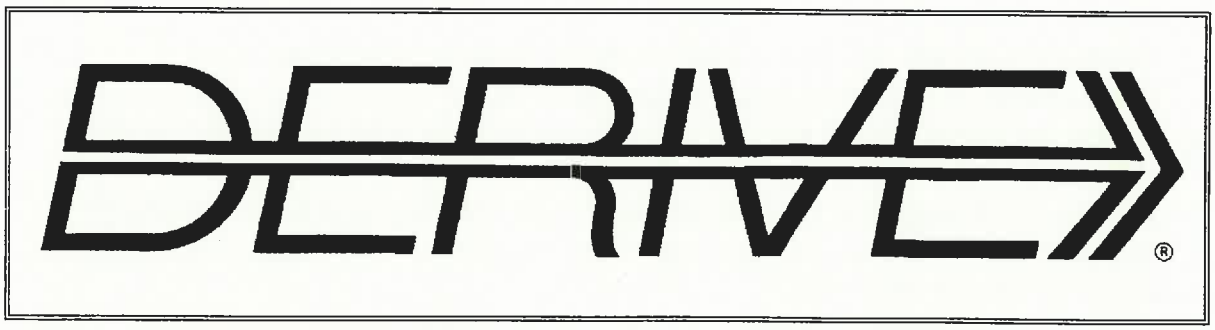
THE DERIVE - NEWSLETTER #47

THE BULLETIN OF THE



USER GROUP

+ TI 92

Contents:

September 2002

[1] **calcul formal cu DERIVE**, Valeriu Anisiu, Editura Albastra, Cluj-Napoca, Romania 2002, ISBN 973 650 047 0

[2] **La Experencia de Descrubir en Geometría**, Miguel de Guzmán Ozámiz, Nivola libros 2002, ISBN 84 95599 34 1

[3] **Stochastik interaktiv mit Derive 5**, Stochastik Lehrgang für die Sekundarstufe, Benno Grabinger, CD, bk-teachware SR-29, ISBN 3 901769 47 1

[4] **Einführung in den Voyage 200** anhand von Beispielen aus den Sekundarstufen I und II, bk-teachware SR-30, ISBN 3 9901769 48 X

[5] **Exponenzial- Potenz- und Winkelfunktionen in der Sek I mit dem TI-83/89/92/92+**, W. Herget und E. Lehmann (Hrsg), Ausgearbeitete, nachvollziehbare Unterrichtsbeispiele inkl. Kopiervorlagen, bk-teachware

[6] **WUM(BS) – Materialien für die Lehrerfortbildung:** Alternative Unterrichtsformen, Etwas andere Aufgaben, Aus Fehlern lernen und Computeralgebra-Systeme, bk-teachware

[7] **Mathematiklehren mit Computeralgebrasystem-Bausteinen**, Eberhard Lehmann, div verlag franzbecker 2002, ISBN 3-881120-343-5

Don´t forget to order the VISIT-ME 2002 Proceedings CD. You will find all lectures (with one single exception) and workshops. On more than 1500 pdf-pages you will find the opening addresses, the keynote lectures, all lectures – with only one exception – and all workshops. Numerous DERIVE, TI, Cabri, GSP- and other utility files are collected presenting a huge ressource of reports, research materials and hands on activities. Available in October, published by bk-teachware.

Order at www.bk-teachware.com.

A Golden Nica for Karola Hummer, Austria, and her TI-92.

Karola Hummer won the U19-Cyber Generation Computer Competition in the frame of the ARS ELECTRONICA 2002, one of the most important Computer Art Events of the World.

The project "TI-92" uses mathematical functions from analytical geometry to draw on the TI-92 calculator (without a cursor). Karola Hummer: "The abstraction is preconditioned by the technology (a screen with about 5000 dots), and I strive for maximum reality with my modest means. If art links idea and technology, then my pictures are at least a modest artistic expression."

Visit Karola at
www.u19.at/htm/projekts2002/TI92.htm
and admire her TI-92 designed slideshow.

99 Functions bring a snowboarder on the TI-Screen.

Congratulations to Karola from the whole CAS-Community. I tried to contact her and hope to receive an answer.

Dear DUG Members,

From my point of view we had two remarkable events this summer: (1) the heavy floods in August and (2) VISIT-ME 2002 in July. Johann Wiesenbauer would add: "Don´t forget (3) the new revolutionary primality test!".

Let me focus on the two enjoyable points. VISIT-ME was a memorable conference – and I am still working on the memories finishing the Conference Proceedings. Please take note of the announcement on the opposite page. For mouth watering you can find some impressions from the many papers on pages 3 and 4 together with some pictures shot by Karl-Heinz Keunecke. I promise some more pictures on the CD.

I take the opportunity to thank all speakers and participants for their cooperation before, during and after the conference. An extra "Thank You" goes to my friends Bernhard Kutzler, Carl Leinbach, Helmut Achleitner, Helmut Heugl, Nancy Köstlbauer Peter Nußbaumer and Vlasta Kokol-Voljc (in alphabetical order) for their wonderful partnership. Thanks to the members of both Scientific Committees supporting us by their valueable advice. Many of them are members of the DUG.

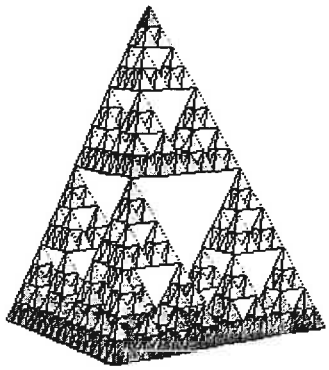
In this DNL you can find an article written by Josef Lechner. I am very glad to congratulate Josef on behalf of the DUG to his PhD. He finished his PhD-study this summer and delivered a thesis on – what else – "New Perspectives in Mathematics Education by using Computer Algebra Systems". His thesis is a small booklet consisting of only 800 pages!! I found his article on modelling the spread of HIV in his thesis and Josef provided a translation. Many thanks and our best wishes.
Im am very glad that Josef´s article as well as most others in this issue might meet the expectations of both *DERIVE*- and TI-users. I like the interchange between PC and handheld technology very much and I hope then most of you are sharing my interest.

When I heard the news about a new primality test in the media I had two ideas. You can find my first one quoted in Johann´s Titbits. The second one was: write an email to Johann and ask him to cover this interesting finding in his next Titbits – and as you can see on pages 40-44, he did. Thank you Johann.

Maybe that Johann (right) together with Albert Rich (center) and Terence Etchells (left) worked on another proof when we had a farewell - barbecue in Würmla after VISIT-ME.

Please have a look on page 2 reading the preview. As you can see we have a bundle of new articles. Among many other very excellent contributions I am looking forward to presenting a wonderful tool for Referential Statistics submitted by MacDonald Phillips. I am sure that you will be surprised how easy to perform hypothesis testing and ANOVA can be using *DERIVE*.

Our friend Heinz-Rainer Geyer worked together with his students on special fractals – Sierpinski Tetrahedrons and Octahedrons – have a look!



The page is full. I´d like to write more, but it must have an end. So finally,
bye, bye until DNL#48

The *DERIVE-NEWSLETTER* is the Bulletin of the *DERIVE & TI-92 User Group*. It is published at least four times a year with a contents of 44 pages minimum. The goals of the *DNL* are to enable the exchange of experiences made with *DERIVE* and the *TI*-92/89 as well as to create a group to discuss the possibilities of new methodical and didactical manners in teaching mathematics.

As many of the *DERIVE* Users are also using the *TI*-92/89 the *DNL* tries to combine the applications of these modern technologies.

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 & TI-92 Newsletter* will be.

Editor: Mag. Josef Böhm
A-3042 Würmla
D'Lust 1
Austria
Phone/FAX: 43-(0)2275/8207
e-mail: nojo.boehm@pgv.at

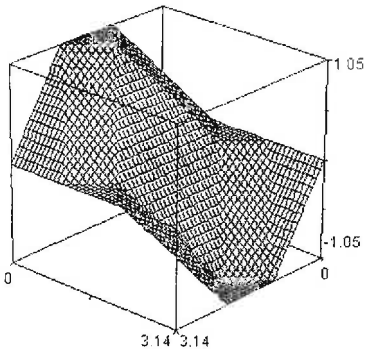
Next issue: December 2002
Deadline 15 November 2002

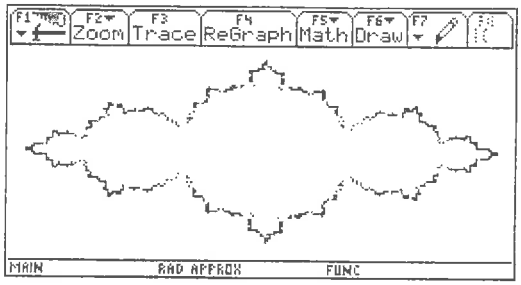
Preview: Contributions waiting to be published

Tanz der Wallace-Geraden / Dance of Wallace-Lines, Baumann, GER
Type checking, Finite continued fractions, Welke, GER
Kaprekar's "Self numbers", Schorn, GER
Some simulations of Random Experiments, Böhm, AUT
Wonderful World of Pedal Curves, Böhm
Another Task for End Examination, Lechner, AUT
Tools for 3D-Problems, Lüke-Rosendahl, GER
Penrose Inverse of a Matrix, Karsten Schmidt, GER
Putzer's Method for the Calculation of e^{At}, Francisco Fernández, ARG
Hypothesis Testing and ANOVA with DERIVE/TI, MacDonald, USA
Hill-Encription, Böhm
"Moon has risen", Malitte, GER
3D-Arrows, Kopp, GER
CAD-Design with DERIVE and the TI, Böhm
Sierpinski-Tetrahedrons and Octahedrons, GER
Advanced Problem Solving with DERIVE, Welke
Tools for Solving DEs, Etchells, UK
and
Setif, FRA; Vermeylen, BEL; Leinbach, USA; Aue, GER; Koller, AUT, Keunecke, GER, ...and others

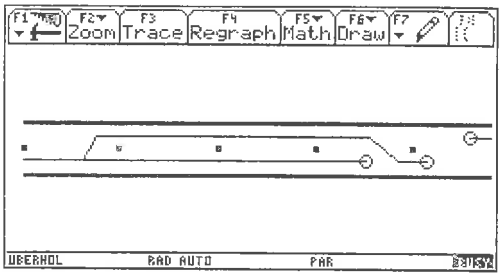
Impressum:
Medieninhaber: DERIVE User Group, A-3042 Würmla, D'Lust 1, AUSTRIA
Richtung: Fachzeitschrift
Herausgeber: Mag.Josef Böhm
Herstellung: Selbstverlag

Michel Beaudin



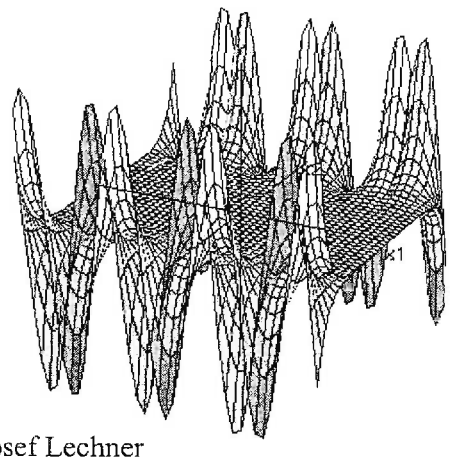


Regis Ockermann

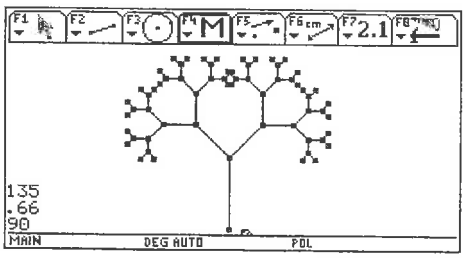


Thomas Himmelbauer

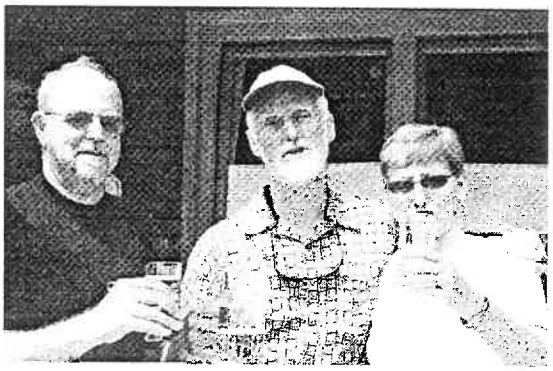
bk – Bernhard Kutzler

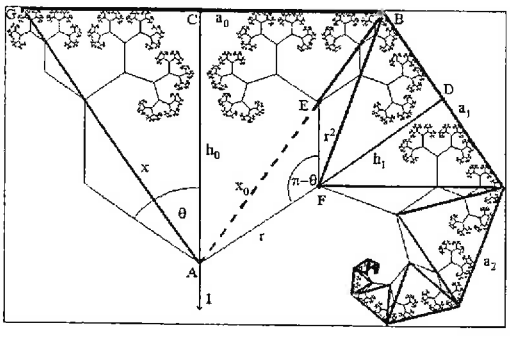


Josef Lechner

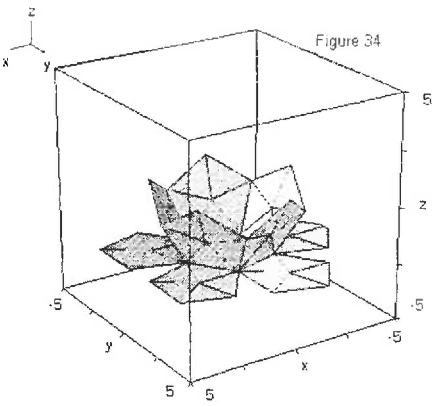


Dusan Pagon

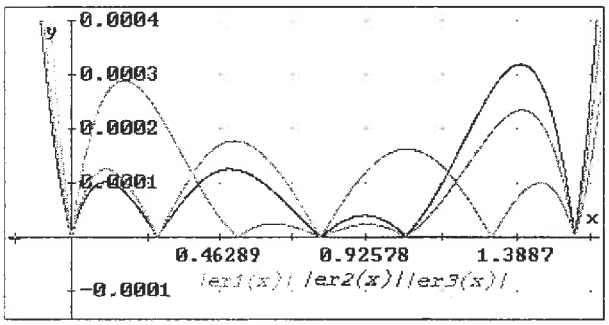




Karen & David Stoutemyer and Carl Leinbach

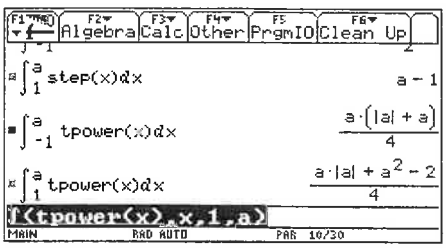


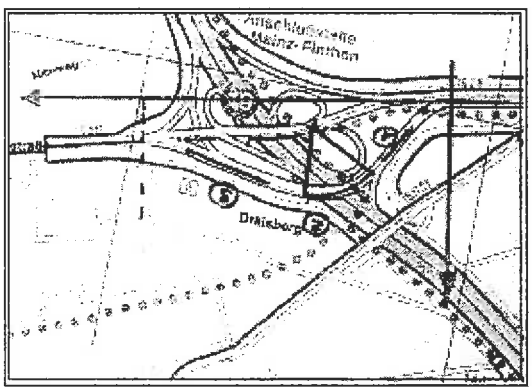
Theresa Shelby admires Peter Schofields workshop results?



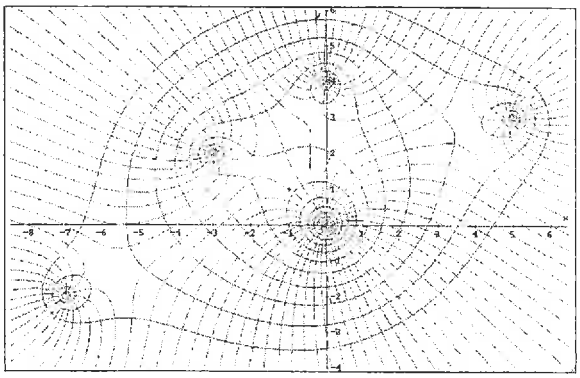
Esther Oppenhaim

Dennis Pence





Max-Günter Schröfel



Steven Schonefeld

Theresa and Noor Böhm, definitely not talking about *DERIVE* and *TI-92*.

Many thanks to Karl-Heinz Keunecke for sending the pictures, Josef

Rüdeger Baumann´s Challenge

A Fibonacci-like sequence is defined by

fieb(2) = 2, fieb(3) = 3

fieb(k+2) = fieb(k+1)+fieb(k) + 1

It should be checked, if p is divisor of fieb(p) for p is a prime number.

A non – efficient checkprogram is:

```
#1: fieb(k):=IF(k<=2,2,IF(k=3,3,fieb(k-2)+fieb(k-1)+1))
#2: Pz(n):=ITERATES(NEXT_PRIME(k),k,2,n)
#3: fiebeck(n):=VECTOR([p,fieb(p),FACTORS(fieb(p))],p,Pz(n))
```

Tasks: (a) Write an efficient *DERIVE*-procedure.

(b) Try to prove or to disprove the conjecture.

On Rüdeger Baumann's Challenge

Stefan Welke, Bonn, Germany, spwelke@aol.com

1 Introduction

We will present a complete solution to Rüdeger Baumann's conjecture on a Fibonacci-like sequence, which is stated in [1]. This sequence is defined recursively by

$$f_1 := 0, f_2 := 2 \text{ and } f_{n+2} := f_{n+1} + f_n + 1 \text{ for } n > 2 .$$

R. Baumann demands a proof or disproof of the following

Conjecture: If p is prime, then p is a divisor of f_p.

2 Proof

We follow the lines of [2] and [3] where details are thoroughly explained. The sequence $\{f_n\}$ is related to a sequence of vectors $\{v_n\} := \left\{\begin{pmatrix} f_n \\ f_{n+1} \end{pmatrix}\right\}$, and that sequence is recursively defined by the following matrix-vector equation:

$$v_1 := \begin{pmatrix} 0 \\ 2 \end{pmatrix} \text{ and } v_{n+1} := \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} v_n + \begin{pmatrix} 0 \\ 1 \end{pmatrix} \text{ for } n > 1$$

We denote the generating matrix $\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$ by A, thus the defining equation reads $v_{n+1} := Av_n + b$. The advantage of this approach is that we can perform calculations with matrices nearly as easy as with numbers. We have by induction: $v_n := A^{n-1}v_1 + \sum_{k=0}^{n-2} A^k b$ for $n > 2$. Since A is invertible, the sum can be written as $\sum_{k=0}^{n-2} A^k b = (A^{n-1} - id)(A - id)^{-1} b$.

