 & 0 & 1 & 2 & -1-i & -i & 1-i & 2-i & -1-2i & -2i & 2i \\ -1-i & -i & 1-i & 2-i & -1-2i & -2i & 2i & 1+2i & -2+i & -1+i & i \\ -1-2i & -2i & 2i & 1+2i & -2+i & -1+i & i & 1+i & -2 & -1 & 0 \\ -2+i & -1+i & i & 1+i & -2 & -1 & 0 & 1 & 2 & -1-i & -i \\ -2 & -1 & 0 & 1 & 2 & -1-i & -i & 1-i & 2-i & -1-2i & -2i \end{bmatrix}$$

Comment: Let's compare the first rows of this matrix created using DERIVE's MODS-function:

```
VECTOR(VECTOR(MODS(a + b·i, 4 + i), a, -5, 5), b, 5, -5, -1)
```

$$\begin{bmatrix} 1-2i & 2-2i & -1-3i & -3i & 1-3i & 2-3i & -1-4i & -4i & 0 & 1 & 2 \\ 1-3i & 2-3i & -1-4i & -4i & 0 & 1 & 2 & -1-i & -i & 1-i & 2-i \\ 0 & 1 & 2 & -1-i & -i & 1-i & 2-i & -1-2i & -2i & 1-2i & 2-2i \\ -i & 1-i & 2-i & -1-2i & -2i & 1-2i & 2-2i & -1-3i & -3i & 1-3i & 2-3i \\ -2i & 1-2i & 2-2i & -1-3i & -3i & 1-3i & 2-3i & -1-4i & -4i & 0 & 1 \end{bmatrix}$$

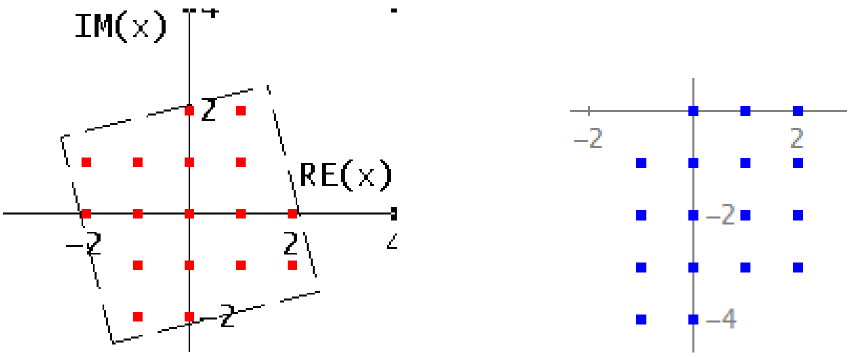
Comment: So we have different symmetric remainders. I checked by subtracting both remainders from the dividend and then calculating the MOD-, MODS-, MODC-, and MODCS-values.

Comment: I am sorry but I cannot give an explanation for this. I am sure that we have several complex number specialists among us who can.

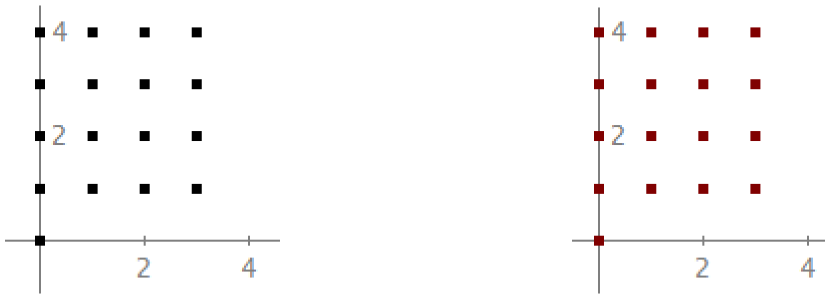
```
MODS(-5 + 5·i, 4 + i) = 1 - 2·i
MODCS(-5 + 5·i, 4 + i) = 2·i
MOD(-5 + 5·i - 1 + 2·i, 4 + i) = 0
MOD(-5 + 3·i, 4 + i) = 0
MODCS(-5 + 5·i - 1 + 2·i, 4 + i) = 0
MODS(-5 + 5·i - 1 + 2·i, 4 + i) = 0
MODS(-5 + 5·i - 2·i, 4 + i) = 0
MODCS(-5 + 5·i - 2·i, 4 + i) = 0
```

Eines lässt sich sofort erkennen: Die Funktion f(x) = MODCS(x, m) ist „2-dim-periodisch“: Geht man in der Matrix 4 Spalten nach rechts und eine Zeile nach oben, dann erhält man ungeachtet des Ausgangspunkts dieselbe Zahl. Das gleiche sieht man, wenn man eine Spalte nach links und 4 Zeilen nach oben geht. Eine weitere Beobachtung macht man, wenn man alle vorkommenden Funktionswerte zusammenträgt: Sie liegen in der komplexen Ebene in einem Quadrat, dessen eine Seite m ist und dessen andere m·i. Für m = 4+i findet man daher insgesamt $|m|^2$ = 17 verschiedene Werte. (Diese Feststellungen sind unmittelbare Auswirkungen der Funktionsvorschrift von f(x).)

