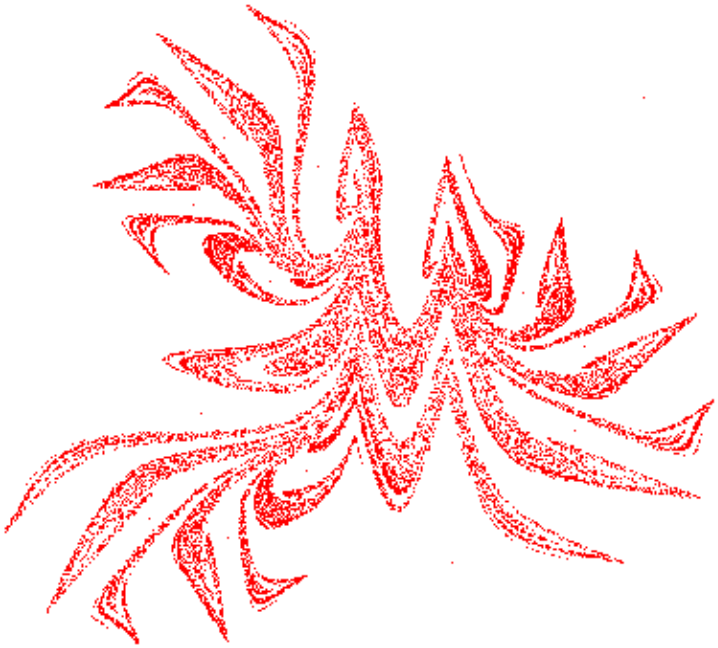
Dynamic Systems on Various Platforms



An Excursion from
Environment & Tourism to Strange Attractors

Josef Böhm

ACDCA Austrian Center for Didactics of Computer Algebra

and

DUG

Dynamic Systems on Various Platforms

Contents

	Introduction	2
1	Tourism and Environment	3
2	Predator and Prey (times 2)	14
3	Collapse of an Ecological System	22
4	Population dynamics with variable rates of birth and death	43
5	The reservoir is flowing over!	50
6	Density dependent Growth: Michaelis-Menten-Kinetics	59
7	What's a Brusselator?	67
8	Bistable Oscillator	79
9	Stock-keeping – with random numbers	87
10	Roessler Attractor	96
11	Gumowski-Mira – and another "attractive" Attractor	105

Introduction

I have been interested in fractals, “Chaos“ and Dynamic Systems since many years. Treating these issues became possible for everybody with availability of computers and the respective software. Special programs like *FractInt* have been on the market since long. But now supported by spreadsheets, computer algebra and own programming it makes much more sense and fun as well to investigate these phenomena.

By a book review a came across *Hartmut Bossel*’s “*System Zoo*” book series. These books are a real repository and treasure box for applied mathematics. There was also information about the program *VENSIM*. This is a commercial simulation software free of charge for teaching purposes.

Then I purchased the *System Zoo*-CD and was very enthusiastic about the many possibilities using *VENSIM*. My ambition came up to treat a not too complex problem (Tourism and Environment) with other tools which are available in our schools. I wanted to learn about the special features, their advantages and disadvantages working through this example.

I had in mind *MS-Excel*, *DERIVE*, *WIRIS*, *TI-NspireCAS* and *GeoGebra*. All these programs offer sliders which promised making the simulations much more dynamic varying the parameters. An additional challenge was to transfer the model into a differential equation or a system of differential equations and then solving it numerically or – if possible – analytically.

I was so much fascinated by this first example that I could not resist proceeding and trying other ones. So it could happen that the paper comprises more than 100 pages finally.

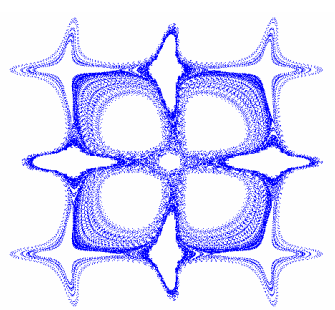
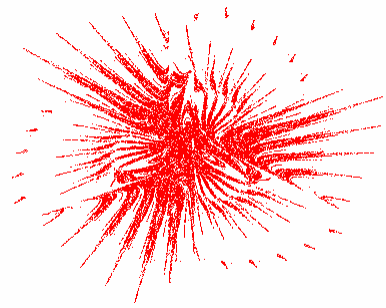
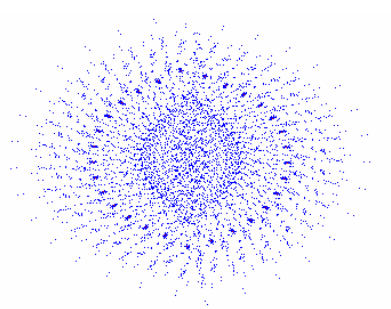
My results are aesthetically appealing - at least in my opinion - and they may wake up appetite for further experimenting and discovering. The “beautiful” and “strange” attractors might make the systems of differential equations interesting even for students who are not so enthusiastic with mathematics.

Unfortunately I could not address here essential interpretation of the generated tables and diagrams. I refer to *Bossel*’s books and many other resources.

All files which are presented in this paper are available on request. Please send an email.

I wish much fun and would be very delighted receiving reactions.

Josef Böhm
nojo.boehm@pgv.at

1 Tourism and Environment

An easy simulation – using various tools

Among the many complex systems which can be found in Chapter 4 *Ecological Systems and Resources* in *Hartmut Bossel*'s *System Zoo* 2[1,7] one can find as example Z411 *Tourism and Environment*.

All *System Zoo* examples are treated with the excellent simulation software *VENSIM PLE*[2], which is free for educational purposes.

After description of the model I will present performing its simulation first with *VENSIM PLE*. It will be followed by a "reproduction" with *MS-Excel* and the *VENSIM*-results serving as reference.

The model can be described by a system of differential equations. Its numerical solution will be achieved applying the Runge-Kutta-method which is implemented in *DERIVE*. We then can compare the results with the results of the discrete model.

Whereas we cannot use sliders in *DERIVE* which could enable studying the influence of various parameters on the behaviour of the model, this is possible with *GeoGebra*[3] and with *TI-Nspire*[4] as well.

Finally we will find the state of equilibrium for important stocks.

Description

It is quite sure that there exists a dynamic linkage between *Tourism* (e.g. as measured by the number of overnight stays) and the *Environment Quality*.

(1) The *Growth* of environment quality is defined by the ENVIRONMENT REGENERATION RATE and is limited by a logistic growth function according to its ENVIRONMENT CARRYING CAPACITY.

Ski resort in the Sierra Nevada, Spain

(2) Occurrence of an *Environmental Stress* by *Tourism* results in a *Loss of Environment Quality* proportional to an ENVIRONMENT DESTRUCTION RATE.

(3) The *Environmental Stress* depends on *Tourism* and on *Environment Quality* and is proportional to both stocks.

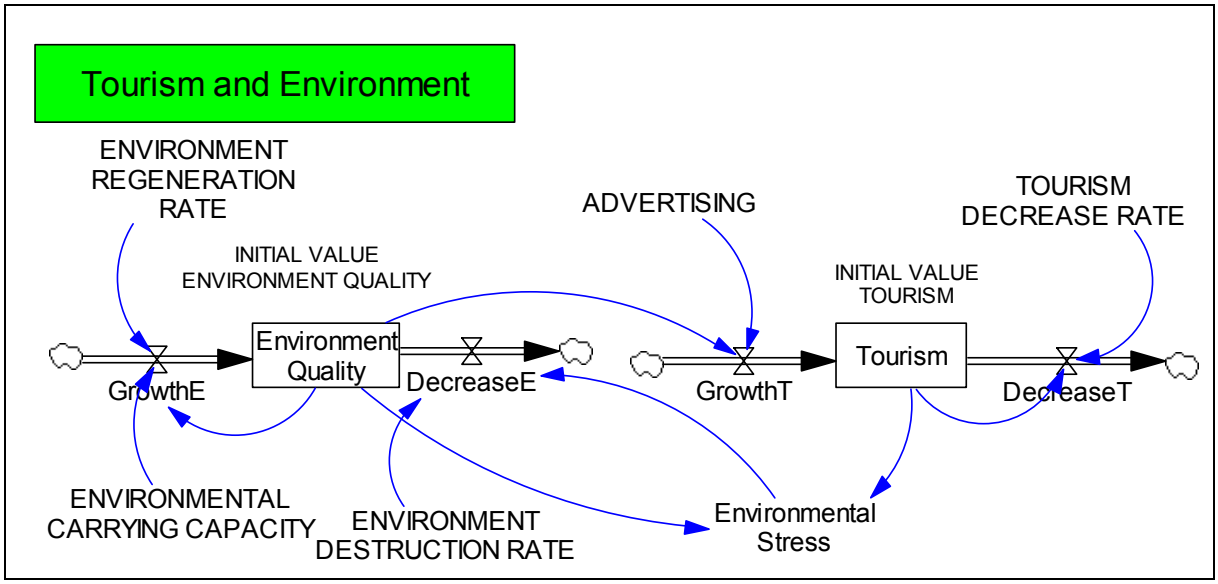
(4) Growth of *Tourism* depends directly on *Environment Quality* and can be reinforced by ADVERTISING.

(5) *Decrease of Tourism* is described by a certain TOURISM DECREASE RATE.

The *VENSIM*-Model

One can load the ready made model[5,7]. But it is much more fruitful to work through this model accompanied by the *VENSIM*-Tutorial[6] and -Manual.

We start sketching the *Stock and Flow diagram* (simulation diagram):



The variable parameters are written in upper case and the stocks in ordinary characters. The boxes contain the interesting stocks.

Relationships between the quantities are entered as equations. The summary of all equations is given in a *VENSIM*-document. (In the original document the equations are given in alphabetical order, here they are ordered according to the type of the quantities.) I will come back to the "Units" in a later chapter.

(17) Tourism = INTEG (GrowthT – DecreaseT, INITIAL VALUE TOURISM)
Units: Tourists

(11) GrowthT = ADVERTISING*Environment Quality
Units: Tourists/Year

(03) DecreaseT = TOURISM DECREASE RATE*Tourism
Units: Tourists/Year

(05) Environment Quality = INTEG (GrowthE – DecreaseE, INITIAL VALUE ENVIRONMENT QUALITY)
Units: Quality

(10) GrowthE = ENVIRONMENT REGENERATION RATE*Environment Quality*
(1 – Environment Quality/ENVIRONMENTAL CARRYING CAPACITY)
Units: Quality/Year

(02) DecreaseE = Environmental Stress*ENVIRONMENT DESTRUCTION RATE
Units: Quality/Year

(08) Environmental Stress = Tourism*Environment Quality
Units: Quality*Tourists

The parameter values and the initial values for *Environment Quality* and *Tourism* are fixed. They can be found in the document and also be printed:

(14) INITIAL VALUE TOURISM = 0.1
Units: Tourists

(13) INITIAL VALUE ENVIRONMENT QUALITY = 1
Units: Quality

(09) FINAL TIME = 20
Units: Year
The final time for the simulation.

(04) INITIAL TIME = 0
Units: Year
The initial time for the simulation.

(15) SAVEPER = TIME STEP
Units: Year [0,?]
The frequency with which output is stored.

(16) TIME STEP = 0.02
Units: Year [0,?]

(07) ENVIRONMENTAL CARRYING CAPACITY = 1
Units: Quality

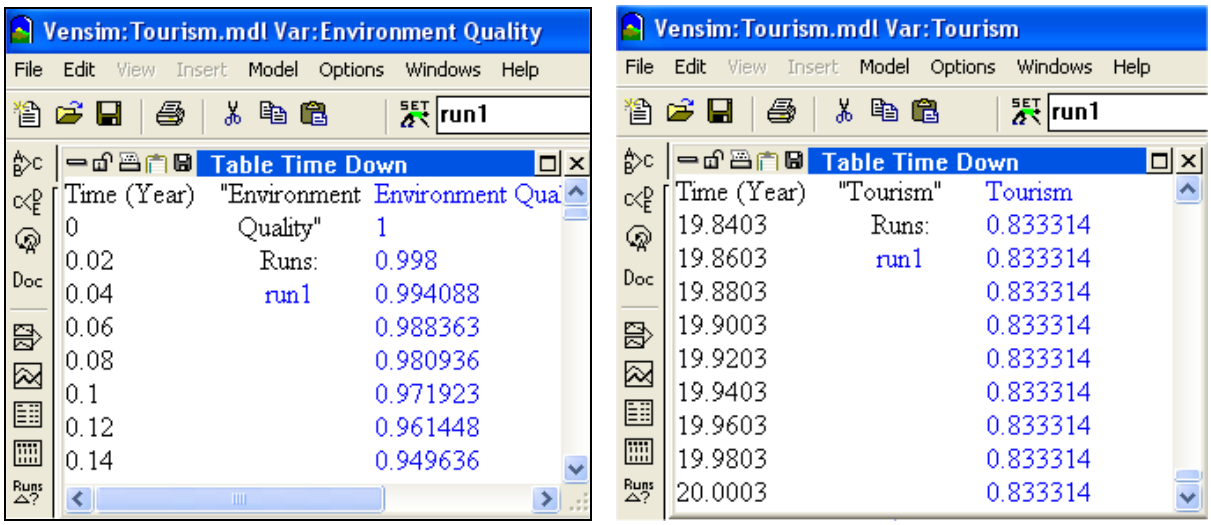
(06) ENVIRONMENT REGENERATION RATE = 1
Units: 1/Year

(04) ENVIRONMENT DESTRUCTION RATE = 1
Units: 1/(Tourists * Year)

(18) TOURISM DECREASE RATE = 1
Units: 1/Year

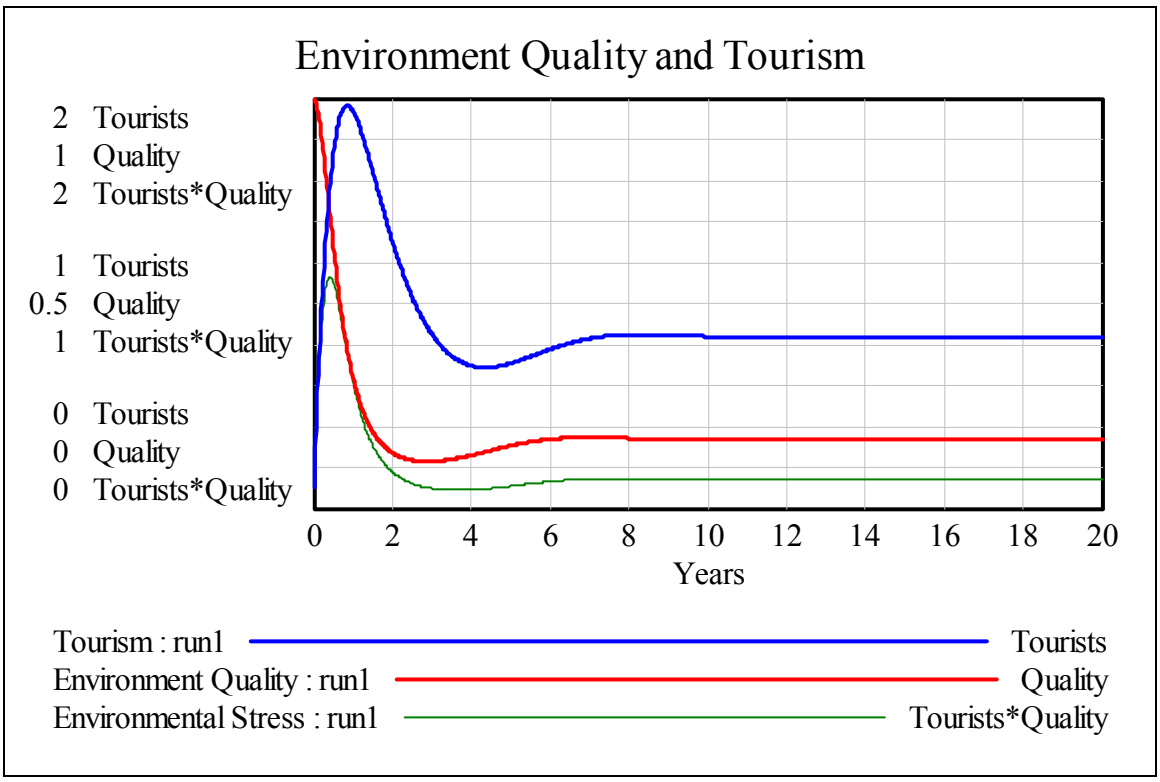
(16) ADVERTISING = 5
Units: Tourists/(Quality * Year)

Now we can run the first simulation. Then we will inspect the results in form of tables and diagrams as well.



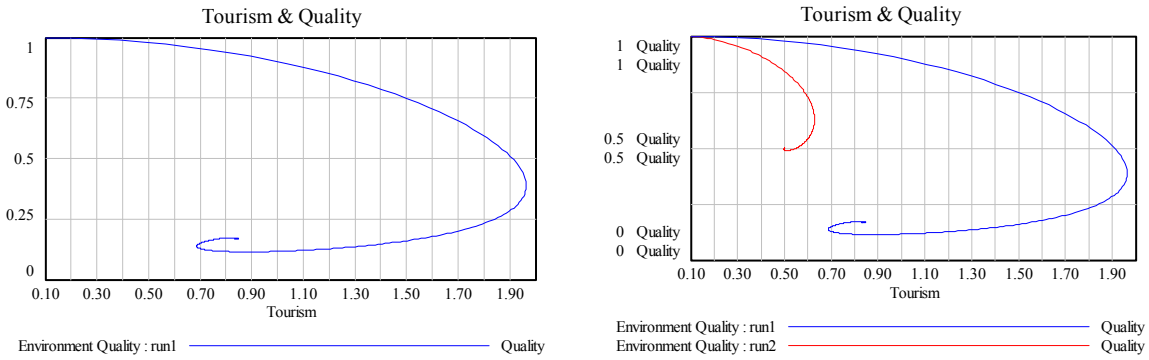
Begin of the *Environment Quality*-table (left) and end of the *Tourism*-table (right)

The diagram shows the development of *Environment Quality*, *Tourism* and *Environmental Stress* for a period of 20 years.
(Take notice of the different scaling on the vertical axis.)



If we are interested in the relationship between *Tourism* and *Environment Quality* then we can plot the respective phase diagram (and pose and answer questions like the following).

What is the effect of more or less advertisement?
We compare ADVERTISING = 5 with ADVERTISING = 1:



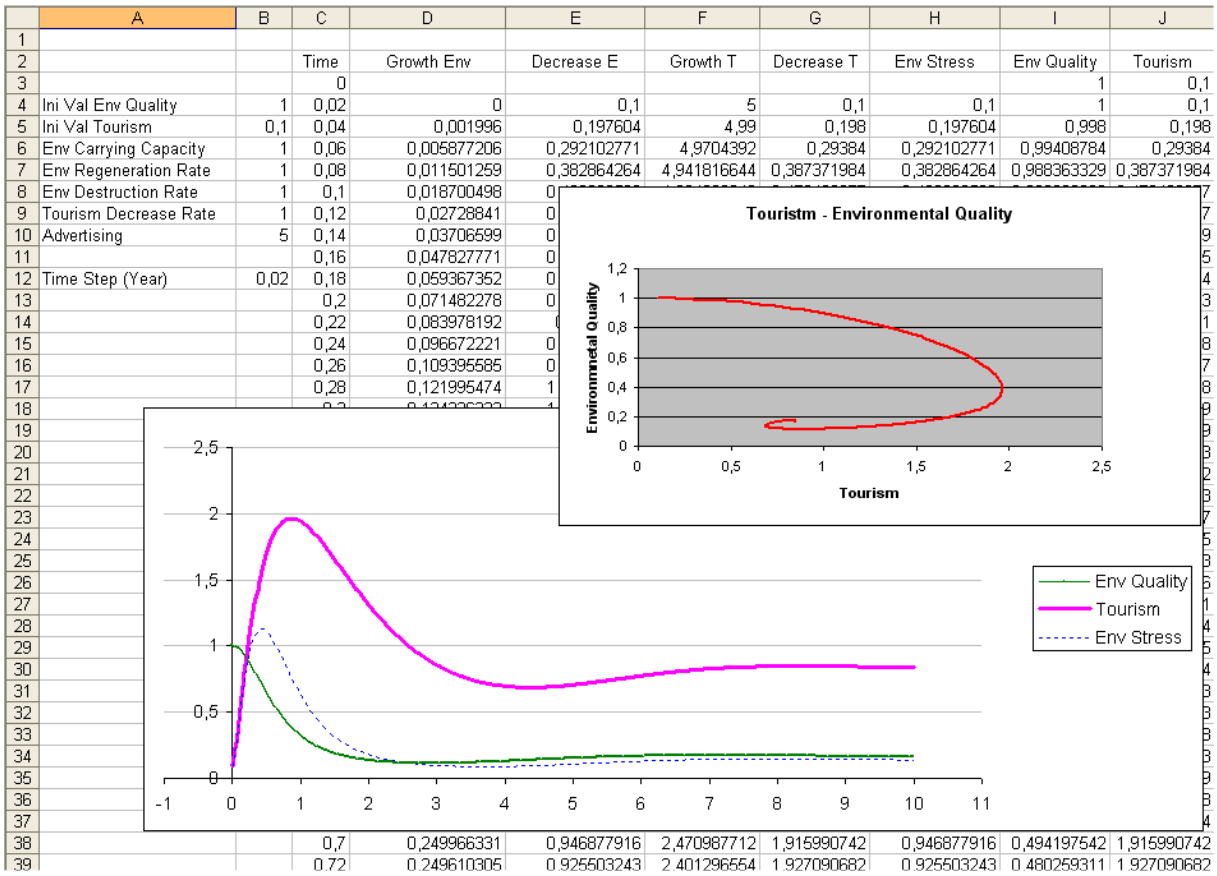
ADVERTISING = 5 (left) compared with ADVERTISING = 1 (right).

We see that increasing advertising results in a remarkable short-term growth of *Tourism* but also in remarkable loss of *Environment Quality*.

Phase diagrams for ADVERTISING = 1, 2, 3, … could be displayed on the same axes.

The *MS-Excel*-Model

The "equations" of the *VENSIM*-model can be transferred 1:1 into the spreadsheet.



	A	B	C	D	E	F	G	H	I	J
1										
2			Time	Growth Env	Decrease E	Growth T	Decrease T	Env Stress	Env Quality	Tourism
3			0						1	0,1
4	Ini Val Env Quality	1	0,02	0	0,1	5	0,1	0,1	1	0,1
5	Ini Val Tourism	0,1	0,04	0,001996	0,197604	4,99	0,198	0,197604	0,998	0,198
6	Env Carrying Capacity	1	0,06	0,005877206	0,292102771	4,9704392	0,29384	0,292102771	0,99408784	0,29384
7	Env Regeneration Rate	1	0,08	0,011501259	0,382864264	4,941816644	0,387371984	0,382864264	0,988363329	0,387371984
8	Env Destruction Rate	1	0,1	0,018700498	0					
9	Tourism Decrease Rate	1	0,12	0,02728841	0					
10	Advertising	5	0,14	0,03706599	0					
11			0,16	0,047827771	0					
12	Time Step (Year)	0,02	0,18	0,059367352	0					
13			0,2	0,071482278	0					
14			0,22	0,083978192	0					
15			0,24	0,096672221	0					
16			0,26	0,109395585	0					
17			0,28	0,121995474	1					
18										
19										
20										
21										
22										
23										
24										
25										
26										
27										
28										
29										
30										
31										
32										
33										
34										
35										
36										
37										
38			0,7	0,249966331	0,946877916	2,470987712	1,915990742	0,946877916	0,494197542	1,915990742
39			0,72	0,249610305	0,925503243	2,401296554	1,927090682	0,925503243	0,480259311	1,927090682

The respective equations are:

Time	Growth E	Decrease E	Growth T	Decrease T	Env Stress
0					
=C3+$B$12	=$B$7*I4*(1-I4/$B$6)	=$B$8*H4	=$B$10*I4	=$B$9*J4	=I4*J4

Env Quality	Tourism
=B4	=B5
=I3+(D3-E3)*$B$12	=J3+(F3-G3)*$B$12

The last rows of the columns for Env Quality and Tourism are:

0.166318761	0.83862131
0.166302328	0.838480759
0.166286419	0.838341377

We can compare these values with the output of the *VENSIM*-model.

The Differential Equation Model with *DERIVE*

The system of differential equations for u (Environmental Quality) and v (Tourism) can easily be derived from the description. The system reads as follows:

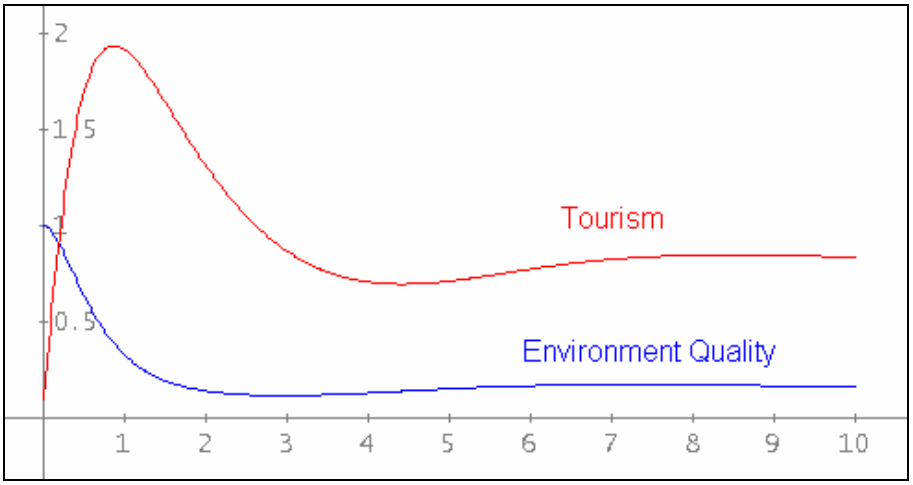
$$eq' = err \cdot eq \cdot \left(1 - \frac{eq}{ecc}\right) - edr \cdot eq \cdot to$$

$$to' = adv \cdot eq - tdr \cdot to$$

The Runge-Kutta method for numerical solution of a DE-system is implemented in *DERIVE*.
Hence:

```
env_tour(err, edr, tdr, adv, ecc, eq_start, to_start, dt, n) :=
      RK([err·eq·(1 – eq/ecc) – eq·to·edr, adv·eq – tdr·to], [t, eq, to],
      [0, eq_start, to_start], dt, n)
```

```
(env_tour(1, 1, 1, 5, 1, 1, 0.1, 0.02, 500))↓↓[1, 2]

(env_tour(1, 1, 1, 5, 1, 1, 0.1, 0.02, 500))↓↓[1, 3]
```



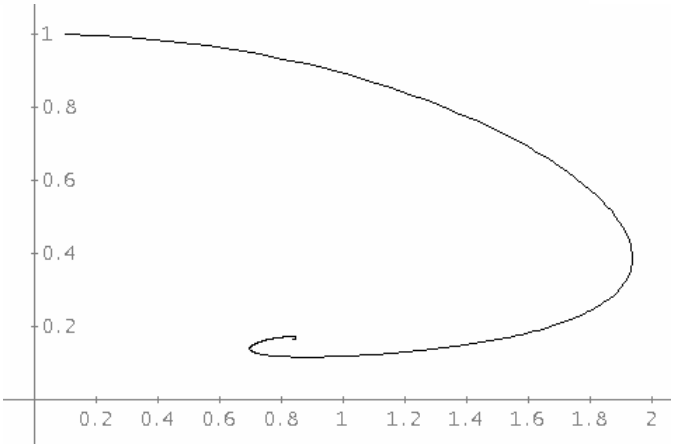
The graphs of *Environment Quality* and *Tourism* are well known.

The last row of the table reads:

```
(env_tour(1, 1, 1, 5, 1, 1, 0.1, 0.02, 500))     = [10, 0.1664091531, 0.8384357232]
                                          501
```

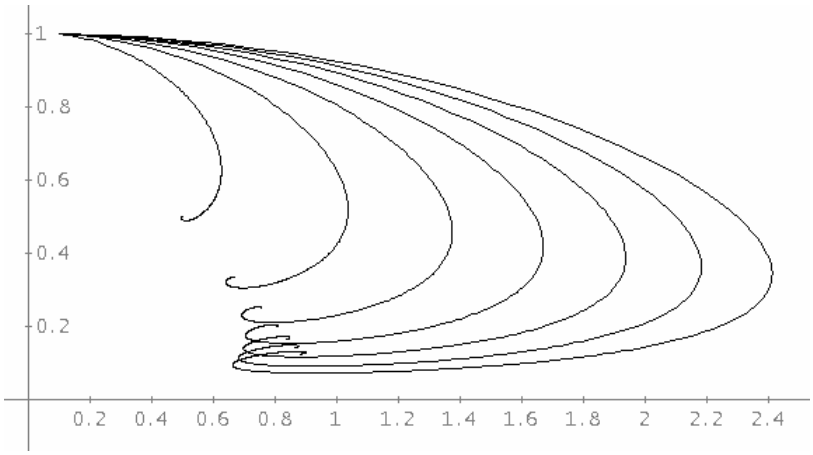
We plot the phase diagram for ADVERTISING $adv = 5$:

```
(env_tour(1, 1, 1, 5, 1, 1, 0.1, 0.02, 500))↓↓[3, 2]
```



Applying the VECTOR-command we can plot all phase diagrams for *adv* = 1 through *adv* = 7 on the same axes:

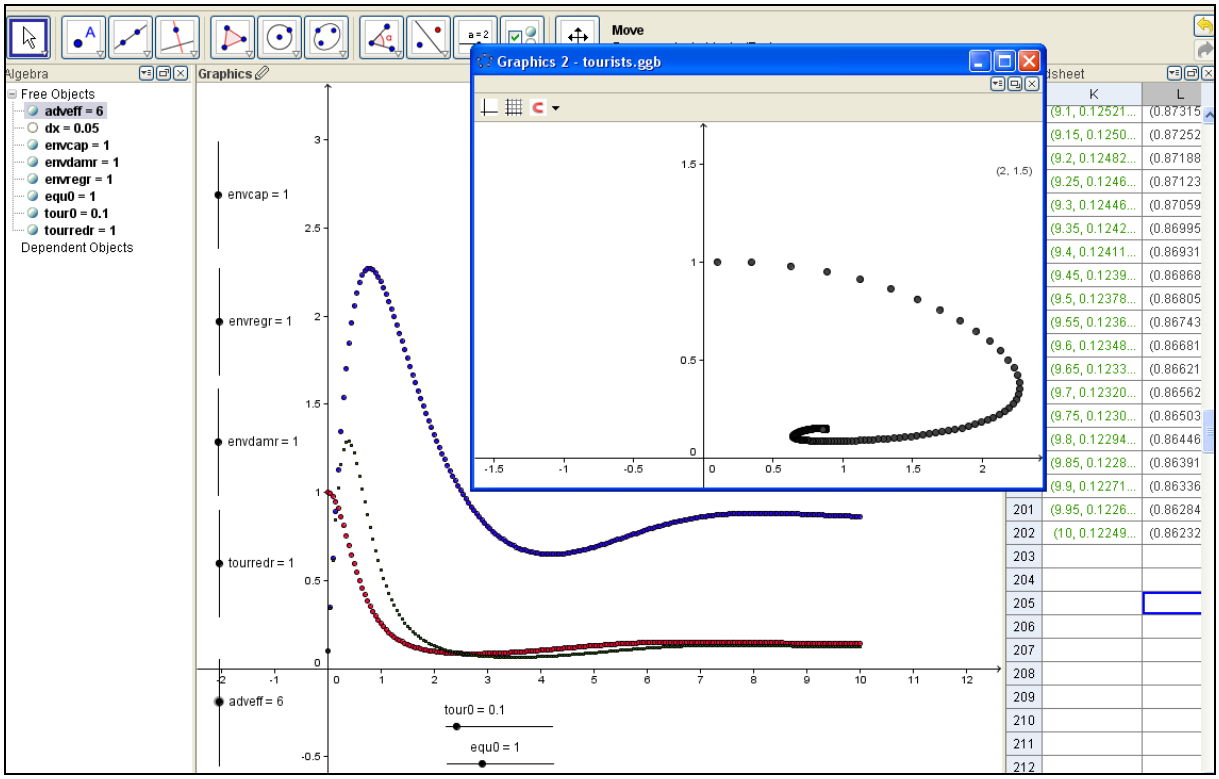
```
VECTOR((umw_tour(1, 1, 1, w, 1, 1, 0.1, 0.02, 500))↓↓[3, 2], w, 1, 7)
```



Later the DE model will help to find possible fix values.

The *DERIVE* implemented slider bars cannot be applied for studying the influence of various parameters. The reason for this lies in the *DERIVE* programming. The RK-routine is based on a recursive structure which needs too much memory when used with general parameters. To study the influence of the parameters better I will try to model the problem with *GeoGebra* and with *TI-NspireCAS*. However, there is also an attractive way for varying the parameters with *VENSIM* (see end of the chapter).

The *GeoGebra*-Model



It doesn't need much time to set up the model but with *dx* = 0.02 we will meet very extended calculation times with this large *GeoGebra* spreadsheet. It can happen that the system hangs up completely. Working with *dx* = 0.05 is successful.

I can display the phase diagram in the second plot window.

How to do it with *TI-NspireCAS*

I found much better calculation properties using the spreadsheet application of *TI-NspireCAS* (Version 3.0). It is no problem at all running 500 time steps of 0.02 years.

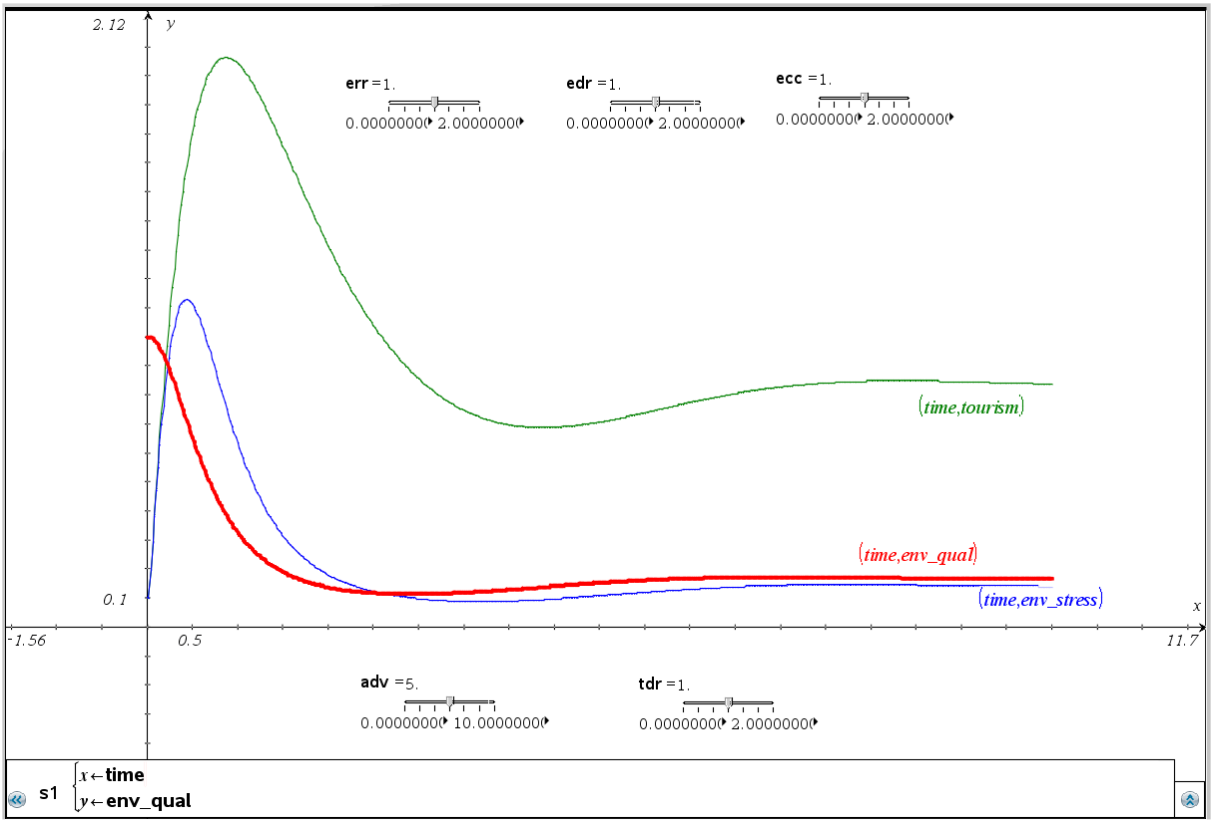
For the parameters connected with the bold presented values sliders must be introduced in the Graphs Application.
Then we can run the simulation in the Lists & Spreadsheet application. The values set by the sliders are valid in the spreadsheet, too, because the variables are linked.

	A	B	C time	D	E	F	G	H	I env_stress	J env_qual	K tourism	L
1	Tourism & En...		_	GrowthE	Decrease...	GrowthT...	Decrease...		_	_	_	
2			_	_					_	_	_	
3	EnvRegRate	**1.0000...**	0	0	0	0	0		0.10000000	1	0.100000...	
4	EnvCarrCap	**1.0000...**	0.020...	0.000000...	0.100000...	5.0000...	0.100000...		0.10000000	1.000000...	0.100000...	
5	EnvDestrRate	**1.0000...**	0.040...	0.001996...	0.197604...	4.9900...	0.198000...		0.19760400	0.998000...	0.198000...	
6	Advertising	**5.0000...**	0.060...	0.005877...	0.292102...	4.9704...	0.293840...		0.29210277	0.994087...	0.293840...	
7	TourDecrRate...	**1.0000...**	0.080...	0.011501...	0.382864...	4.9418...	0.387371...		0.38286426	0.988363...	0.387371...	
8	dx	0.0200...	0.100...	0.018700...	0.469339...	4.9046...	0.478460...		0.46933953	0.980936...	0.478460...	
9			0.120...	0.027288...	0.551066...	4.8596...	0.566985...		0.55106618	0.971923...	0.566985...	
10	EnvQualIni	1	0.140...	0.037065...	0.627669...	4.8072...	0.652837...		0.62766951	0.961447...	0.652837...	
11	TourismIni	0.1000...	0.160...	0.047827...	0.698861...	4.7481...	0.735925...		0.69886148	0.949635...	0.735925...	
12			0.180...	0.059367...	0.764437...	4.6830...	0.816170...		0.76443795	0.936614...	0.816170...	
13			0.200...	0.071482...	0.824274...	4.6125...	0.893509...		0.82427421	0.922513...	0.893509...	
14			0.220...	0.083978...	0.878319...	4.5372...	0.967890...		0.87831946	0.907457...	0.967890...	
15			0.240...	0.096672...	0.926590...	4.4578...	1.039278...		0.92659019	0.891570...	1.039278...	
16			0.260...	0.109395...	0.969163...	4.3748...	1.107649...		0.96916309	0.874972...	1.107649...	
17			0.280...	0.121995...	1.006167...	4.2888...	1.172993...		1.00616748	0.857777...	1.172993...	
18			0.300...	0.134336...	1.037777...	4.2004...	1.235311...		1.03777775	0.840093...	1.235311...	
19			0.320...	0.146299...	1.064205...	4.1101...	1.294614...		1.06420577	0.822024...	1.294614...	
20			0.340...	0.157786...	1.085693...	4.0183...	1.350925...		1.08569371	0.803666...	1.350925...	
21			0.360...	0.168713...	1.102507...	3.9255...	1.404273...		1.10250716	0.785108...	1.404273...	
22			0.380...	0.179013...	1.114928...	3.8321...	1.454698...		1.11492880	0.766432...	1.454698...	
23			0.400...	0.188637...	1.123252...	3.7385...	1.502248...		1.12325261	0.747714...	1.502248...	
24			0.420...	0.197548...	1.127778...	3.6451...	1.546974...		1.12777874	0.729022...	1.546974...	
25			0.440...	0.205724...	1.128808...	3.5520...	1.588937...		1.12880896	0.710417...	1.588937...	

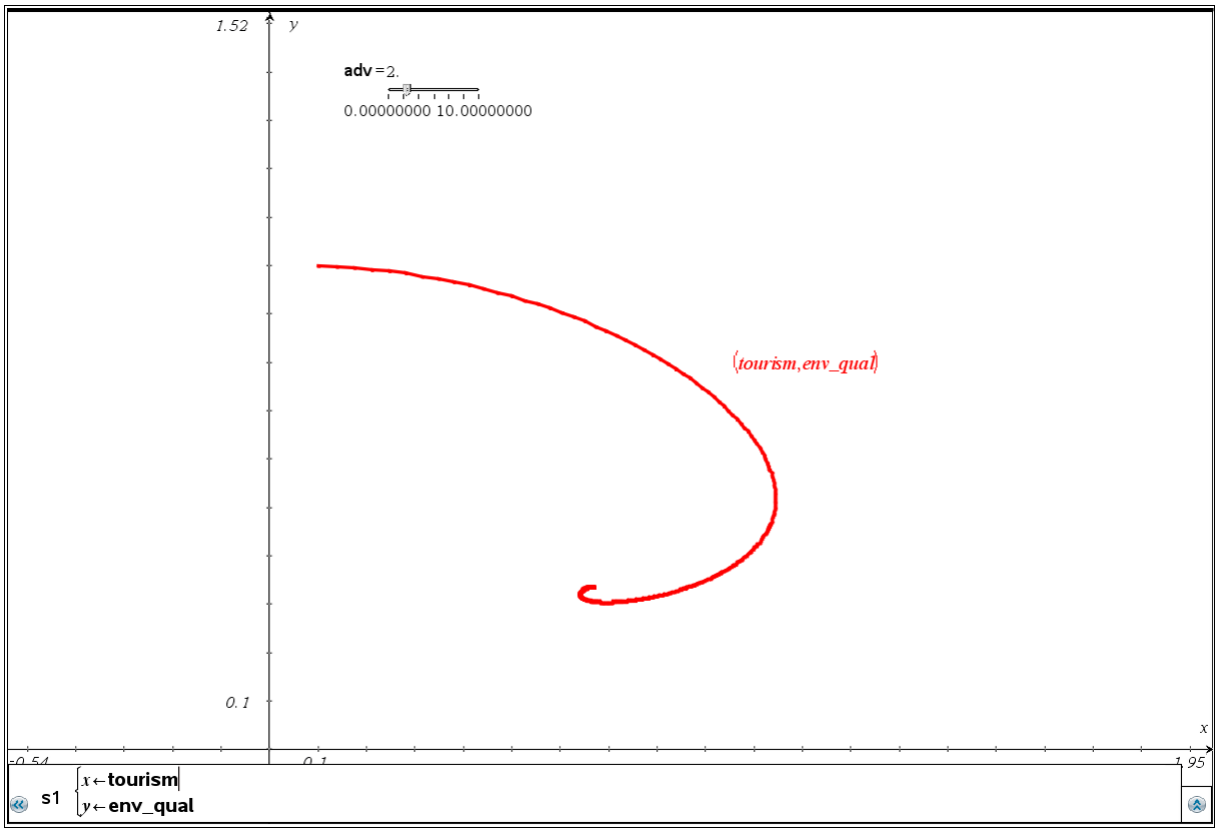
A1 "Tourism & Env"

Enter the list names in the very first rows of columns C, I, J, and K. The list names serve for comfortable plotting the scatter diagrams.