It is easily checked that $(A - id)^{-1} = \begin{pmatrix} -1 & 1 \\ 0 & 1 \end{pmatrix}$, so this sum becomes

$$\sum_{k=0}^{n-2} A^k b = (A^{n-1} - id)\begin{pmatrix} -1 & 1 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix} = (A^{n-1} - id)\begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

So far we have obtained a fast way to compute R. Baumann's sequence because *DERIVE* performs powers of matrices with integer entries very fast. The first component of v_n is what R. Baumann calls fieb(n):

```
#1:   v(n) := [0  1]^(n - 1) · [1, 3] - [1, 1]
              [1  1]

#2:   v(1)

#3:                                    [0, 2]

#4:   v(2)

#5:                                    [2, 3]

#6:   fieb(n) := FIRST(v(n))

#7:   VECTOR(fieb(k), k, 1, 20)

#8:   [0, 2, 3, 6, 10, 17, 28, 46, 75, 122, 198, 321, 520, 842, 1363,
       2206, 3570, 5777, 9348, 15126]
```

The calculation of fieb(3000) needs less than 0.005 seconds on my AMD processor with 800 MHz.

```
#9:   fieb(3000)

#10:  91816503442594888422686731701278966326820191626819726640227410015833~
      68440483251110615691303051495962214344167699119879903041293018433 4~
      23671776650452385953378904543414919964436328678164690023513600991 0~
      79550893255869720561249967405687830839254211542449578309835573853 6~
      17098918308069988033243702522641674461396389498693623912542959915 0~
      36664374734432865545481167276447239833514672915937079435105712777 8~
      82162703125454702440677667013794700882514305375891348614047092399 7~
      33752635251442053936659895287421980709071953311101104332917565150 5~
      64119310090008345656172168802488666415134867269667161268139414799 1~
      43628911337502451029520200000001
```

A second demand in [1], which is unnecessary in spite of the final result, was an efficient *DERIVE*-procedure to check R. Baumann's conjecture. We use the simple fact that if p is a divisor of n, then $\gcd(p,n) = p$ resp. $\gcd(p,n) - p = 0$.

```
#11:  GCD(7, fieb(7))-7

#12:                                    0

#13:  VECTOR(GCD(k, fieb(k)) - k, k, VECTOR(NTH_PRIME(l), l, 1, 100))

#14:  [0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

We know until now that the conjecture is true for the first hundred primes.

Our next step is to develop a Binet-like formula for f_n. The eigenvalues of A are $\lambda_1 = \frac{1+\sqrt{5}}{2}$ and $\lambda_2 = \frac{1-\sqrt{2}}{2}$, where $\lambda_1 : 1$ is the golden ratio. Corresponding eigenvectors are $e_1 = \begin{pmatrix} 1 \\ \lambda_1 \end{pmatrix}$ and $e_2 = \begin{pmatrix} 1 \\ \lambda_2 \end{pmatrix}$. Thus we have $\begin{pmatrix} 1 \\ 3 \end{pmatrix} = \lambda_1 e_1 + \lambda_2 e_2$, and finally:

$$v_n = \lambda_1^{\,n} e_1 + \lambda_2^{\,n} e_2 - \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

The first row of this vector-equation gives the announced Binet-like formula:

$$f_n = \lambda_1^{\,n} + \lambda_2^{\,n} - 1 = \left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{1-\sqrt{5}}{2}\right)^n - 1$$

Now we have to show that p is a divisor of f_n if p is prime. We apply the Binomial theorem to the last formula, which is more appropriate for our investigations:

$$f_n = \frac{1}{2^n}\left(\sum_{k=0}^{n}\binom{n}{k}(\sqrt{5})^k + \sum_{k=0}^{n}\binom{n}{k}(-\sqrt{5})^k\right) - 1 = \frac{2\sum_{k=0}^{\lfloor \frac{n}{2} \rfloor}\binom{n}{2k}5^k - 2^n}{2^n} = \frac{\sum_{k=0}^{\lfloor \frac{n}{2} \rfloor}\binom{n}{2k}5^k - 2^{n-1}}{2^{n-1}}$$

Here $\lfloor x \rfloor$ means the greatest integer n with $n \le x$.

Since the conjecture is certainly true for $p = 2$ we can assume that p is a prime greater than 2. In this case we only have to show that p is a divisor of

$$\sum_{k=0}^{\lfloor \frac{p}{2} \rfloor}\binom{p}{2k}5^k - 2^{p-1} = 1 - 2^{p-1} + \sum_{k=1}^{\lfloor \frac{p}{2} \rfloor}\binom{p}{2k}5^k .$$

But as p is prime we have $2k < p$, so $\binom{p}{2k} = \frac{p!}{(2k)!(p-2k)!}$ must be divisible by p without remainder, because no prime factor in the denominator can cancel the factor p in the numerator. A consequence is that a prime $p > 2$ is a divisor of f_n if and only if p is a divisor of $1 - 2^{p-1}$ or equivalently of $2^{p-1} - 1$.
This last statement is a consequence of

Fermat's Little Theorem:

If p is a prime and a is an integer with $\gcd(p,a) = 1$, then $a^{p-1} = 1 \bmod p$.

This proves R. Baumann's conjecture.

3 The inverse conjecture is not true

A natural question is to ask whether this conjecture is true only for prime numbers or not. If this were true, then we had a new characterisation for prime numbers. A check of the first 704 integers would suggest a positive answer. But the following calculation shows that the contrary is true:

```
      test(k) :=
        If PRIME?(k) = false ∧ GCD(fieb(k), k) = k
#15:       [k, true]
           false

#16:  VECTOR(test(k), k, 700, 720)

#17:  [false, false, false, false, false, [705, true], false, false,

       false, false, false, false, false, false, false, false, false,

       false, false, false, false]

#18:  GCD(fieb(705), 705)

#19:                                    705
```

The fact that a natural number n, which is not prime, is a divisor of f_n seems to be a rare event. The next calculation shows that there are only seven numbers with this quality between 2 and 10000. The computation time is about 21 seconds:

```
#20:  SELECT(m ≠ false, m, VECTOR(test(k), k, 2, 10000))
```

$$\#21: \quad \begin{bmatrix} 705 & \text{true} \\ 2465 & \text{true} \\ 2737 & \text{true} \\ 3745 & \text{true} \\ 4181 & \text{true} \\ 5777 & \text{true} \\ 6721 & \text{true} \end{bmatrix}$$

Thus R. Baumann's conjecture does not lead to a new characterisation of prime numbers.

References

[1] "Rüdeger Baumann's Challenge", DNL #46, p. 17

[2] Stefan Welke: "Lucas, Fibonacci, Pell & Company" , DNL '29, p. 30

[3] Stefan Welke: "Eine Verallgemeinerung der Binet-Formel für linear rekursive Folgen", MNU 51/5, 1998

HIV and the Immune System - A Mathematical Model

Josef Lechner, Amstetten, Lower Austria

Abstract

We discuss a mathematical model of increasing the HI-viruses in the human body. This model gives an answer to the question why the space of time between HI-infection and outbreak of AIDS differs to a great extent. It also gives insight into the phenomenon that our immune system is generally not able to root out HI-viruses completely.

This example is interesting from different points of view: It is well suited for lessons combining subjects like mathematics and biology. It demonstrates that even extensive simulations can be well done by the TI92/TI89 or DERIVE. It emphasizes the usefullness of recursively defined sequencies which prove to be a multi-functional tool.

The starting point of the model to be discussed is the increase of HI-viruses in the human body. This model gives an answer to the question why the space of time between HI-infection and outbreak of AIDS differs to a great extent. Whereas some of the concerned die within months after infection, others (but only a rather small number) are still alive without any troubles even 20 years later. After about 10 years half of all the infected suffer from AIDS. The following model - mainly developed by the Austrian mathematician M.Nowak - gives also insight into the phenomenon that our immune system is generally not able to root out HI-viruses completely as it is able to do with other types of viruses.

A paradox result of this model is the evidence that HI-viruses are that „successfull" because their replication is susceptible to mistakes. As the transcription of virus-RNA to DNA is regularly connected with random mistakes, the likelihood for a completely correct transmission of the genetic information is only about 40 percent.

Against every mutated virus the immune system must create new specific killer cells, which can only fight this special kind. The resistant cells act as *specialists*. On the contrary all mutating viruses can destroy *all* kinds of resistant cells against HIV or at least impair their function. The work as *generalists*.

In a tough fight against steadily appearing new kinds of AIDS-agents, which are generated in the body of the infected person, the immunse system is not able to work as efficiently as it should to maintain the balance. If a certain variety of viruses is exceeded, the immune system finally loses control of them and AIDS breaks out. In this way the number of virus-paricels increases sharply whereas the number of immune cells drastically decreases. [NOWAK, 1992]

According to [LIPPA, 1997 bzw. REICHEL, 1999] the number $v_j^{(i)}$ of the viruses of the kind i, j steps of time after infection can be described by means of the following (iterative) equation:

$$v_{j+1}^{(i)} = v_j^{(i)} + v_j^{(i)} \cdot (R - P \cdot a_j^{(i)})$$

In this context $a_j^{(i)}$, which affects the virus mutant i, is the number of immune cells per volume of blood. We call the rate of increase of the viruses R, supposing (in this simplified model) that R is equal ly big for all mutants (which is actually not the case!) The factor P describes the efficiency of the immune cells in their fight of resistance. $0 < P < 1$ must hold, otherwise one immune cell would destroy more viruses than could exist at all.

For the developement of the resistant cells (T-helper cells, B-cells and antibodies respectively, killer-cells) an (iterative) equation of the following kind can be given.

$$a_{j+1}^{(i)} = a_j^{(i)} + K \cdot v_j^{(i)} - U \cdot v_j \cdot a_j^{(i)}$$

Here the factor of proportionality K describes the increase of the kind i of the resistant cells, which is generated by the mutant i of the viruses. The factor U characterises the aggressiveness of the viruses. By analogy it is: $0<U<1$. The term v_j describes the total number of all viruses after j steps of time.

The following aspects show us why the simulation of the mathematical model is not easy:

- First of all the number of mutants is unknown. In the simplest case this number is 1.This case shows the effect of the resistant cells well, but cannot explain the breakout of AIDS. Using relatively realistic parameters there are 11 mutants, resulting in at least 22 difference equations.

- In every step of iteration the total number of viruses v_j must be calculated. The (iterative) equations, which describe the development of the various types of viruses (and consequentely the ones of the (iterativ) equations of the resistant cells) are combined by the term $v_j := \sum_{i=1}^{j} v_i$.

- The number of virus mutants as well as the kind and time of their appearance shall be controlled in this simulation.
- The more (iterative) equations (difference equations) are used the higher the likelihood of mistakes and the less useful the simulation for application at school. Therefore it is decisive that the model remains applicable. Supporting programs or utility files, which create the respective equations automatically, prove to be a practicable way.

Tools which are available for math lessons have their specific strengths and weaknesses. No matter which tool is used, by using the medium adequately a simulation of this extensive model is practicable.

Realisation of HIV-simulation by means of TI-92

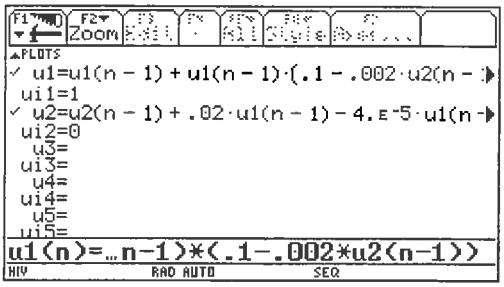
Simulation 1

Let's start with a trivial model with only one type of virus. In this case it's easy to perform a simulation with the CA-TR TI-92 and the TI-89 respectively. According to [LIPPA, 1997, NOWAK, 1992] the following parameters are relatively realistic:

> R=0,1 / P=0,002 / K=0,02 / U=0,00004
> One step in time represents 0,005 years (i.e. 200 steps describe a year). In this example there are M=5000 cells in the reviewed volume of body liquid. The treshold value for the outbreak of AIDS is in this case 11 mutants, the limit for the total number of immune cells is 500.

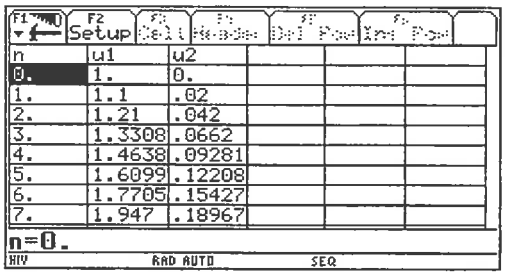
The sequence-mode of TI-92 allows the direct input of the respective equations of the model. The names of the equations are determined by `u1, u2, u3, ...` in this calculating system. In our case `u1(n)` is the number of viruses of type 1 after n steps of time, `u2(n)` the number of resistant cells of type 1 after n steps.

Viruses (type 1)	u1(n)=u1(n-1)+u1(n-1)·(0.1-0.002·u2(n-1))
Starting value viruses	ui1=1
Resistant cells (type 1)	u2(n)=u2(n-1)+0.02·u1(n-1)-0.00004·u1(n-1)·u2(n-1)
Starting value res. cells	ui2=0



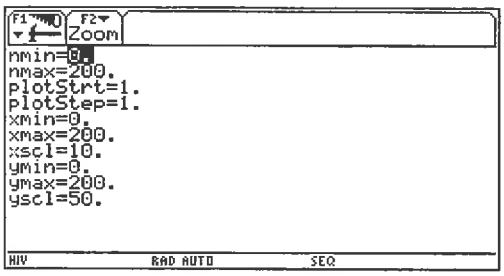
Now the development of the viruses and the resistant cells respectively can be observed and evaluated by means of tables or graphs with all the available tools of the calculator (e.g. TRACE).

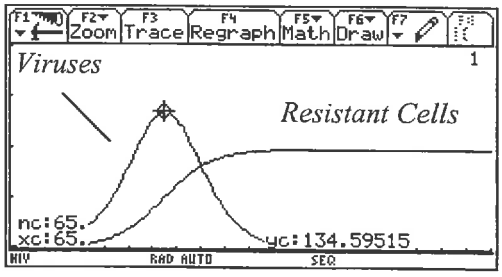
Table



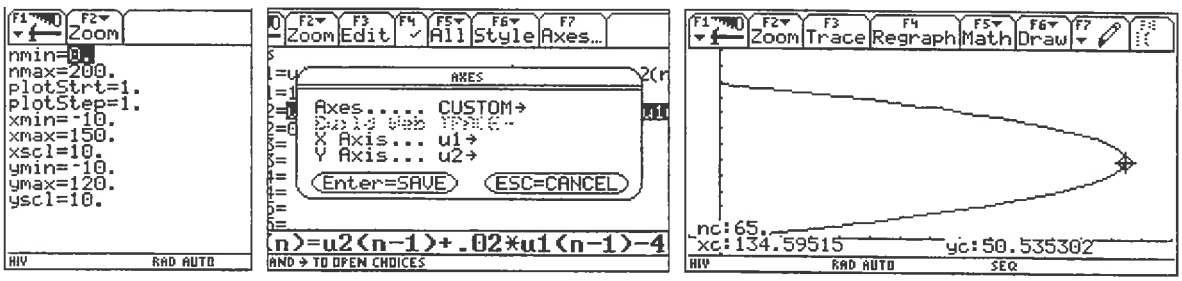
n	u1	u2
0.	1.	0.
1.	1.1	.02
2.	1.21	.042
3.	1.3308	.0662
4.	1.4638	.09281
5.	1.6099	.12208
6.	1.7705	.15427
7.	1.947	.18967

Graphic presentation of the development in time





Graphic presentation of resistant cells against viruses



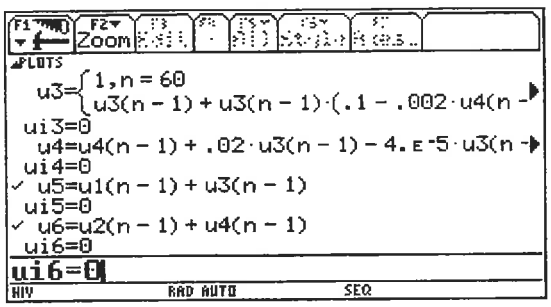
Comment: One must switch SEQUENCE Mode / F7 to Custom. Choose u1 and u2 for axes.

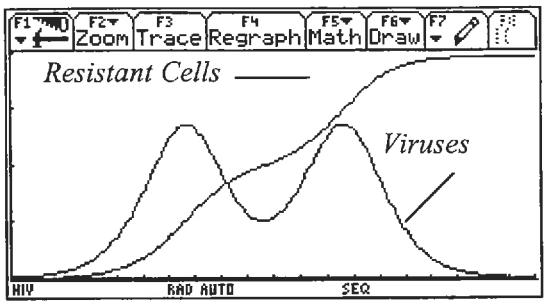
Simulation 2

Now we want to work with two mutants, the second of which shall appear after 60 steps of time (which means after about 3.6 months). This can be reached by a simple branching out using the when-control structure. The respective inputs in the sequence-mode of the TI now read:

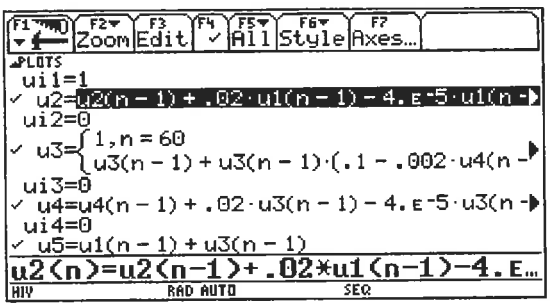
Viruses (Typ1)	`u1(n)=u1(n-1)+u1(n-1)·(0.1- 0.002·u2(n-1))`
Starting value viruses	`ui1=1`
Resistant cells (Typ 1)	`u2(n)=u2(n-1)+0.02·u1(n-1)-0.00004·u1(n-1)·u2(n-1)`
Starting value resistant cells	`ui2=0`
Viruses (Typ2)	`u3(n)= when(n=60,1 ,` `u3(n-1)+u3(n-1)·(0.1-0.002·u4(n-1)))`
Starting value viruses	`ui3=0`
Restistant cells (Typ 2)	`u4(n)=u4(n-1)+0.02·u3(n-1)-0.00004·u3(n-1)·u4(n-1)`
Starting value resistant cells	`ui4=0`
Total number viruses	`u5(n)=u1(n-1)+u3(n-1)`
Starting value viruses	`ui5(n)=0`
Total number resistant cells	`u6(n)=u2(n-1)+u4(n-1)`
Starting value resistant cells	`ui6(n)=0`

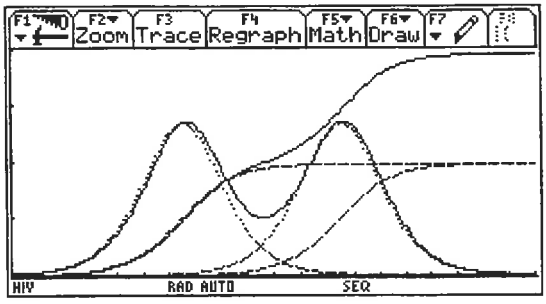
Total number of viruses and resistant cells.





By ticking the respective (iterative) equations and choosing an appropriate graphing style F6 the development of the individual types of viruses can be observed.

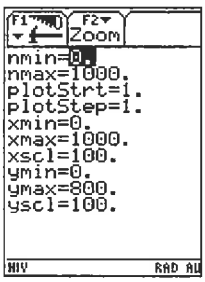


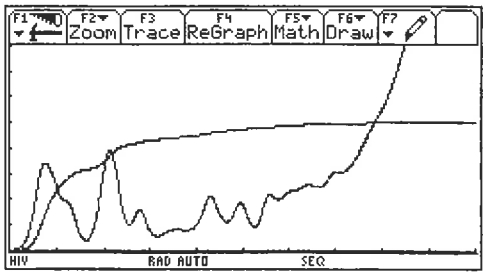


As it can be seen above, the input of the simulation equations is already extensive with only two mutants. Therefore it seems to be advisable to use a suitable program for simulations with more mutants.

Simulation 3

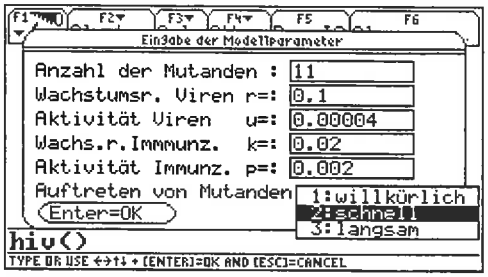
Now the number of mutants shall be increased to 11. The time for the appearance of the mutations shall be determined at random. This may lead to the following result. Here the breakout of AIDS happens after four years. The screen below shows another simulation on progress.



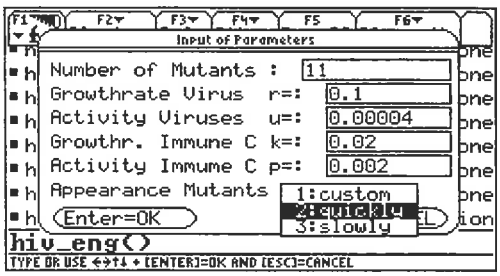


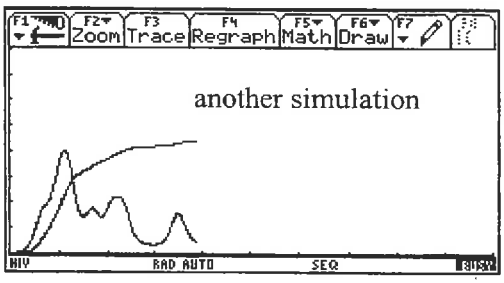
How was this performed?

We used the program **hiv**, which was started by the input of `hiv()` in Home Screen.
The program allows the input of model parameters in the first step.



(*You will find also an English version* `hiv_eng()` *of the program to download from our website.*)

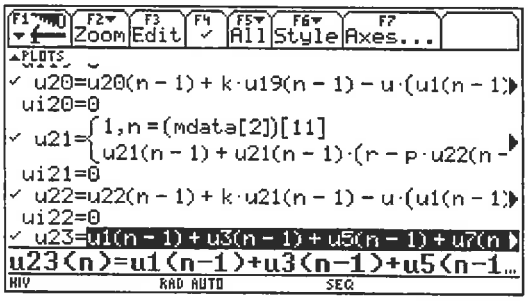




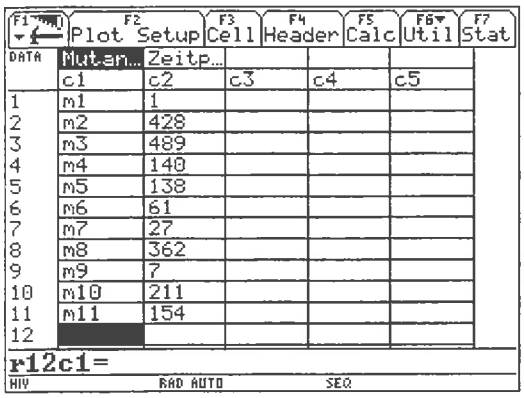
The item „Appearance of mutants" allows the use to control the time of appearance of the individual mutants on purpose or to select it quickly or slowly by accident. In the case of ‚quickly' all mutants appear within 500 steps of time (2.5 years), in the case of slowly within 4000 steps of time (20 years). These values can be easily changed. (One step ≡ 0.005 years)

After this first step the calculator is busy for some time - the respective equations are generated.

Finally one can see that the equations have been compiled correctly in the [Y=]-Editor (in SEQUENCE-Mode).



In a second step the appearance of mutants is controlled. If one has decided to choose one of the two random methods, one can see in the `Data/Matrix`-Editor, when new mutants will appear.



The values in column c2 describe, after how many steps of time the individual mutants appear. They can be changed at any time. If the user has chosen not a random method, the values must be supplied.

Three final remarks:

(1) It is incredible that a calulator - like the TI-92 - allows to perform simulations, which is a pretty hard job for older PCs. When carrying out the simulation with the help of a spreadsheet program (as shown by [LIPPA 1997]), one gets huge tables, which can hardly be handled.
In my opinion a simulation works successfully, because the TI calculates anew every time when a table is generated. Moreover it only calculates for the period you choose.
When a graph is drawn, not all calculated values a stored, but only the positions of the pixel on the screen.

(2) One should not neglect to mention the long duration of such a calculation. A simulation with 11 mutants and 1000 steps of time may take you up to one hour (!).

(3) We will not print the program but one remark is necessary. The program starts