What we can recognize immediately is that function f(x) = MODCS(x,m) is “2-dim-periodical”: Starting at any element of the matrix and stepping 4 columns to the right and then one row up we will end at the same number. The same happens making 4 steps to the left followed by one step down. We can make a second observation when collecting all appearing function values. They are within a square in the complex plane with one side m and the other one m·i. Therefore for m = 4 + i there are totally $|m|^2$ = 17 different values. (These facts are consequences of function rule of f(x).)



MODCS generated remainders and MODS generated remainders



MOD and MODC give the same 17 points

Quadratische Reste, Quadratic Remainders

Interessant wird es erst, wenn man f(x) = MODCS(x, m) mit den *Quadraten* aller ganzen Zahlen füttert. Jetzt treten nur *bestimmte* Funktionswerte (= Reste) auf. Wählt man zum Beispiel m = 13, so treten von den 13 möglichen Resten nur 7 verschiedene auf, nämlich:

{-4, -3, -1, 0, 1, 3, 4}.

(Die Anzahl dieser so genannten quadratischen Reste in Abhängigkeit vom Modul ist eine sehr interessante Funktion; sie soll aber hier nicht weiter untersucht werden.)

Die Idee, nur die Reste von Quadratzahlen zu betrachten, soll auf **Z**[i] übertragen werden. Wir erzeugen die Matrix von oben nochmals, diesmal aber mit dem Argument $(a + b\cdot i)^2$:

It becomes really interesting if we feed f(x) = MODCS(x, m) with the *squares* of all integers. Now only certain function values (= remainders) appear. Choosing e.g. m = 13 instead of 13 possible remainders we find only seven ones, which are:

{-4, -3, -1, 0, 1, 3, 4}.

(The number of these so called quadratic remainders depending on the modulus is a very interesting function, which cannot be investigated here.)

The idea to only observe the remainders of squares shall be transferred on **Z**[i]. We evaluate once more the matrix from above exchanging a + b·i by $(a + b\cdot i)^2$:

```
VECTOR(VECTOR(MODCS((a + b·i)^2, 4 + i), a, -5, 5), b, 5, -5, -1)
```

$$\begin{bmatrix} i & -2 & 2 & -2i & -1 & 2i & -i & 1 & 0 & 1 & -i \\ -1 & 2i & -i & 1 & 0 & 1 & -i & 2i & -1 & -2i & 2 \\ 0 & 1 & -i & 2i & -1 & -2i & 2 & -2 & i & i & -2 \\ -1 & -2i & 2 & -2 & i & i & -2 & 2 & -2i & -1 & 2i \\ i & i & -2 & 2 & -2i & -1 & 2i & -i & 1 & 0 & 1 \\ -2i & -1 & 2i & -i & 1 & 0 & 1 & -i & 2i & -1 & -2i \\ 1 & 0 & 1 & -i & 2i & -1 & -2i & 2 & -2 & i & i \\ 2i & -1 & -2i & 2 & -2 & i & i & -2 & 2 & -2i & -1 \\ -2 & i & i & -2 & 2 & -2i & -1 & 2i & -i & 1 & 0 \\ 2 & -2i & -1 & 2i & -i & 1 & 0 & 1 & -i & 2i & -1 \\ -i & 1 & 0 & 1 & -i & 2i & -1 & -2i & 2 & -2 & i \end{bmatrix}$$

An der 2-dim-Periodizität hat sich, wie zu erwarten war, nichts geändert. Die Anzahl verschiedener Funktionswerte hat gegenüber vorher aber abgenommen. Von den ursprünglich 17 sind nur diese 9 geblieben:

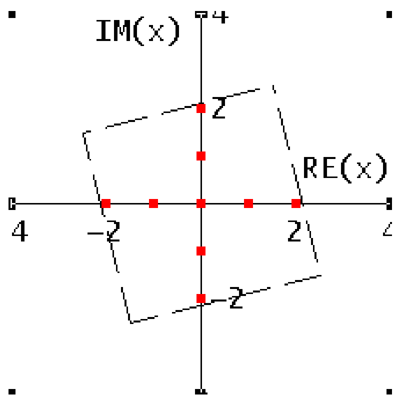
$$\{-2, -1, -2\cdot i, -i, 0, i, 2\cdot i, 1, 2\}.$$

Die Wertemenge der Funktion $g(x) = \text{MODCS}(x^2, m)$ ist also nur eine Teilmenge der Wertemenge von f(x). Während f(x) ein lückenloses Punktegitter innerhalb eines Quadrats erzeugt (s. Abb auf Seite 34), erzeugt g(x) eine Punktmenge mit etlichen „Löchern“. Ein Muster entsteht (siehe die nächste Abbildung).

2-dim-periodicity did not - as expected - change. But the number of different function values decreased. Instead of 17 from before only 9 of them remained, namely:

$$\{-2, -1, -2\cdot i, -i, 0, i, 2\cdot i, 1, 2\}.$$

The range of $g(x) = \text{MODCS}(x^2, m)$ is only a subset of the range of f(x) = MODCS(x,m). While f(x) produces a grid of points without gaps within a square g(x) results in a point set with several “holes“, a pattern appears (see the graph).



Bilderzeugung, Generating the Patterns

Zwei Dinge fehlen noch. Zum einen ist das eben erzeugte „Muster“ noch nicht wirklich interessant. Doch das liegt nur an der Wahl des Moduls m, der in dem relativ kleinen Quadrat eben nur ein einfaches Kreuz entstehen lässt. Zum anderen sollte das Sammeln der verschiedenen Teilungsreste aus Matrizen wie oben gegeben einem DERIVE-Programm übertragen werden, da eine manuelle Bearbeitung mühsam wäre.