The calculated points are presented connected and according to the choice of the lists we get very attractive graphs which are reacting immediately on the sliders.

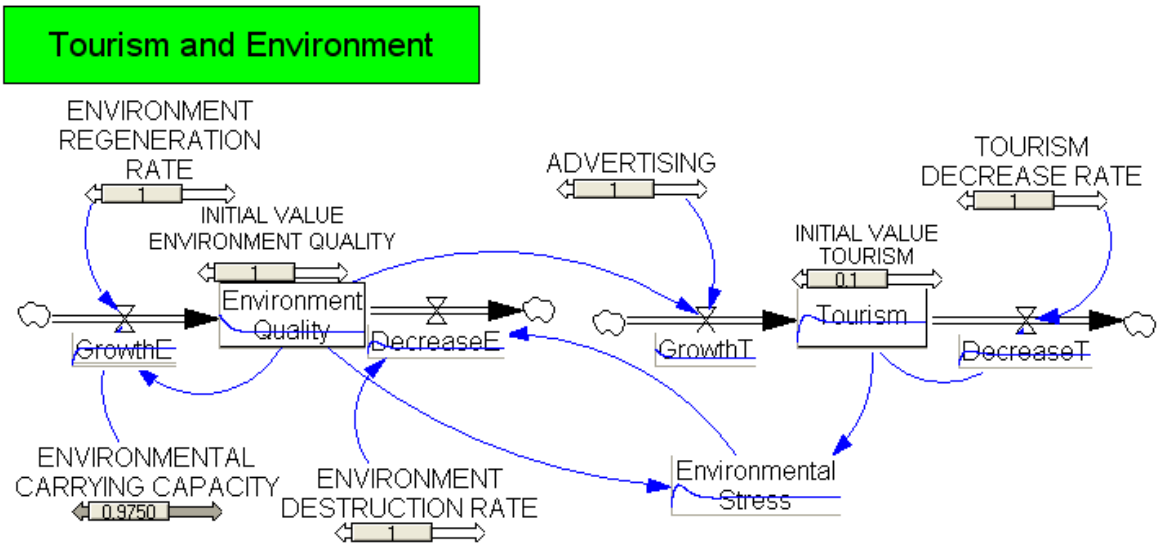


See the phase diagram for *adv* = 2:



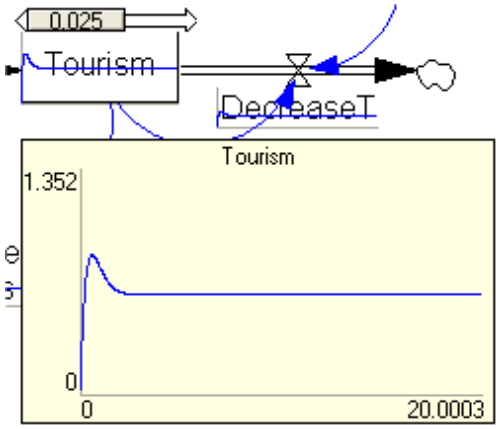
SyntheSim with *VENSIM*

VENSIM offers a kind of sliders, which allows inspecting all stocks (box variables) and flows simultaneously when changing the parameters. However, the inspection windows are quite small. *SyntheSim* is activated via the menu bar and results in a display as follows.

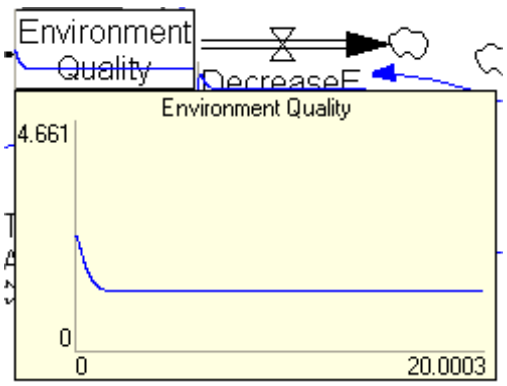


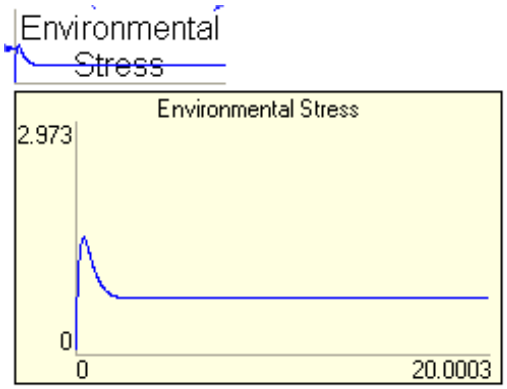
I turned some "parameter screws" and you can see the graphs in blue in the mini-windows. However, when moving the mouse over these mini windows they change their size offering a larger display.

We inspect the display of *Tourism*, *Environment Quality* and *Environment Stress* for the changed parameter values.



These graphs are not of the quality of *Nspire*- or *GeoGebra*-diagrams, but they can be produced without any additional effort. It is not possible to use *SyntheSim* for displaying phase diagrams.





Analytical calculation of the fix values for *Environment Quality* and *Tourism*

The fix values are the results of the demand that change rates of the stocks must become zero.

Therefore, we try to solve the following system of equations (with *DERIVE*):

#11: $\text{SOLUTIONS}\left(0 = err \cdot eq \cdot \left(1 - \frac{eq}{ecc}\right) - edr \cdot eq \cdot to \wedge 0 = adv \cdot eq - tdr \cdot to,\ [eq,\ to]\right)$

#12: $\left[\begin{array}{cc} 0 & 0 \\ \frac{ecc \cdot err \cdot tdr}{adv \cdot ecc \cdot edr + err \cdot tdr} & \frac{adv \cdot ecc \cdot err}{adv \cdot ecc \cdot edr + err \cdot tdr} \end{array}\right]$

#13: $\left[\frac{1 \cdot 1 \cdot 1}{1 \cdot 1 + 1 \cdot 1 \cdot 5},\ \frac{1 \cdot 1 \cdot 5}{1 \cdot 1 + 1 \cdot 1 \cdot 5}\right]$

#14: [0.1666666666, 0.8333333333]

Compare the last rows of the tables of the models from above for *adv* = 5 on page 8!

---

[1] Hartmut Bossel, *System Zoo1, 2, 3*, Books on Demand, Norderstedt
http://www.hartmutbossel.de/ezooinf.htm

[2] *Vensim PLE*, Simulationssoftware, für den Unterrichtsgebrauch, download free of charge at: http://www.vensim.com/download.html

[3] http://www.geogebra.org

[4] http://education.ti.com

[5] Hartmut Bossel, *Systemzoo*, coTec Verlag Rosenheim (CD inkl. *VensimPLE*)
http://www.cotec-verlag.de

[6] http://www.public.asu.edu/~kirkwood/sysdyn/SDRes.htm

[7] http://www.usf.uni-kassel.de/cesr/index.php?option=com_remository&Itemid=141&func=fileinfo&id=109
(The *ZOO MDL*.zip archive contains English language versions of all computer simulation models)

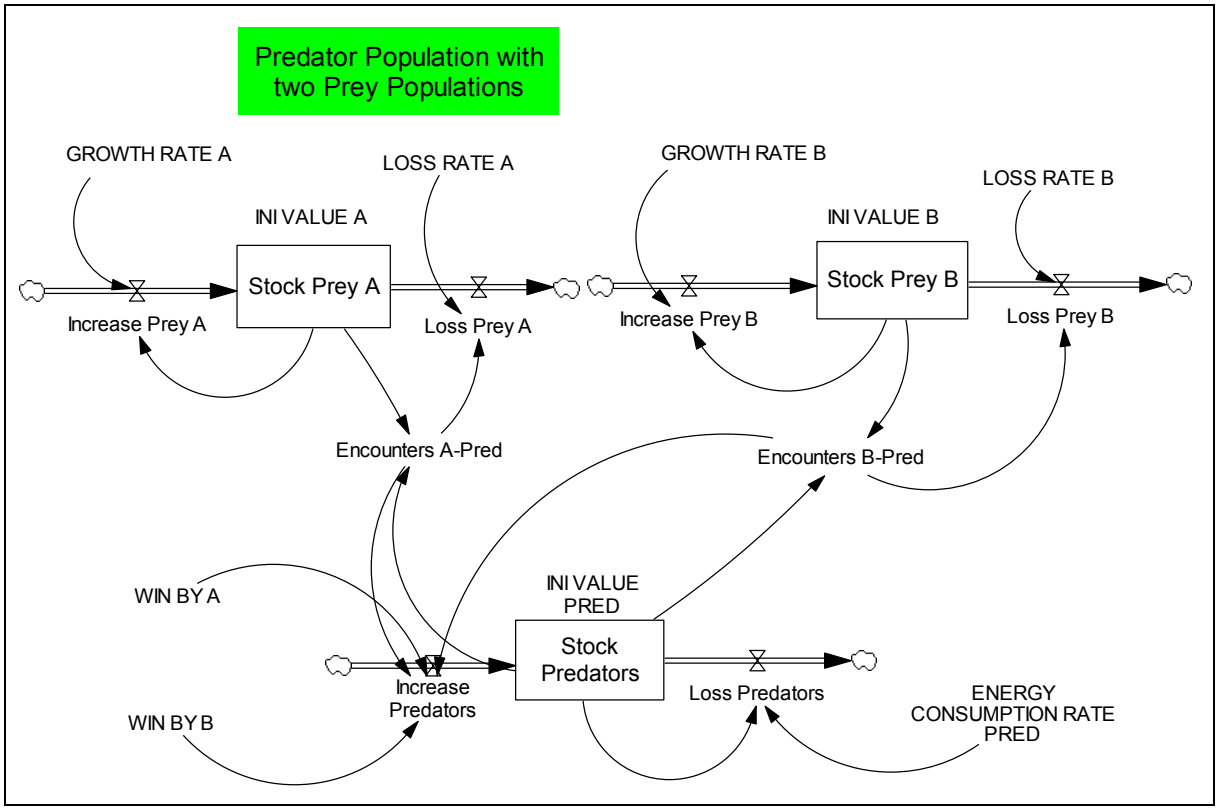
2 Predator and Prey (times 2)

A Variation of the classic Predator-Prey-Model

A predator population takes its energy mainly from two prey populations.

The stock of predators is coupled with the stocks of the prey populations by its hunting successes when meeting (with the respective success rates). The prey stocks are increasing with – different – growth rates. We have success rates of the predators and on the other hand we have specific loss rates of the prey.

Again we start with the *VENSIM PLE* stock and flow diagram and the description of the parameters and definition of the equations.



All parameters and necessary equations are entered and can then be checked and printed in the *VENSIM*-document. I did without entering units. Adding units makes a dimension analysis possible. I also suppressed numbering of the entries.

INI VALUE A = 1

INI VALUE B = 1

INI VALUE PRED = 1

WIN BY A = 0.1

WIN BY B = 0.1

LOSS RATE A = 0.1

LOSS RATE B = 0.1

GROWTH RATE A = 0.1

Lioness in Serengeti NP, Tanzania

GROWTH RATE B = 0.12

ENERGY CONSUMPTION RATE PRED = 0.1

Increase Prey A = GROWTH RATE A ∗ Stock Prey A

Increase Prey B = GROWTH RATE B ∗ Stock Prey B

"Encounters A-P" = Stock Prey A ∗ Stock Predators

"Encounters B-P" = Stock Prey ∗ Stock Predators

Loss Prey A = LOSS RATE A ∗ "Encounters A-P"

Loss Prey B = LOSS RATE B ∗ "Encounters B-P"

Increase Predators = WIN BY A ∗ "Encounters A-P" +
WIN BY B ∗ "Encounters B-P"

Loss Predators = ENERGY CONSUMPTION RATE PRED ∗ Stock Predators

Stock Prey A = INTEG (+Increase Prey A – Loss Prey A,
INI VALUE A)

Stock Prey B = INTEG (+Increase Prey B – Loss Prey B,
INI VALUE B)

Stock Predators = INTEG (+Increase Predators – Loss Predators,
INI VALUE PRED)

INITIAL TIME = 0
Units: Month
The initial time for the simulation.

FINAL TIME = 200
Units: Month
The final time for the simulation.

SAVEPER = TIME STEP
Units: Month [0,?]
The frequency with which output is stored.

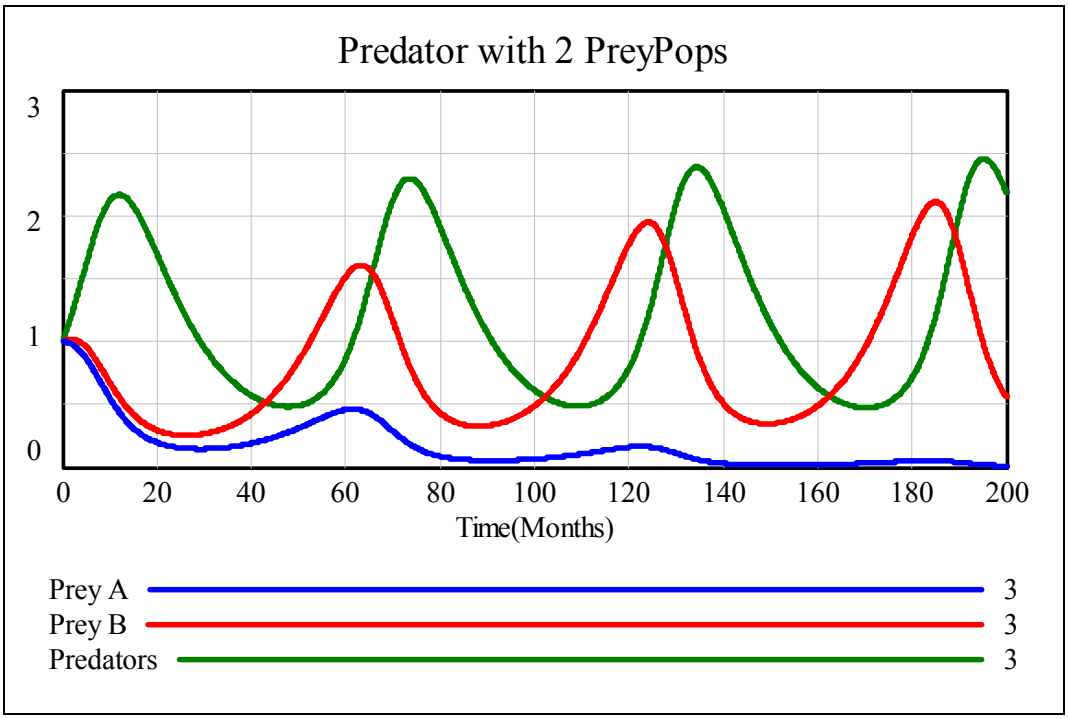
TIME STEP = 0.1
Units: Month [0,?]
The time step for the simulation.

Some prey in Serengeti National Park, Tanzania

I changed the order of the document output.

In *Bossel's* original model a time step of 0.05 is used. Taking 0.1 does not result in a recognisable change for the worse.

In the following the diagrams and the table can be produced, displayed and printed. All the diagrams and tables can easily be transferred to other documents with Copy and Paste.



These are the last rows of the table (for a later comparison):

Predator with 2 PreyPops

Time (Month	Stock Prey A	Stock Prey B	Stock Preda
199.303	0.0110	0.5977	2.246
199.403	0.0109	0.5914	2.237
199.503	0.0108	0.5853	2.228
199.603	0.0106	0.5793	2.219
199.703	0.0105	0.5734	2.210
199.803	0.0104	0.5676	2.201
199.903	0.0103	0.5619	2.192

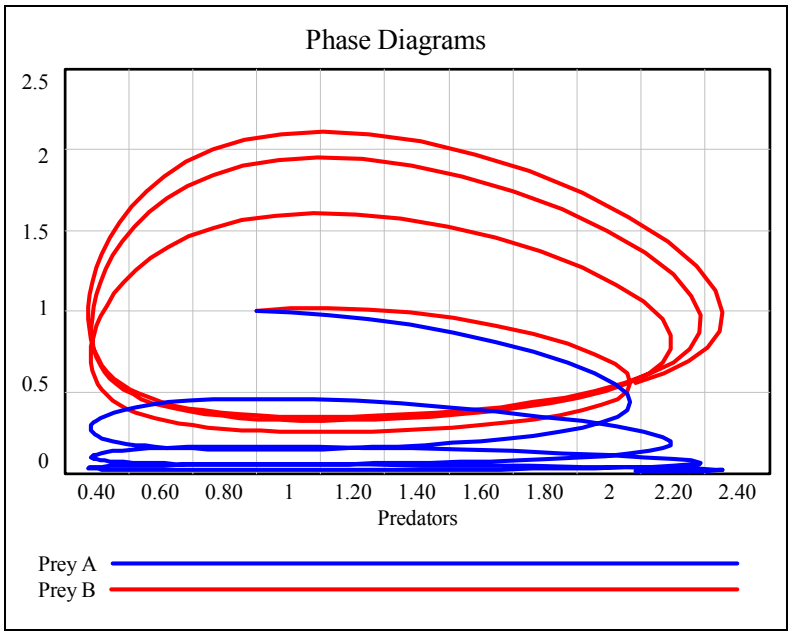
Euler method

Predator with 2 PreyPops

Time (Month	Stock Prey A	Stock Prey B	Stock Pred
199.303	0.0110	0.5924	2.047
199.403	0.0109	0.5874	2.039
199.503	0.0108	0.5826	2.030
199.603	0.0107	0.5778	2.022
199.703	0.0106	0.5731	2.014
199.803	0.0105	0.5684	2.005
199.903	0.0103	0.5639	1.997

Runge-Kutta-method

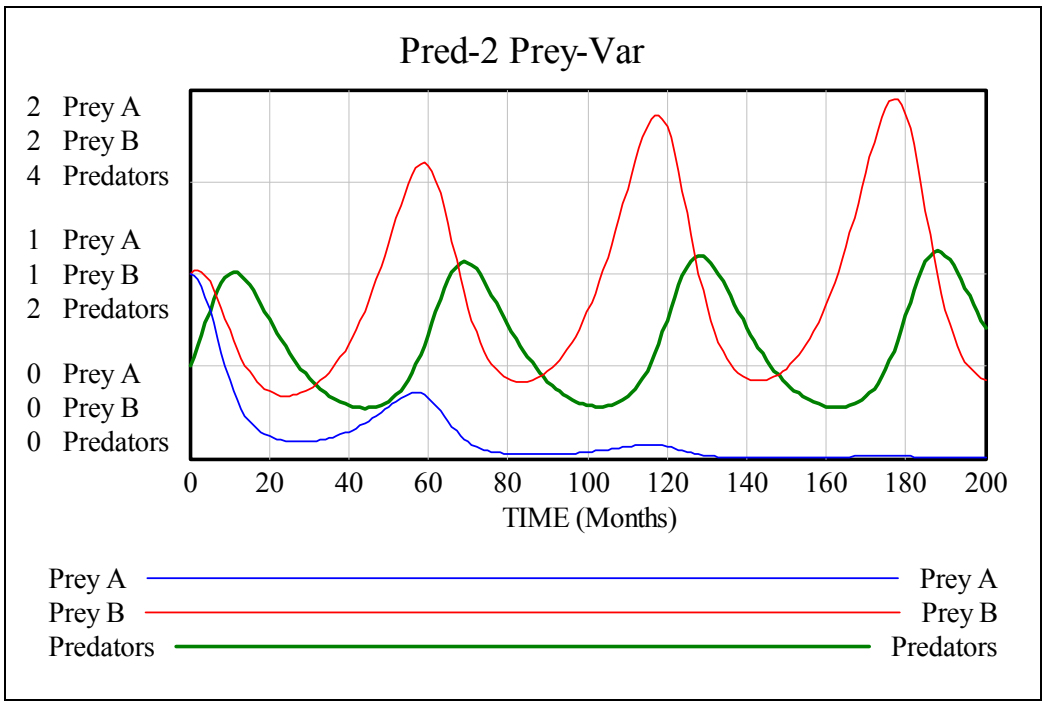
For the numerical calculation we can choose between EULER-method and RUNGE-KUTTA-method. The graphs presented here are based on the EULER-method realization.



I change some parameter values as follows:

GROWTH RATE A = 0.15
LOSS RATE A = 0.15
WIN BY A = 0.12

SAVEPER = 1



The values at the end of the months are saved and printed in tables and graphs (SAVEPER = 1).

The differential equation model with *DERIVE*

For treatment with *DERIVE* I rewrite the problem as a system of ODEs which then will be solved numerically using the Runge-Kutta-algorithm. I can use the *System Zoo*-time step = 0.05. Then I will compare the resulting values for the last months with the values obtained with *VENSIM PLE*.

$$\begin{aligned} pr' &= pr(wa \cdot pa + wb \cdot pb - lr) \\ pa' &= ga \cdot pa - pr \cdot pa \cdot la \qquad ; pr(0) = pa(0) = pb(0) = 1 \\ pb' &= gb \cdot pb - pr \cdot pb \cdot lb \end{aligned}$$

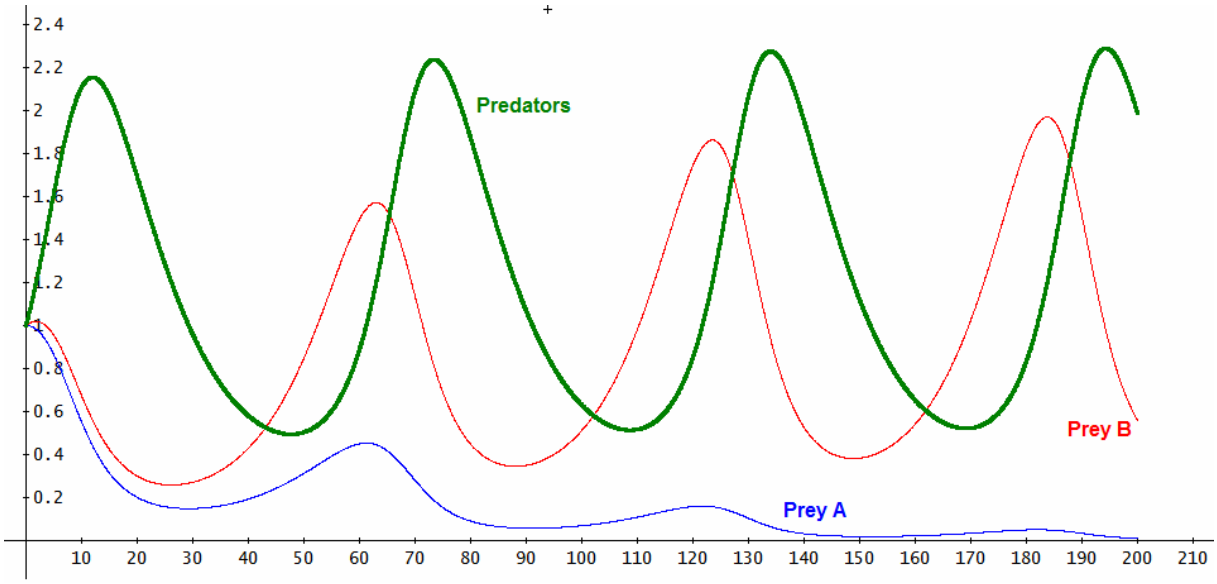
The respective function with all parameters can be defined …

```
pp2(ga, gb, la, lb, lp, wa, wb, pa_ini, pb_ini, pr_ini, dt, n) :=
  RK([ga·pa - la·pa·pr, gb·pb - lb·pb·pr, pr·(wa·pa + wb·pb - lp)],
  [t, pa, pb, pr], [0, pa_ini, pb_ini, pr_ini], dt, n)
```

… and then be evaluated (compare with the last rows of the *VENSIM*-table!).

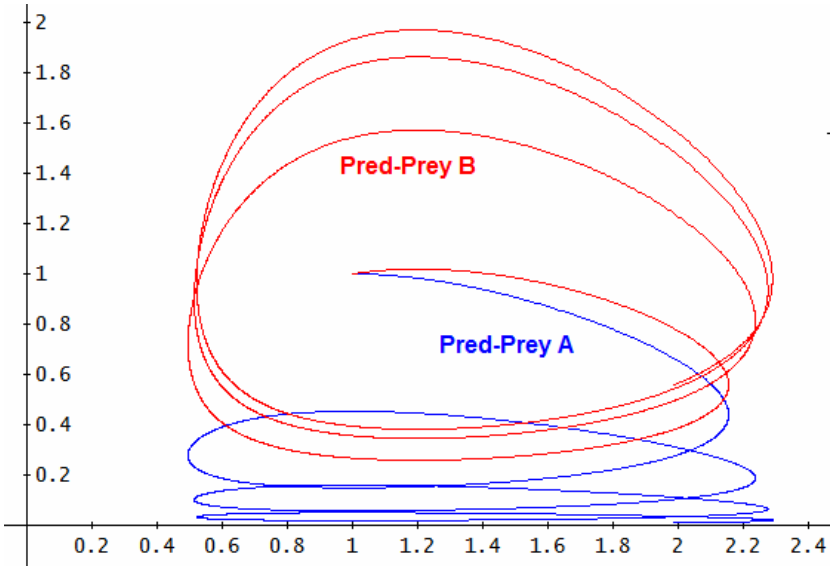
$$\begin{bmatrix} 199.85 & 0.01040077792 & 0.5661621969 & 2.000997472 \\ 199.9 & 0.01034896194 & 0.563905238 & 1.996753928 \\ 199.95 & 0.01029762295 & 0.5616692132 & 1.99249649 \\ 200 & 0.01024675712 & 0.5594539828 & 1.988225495 \end{bmatrix}$$

```
(pp2(0.1, 0.12, 0.1, 0.1, 0.1, 0.1, 0.1, 1, 1, 1, 0.05, 4000))↓↓[1, 2]
(pp2(0.1, 0.12, 0.1, 0.1, 0.1, 0.1, 0.1, 1, 1, 1, 0.05, 4000))↓↓[1, 3]
(pp2(0.1, 0.12, 0.1, 0.1, 0.1, 0.1, 0.1, 1, 1, 1, 0.05, 4000))↓↓[1, 4]
```



Finally we will create the phase diagrams selecting the respective columns of the matrix:

```
(rbb(0.1, 0.12, 0.1, 0.1, 0.1, 0.1, 0.1, 1, 1, 1, 0.05, 4000))↓↓[4, 3]
(rbb(0.1, 0.12, 0.1, 0.1, 0.1, 0.1, 0.1, 1, 1, 1, 0.05, 4000))↓↓[4, 2]
```



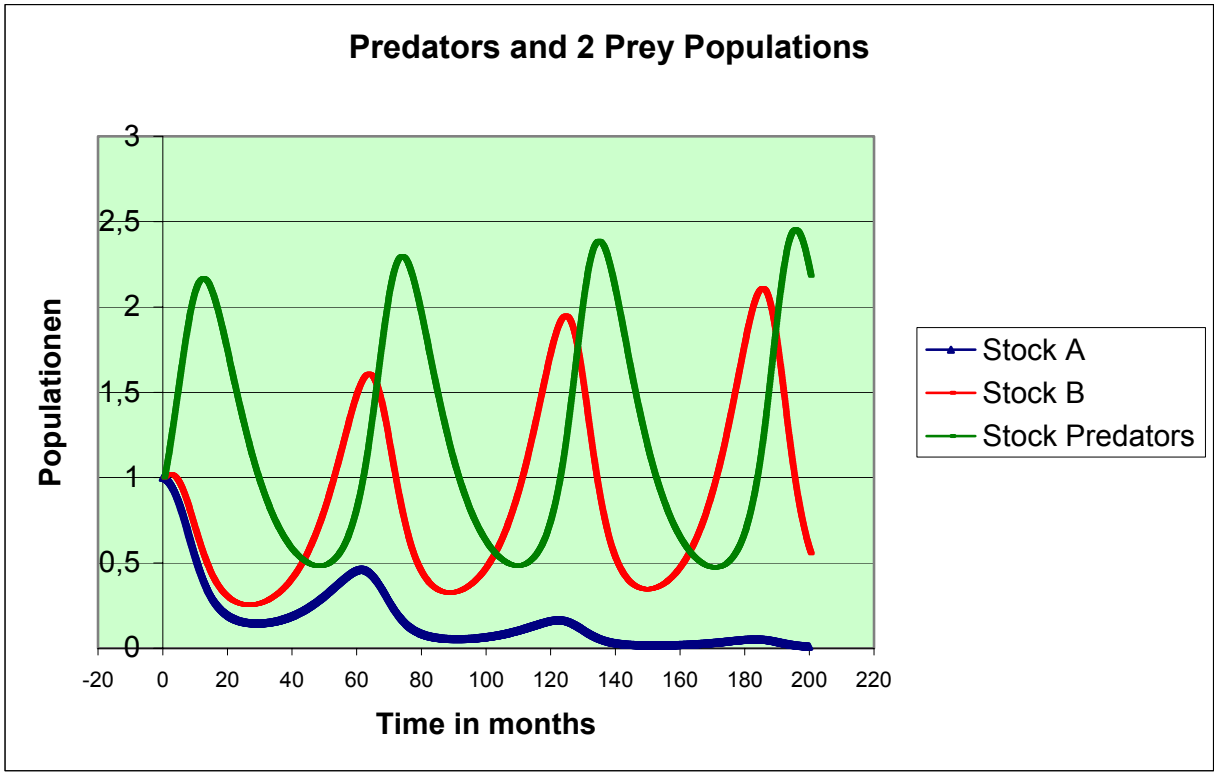
The *MS-Excel*-Model

The equations of the *VENSIM*-model can easily be transferred into the spreadsheet.

J4 =J3+(E4-G4)*$B$15

	A	B	C	D	E	F	G	H	I	J	K
1											
2			Time	Incr A	Incr B	Loss A	Loss B	Incr Pr	Stock A	Stock B	Stock Predators
3			0						1	1	1
4	Ini Value A	1	0,1	0,1	0,12	0,1	0,1	0,1	1	1,002	1,01
5	Ini Value B	1	0,2	0,1	0,1202	0,101	0,101	0,1012	0,9999	1,0039038	1,0201202
6	Ini Value Predators	1	0,3	0,1	0,1205	0,102	0,102	0,1024	0,99969882	1,00570962	1,030360205
7	Growth Rate A	0,1	0,4	0,1	0,1207	0,103	0,104	0,1036	0,99939531	1,007415704	1,040719534
8	Growth Rate B	0,12	0,5	0,1	0,1209	0,104	0,105	0,1048	0,99898836	1,00902032	1,051197613
9	Loss Rate A	0,1	0,6	0,1	0,1211	0,105	0,106	0,106	0,9984769	1,010521767	1,061793776
10	Loss Rate B	0,1	0,7	0,1	0,1213	0,106	0,107	0,1071	0,9978599	1,011918371	1,072507261
11	Energy Cons Rate Pred	0,1	0,8	0,1	0,1214	0,107	0,109	0,1083	0,99713638	1,013208493	1,083337206
12	Win by A	0,1	0,9	0,1	0,1216	0,108	0,11	0,1095	0,9963054	1,01439053	1,094282648
13	Win By B	0,1	1	0,1	0,1217	0,109	0,111	0,1106	0,99536605	1,015462917	1,105342518
14			1,1	0,1	0,1219	0,11	0,112	0,1117	0,99431751	1,016424129	1,116515641
15	Time Step (Month)	0,1	1,2	0,099	0,122	0,111	0,113	0,1129	0,99315897	1,017272684	1,127800729
16			1,3	0,099	0,1221	0,112	0,115	0,114	0,99188971	1,018007147	1,139196385
17			1,4	0,099	0,1222	0,113	0,116	0,115	0,99050904	1,018626133	1,150701093
18			1,5	0,099	0,1222	0,114	0,117	0,1161	0,98901633	1,019128304	1,162313223

It is no problem at all producing the time diagrams (I don't present the phase diagram).



Compare again the values given in the last rows of the *Excel* table (2000 rows!) with the respective *VENSIM*- and *DERIVE*-values:

Stock A	Stock B	Stock Predators
0.01051026	0.573381557	2.21030495
0.01038306	0.567588655	2.20110769
0.01025834	0.561906481	2.191818393
0.01013608	0.556333389	2.182441023

In *System Zoo 2* we find the question which provision(s) could (should) be taken that population A despite its shortcoming by its growth rate could survive population B?

This question could be answered much more comfortable applying sliders. We will introduce sliders in *Excel* in a later chapter.

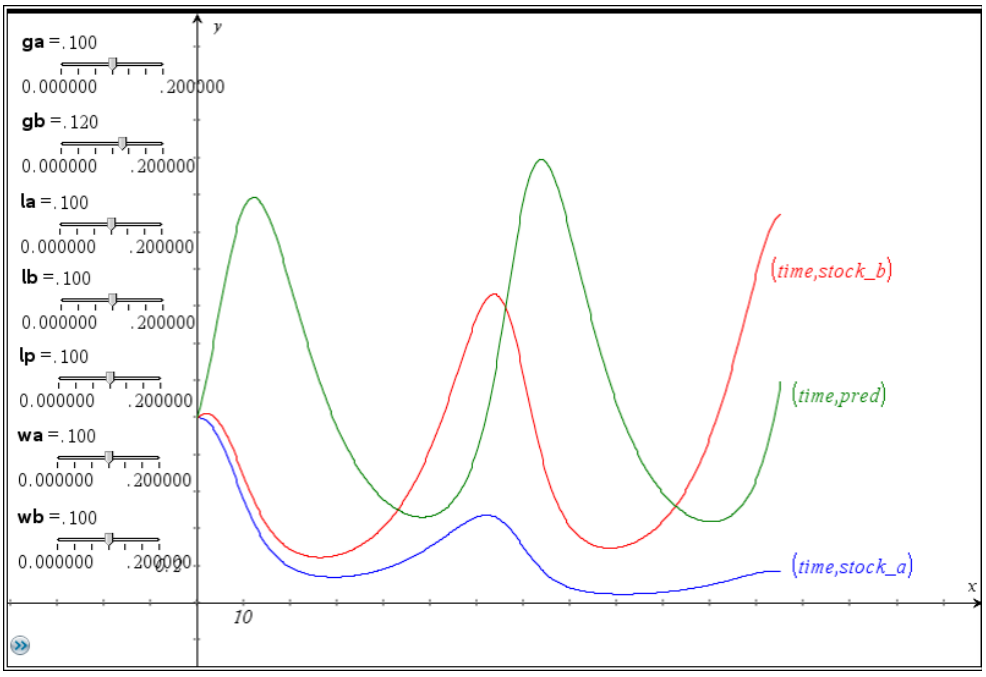
GeoGebra offers sliders but larger tables like this one need (too) long calculation times. We introduce sliders with *TI-NspireCAS*.

Working out with *TI-NspireCAS*

In order to keep calculation time reasonable we take a time step of 0.25 (in *System Zoo* we have 0.05!).

	A	B	C time	D	E	F	G	H	I stock_a	J stock_b	K pred
1			Time...	Incr A	Incr B	Loss A	Loss B	Incr P	Stock A	Stock B	Stock Pr...
2			0	0.100000	0.120000	0.085000	0.110000	0.095000	1	1	1
3	Ini Value A	1	0.25...	0.100375	0.120300	0.087345	0.112894	0.097864	1.003750	1.002500	1.023750
4	Ini Value B	1	0.50...	0.100701	0.120522	0.089723	0.115806	0.100707	1.007007	1.004351	1.048216
5	Ini Value Pred	1	0.75...	0.100975	0.120664	0.092128	0.118726	0.103515	1.009752	1.005531	1.073393
6			1.00...	0.101196	0.120722	0.094556	0.121647	0.106276	1.011964	1.006015	1.099271
7	Growth Rate A	**0.1000...**	1.25...	0.101362	0.120694	0.097000	0.124559	0.108975	1.013624	1.005784	1.125840
8	Growth Rate B	**0.1200...**	1.50...	0.101471	0.120578	0.099454	0.127450	0.111596	1.014714	1.004818	1.153084
9	Loss Rate A	**0.0850...**	1.75...	0.101522	0.120372	0.101911	0.130311	0.114124	1.015219	1.003099	1.180983
10	Loss Rate B	**0.1100...**	2.00...	0.101512	0.120074	0.104363	0.133128	0.116540	1.015121	1.000615	1.209514
11	Loss Rate Pred	**0.1000...**	2.25...	0.101441	0.119682	0.106802	0.135890	0.118828	1.014409	0.997351	1.238649
12	Win by A	**0.0850...**	2.50...	0.101307	0.119196	0.109219	0.138584	0.120968	1.013068	0.993299	1.268356
13	Win by B	**0.1100...**	2.75...	0.101109	0.118614	0.111605	0.141196	0.122941	1.011090	0.988452	1.298598
14			3.00...	0.100847	0.117937	0.113950	0.143713	0.124729	1.008466	0.982806	1.329333
15	Time step (month)	0.2500...	3.25...	0.100519	0.117163	0.116244	0.146119	0.126312	1.005190	0.976362	1.360516
16			3.50...	0.100126	0.116295	0.118477	0.148402	0.127670	1.001259	0.969124	1.392094

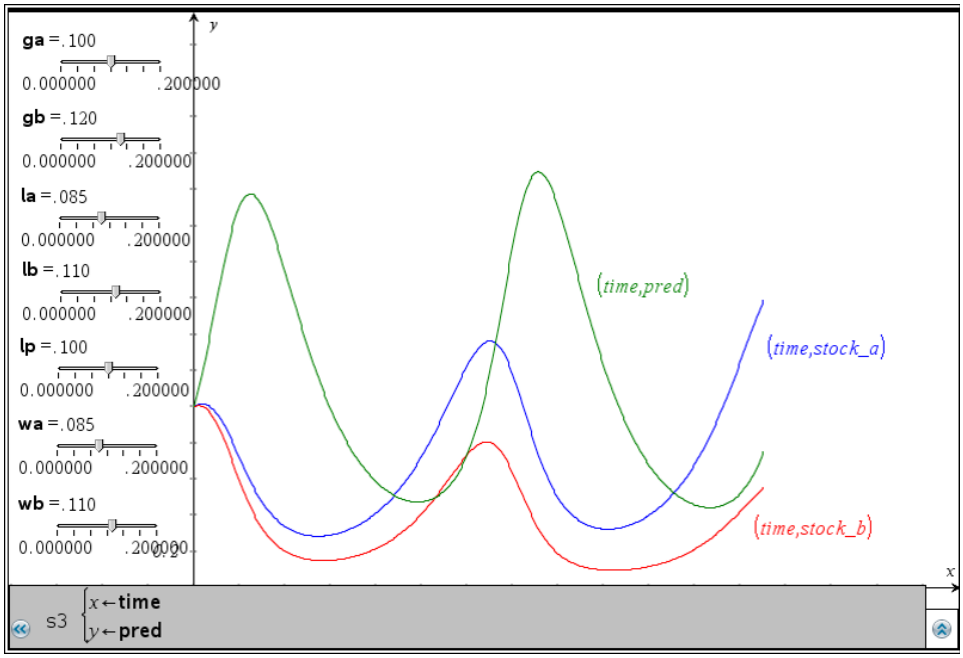
The table for "only" 125 months:



The stocks for A, B and the predators after 125 months (end of the table) are: 0.1716, 2.0971 und 1.1919. The respective values in the *Excel*-table are: 0.1590, 1.9387 und 1.3307.

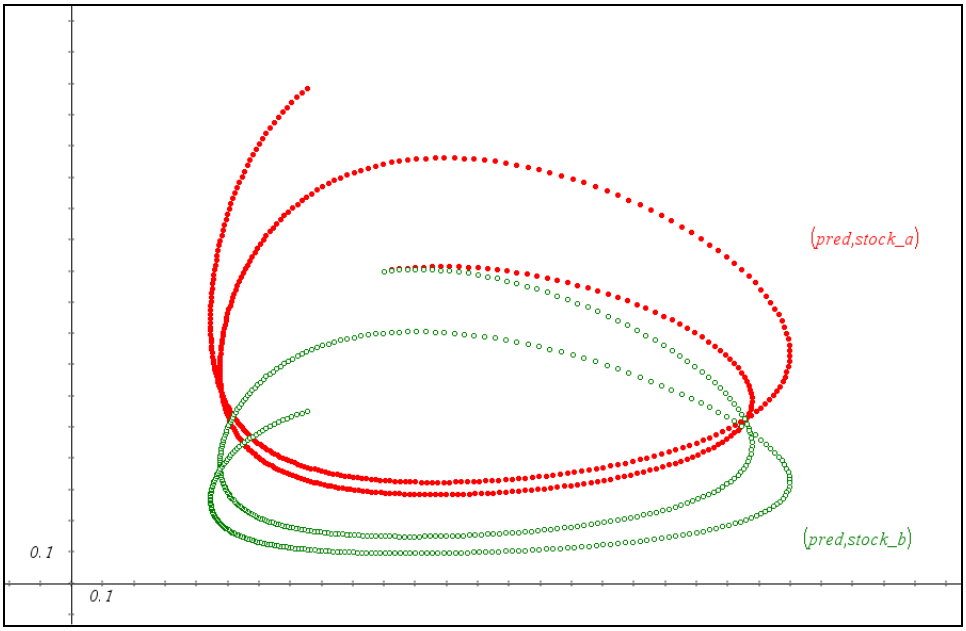
The diagram looks very similar. It seems that the very rough simplification does not make worse the modelling.

What could be done supporting prey population B to survive population A?



The diagram shows that appropriate steps should be taken to decrease the loss rate of A and to simultaneously increase the loss rate of B. A must be made “less attracting“ for the predators. Is it possible to adopt such protection mechanisms?

Finally we create the predator-prey phase diagrams for this “future“ model:



3 Collapse of an Ecosystem

A more complex simulation with a historical background

Bossel cites a source which explains the collapse of the white-tailed-deer population in the Kaibab Forest (North Rim of Grand Canyon) as a consequence of shooting the predators which feed on these deer.

Prior to 1907 there was a population of approx 4000 deer living on an area of about 320 000 ha. Within a period of 15 to 20 years hunting predators (cougars, wolves and coyotes) was forced and about 8000 of them were shot. This was followed by an enormous growth of the deer population.

Sycamore Canyon, Kaibab National Forest

White-tailed-deer (Odocoilus virginianus)

It was 1918 when the stock of deer had more than decupled. This caused an overexertion of food supplies. Until 1924 the deer population reached a number of 100 000 animals. Caused by lack of food 60% of the animals perished in the following two winter periods.