```
hiv()
Prgm
©(c) Josef Lechner, Mai 1999
©lejos@nol.at

© Sequence-Bereich loeschen
© Clear the Sequence-Editor

RclGDB default

© Lokale Variablen
```

You first must store the empty Graphic Data Base under the variable name `default`. So set the Sequence mode. Clear all entries in the Y=Editor and then type enter in the Home Screen `stogdb default`.

Realisation of the HIV-Simulation by means of *DERIVE*

If you want to use the CAS *DERIVE* for a simulation, you have to choose an other way. As every simulation of a dynamic system is an iterative process, you have to describe how to get from one condition to the next. Using *DERIVE* `ITERATES` is the proper command. Its efficiency is the basis for the following implementations.

Simulation 1

Again we will start with the trivial model with only one type of virus. The following parameters shall be remembered: R=0,1 / P=0,002 / K=0,02 / U=0,00004

HIV-Simulation
(c) by Josef Lechner, Mai 1999
performed with Version 5.05 in summer 2002

```
#1:   [r := 0.1, p := 0.002, k := 0.02, u := 0.00004, mz := 3]
```

After choosing the model parameters the iterative equations are defined. `NV(v,a)` describes the number of viruses after every step of iteration, `VA(v,a)` the number of resistant cells after every step of iteration, v and a represent the currently existing viruses and resistant cells respectively.

```
#2:   NV(v, a) := v + v·(r - p·a)
#3:   NA(v, a) := a + k·v - u·a·v
```

With the help of `ITERATES` the different values are calculated. The first list shows the two iteration equations, the second list the iteration variables, the third the initial values and the last parameter indicates how many iteration steps are wanted. Afterwards the calculation is started using the `APPROX`-command.

```
#4:   ITERATES([NV(v, a), NA(v, a)], [v, a], [1, 0], 3)
```

```
#5:
⎡   1         0     ⎤
⎢                   ⎥
⎢  1.1       0.02   ⎥
⎢                   ⎥
⎢1.20995  0.0419991 ⎥
⎢                   ⎥
⎣1.33084  0.0661962 ⎦
```

If you also want to visualize the steps of time in the table - which is important for a later graph - you can calculate the values in the time-column iteratively simply by adding a further item in the corresponding list.

```
#6:   ITERATES([t + 1, NV(v, a), NA(v, a)], [t, v, a], [0, 1, 0], 3)
```

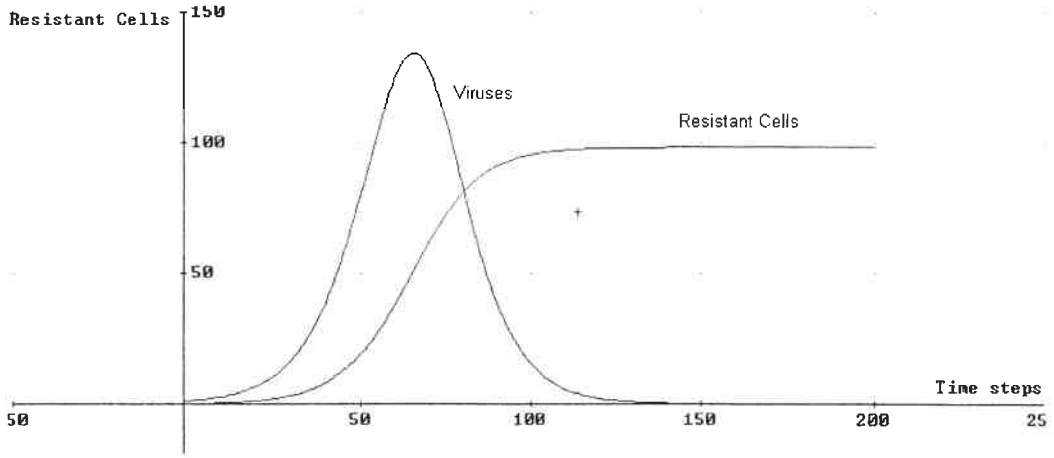
```
#7:
⎡0     1         0     ⎤
⎢                      ⎥
⎢1    1.1       0.02   ⎥
⎢                      ⎥
⎢2  1.20995  0.0419991 ⎥
⎢                      ⎥
⎣3  1.33084  0.0661962 ⎦
```

To be able to show the time-development of the system, you must create two tables: one table time versus viruses, the other time-vs resistant cells. The output is restricted to the desired columns for plotting by using the command SUB SUB [column1, column2].

```
#8:   (ITERATES([t + 1, NV(v, a), NA(v, a)], [t, v, a], [0, 1, 0], 200))↓↓[1, 2]
#9:   (ITERATES([t + 1, NV(v, a), NA(v, a)], [t, v, a], [0, 1, 0], 200))↓↓[1, 3]
```

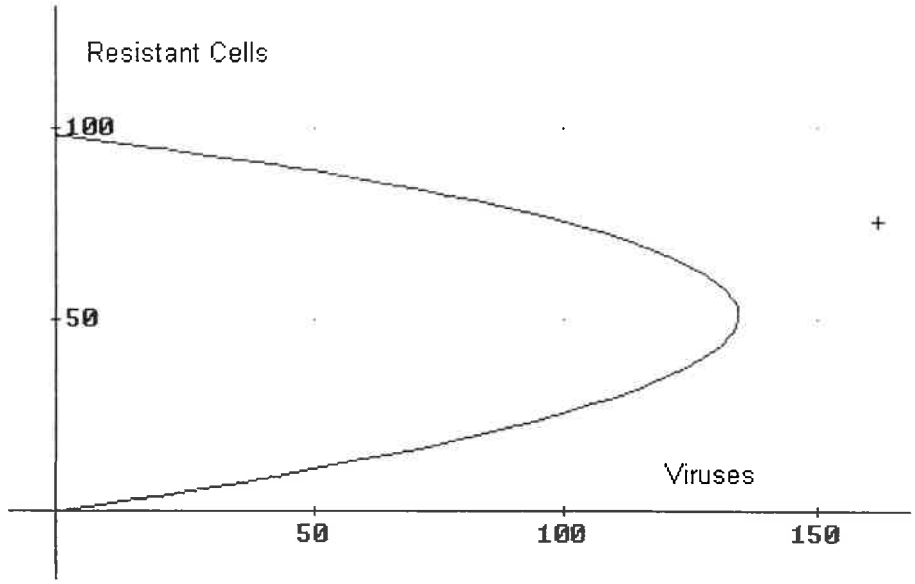
In Version 5 you can apply the COL Operator instead of sub sub .....

```
#10:  (ITERATES([t + 1, NV(v, a), NA(v, a)], [t, v, a], [0, 1, 0], 200)) COL [1, 2]
#11:  (ITERATES([t + 1, NV(v, a), NA(v, a)], [t, v, a], [0, 1, 0], 200)) COL [1, 3]
```



If you only plot the two original colums, you get the phase diagram of the system:

```
#12:  ITERATES([NV(v, a), NA(v, a)], [v, a], [1, 0], 200)
```



Simulation 2

If you want to work with more mutants, the shown method is only partly suitable, because you need at least the double number of iterative equations than the number of mutants. So that the number of mutants does not play a role on the one hand and on the other hand to create a clearly structured simulation it is recommended to express the current condition of the system in the form of a list (vector) `z`. It shall contain the step of time, for each mutant and its type of resistant cell the respective equations and finally the total numbers of viruses and resistant cells. Additionally a second list (a second vector) `mdata` shall indicate the steps of time of the appearing mutants. This vector can be - if you want to - generated at random.

The current system parameters are chosen and defined first. You also have to provide the number of mutants. The times of appearance are combined in the list `mdata`.

```
#11:  mdata := [1, 5, 10]
#12:  z0 := VECTOR(0, i, 2·mz + 3)
```

The list z_0 contains the starting situation (in this case a list with only zeros as items).

The two definitions calculate the total number of viruses and resistant cells:

$$\#13:\quad SU(z) := \sum_{k=1}^{mz} z_{2\cdot k}$$

$$\#14:\quad SA(z) := \sum_{k=1}^{mz} z_{2\cdot k+1}$$

The now following three definitions form the core of the simulation. `NV(z,i)` calculates the number of viruses of type i using the current condition z of the system. `NA(z,i)` calculates the number of resistant cells against the virus mutant i also using the current condition z.
`NZ(z)` finally summarizes the iterativ equation for time, all mz iterativ equations for the respective mutants and the total numbers of viruses and resistant cells.

```
      NV(z, i) :=
        if z↓1 = mdata↓(i/2)
#15:       1
           z↓i + z↓i·(r - p·z↓(i + 1))

      NA(z, i) := z        + k·z    - u·SU(z)·z
#16:                i + 1       i              i + 1

      NZ(z) := APPEND([z  + 1], APPEND(VECTOR([NV(z, i), NA(z, i)], i, 2, 2·mz, 2)),
#17:                  [1]
        [SU(z), SA(z)])
```

Again the calculation is carried out by means of the `ITERATES`-command. This can be done in a surprisingly simple manner. The first function `NZ(z)` calculates the next using the current condition of the system, `z` contains the respective current condition, `z0` the initial condition and the last parameter indicates the number of iterations.