Die Aufgabe des Restesammelns übernimmt die Funktion ShowAllRemainders(m). Setzt man für den Parameter m eine Zahl aus **Z**[i] \ {0} ein, so gibt die Funktion einen Vektor zurück, der (ohne Duplikate) alle Reste bezüglich m enthält.

OBACHT: Für m wird vorausgesetzt, dass RE(m) ≥ 0 und IM(m) ≥ 0. Das bedeutet keine Beschränkung der Allgemeinheit, da m, m·i, -m und -m·i dieselben Reste erzeugen und mindestens eine dieser vier Zahlen die Bedingung erfüllt.

Two things are still missing. Firstly, the “pattern“ presented above is not really breath taking. The reason for this is only the choice of the modulus m, which can create only a small cross within the small square – this can be changed easily. And secondly we should find a DERIVE program for collecting the different remainders given in the matrices above.

The task of collecting the remainders is done by function ShowAllRemainders(m). For any m∈**Z**[i] \ {0} this function gives back a vector (list) containing all remainders with respect to m (without duplicates).

TAKE CARE: RE(m) ≥ 0 and IM(m) ≥ 0. This no restriction of generality because m, m·i, -m and -m·i produce the same remainders and at least one of these numbers fulfill the condition.

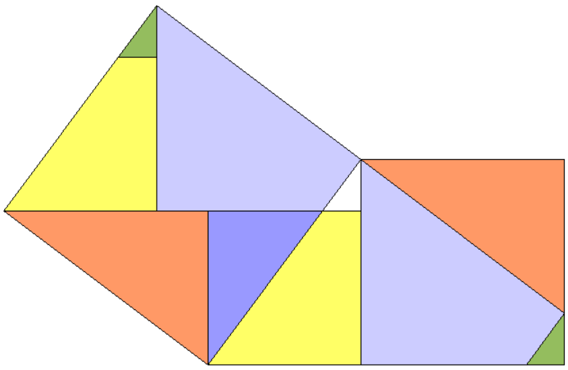
```
        2
g(x) := x

ShowAllRemainders(m, k, j, r, s := [0]) :=
  Prog
    k := 0
    Loop
      If k = RE(m) exit
      j := 0
      Loop
        If j = RE(m) exit
        r := MODCS(g(k + j·ί), m)
        If ¬ MEMBER?(r, s)
           s := ADJOIN(r, s)
        j :+ 1
      k :+ 1
    k := 0
    Loop
      If k = IM(m) exit
      j := 0
      Loop
        If j = IM(m) exit
        r := MODCS(g(k + RE(m) + j·ί), m)
        If ¬ MEMBER?(r, s)
           s := ADJOIN(r, s)
        j :+ 1
      k :+ 1
    RETURN s

ShowAllRemainders(4 + ί) = [-ί, 2, -2, 2·ί, 1, - 2·ί, ί, -1, 0]

DIM(ShowAllRemainders(40 + 11·ί)) = 861
```

Die Struktur der Funktion ShowAllRemainders(m) ist leicht zu erkennen. Allerdings macht sie Gebrauch von einer Umsortierung, die kommentiert werden sollte.

Für jeden gegebenen Modul m ist die Menge der Werte der Funktion MODCS(x,m) endlich, weil MODCS(x,m) für festes m hinsichtlich x 2-dim-periodisch ist. Darum genügt es, eine Periode zu durchmustern. Die Periode hat in der Gaußschen Ebene die Form eines Quadrats mit den Ecken m/2*[-1-i, 1-i, 1+i, -1+i]. (Dieses Quadrat ist in den beiden Abbildungen strichliert eingetragen.) Das Quadrat umfasst ||m|| Punkte aus **Z**[i]. Weil die Seiten des Quadrats i.a. nicht achsenparallel liegen, eignet es sich schlecht für eine Durchmusterung. Darum wird es nach Art des Beweises des Höhensatzes in zwei neue Quadrate umgewandelt. Betrachte dazu die nächste Illustration. Die zwei neuen Quadrate enthalten dieselben Reste wie das ursprüngliche, liegen aber achsenparallel. Den beiden Quadraten entsprechend besteht das Programm aus zwei Teilen, deren jeder jeweils eines der beiden Quadrate durchmustert.

Structure of function ShowAllRemainders(m) can be recognized easily. But it applies a resorting which shall be commented.

For a given modulus m the set of function values of MODCS(x,m) is finite, because MODCS(x,m) for a fixed m with respect to x is 2-dim-periodical. So it is sufficient to observe only one period. The period has the form of a square with vertices m/2*[-1-i, 1-i, 1+i, -1+i] in the Gauß plane. (This square is added in the graphs above with dashed lines.) This square contains ||m|| points of **Z**[i]. Because the sides of this square usually are not parallel to the axes it is not very suitable for searching for points. Therefore it will be – like in the proof of the altitude theorem – transformed into two new squares. The graph below is illustrating the procedure. Both new squares contain the same remainders as the original ones but are lying parallel to the axes. The program consists of two parts each of them investigating one of the two squares.