Vegetation was destroyed in such a way, that only half of the deer population compared with its size before this development could exist in the long run.

1974 tried *Goodman* to simulate this system by a model which delivered results matching satisfying with the real process.

Explanation of the model

The *Deer* feed on an AREA (320 000 ha) on *Food*. *Increase Food* is governed by its *Regeneration Time*. The *Growth rate Deer* is a function of *Food Supply*. This is the amount of food available for each animal. *Food Demand* depends on the stock of *Deer* and on the DAILY REQUirement of one deer (2000 Kcal). *Food* grows again according to the MAX FOOD CAPACITY (480 Mio Kcal). *Increase Food* is determined by the *Regeneration Time* which is a function of *Vegetation Density*.

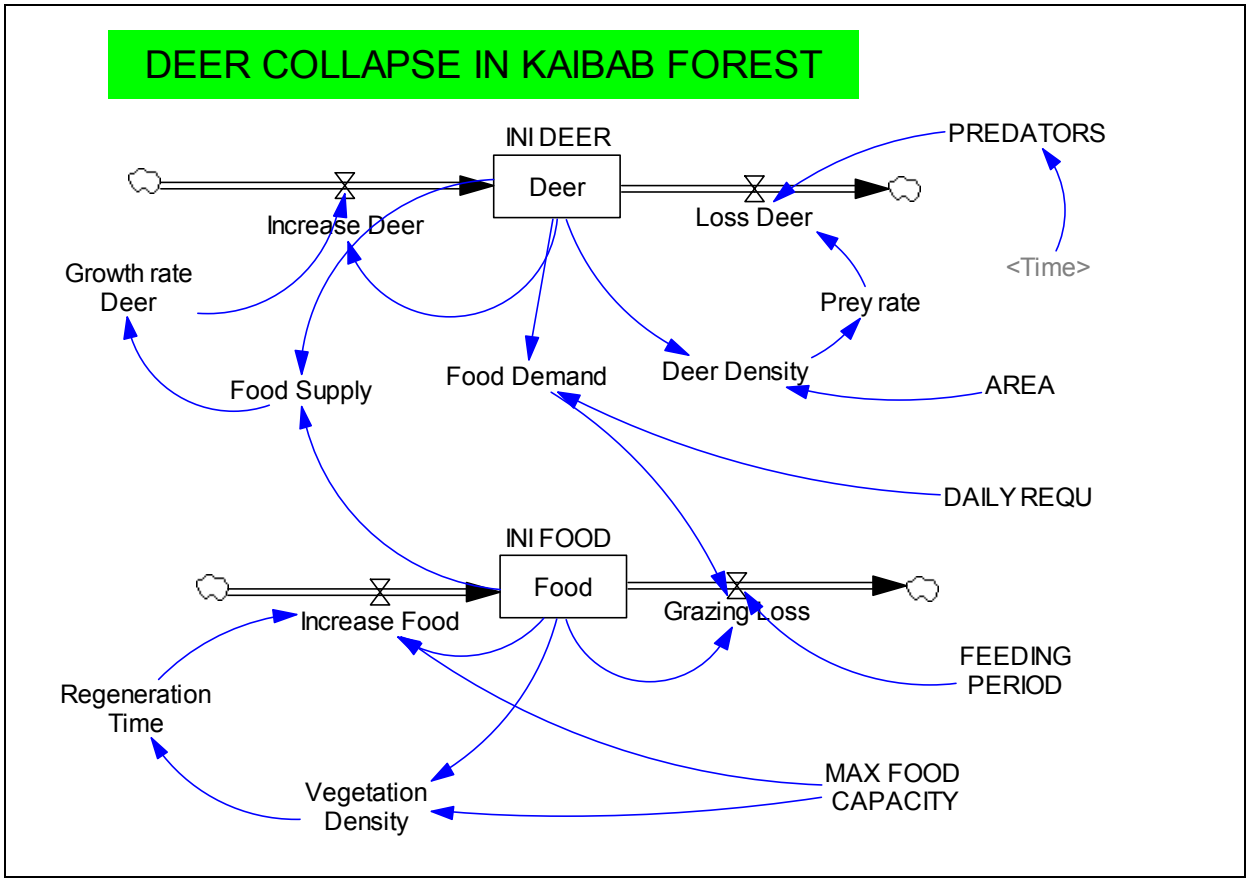
What about the predators? The *Deer* population suffers *Loss Deer* by the PREDATORS, whose number decreases linearly caused by shooting numbers. The *Prey rate* is a function of the *Deer Density* (= deer/ha).

A more detailed explanation of the parameters is given in *System Zoo*.

Especially interesting is the use of functional dependencies which are given by tables (= nodes of the describing functions). We will find these tables in the document under WITH LOOKUP.

The simulation is running for 50 years with an increment of 0.25 years.

Note the use of the IF-function with its syntax very similar to the syntax used in Computer Algebra Systems.



The document comprises all constants, all equations and all simulation parameters (originally presented numbered and in alphabetical order):

(01) AREA = 320000

(02) DAILY REQU = 2000

(03) Deer= INTEG (+Increase Deer – Loss Deer, INI DEER)

(04) Deer Density = Deer/AREA

(05) FEEDING PERIOD = 1

(06) FINAL TIME = 50

(07) Food = INTEG (+Increase Food – Grazing Loss, INI FOOD)

(08) Food Demand = DAILY REQU * Deer

(09) Food Supply = Food/Deer

(10) Grazing Loss =
IF THEN ELSE(Food Demand >= (Food/FEEDING PERIOD),
Food/FEEDING PERIOD, Food Demand)

(11) Growth rate Deer = WITH LOOKUP (Food Supply, ([(0,-1)-(10000,1)],
(0,-0.5), (500,-0.15),(1000,0),(1500,0.15),(2000,0.2) ,(200000,0.2)))

(12) Increase Deer = Growth rate Deer * Deer

(13) Increase Food = (MAX FOOD CAPACITY – Food)/Regeneration Time

(14) INI DEER = 4000

(15) INI FOOD = 4.7e+008

(16) INITIAL TIME = 0

(17) Loss Deer = Growth rate Pred * PREDATORS

(18) MAX FOOD CAPACITY = 4.8e+008

(19) PREDATORS = WITH LOOKUP (Time, ([(0,0)-(50,300)],
(0,265),(5,245),(10,200),(15,65),(20,8),(25,0),(30,0), (35,0),(40,0),(50,0)))

(20) Prey rate = WITH LOOKUP (Deer Density, ([(0,0)-(0.35,60)], (0,0),
(0.0125,3),(0.025,13),(0.0375,28),(0.05,51),(0.0625 ,56),(0.125,56),(0.4,56)))

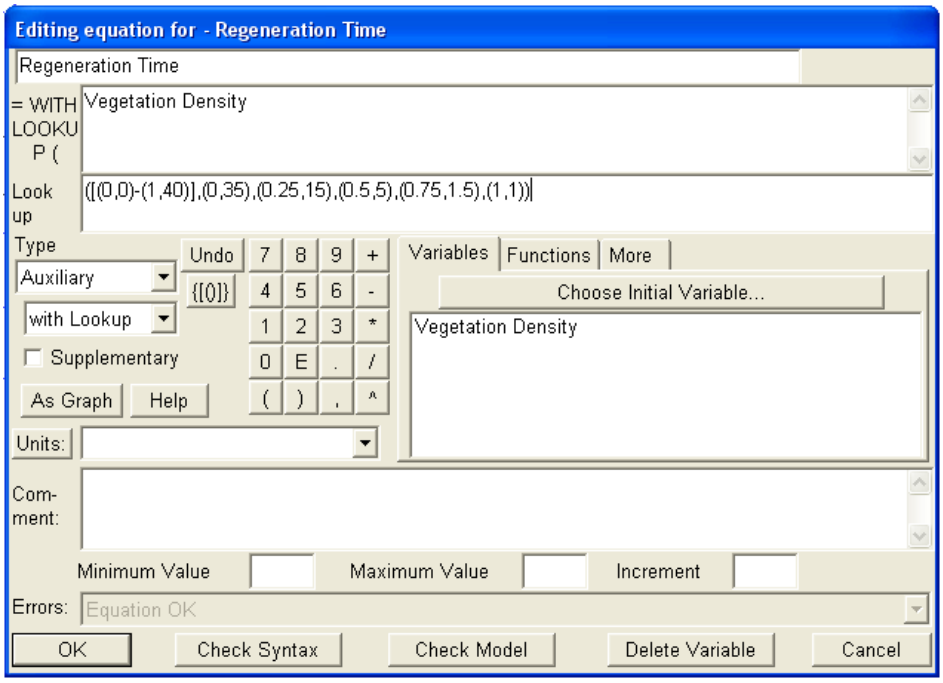
(21) Regeneration Time = WITH LOOKUP (Vegetation Density,([(0,0)-(1,40)],
(0,35),(0.25,15),(0.5,5),(0.75,1.5),(1,1)))

(22) SAVEPER = TIME STEP

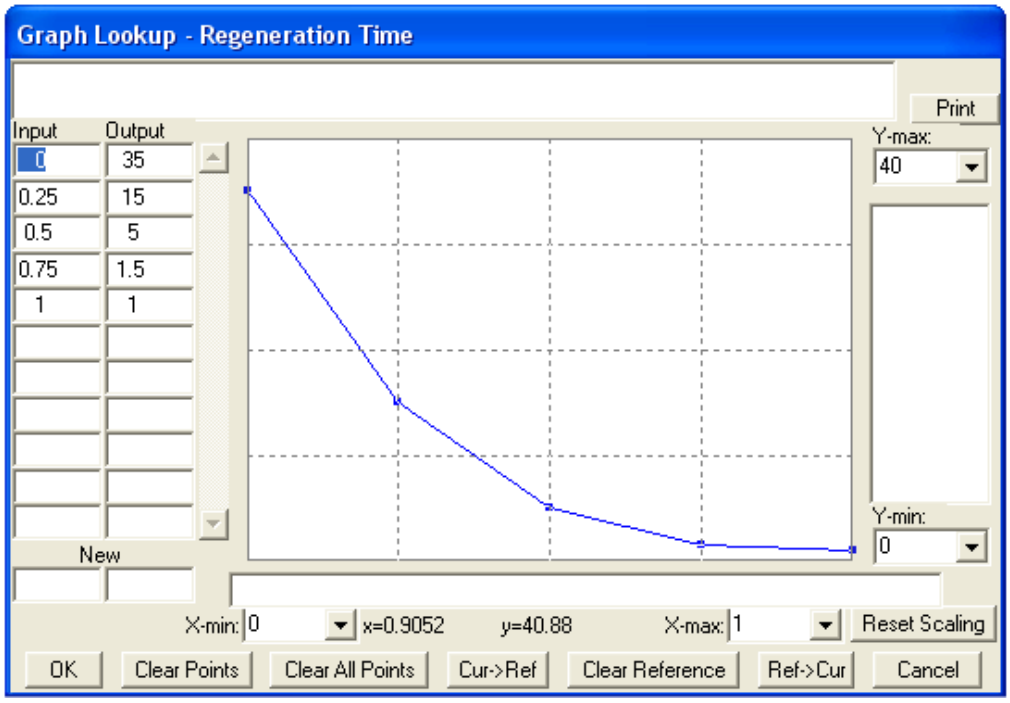
(23) TIME STEP = 0.25

(24) Vegetation Density = Food/MAX FOOD CAPACITY

Let's inspect function (21) *Regeneration Time (Vegetation Density)* as an example for working WITH LOOKUP:



After pressing the As Graph-button the graph of the piecewise defined function is presented:



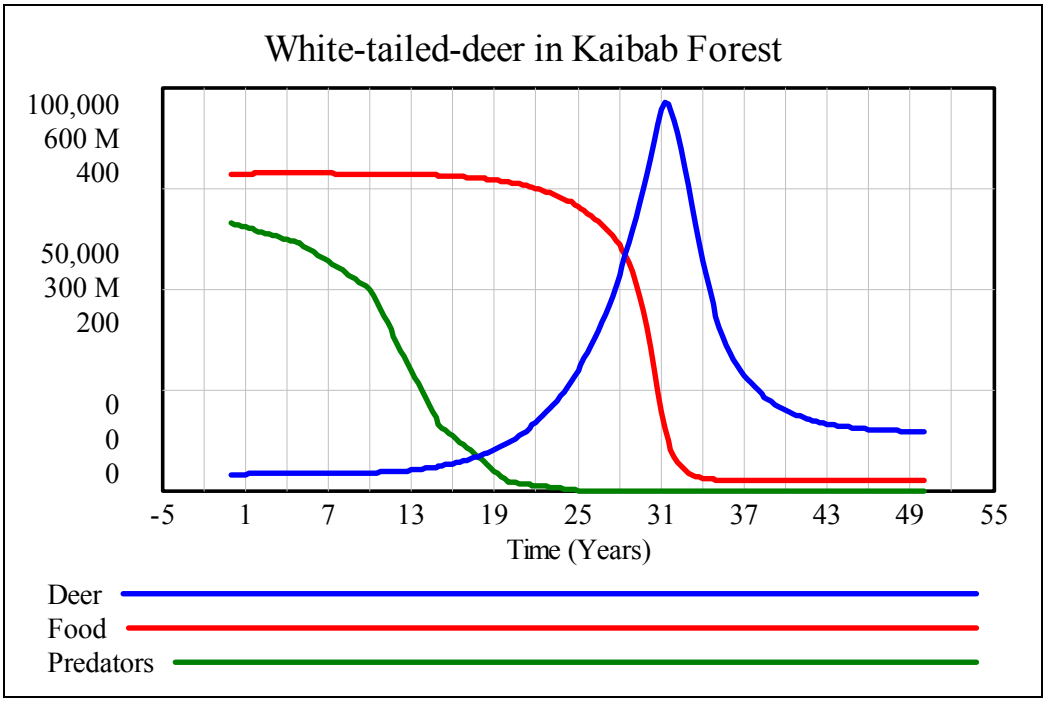
We would be able to enter the nodes directly into the grid. It is easy to recognize that there is a linear interpolation between the given points (nodes).

We run the simulation and inspect the first results.

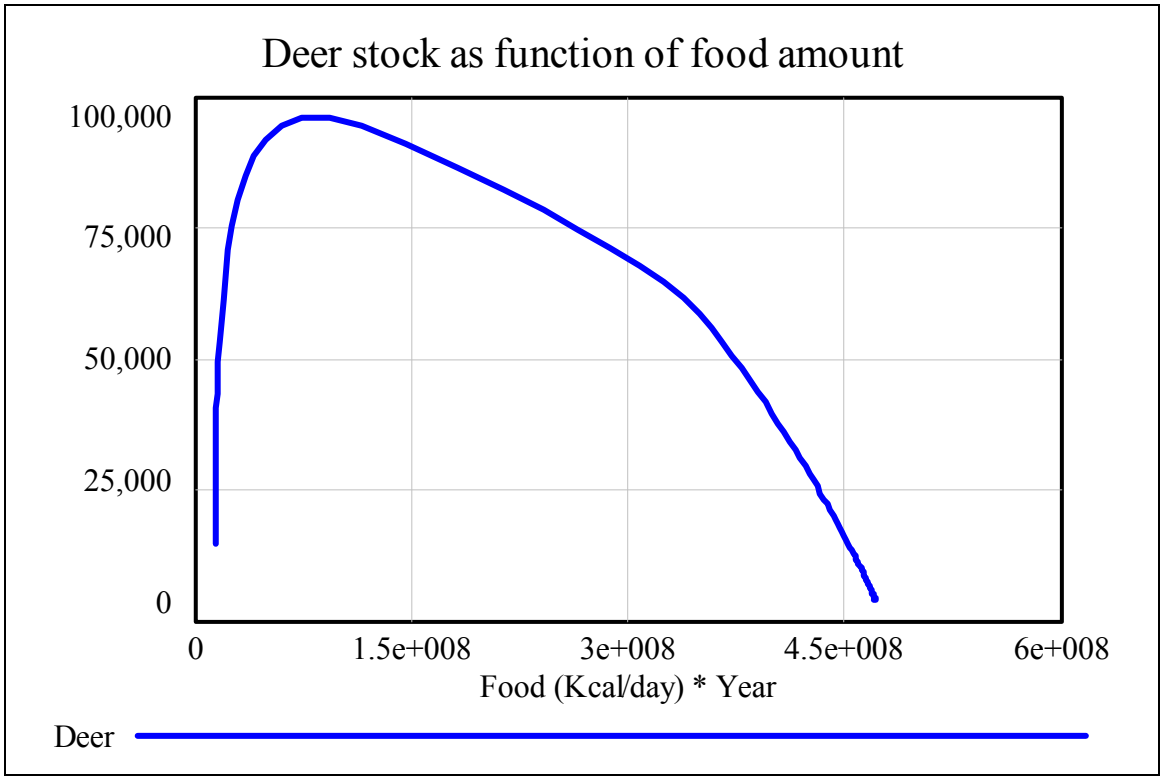
How are the deer doing?

How develops the available amount of food?

How are the predators doing? (Thanks human interaction – they are doing obviously badly!)



The second graph shows the stock of deer as function of the food amount.



Starting point is at right bottom and the development ends on the left hand side.

The result of the simulation matches with the real historical occurrence. The reduction of the vermin led to an explosion of the deer population which caused a disastrous overgrazing of the available food capacity. A huge number of deer died of hunger and finally the deer stock became stabilized on a level based on the much reduced amount of food.

As I am – unfortunately enough – no *Excel*-expert, I don't know how to realize the functional dependencies together with their connected linear interpolations in an easy way in a spreadsheet.

It would be great if any reader of these lines could accomplish this chapter performing the simulation with *Excel*. I would be very grateful for respective information.

I will come back to *MS-Excel* later.

But we can be glad having some other tools available to try with!

The *DERIVE*-Model

I accepted the challenge treating this system with *DERIVE*.

There appears the same problem: how to realize the piecewise defined functions with the linear interpolations?

As a DERIVIAN one has immediately the idea to connect the points given in a matrix using the CHI-function.

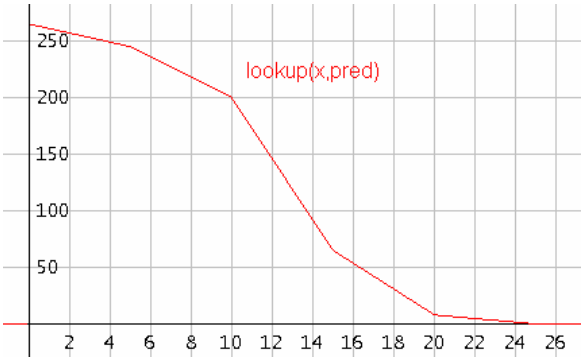
First of all the given data are fixed. Then my first attempt follows finding a function which is equivalent to the LOOKUP-function.

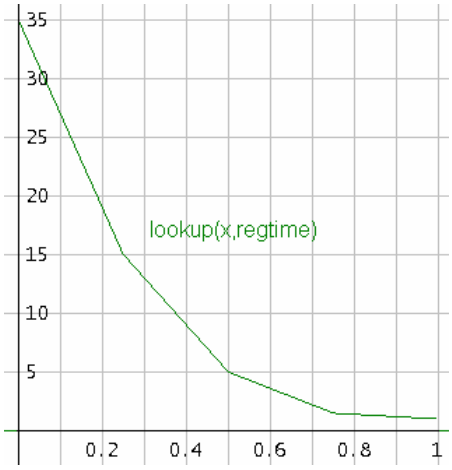
$$\left[\text{area} := 320000,\ \text{feedper} := 1,\ \text{max\_food\_cap} := 4.8\cdot 10^{8},\ \text{dayly\_requ} := 2000,\ \text{ini\_food} := 4.7\cdot 10^{8},\ \text{ini\_deer} := 4000\right]$$

$$\left[\text{pred} := \begin{bmatrix} 0 & 265 \\ 5 & 245 \\ 10 & 200 \\ 15 & 65 \\ 20 & 8 \\ 25 & 0 \\ 50 & 0 \end{bmatrix},\ \text{preyr} := \begin{bmatrix} 0 & 0 \\ 0.0125 & 3 \\ 0.025 & 13 \\ 0.0375 & 28 \\ 0.05 & 51 \\ 0.0625 & 56 \\ 0.125 & 56 \\ 0.4 & 56 \end{bmatrix},\ \text{regtime} := \begin{bmatrix} 0 & 35 \\ 0.25 & 15 \\ 0.5 & 5 \\ 0.75 & 1.5 \\ 1 & 1 \end{bmatrix},\ \text{gr\_deer} := \begin{bmatrix} 0 & -0.5 \\ 500 & -0.15 \\ 1000 & 0 \\ 1500 & 0.15 \\ 2000 & 0.2 \\ 200000 & 0.2 \end{bmatrix}\right]$$

$$\text{lookup}(x\_,\ pk) := \sum_{i=1}^{\text{DIM}(pk)-1} \chi(pk_{i,1},\ x\_,\ pk_{i+1,1}) \cdot \left(\frac{pk_{i+1,2} - pk_{i,2}}{pk_{i+1,1} - pk_{i,1}} \cdot (x\_ - pk_{i,1}) + pk_{i,2}\right)$$

The graphs are looking pretty nice. Compare the graph for the regeneration time with the respective *VENSIM*-graph (page 25)! On the first glance you will not recognize any difference.





But don't be happy too early! Inspecting the value tables (e.g. the numbers of the PREDATORS) we recognize the deficiency of the CHI-function in the nodes where the function is undefined. Hence this implementation is of no need for us.

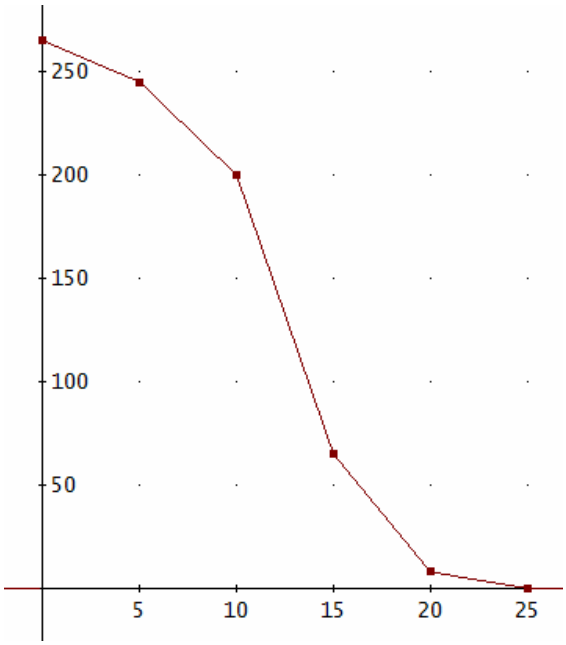
0	1	2	3	4	5	6
±132.5 + 132.5	261	257	253	249	?	236

24	25	26	27	28	29	30	31	32	33	34
1.6	0	0	0	0	0	0	0	0	0	0

The next function fulfils our requirements.

```
lu(x_, pk, f) :=
  Prog
    f := IF(pk↓1↓1 ≤ x_ ≤ pk↓2↓1, (pk↓2↓2 - pk↓1↓2)/(pk↓2↓1 - pk↓1↓1)·(x_ - pk↓1↓1) + pk↓1↓2, 0)
    pk := REST(pk)
    Loop
      If DIM(pk) = 1 exit
      f := f + IF(pk↓1↓1 < x_ ≤ pk↓2↓1, (pk↓2↓2 - pk↓1↓2)/(pk↓2↓1 - pk↓1↓1)·(x_ - pk↓1↓1) + pk↓1↓2, 0)
      pk := REST(pk)
    LIM(f, x, x_)
```

The graphs fit exactly (nodes and segments) and the value tables don't show any exceptions.



TABLE(lu(x, pred), x, 0, 50)'

0	1	2	3	4	5	6	7
0	261	257	253	249	245	236	227
24	25	26	27	28	29	30	31	32
1.6	0	0	0	0	0	0	0	0

Before programming I always tried to work through the system(s) acting as a "human spreadsheet". I'd like to recommend this way treating such systems in classroom. Then the interconnections become clear and programming and/or transfer to a "real" spreadsheet becomes very easy.

Here I benefit from the results of the *VENSIM* simulation because I can use its tables as a reference for programs and/or any other treatments.

I try demonstrating the manual procedure – supported by the *DERIVE* made lu-functions – step by step.

The table consists of 16 columns.

In row 1 I start with Time = 0, FOOD = $4{,}7 \cdot 10^8$, DEER = 4000 and PREDAT = 265. The numbers in the last row indicate the order of calculation.

I go on with the entries for *Food Demand*, proceed with *Browsing Loss* and close the line with *Increase Deer*. Then we enter in row 2 (Time = 0.25) the new amount of FOOD and the new DEER population (13, 14) followed by 1 through 12. We can follow the formulae (equations) how they are listed in the document.

Row nr	Time	FoodDem	BrowsLoss	VegDens	RegTime	FoodIncr	FOOD
1	0	$8{\cdot}10^6$	$8{\cdot}10^6$	0.979167	1.041666	$9.6{\cdot}10^6$	$4.7{\cdot}10^8$ (*)
2	0.25	$8.0025{\cdot}10^6$	$8.0025{\cdot}10^6$	0.98	1.04	$9.23077{\cdot}10^6$	$4.704{\cdot}10^8$
3	0.50						$4.70707{\cdot}10^8$
		①	②	③	⑥	⑦	①③

PREDAT	DeerDens	PreyRate	Loss Deer	Food Sup	GrRateDe	IncrDeer	DEER
265	0.0125	3	795	117500	0.2	800	4000
264	0.0125039	3.00312	792.824	117563	0.2	800.25	4001.25 (*)
							4003.11
⑤	④	⑧	⑨	⑩	①①	①②	①④

(*) Calculating the increases (for FOOD and DEER) one has to consider the time increment *dx*.
So for DEER(Time = 0.25) = 4000 + (800 – 795) · 0.25 = 4001.25.

The values in the columns for *Regeneration Time* (RegTime), *Predators* (PREDAT), PreyRate and *Growth rate Deer* (GrRateDe) were found using the `lu`-function (in analogy to WITH LOOKUP).

```
lu(0.979167, nachwzt) = 1.041666        lu(0.98, nachwzt) = 1.04
lu(0.0125, beuterate) = 3               lu(0.0125039, beuterate) = 3.00312
lu(117500, zuw_r_h) = 0.2               lu(0.25, raeub) = 264
```

The table from above can be transferred one by one into a *DERIVE*-program.

With *DERIVE* I collect all values in a table, too. For plotting the diagrams I have to select the respective columns.

First of all is a short function needed for the grazing or *Browsing Loss*.

```
graz_loss(x, y) :=
  If x ≥ y/feedper
     y/feedper
     x
```

The `lu`-function was introduced earlier. The full program is following.

```
kaibab(n, dx, i, tab, t, f_dem, brows_loss, veg_d, deer_d, predators,
       reg_time, food_inc, prey_r, inc_deer, food_supply, loss_deer,
       deer, food) :=
PROG(
   i := 1,
   tab := [["RNr", "Time", "FDem", "GrazL", "VegD", "RegTime",
        "FoodIncr", "PreyR", "DeerLoss", "FSupply", "DeerIncr", "Food",
        "Deer", "Predators"]],
     t := 0, [deer := ini_deer, food := ini_food],
     LOOP(
          IF(i > n, RETURN tab),
          f_dem := deer·daily_requ,
          brows_loss := graz_loss(f_dem, food),
          veg_d := food/max_food_cap,
          deer_d := deer/area,
          predators := lu(t, pred),
          reg_time := lu(veg_d, regtime),
          food_inc := (max_food_cap - food)/reg_time,
          prey_r := lu(deer_d, predr),
          loss_deer := prey_r·predators,
          food_supply := food/deer,
          inc_deer := lu(food_supply, gr_deer)·deer,
          tab := APPEND(tab, [[i, t, f_dem, brows_loss, veg_d, reg_time,
                food_inc, prey_r, loss_deer, food_supply, inc_deer, food,
                deer, predators]]),
          deer := deer + (inc_deer - loss_deer)·dx,
          food := food + (food_inc - brows_loss)·dx,
          t :+ dx, i :+ 1))
```

The first 4 rows are – very enjoyable – completely corresponding with the manually calculated table and the *VENSIM*-results as well.

```
kaibab(4, 0.25)
```

RNr	Time	FDem	GrazL	VegD	RegTime	FoodIncr	PreyR
1	0	8000000	8000000	0.9791666666	1.041666666	9600000	3
2	0.25	8002500	8002500	0.98	1.04	9230769.23	3.003125
3	0.5	8006212.5	8006212.5	0.9806397235	1.038720552	8946518.547	3.007765625
4	0.75	8011001.945	8011001.945	0.9811294662	1.037741067	8728435.7	3.013752431

DeerLoss	FSupply	DeerIncr	Food	Deer	Predators
795	117500	800	470000000	4000	265
792.825	117563.2614	800.25	470400000	4001.25	264
791.0423593	117585.4543	800.62125	470707067.3	4003.10625	263
789.603137	117573.8433	801.1001945	470942143.8	4005.500972	262

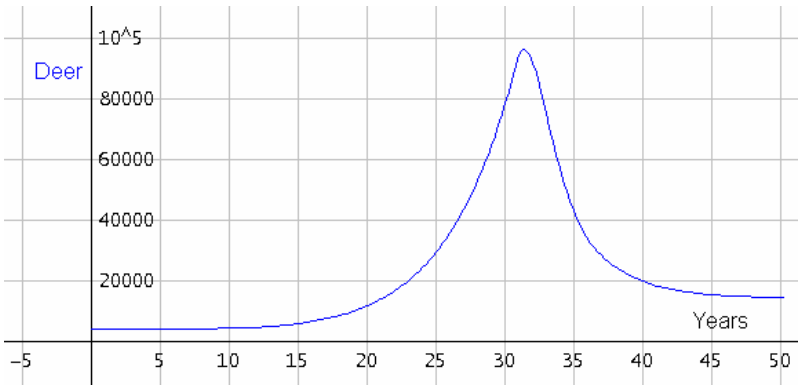
These are the values for FOOD and DEER for the last three quarters of a year.

```
(kaibab(202, 0.25))              ↓↓[2, 12, 13]
                  [200, 201, 202]
```

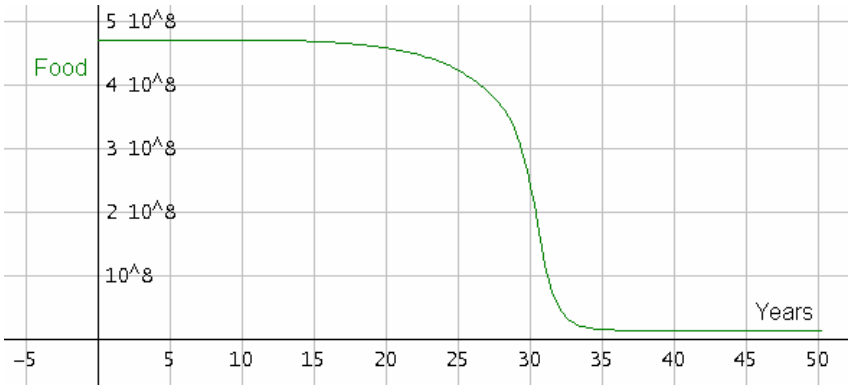
49.5	14277004.0601	14566.1776742
49.75	14277004.0192	14544.4896531
50	14277003.988	14524.4282306

Please compare the *DERIVE*-diagrams with the plots generated with *VENSIM* (pages 25, 26)

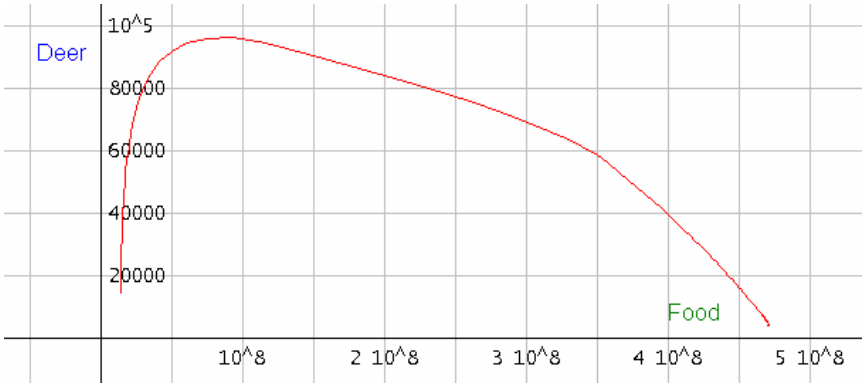
```
DELETE((kaibab(202, 0.25))↓↓[2, 13], 1)
```




```
DELETE((kaibab(202, 0.25))↓↓[2, 12], 1)
```




```
DELETE((kaibab(202, 0.25))↓↓[12, 13], 1)
```



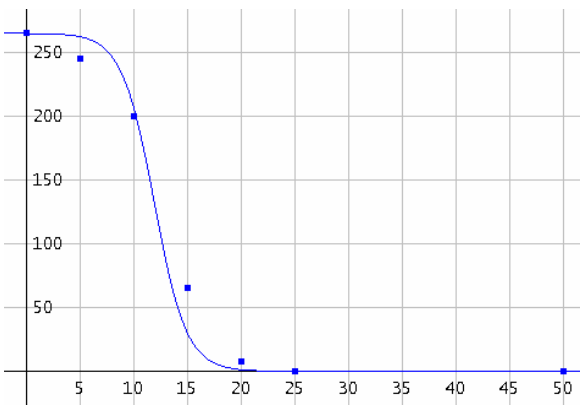
I promise to try modelling the system by using differential equations later.

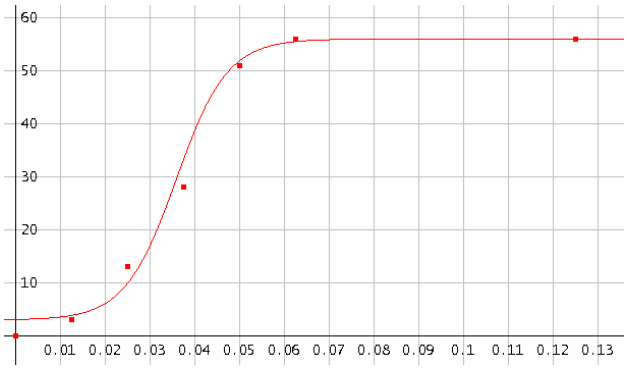
For treating this problem with a spreadsheet program it would be useful to approximate the WITH LOOKUP functions for PREDATORS, *Prey rate*, *Regeneration Time* and *Growth rate Deer* by an „ordinary“ function.

This is a nice task for its own. Sliders and meaningful considerations lead to appropriate functions.

$$pred\_f(x) := \frac{701985}{e^{0.6625 \cdot x} + 2649}$$

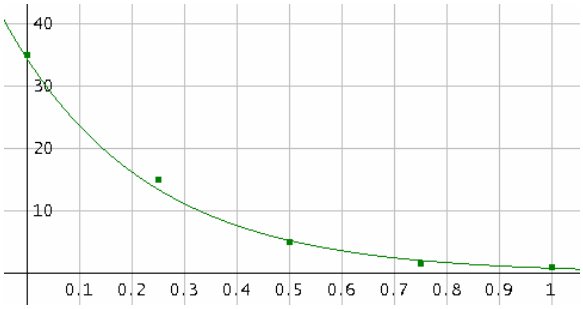
$$preyr\_f(x) := 3 + \frac{53}{1 + 529 \cdot e^{-174.9 \cdot x}}$$

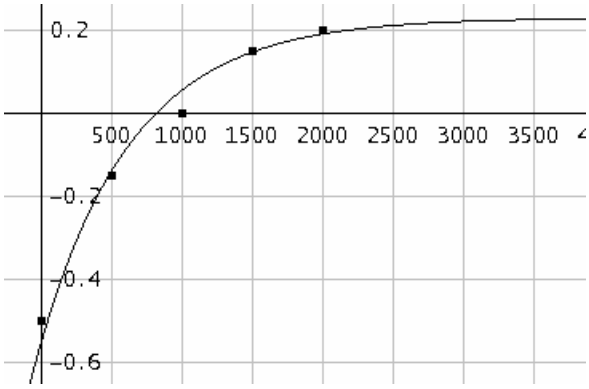




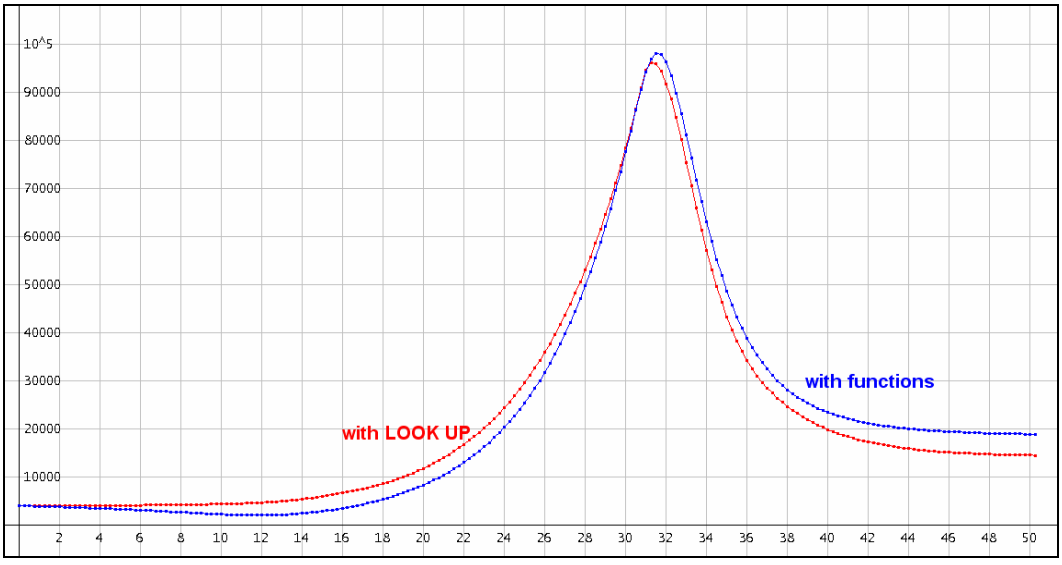
$$regtime\_f(x) := 34.4089 \cdot e^{-3.7653 \cdot x}$$

$$gr\_deer\_f(x) := 0.23 - 0.78 \cdot e^{-0.0015 \cdot x}$$





How good are these approximations? We can check this by replacing the lu-functions in program `kaibab` gaining program `kaibab_f`. Then we will compare the graphs of the deer population derived from both programs.

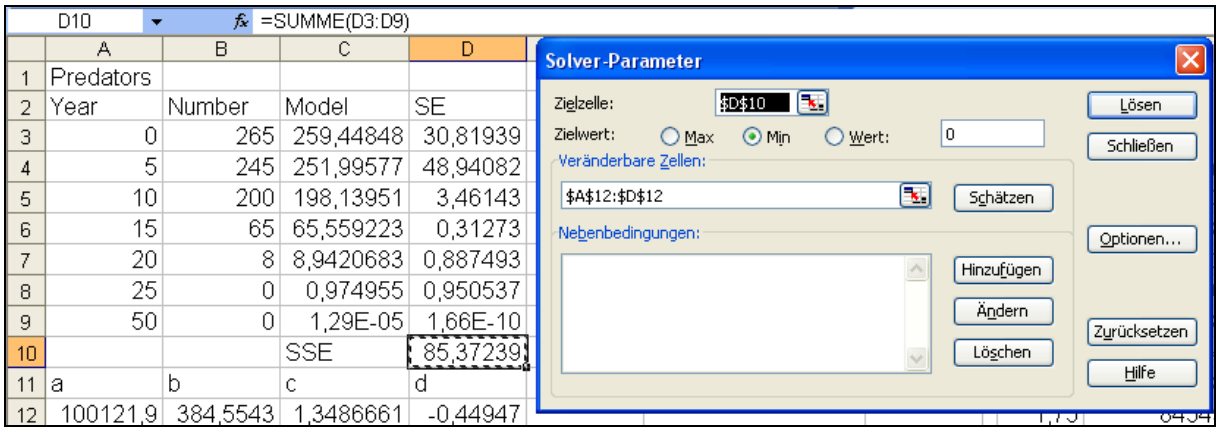


The result is impressive, isn't it? The graphs for the food are almost identical, too.

These functions make modelling with spreadsheet much more comfortable. Now let's try *MS-Excel*!

The *MS-Excel*-model

We could take over the functions from *DERIVE* but there is a "SOLVER" available in the spreadsheet program which is a very versatile tool. We need some "inspiration" from the form of the scatter diagrams in order to make the right decision for the type of function which we should choose for the approximation.



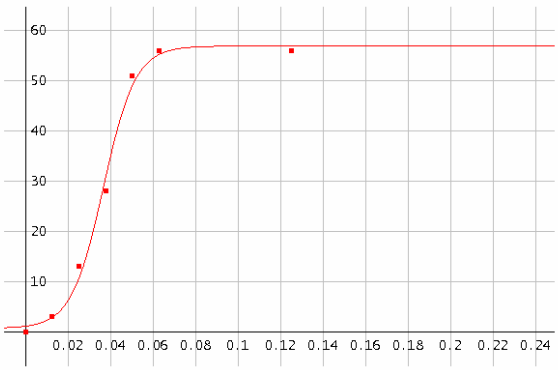
	A	B	C	D
1	Predators			
2	Year	Number	Model	SE
3	0	265	259,44848	30,81939
4	5	245	251,99577	48,94082
5	10	200	198,13951	3,46143
6	15	65	65,559223	0,31273
7	20	8	8,9420683	0,887493
8	25	0	0,974955	0,950537
9	50	0	1,29E-05	1,66E-10
10			SSE	85,37239
11	a	b	c	d
12	100121,9	384,5543	1,3486661	-0,44947

"Inspired" by *DERIVE* I choose for the predator function the form $\frac{a}{b+c\cdot e^{-d\cdot x}}$ and enter in cell C3 as follows: =$A$12/($B$12+$C$12*EXP(-$D$12*A3)).

We enter initial values for the solving (= iteration-) procedure in cells A12 to D12 – and this is the trickier part of the task. However, here we can refer to earlier results again. I found approximating the growth rate of the deer the most difficult.