```
#18:  ITERATES(NZ(z), z, z0, 15)
```

provides the following table:

#21:

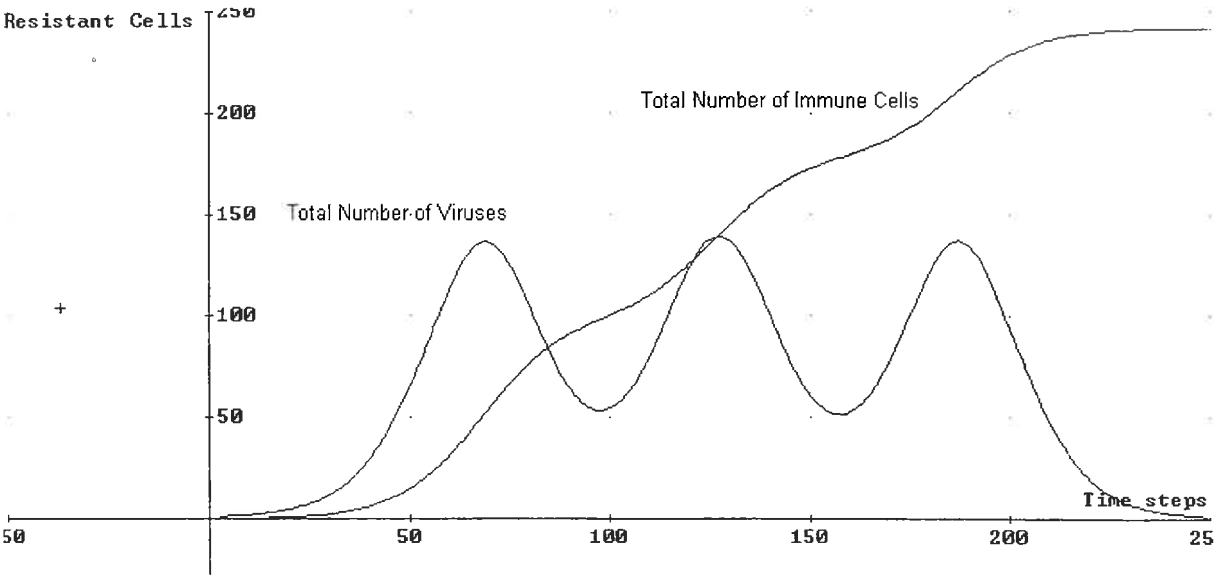
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	1	0	0	0	0	0	0	0
3	1.1	0.02	0	0	0	0	1	0
4	1.2099	0.041999	0	0	0	0	1.1	0.02
5	1.3308	0.066196	0	0	0	0	1.2099	0.041999
6	1.4637	0.092809	1	0	0	0	1.3308	0.066196
7	1.6098	0.12207	1.1	0.02	0	0	2.4637	0.092809
8	1.7704	0.15425	1.2099	0.041997	0	0	2.7098	0.14207
9	1.9469	0.18965	1.3308	0.066191	0	0	2.9804	0.19625
10	2.1409	0.22856	1.4637	0.092800	0	0	3.2778	0.25584
11	2.3540	0.27135	1.6098	0.12206	1	0	3.6046	0.32136
12	2.5881	0.31837	1.7704	0.15423	1.1	0.02	4.9638	0.39341
13	2.8453	0.37007	1.9469	0.18961	1.2099	0.041995	5.4586	0.49261
14	3.1277	0.42688	2.1409	0.22850	1.3308	0.066184	6.0022	0.60167
15	3.4378	0.48932	2.3540	0.27126	1.4637	0.092784	6.5995	0.72157

Finally a graph of any mutant and any type of resistant cell can be plotted by selecting only two columns.

Example 1: 3 Mutants

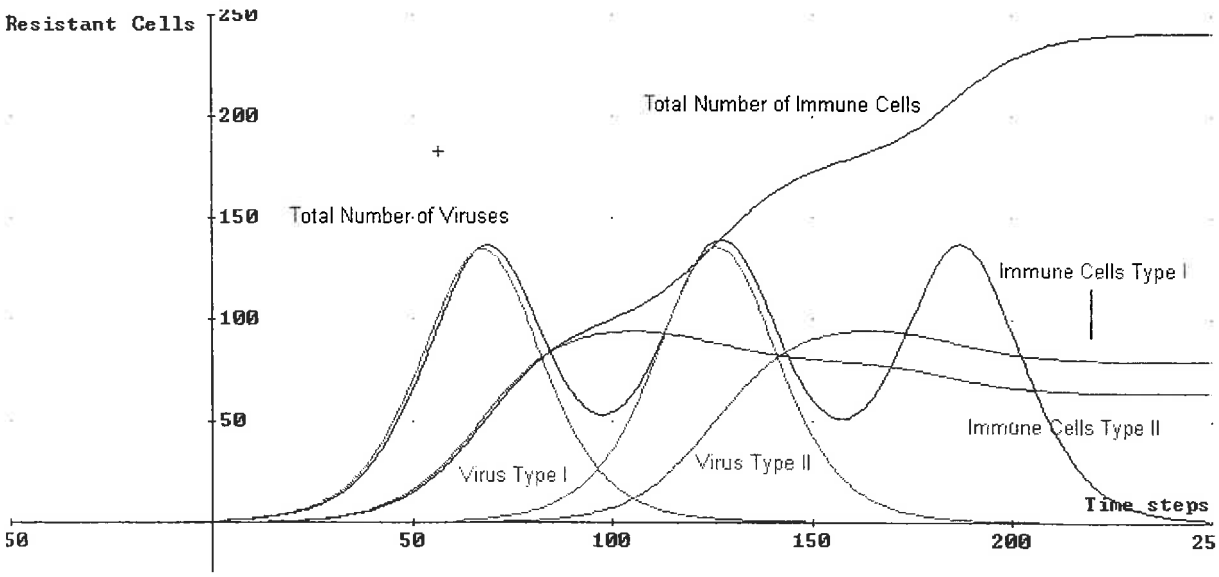
First choosing the model parameters and then calculating the model:

```
#22:  [r := 0.1, p := 0.002, k := 0.02, u := 0.00004, mz := 3]
#23:  mdata := [1, 60, 120]
#24:  (ITERATES(NZ(z), z, z0, 300))↓↓[1, 8]
#25:  (ITERATES(NZ(z), z, z0, 300))↓↓[1, 9]
```



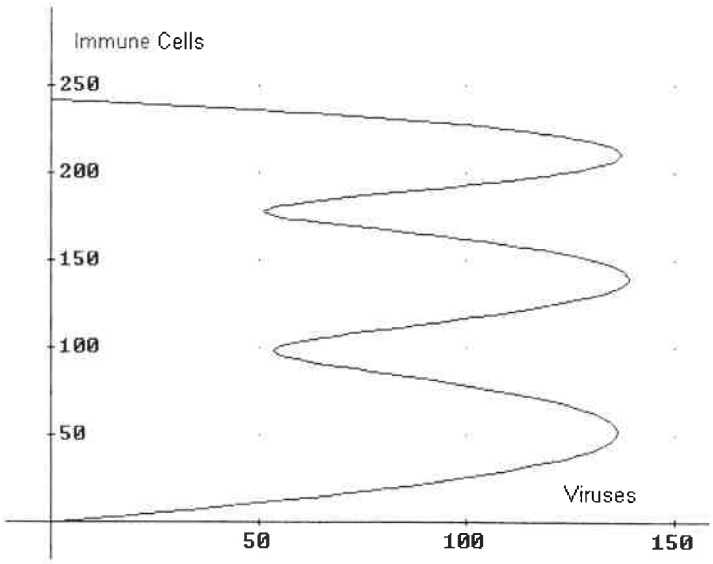
The development of the individual mutants and the immune cells respectively can be observed very well:

```
#26:  (ITERATES(NZ(z), z, z0, 300))↓↓[1, 2]
#27:  (ITERATES(NZ(z), z, z0, 300))↓↓[1, 3]
#28:  (ITERATES(NZ(z), z, z0, 300))↓↓[1, 4]
#29:  (ITERATES(NZ(z), z, z0, 300))↓↓[1, 5]
#30:  (ITERATES(NZ(z), z, z0, 300))↓↓[1, 6]
```



When limiting the table to the columns showing the total numbers of viruses and resistant cells you get the phase diagram of the system:

```
#31:  (ITERATES(NZ(z), z, z0, 300))↓↓[8, 9]
```



Example 2: 11 Mutants

First choosing the model parameters:

```
#32:  [r := 0.1, p := 0.002, k := 0.02, u := 0.00004, mz := 11]
#33:  mdata := [1, 428, 489, 140, 138, 61, 27, 362, 7, 211, 154]
```

The same values for the time of appearance of the mutants as for the simulation with the help of the TI-92 have been chosen.

This list should be compiled at random, which can be achieved in the following way:

Vector of mutants appearances randomly generated

```
#34:  mutants(mz, periods) := APPEND([1], VECTOR(RANDOM(periods) + 1, i, 1, mz - 1))
```

Examples:

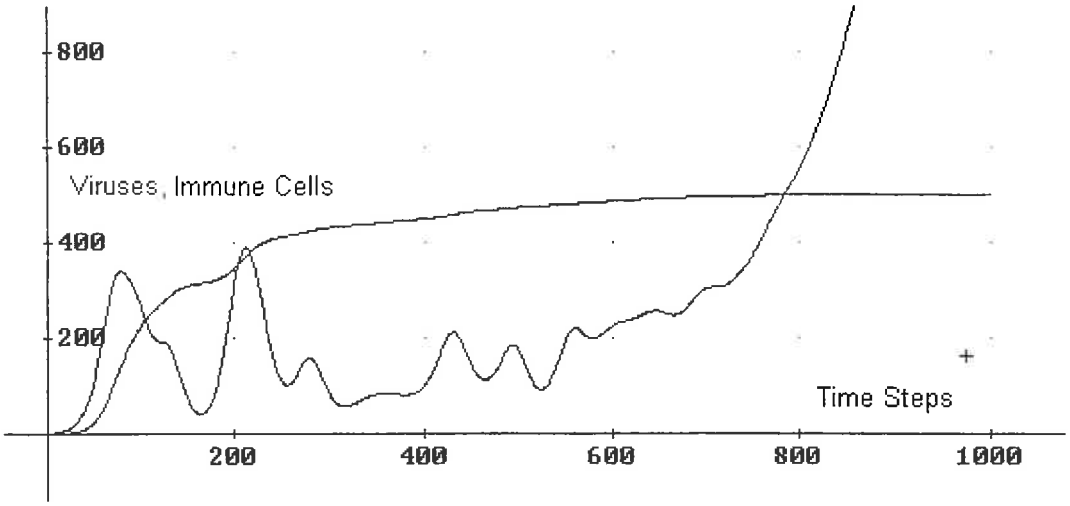
```
#35:  (mdata := mutants(3, 200)) = [1, 47, 103]
#36:  mdata := mutants(11, 400)
#37:  mdata = [1, 389, 267, 89, 31, 259, 57, 288, 358, 173, 110]
```

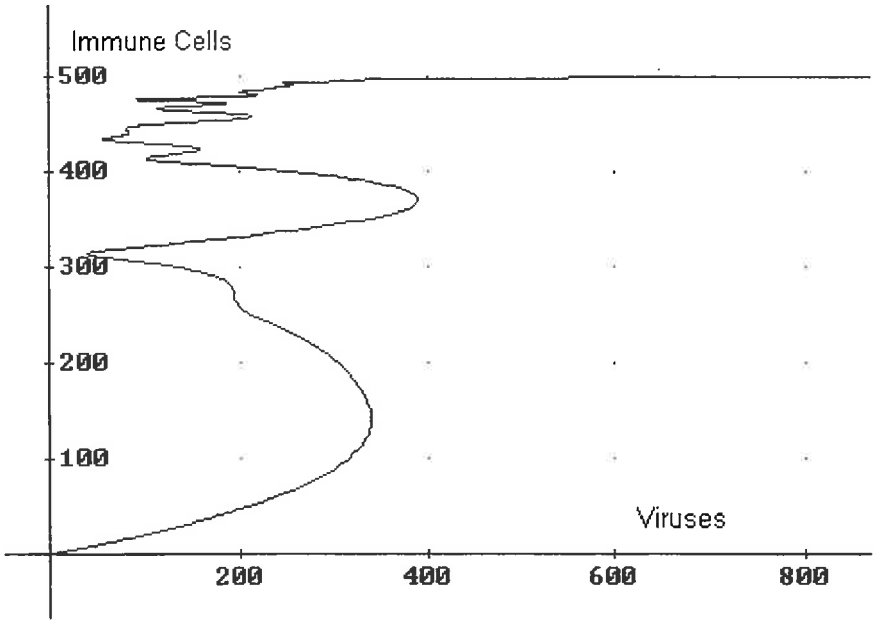
The modul creates a list with starting times, if you provide the number of mutants `mz` and the time span `periods` within they shall appear. Each evaluation leads to a new random list.

The two following definitions may help you to find the correct columns where you can find the total numbers of viruses and resistant cells respectively. The evaluation of the three `ITERATES`-lines allows to get graphs.

Go back to the given data from above and calculate the table

```
#38:  mdata := [1, 428, 489, 140, 138, 61, 27, 362, 7, 211, 154]
#39:  [v := 2·mz + 2, a := 2·mz + 3]
#40:  (ITERATES(NZ(z), z, z0, 1000)) COL [1, v]
#41:  (ITERATES(NZ(z), z, z0, 1000)) COL [1, a]
#42:  (ITERATES(NZ(z), z, z0, 1000)) COL [v, a]
```



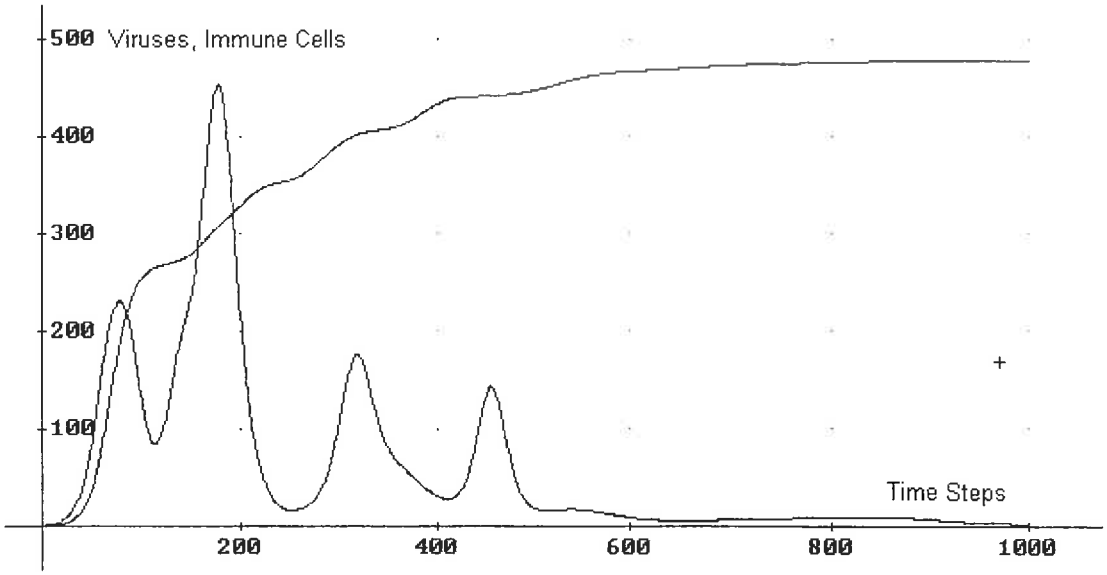


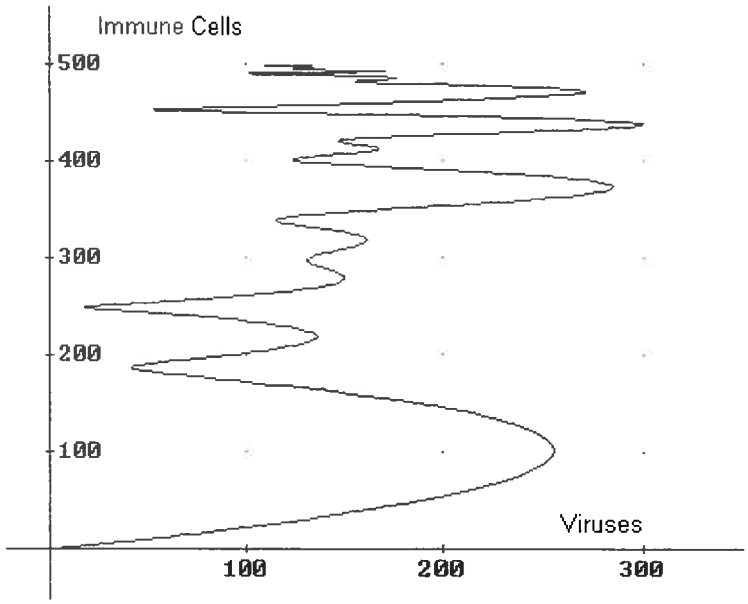
Calculation time ca 270 secs using PC 486, 120 MHz and ca 60s using Pentium II, 333 Mhz.

(This was in 1999. It took me now – 2002 – about 13.6s for each calculation on an very ordinary PC - and this is quite reasonable. Josef)

The next page shows another simulation using only another randomly chosen list `mdata`.

```
#43:  mdata = [1, 109, 362, 374, 141, 197, 35, 51, 231, 152, 114]
```





In this case we will have no break out of the disease.

Literature

LIPPA, M. (1997): HIV und Immunsystem - ein mathematisches Modell und seine Realisierung mit EXCEL, MNU 50/5, 295-301

NOWAK, M.A. (1992): Variability of HIV infections. J. of Theoretical Biology 155, 1-20

NOWAK, M.A.; MCMICHAEL (1995): Scientific American, August 1995.

REICHEL, H.-CH. (1999): Differenzengleichungen in der Oberstufe und ein aktuelles Beispiel über Mathematik und AIDS. To appear in: Didaktik-Reihe der ÖMG.

How I Learned Loving Parameters
or
How Many Faces Are There in the World?

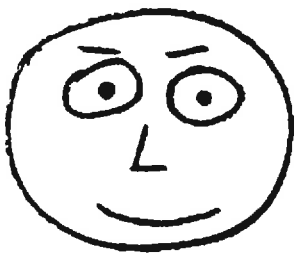
Josef Böhm, Würmla, Austria

I wrote this paper in German and showed it to my friend Carl Leinbach when he stayed with us for some days at the occasion of VISIT-ME 2002. He took the paper with him and one months later I received an email with an attachment, containing my paper translated into English. He wrote:
Finally, I enjoyed your parameters paper so much that I was moved to translate it. I have attached the result. I used my German-English Dictionary and knowledge of the math and Derive/TI to work out a paper that makes sense to me.
I am very grateful that he did a marvellous job. This is a nice way to learn German, Carl.
In my last DNL I gave some hints and advice how to overcome the boring way editing programs and more complex functions using the edit line only. I'll repeat this once more and I am very happy to be able to present a tool provided for this purpose by Albert Rich for the convenience of the DERIVIANS. Now let's go on:

In Clifford Pickover's very excellent book, *Computers, Pattern, Chaos and Beauty – Graphics from an Unseen World* [1], there is a short chapter called "Autocorrelation Cartoon-Faces for Speech". In this chapter several different forms of faces were generated. This struck me as a subject for a lesson on parameters. In fact it struck me that this idea could lead to an excellent motivational tool for learning about parameters.

I have done this project first in *DERIVE* and then converted it almost step by step for the TI-92.

Beginning with a stylized face, similar to a children's rhyme belonging to a stick figure, called "Punkti-Punkti, Strichi-Strichi" in Austria and Germany. Isn't it a real "Moon Face" (= "Mondgesichti")?



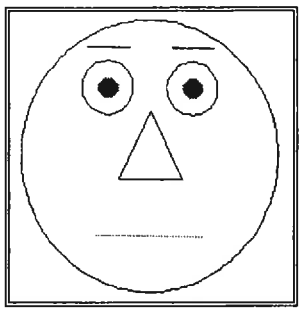
OK, now what characteristics of this face can we vary? The items in this list are not the only possibilities, but to make the task not too easy, here are my choices. The list can be extended (Ears, Hair, etc.).

1. The shape of the face (`face_f = p1`)
2. The shape of the eyes (`eye_f = p2`)
3. The size of the eyes (`eye_s = p3`)
4. The space between the eyes (`eye_sp = p4`)
5. The size of the pupils (`pup_s = p5`)
6. The length of the eyebrows (`brow_l = p6`)
7. The inclination of the left eyebrow (`brow_sl = p7`)
8. The inclination of the right eyebrow (`brow_sr = p8`)
9. The size of the nose (`nose_s = p9`)
10. The distance between the ends of the mouth (`mouth_l = p10`)
11. The shape of the lower lip (`uplip_f = p11`)
12. The shape of the upper lip (`lowlip_f = p12`)

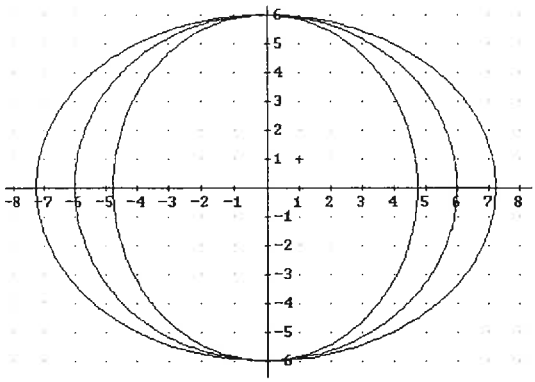
In addition to the TI-Version I add two parameters `center_x` and `center_y` (default both = 0) which are the coordinates of the face center. On the TI I can only show one face but on the PC-screen I can present a whole sequence of faces having different centers.

I leave the size of the face fixed, this will give us the prettiest ones to look at. In the list I have included within the parentheses the names of the parameters which are responsible for the variation of the attributes of the face. We will choose values of our parameters in the interval ($-5 \leq$ *parameter value* ≤ 5). It is not necessary to use integers only, but for now we will start with integer parameters.

The "Typical Value" for all characteristic parameters is 0. This gives us the "Typical Face" from which all others follow:



We begin with the shape of the face as a circle with radius 6. Via the parameter form of an ellipse we could obtain various face forms varying from longish to broad – this is valid for the TIs. In *DERIVE* we can use the implicit form of the ellipse to obtain the face `fa` depending on the parameter `p1`.