Die komplexen Zahlen, die ShowAllRemainders(m) liefert, müssen noch in ein kartesisches Koordinatensystem eingetragen werden. Dazu muss jede Zahl in einen 2-dimensionalen Vektor umgewandelt werden. Das besorgt die Funktion Pattern(m). Wie ihr Name verrät, erzeugt Pattern(m) endlich das, was eingangs versprochen wurde: schöne Muster. Ihr Reiz liegt darin, dass man für (beinahe) jede Zahl m ein neues, andersartiges Muster bekommt und sich nur schwer vorhersagen lässt, welche Werte m besonders schöne Muster ergeben werden. Das lädt ein zum Probieren und bringt immer wieder Überraschungen mit sich.

Pattern5(m) und Pattern9(m) sind zwei Erweiterungen von Pattern(m), die das Muster zu den 4 Seiten bzw. rundherum mit Duplikaten fortsetzen, so dass es ausdrucksvoller zur Geltung kommt.

The complex numbers delivered by ShowAllRemainders(m) must be entered in a Cartesian system of coordinates. Every number must be transformed into a 2-dimensional vector. This is performed by function Pattern(m). According to its name Pattern(m) produces pretty patterns. Its special attraction is that nearly every m results in a new and different pattern and it cannot be predicted in advance which values of m will give extra pretty patterns. This invites for experimenting.

Pattern5(m) and Pattern9(m) are two extensions of Pattern(m), which append the pattern in four and eight directions to make it much more impressive.

```
Pattern(m) := VECTOR([RE(u), IM(u)], u, ShowAllRemainders(m))
```

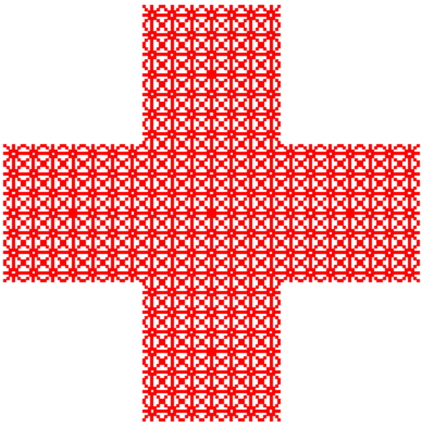
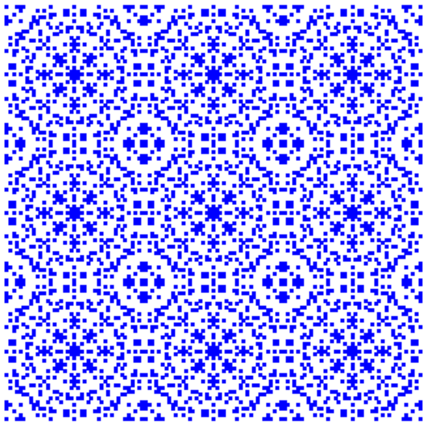
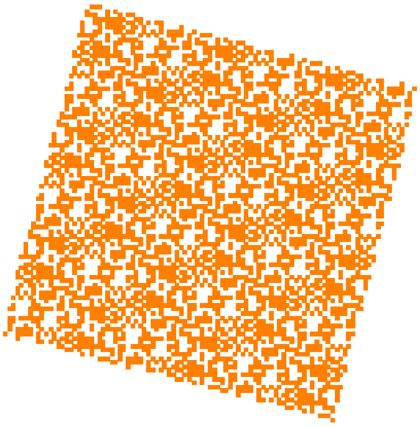
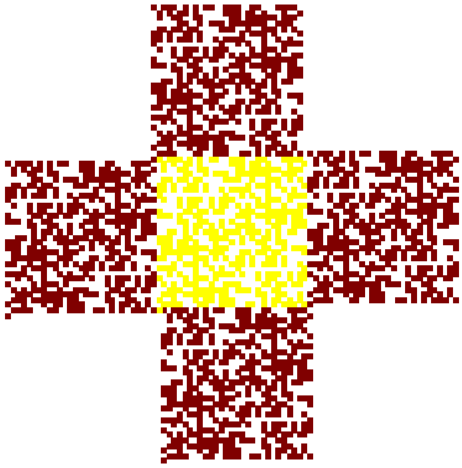
$$\mathrm{Pattern5(m)} := \mathrm{APPEND}\left(\mathrm{VECTOR}\left(\begin{bmatrix} \mathrm{RE(u)} & \mathrm{IM(u)} \\ \mathrm{RE(u + m)} & \mathrm{IM(u + m)} \\ \mathrm{RE(u - m)} & \mathrm{IM(u - m)} \\ \mathrm{RE(u + m\cdot}\hat{\imath}) & \mathrm{IM(u + m\cdot}\hat{\imath}) \\ \mathrm{RE(u - m\cdot}\hat{\imath}) & \mathrm{IM(u - m\cdot}\hat{\imath}) \end{bmatrix}, \mathrm{u, ShowAllRemainders(m)}\right)\right)$$