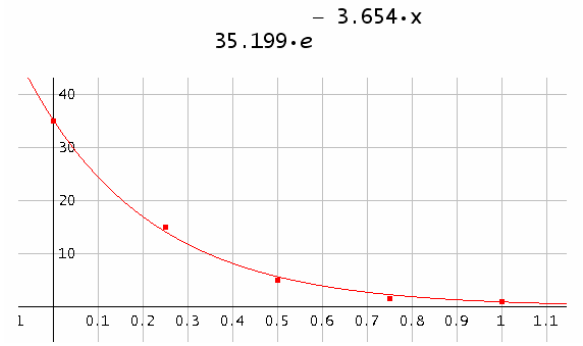
In the SE-column are the squared errors of the model values with respect to the real values. Cell D10 contains the sum of the squared errors which should become (absolutely) minimized (= 0).

$$0.733 + \frac{317.477}{5.654 + 731.755\cdot e^{-132.95\cdot x}}$$

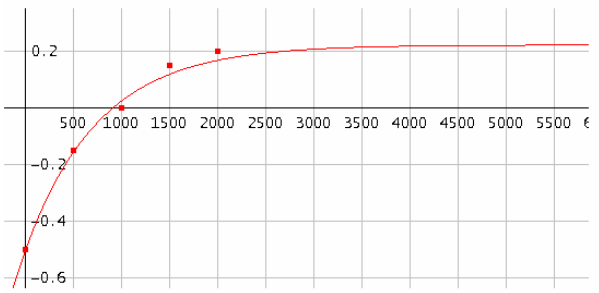


Now we see that the SOLVER delivers obviously better approximating functions than we had found earlier. It doesn't need some calculation to get this insight, just compare the graphs with the graphs on page 32!

$$35.199\cdot e^{-3.654\cdot x}$$

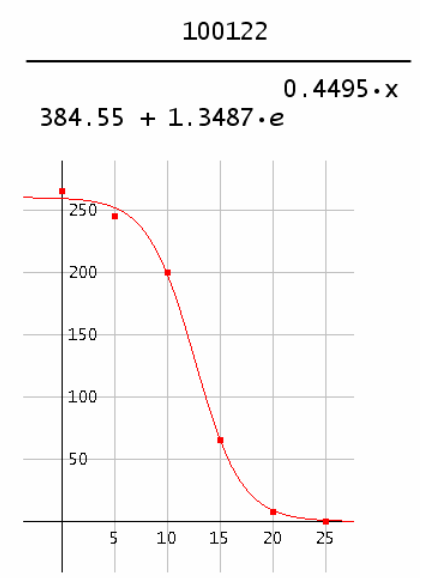


$$0.222 - 0.726\cdot e^{-0.0013\cdot x}$$



Using these functions and according to the strategy of page 30 one can fill in the *Excel* worksheet.

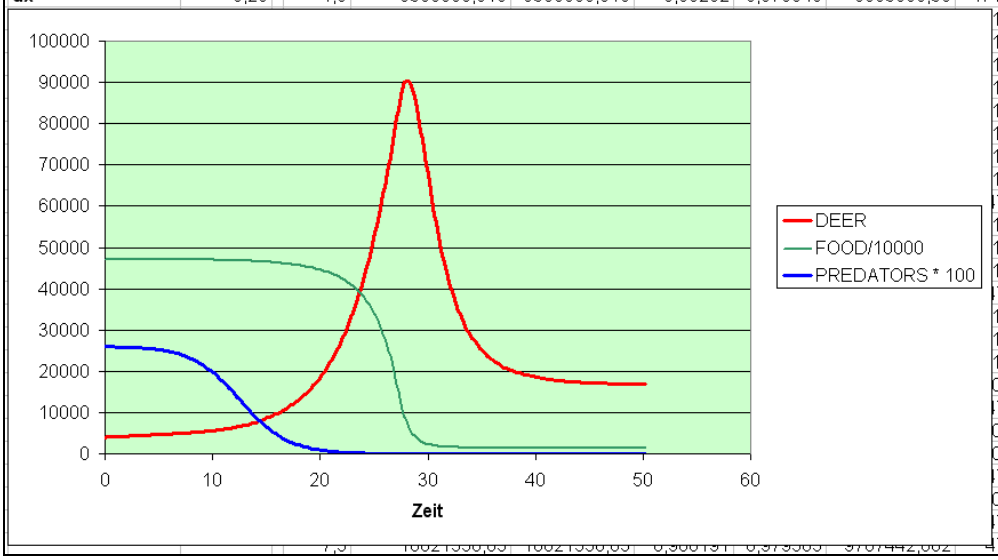
It is no problem to take a time increment of 0.25 years. Calculation is very fast.



The peak of the deer population is a little bit shifted but the message of the graph is quite the same as before.

See here a part of the worksheet together with the respective diagram.

Parameters:									
		Time	Food Demand	Grazing L	Veg Dens	Reg Time	Food Incr	FOOD	PREDATORS
Area	320000	0	8000000	8000000	0,979167	0,983177	10171106,17	470000000	259,4484788
Feeding Period	1	0,25	8064887,293	8064887,293	0,980297	0,979123	9658870,723	470542776,5	259,3406929
MaxFoodCap	4,8E+08	0,5	8129828,993	8129828,993	0,981128	0,976157	9279987,698	470941272,4	259,2201949
Ini-Deer	4000	0,75	8194800,688	8194800,688	0,981727	0,974023	9005115,196	471228812,1	259,0854996
Ini-Food	4,7E+08	1	8259780,262	8259780,262	0,982149	0,972522	8810711,041	471431390,7	258,9349519
Dayly Requirement	2000	1,25	8324748,316	8324748,316	0,982436	0,971503	8678181,298	471569123,4	258,7667078
dx	0,25	1,5	8389688,618	8389688,618	0,98262	0,970849	8593009,36	471657481,6	258,5787148
								1708311,8	258,3686886
								1730654,3	258,1340896
								1731390,8	257,8720959
								1715749,7	257,579575
								1687697,4	257,2530525
								1650238,2	256,8886789
								1605644,8	256,4821947
								1555632,4	256,0288926
								71501492	255,5235782
								1444191,2	254,9605298
								1384449,6	254,3334565
								1322796,8	253,6354564
								71259615	252,8589761
								1195171,4	251,9957713
								1129642,2	251,0368717
								1063130,6	249,9725516
								0995679,7	248,7923075
								70927282	247,484848
								0857886,1	246,0380972
								0787401,2	244,4392162
								70715700	242,6746487
								0642620,6	240,7301929
								70567967	238,5911079
								470491509	236,2422586



Bossel poses an interesting question and task:

> *What would have been an appropriate shooting strategy (for the predators) to achieve a stable deer population without causing the collapse of the grazing capacity?*

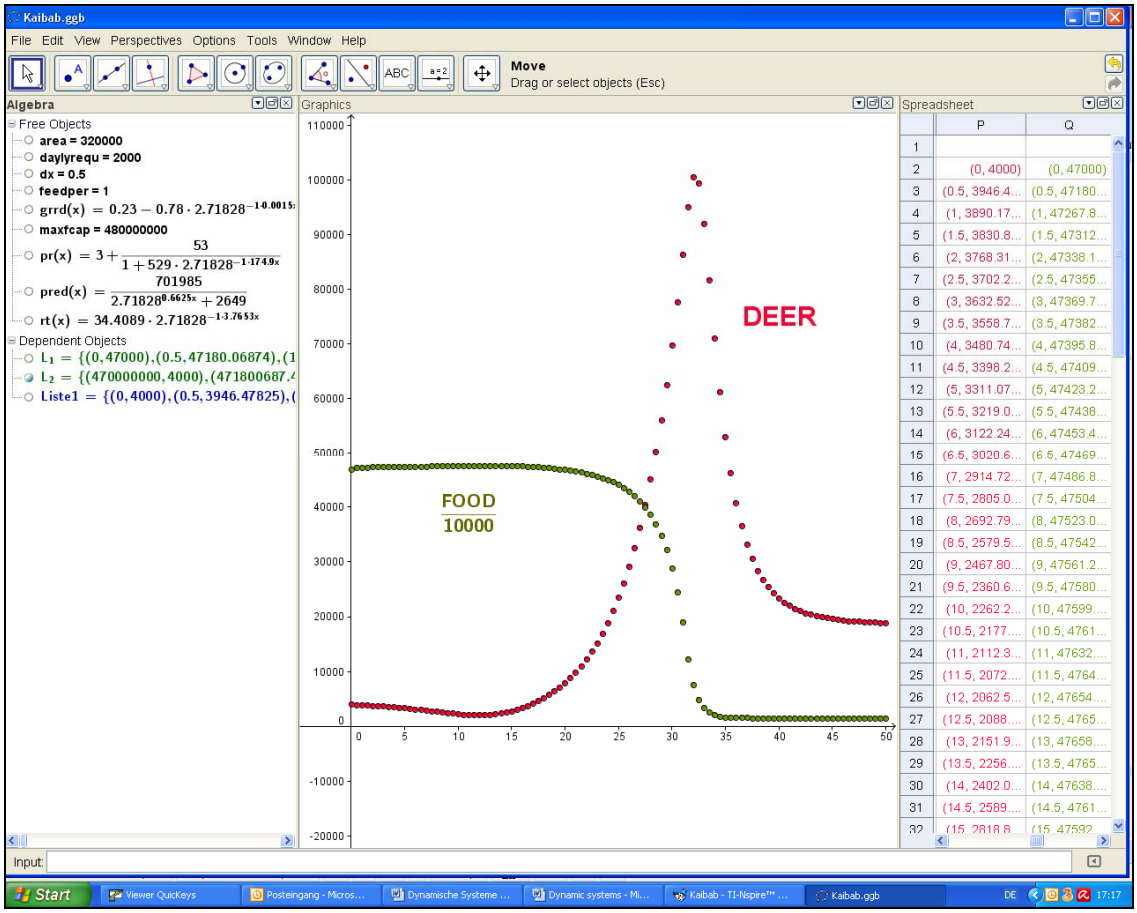
For answering a question like this application of sliders seems to be best suitable. *GeoGebra* and *TI-Nspire* (and *MS Excel*, of course) provide this valuable tool. *DERIVE* does also but we explained earlier why we cannot use the sliders with *DERIVE* in problems like this.

The *GeoGebra* Model

We use again the *DERIVE*-made approximating functions. The first model is set up with the given predator function in order to check whether the results are the expected ones.

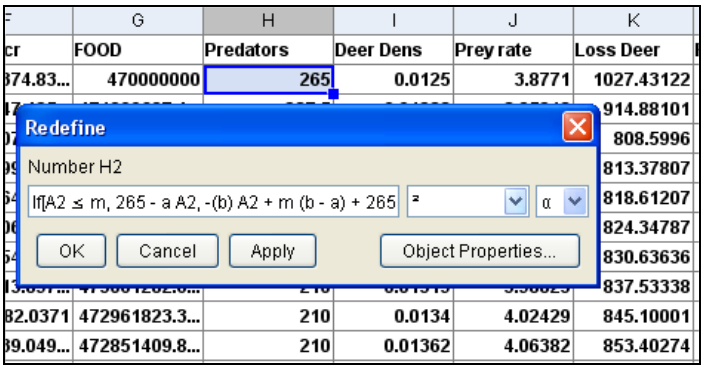
The next screen shot is a copy of the *GeoGebra*-screen with the diagram of the deer population and the scaled food stock (FOOD/10000).

As the *GeoGebra*-spreadsheet needs long calculation times I increased the time step up to 0.5 which does not cause essential changes of the results as the diagram is showing.



Let's try to find an answer for *Bossel*'s question. After some – exciting – attempts I decided to introduce the following shooting strategy.

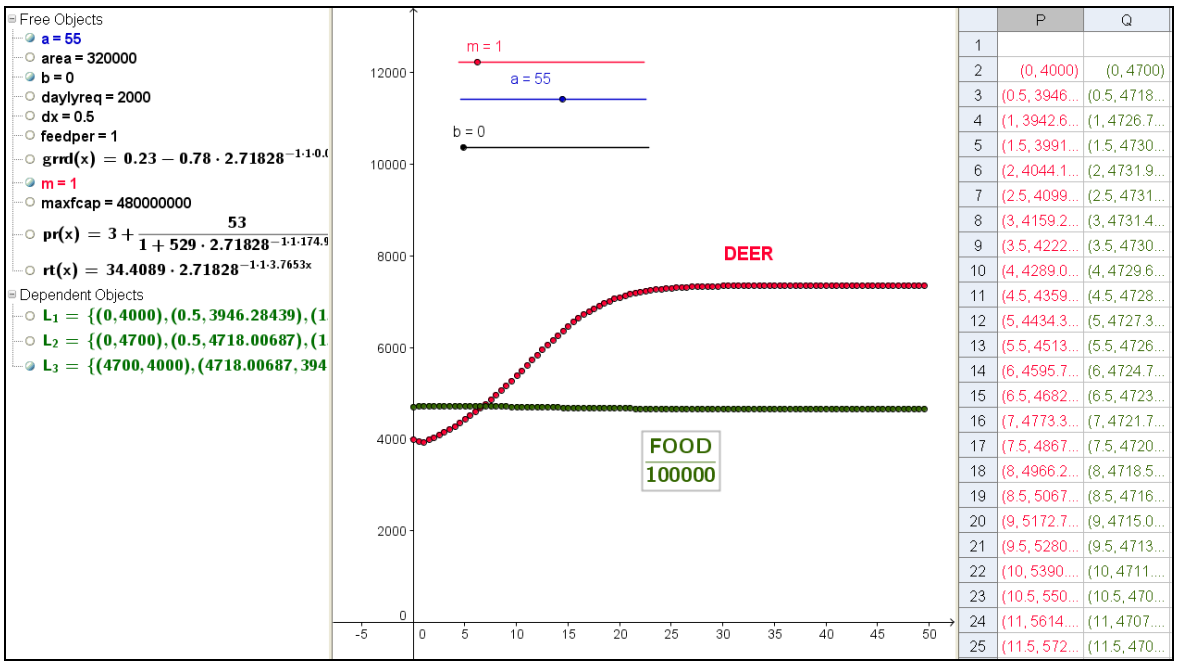
I will have a radical shooting of *a* animals annually for the first *m* years followed by reduction to *b* beasts per year. The respective "predator function" is entered in cell H2 (with the corresponding time in cell A2). There is nothing else to change in the spreadsheet from above.



cr	G FOOD	H Predators	I Deer Dens	J Prey rate	K Loss Deer
374.83...	470000000	265	0.0125	3.8771	1027.43122
					914.88101
					808.5996
					813.37807
					818.61207
					824.34787
					830.63636
					837.53338
82.0371	472961823.3...	210	0.0134	4.02429	845.10001
89.049...	472851409.8...	210	0.01362	4.06382	853.40274

Calculation of the first complete table needs some time but then the diagram is reacting immediately on the change of the parameters by moving the sliders.

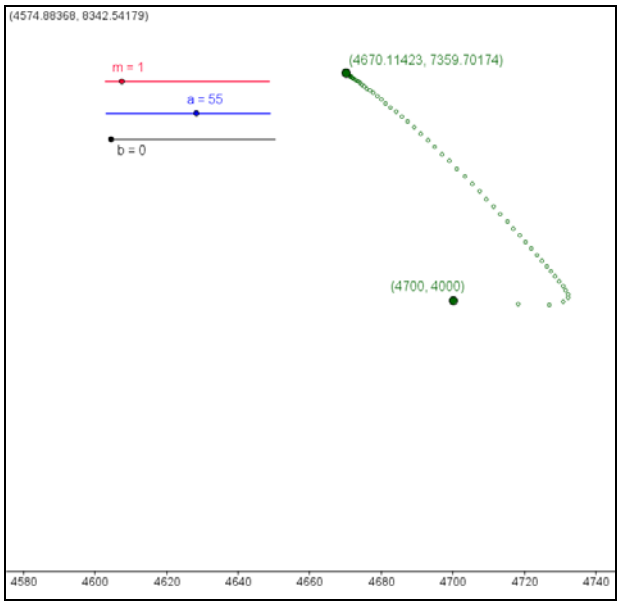
Starting with shooting 55 animals in the first year we can then keep the predator population on a level of 210 in order to achieve constant food supply for the deer. The number of deer stays stable with a stock of about 7360.



The phase diagram Food-Deer shows a significant convergence, too.

But this is not the only one possibility to obtain a stable high deer population.

It makes fun to experiment to reach a more or less stable deer population on a lower or higher level.



You can also introduce a moderate constant shooting rate or any combination. Here we have only an “exogenous” regulation of the predators. But it would also be possible to consider “endogenous“ factors like natural dying rate etc and include this in the simulation.

Good Sport!

Working with *TI-NspireCAS*

In the beginning I had some troubles with the spreadsheet application but by and by it worked pretty well finally. The diagram looks the same as the *GeoGebra* graph. Calculation of the table works much faster which makes smaller time increments possible.

Transfer of the *DERIVE*-program into the *TI-NspireCAS*-language is an easy task.

luf(x,pk) is the table function corresponding with the lu-function in *DERIVE*:

```
Define luf(x,pk)=
Func
:Local f
:f:=when(pk[1,1]≤x_<pk[2,1],((pk[2,2]−pk[1,2])/(pk[2,1]−pk[1,1]))*(x_−pk[1,1])+pk[1,2],0)
:pk:=subMat(pk,2,1,dim(pk)[1],2)
:While dim(pk)[1]>1
:f:=f+when(pk[1,1]<x_≤pk[2,1],((pk[2,2]−pk[1,2])/(pk[2,1]−pk[1,1]))*(x_−pk[1,1])+pk[1,2],0)
:pk:=subMat(pk,2,1,dim(pk)[1],2)
:EndWhile
:f|x_=x
:EndFunc
```

See the program which provides the respective lists which are the necessary base for the graphic representations.

```
Define kaibab(n,dx)=
Prgm
:Local i,t,deer,food,f_dem,brows_loss,veg_d,deer
:Local predators,reg_time,food_inc,prey_r
:Local loss_deer,food_supply,inc_deer
:i:=1: t:=0
:deer:=ini_deer:food:=ini_food
:ld:={deer}:lf:={food}:ltime:={t}
:While i≤n
:  f_dem:=deer*daily_requ
:  brows_loss:=when(f_dem≥((food)/(feedper)),((food)/(feedper)),f_dem)
:  veg_d:=((food)/(max_food_cap))
:  deer_d:=((deer)/(area))*1.
:  predators:=luf(t,pred)
:  reg_time:=luf(veg_d,regtime)
```

```
:  food_inc:=((max_food_cap−food)/(reg_time))
:  prey_r:=luf(deer_d,preyr)
:  loss_deer:=prey_r*predators
:   food_supply:=((food)/(deer))
:  inc_deer:=luf(food_supply,gr_deer)*deer
:  deer:=deer+(inc_deer−loss_deer)*dx
:  food:=food+(food_inc−brows_loss)*dx
:  t:=t+dx
:  ld:=augment(ld,{deer})
:  lf:=augment(lf,{food})
:  ltime:=augment(ltime,{t})
:  i:=i+1
:EndWhile
:Disp "Deer in ld, scaled Food in lfs, Time in ltime"
:EndPrgm
```

The Calculator-application contains the data and the program call.

$$regtime:=\begin{bmatrix}0 & 35\\ 0.25 & 15\\ 0.5 & 5\\ 0.75 & 1.5\\ 1 & 1\end{bmatrix} \qquad \begin{bmatrix}0.000000 & 35.000000\\ 0.250000 & 15.000000\\ 0.500000 & 5.000000\\ 0.750000 & 1.500000\\ 1.000000 & 1.000000\end{bmatrix}$$

$$gr\_deer:=\begin{bmatrix}0 & -0.5\\ 500 & -0.15\\ 1000 & 0\\ 1500 & 0.15\\ 2000 & 0.2\\ 200000 & 0.2\end{bmatrix} \qquad \begin{bmatrix}0.000000 & -0.500000\\ 500.000000 & -0.150000\\ 1000.000000 & 0.000000\\ 1500.000000 & 0.150000\\ 2000.000000 & 0.200000\\ 200000.000000 & 0.200000\end{bmatrix}$$

$area:=320000:feedper:=1:max\_food\_cap:=4.8\cdot 10^{8}$ 480000000.000

$daily\_requ:=2000:ini\_food:=4.7\cdot 10^{8}:ini\_deer:=4000$ 4000.000000

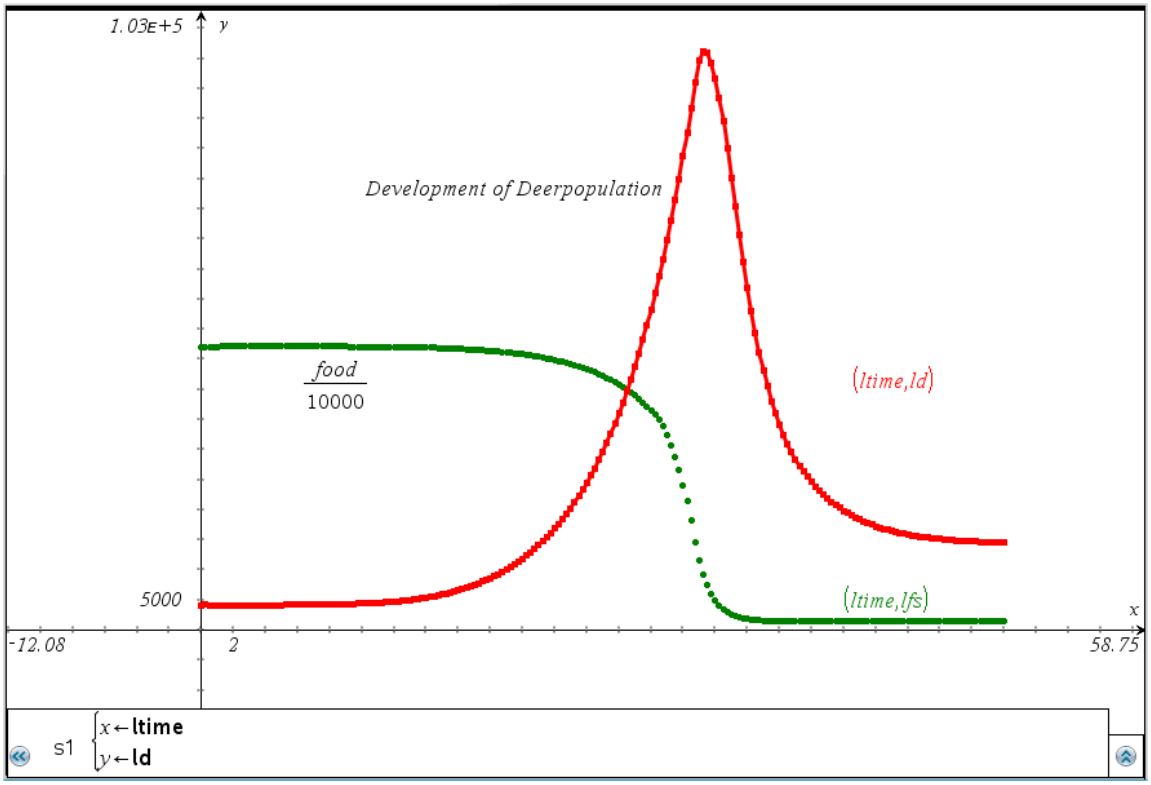
$kaibab\_var(200,0.25)$

Time in ltime, Deer in ldv, Food in lfv_scal, Predators in lprv_scal

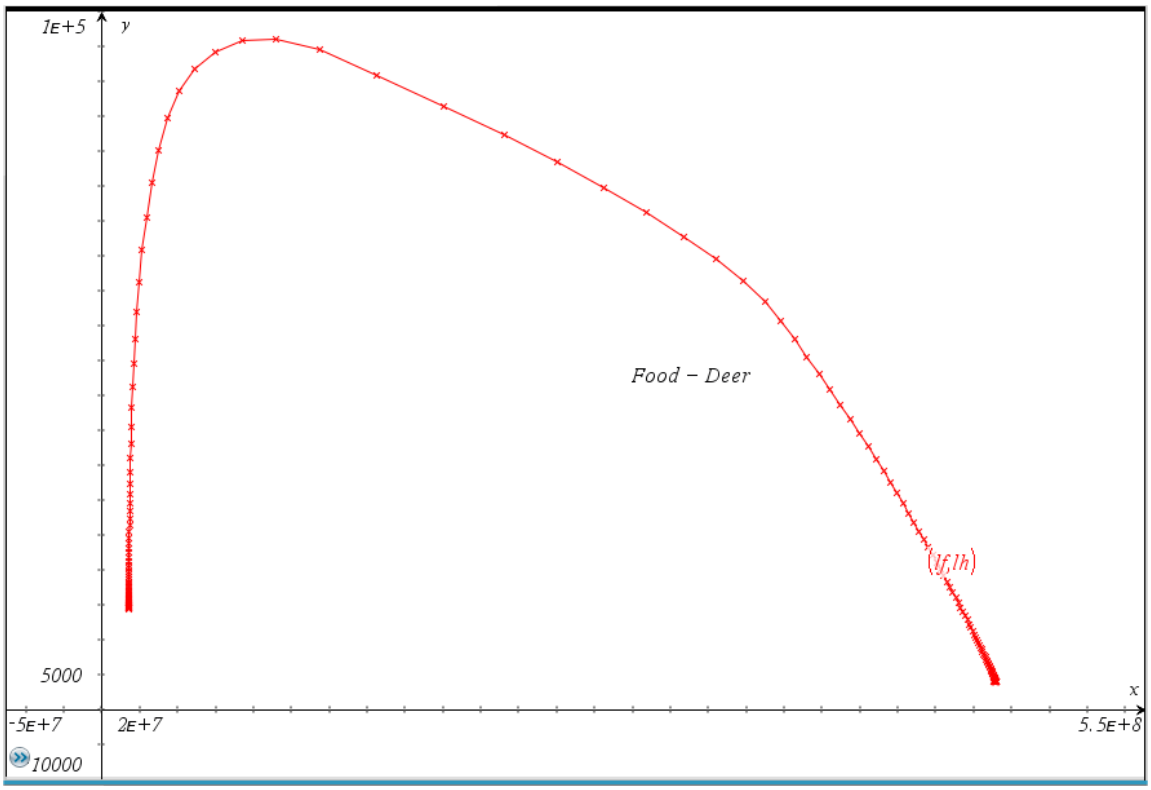
Done

Lists ltime, ld and lfs are the base for the scatter diagrams in the Graphs & Geometry application.

The next screenshot shows the already known development of the deer population together with a representation of the food available in a suitable scaling.



We have seen the phase diagram, too, produced with other tools.

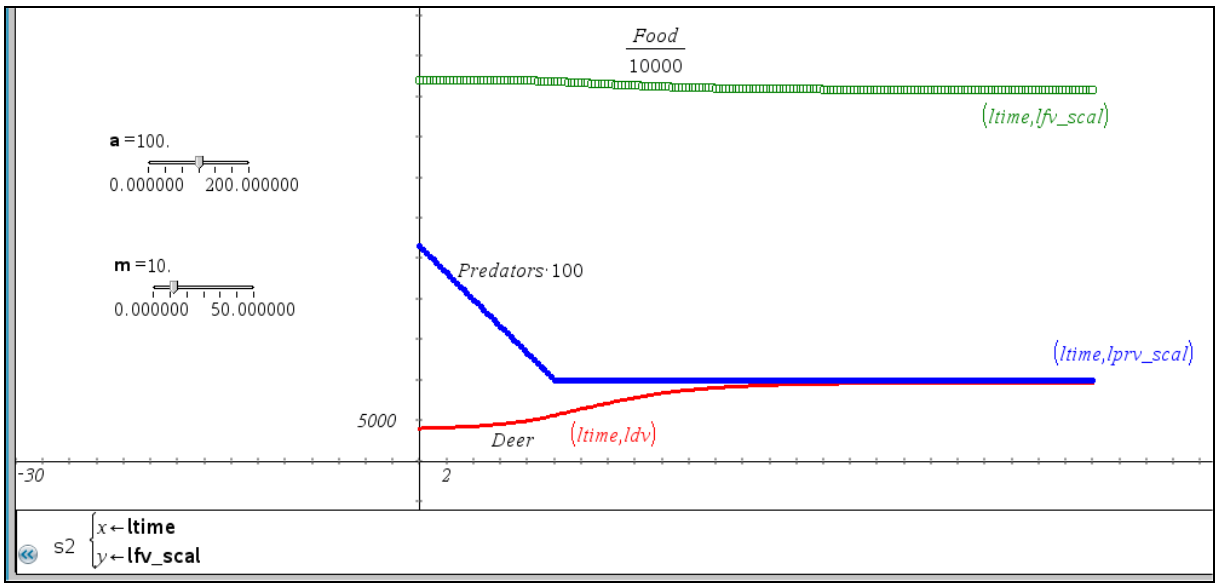


I promised to use the sliders with *TI-Nspire*. So, let us try!

I change the definition of the shooting strategy a little bit. I will keep the shooting numbers constant for the first *m* years until the predator population reaches a certain given number *a*. This number shall be kept stable. I will stickto my “philosophy” and introduce sliders for *m* and *a*. See a part of the program.

```
While i≤n
 f_dem:=deer·daily_requ
 brows_loss:=when(f_dem≥food/feedper, food/feedper, f_dem)
 veg_d:=food/max_food_cap
 deer_d:=deer/area·1.
 reg_time:=luf(veg_d,regtime)
 food_inc:=(max_food_cap−food)/reg_time
 prey_r:=luf(deer_d,preyr)
 loss_deer:=prey_r·predators
 food_supply:=food/deer
 inc_deer:=deer·luf(food_supply,gr_deer)
 deer:=deer+(inc_deer−loss_deer)·dx
 food:=food+(food_inc−brows_loss)·dx
 t:=t+dx
 predators:=when(x≤m, (a−265)/m·x+265, a)|x=t
  ldv:=augment(ldv,{deer}): lfv:=augment(lfv,{food})
 lprv:=augment(lprv,{predators}): ltime:=augment(ltime,{t})
 i:=i+1
EndWhile
lfv_scal:=lfv/10000:lprv_scal:=lprv·100
Disp "Time in ltime, Deer in ldv, Food in lfv_scal, Predators in lprv_scal"
EndPrgm
```

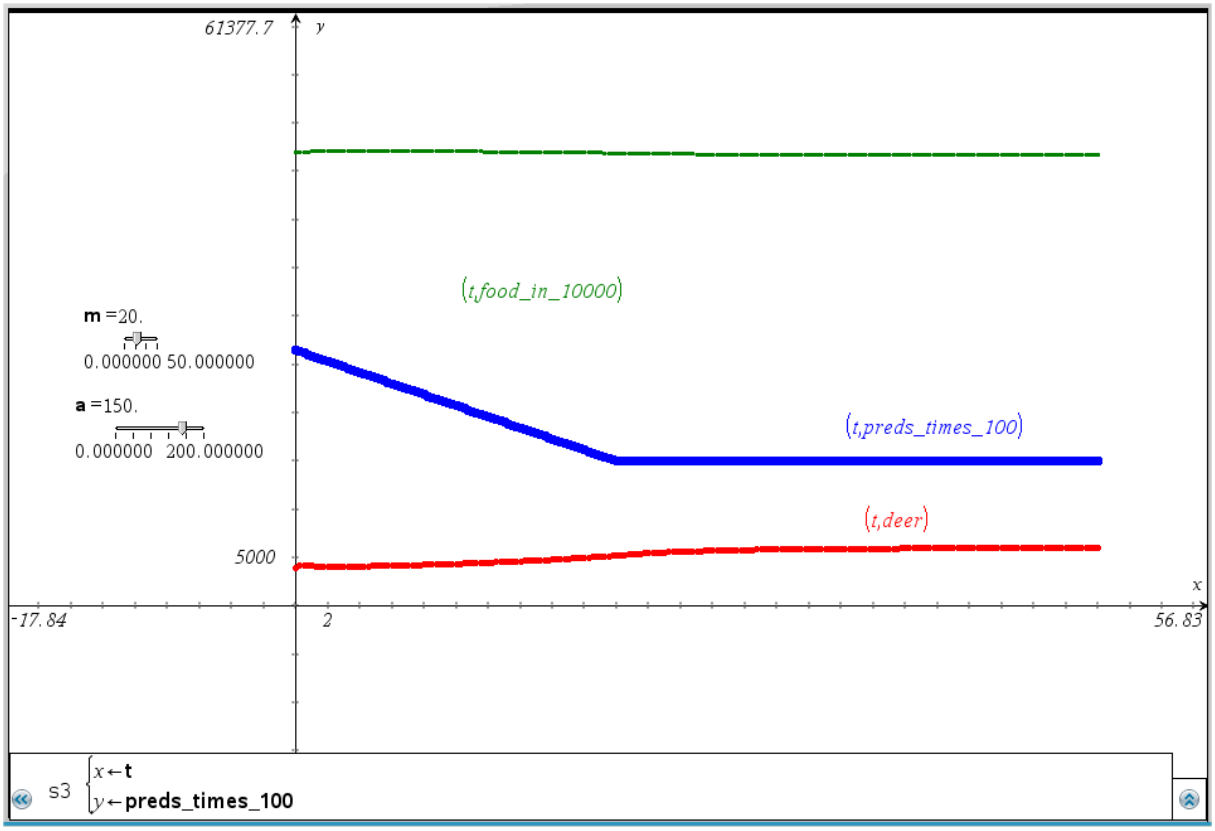
The predator function is given under *predators*:=.



Working with sliders using the program has the disadvantage that after every change of the parameters (moving the sliders) the program must be run again. Then the diagram is adapted immediately.

It is much more comfortable to use the Spreadsheet application. The graphs are manipulated directly moving the sliders. The Spreadsheet is not presented here.

You can see two – of many others – ways to reach *Bossel*'s goal with different stable deer populations. According to the graph below we should decrease the predator stock within the first 20 years linearly down to a number of 150 and then keep this number of predators for the future.



I mentioned in an earlier note my intention to try an approach via the numerical solution of the respective system of differential equations. Voila, it works as you can see in the following.

The ecological catastrophe as a system of differential equations

The form of the differential equations can be derived directly from the *VENSIM*-equations:

$$\frac{dd}{dt} = d \cdot gr\_deer\_f\left(\frac{f}{d}\right) - preyr\_f\left(\frac{d}{area}\right) \cdot pred\_f(t)$$

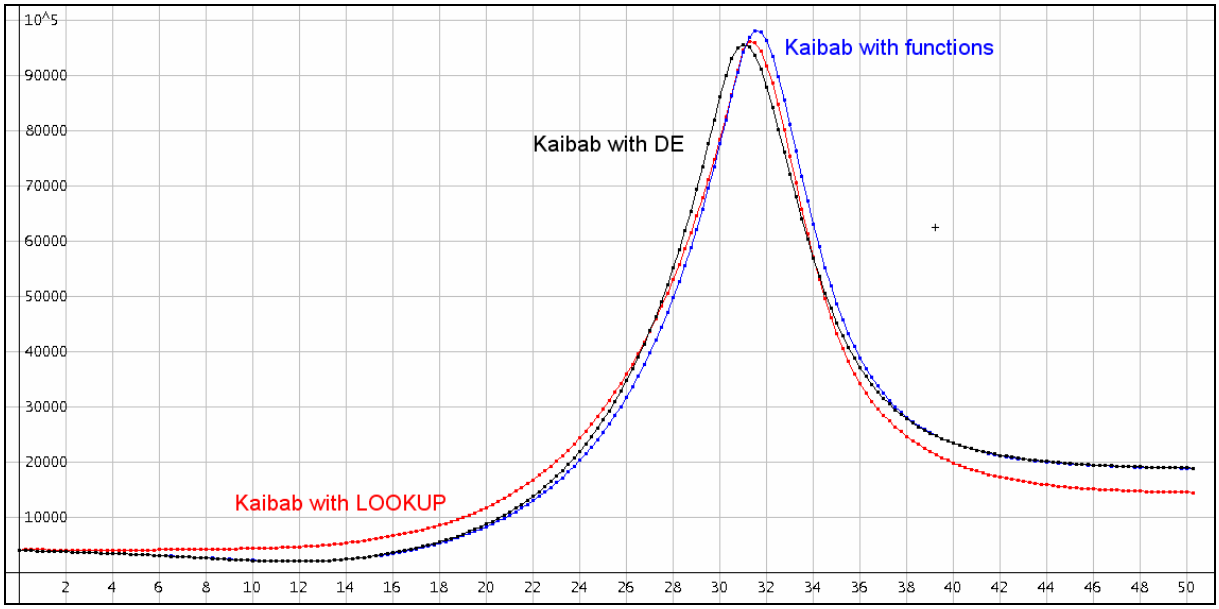
$$\frac{df}{dt} = \frac{max\_food\_cap - f}{regtime\_f\left(\frac{f}{\max\_food\_cap}\right) - graz\_loss(dayly\_requ \cdot d)}$$

I am using the Runge-Kutta-routine of *DERIVE* again.

$$\left(\mathrm{RK}\left(\left[\mathrm{gr\_deer\_f}\left(\frac{f}{d}\right)\cdot d - \mathrm{preyr\_f}\left(\frac{d}{area}\right)\cdot \mathrm{pred\_f}(t),\ \frac{max\_food\_cap - f}{\mathrm{regtime\_f}\left(\frac{f}{max\_food\_cap}\right)} - \mathrm{graz\_loss}(dayly\_requ\cdot d,\ f)\right],\ [t,\ d,\ f],\ [0,\ ini\_deer,\ ini\_food],\ 0.25,\ 201\right)\right)\downarrow\downarrow[1,\ 2]$$

It would be possible to apply the LOOKUP-routines but RK cannot work through all 201 rows of the table in one step.

Selection of the first and second column shows rise and fall of the deer population:



The plot displays all deer-plots and allows comparison..

This model fascinated me indeed, because it offers so many opportunities for treatment. Description of the piecewise defined functions (WITH LOOKUP) by one single function requires some fantasy and knowledge about possible function types. There is no “right” answer and this can be stated about most of modelling problems.

Comparing between applying a program (which must be written in advance) and spreadsheet is charming and exciting as well.

Use of sliders offers an important additional quality and provokes again interpreting the results.

Besides the mathematical point of view this model is demonstrating once more how an intervention in natural procedure (even if in best intention) can destroy the balance of environment and can result in unforeseen consequences.

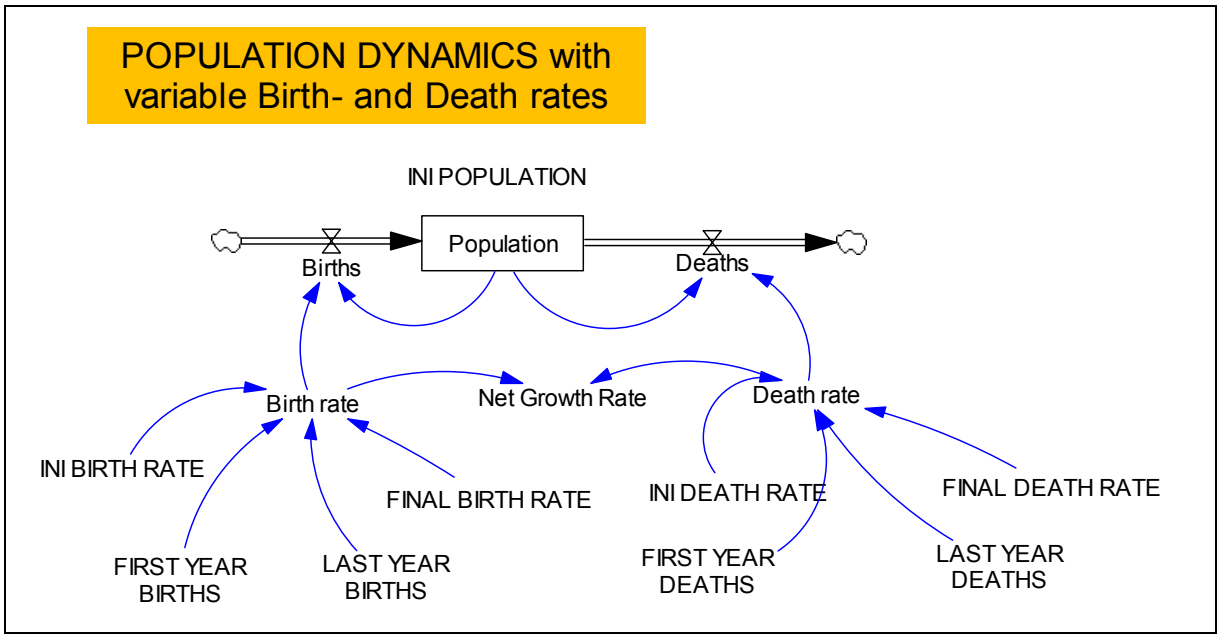
4 Population Dynamics

with variable birth and death rates

We describe the development of a population (initial value = 1000) where births and deaths as well are changing linearly with time. Both rates decrease within a certain time interval constant from an initial to a final value.

I start with the *VENSIM* – simulation diagram.

Right: *Elementary school in Hellville, Nosy Be,Madagascar*



Here are more constants and fewer equations. But we meet something new, the RAMP-function. It describes the constant change of *Birth*- and *Death rates*.

The document contains all parameters together with their assigned values and all equations which all have been entered in alphabetical order:

(01) Birth rate = INI BIRTH RATE + RAMP((FINAL BIRTH RATE – INI BIRTH RATE)/ (LAST YEAR BIRTHS – FIRST YEAR BIRTHS), FIRST YEAR BIRTHS , LAST YEAR BIRTHS)

(02) Births = Birth rate*Population

(03) Death rate = INI DEATH RATE + RAMP((FINAL DEATH RATE – INI DEATH RATE)/ (LAST YEAR DEATHS – FIRST YEAR DEATHS), FIRST YEAR DEATHS , LAST YEAR DEATHS)

(04) Deaths = Death rate*Population

(05) FINAL BIRTH RATE = 0.01

(06) FINAL DEATH RATE = 0.012

(07) FINAL TIME = 2100

(08) FIRST YEAR BIRTHS = 2010

(09) FIRST YEAR DEATHS = 2010

(10) INI BIRTH RATE = 0.04

(11) INI DEATH RATE = 0.015

(12) INI POPULATION = 1000

(13) INITIAL TIME = 2000
The initial time for the simulation.