```
fa:=(x-m)^2/(6+0.25p1)^2+(y-n)^2/36=1
```

With $-5 \leq$ `p1` $\leq +5$ and `m, n` $= 0$ we will receive ellipses from $\frac{x^2}{4.75^2}+\frac{y^2}{6^2}=1$ to $\frac{x^2}{6.25^2}+\frac{y^2}{6^2}=1$.

(For the TI I took [(5+0.4*f_form)*cos(t), 5 + sin(t)] with $0 \leq t \leq 2\pi$ and $-5 \leq$ f_f ≤ 5, giving smaller and broader faces because of taking 0.4 instead of 0.25 and another effect of the parameter in this form.)

This first example shows, that it does not depend if working with the trigonometric representation or not, the important fact – application of a parameter and studying its influence – is remaining.

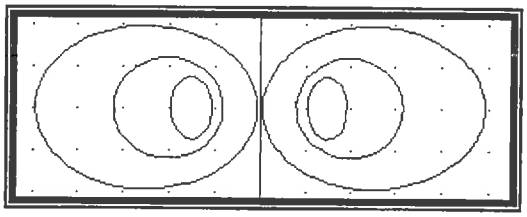
Three parameters are required to describe the eyes. They are the distance between the eyes, the size of the eyes, and the shape of the eyes.
The "Standard Eyes" have their centers in (± 2, 3) and have the form of a circle with radius 1.2.
For the right eye, we can write the following implicit form of an ellipse containing three parameters `p2`, `p3` and `p4` (and `m, n`, of course – which must be considered in almost all details).

```
eyr_f:=(x-m-2-0.1p4)^2/(1.2+0.15p3+0.1p2)^2+(y-n-3)^2/(1.2+0.15p3)^2=1
```

You can see: `p4` translates horizontally the right eye between m+1.5 and m-1.5, `p2` is resposible for the form of the eyes (in the same way as `p1` did for the face form) and as `p3` is representated with the same factor 0.15 in both expressions it varies the size of the eye between 0.45 and 1.95.
By substituting for the parameters the students can inspect the effect of the parameters one after the other and in combination as well. Changing two signs gives the instruction for the left eye.
(If you would like to "improve" your model you could introduce different parameters for both eyes!)

See on the next page the "Standard Eyes" together with the [-5,-3,-3] and the [5,5,5] eyes. The pupils are still missing and they will be added later.

As a teacher you could also present the expression `eyr_f` and let the students investigate and describe the effect of the parameters and their interplay. So step by step we can creating and designing our face.



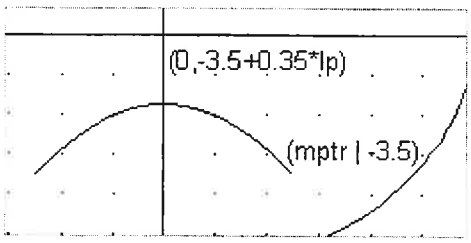
As I want to accentuate the pupils I define the area of the pupils by two inequalities – considering parameter `p5` for their size and not to forget also parameter `p4`, because the location of the pupils will strongly depend on the distance between the eyes – see in the listing below `pupr_s` and `pupl_s`.

Displaying the nose is not a big problem. One defines the nose as a matrix, displayed as three different points to make a triangle: `[0,2;-1.5,-1;1.5,-1;0,2]`. The order is important. The points are stretched or compressed from the center of the face by the factor `(1+0.1*p9)`. I used two commands to create the nose:

```
nose:=(1+0.1*p9)*[0,2;-1.5,-1;1.5,-1;0,2]
nose:=vector([k_ sub 1+m,k_ sub 2 +n],k_,nose)
```

Especially interesting and profitable for understanding parameters is considering the eyebrows and the mouth. In my version the middle of the eyebrows (`br, bl`) are associated with the distance of the eyes and their size (`p3 , p4`). In order to control the length of the eyebrows one has to normalize their direction vector. The slope of the eyebrow will be determined by `tan(p7*π/24)` for the left brow and `tan(p8*π/24)` for the right one respectively – since each brow is "unique". The extent of the brow will again be drawn from a specific parameter's value - `p6` - using an appropriate parameter representation.

Thus far we have used line segments, ellipses, and matrices. We are still missing quadratic functions. Lower and upper lip give us an opportunity to introduce parabolas. Coming out from the two corners of the mouth (±(2.5+0.1*p10),-3) `p11/p12` fixes the mouth shape as two parabolas – one for the bottom, the other for the top. The space between the lips is determined by the two parameters `p11` and `p12`. The parabolas will be drawn only in the area between the two corners of the mouth, `-mptr` and `+mptr`. Someone who wishes to take a little longer drawing can, naturally, also shade the mouth.



$$f(x) := a \cdot x^2 + b$$

$$[mptr :=, lp :=]$$

$$[f(0) = -3.5 + 0.35 \cdot lp,\ f(mptr) = -3.5]$$

$$SOLVE([f(0) = -3.5 + 0.35 \cdot lp,\ f(mptr) = -3.5],\ [a,\ b])$$

$$\left[a = -\frac{0.35 \cdot lp}{mptr^2} \wedge b = 0.35 \cdot (lp - 10)\right]$$

$$FIT\left(\left[x,\ a \cdot x^2 + b\right],\ \begin{bmatrix} 0 & -3.5 + 0.35 \cdot lp \\ mptr & -3.5 \end{bmatrix}\right)$$

$$-\frac{0.35 \cdot lp \cdot x^2}{mptr^2} + 0.35 \cdot lp - 3.5$$

Find a parabola by given points:
Either solve the simultaneous equations by hands or by using *DERIVE* or use the quadratic regression.
(`lp` stands for `p11` and `p12`.)

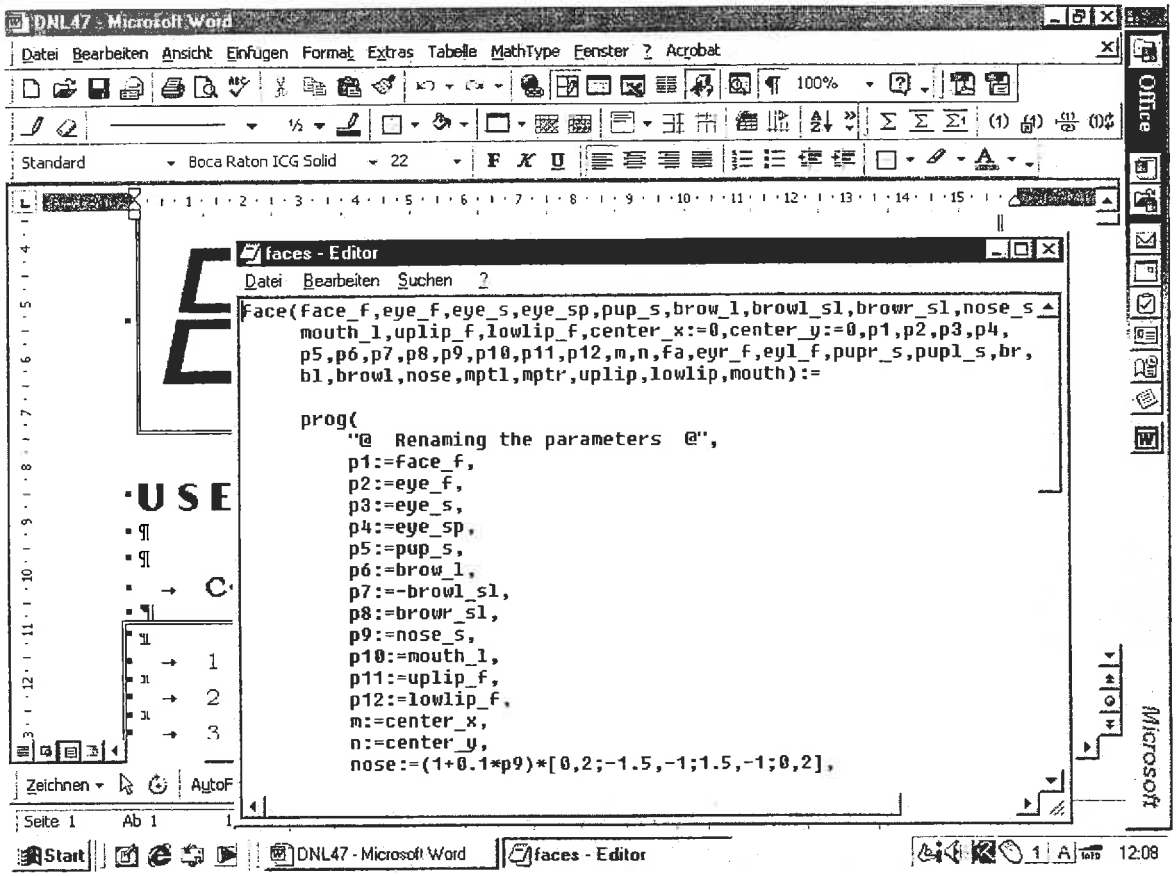
Implement `uplip` and `lowlip` considering again `m` and `n`. For shading the mouth you can use the inequality – see the listing.

```
face(face_f,eye_f,eye_s,eye_sp,pup_s,brow_l,browl_sl,browr_sl,nose_s,
     mouth_l,uplip_f,lowlip_f,center_x:=0,center_y:=0,p1,p2,p3,p4,
     p5,p6,p7,p8,p9,p10,p11,p12,m,n,fa,eyr_f,eyl_f,pupr_s,pupl_s,br,browr,
     bl,browl,nose,mptl,mptr,uplip,lowlip,mouth):=prog(
"@ Abbreviations for the parameters @",
     p1:=face_f,
     p2:=eye_f,
     p3:=eye_s,
     p4:=eye_sp,
     p5:=pup_s,
     p6:=brow_l,
     p7:=browl_sl,
     p8:=-browr_sl,
     p9:=nose_s,
     p10:=mouth_l,
     p11:=uplip_f,
     p12:=lowlip_f,
     m:=center_x,
     n:=center_y,
     fa:=(x-m)^2/(6+0.25p1)^2+(y-n)^2/36=1,
     nose:=(1+0.1*p9)*[0,2;-1.5,-1;1.5,-1;0,2],
     eyr_f:=(x-m-2-0.1p4)^2/(1.2+0.15p3+0.1p2)^2+
            (y-n-3)^2/(1.2+0.15p3)^2=1,
     eyl_f:=(x-m+2+0.1p4)^2/(1.2+0.15p3+0.1p2)^2+
            (y-n-3)^2/(1.2+0.15p3)^2=1,
     pupr_s:=(x-m-2-0.1p4)^2+(y-n-3)^2<=(0.5+0.05p5)^2,
     pupl_s:=(x-m+2+0.1p4)^2+(y-n-3)^2<=(0.5+0.05p5)^2,
     br:=[m+2+0.1p4,n+4.8+0.1p3],
     browr:=append([br+(1+0.1p6)*[1,tan(p7*PI/24)]/
           abs([1,tan(p7*PI/24)])],[br-(1+0.1p6)*[1,tan(p7*PI/24)]/
           abs([1,tan(p7*PI/24)])]),
     bl:=[m-2-0.1p4,n+4.8+0.1p3],
     browl:=append([bl+(1+0.1p6)*[1,tan(p8*PI/24)]/
         abs([1,tan(p8*PI/24)])],[bl-(1+0.1p6)*[1,tan(p8*PI/24)]/
         abs([1,tan(p8*PI/24)])]),
     nose:=vector([k_ sub 1+m,k_ sub 2 +n],k_,nose),
     mptl:=-2.5-.1*p10,
     mptr:=+2.5+.1*p10,
     uplip:=n-0.35*p11/mptl^2*(x-m)^2-3.5+0.35*p11,
     lowlip:=n+0.35*p12/mptl^2*(x-m)^2-3.5-0.35*p12,
     uplip:=if(mptl+m<=x<=mptr+m,uplip,?),
     lowlip:=if(mptl+m <=x<mptr+m,lowlip,?),
     "@ mouth:=if (p11>=p12,lowlip<=y<=uplip,uplip<=y<=lowlip) @",
     "@ mouth:= mptl+m<=x<=mptr+m and mouth @",
     "@ return[fa,eyr_f,eyl_f,pupr_s,pupl_s,browr,browl,nose,mouth] @",
      return[fa,eyr_f,eyl_f,pupr_s,pupl_s,browr,browl,nose,lowlip,uplip]
                  )
```

And now we should bring this huge program into the machine – including all parentheses, brackets, commas, ... – using our poor one-line-editor. I explained the process in DNL#46, but I'll do once more – accomplishing it with a tool provided by Albert Rich

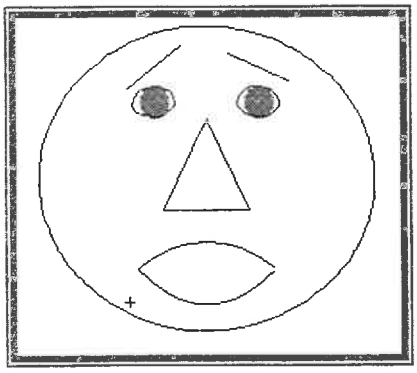
Not at all: Do the easiest thing of the world, open your Text-Editor – the poorest one you have, take WordPad, take any Editor. I don't take a sophisticated Textprocessor, because it might happen that there are some formattings included which are not welcome now.

So, once again: Take your Editor and type in the program. Do what you like to have a structured source code, use line brakes, use spaces, use margins. Be careful to not forget the parentheses and commas.



When you are ready, then simply copy the complete text to the clipboard Ctrl + C and paste it into the *DERIVE* Edit line Ctrl +V, press ↵.... and if you haven't made any syntactical error, then you should admire the whole program block in the *DERIVE* Algebra Window. One remark at this position: In this way you can edit and transfer all expressions (functions, programs, ...), but only one by one. You cannot convert a sequence of *DERIVE* expressions this way. How to perform this, see later in this paper.

I prefer to include comments in my source programs. They are under quotes and marked by "@". Do as you like, please. Now it is time to test our program. The "Standard Citizen" is given by:
`face(0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0)` – see on page 23.
`face(4, 3, -4, 2, 3, 4, -3, 5, 2, 4, 3, 4)` gives a specific one.



If you would like to place the face anywhere on the screen, then you have to add two more parameters for the center coordinates.

This goes in the rigth direction, but we would like to leave things to chance. Not only do we want to fix the parameters, but also let them be chosen randomly. At first we automate what we have done thus far and choose integer parameters from –5 to +5.

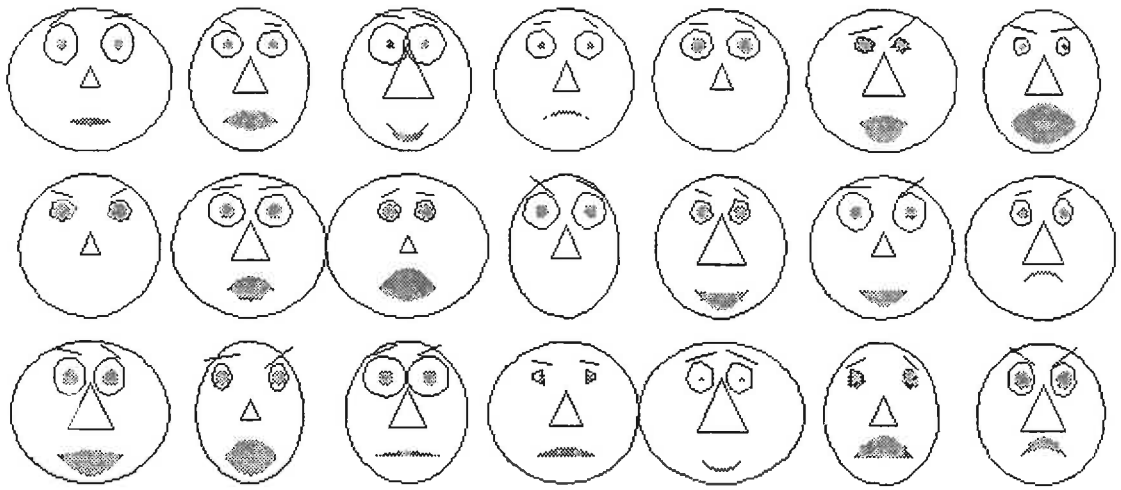
You should now know how to proceed: Go back to the editor and write your program:

```
faceint(x0 := 0, y0 := 0, v_) :=
   PROG(
        v_ := VECTOR(RANDOM(11) - 5, k, 12),
        face(v_ sub 1, v_ sub 2, v_ sub 3, v_ sub 4, v_ sub 5, v_ sub 6,
             v_ sub 7, v_ sub 8, v_ sub 9, v_ sub 10, v_ sub 11, v_ sub 12,
             x0, y0)
       )
```

Copy, paste and test, producing a whole company:

First simplify `random(0)` to initialize the random number generator (or you might include it into `face()`, adding the variable `dummy_` into the parameterlist and inserting one line: `dummy_ := random(0)`).