$$\mathrm{Pattern9(m)} := \mathrm{APPEND}\left(\mathrm{VECTOR}\left(\begin{bmatrix} \mathrm{RE(u)} & \mathrm{IM(u)} \\ \mathrm{RE(u + m)} & \mathrm{IM(u + m)} \\ \mathrm{RE(u - m)} & \mathrm{IM(u - m)} \\ \mathrm{RE(u + m\cdot}\hat{\imath}) & \mathrm{IM(u + m\cdot}\hat{\imath}) \\ \mathrm{RE(u - m\cdot}\hat{\imath}) & \mathrm{IM(u - m\cdot}\hat{\imath}) \\ \mathrm{RE(u + m\cdot(}\hat{\imath} + 1)) & \mathrm{IM(u + m\cdot(}\hat{\imath} + 1)) \\ \mathrm{RE(u + m\cdot(}\hat{\imath} - 1)) & \mathrm{IM(u + m\cdot(}\hat{\imath} - 1)) \\ \mathrm{RE(u + m\cdot(-}\hat{\imath} + 1)) & \mathrm{IM(u + m\cdot(-}\hat{\imath} + 1)) \\ \mathrm{RE(u + m\cdot(-}\hat{\imath} - 1)) & \mathrm{IM(u + m\cdot(-}\hat{\imath} - 1)) \end{bmatrix}, \mathrm{u, ShowAllRemainders(m)}\right)\right)$$

Wer die Ausführungen bis hierher verfolgt hat, dem seien die versprochenen Muster nicht länger vorenthalten! Man öffne ein 2D-Graphikfenster, stelle einen Zeichenbereich der Breite und Höhe 100 mit Zentrum bei [0,0] ein und setze das Verzerrungsverhältnis auf 1:1. Jetzt sollte DERIVE noch angewiesen werden, dicke oder mittelgroße Punkte zu zeichnen und diese *nicht* zu verbinden. Dann lasse man den folgenden Ausdruck plotten:

If you have followed the article until now you should be presented the promised patterns! Open a 2D-plot window and set the aspect ratio to 1:1. Take Point Size Large (or medium) and don't connect the points. Then plot the following expressions:

```
Pattern5(49)

Pattern9(41)

Pattern9(7 + 27·ı)

Pattern5(29 + ı)

Pattern(29 + ı)
```

Durch Variation des Moduls m lassen sich unzählige weitere Muster gewinnen. Letztlich sind der Vielfalt der Muster nur durch die Rechenzeit Grenzen gesetzt.

By varying the modulus m we can get numerous other patterns. Restrictions are given only by calculation time.

Zum Schluss füge ich noch einige Kommentare und Fragen an:

Anmerkung1:

Interessante Muster findet man insbesonders für Moduln (=Divisoren) m, für die gilt:
a) m aus **Z** und ungerade, besonders hübsch sind m = 7, 9, 27, 41, 49, 73.
b) ||m|| = m·m* ist Quadratzahl, (d.h. Real- und Imaginärteil gehören zu einem pythagoräischen Tripel), z.B. m = 40+9i
Nett sind auch: m = 9+i, 15+i, 39+i, 39+5i, 39+22i, 39+29i, 40+11i, 40+23i, 40+29i, 40+39i, 41+i, 41+9i, 41+20i, 41+38i.
Die Muster für a + b·i und b + a·i sind bis auf Spiegelung gleich.
Für a aus **Z** und a <> 0 (mod 4) scheint zu gelten: Die Muster bezüglich a und a+a·i sind gleich.

Anmerkung2:

Über Hinweise auf Moduln m, die besonders „schöne“ Muster ergeben, würde ich mich freuen (email: lud.wig@web.de).

Anmerkung3:

Wichtig für den Eindruck, der beim Betrachten eines Musters entsteht, ist die Größe der „Punkte“, die das 2D-Fenster zeichnet. Entscheidend ist der Füllgrad des „Punktes“, d.h. das Verhältnis der Fläche des Punktes zur Einheitsfläche des Koordinatensystems. Gute Ergebnisse erhält man, wenn der Füllgrad zwischen 25% und 150% liegt. (Mehr als 100% bedeutet, dass benachbarte Punkte verschmelzen.) Zur schnellen Feinabstimmung kann man das 2D-Fenster verkleinern oder vergrößern.

Anmerkung4:

Man kann die Funktion g(x) durch ein anderes Polynom mindestens 2. Grades und mit Koeffizienten aus **Z**[i] ersetzen und weitere Muster studieren.

Anmerkung5:

Die Funktion MODCS kann ohne Nachteil durch die Funktion MODC ersetzt werden. Dadurch verschiebt sich das Quadrat der Reste um seine halbe Diagonale.

Anmerkung6:

Der Begriff „doppelt-periodisch“ wird üblicherweise in der Funktionentheorie verwendet. Wenn hier eine Funktion „2-dim-periodisch“ genannt wird, soll das bedeuten, dass es zu dieser Funktion zwei über **Q** linear unabhängige Zahlen u und v aus **Z**[i] gibt, so dass für alle x aus **Z**[i] gilt: f(x+u) = f(x+v) = f(x).

Anmerkung7:

Neben Z[i] gibt es weitere euklidischen Ringe der Art **Z**[sqrt(k)], nämlich zumindest für k aus {3, 2, -1, -2, -3, -7, -11}. Es wäre interessant auch deren Muster zu untersuchen.