(14) LAST YEAR BIRTHS = 2060

(15) LAST YEAR DEATHS = 2030

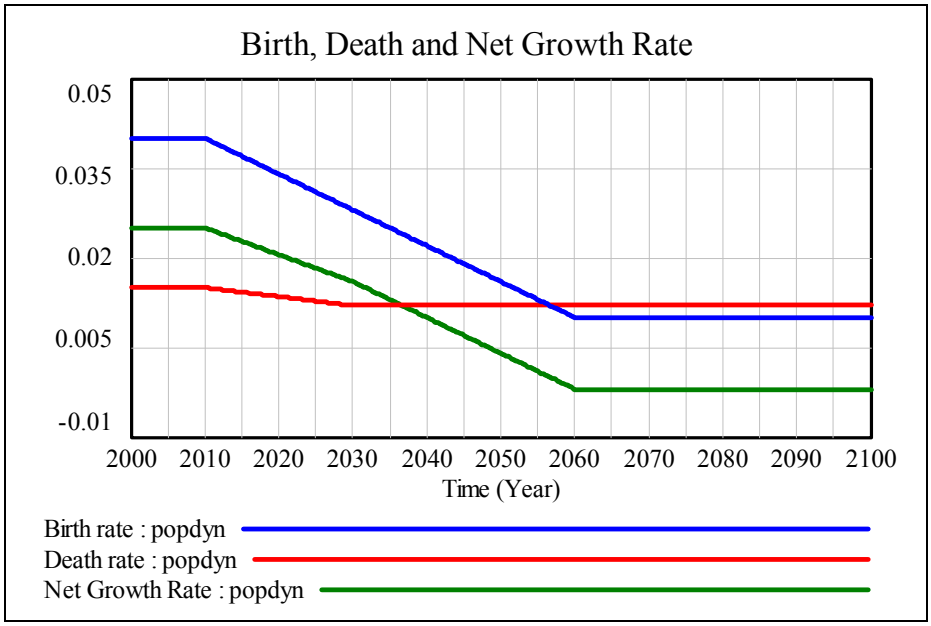
(16) Net Growth Rate = Birth rate – Death rate

(17) Population= INTEG (Births – Deaths, INI POPULATION)

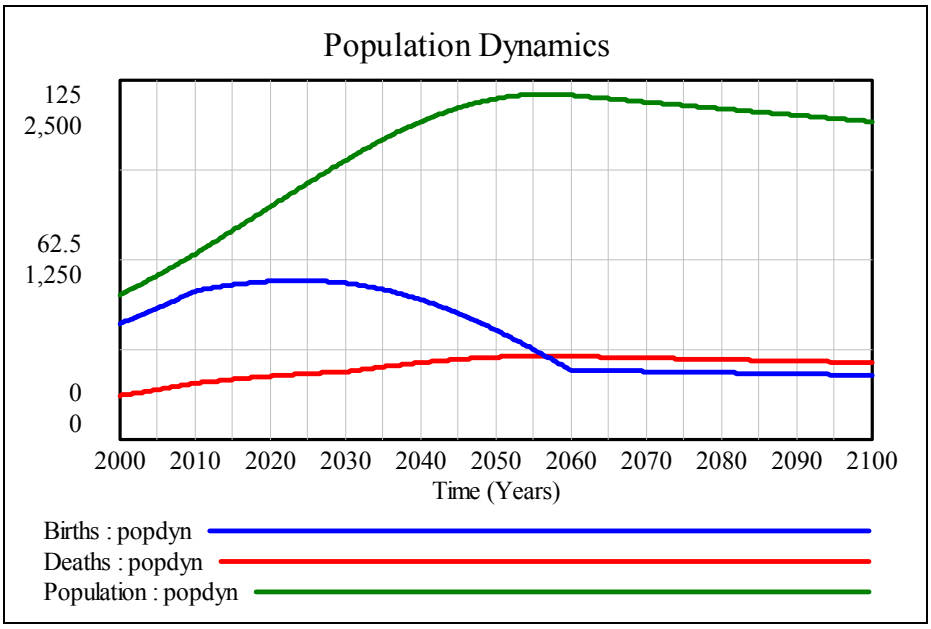
(18) SAVEPER = TIME STEP
The frequency with which output is stored.

(19) TIME STEP = 0.25
The time step for the simulation.

Graphs of the rates explain the name of the function RAMP:

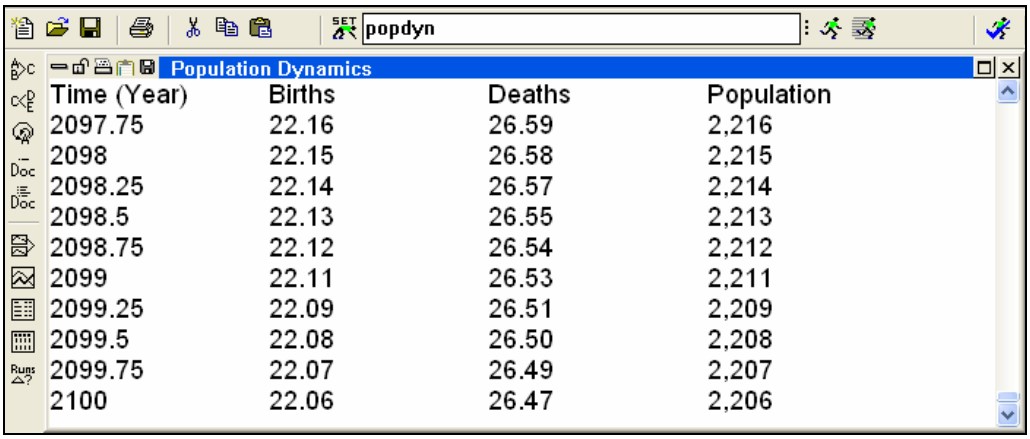


The next diagram shows how population develops. In our case birth rate decreases faster than the death rate.

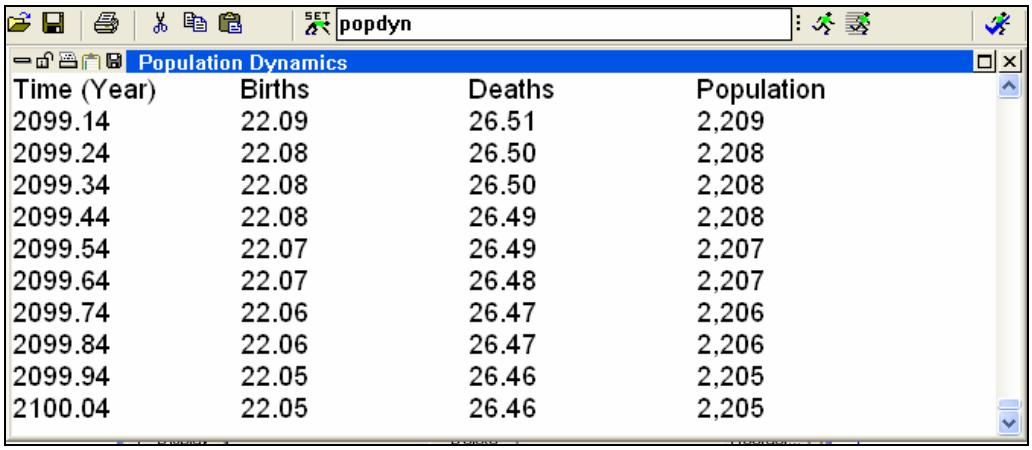


The results can be presented in form of a table, too. See here the last rows of the table.

TIME STEP in *System Zoo* is 0.1. (My TIME STEP used is 0.25). The results do not really differ as you can see comparing the next two details of the respective tables.



Time (Year)	Births	Deaths	Population
2097.75	22.16	26.59	2,216
2098	22.15	26.58	2,215
2098.25	22.14	26.57	2,214
2098.5	22.13	26.55	2,213
2098.75	22.12	26.54	2,212
2099	22.11	26.53	2,211
2099.25	22.09	26.51	2,209
2099.5	22.08	26.50	2,208
2099.75	22.07	26.49	2,207
2100	22.06	26.47	2,206



Time (Year)	Births	Deaths	Population
2099.14	22.09	26.51	2,209
2099.24	22.08	26.50	2,208
2099.34	22.08	26.50	2,208
2099.44	22.08	26.49	2,208
2099.54	22.07	26.49	2,207
2099.64	22.07	26.48	2,207
2099.74	22.06	26.47	2,206
2099.84	22.06	26.47	2,206
2099.94	22.05	26.46	2,205
2100.04	22.05	26.46	2,205

The number of the births and deaths in columns 2 and 3 must be interpreted as numbers per year – and not per time step. Hence net increase of population per quarter of a year is $\frac{\text{Births} - \text{Deaths}}{4}$.

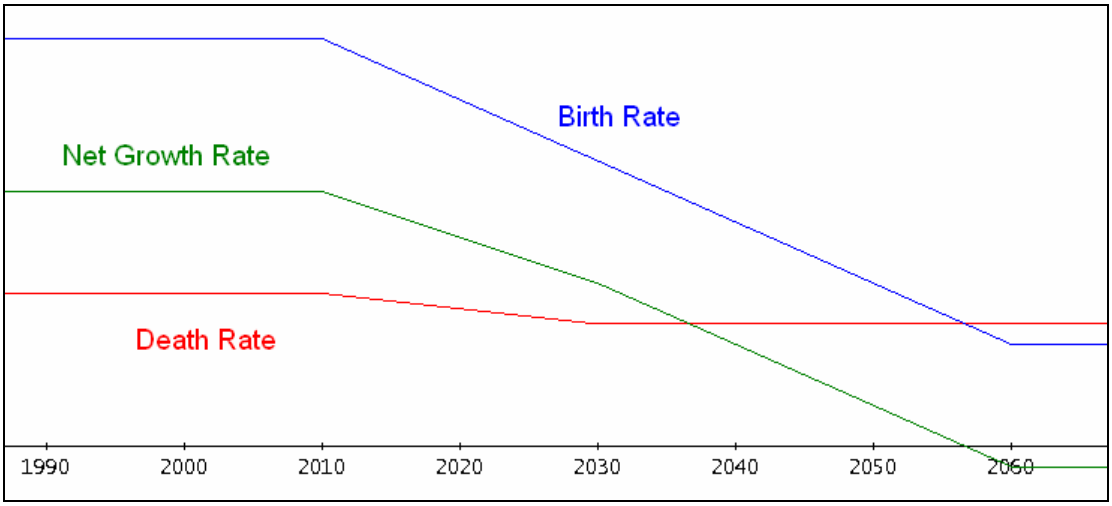
The *DERIVE* model with ITERATES

The ITERATES function is an excellent tool of *DERIVE* for modelling recursive procedures. It is not very easy to handle for many students and other users but it is very efficient.
Since *DERIVE* is programmable ITERATES can be replaced by a small program with a loop.

```
ramp(x, aw, ew, az, ez) :=
  If x ≤ az
     aw
     If x ≤ ez
        aw + (ew - aw)/(ez - az)·(x - az)
        ew

population(n, dt) := ITERATES([t + dt, bev + (ramp(t + dt, 0.04, 0.01,

  2010, 2060) - ramp(t + dt, 0.015, 0.012, 2010, 2030))·bev·dt], [t, bev],

  [2000, 1000], n)
```

The "RAMP-function" which we met in *VENSIM* has been defined now and all rates can be presented:

```
ramp(x, 0.04, 0.01, 2010, 2060)

ramp(x, 0.015, 0.012, 2010, 2030)

ramp(x, 0.04, 0.01, 2010, 2060) - ramp(x, 0.015, 0.012, 2010, 2030)
```

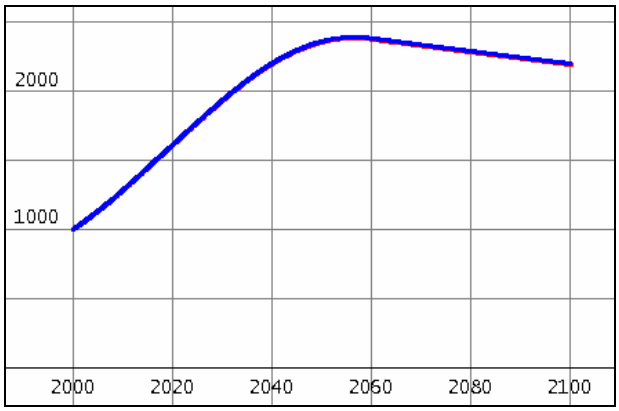


Let's iterate 401 times using a time step of 0.25 years. This is sufficient to reach the 100 years between 2000 and 2100. Syntax of the ITERATES command is a *DERIVE* speciality.

Output is given in form of a matrix containing the coordinates (time, population) of the points which can be plotted immediately.

It turns out that reduction of time step to 0.1 does not result in a change of the graph. The graph on the next page shows both plots superimposed. One can hardly notify any difference.

```
population(401, 0.25)

population(1001, 0.1)
```



Next table shows the three first and three last rows of the matrix for time step 0.25 using first the ITERATES-construct population and then the program pop from below.

```
(population(401, 0.25))
                        [1, 2, 3, 399, 400, 401]
```

2000	1000
2000.25	1006.25
2000.5	1012.53
2099.5	2193.47
2099.75	2192.38
2100	2191.28

```
(pop(401, 0.25))
                        [1, 2, 3, 399, 400, 401]
```

2000	1000
2000.25	1006.25
2000.5	1012.53
2099.5	2208.29
2099.75	2207.18
2100	2206.08

I don't have any explanation for the – small – difference in the population size compared with the *VENSIM* values the end of the table appearing as result of the population-function.

I mentioned above that the ITERATES-command can be replaced by an equivalent program:

```
pop(n, dt, tab, i, p, t) :=
  Prog
    p := 1000
    t := 2000
    tab := [[t, p]]
    i := 1
    Loop
      If i > n
        RETURN tab
        p := p + p·(ramp(t,0.04,0.01,2010,2060) –
                    ramp(t,0.015,0.012,2010,2030))·dt
      t :+ dt
      tab := APPEND(tab, [[t, p]])
      i :+ 1
```

The pop-generated values are matching exactly with the *VENSIM* values as you can verify above!

Finally it would make sense to treat the population development supported by a differential equation.

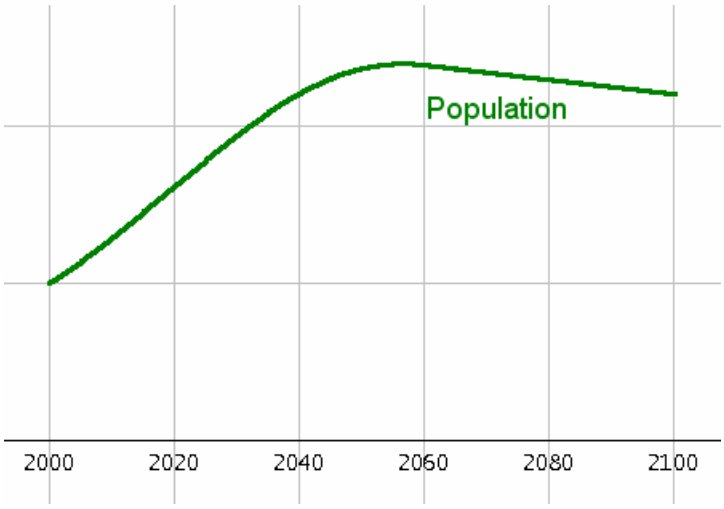
It needs only "translating" the equations (01, 02, 03, 04, and 17) given in the *VENSIM*-document.

$$\frac{dp}{dt} = p \cdot (birth\ rate - death\ rate), \quad p(t = 2000) = 1000$$

We use the self defined RAMP-function for birth rate and death rate and apply the Runge-Kutta numerical method again.

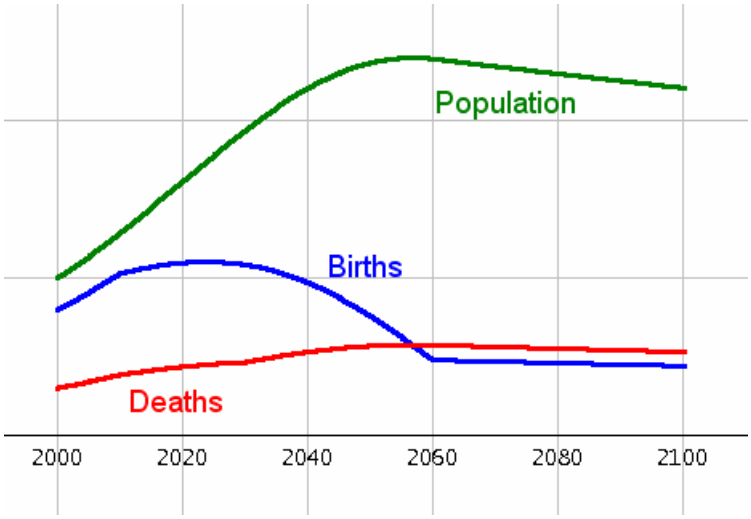
```
(RK([p·(ramp(t, 0.04, 0.01, 2010, 2060) - ramp(t, 0.015, 0.012, 2010, 2030))], [t, p],

  [2000, 1000], 0.25, 401))
                          [1, 2, 3, 399, 400, 401]
```

2000	1000
2000.25	1006.26
2000.5	1012.57
2099.5	2205.60
2099.75	2204.49
2100	2203.39



As you can see table and graph agree with the previously obtained results.

A slight change in the *DERIVE* program presented above gives the opportunity to add the diagrams of births and deaths.



Births and Deaths are multiplied by the scaling factor 80.

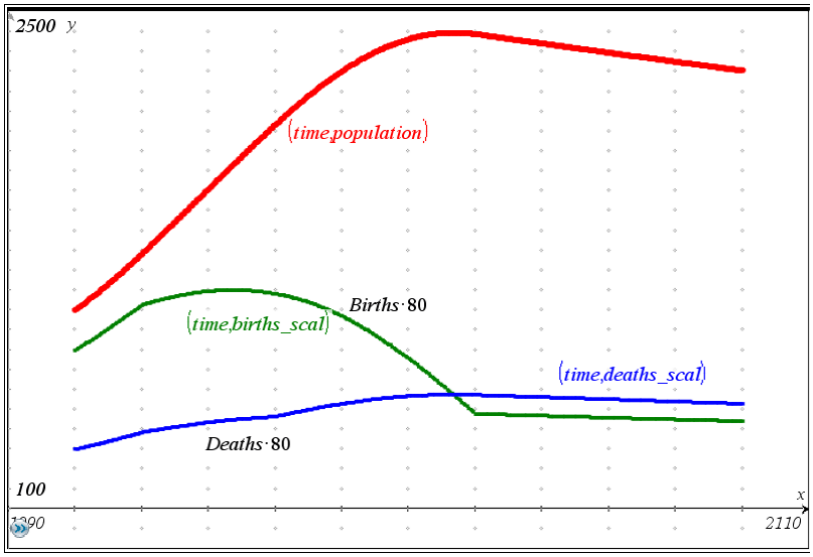
$ramp(x,ax,ex,ay,ey):=when\left(x\le ax,ay,when\left(x\le ex,ay+\frac{ey-ay}{ex-ax}\cdot(x-ax),ey\right)\right)$	*Fertig*
$dx:=0.25$	0.25
$birthr(x):=ramp(x,2010,2060,0.04,0.01)$	*Fertig*
$deathr(x):=ramp(x,2010,2030,0.015,0.012)$	*Fertig*
$ini\_pop:=1000$	1000

The Lists & Spreadsheet application delivers lists for *births*, *deaths* and *population*.

	A time	B births	C deaths	D population	E births_scal	F deaths_scal	G
◆					=births*80	=deaths*80	
1	2000	10.	3.75	1000	800.	300.	
2	2000.25	10.0625	3.77344	1006.25	805.	301.875	
3	2000.5	10.1254	3.79702	1012.54	810.031	303.762	
4	2000.75	10.1887	3.82075	1018.87	815.094	305.66	
5	2001.	10.2524	3.84463	1025.24	820.188	307.571	
6	2001.25	10.3164	3.86866	1031.64	825.314	309.493	
7	2001.5	10.3809	3.89284	1038.09	830.473	311.427	
8	2001.75	10.4458	3.91717	1044.58	835.663	313.374	
9	2002.	10.5111	3.94165	1051.11	840.886	315.332	
10	2002.25	10.5768	3.96629	1057.68	846.142	317.303	
11	2002.5	10.6429	3.99108	1064.29	851.43	319.286	
12	2002.75	10.7094	4.01602	1070.94	856.751	321.282	
13	2003.	10.7763	4.04112	1077.63	862.106	323.29	
14	2003.25	10.8437	4.06638	1084.37	867.494	325.31	

B1 =d1·birthr(a1)·dx

I multiplied *Births* and *Deaths* by 80 in order to represent all lists on the same axes which results in a well known diagram.



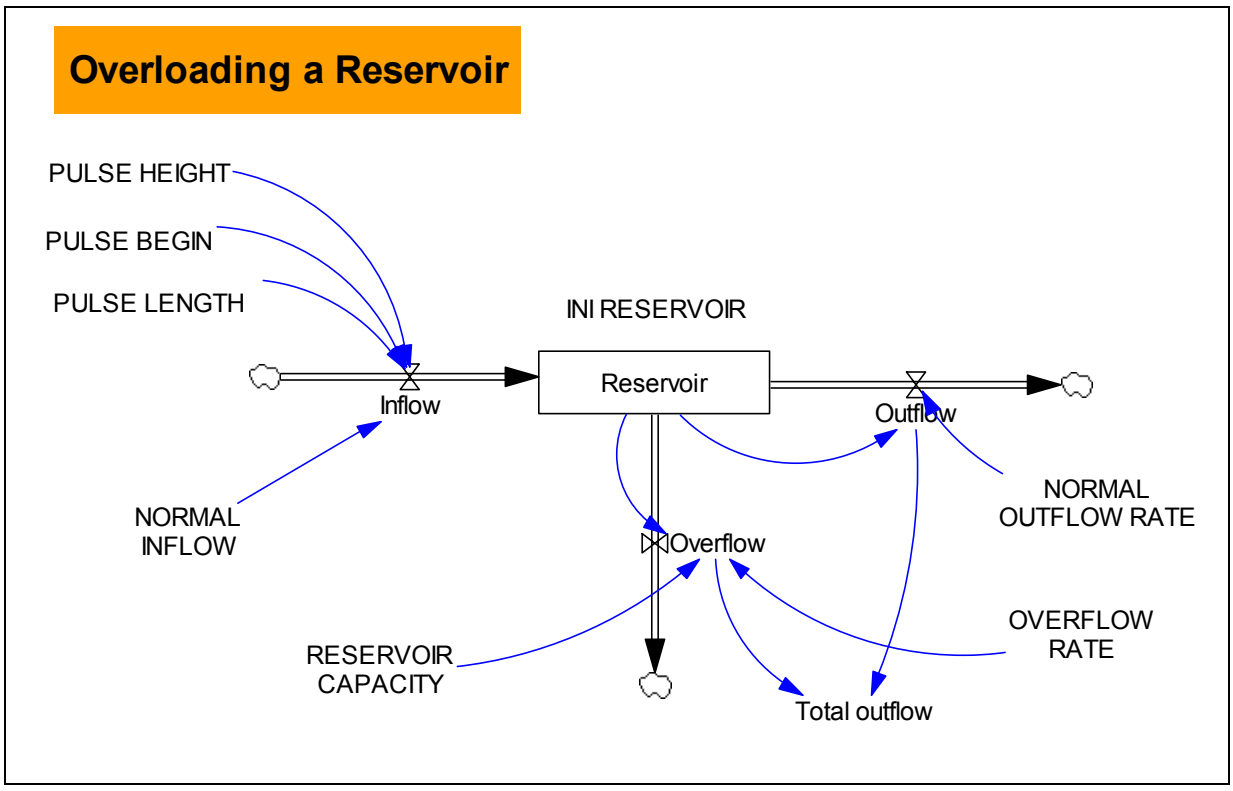
5 The Reservoir is flowing over!

We simulate the dynamics of a *Reservoir*. Its RESERVOIR CAPACITY can be exceeded by a constant NORMAL INFLOW and additionally by an *Inflow* in form of a pulse (melting of snow, heavy rains, …). We assume that the normal *Outflow* is proportional to the current contents of the *Reservoir* with a NORMAL OUTFLOW RATE. In case of overloading the reservoir, the surplus results as *Overflow* with an OVERFLOW RATE, which is greater than the NORMAL OUTFLOW RATE.

Kaprun storage reservoir, Salzburg, Austria

The extra load in form of a pulse is described by the PULSE HEIGHT which starts at PULSE BEGIN with a duration PULSE LENGTH.

It is no problem to set up the *VENSIM* stock and flow diagram. Later you will see how to describe the PULSE-function by a short function. Take it now as it is.



The document comprises all data of the model (I changed the order):

(03) INI RESERVOIR = 0.3

(05) NORMAL INFLOW = 0.25

(06) NORMAL OUTFLOW RATE = 0.5

(09) OVERFLOW RATE = 10

(10) PULSE BEGIN = 5

(11) PULSE HEIGHT = 10

(12) PULSE LENGTH = 0.2

(14) RESERVOIR CAPACITY = 1

(02) Inflow = NORMAL INFLOW + PULSE HEIGHT * PULSE(PULSE BEGIN, PULSE LENGTH)

(07) Outflow = NORMAL OUTFLOW RATE * Reservoir

(08) Overflow= IF THEN ELSE(Reservoir > RESERVOIR CAPACITY, OVERFLOW RATE*(Reservoir – RESERVOIR CAPACITY), 0)

(13) Reservoir = INTEG(+Inflow – Outflow – Overflow, INI RESERVOIR)

(17) Total outflow = Outflow + Overflow

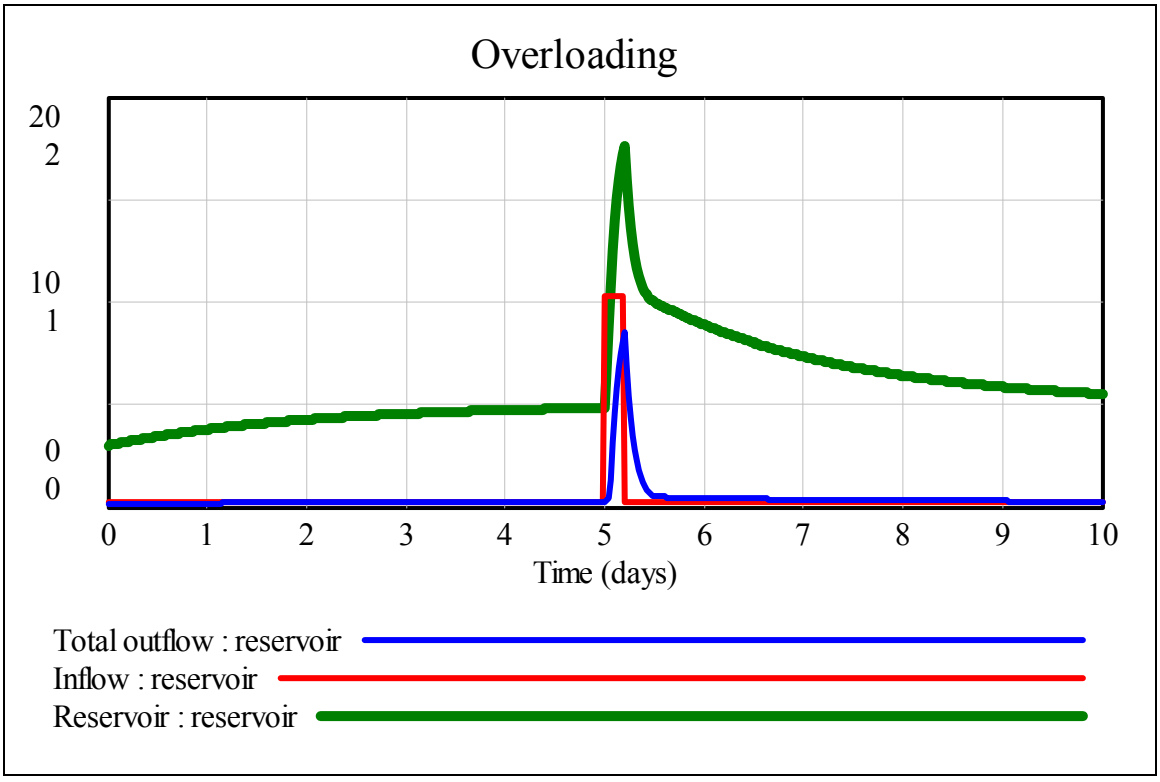
(04) INITIAL TIME = 0

(01) FINAL TIME = 20

(16) TIME STEP = 0.02

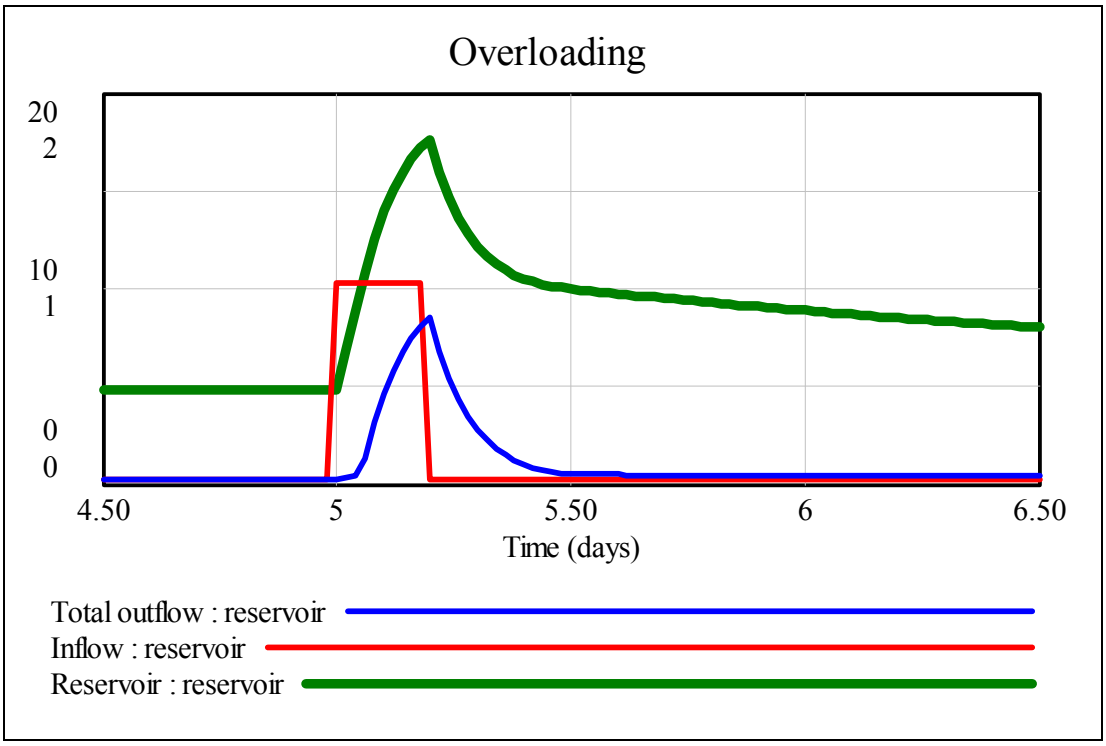
(15) SAVEPER = TIME STEP

We will inspect the diagram based on the proposed parameters:



Scaling: $0 \le$ *Reservoir* ≤ 2 and $0 \le$ *Inflow, Total outflow* ≤ 20

We can spread the exciting time span between days 4 and 6.

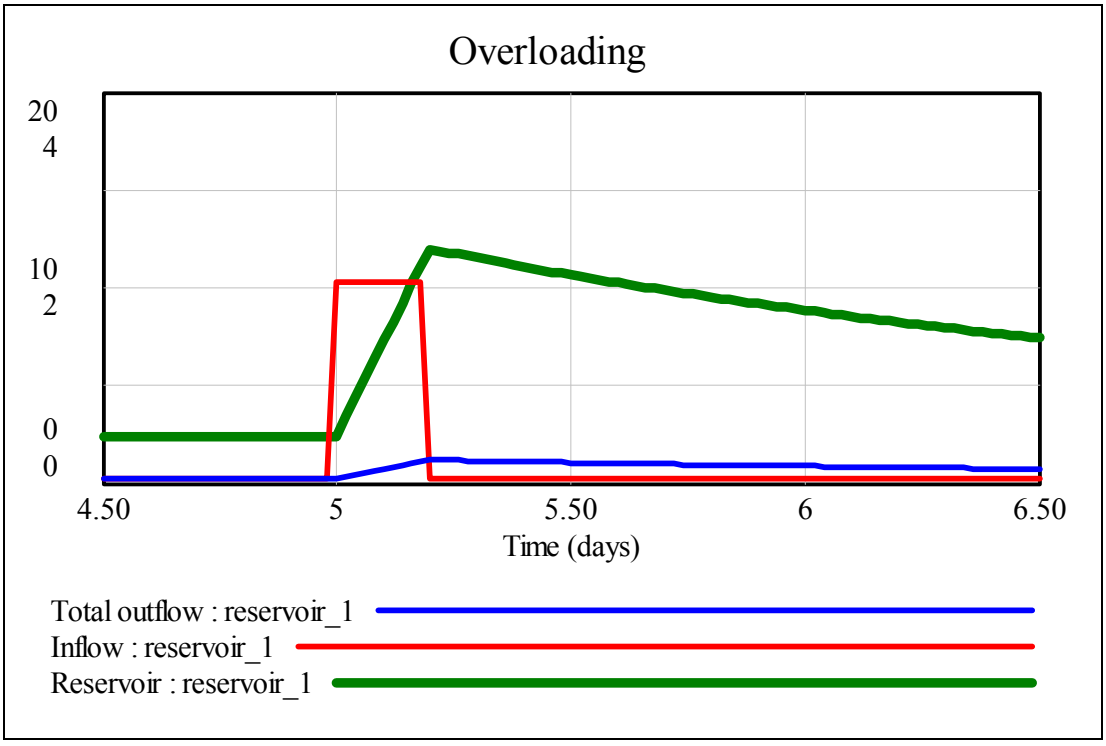


How to interpret this diagram?

The *Reservoir* fills slowly and reaches the equilibrium value of 0.5 units after approximately 4.5 days. Then suddenly an additional *inflow* – the pulse – arrives which results quickly in an *overflow*. This *overflow* is lasting until the RESERVOIR CAPACITY has increased down to 1 unit. Then it needs about 10 days to reach the normal state again.

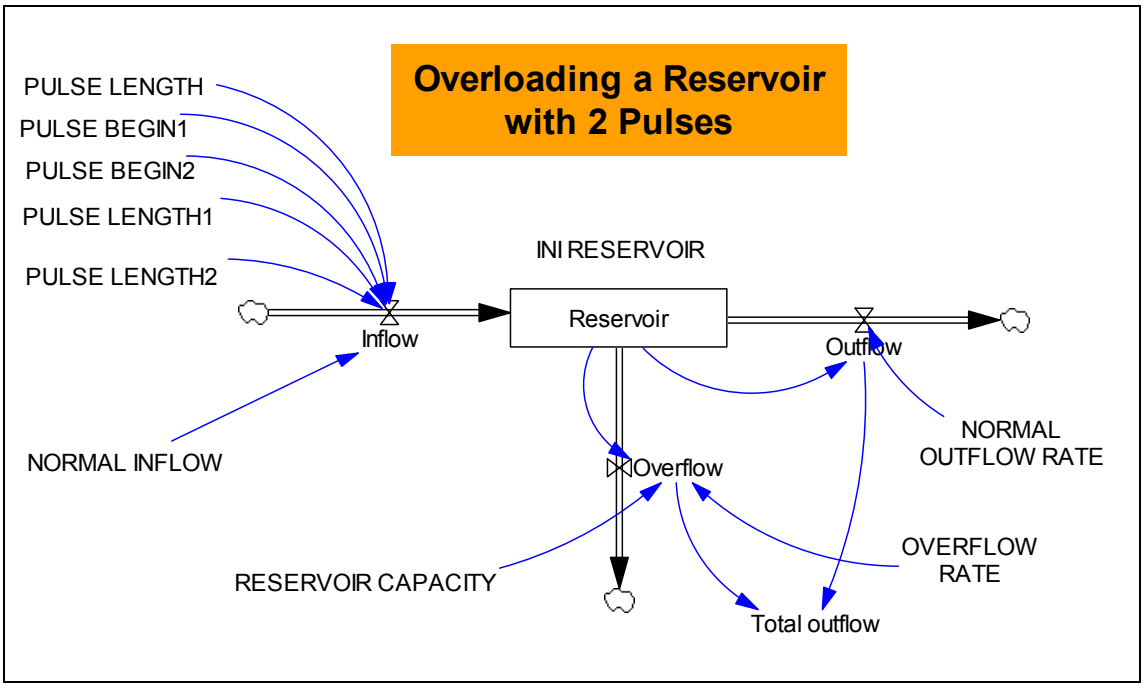
Of course, the system depends mainly on the RESERVOIR CAPACITY and the NORMAL OUTFLOW RATE. If both are sufficiently large then we will hardly face an overflow.

We simulate the process increasing the capacity to 2.5 units.



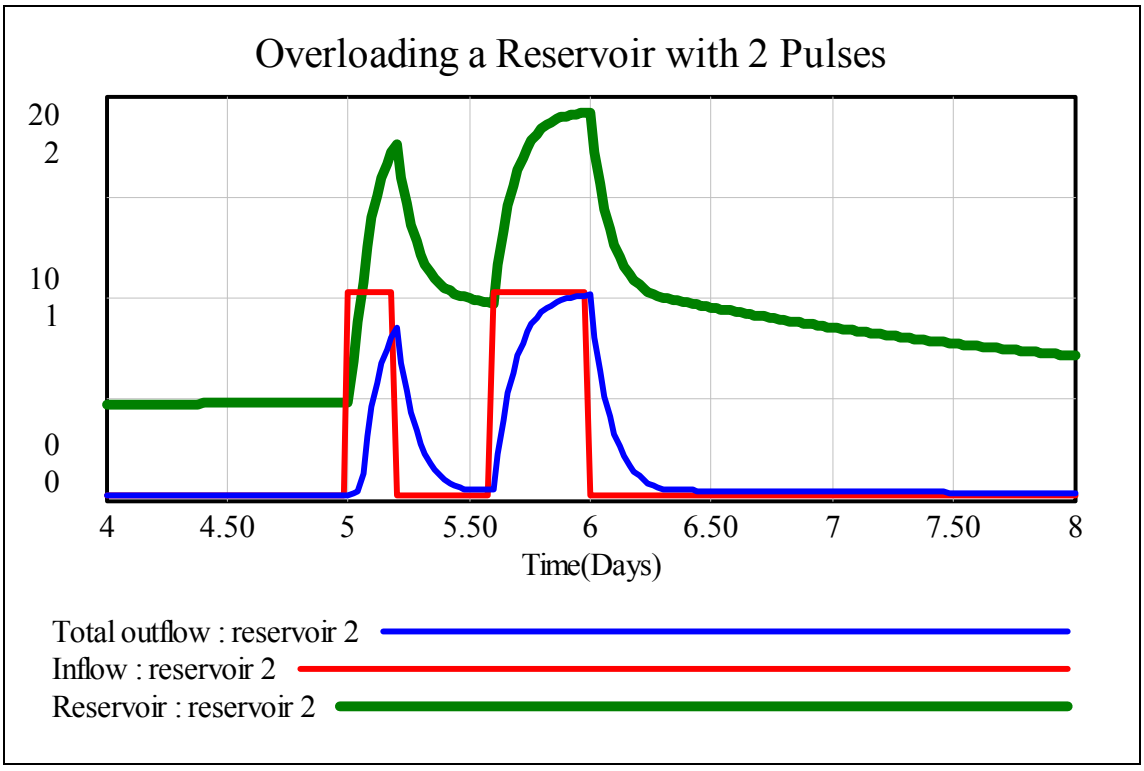
Take care of the vertical scaling: now we have $0 \leq Reservoir \leq 4$. The greater capacity removes the peak of sudden inflow.

Modelling the behaviour of a reservoir we will investigate what is happening if soon after the first pulse a second one will follow – the thunderstorm with heavy rains returns. We just add another PULSE.



PULSE HEIGHT shall remain the same for both pulses; the second pulse will start at 5.6 with a PULSE LENGTH of 0.4 days. The *Inflow*-equation changes.

Inflow = NORMAL INFLOW + PULSE LENGTH * PULSE(PULSE BEGIN1, PULSE LENGTH1)
+ PULSE LENGTH * PULSE(PULSE BEGIN2, PULSE LENGTH2)



Here are the most interesting rows of the table for better comparison with the next realisations of the model:

Time (Day)	Total outflow	Inflow	Reservoir
0	0.15	0.25	0.3
0.02	0.151	0.25	0.3020
0.04	0.1519	0.25	0.3039
0.06	0.1529	0.25	0.3059
0.08	0.1539	0.25	0.3078
0.1	0.1549	0.25	0.3098

The first pulse arrives in the reservoir:

4.96	0.2417	0.25	0.4834
4.98	0.2418	0.25	0.4836
5	0.2418	10.25	0.4837
5.02	0.3419	10.25	0.6839
5.04	0.4410	10.25	0.8821
5.06	1.322	10.25	1.078

At time 5.6 the second pulse arrives and pours into the reservoir:

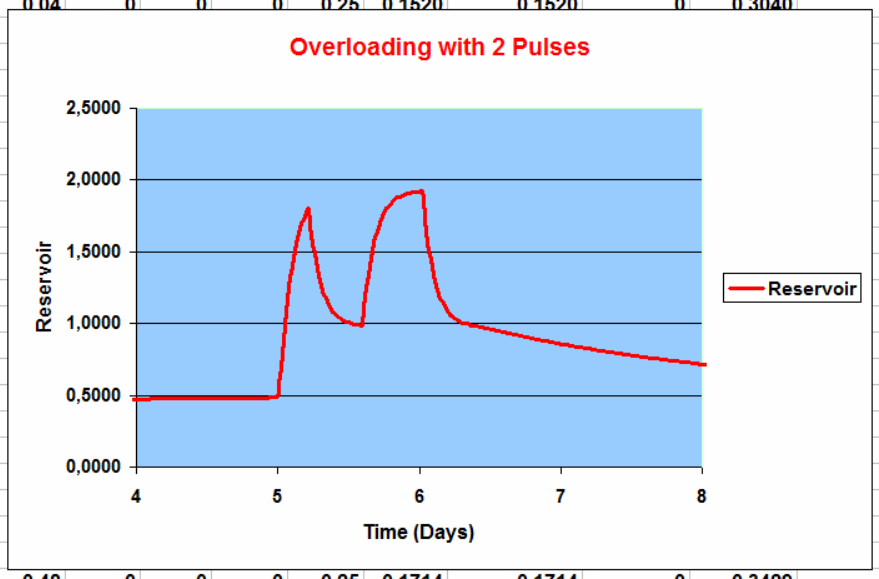
5.5	0.4995	0.25	0.9991
5.52	0.4970	0.25	0.9941
5.54	0.4946	0.25	0.9892
5.56	0.4921	0.25	0.9843
5.58	0.4897	0.25	0.9795
5.6	0.4873	10.25	0.9747
5.62	2.284	10.25	1.169
5.64	3.957	10.25	1.329
5.66	5.278	10.25	1.455
5.68	6.322	10.25	1.554
5.7	7.147	10.25	1.633
6.29999	0.5387	0.25	1.003
6.31999	0.4989	0.25	0.9979
6.33999	0.4964	0.25	0.9929
6.35999	0.4940	0.25	0.9880
6.37999	0.4915	0.25	0.9831
6.39999	0.4891	0.25	0.9782
19.9603	0.2502	0.25	0.5005
19.9803	0.2502	0.25	0.5005
20.0003	0.2502	0.25	0.5005

Now we have retruned to normal conditions since some time.