```
VECTOR(VECTOR(faceint(k_, l_), k_, 42, -42, -14), l_, 14, -14, -14)
```



Now our – hopefully enthusiastic - audience might raise the question – if they don't then the teacher could - with this way of doing things is it possible to assign an individual face to every person in the world? How many different faces could be designed in this way?

Then we let go the – unnecessary – demand for integer parameter values and allow all random values between –6 and +6.
There is only to exchange one line:

```
v_ := VECTOR(12*RANDOM(1) - 6, k, 12)
```

and name the program `facernd()`. Test the program.

This is all very nice, but when nature deals the cards, the characteristics are not equally distributed within an interval, but they are normally distributed. And here we have a great chance to introduce a demonstration and a visualization of the Mean and Variance. To simplify matters we give the same mean and standard deviation to all parameters, and with these values generate faces of a certain type, which might vary more or less.

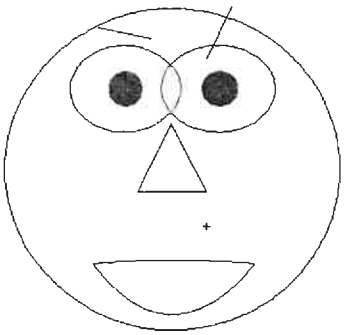
Again only exchanging one line is necessary to have normally distributed numbers – with given mean and standard deviation – in the vector v_. Take

```
v_ := VECTOR(RANDOM_NORMAL(μ,σ),k,12)
```

or

```
v_ := VECTOR(RHS(SOLVE(NORMAL(z,μ,σ) = random(1),z)),k,12)
```

Save as `facenorm()` and let's test immediatlely:

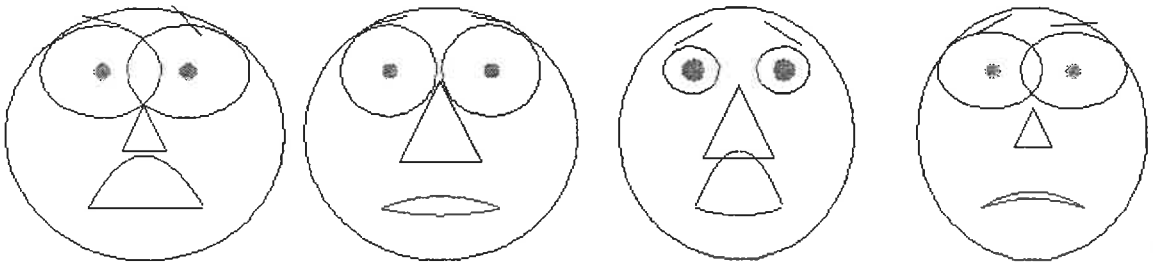


```
facenorm(3, 3)
```

When statistics enters the game, then one single event is not sufficient at all. I will create a whole "Ancestor's Gallery" of my forebearers on the screen.

So I can perhaps reproduce my Forefathers on my father's side of family – they all were characterized by broader faces and they lived in very stable circumstances such that their appearance didn't vary very strong ($\mu = 3, \sigma = 1$)

```
VECTOR(facenorm2(3, 1, j, 0), j, -21, 21, 14)
```



Greatgrandfather Josef

Greatgrandgrandfather Joseph

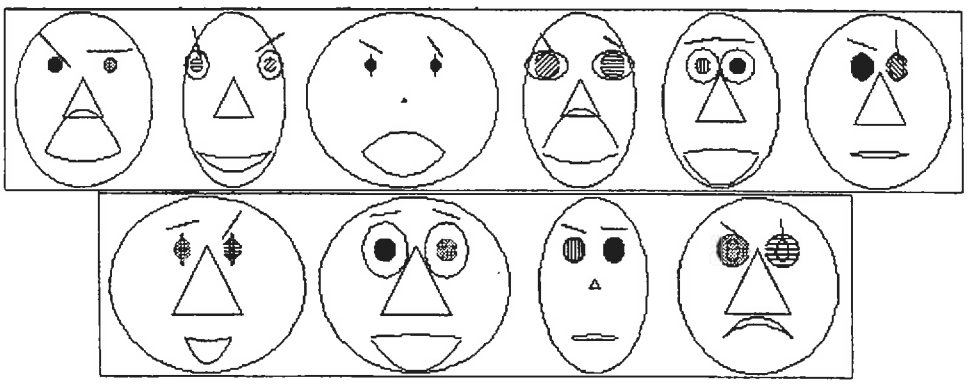
Greatgrandgradfather Pepi (Viennese form of Josef)

Greatgranduncle Rudolf

When we do the same investigation on my mother's side of the family we find a much greater variety in their physical characteristics, caused by their environment and history: ($\mu = -2, \sigma = 10$)

Here I'd like to present some pictures produced by the TI.

`ancestors(-2,3,10)` creates the faces and `showfam(10)` gives the presentation on the TI-92 Screen. The respective TI-92 programs can be downloaded from the website.



I believe that it is now time to explain how to convert more than one program/function from the text-editor to *DERIVE*. You can find one way – working with the tilda "~" to announce a line break in DNL#46 and I don't want to repeat this procedure, too.

On Sunday 4 August I received an email from Albert Rich:

Hello Josef,
Beginning on page 28 of DUG Newsletter #46 you suggested a way to use an external text editor to write Derive programs. In response to your remarks and Terence's email below, I wrote the utility program TxtToMth.exe that should make the process easier. Attached is a zip file containing the file. I look forward to your comments.

This was Terence Etchells' previous mail to Albert:

I've started using Notepad to write my programmes, it's really neat once you get a method going. In order that I match parenthesis correctly I've adopted the approach below of closing the parenthesis of a function vertically below the start of the function and indenting using a space. It appears that the tilda ~ is not necessary for the end of lines, just use enter and every thing copies into the Derive entry line just fine.

So the reason for this email:

From now (or until Derive 6) I will write all my functions in notepad and save these as txt files. Also I will of course enter the functions in Derive and save as mth or dfw, but the writing and editing will be in notepad (or any good text editor). Now it strikes me that in the text version of the functions I can easily put remarks in to explain the meaning of variables and what each line does. However, I don't want these in the Derive functions in the algebra window. Now I notice that when text is pasted into the entry line tildas are ignored! Is there a way to use a special symbol (say $,%,£ or &) so that the text in between these symbols is ignored when pasted in to the entry line. So for the example above, the text version would look like

```
SIN?(f, str) :=
PROG(
 $ convert f into a string $
 str := STRING(f),
 $ is the top level function a SIN? $
 IF(str SUB 1 = "SIN",
  $ if yes return true and the arguments in a vector"
  RETURN [true, str SUB 2]
 ),
 $ else return false $
 [false]
)
```

When this is pasted into the entry line (and subsequently entered into the algebra window) all the remarks do not appear! This will have the same effect of stripping out remarks when compiling code in C++ etc.

What do ya think?

And here is Albert's solution:

As you discovered, there is no good way to insert comments into function definitions in Derive 5. We hope to resolve this for Derive 6.

Until then I have written a little utility program that allows you to write large Derive programs as txt files using Notepad. These txt files can include comments within and between function definitions. Comments are those lines of text whose first nonblank character is a semicolon. Function definitions and other expressions in the txt file must be separated by at least one blank line.

The utility program TxtToMth.exe converts such txt files into mth files by removing the embedded comments and adding the tilde character at the ends of lines. TxtToMth.exe preserves comments occurring between function definitions. For example, given the commented txt file

```
; This is a test file for the txt to mth file conversion program.
SIN?(f, str) :=
  PROG(
; convert f into a string
    str := STRING(f),
; is the top level function a SIN?
    IF(str SUB 1 = "SIN",
; if yes return true and the arguments in a vector
       RETURN [true, str SUB 2]
    ),
; else return false
    [false]
  )

; Note that expressions in txt file must be separated by a blank line!!!
DOG(x):=SIN(x)^2
```

Save this file eg. as `sintest.txt` and **`TxtToMth.exe`** *will generate the mth file* `SINTEST.MTH`. *You can make a shortcut to* `TxtToMth` *on your desktop if you want.*

So let us try doing as Albert suggested: We write the two forms for designing normal distributed faces in one texfile and save it as `facemth.txt`. If you – in contrary to Terence - want to see the comments in the *DERIVE* appearance of the program you have to include the "" commentary lines.

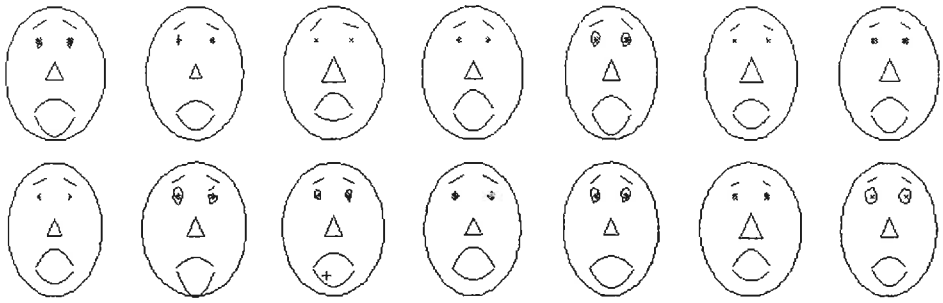
```
; first variant of normal distributed characteristics
facenorm(mu, sigma, x0:=0, y0:=0, v_):=
              PROG(
        ; create the mu,sigma-normal distributed numbers via NORMAL
              v_:=VECTOR(RHS(SOLVE(NORMAL(z, mu, sigma) = RANDOM(1), z)),
                                 k, 12),
        "@ Call program face() @",
        ; Call program face()
        ; Must be preloaded
              face(v_ SUB 1, v_ SUB 2, v_ SUB 3, v_ SUB 4, v_ SUB 5,
                   v_ SUB 6, v_ SUB 7, v_ SUB 8, v_ SUB 9, v_ SUB 10,
                   v_ SUB 11, v_ SUB 12, x0, y0)
                   )

; second way using implemented RANDOM_NORMAL
facenorm2(mu, sigma, x0 := 0, y0 := 0, v_):=
                   PROG(
              ; the random vector
                     v_ := VECTOR(RANDOM_NORMAL(mu, sigma), k, 12),
                     face(v_ SUB 1, v_ SUB 2, v_ SUB 3, v_ SUB 4,
                     v_ SUB 5, v_ SUB 6,v_ SUB 7, v_ SUB 8, v_ SUB 9,
                     v_ SUB 10, v_ SUB 11, v_ SUB 12,x0, y0)
                      )
```

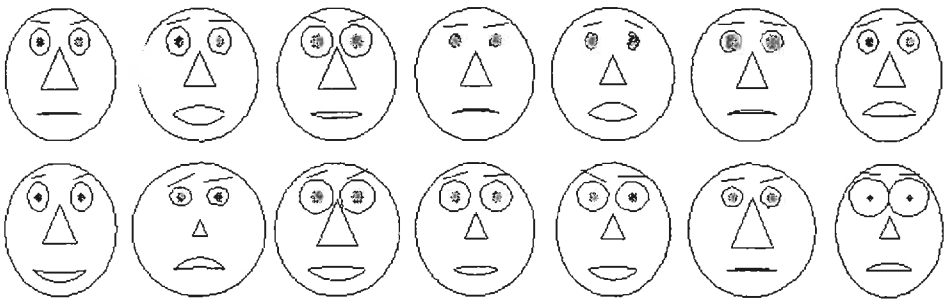
Then launch TxTToMth.exe – as you can see below – and you have a ready made MTH-file, which can be opened or loaded to *DERIVE*. In our case `face()` must have been be preloaded. Otherwise you would receive an error message.

It is important to use Uppercase notation for *DERIVE*-functions.

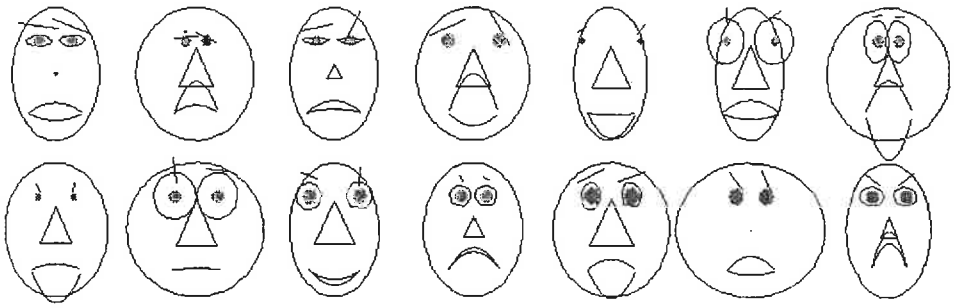
A family of Longheads ($\mu = -4$, $\sigma = 1$)



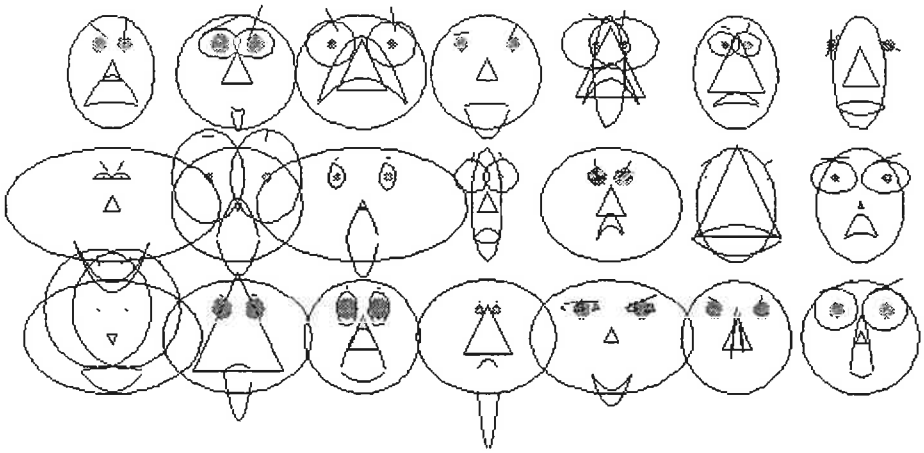
Average faces with slight variations: ($\mu = 0$, $\sigma = 2$)



Mean is again 0 (= average), but $\sigma = 5$ should cause significant differences:



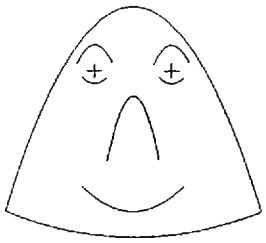
What´s about a standard deviation of 10?:



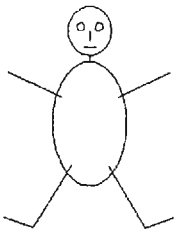
HOW MANY STRANGE AND GROTESQUE FACES ARE THERE IN THE WORLD?

I will close by presenting a couple of other proposals for motivating figures in order to support the students´ feelings for applying parameters.

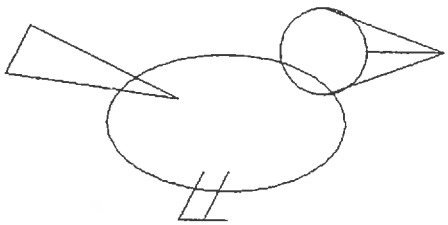
All what to do is, to have an idea for the basic figure and then select those characteristics which can usefully be varied. But from my experience I can recommend to rely on the phantasy and creativity of your students. You surely will be surprised.



Parabolicus



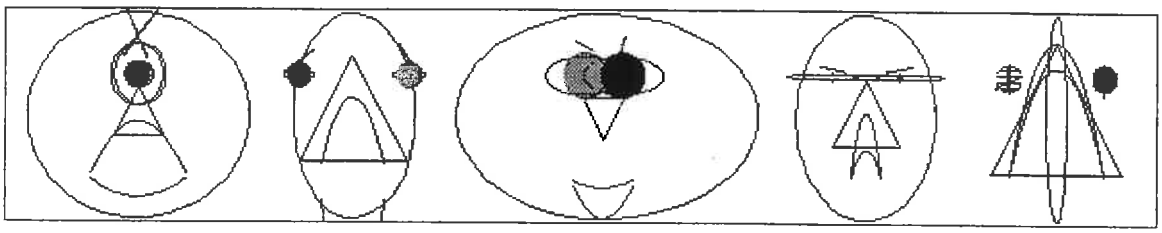
Dot,Dot-Dash,Dash



Wings for the Bird

As a summary I´d like to add that I am really enthusiastic by the connection of creativity, the use of parameters and the visualization of basic statistics concepts.

See finally 5 "Faces" designed by the TI-program.



Uncle Cyclopios, Auntie Duck, Willi (why is his nose on the top?), my maths teacher KnowsAllBetter and finally my wealthy aunt Agathe.

I´d like to express my gratefulness to Carl Leinbach, who translated the German original using his German-English-dictionary. He did a great job and once more proved his enthusiasm for using technology in teaching mathematics and also the fact that maths language is international. The next challenge for him should be a Japanese or Chinese text.

References

[1] Clifford A. Pickover, *Computers, Patterns, Chaos and Beauty*, Dover Publications, 2001

[2] J.Böhm, *Wie ich die Parameter lieben lernte*, TI-Nachrichten 2/02 p2 – p5

DEs in Austrian Secondary Schools

Günter Schödl, Wiener Neustadt, Austria

Günter presents on his website `http://www.bg-bab.ac.at/~mathe/index.htm` *a huge collection of Secondary School problems, tasks and assessments. He prefers working with DERIVE. I selected some of his differential equation examples and translated them. They are elementary but they force the students modelling, showing some basic competencies in operational skills, finding and applying appropriate CAS-tools. I take the chance to direct your attention on two DERIVE utility files* `ODE1.MTH` *and* `ODE_APPR.MTH` *– by the way it is strongly recommmend to browse this enormous ressource – and on solving the problems on the TI-89/92+/Voyage supported by the DE-Mode, Josef*

Problem 1:

If a projectile is shot in a sand dam one can assume that its deceleration is equal to the square root of its entrance velocity. How long will a projectile move having an entrance velocity of 144dm/sec?
Set up the model and solve the DE.

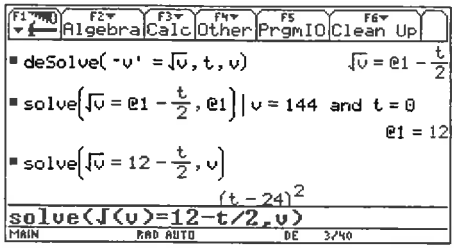
$$-\frac{dv}{dt} = \sqrt{v} \Rightarrow \frac{dv}{\sqrt{v}} - dt \Rightarrow 2\sqrt{v} = -t + c$$

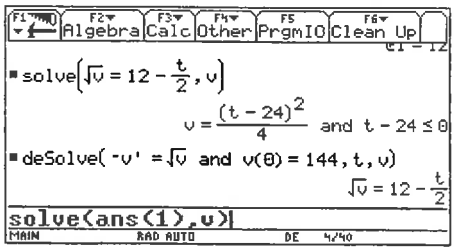
$$-v' = \sqrt{v}$$

$$\text{in form } p(x,y) + q(x,y)\,y' = 0 : \sqrt{v} + 1 \cdot v' = 0$$

Don´t forget to load first the Utility file ODE1.MTH (recommended in the background).

Equations of form p(x,y) + q(x,y) y' = 0 can be solved applying `DSOLVE1_GEN(p(x,y),q(x,y),x,y,c)`





`DSOLVE1_GEN(√v, 1, t, v, c)`

$t + 2\cdot\sqrt{v} = c$

$0 + 2\cdot\sqrt{144} = c$

`SOLVE(0 + 2·√144 = c, c) = (c = 24)`

$t + 2\cdot\sqrt{v} = 24$

`SOLVE(t + 2·√v = 24, v)`

$$v = \text{IF}\left(t \le 24, \frac{(t-24)^2}{4}\right)$$

$$v(t) := \frac{(t-24)^2}{4}$$

`SOLVE(v(t) = 0, t) = (t = 24)`

Obtain immediately the special solution for v(0) = 144 applying DSOLVE1:
First redefine v!

`v :=`

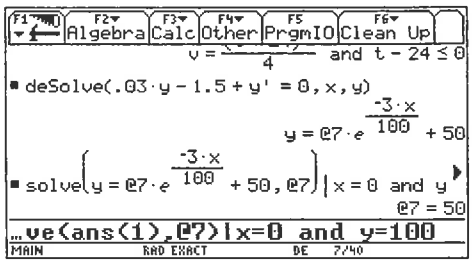
$$\text{SOLVE}(\text{DSOLVE1}(\sqrt{v}, 1, t, v, 0, 144), v) = \left(v = \text{IF}\left(t \le 24, \frac{(t-24)^2}{4}\right)\right)$$

Problem 2:
A tank contains 400 litres of saline solution containing 100 kg salt. 12 litres/min of another solution with a concentration of 1/8 kg salt/litre are flowing into the tank. By stirring the mixture is kept uniform and it is flowing out of the tank with 12 litres/min.

What is the amount of salt in the tank after 90 minutes?

The model: Let y(x) the amount of salt after x minutes.
Per minute 1.5 kg of salt are flowing in and 12/400 y = 0.03 y kg are flowing out.

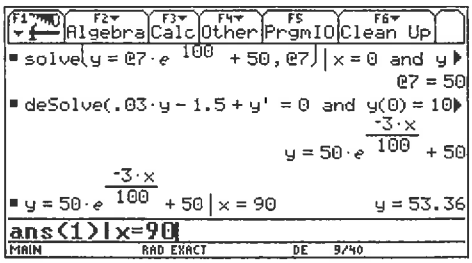
$$\frac{dy}{dx} = 1.5 - 0.03y \Rightarrow y' = 1.5 - 0.03y \; ; \; y(0) = 100$$
$$0.03y - 1.5 + 1 \cdot y' = 0$$



solving by hands:

$$\frac{dy}{1.5 - 0.03y} = dx \Rightarrow \frac{\ln(1.5 - 0.03y)}{0.03} = -x + c \ldots..$$

$$p(x,y) = 0.03y - 1.5; \; q(x,y) = 1$$



The *DERIVE* solution:

```
DSOLVE1_GEN(0.03·y - 1.5, 1, x, y, c)

100·LN(y - 50) + 3·x = 100·c
```

which is equal to the expression yielded by the above calculation substituting constant c by another constant c + ln(100/3).

```
                                                    c - 3·x/100
SOLVE(100·LN(y - 50) + 3·x = 100·c, y) = (y = ê             + 50)

                  c - 3·x/100
SOLVE(SUBST(y = ê            + 50, [x, y], [0, 100]), c, Real)

c = LN(50)

          LN(50) - 3·x/100              - 3·x/100
(f(x) := ê                + 50) = 50·ê          + 50

f(90) = 53.36027563
```

If you are only interested to let the students set up the model and then work with it you can omit the general solution and find immediately the specific solution. So step by step you can move from the White Box (doing all calculations by hands) to the Black Box (leaving all calculations to the CAS).

```
SOLVE(DSOLVE1(0.03·y - 1.5, 1, x, y, 0, 100), y)

        - 3·x/100
y = 50·ê          + 50
```

Problem 3:
The current in an electric circuit i changes in dependency of time t according di/dt = 5 - 2i.
Find the endbehaviour of the current for increasing time.

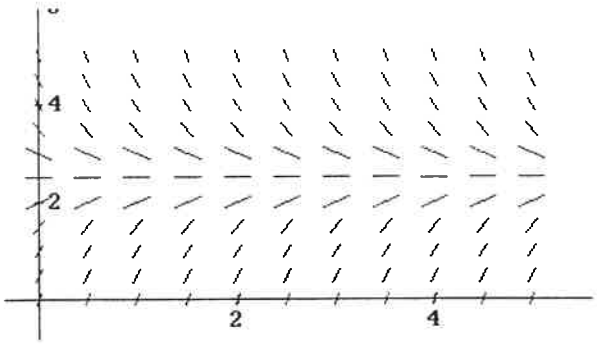
Find the general solution (without CAS) and test your solution by superimposing it on the *DERIVE*-created direction field.

Then check your solution(s) applying an appropriate CAS-method.

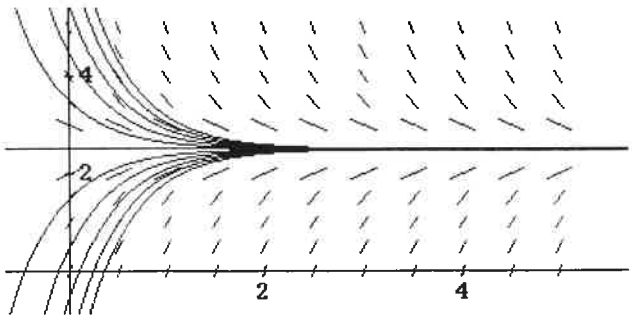
$$\frac{di}{dt}=5-2i \Rightarrow \frac{di}{5-2i}=dt \Rightarrow -\frac{1}{2}\ln(5-2i)=t+c \Rightarrow$$

$$\Rightarrow \ln(5-2i)\,0-2t+c \Rightarrow 5-2i=ce^{-2t} \Rightarrow i=\frac{5}{2}-ce^{-2t}$$

```
DIRECTION_FIELD(5 - 2·i, t, 0, 5, 10, i, 0, 5, 10)
```




```
VECTOR(5/2 - c·ê^(- 2·t), c, -5, 5)
```




```
SOLVE(DSOLVE1_GEN(5 - 2·i, -1, t, i, c), i)
```

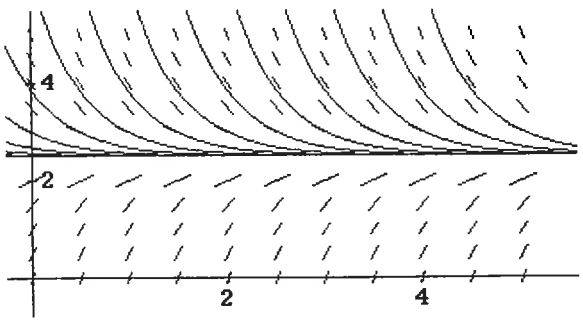
$$i = \frac{\hat{e}^{-2\cdot c-2\cdot t}}{2}+\frac{5}{2}$$

Is this solution offered by the CAS equivalent to the pencil&paper solution ? Let´s use again the CAS:

```
SOLVE(ê^(- 2·c - 2·t)/2 + 5/2 = 5/2 - x·ê^(- 2·t), x)
```

$$x = -\frac{\hat{e}^{-2\cdot c}}{2}$$

As one constant can be replaced by another one, the two solutions are equivalent.

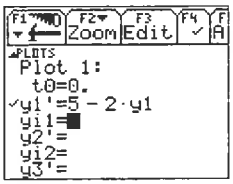


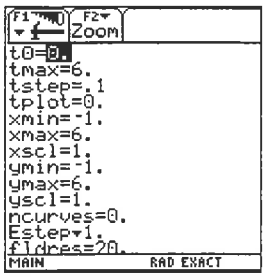
What is the endbehaviour?

The current tends to 2.5.

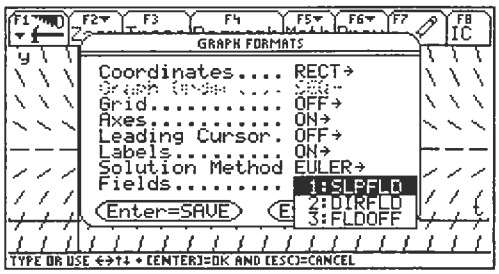
What makes the difference to have curves only above y = 2.5 or above and below?

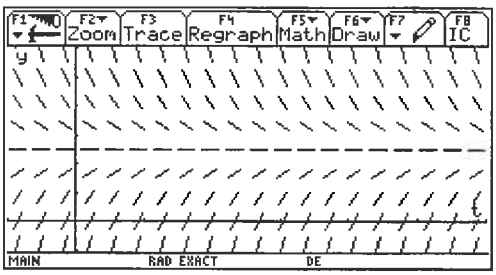
Here are some screen shots to illustrate how to perform the task on the TIs.



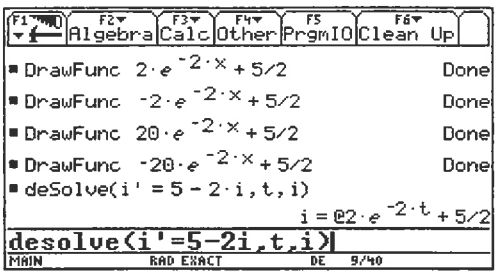


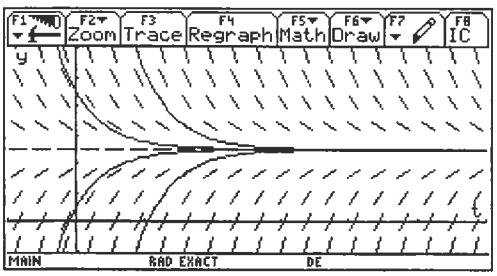
Define the DE and set the Windows Values.
The direction field for a DE is SLPFLD and not DIRFLD (that's for a system of DEs!).



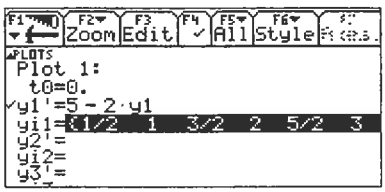


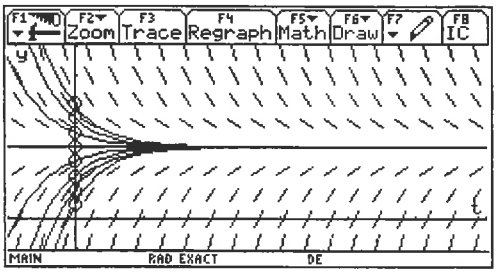
How to plot one or more specific solutions? You can't enter the function in the [Y=]-Editor and you cannot use the Graph-command in the Home Screen because of the DE-Mode. So use the DrawFunc – command to add one or more specific solutions





Or you add one initial condition (or a list of them) to obtain a family of integral curves.





Problem 4:
Given is the DE $dy = x\,y^2\,dx$.

a) Find the exact solution passing P(0,1) without CAS
b) Verify your solution using your CAS-tool.
c) Plot the direction field together with the specific solution. Add two more solutions.
d) Find the function value of y(x=0.3)
   d1) exact
   d2) graphically
   d3 using a numerical method
e) What is the error percentage of the numerical solution compared with the exact one?

You need to preload ODE_APPR.MTH, too!

a)

$$\frac{dy}{dx} = x\,y^2 \Rightarrow \frac{dy}{y^2} = x\,dx \Rightarrow -\frac{1}{y} = \frac{x^2}{2} + c \Rightarrow y = -\frac{2}{x^2 + 2c} \Rightarrow y = -\frac{2}{x^2 + c}$$

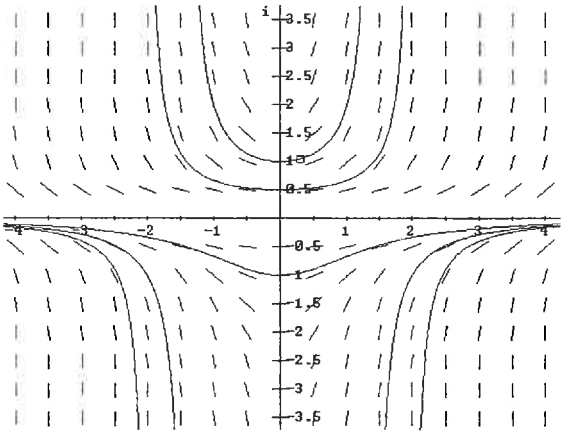
$$y(0) = 1 \Rightarrow 1 = -\frac{2}{c} \Rightarrow c = -2 \Rightarrow y = -\frac{2}{x^2 - 2};\ y\left(\frac{3}{10}\right) = 200/191 \sim 1.047$$

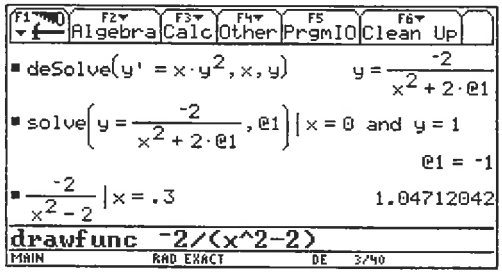
b)

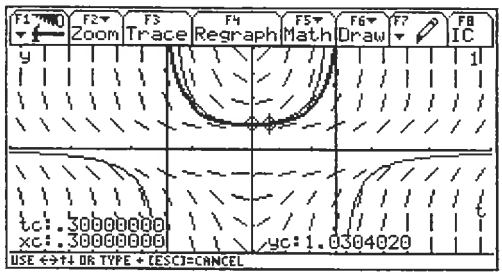
SOLVE(DSOLVE1(- x·y^2, 1, x, y, 0, 1), y) = $\left[y = \frac{2}{2 - x^2}\right]$

$$\frac{2}{2 - 0.3^2} = \frac{200}{191} \approx 1.047120$$

c)







You can compare the numerical solution (thick - using EULER´s Method) with the exact solution.
Apply RK-Method in order to improve the numerical answer on the TI. Question for or of the students:Why do we see only one part thickly plotted?

DIRECTION_FIELD(x·y^2, x, -4, 4, 16, y, -4, 6, 20)

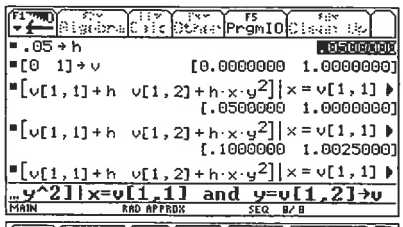
[hCross, vCross] = [0.3, 1.047120418]

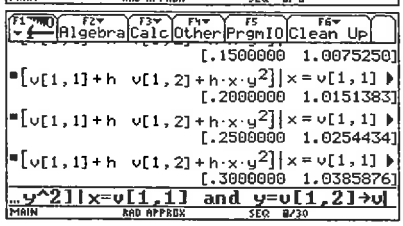
d) e)

EULER_ODE(x·y^2, x, y, 0, 1, 0.05, 6)

0	1
0.05	1
0.1	1.0025
0.15	1.007525031
0.2	1.015138331
0.25	1.025443389
0.3	1.038587566

See on the right the TI-process on the Home Screen and then – very nice – as a spreadsheet working with CellSheet.
Here one could easily change the input data (h = 0.01, ...).



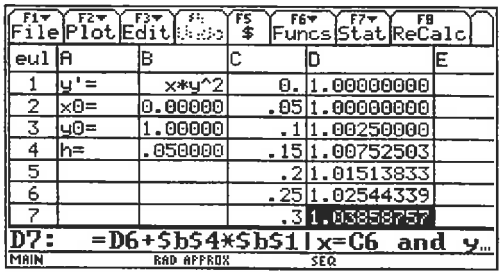


EULER_ODE(x·y^2, x, y, 0, 1, 0.02, 15)

[0.3, 1.043661051]

$$\left[\frac{\frac{200}{191} - 1.038587566}{\frac{200}{191}} \cdot 100,\ \frac{\frac{200}{191} - 1.043661051}{\frac{200}{191}} \cdot 100\right]$$

[0.8148874470, 0.3303696295]



Amusing Computerpuzzles for DERIVE-Colleagues

Ziffernsummen und Zyklen – Sums of Digits and Cycles

Rüdeger Baumann, Celle, Germany

Rüdiger prefers writing in German and he several times asked for contributions in the submitter´s mother tongue. I can understand his point of view, but on the other hand we fortunately have so many members from all over the world, who have all English as common means for understanding each other. So I try to translate Rüderger´s "Knobeleien" into English – our "lingua franca" (including the DERIVE-variables´ names). You can download also his original German version. I hope that Rüdiger will understand the Editor´s position. Now I wish much fun with his ACDC. Josef

In DNL 44 (page 30) Josef reports on a trip to Colombia and presents his souvenirs in form of three riddles, which were given by Milton Lesmes Acosta. These riddles are not only amusing but also very valueable, because we can use them to practise programming with *DERIVE*. In this sense I´like to following I´d like to contribute refering to Milton´s problems as follows.

Problem 1 (Milton´s riddle #3): Choose any natural number *n* and form its sum of digits of *order 2*, i.e. the sum of the squares of the digits. What happens repeating this process?

This problem - tackled by A. Porges (1945) – can be generalized for exponents > 2 as follows:

```
    QS(n, k) :=
      If n ≤ 0
#1:      0
         MOD(n, 10)^k + QS(FLOOR(n, 10), k)

    Milton(number, exponent, length, i := 0, list := []) :=
      Loop
        i :+ 1
#2:     list := ADJOIN(number, list)
        If i ≥ length
           RETURN REVERSE(list)
        number := QS(number, exponent)
```