Fragen:

Für welche Moduln

- hat das Muster die Symmetrie eines Quadrats?
- hat das Muster die Symmetrie eines Rechtecks (bzw. einer Raute)?
- ist das Muster invariant unter Drehung um 90°?
- ist die reelle oder die imaginäre Achse vollständig besetzt?
- ist eine Diagonale durch den Ursprung vollständig besetzt?
- hat das Muster Unterperioden, d.h. Perioden, die ||m|| / n, n $\geq$ 2, Punkte umfassen?
- …

Some comments and questions are following:

Comment1:

Interesting patterns can be found for moduli (= divisors) m with:

a) m $\in$ **Z** and odd, e.g. m = 7, 9, 27, 41, 49, 73.

b) ||m|| = m·m* is a square (i.e. real and imaginary part belong to a pythagorean triple, e.g. m = 40+9i

Pretty patterns are created by: m = 9+i, 15+i, 39+i, 39+5i, 39+22i, 39+29i, 40+11i, 40+23i, 40+29i, 40+39i, 41+i, 41+9i, 41+20i, 41+38i. Patterns for a + b·i and b + a·i are symmetric. For a $\in$ **Z** and a <> 0 (mod 4) seems to be that the patterns with respect to a and a+a·i are the same.

Comment2:

I would be very grateful to receive any notice about moduli m which give extremely “beautiful“ patterns (email: lud.wig@web.de).

Comment3:

Important for the impression given by the plot is the point size in the 2D-plot window. I mean the ratio of the area filled by one single point with respect to the unit area of the coordinate system. Good results can be expected with a ratio between 25% and 150%. (More than 100% let neighboring points melt into one another.)

Comment4:

Function g(x) can be substituted by another polynomial with order ≥ 2 and coefficients from **Z**[i] in order to investigate or study other patterns.

Comment5:

MODCS can be substituted by MODC. The square of the remainders is then shifted by its half diagonal.

Comment6:

The term “double periodical“ is usually used in function theory. We call here a function “2-dim- periodical“ to indicate that there are two above **Q** linear independent numbers u, v∈**Z**[i] such that for all x∈**Z**[i]: f(x+u) = f(x+v) = f(x).

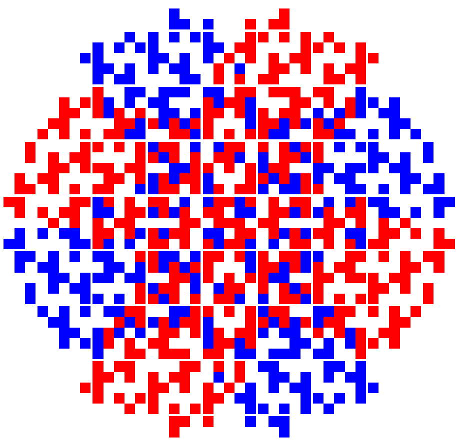
Comment7:

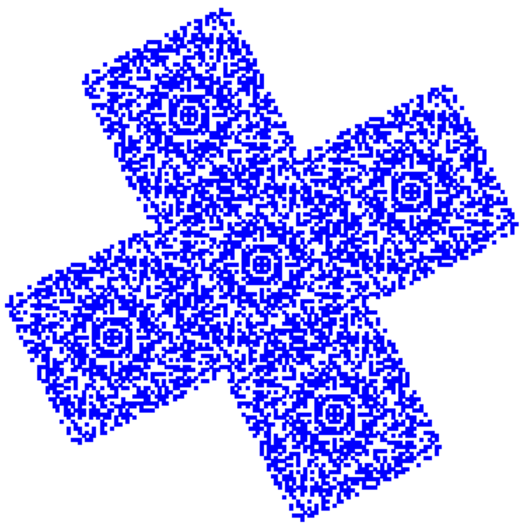
Beside **Z**[i] there are other euclidean rings of kind **Z**[sqrt(k)], namely at least for k ∈ {3, 2, -1, -2, -3, -7, -11}. It might be interesting to investigate their patterns, too.

Questions:

For which moduli m

- do we receive a pattern with the symmetry of a square?
- do we receive a pattern showing the symmetry of a rectangle (or a rhombus)?
- is the pattern invariant under a 90° rotation?
- is the real or the imaginary axis completely occupied (by points)?
- is one diagonal passing the origin completely occupied?
- are there subperiods, i.e. periods which contain ||m|| / n, n ≥ 2, points?
- …





Histograms of discrete probability distributions

Benno Grabinger, Guido Herweyers, Josef Böhm

I received a mail from our friend Benno Grabinger, complaining that it seems to be impossible to present the probability function of e.g. a binomial distribution as a histogram. In MS-Excel is this no problem at all.

Benno had an idea to overcome this deficiency by producing a list of many - say 1000 or more - outcomes of this distribution and then plot the respective histogram.
Example: n = 2 and p = 0.5 gives probabilities {0.25, 0.5, 0.25} for x = 0, 1, 2. liste_bin contains 250 zeros, 500 ones and 250 twos. And this list can be presented as a histogram.

Lieber Josef,

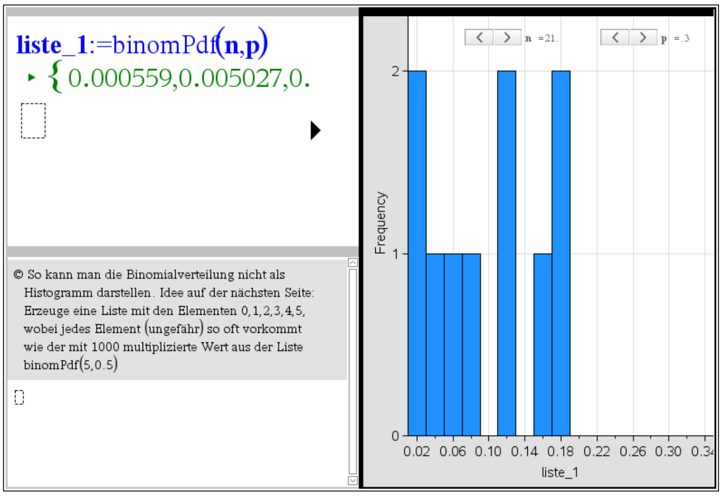
mich hat es schon immer geärgert, dass man von einer Wahrscheinlichkeitsfunktion die als Liste vorliegt nicht ohne weiteres ein Histogramm zeichnen kann (zumindest ist mir das nicht gelungen), so wie das z.B. in Excel ohne weiteres möglich ist.