How the Model can look like with *MS-Excel*

For the first model we take the parameters from above (2 pulses).

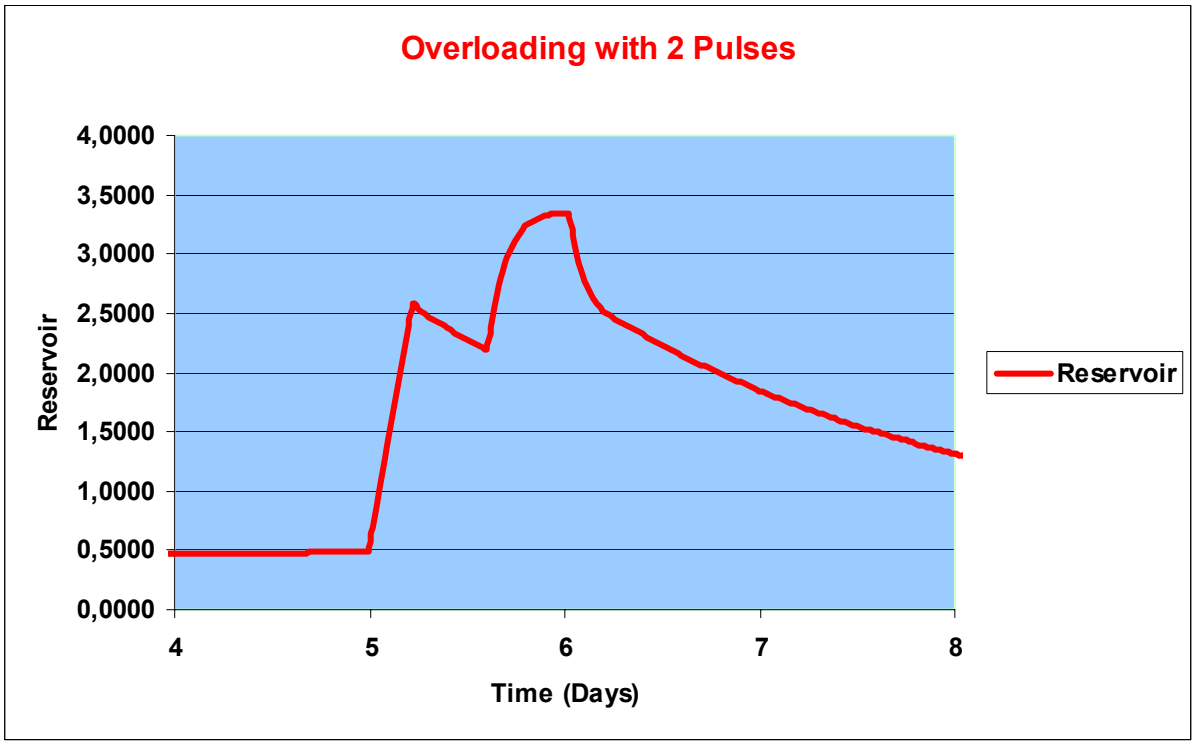
	A	B	C	D	E	F	G	H	I	J	K	L
3	Data		Time	Pulse 1	Pulse2	Pulses	Inflow	Outflow	Total Outflow	Overflow	Reservoir	
4			0	0	0	0	0,25	0,1500	0,1500	0	0,3000	
5	Ini Reservoir	0,3	0,02	0	0	0	0,25	0,1510	0,1510	0	0,3020	
6			0,04	0	0	0	0,25	0,1520	0,1520	0	0,3040	
7	Reservoir Cap.	1										
8												
9	Normal Inflow	0,25										
10												
11	Normal Outflow Rate	0,5										
12												
13	Overflow Rate	10										
14												
15	Pulse Height	10										
16												
17	Pulse Begin 1	5										
18												
19	Pulse Length 1	0,2										
20												
21	Pulse Begin 2	5,6										
22												
23	Pulse Length 2	0,4										
24												
25	Time Step	0,02										
26												
27												
28			0,48	0	0	0	0,25	0,1714	0,1714	0	0,3429	
29			0,5	0	0	0	0,25	0,1722	0,1722	0	0,3444	



After 20 days it looks like the *VENSIM* model:

20	**0**	**0**	**0**	**0.25**	**0.2502**	**0.2502**	**0**	**0.5005**

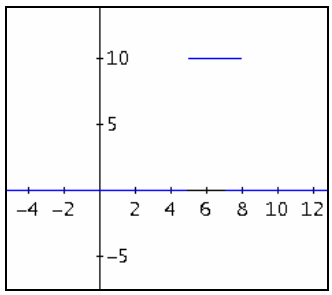
Increasing the RESERVOIR CAPACITY up to 2.5 units the diagram changes:



The Pulses and *DERIVE*

Having been successful modelling the problem twice it should be easy to transfer the "performance" of the reservoir to a *DERIVE* program. We predefine the PULSE-function.

```
      pulse(m, b, d, x) :=
        If b ≤ x ≤ b + d
#1:        m
           0

#2:   pulse(10, 5, 3)
```



The simulation program is very short:

```
reservoir(normout_r, normin, res, rescap, p1, stp1, lep1, p2, stp2, lep2,
            ovfl_r, t_start, t_end, dt, tab, t, totalin, ovfl, totalout) :=
  Prog
    tab := []
    t := t_start
    Loop
      If t > t_end
         RETURN tab
      totalin := normin + pulse(p1, stp1, lep1, t) + pulse(p2, stp2, lep2, t)
      ovfl := IF(res > rescap, ovfl_r·(res − rescap), 0)
      totalout := normout_r·res + ovfl
      tab := APPEND(tab, [[t, totalout, totalin, res]])
      res := res + dt·(totalin − totalout)
      t :+ dt
```

I present some values of the table; please compare with the respective *VENSIM* and *MS-Excel* values:

```
reservoir(0.5, 0.25, 0.3, 1, 10, 5, 0.2, 10, 5.6, 0.4, 10, 0, 0.1, 0.02)
```

0	0.15	0.25	0.3
0.02	0.151	0.25	0.302
0.04	0.1519	0.25	0.3039
0.06	0.1529	0.25	0.3059
0.08	0.1539	0.25	0.3078
0.1	0.1549	0.25	0.3098

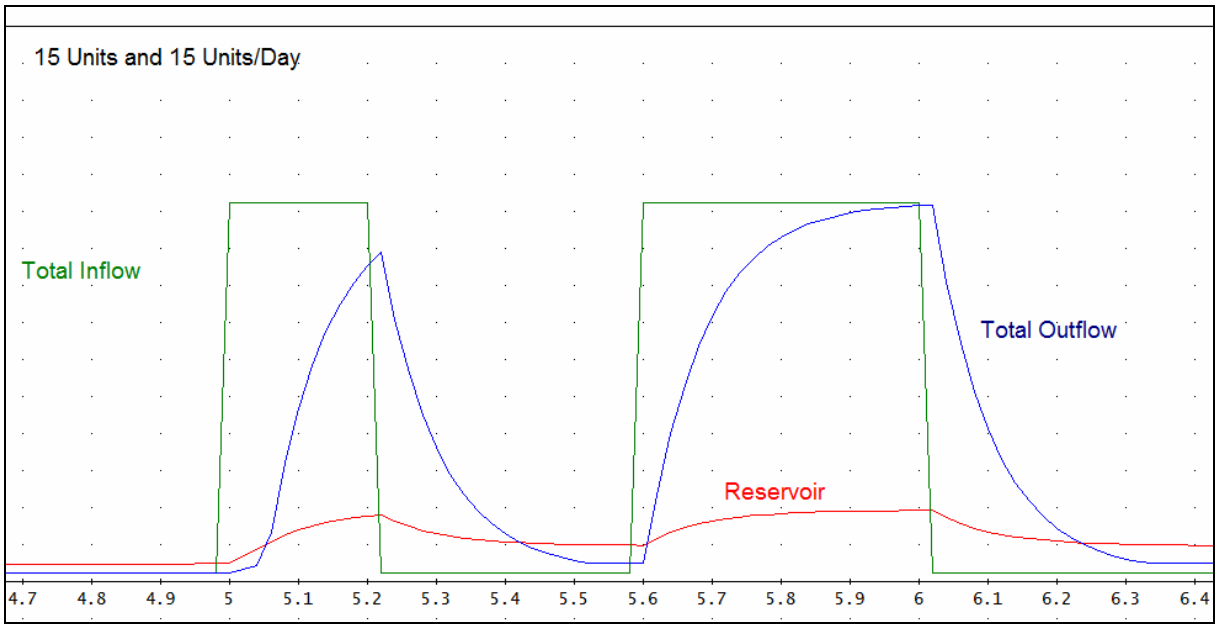
```
(reservoir(0.5, 0.25, 0.3, 1, 10, 5, 0.2, 10, 5.6, 0.4, 10, 0, 20, 0.02))
                                                                [249, ..., 254]
```

4.96	0.2417	0.25	0.4834
4.98	0.2418	0.25	0.4836
5	0.2418	10.25	0.4837
5.02	0.3419	10.25	0.6839
5.04	0.4410	10.25	0.8821
5.06	1.322	10.25	1.078

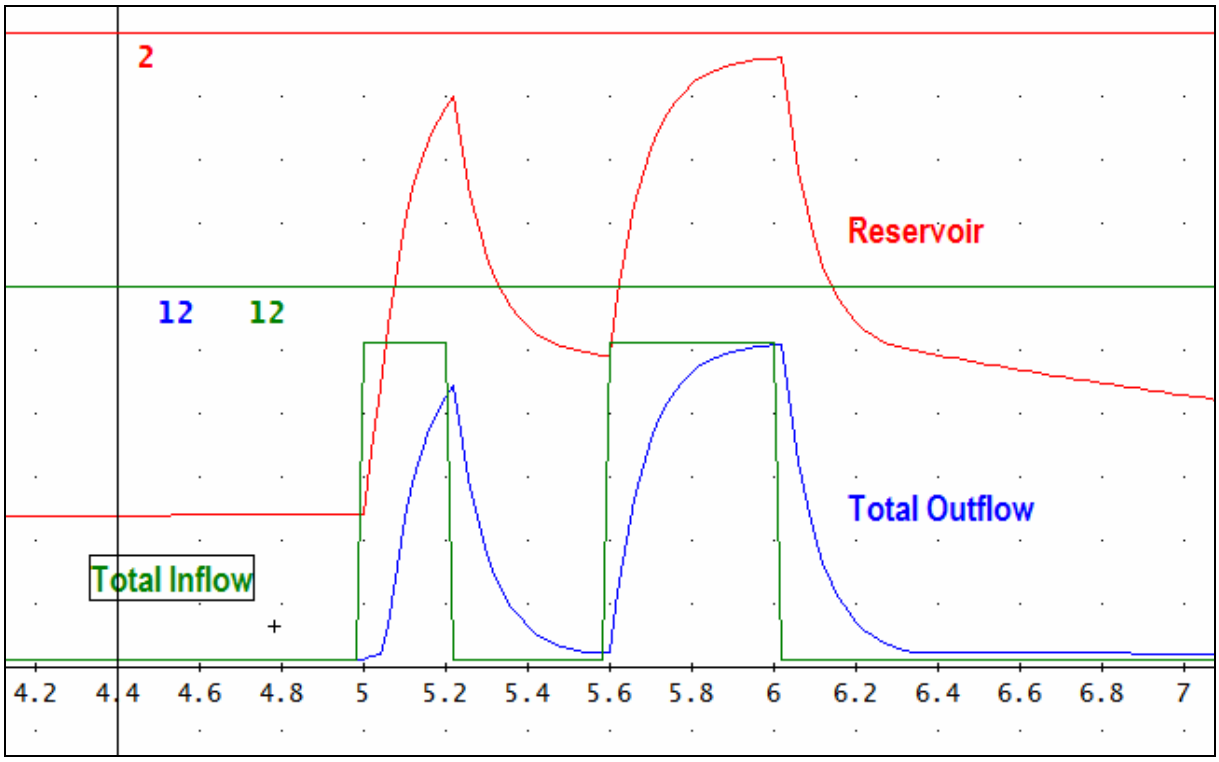
This looks pretty satisfying. I leave the same y-scaling for all values.

```
(reservoir_ext(0.5, 0.25, 0.3, 1, 10, 5, 0.2, 10, 5.6, 0.4, 10, 0, 10, 0.02))↓↓[1, 4]
(reservoir_ext(0.5, 0.25, 0.3, 1, 10, 5, 0.2, 10, 5.6, 0.4, 10, 0, 10, 0.02))↓↓[1, 2]
(reservoir_ext(0.5, 0.25, 0.3, 1, 10, 5, 0.2, 10, 5.6, 0.4, 10, 0, 10, 0.02))↓↓[1, 3]
y = 15
```

In order to get a better orientation I include the horizontal line $y = 15$.



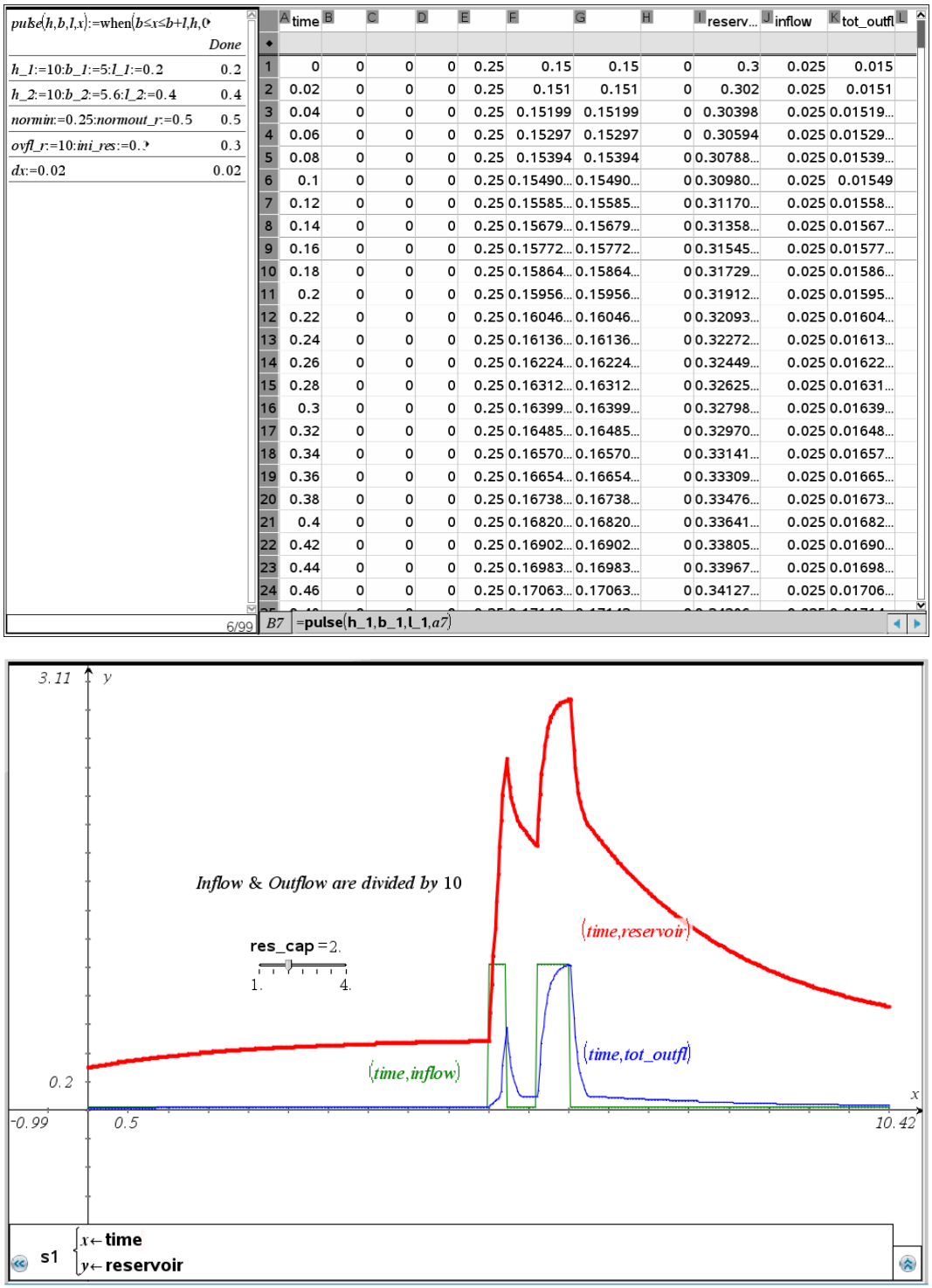
Again I am regretting that it is not able to introduce sliders for the various parameters. Using a little trick we can produce the same diagram as with *VENSIM* (different scalings).



Bossel poses the task to investigate the influence of RESERVOIR CAPACITY on avoiding peaks in the reservoir *Outflow*. A slider would be very helpful.

We could realize this with *GeoGebra* but at the moment this system is not stable enough and calculation times are too long for handling this amount of data – up to a total „hang up“. We can achieve this with *TI-Nspire* with little effort.

Modeling with sliders and *TI-NspireCAS*



All calculations are performed in the spreadsheet. It is easy to represent the lists in the diagram. It would be no problem to introduce more sliders for other parameters, too.

6 Density dependent Growth: Michaelis-Menten-Kinetics

A growth function which is similar to the logistic growth is based on the Michaelis-Menten-Kinetics.

Leonor Michaelis, 1875 – 1949, German biochemist and Maud Leonora Menten, 1879 – 1960, Canadian medical scientist

This function is especially suitable for describing saturation processes in chemistry and biochemistry. Reaction velocity v depends among others on the concentration of the respective stock and can be described by the equation

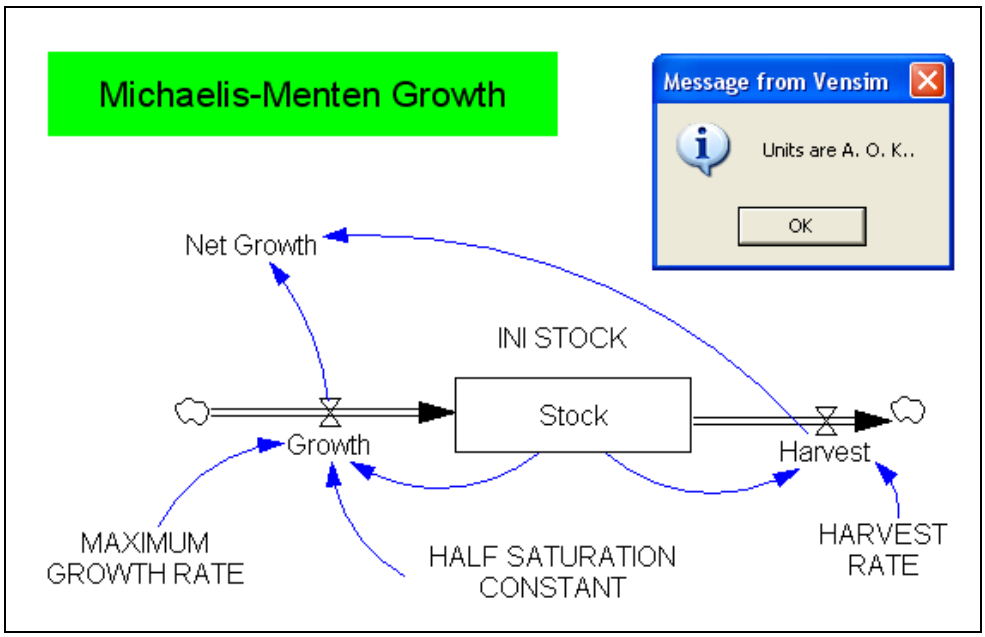
$$v = v_{\max} \frac{S}{S+c}.$$

S is the respective concentration of the substrate. c is the Michaelis-Menten-Constant or the Dissociation Constant or Half Saturation Constant. The latter name is derived by the fact that for $S = c$ the outcome is half of the saturation effect.

Whereas in logistic growth the increase of stock is described by $r \cdot Stock \cdot \left(1 - \frac{Stock}{Capacity}\right)$, in this case we have for the increase the formula: $r \cdot Stock \cdot \left(1 - \frac{Stock}{Stock + c}\right)$ with r being the MAXIMUM GROWTH RATE. We introduce a HARVEST RATE h for considering the stock decrease.

This leads to the differential equation for this special form of growth: $S' = r \cdot S \cdot \left(1 - \frac{S}{S+c}\right) - h \cdot S.$

The *VENSIM*-stock and flow diagram is not very complicated.



The model document follows (here in alphabetical order). Something is new: I entered the dimensions of the variables. Then I can induce *VENSIM* to perform a dimension analysis checking consistency of all quantities. Here the answer is: *Units are O.K*. I can also check the model as a whole and I receive as answer: *Net Growth is not used in the model*. This is ok, because *Net Growth* is only an intermediate value which can be used for plotting or reading off in the table.

(01) FINAL TIME = 50
Units: Day
The final time for the simulation.

(02) Growth = GROWTH RATE * Stock * (1 – Stock/(HALF SATURATION CONSTANT + Stock))
Units: M/Day

(03) MAXIMUM GROWTH RATE = 1
Units: 1/Day

(04) HALF SATURATION CONSTANT = 1
Units: M

(05) Harvest = HARVEST RATE * Stock
Units: M/Day

(06) HARVEST RATE = 0.5
Units: 1/Day

(07) INI STOCK = 0.02
Units: M

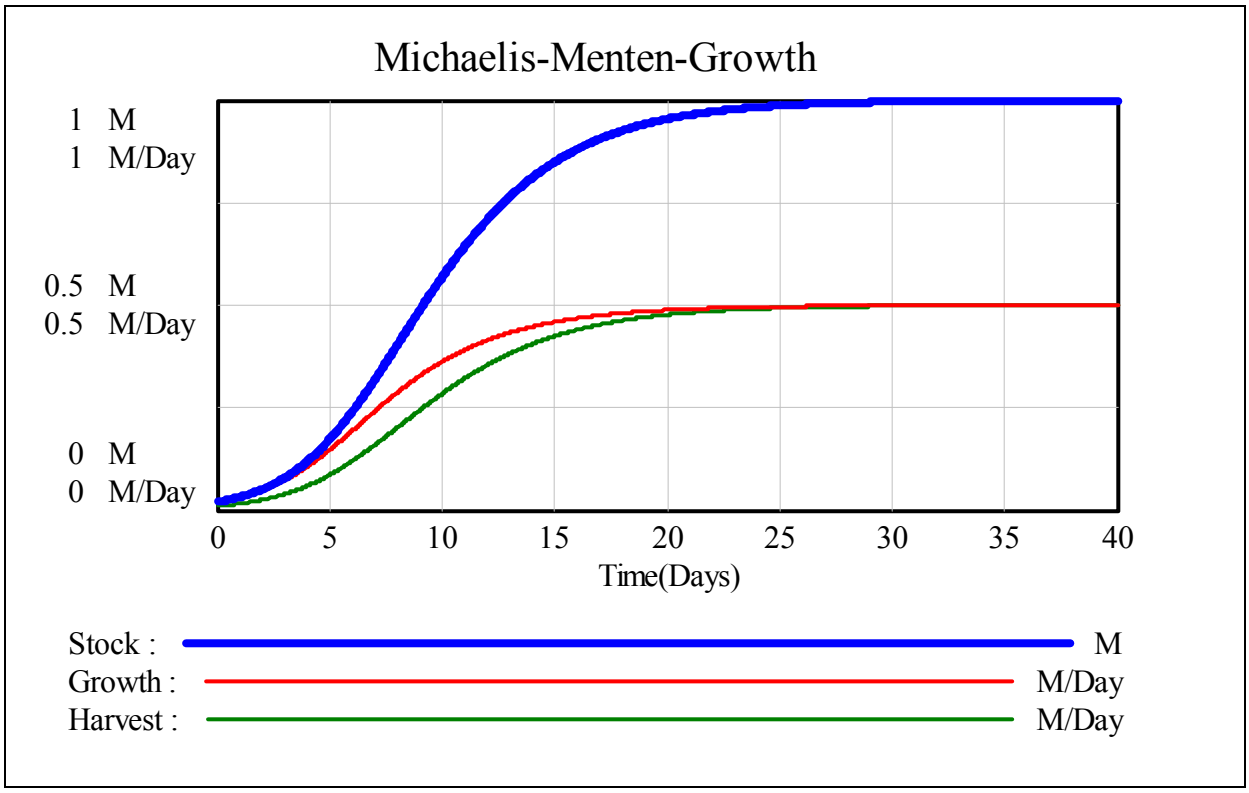
(08) INITIAL TIME = 0
Units: Day
The initial time for the simulation.

(09) Net Growth = Growth – Harvest
Units: M/day

(10) SAVEPER = TIME STEP
Units: Day [0,?]
The frequency with which output is stored.

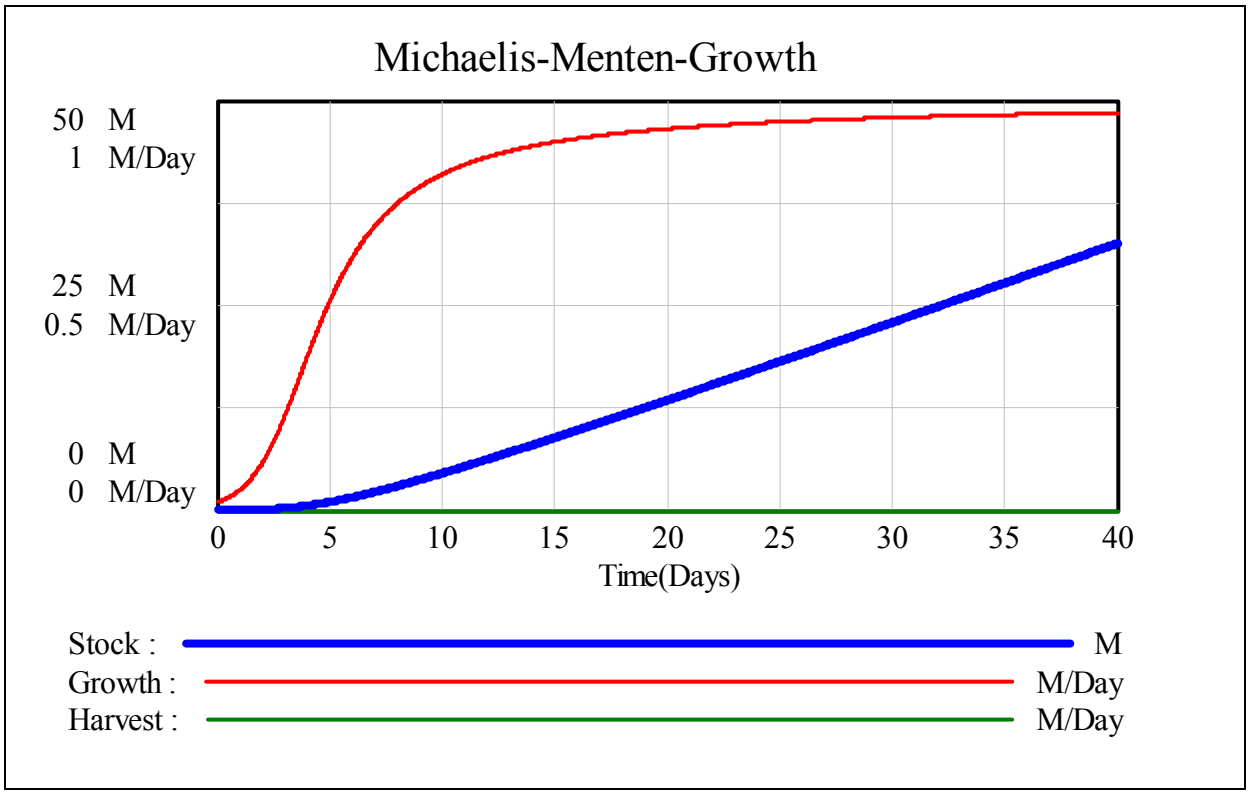
(11) Stock= INTEG (Growth – Harvest, INI STOCK)
Units: M

(12) TIME STEP = 0.02
Units: Day [0,?]
The time step for the simulation.



Bossel discusses two special cases:

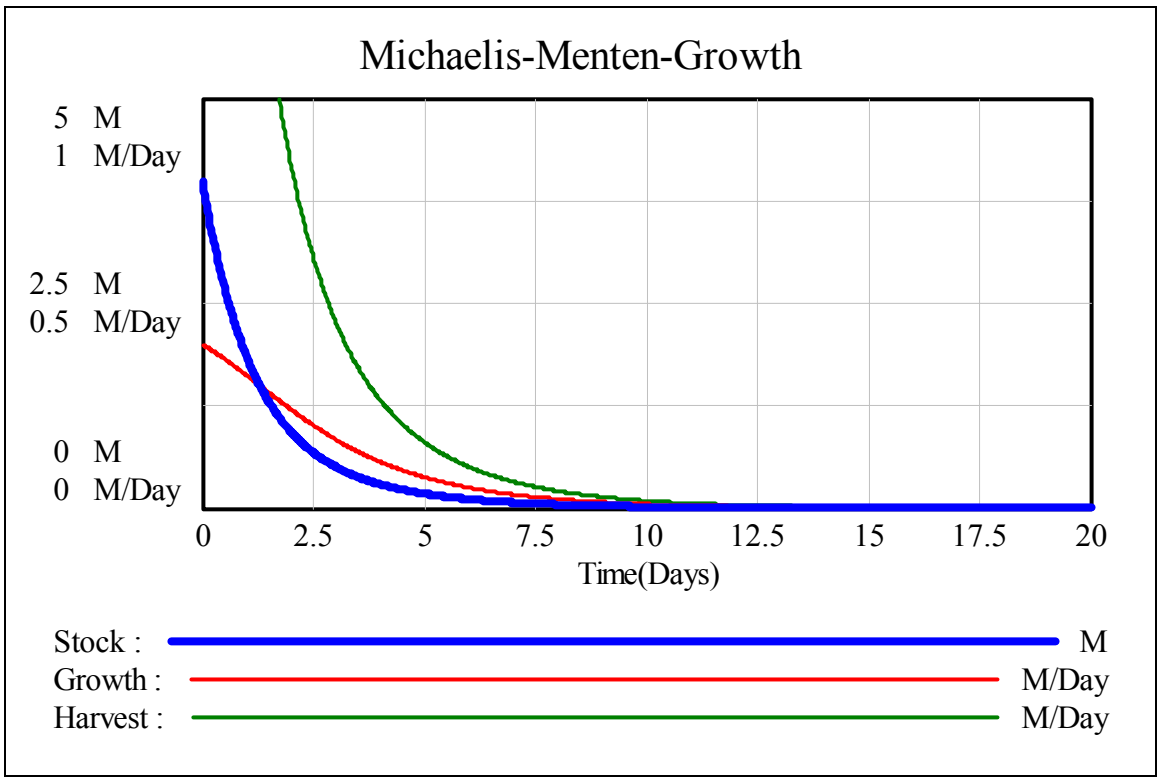
(1) What will happen if there is no *Harvest* (?



No *harvest* (HARVEST RATE = 0)

(2) What will happen if the HARVEST RATE is (too) high? (e.g. $r = 0.5$, $h = 0.9$ and INI STOCK = 4)

As the next diagram is showing very clearly – and not surprisingly – the stock is breaking down very fast.



Bossel suggests further investigations connected with this model:

Investigate the behaviour of the system for

(a) various GROWTH RATES r with constant HARVEST RATE $h > 0$,

(b) various HARVEST RATES h with constant GROWTH RATE r,

(c) various HALF SATURATION CONSTANTS c.

Sliders would be ideal for all these investigations. But various *Stock*-curves on the same axes for a series of parameters are also very convincing. For this is the VECTOR-command of *DERIVE* an excellent tool.

Treating the model with *DERIVE* as a differential equation

We are using the CAS to find the equilibrium value (= saturation value).

$$S^* = \frac{c \cdot (r - h)}{h}$$

$$\text{SOLVE}\left(S = S + r \cdot S \cdot \left(1 - \frac{S}{S + c}\right) - h \cdot S,\ S\right)$$

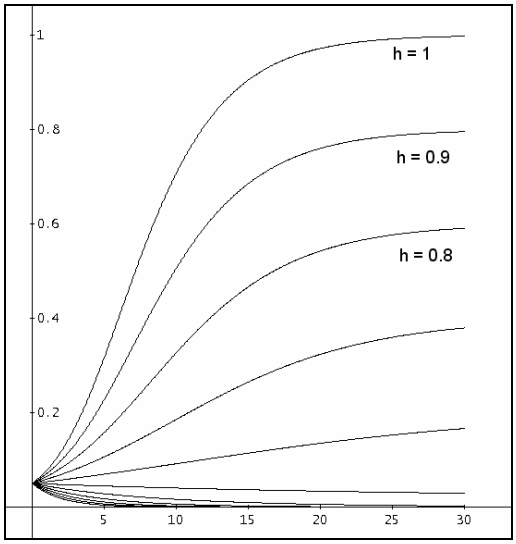
$$S = \frac{c \cdot (r - h)}{h} \ \vee\ S = 0$$

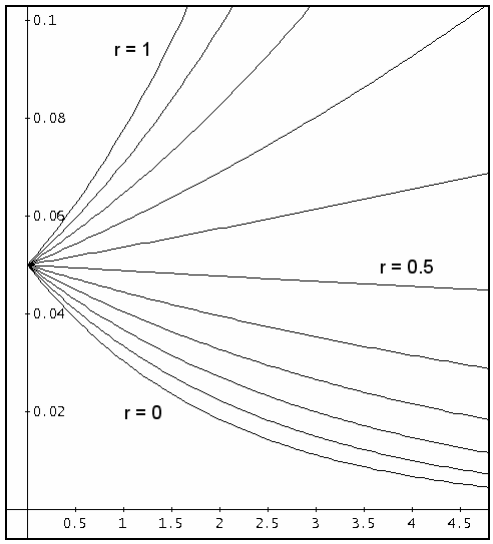
For the numerical solution of the DE and subsequently for the graphic representation of the integral curve(s) we use again the Runge-Kutta-method:

$$\text{MMG}(r, h, c, dt, n) := \text{RK}\left(\left[r \cdot S \cdot \left(1 - \frac{S}{S + c}\right) - h \cdot S\right], [t, S], [0, dt], dt, \frac{n}{dt}\right)$$

```
VECTOR(MMG(r, 0.5, 1, 0.05, 30), r, 0, 1, 0.1)
```

The next graph shows the *Stock* with variable GROWTH RATES r with $0 \le r \le 1$ ($h = 0.5$ and $c = 1$). The left graph gives details for the first years.



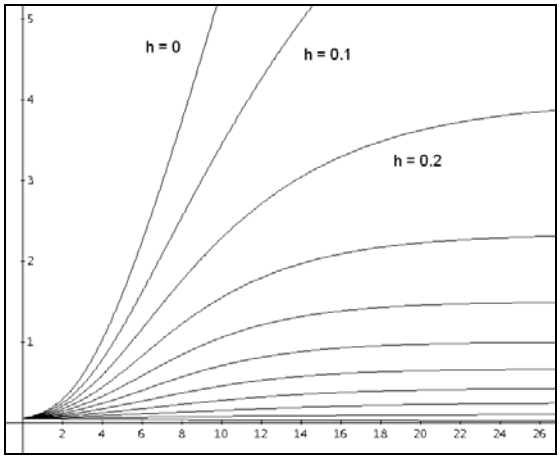


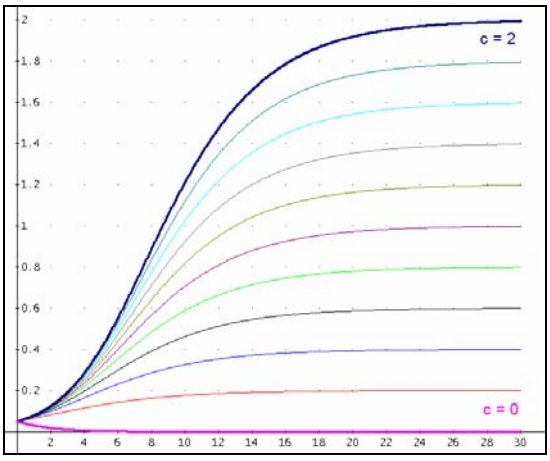
Now I will vary the HARVEST RATE h between 0 and 1 ($r = c = 1$):

One investigation is still missing: what is the influence of the constant c with $0 \le c \le 2$ ($r = 1$ and $h = 0.5$) on the *Stock* amount? See the respective VECTOR-commands and the plots.

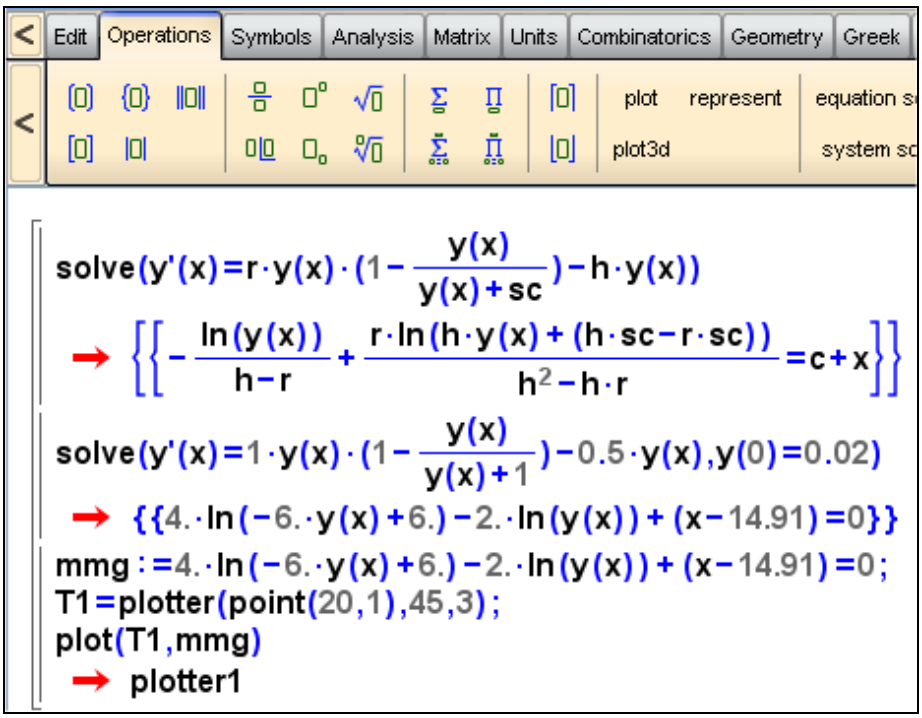
```
VECTOR(MMG(1, h, 1, 0.05, 30), h, 0, 1, 0.1)
```

```
VECTOR(MMG(1, 0.5, c, 0.05, 30), c, 0, 2, 0.2)
```

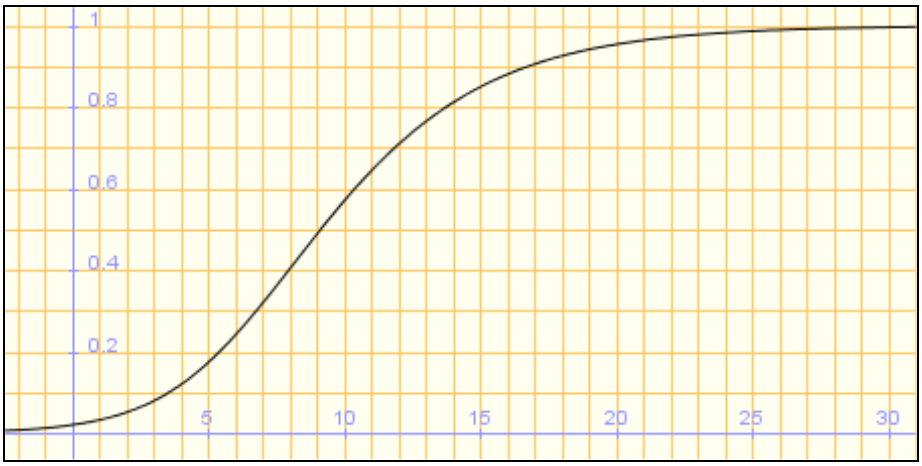




Just for fun I will grab one more tool of my tool box. I will try solving the differential equation analytically launching *WIRIS*[8].



c is the integration constant



It seems to work, so I will repeat this experiment with *DERIVE*. My hope is that I then could be able to work with my precious sliders.

$$\text{DSOLVE1\_GEN}\left(h \cdot S - r \cdot S \cdot \left(1 - \frac{S}{S + c}\right),\ 1,\ t,\ S,\ k\right)$$

$$\frac{r \cdot \text{LN}(S \cdot h + c \cdot (h - r))}{h \cdot (h - r)} + \frac{\text{LN}(S)}{r - h} - t = -k$$

This is not so bad, we have the general solution in implicit form.
We substitute $S(t = 0) = i$ (for *initial value*) in order to obtain immediately the value for the integration constant k.

$$\frac{r\cdot LN(i\cdot h + c\cdot(h - r))}{h\cdot(h - r)} + \frac{LN(i)}{r - h} - 0 = -k$$

$$\frac{r\cdot LN(c\cdot(h - r) + h\cdot i)}{h\cdot(h - r)} + \frac{LN(i)}{r - h} = -k$$

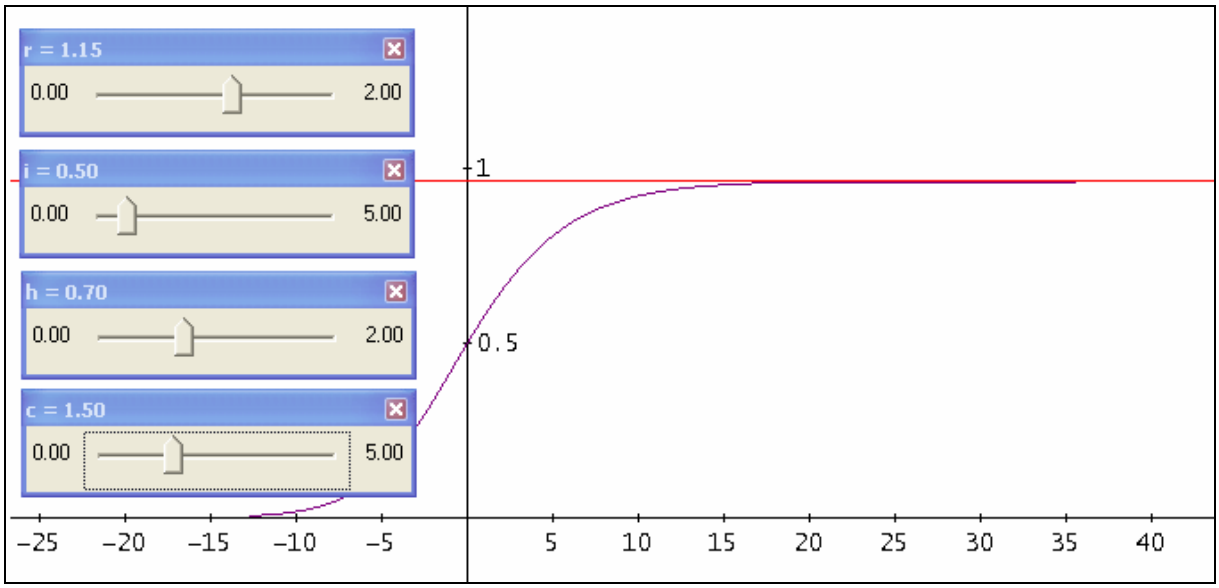
$$\frac{r\cdot LN(S\cdot h + c\cdot(h - r))}{h\cdot(h - r)} + \frac{LN(S)}{r - h} - t = \frac{r\cdot LN(c\cdot(h - r) + h\cdot i)}{h\cdot(h - r)} + \frac{LN(i)}{r - h}$$

$$\frac{r\cdot LN(y\cdot h + c\cdot(h - r))}{h\cdot(h - r)} + \frac{LN(y)}{r - h} - x = \frac{r\cdot LN(c\cdot(h - r) + h\cdot i)}{h\cdot(h - r)} + \frac{LN(i)}{r - h}$$

$$\frac{c\cdot(r - h)}{h}$$

The next to last expression is the solution (in implicit form) and the last expression is the equilibrium.