```
#3: Milton(153,2,15) =
      [153,35,34,25,29,85,89,145,42,20,4,16,37,58,89]

#4: Milton(153,3,2) = [153,153]

#5: Milton(153,4,20) =
      [153,707,4802,4368,5729,9603,7938,13139,6725,4338,4514,
       1138,4179,9219,13139,6725,4338,4514,1138,4179]

#6: Milton(136,2,14) =
      [136,46,52,29,85,89,145,42,20,4,16,37,58,89]

#6: Milton(136, 3, 3) = [136,244,136]

#7: Milton(136,4,32) =
      [136,1378,6579,10883,8274,6769,11554,1508,4722,2689,11969,
       14420,529,7202,2433,434,593,7267,6114,1554,1507,3027,
       2498,10929,13139,725,4338,4514,1138,4179,9219,13139]
```

Periods (Cycles) of various lengths are appearing. It would be great if the program could recognize the appearance of such a period.

Challenge for the DNL-Readers: Implement such a "Period-Recognizing-Routine". (Possible output: Position of first element of the period together with the period).

Problem 2 (Milton's riddle #2): Program a function, which relates a number to its reverse,
eg f(123) = 321.

The following is a recursive solution:

```
    mirror(number, mirror_numb := 0) :=
      If number ≤ 0
#9:      mirror_numb
         mirror(FLOOR(number, 10), 10·mirror_numb + MOD(number, 10))
```

This function is useful for solving the so called *Palindrom-Problem*. A *Palindrom* is a word which coincides with its reverse:

```
     Palindrom?(Sentence) :=
       If Sentence = REVERSE(Sentence)
#10:      "Yes"
          "No"

#11: Sentence1 := EINNEGERMITGAZELLEZAGTIMREGENNIE        (1)

#12: Sentence2 := docnoteidissentafastneverpreventsafatnessidietoncod

#13: [Palindrom?(Sentence1), Palindrom?(Sentence2)] = [Yes, Yes]

     PalindromProcess(number, length := 30, i := 0, list := []) :=
       Loop
         i :+ 1
         list := ADJOIN(number, list)
#14:     mirror_numb := mirror(number)
         If number = mirror_numb ∨ i ≥ length
            RETURN REVERSE(list)
         number := number + mirror_numb

#15: PalindromProcess(18) = [18, 99]

#16: PalindromProcess(19) = [19, 110, 121]

#17: PalindromProcess(6999) = [6999, 16995, 76956, 142923, 472164, 933438,
1767777, 9545448, 17990907, 88900878, 176701866, 844809537, 1580717985, 747
7888836, 13866776583, 52434543414, 93869086839, 187737183678, 1064118921459
, 10605417036060, 16668488486661]

#18: PalindromProcess(196) = [196, 887, 1675, 7436, 13783, 52514, 94039, 18
7088, 1067869, 10755470, 18211171, 35322452, 60744805, 111589511, 227574622
, 454050344, 897100798, 1794102596, 8746117567, 16403234045, 70446464506, 1
30992928913, 450822227944, 900544455998, 1800098901007, 8801197801088, 1760
2285712176, 84724043932847, 159547977975595, 755127757721546]
```

The strings in `Sentence1` and `Sentence2` must be entered unter quotes "", which are – unfortunately – not displayed in the Algebra Window.

Titbits from Algebra and Number Theory (23)

by Johann Wiesenbauer, Vienna

Quite a few things have happened since I wrote my last column here... In the first place, there was that great flood you may have seen on your TV, which affected Austria like many other countries in Europe. As for Austria there was the interesting side effect that the coalition of our government went down the drain as well. I would have liked to tell you more about this, but this is definitely off topic...

What might be more interesting for Derivers is the fact that the 5th International Derive conference also took place in Vienna this summer. I enjoyed it very much as it was again a great opportunity to meet good old friends from everywhere and also to make new ones. Of course, there were also many interesting talks and workshops and I look very much forward to the proceeding of this conference, which will appear on CD-ROM in October.

As for me I gave a workshop there dealing with primality testing and the factorization of large numbers with Derive (cf. [2]).. Unfortunately this was only 4 weeks before 3 Indian mathematicians (cf. [1]) discovered a deterministic primality test in polynomial time, which is called now AKS-test after its inventors and is certainly a theoretical breakthrough in this field. (By the way, Josef told me that when reading this result in the newspapers – yes, there were headlines e.g. in the "New York Times", which doesn't exactly occur very often, when it comes to mathematics – he had hoped that I was involved. Boy, he seems to have a very high opinion of me, indeed!)

Now, what's so special about this new primality test? In order to find this out it might be a good idea to compare it with the existing ones and comment on their various deficiencies. Probably the most widely used primality test is the so-called Rabin-Miller test. In fact, I have already mentioned this test in this series several times (see e.g. my Titbits(16) in DNL # 37, p28-32, for the theoretical background), but let me take this opportunity to give in the following the most versatile implementation so far along with the remarkable example n=119506876879526579251836131572511635189824558 1. (Note that a may be a single base here, which is 2 by default, or any list of bases. If a is a negative number, then it is replaced by the list of all primes up to $|a|$.)

```
Rabin_Miller(n, a := 2, a_, s_, t_) :=
  Prog
    If n = 1
       RETURN false
    If EVEN?(n)
       RETURN SOLVE(n = 2)
    If NUMBER?(a)
       If a > 0
          a := [a]
          a := SELECT(PRIME(q_), q_, -a)
    t_ := n - 1
    Loop
      t_ :/ 2
      If ODD?(t_) exit
    Loop
      If a = [] exit
      s_ := t_
      a_ := - ABS(MODS(FIRST(a)^s_, n))
      Loop
        If a_ = -1
           [a := REST(a), exit]
        s_ :* 2
        If s_ = n - 1
           RETURN false
        a_ := MODS(a_^2, n)
```

```
Rabin_Miller(119506876879526579251836131572511635189824558 1, -31) = true
```

```
SELECT(¬ Rabin_Miller(119506876879526579251836131572511635189824558 1, a), a, 1, 37) = [22, 26, 34, 37]
```

Basically, if n is the number to be tested and a any integer in the range 0<a<n, then one has to compute some power a^t mod n for the test, where t has the same order of magnitude as n itself. In particular, its complexity is given by $O((\ln n)^3)$, which implies that a single test can be carried out in polynomial time. However, unlike the AKS-test it is not "deterministic", but only "probabilistic", i.e. if n passes the test, the statement "n is prime" can be made only with a small probabilty of error (at most ¼ here), whereas n is certainly composite, if it fails the test.

In practice, the Rabin-Miller test is often carried out for all bases in a given list of small primes. On the other hand, it can be shown that for any finite set S of positive integers there is always a composite number n, which passes the Rabin-Miller test for all bases $a \in S$. (In our example above we took S to be the set of all primes $p \leq 31$.) It must be said though that assuming the so-called ERH (=Extended Riemann Hypothesis) to be true, it can be shown that only the bases up to $2(\ln n)^2$ need to be tested, which results in a deterministic primality test of order $O((\ln n)^5)$. Although this is still a polynomial time test, we are heavily relying on a conjecture here, which is unproven though plausible. In contrast, the new AKS-test doesn't make use of any unproven assumptions.

There are two primality tests, which I would like be mentioned here, because they came very close a strictly deterministic primality test in polynomial time. The first one was found by L.Adleman, C.Pomerance and R.Rumely in 1983 and improved later on by H.Cohen and H.W.Lenstra. Nowadays it's called APRCL-Test after those inventors. Its running time is of order

$$(\ln n)^{c(\ln \ln \ln n)}$$

for some constant c>0. Since $\ln \ln \ln n$ increases very, very slowly (note that e.g. $\ln \ln \ln n \approx 2$ for $n \approx 10^{1000}$), this is almost a polynomial-time algorithm. The second test was found by Goldwasser and Kilian in 1986 and improved by many people thereafter. It uses facts from the theory of elliptic curves and has an expected running time of order $O((\ln n)^6)$. The attribute "expected" refers to the fact that certain parameters are needed for this test, which can be found very quickly in practice, but there is no guarantee for it. Even though, as of today this is the deterministic primality test which is used most often for numbers of no special form up to several thousands of digits!

Well, after making your mouth watery, it's high time I explained now how the new primality test works, at least in principle. Let's have a look at the first rows of Pascal's triangle for this.

0	[1]
1	[1, 1]
2	[1, 2, 1]
3	[1, 3, 3, 1]
4	[1, 4, 6, 4, 1]
5	[1, 5, 10, 10, 5, 1]
6	[1, 6, 15, 20, 15, 6, 1]
7	[1, 7, 21, 35, 35, 21, 7, 1]
8	[1, 8, 28, 56, 70, 56, 28, 8, 1]
9	[1, 9, 36, 84, 126, 126, 84, 36, 9, 1]
10	[1, 10, 45, 120, 210, 252, 210, 120, 45, 10, 1]
11	[1, 11, 55, 165, 330, 462, 462, 330, 165, 55, 11, 1]
12	[1, 12, 66, 220, 495, 792, 924, 792, 495, 220, 66, 12, 1]
13	[1, 13, 78, 286, 715, 1287, 1716, 1716, 1287, 715, 286, 78, 13, 1]
14	[1, 14, 91, 364, 1001, 2002, 3003, 3432, 3003, 2002, 1001, 364, 91, 14, 1]
15	[1, 15, 105, 455, 1365, 3003, 5005, 6435, 6435, 5005, 3003, 1365, 455, 105, 15, 1]
16	[1, 16, 120, 560, 1820, 4368, 8008, 11440, 12870, 11440, 8008, 4368, 1820, 560, 120, 16, 1]
17	[1, 17, 136, 680, 2380, 6188, 12376, 19448, 24310, 24310, 19448, 12376, 6188, 2380, 680, 136, 17, 1]
18	[1, 18, 153, 816, 3060, 8568, 18564, 31824, 43758, 48620, 43758, 31824, 18564, 8568, 3060, 816, 153, 18, 1]

If you check the rows carefully, you may notice that for the visible part of this triangle it is true that whenever n is a prime all numbers of the corresponding row apart from the first and last one are divisible by n and vice versa. Take for example the 7^{th} row 1,7,21,35,35,21,7,1 were all numbers except for the 1's are divisible by 7, indeed, whereas e.g. in the 9^{th} row 84 is not divisible by 9. Try to prove this result! (It's not very hard, if you make use of the basic definition of a binomial coefficient! By the way, here is also a small programming challenge for those of you who fear neither death nor devil when it comes to programming: What could be the quickest way to check the condition above for a given nonnegative integer n in Derive? I'll give the solution at the end of this paper.)

Due to the binomial theorem, we can state our primality test for n also in the following slightly more general form, which is actually needed for the AKS-test:

Theorem: Suppose that a is any integer coprime to n. Then n is prime if and only if

$$(x-a)^n \equiv x^n - a \mod n$$

This condition actually means that we have to expand the polynomial $(x-a)^n$, reduce its coefficients mod n and check whether the outcome coincides with $x^n + (-a \mod n)$. It should be rather obvious that this is not feasible even for an n of moderate size, say with 20 digits. But here comes a new idea into play. As the inventors of the AKS-test found out, it is sufficient to check the condition in the theorem mod $x^r - 1$ for some small integer r and for all integers a in the range $1,2,\ldots,\lfloor 2\sqrt{r}\ln n \rfloor$. The integer r is chosen to be the smallest odd prime that fulfills the following two conditions:

1. If q is the largest prime factor of r-1, then $q \geq 4\sqrt{r}\ln n$.

2. $n^{(r-1)/q} \neq 1 \mod r$

What is important for the resulting algorithm is the fact that this r is rather small. In fact, its order of magnitude can be shown to be at most $O((\ln n)^6)$ and the whole algorithm has complexity at most $O((\ln n)^{12})$. If a certain conjecture about the density of so-called Sophie-Germaine primes is true, the r is even smaller, namely at most $O((\ln n)^2)$, and the complexity of the whole algorithm drops down to even $O((\ln n)^6)$.

Note that as a first step of the test, n should be checked, if it is even or a perfect power, i.e. of the form a^b or some b>1, as these are exceptional cases. As for the latter condition, it can be checked easily as shown in the following Derive-routine taking advantage of the fact here that every perfect power is also a perfect power for a prime exponent. (Programming gourmets may also notice that I used "exit" at one place instead of the plain "RETURN true".)

```
perfect_power?(n, a_, p_ := 2) :=
  Loop
    a_ := FLOOR(n^(1/p_))
    If a_^p_ = n exit
    If a_ < 2
      RETURN false
    p_ := NEXT_PRIME(p_)

                  1217
perfect_power?(2     ) = true

                   1217
perfect_power?(5 - 2     ) = false
```

What we need next is a routine that computes the powers $(x-a)^n \bmod (x^r-1,n)$. (Note that there is nothing mysterious about this notation. It simply means that we do the computations mod (x^r-1) in the first place and subsequently reduce the coefficients mod n.) Of course, the underlying algorithm is the good old "square and multiply" method, which was already used by the ancient Egyptians to compute additive powers of integers, i.e. products of integers.

```
powers(n, a, r, x, n_, u_, v_ := 1) :=
  Prog
    n_ := n
    u_ := x - a
    Loop
      If n_ = 0
        RETURN v_
      If ODD?(n_)
        v_ := POLY_MODS(REMAINDER(u_·v_, x^r - 1, x), n)
      u_ := POLY_MODS(REMAINDER(u_·u_, x^r - 1, x), n)
      n_ := FLOOR(n_, 2)
```

$$\text{powers}(1009,\ 1,\ 2000) = x^{1009} - 1$$

$$\text{powers}(1009,\ 1,\ 101) = x^{100} - 1$$

At last, we have got all the tools to give an implementation of the AKS-test! (Programming novices may learn a lot by looking at the way how the equality of two polynomials is checked in the program!)

```
AKS_test(n, a_ := 1, q_, r_ := 3) :=
  Prog
    If MOD(n, 2)
      RETURN SOLVE(n = 2)
    If perfect_power?(n)
      RETURN false
    Loop
      If GCD(n, r_) > 1
        RETURN SOLVE(n = r_)
      q_ := MAX((FACTORS(r_ - 1)) COL 1)
      If q_ ≥ 4·√r_·LN(n) ∧ MOD(n^((r_ - 1)/q_), r_) ≠ 1 exit
      r_ := NEXT_PRIME(r_)
    If a_ ≠ 1
      RETURN r_
    Loop
      If TERMS(powers(n, a_, r_, x) - x^MOD(n, r_) - MODS(-a_, n)) = []
        RETURN false
      a_ :+ 1
      If a_ > 2·√r_·LN(n) exit
```

```
AKS_test(4507) = true

AKS_test(4549, 0) = 4547

AKS_test(3212347, 0) = 14387
```

Well, although the example n=4507 was computed correctly in 0.2 second, on the whole the performance is very disappointing. For all n<4547 it is actually determined in the first loop, whether n is prime or not. As soon as the program enters the second loop, the computations becomes either never-ending or result in a memory overflow! This is the case for n=4549 the very first time, and the output above is the value of r. It means that we are dealing with polynomials of degree 4546 here. No wonder, it takes so long! On the other hand, r grows very slowly as the example of the much larger n=3212347 shows, which clearly indicates the slow growth of the complexity.

Hence the program above can serve as an accurate description of the algorithm, but it is far too slow to do primality testing with it actually. And this is a general problem with this algorithm and most experts agree here that the AKS-test is unlikely to become a serious competitor for the tests I have mentioned before, when it comes to deterministic primality testing in practice.

When taking a look at the last chapter of the paper [1] though, there is a very interesting conjecture dealing with a far simpler condition, namely:

Conjecture: If r does not divide n and if

$$(x-1)^n \equiv x^n - 1 \bmod (x^r - 1, n)$$

then either n is prime or $n^2 \equiv 1 \bmod r$.

If this conjecture could be proven, we would have a very simple primality test at hand, indeed! In fact, all we need here is an r that doesn't divide neither n nor $n^2 - 1$, which for any n>1 can be found very quickly as one can be shown to exist in the interval [2, 4 ln n]. Thereafter, testing whether the condition above holds would only take time of order $O((\ln n)^3)$, i.e. the same order of magnitude as for a single Rabin-Miller test! Unlike the Rabin-Miller test, this test would be deterministic though on condition that the conjecture above is true.

Here is again an implementation in Derive. (The letter C in the program name should refer to the conjecture above!)

```
AKSC_Test(n, r_ := 2) :=
  Prog
    If n < 2
      RETURN false
    Loop
      If MOD(n, r_)·MOD(n^2 - 1, r_) ≠ 0 exit
      r_ :+ 1
    SOLVE(TERMS(powers(n, 1, r_, x) - x^MOD(n, r_) - MODS(-1, n)) = [])

                100
AKSC_Test(10     + 267) = true

SELECT(AKSC_Test(n) ≠ PRIME?(n), n, 1, 1000) = []
```

The computation above for my favorite prime $10^{100}+267$ (by the way, the first above 10^{100} !) was carried out in only 0.3s on my machine!

There is not much space left, so let me turn to the programming challenge I gave you when dealing with Pascal's triangle: What is quickest way to set up a primality test based on the fact that a non-negative integer n is a prime if and only if the corresponding row in Pascal's triangle yields exactly two 1's after reducing all entries mod n? Compare your solution with the following one:

```
test(n) := SOLVE(E(MOD(ITERATE(ADJOIN(0, v) + APPEND(v, [0]), v, [1], n), n)) = 2)

test(1009) = true
```

Here are two more programming challenges which I'm going to deal with in my next "Titbits".

1. How many ways can you make a dollar using pennies, nickels, dimes, quarters, half-dollars and dollar coins? (For European readers here the corresponding amounts in cents: 1,5,10,25,50,100.)

2. Is there a way to relabel the faces of two dice such that the probability of rolling any sum will not change? (Note that the numbers of spots are assumed to be positive integers though not necessarily distinct!)

In both cases down-to-earth solutions exist as well as highly sophisticated algebraic ones. More on these problems the next time! Have a go at them in the meanwhile !!! (j.wiesenbauer@tuwien.ac.at)

[1] M. Agrawal, N. Kayal, N. Saxena, *PRIMES is in P*, Preprint (http://www.cse.iitk.ac.in/primality.pdf)

[2] J.Wiesenbauer, *Primality Testing and Factoring Large Numbers with Derive* (to appear)