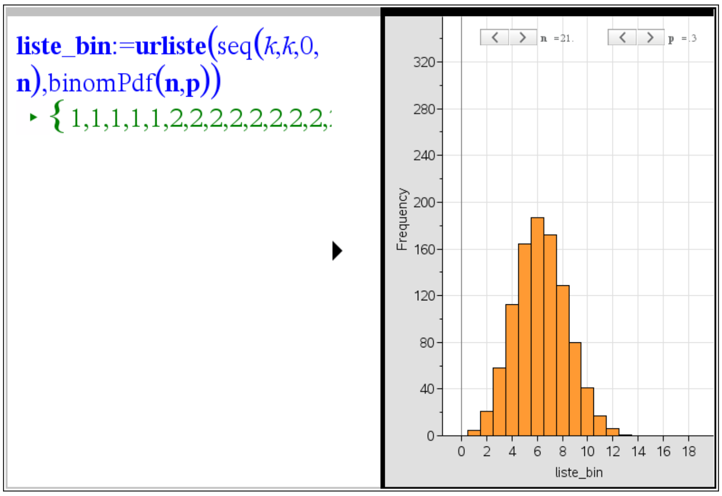
Jetzt hatte ich eine Idee, wie man das doch machen kann. Wenn es dich interessiert, so schau dir einmal die Datei histo an. Hat diese Idee sonst schon jemand gehabt?

Liebe Grüße,
Benno

Beispiel: für n = 2 und p = 0.5 ist die Liste der Wahrscheinlichkeiten für x = 0, 1, 2, natürlich {0.25, 0.5, 0.25}. liste_bin erzeugt dann 250 Nullen, 500 Einser und 250 Zweier.

Für diese Liste kann schließlich das Histogramm erzeugt werden.






```
urliste
Define urliste(wf_x,wf_y)=
Func
Local liste,i
liste:={}
For i,1,dim(wf_x)
  liste:=augment(liste,seq(wf_x[i],k,1,int(wf_y[i]·1000)))
EndFor
Return liste
EndFunc
```

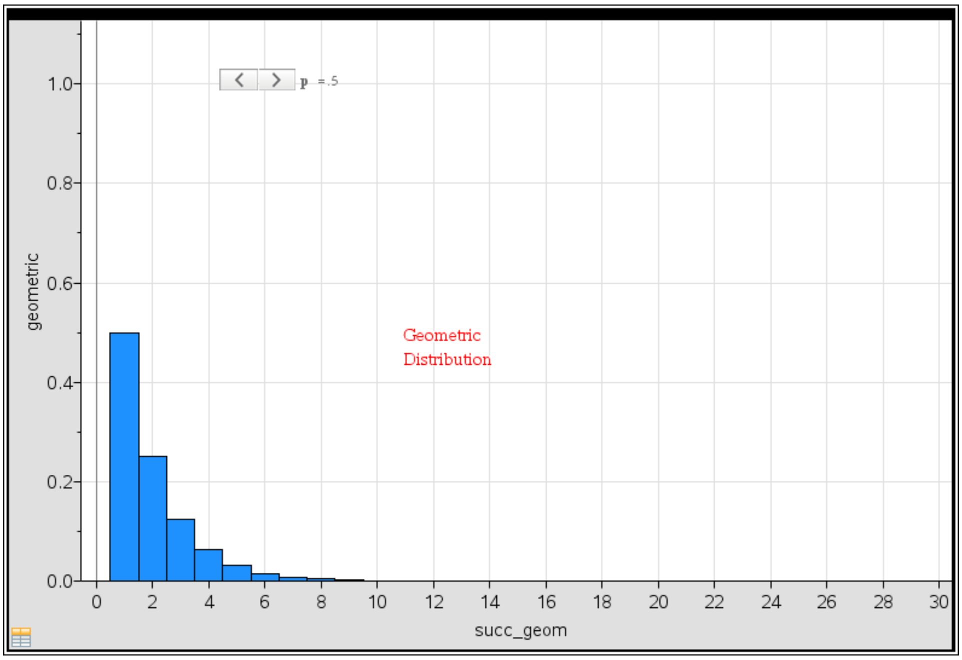
I wrote to our Statistics Guru Guido Herweyers – and he knew the solution: We have to apply a “Summary Plot“ which is in the German version a „Ergebnisdiagramm“ in order to get a histogram of numerical data. Many thanks Guido for your advice. (See also Guido’s contribution on Statistics with TI-Nspire in DNL#85, page 42. I should study the DNLs more accurately, ☺)

Now I could proceed not only presenting a binomial distribution. I wanted to have all popular discrete distributions within one single file working with sliders for the distribution parameters.