Now I can introduce sliders for *r*, *c*, *h* and *i* and plot the solution curve together with the equlibrium line.



We can do all investigations and the effort – which was not too great – was indeed worthwhile!

This is really great. Then I came across some strange results. I tried to find an explicit solution:

$$\text{SOLVE}\left(\frac{r\cdot LN(y\cdot h + c\cdot(h - r))}{h\cdot(h - r)} + \frac{LN(y)}{r - h} - x = \frac{r\cdot LN(c\cdot(h - r) + h\cdot i)}{h\cdot(h - r)} + \frac{LN(i)}{r - h},\ y\right)$$

$$y = \frac{e^{x\cdot(h^2/r - h)}\cdot(c\cdot(h - r) + h\cdot i)}{h} + \frac{c\cdot(r - h)}{h} \wedge y = i$$

I received a result – but the graph of this function does not match with the integration curve from above. I went on and did some manual manipulations to obtain a nicer form of the implicit solution and solved again for *y*. There was again a pretty result – but the graph didn't fit!

$$\text{SOLVE}\left(\left(\frac{c\cdot h - c\cdot r + h\cdot y}{c\cdot h - c\cdot r + h\cdot i}\right)^{r}\cdot\left(\frac{i}{y}\right)^{h} = e^{x\cdot(h^{2} - h\cdot r)},\ y\right)$$

$$y = \frac{e^{-h\cdot x}\cdot(c\cdot(h - r) + h\cdot i)}{h} + \frac{c\cdot(r - h)}{h} \wedge y = i\cdot e^{-h\cdot x}$$

I substituted the parameter values given in the second model and solved again for *y*. There were two solutions and – interestingly enough – one of them was the right curve. I don't have an explanation for so many contradictions. Do you have one? Then please let me know.

$$\left(\frac{1\cdot 0.5 - 1\cdot 1 + 0.5\cdot y}{1\cdot 0.5 - 1\cdot 1 + 0.5\cdot 0.02}\right)^{1}\cdot\left(\frac{0.02}{y}\right)^{0.5} = e^{x\cdot(0.5^{2} - 0.5\cdot 1)}$$

$$\frac{5\cdot\sqrt{2}\cdot(1 - y)\cdot\text{SIGN}(y)}{49\cdot\sqrt{y}} = e^{-x/4}$$

$$\text{SOLVE}\left(\frac{5\cdot\sqrt{2}\cdot(1 - y)\cdot\text{SIGN}(y)}{49\cdot\sqrt{y}} = e^{-x/4},\ y\right)$$

$$y = -\frac{e^{-x/2}\cdot(49\cdot\sqrt{(200\cdot e^{x/2} + 2401)} - 100\cdot e^{x/2} - 2401)}{100} \vee \left(y = \frac{e^{-x/2}\cdot(49\cdot\sqrt{(200\cdot e^{x/2} + 2401)} + 100\cdot e^{x/2} + 2401)}{100} \wedge y \neq 0\right)$$

Additional comment: Only the special combinations of *r* and *h* lead to an explicit form.

One can find very fine descriptions of the Michaelis-Menten-Kinetics among others in:

http://www.isitech.com/fileadmin/pb/pdf-Dateien/Michaelis_Menten_Kinetik.pdf (German)

http://www.ncbi.nlm.nih.gov/books/NBK22430/

http://depts.washington.edu/wmatkins/kinetics/michaelis-menten.html

http://www.cdnmedhall.org/dr-maud-menten

It is very interesting that the German paper contains simulations performed with *VENSIM*.

[8] http://wiris.schule.at/de_en/index.html

7 A Brusselator? Never heard!

Yeah, what is this, a *Brusselator*? As I never came across this term I did some Internet research.

I found in [9] that this is a „simple simulation of a chemical reaction with oscillating dynamics“. This made me much wiser?

You can find a very fine description in [10].

Finally I found out the orgin of the name *Brusselator*:

The **Brusselator** is a theoretical model for a type of autocatalytic reaction. The Brusselator model was proposed by Ilya Prigogine and his collaborators at the Free University of Brussels. It is a portmanteau of Brussels and oscillator.
(Wikipedia)

The Brusselator is described by a system of differential equations:

$$\dot{x} = A - (B+1)\cdot x + x^2 \cdot y$$
$$\dot{y} = B \cdot x - x^2 \cdot y$$

A and B are given constant concentrations and x and y are intermediate products. Their behaviour during the chemical reaction is investigated and simulated.

I will do it here from the other direction and will start solving the DE-system numerically.

The *WIRIS*-solution

```
library
library
  f=a-(b+1)·x+x^2·y
  g=b·x-x^2·y
library
  a=1;b=3;x0=1;y0=4
plotter(point(12,2),26,8);
bru1:=rk4(f,g,0,x0,y0,0.004,6000);
list_to_points(bru1,1,3,3,blue) → plotter1
plotter(point(2,2),5,6);
bru1:=rk4(f,g,0,x0,y0,0.004,6000);
list_to_points(bru1,2,3,3,red) → plotter1
```

[9] http://mscerts.programming4.us/de/912086.aspx
[10] http://www.bibliotecapleyades.net/archivos_pdf/brusselator.pdf

Behind the first – unopened – "library" is hidden my program for the Runge-Kutta-method.

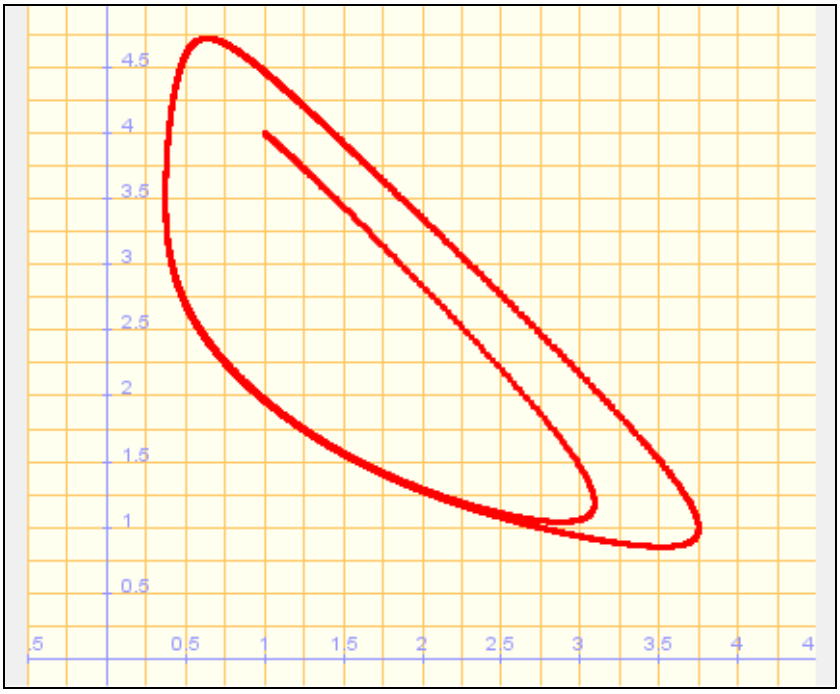
In *WIRIS* is this algorithm – in contrary to *DERIVE* – not implemented.

The second "library" contains both first derivatives, the third one the parameter values which are used by *Bossel* as initial values. The contents of the libraries can be used globally in the whole *WIRIS*-session – and not only in the single paragraphs which are kept together within one (left) bracket [.

I was able to calculate and plot 6000 points using a step width of 0.004. When asking for more points no plot is appearing. Later we will try whether *DERIVE* has more power?

The first plot shows how *y* develops durting the first 24 time units (seconds).

The next plot gives the phase diagram for the *x*- and *y*-values.

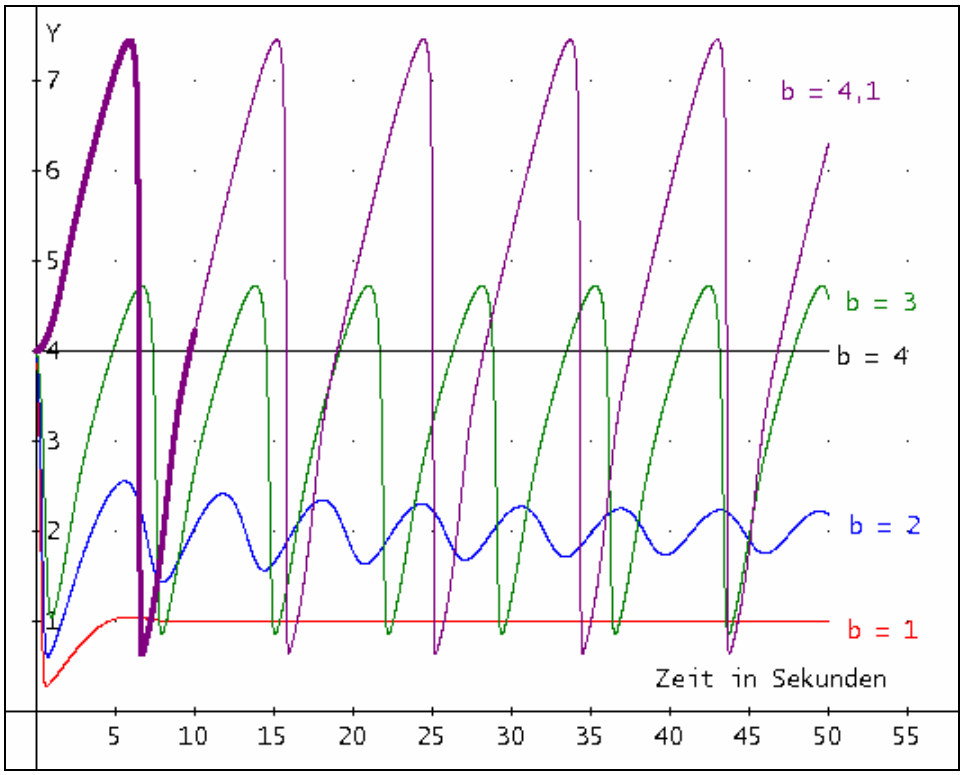


Treatment with *DERIVE* and then with *TI-NspireCAS*

I am using my own Runge-Kutta-routine `rk4` and let plot the *t*-*y*-diagram for various *B*-values.
(My rk4 works faster than RK.)

```
#2:   [a := 1, b := 1, x_ := 1, y_ := 4]

#3:   (rk4([a - (b + 1)·x + x^2·y, b·x - x^2·y], [t, x, y], [0, x_, y_], 0.01, 5000))↓↓[1, 3]

#4:   b := 2

#5:   b := 3

#6:   b := 4

#7:   b := 4.1

#8:   (rk4([a - (b + 1)·x + x^2·y, b·x - x^2·y], [t, x, y], [0, x_, y_], 0.001, 10000))↓↓[1, 3]
```

Expression #3 is high lighted and plotted ($B = 1$), then I define $B = 2$, plot #3 again, etc. I use a step width of 0.01. *Bossel* works with *VENSIM* taking a step width 0.002.

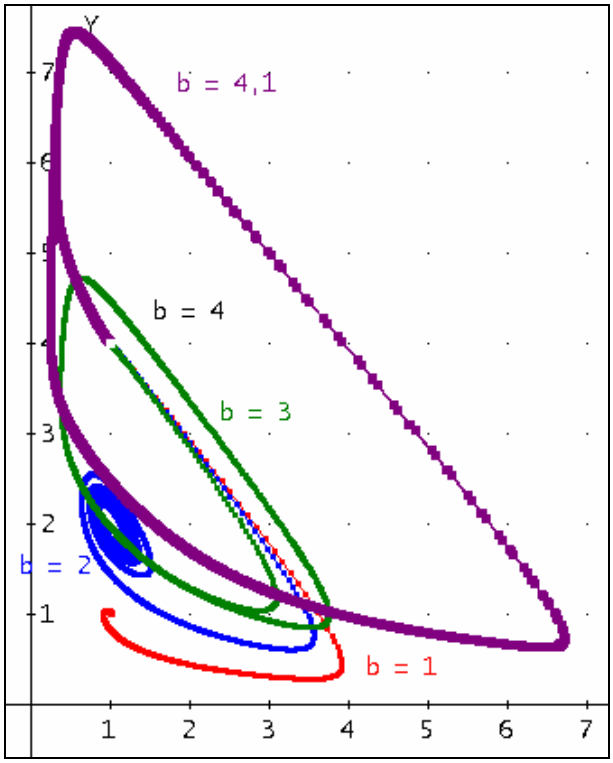


In expression #8 I took step width 0.001, too in order to compare the accuracy. As one can clearly see the bold line (step width $h = 0.001$) is more or less identical with the $h = 0.01$ generated function graph.
The *VENSIM*-diagrams (later in this chapter) look quite the same.

We can see that the curve changes from a damped oscillation to very pronounced oscillations.
For $B = 4$ we recognize that $y = 4$ constant.

Leaving all parameters the same then we plot the respective phase diagrams:

```
#9:   b := 1

#10:  (rk4([a - (b + 1)·x + x^2·y, b·x - x^2·y], [t, x, y], [0, x_, y_], 0.01, 5000))↓↓[2, 3]

#11:  b := 2

#12:  b := 3

#13:  b := 4

#14:  b := 4.1
```



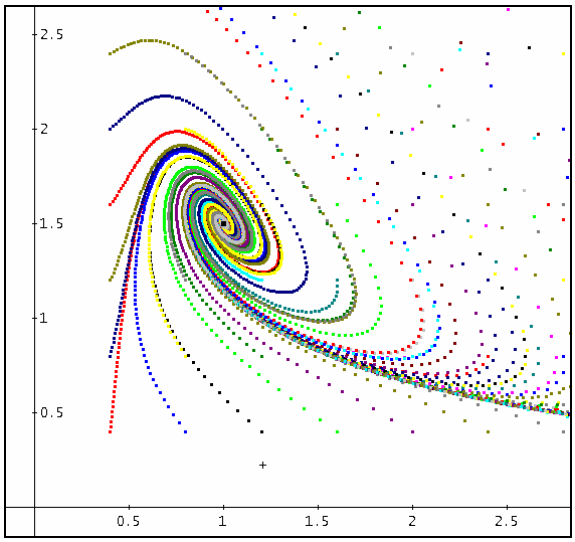
For accentuating the point (1,4) – phase diagram for $B = 4$ – I plotted this point in white colour.

Bossel produces with *VENSIM* a little bit complicated so called "state pictures" (in German: „Zustandsbilder" = families of phase diagrams). The initial values for x and y are running through an interval.

We can do this with *DERIVE* using a nested VECTOR-command for $(0.4 \le x, y \le 4)$. Of course, this needs some calculation time but the plots are really convincing.

```
#15:  b := 1.5

#16:  VECTOR(VECTOR((rk4([a - (b + 1)·x + x^2·y, b·x - x^2·y], [t, x, y], [0, u, v], 0.05,
        1000))↓↓[2, 3], u, 0.4, 4, 0.4), v, 0.4, 4, 0.4)

#17:  [1, 1.5]
```

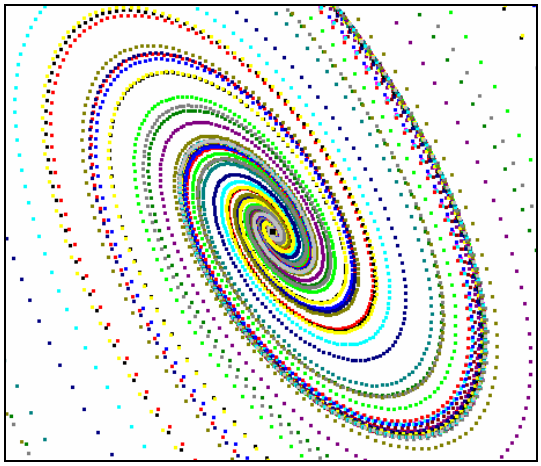
The points are represented “not connected”.

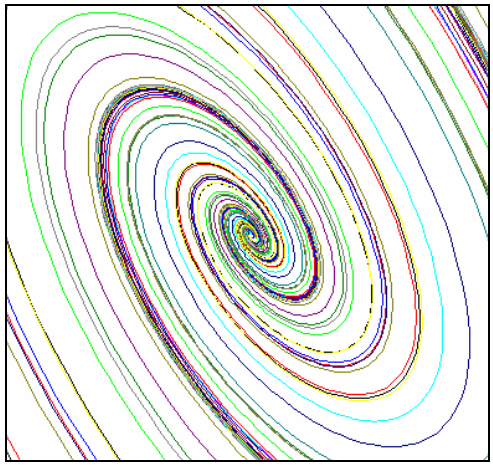


The equilibrium point is easy to find. The derivatives are set to zero and the resulting system must be solved for x and y:

$$0 = A - (B+1)\cdot x + x^2 \cdot y$$
$$0 = B \cdot x - x^2 \cdot y$$

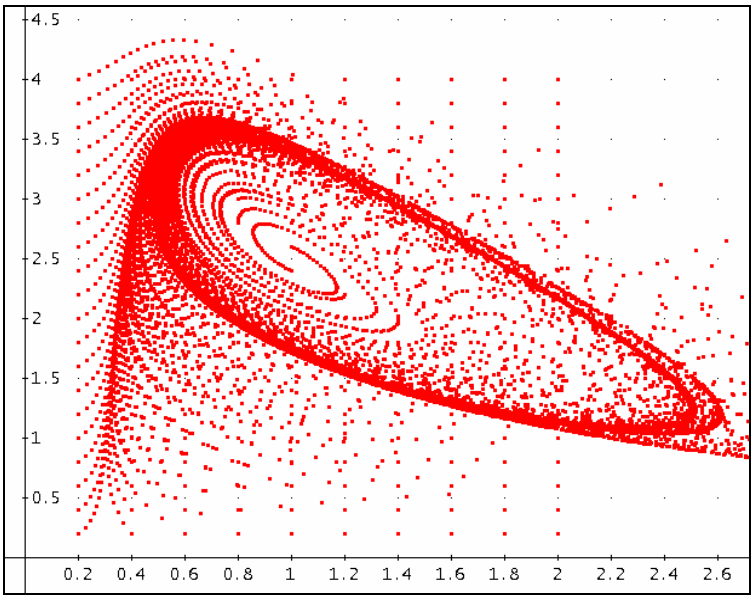
Even without using a CAS the equilibrium point results as $\left(A, \frac{B}{A}\right)$. Following *Bossel* it can be shown easily that this point for $B < A^2 + 1$ appears as a stable whirl, which can be seen very clearly in the state picture from above. The equlilibrium point (1, 1.5) is the black point in the center of the whirl. The next two pictures are zooming into the interior of the whirl with points connected in the right graph displaying the phase diagram curves.





For $B > A^2 + 1$ we have a limit cycle, which contains an unstable equilibrium point in its interior, which is the point (1, 2.5) for the choice $A = 1$ and $B = 2.5$.

```
#18:  b := 2.5

#19:  VECTOR(VECTOR((rk4([a - (b + 1)·x + x^2·y, b·x - x^2·y], [t, x, y], [0, u, v], 0.05,
        1000))↓↓[2, 3], u, 0.2, 4, 0.2), v, 0.2, 4, 0.2)
```



Unstable Equilibrium point in (1,2.5)

There is no denying that these pictures have some esthetic charm.
Later we will see that they can be produced with *VENSIM*, too. However, this needs some efforts.

Of course, we would benefit of the application of sliders because then it would be very comfortable to study the influence of the various parameters.

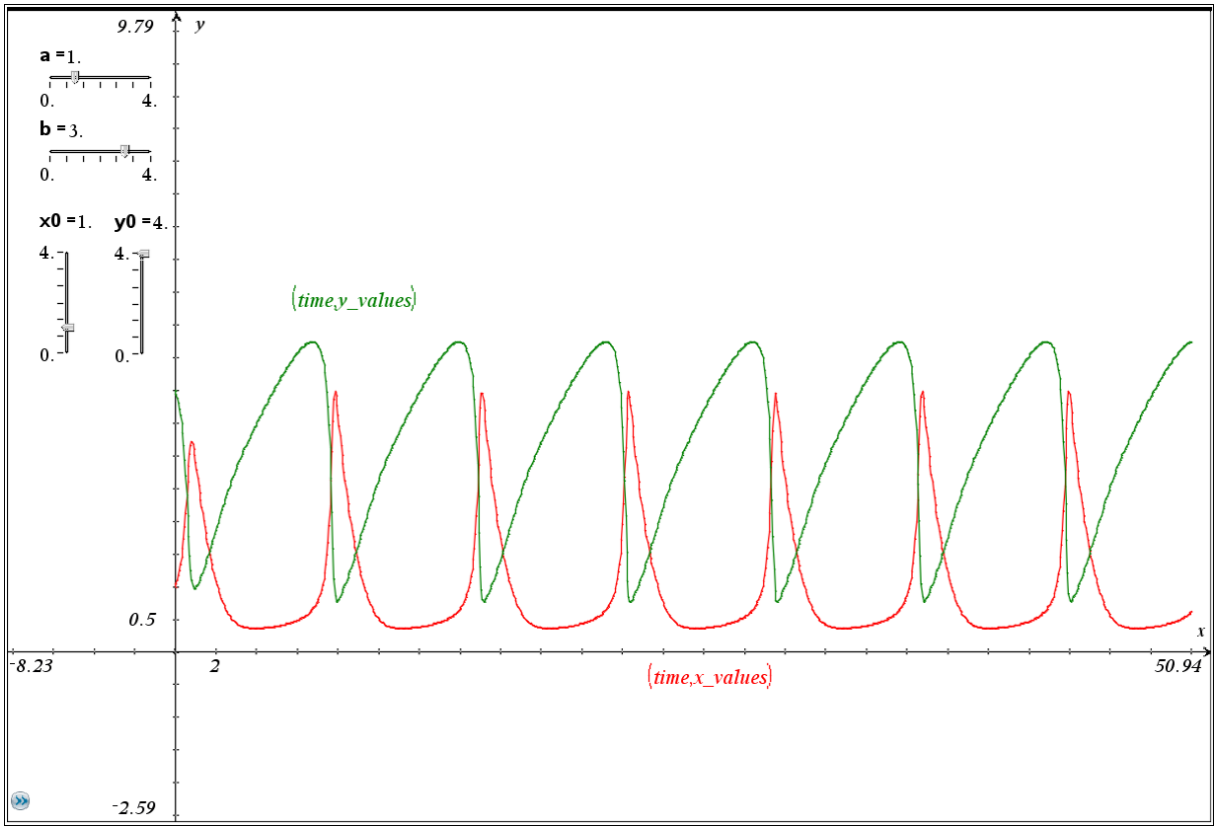
Sliders with *TI-NspireCAS*:

	A	B	C time	D x_val...	E y_val...	F	G	H
1	dt	0.05	0	1.	4.			
2			0.05	1.05	3.95			
3			0.1	1.10774	3.88976			
4			0.15	1.17485	3.81726			
5			0.2	1.25332	3.73005			
6			0.25	1.34562	3.62508			
7			0.3	1.45469	3.49873			
8			0.35	1.58394	3.34675			
9			0.4	1.73698	3.16451			
10			0.45	1.91697	2.94767			
11			0.5	2.12518	2.69362			
12			0.55	2.35841	2.40413			

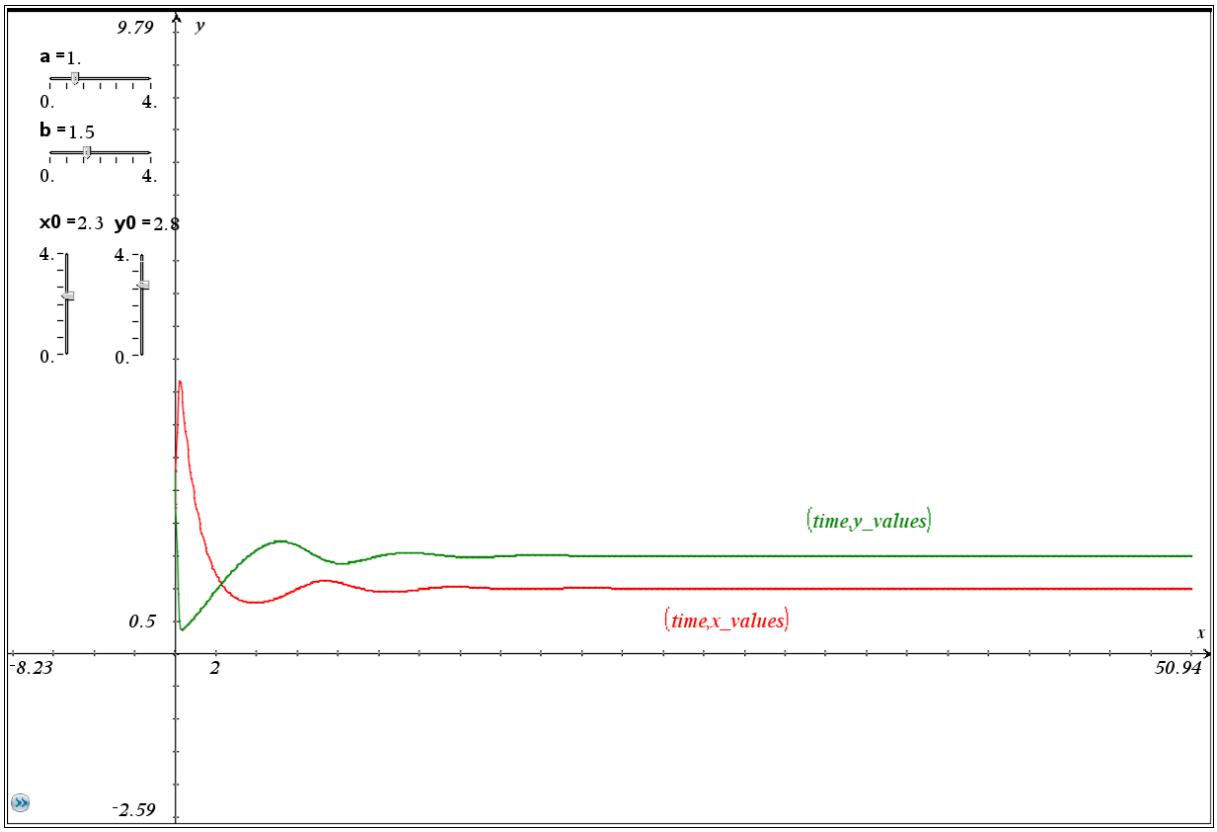
D2 $=d1+\$b\$1\cdot\left(a-\left(b+1\right)\cdot d1+d1^2\cdot e1\right)$

It does not need much effort to produce the table in the spreadsheet. Sliders for *a*, *b*, *x0* and *y*0 in the Geometry application have been installed previously.

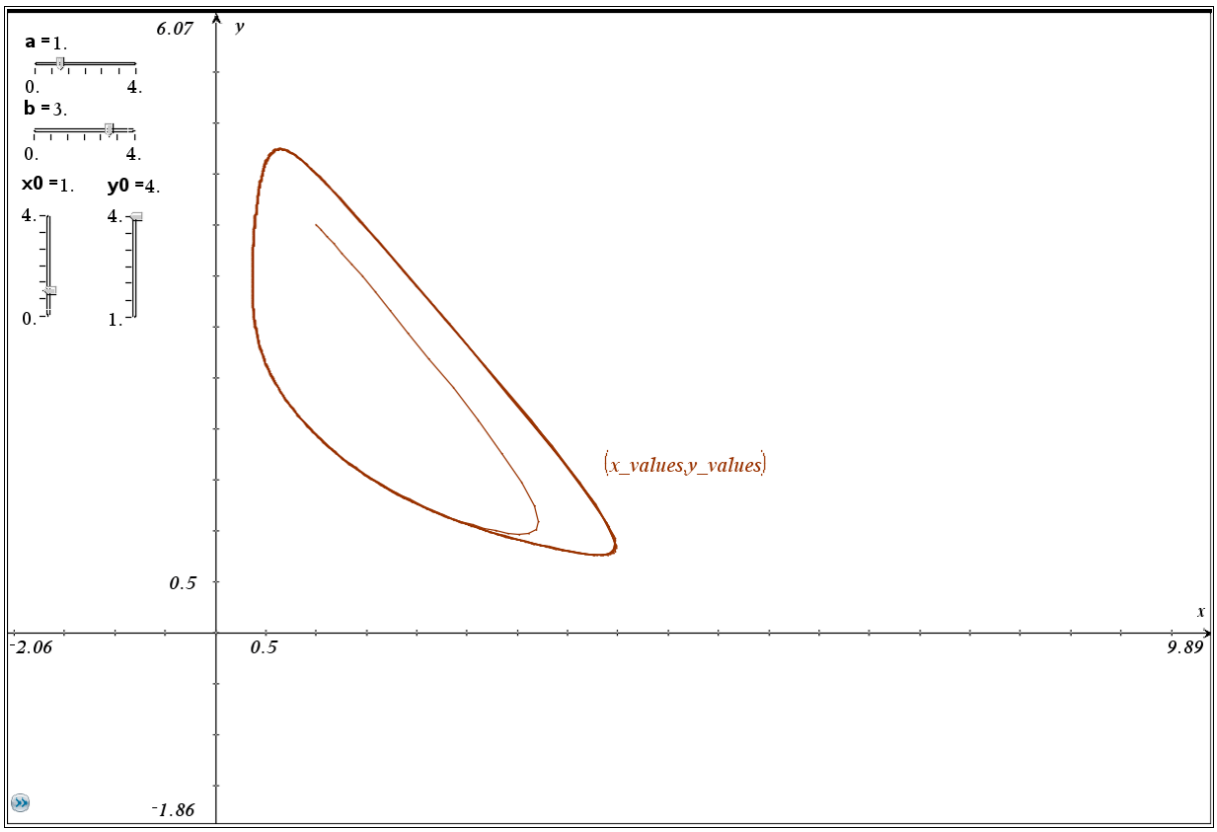
The first diagram shows how x and y develop within the first 50 seconds using the parameters chosen by *Bossel* in *System Zoo*.

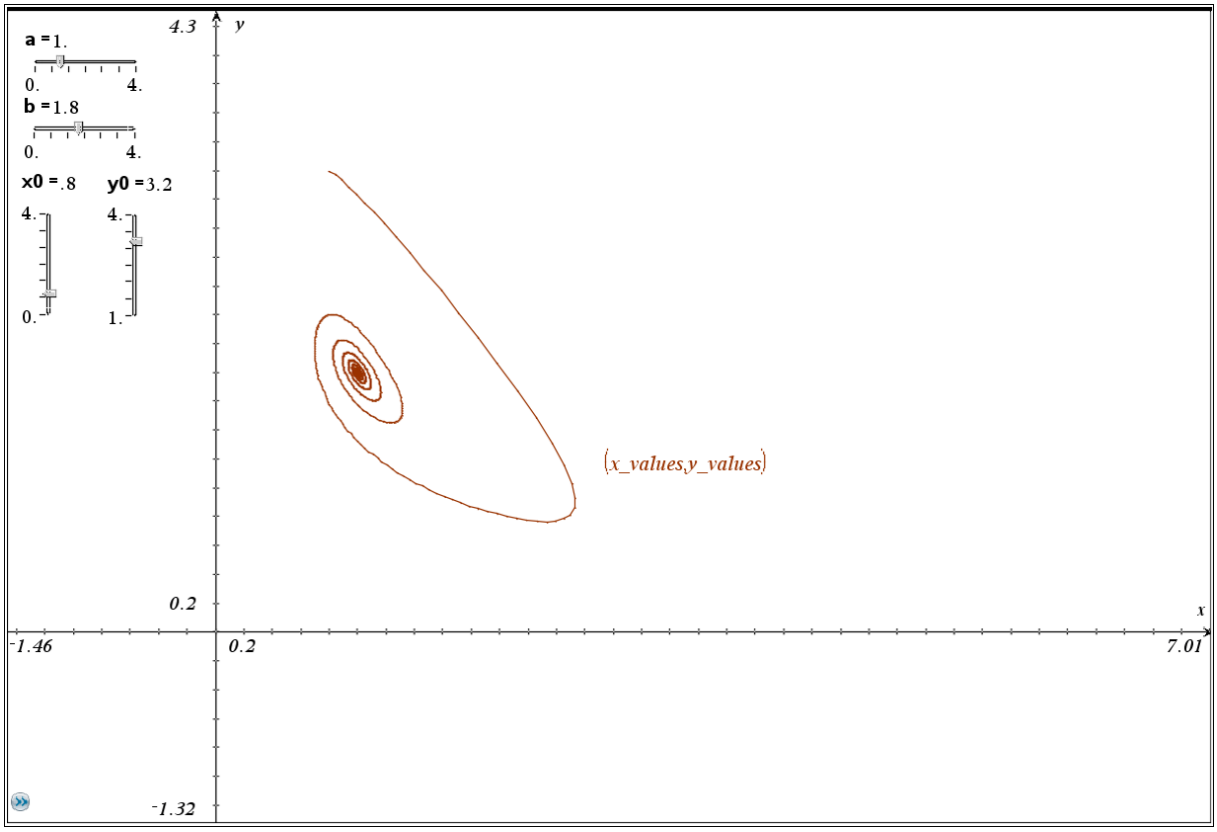


The scatter plots are reacting immediately on every change of the slider settings.



See the phase diagram for the "*System Zoo-parameters*" followed by a diagram based on another another choice of parameters.

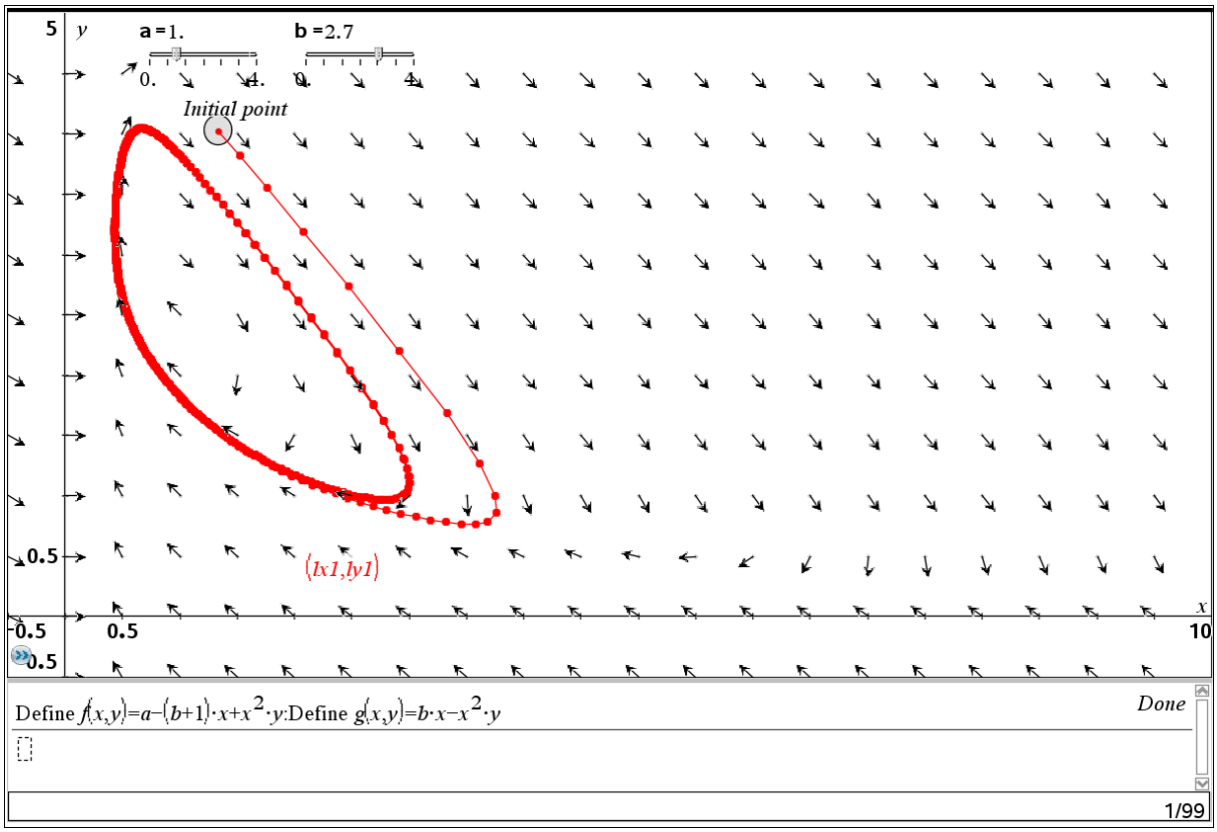




Thanks a private communication with Philippe Fortin I was able to produce another form of representation. We used a program rk4syst (Runge-Kutta) and instead of changing the initial values by sliders we can drag the initial point in the plane and additionally plot the direction field of DE system.

	A	B	C	D	E lx1	F ly1	G dt	H	I
◆				=rk4syst(x0,y0,tmi	=left(d[],dim(d	=right(d[],dim			
1	x0	**1.33621**	initial value of x, at t=tmin	1.52973	1.52973	3.81253	0		
2	y0	**4.02749**	initial value of y, at t=tmin	1.78349	1.78349	3.52623	0.05		
3	xmin	**-0.5**	windows settings	2.11425	2.11425	3.14838	0.1		
4	xmax	**10**	windows settings	2.52714	2.52714	2.66977	0.15		
5	ymin	**-0.5**	windows settings	2.98844	2.98844	2.12062	0.2		
6	ymax	**5**	windows settings	3.40729	3.40729	1.59146	0.25		
7	xg	**0.5**	first x−tickmark	3.68513	3.68513	1.1856	0.3		
8	yg	**0.5**	first y−tickmark	3.7971	3.7971	0.935935	0.35		
9	numpoin...	**800**	number of computed points b...	3.78623	3.78623	0.806803	0.4		
10				3.7055	3.7055	0.750031	0.45		
11	tmin	**0**		3.59071	3.59071	0.732314	0.5		
12	tmax	**40**		3.46148	3.46148	0.735207	0.55		
13				3.3277	3.3277	0.749245	0.6		
14	lv	**0.3**	control the length	3.19427	3.19427	0.769632	0.65		
15			of vectors	3.06358	3.06358	0.793899	0.7		
16				2.93674	2.93674	0.82074	0.75		
17				2.81429	2.81429	0.849436	0.8		
18				2.6964	2.6964	0.879581	0.85		
19				2.58308	2.58308	0.910933	0.9		
20				2.47425	2.47425	0.943341	0.95		

I20

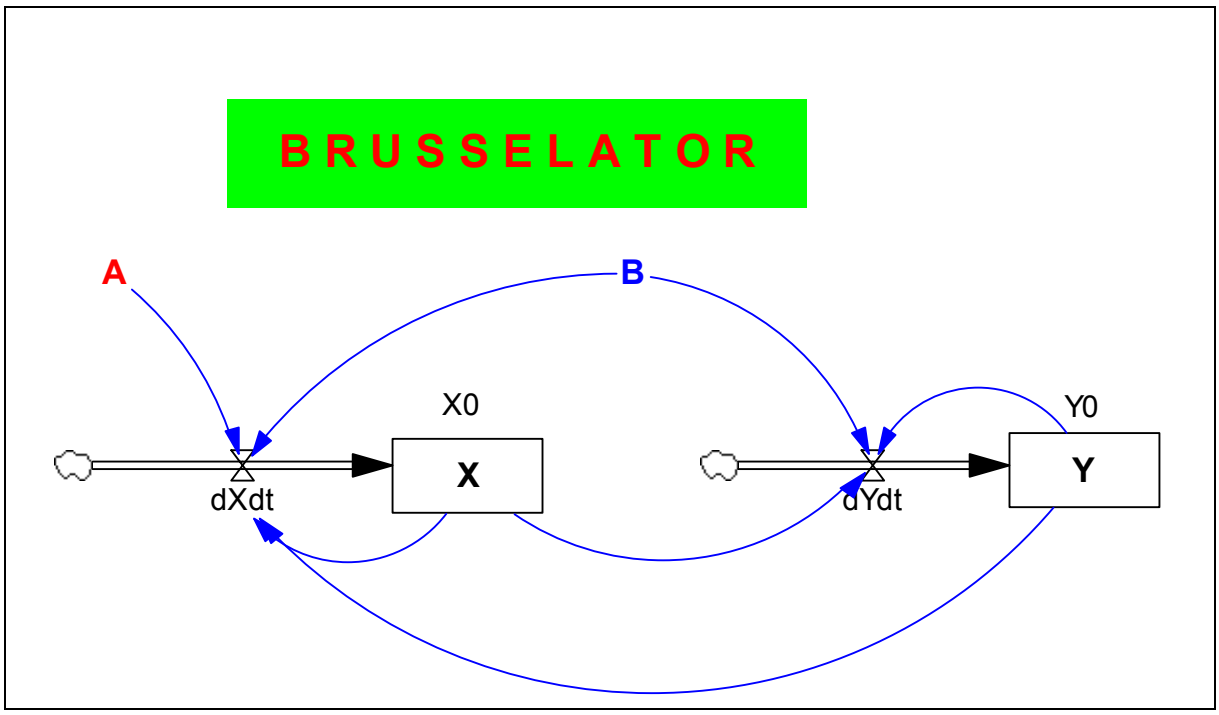


In another Geometry window we could see the time graphs.

But finally we will work with *Vensim PLE*, too.

The model is very simple and so it is done quickly. (Production of the “state pictures“ is a little bit tricky!)