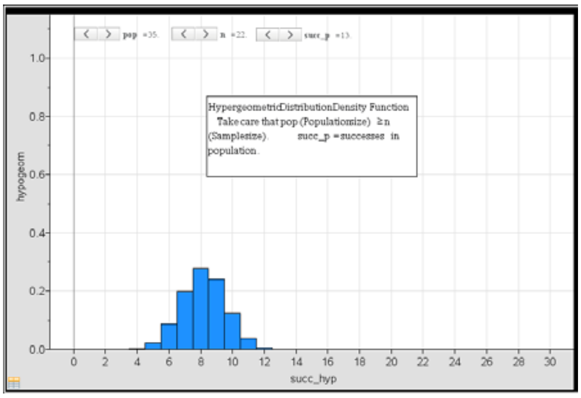
The spreadsheet was filled with the numbers of successes and their respective probabilities. For the parameters n (sample size), p (probability of success in geometric and binomial distribution), pop (population size in hypergeometric distribution), succ_p (number of successes in population – hypergeom. distr.), and finally λ (expectation in Poisson distribution) were sliders installed. See the spreadsheet and the histogram of the geometric distribution.

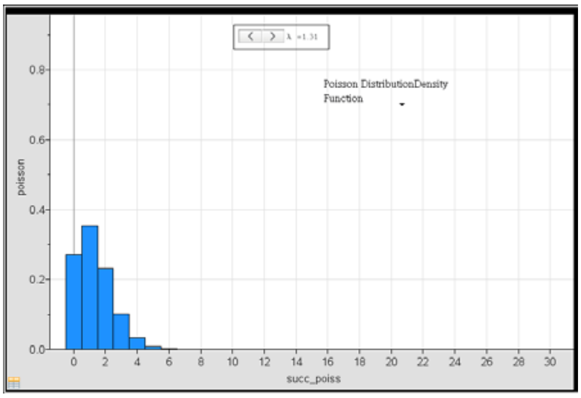
	A succ_bin	B binomial	C succ_geom	D geometric	E succ_poiss	F poiss
=	=seq(k,k,0,'n)	=binompdf('n,'p)	=seq(k,k,1,30	=seq('p*(1−'p)	=seq(k,k,0,30)	=seq(e
1	0	2.38419E-7	1	0.5	0	0.576
2	1	0.000005	2	0.25	1	0.3173
3	2	0.000055	3	0.125	2	0.0872
4	3	0.000367	4	0.0625	3	0.0159
5	4	0.001744	5	0.03125	4	0.00
6	5	0.006279	6	0.015625	5	0.0002
7	6	0.017789	7	0.007813	6	0.0000
8	7	0.040661	8	0.003906	7	0.0000
9	8	0.076239	9	0.001953	8	1.1981
10	9	0.118594	10	0.000977	9	7.3221
11	10	0.154172	11	0.000488	10	4.0271
12	11	0.168188	12	0.000244	11	2.0135
13	12	0.154172	13	0.000122	12	[illegible]

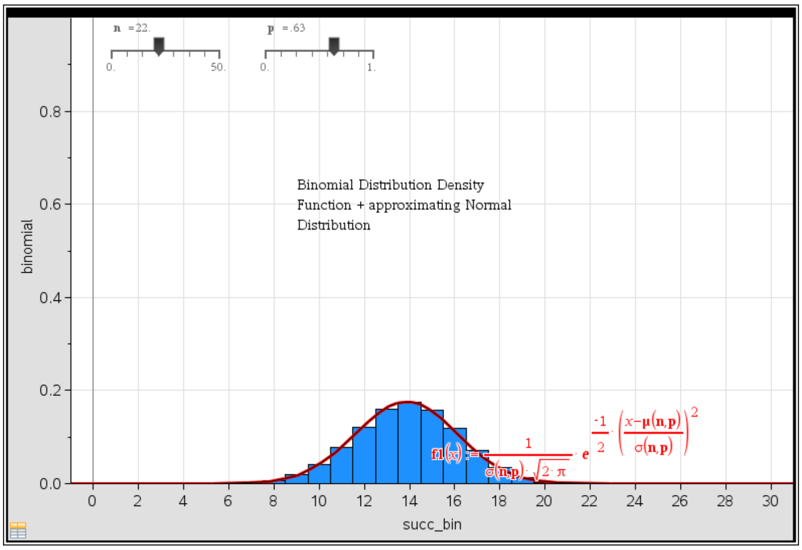
A1 =0



These are the hypergeometric and the Poisson distribution followed by the binomial distribution with the approximating normal distribution function superimposed. The sum of all rectangle areas is always 1. I added some examples which should be solved using the histograms.







Examples:

Solve all examples using the graphic representation of the respective distrubution.

(1) A hunter hits his goal with a probability of 40% for each shot. What is the probability to hit his next goal on his fourth attempt? (8.64%)

(2) Analysis in a rescue organisation finds out that in 47 cases of 100 doctors are attending the ambulance crafts. Yessterday were 8 missions necessary. What is the probability that three of them were attended by a doctor? (24.31%)

(3) In a box containing 20 apples are seven apples not of 1st quality. A customer takes 5 of them without checking their quality. What is the probability that he buys only 1st quality apples? (8.3%)

(4) The prob. that a tourist in an certain tropic country will get infected by a tropic desease is about 0.0002%. Approximately 30 000 tourists come in per year. What is the probability that one of them gets infected? (5.65%)