The Brusselator with *Vensim PLE*



(01) A = 1

(02) B = 3

(03) dXdt = A – (B+1) ∗ X+X ∗ X ∗ Y

(04) dYdt = B ∗ X – X ∗ X ∗ Y

(05) FINAL TIME = 50

(06) INITIAL TIME = 0

(07) SAVEPER = 0.1

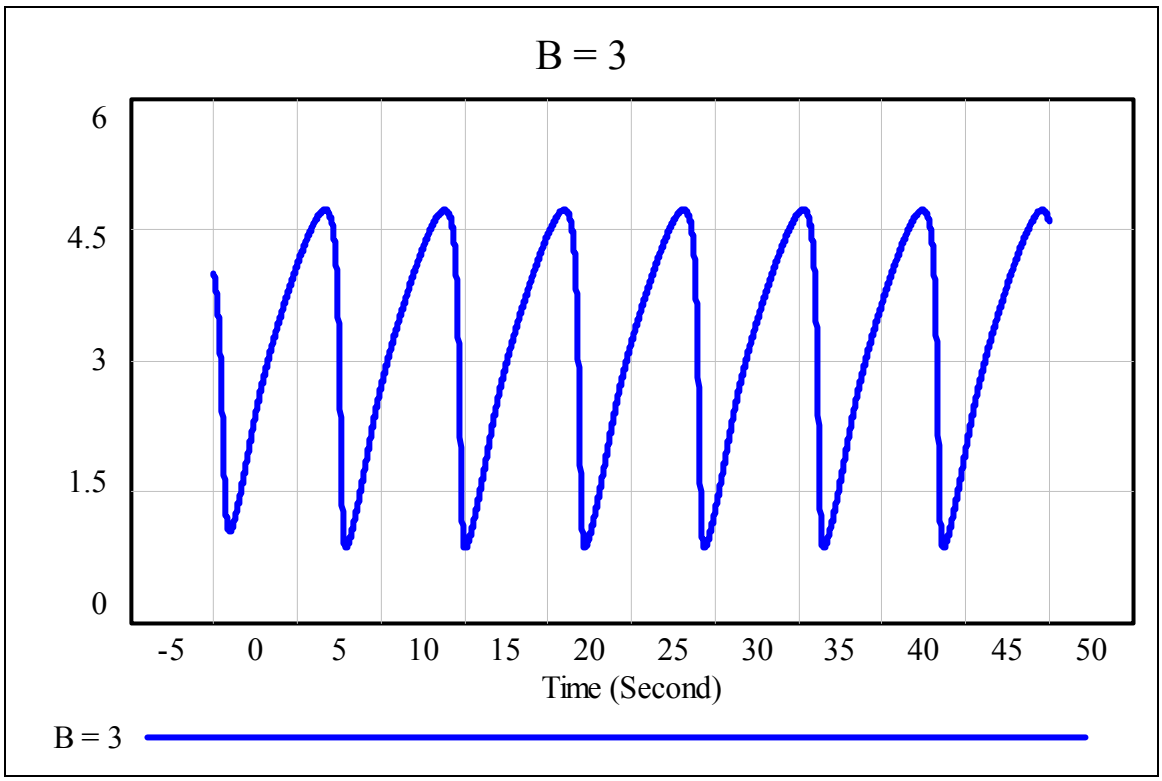
(08) TIME STEP = 0.05

(09) X= INTEG(dXdt, X0)

(10) X0 = 1

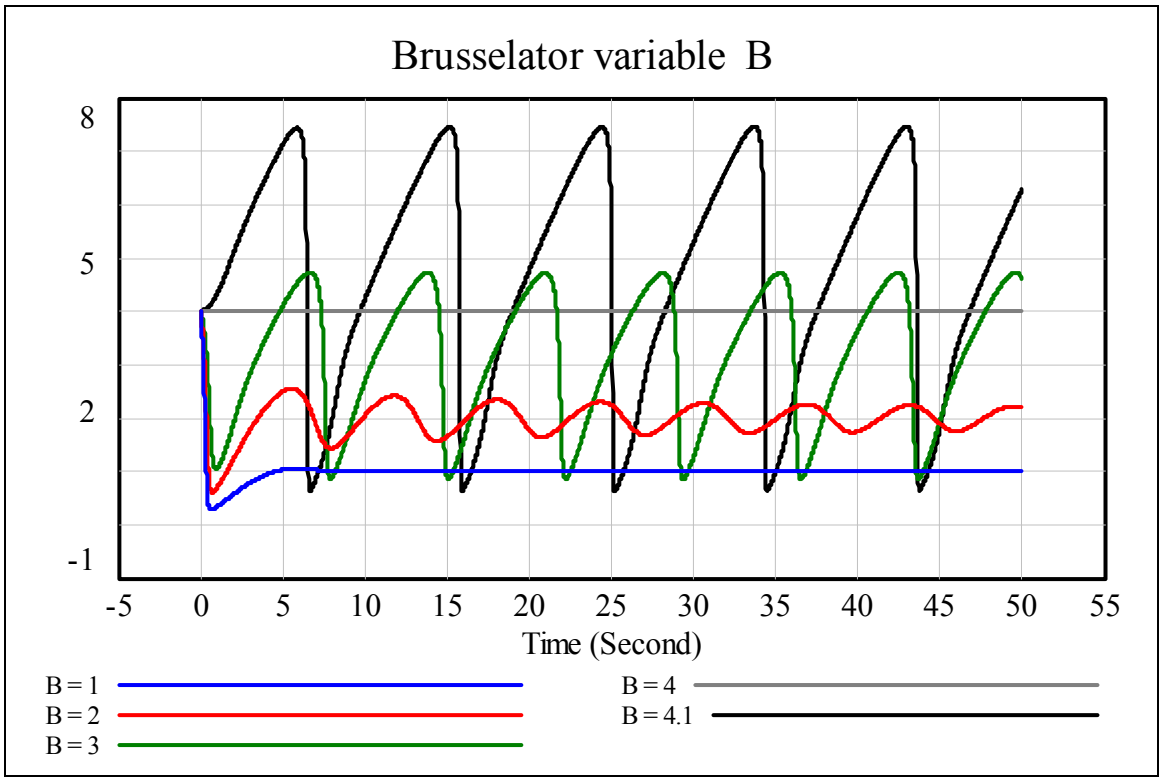
(11) Y = INTEG(dYdt, Y0)

(12) Y0 = 4



Bossel uses the Euler-method with an increment of 0.001 (diagram above). I increased the step width up to 0.05 and did not recognize any change in the graph. So my simplification in the *TI-Nspire*-Version is now justified more or less.

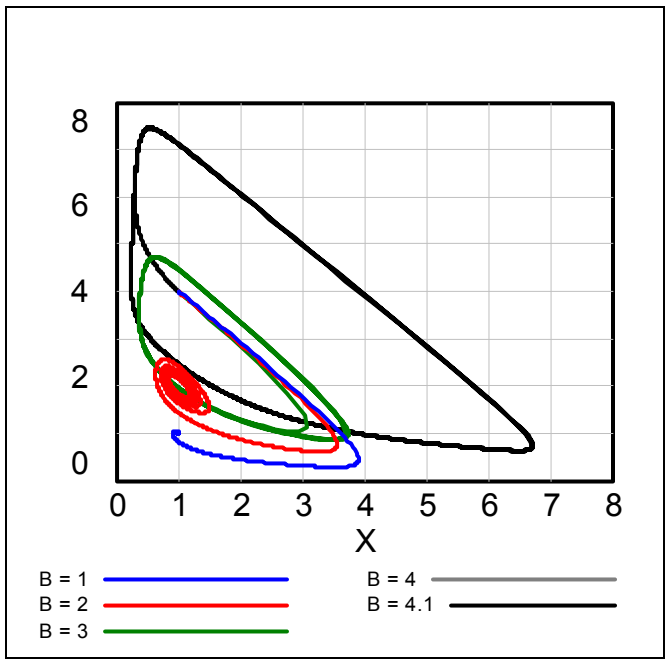
The next diagram shows the comparison for various values for *B*.



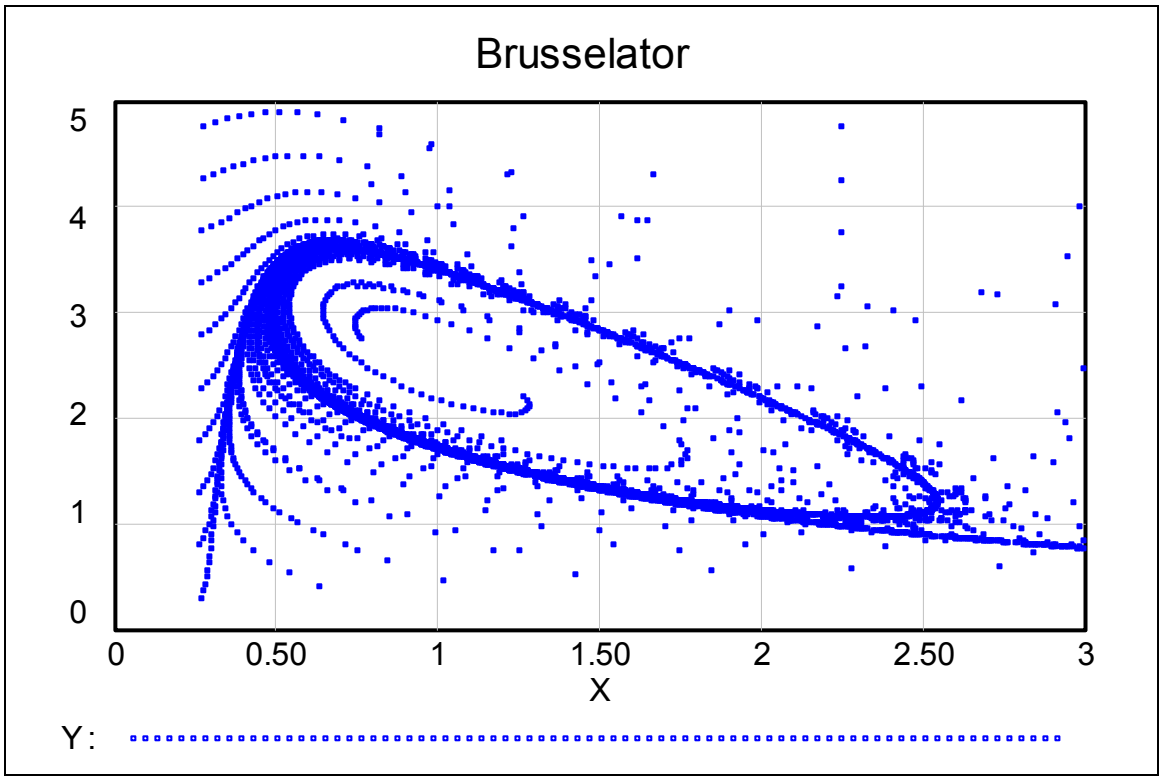
See on the right hand side the phase diagrams.

Applying a trick (an additional module) *Bossel* makes possible a double loop for the *x*- and *y*-values for a grid of 10 by 10 initial conditions for *x* and *y*.

I cannot explain this now but invite you studying *System Zoo 1*.



You should have seen a plot very similar to the following one (but in red) earlier!



8 Bistable Oscillator

Prior to treating the "*bistable oscillator*", I will explain the *linear oscillator* very shortly because it is the base concept of the oscillator.

The general form of the *linear oscillator* is described by the system of differential equations

$$\dot{x} = a\,x + b\,y$$
$$\dot{y} = c\,x + d\,y$$

A bistable oscillator can look like this

A well kown case of this linear oscillator is the *spring equation*:

$$\dot{x} = y$$
$$\dot{y} = -k \cdot x$$

For the non physics – like me: x is the excursion, y is the velocity and k is the spring constant. We can consider a damping parameter d. Then the respective system reads:

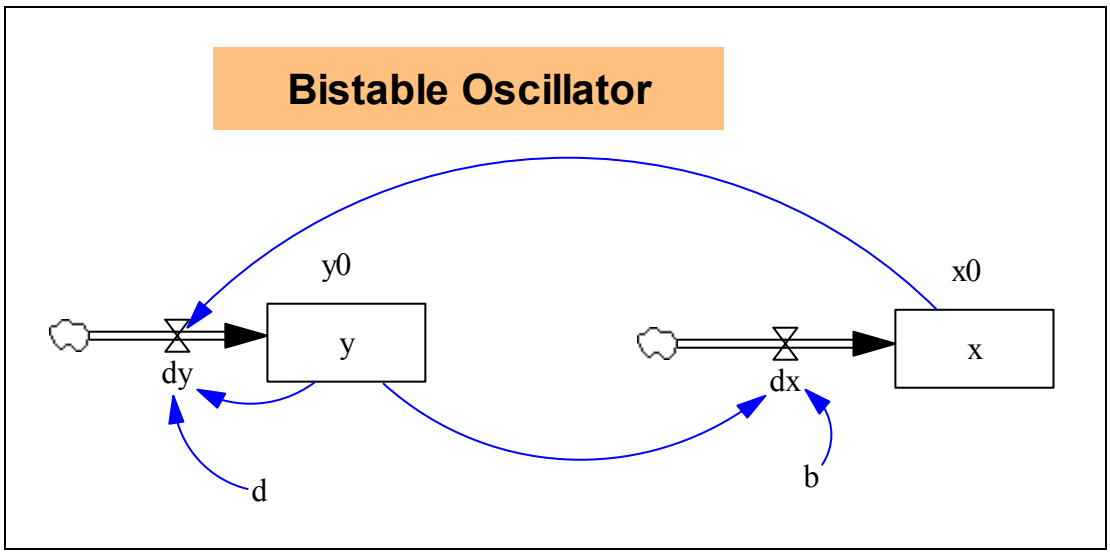
$$\dot{x} = y$$
$$\dot{y} = -k \cdot x - d \cdot y$$

Introducing additionally a non linear coupling we obtain e.g.

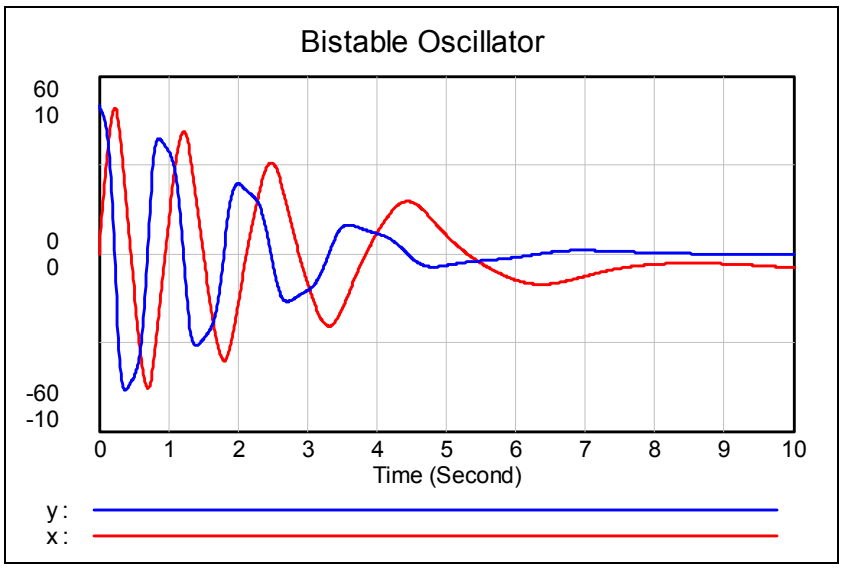
$$\dot{x} = b \cdot y$$
$$\dot{y} = x - x^3 - d \cdot y$$

with a coupling parameter b and a damping parameter d. And this is our bistable oscillator. Its name will explain itself later.

I start with the *VENSIM PLE* model and a very simple designation of the variables.



The following graph shows the behaviour of the excursion X and veleocity Y for the first 10 seconds with an initial velocity $Y0 = 50$.



The describing document is very short, of course:

(01) b = 1

(02) d = 1

(03) dx = b*y

(04) dy = x – x^3 – d*y

(05) FINAL TIME = 20

(06) INITIAL TIME = 0

(07) SAVEPER = TIME STEP

(08) TIME STEP = 0.01

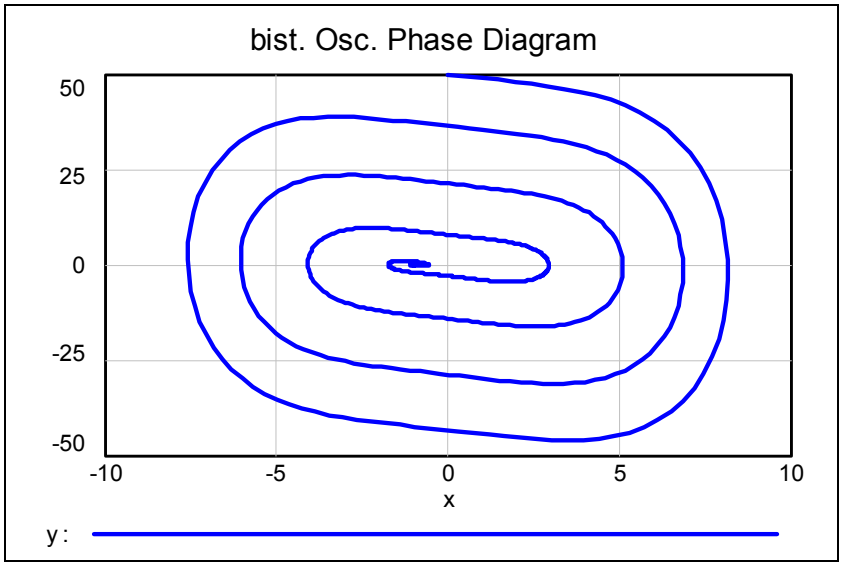
(09) x = INTEG(dx,x0)

(10) x0 = 0

(11) y = INTEG(dy,y0)

(12) y0 = 50

We take a look on the phase diagram. What do you think about the end behaviour?



Inspecting the table for this diagram our impression is increased that there is an equilibrium point at ($x = 0, y = -1$).

bist Phase

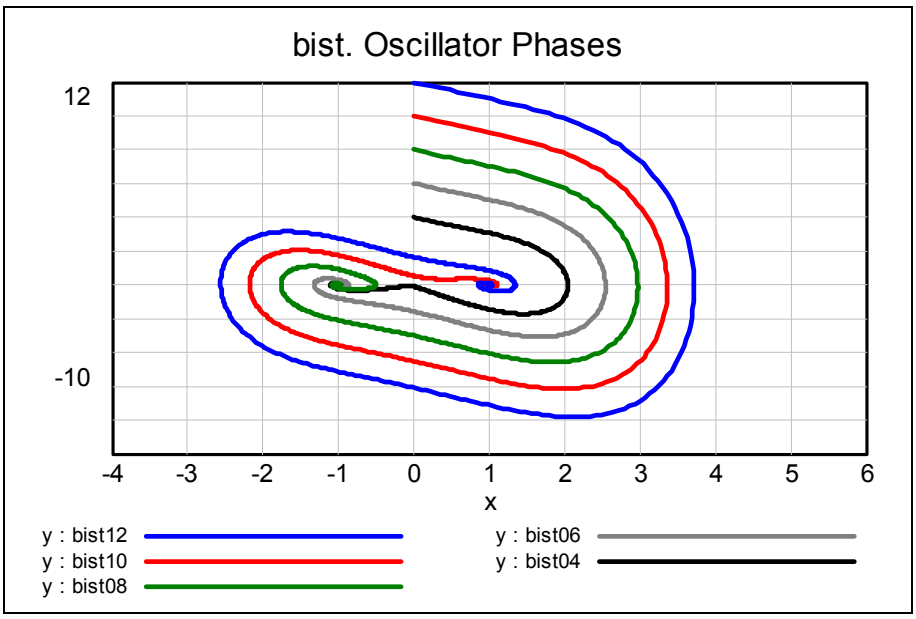
Time (Second)	x	y
19.7804	-0.9992	-0.0025
19.7904	-0.9992	-0.0025
19.8004	-0.9992	-0.0025
19.8104	-0.9993	-0.0025
19.8204	-0.9993	-0.0024

Analytical search for a possible equlibrium point is not difficult. We solve the system for x and y.

$$0 = b \cdot y$$
$$0 = x - x^3 - d \cdot y$$

And we obtain three solutions: $x_1 = 0$, $x_2 = 1$ und $x_3 = -1$. All y-coordinates are 0.

The system oscillates dependent on the initial velocity to (+1,0) or (-1,0). This is easy to see by running the simulation for a sequence of y-values and collecting all phase diagrams in one plot.



Later I will show the global behaviour for $-2.50 \le x \le 2.50$ und $-2.50 \le y \le 2.50$ working with *DERIVE* and using its VECTOR-command.

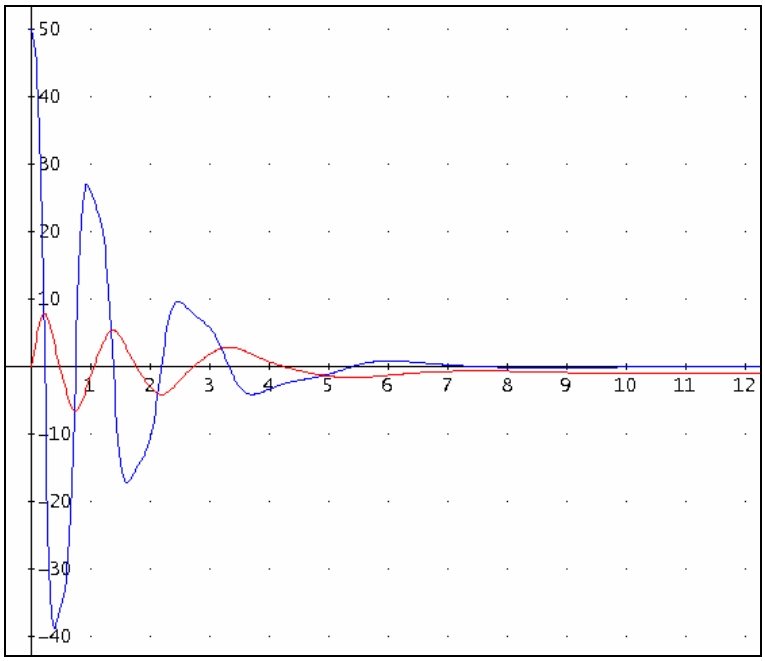
Speaking about *DERIVE* we will take a turn to this CAS and we will try reproducing this not too difficult simulation.

The Bistable Oscillator with *DERIVE*

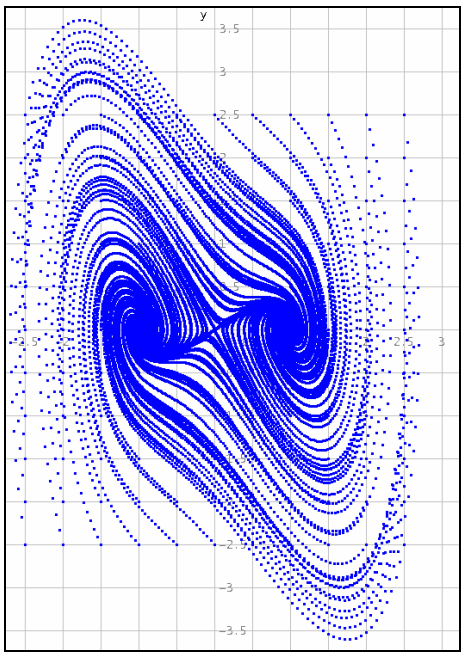
Five *DERIVE*-expressions lead to perfect graphs:

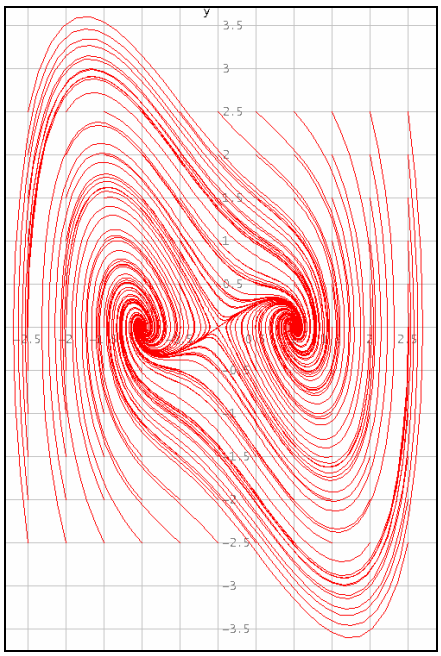
```
bist_osc(b, d, x0, y0, dt, n) := RK([b·y, x - x^3 - d·y], [t, x, y], [0, x0, y0], dt, n)
(bist_osc(1, 1, 0, 50, 0.02, 1000))↓↓[1, 3]
(bist_osc(1, 1, 0, 50, 0.02, 1000))↓↓[1, 2]
VECTOR(VECTOR((bist_osc(1, 1, x0, y0, 0.02, 500))↓↓[2, 3], x0, -2.5, 2.5, 0.5), y0, -2.5, 2.5, 0.5)
VECTOR(VECTOR((bist_osc(1, 1, x0, y0, 0.05, 200))↓↓[2, 3], x0, -2.5, 2.5, 0.5), y0, -2.5, 2.5, 0.5)
```

The first expression defines a function for numerical solving the DE-system. Expressions #2 and #3 give the time-graphs of *X* and *Y* (same colours as in the *VENSIM*-treatment).



The next two expressions produce the "state pictures" presenting a family of phase diagrams.





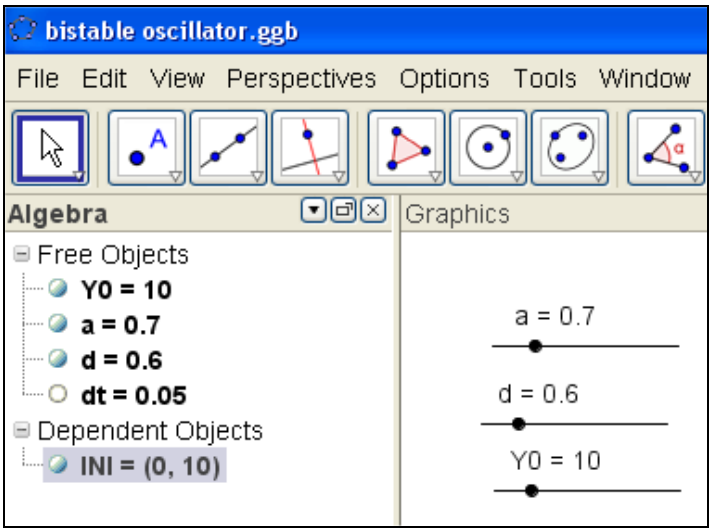
One can recognize very clear the three equilibrium points: *stable whirls* at (± 1,0) and at (0,0) an *instable saddle*. Of course, one could experiment changing both parameters *b* and *d*. We will do this now working with sliders.

The *GeoGebra*-model

I didn't use *GeoGebra* for a while in this text. However, this dynamic system is so simple that it can be modelled with *GeoGebra* (and its spreadsheet) without facing any problems.

I define sliders for *b*, *d* and *Y*0.

Point INI is given by variable coordinates (0, *Y*0).



I produce the respective lists in the spreadsheet according to the formulae as you can see below.

In order to obtain the table I fill the cells in the first two rows as follows:

A1:	0	time
D1:	0	*x*0 (remains constant 0)
E1:	Y0	Initial value for *y* (variable by a slider)
F1:	(D1,E1)	first point of the phase diagram
G1:	(A1,D1)	first point of the time-*x*-diagram
H1:	(A1,E1)	first point of the time-*y*-diagram

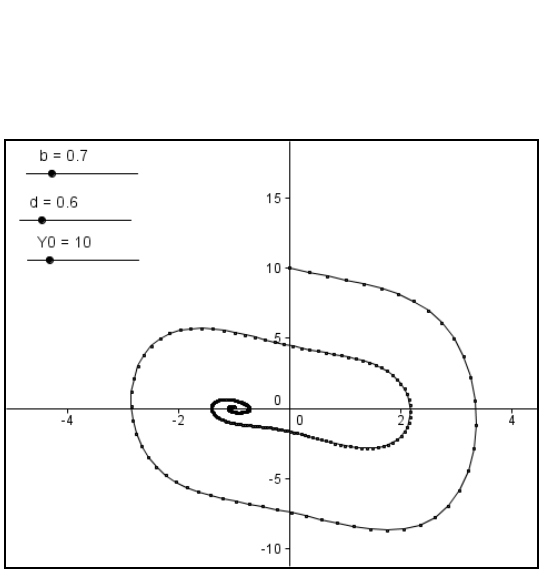
A2:	A + dt	
B2:	dt * b * E1	increase of *x*
C2:	dt * (–d * E1 + D1 – D1^3)	increase of *y*
D2:	D1 + B2	next value of *x*
E2:	E1 + C2	next value of *y*
F2:	(D2,E2)	
G2:	(A2,D2)	
H2:	(A2,E2)	
I2:	Segment(F1,F2)	

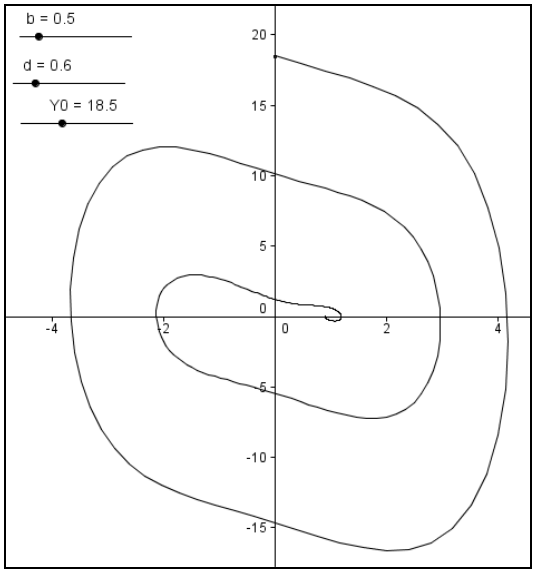
This second row is copied at least down to row 201.

Spreadsheet

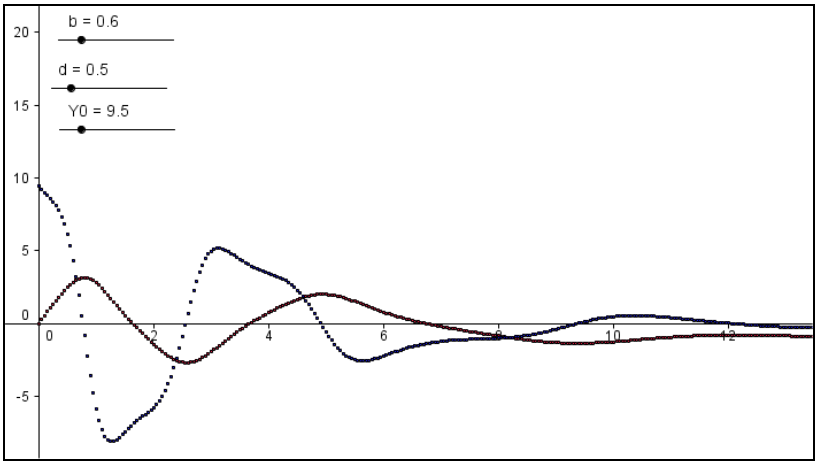
	A	B	C	D	E	F	G	H	I
1	0			0	10	(0, 10)	(0, 0)	(0, 10)	
2	0.05	0.35	-0.3	0.35	9.7	(0.35, ...	(0.05, ...	(0.05, ...	0.461
3	0.1	0.34	-0.276	0.69	9.424	(0.69, ...	(0.1, 0....	(0.1, 9....	0.437
4	0.15	0.33	-0.265	1.019	9.16	(1.019...	(0.15, ...	(0.15, ...	0.423
5	0.2	0.321	-0.277	1.34	8.883	(1.34, ...	(0.2, 1....	(0.2, 8....	0.424

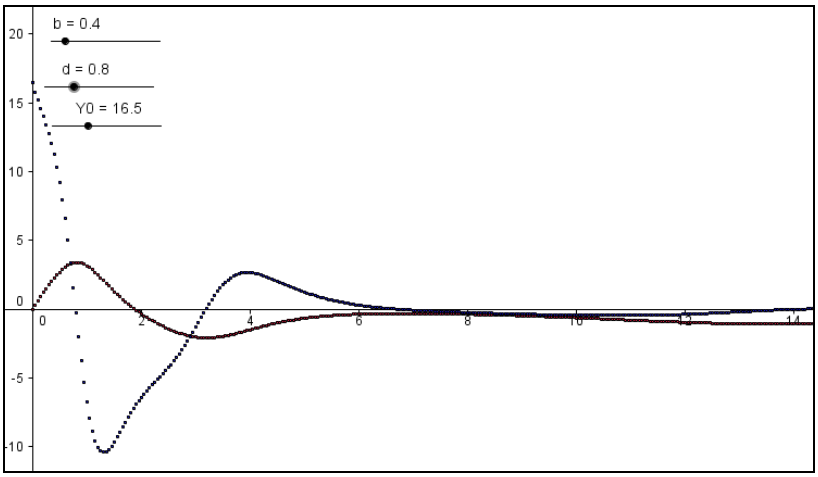
Next two diagrams give the states of *x* and *y* for different parameter values. The "bistability" can easily be observed. The right plot is consisting of the segments between the points only (column I).





You can see below the time-diagrams for two different parameter settings. The *t*-*x*-diagram has its initial point in the origin.





With *MS-Excel* and *TI-Nspire* one can work alike. Installing the sliders in *Excel* needs more efforts.

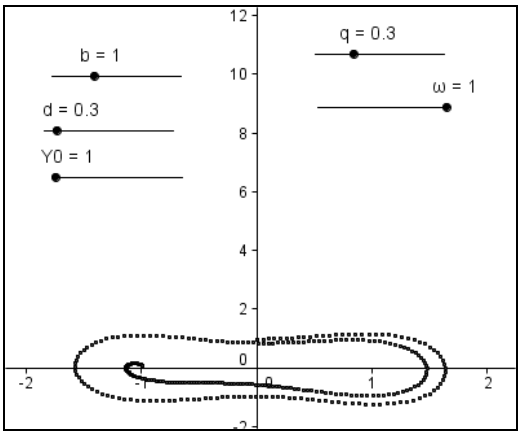
How the oscillator becomes chaotic:

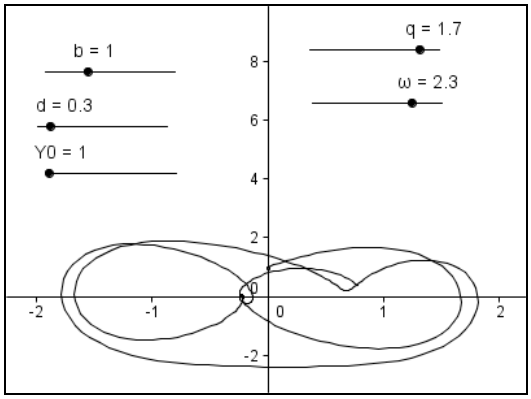
If the oscillator – its behaviour has been pretty predictable so far – will be excited from outside by another periodic oscillation it will react in varied ways which can finally end in chaos.

The differential equations system is modified:

$$\dot{x} = b \cdot y$$
$$\dot{y} = x - x^3 - d \cdot y + q \cdot \cos \omega t$$

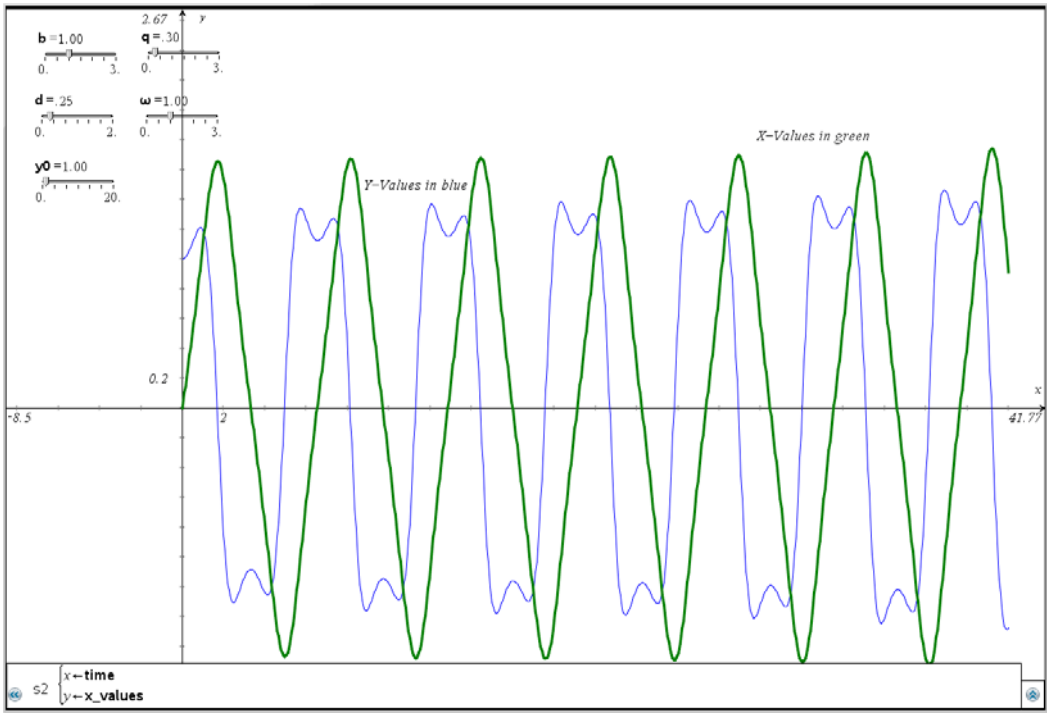
The recent *GeoGebra* version cannot work properly for a larger number of rows in the spreadsheet (>200 rows). Accepting extended calculation times one can obtain nice diagrams. (Cell C2 must be modified according to the second DE including q and ω; additional sliders are to be introduced.)





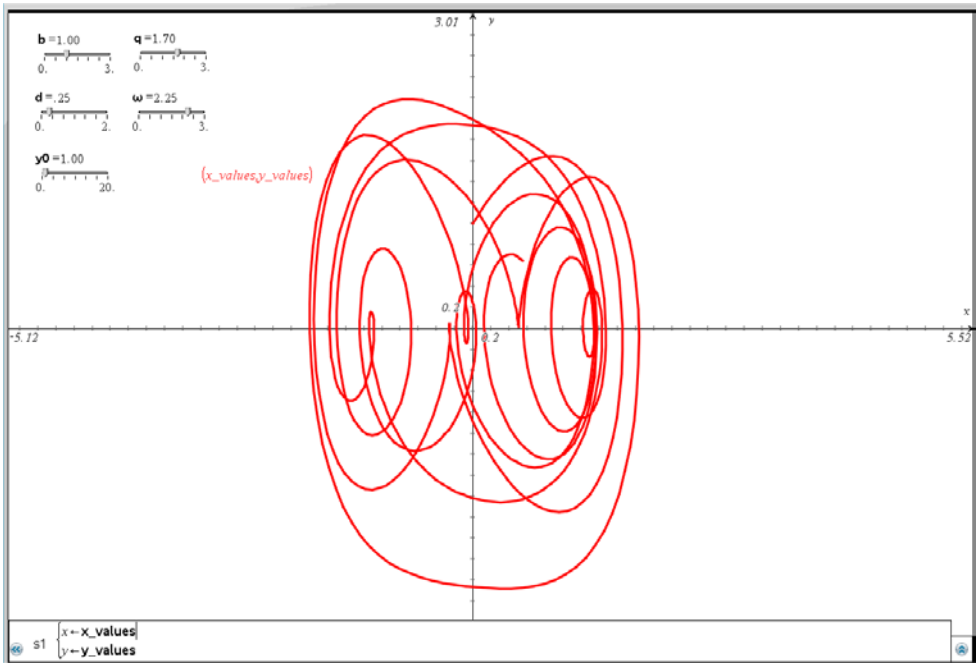
Sliders with *TI-NSpireCAS*

The spreadsheet is structured like in *GeoGebra*. Step width is predefined as $dt = 0.05$.



The time-diagrams for x and y for the first 50 time units.

The phase diagram looks quite interesting:

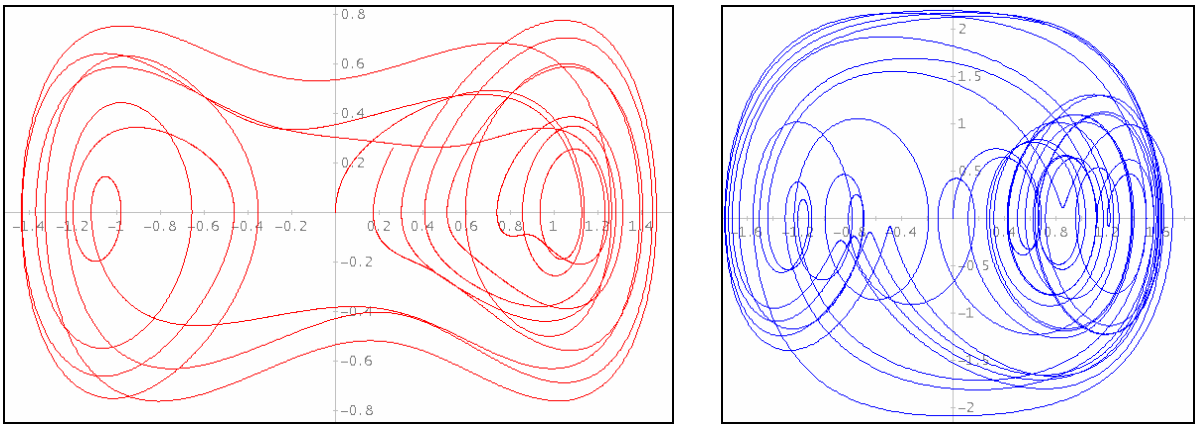


The representation becomes really attractive when animating any – does not matter which one – slider. Then an exciting "movie" is running presenting fast changing diagrams.

Static Diagrams with *DERIVE*

In *DERIVE* we use again Runge-Kutta for solving the differential equation system – but we must do without sliders. At the other hand we benefit of taking a small step width $dt = 0.01$ with a reasonable calculation time (~ 45 seconds) and receiving 10000 points which can be plotted.

```
(RK([b·y, - d·y + x - x^3 + q·COS(ω·t)], [t, x, y], [0, 0, y0], 0.01, 10000))↓↓[2, 3]
[b := 1, d := 0.25, y0 := 1, q := 0.3, ω := 1]
[b := 1, d := 0.25, y0 := 1, q := 1.7, ω := 2.3]
```



Phase diagrams for the parameter settings given above.

(One can get an idea of the two equilibrium points, but they get lost after a short while.)

9 Stock-keeping – with random numbers

Modelling with random events is not only especially attractive but also close to reality as many parameters are varying randomly within certain boundaries.

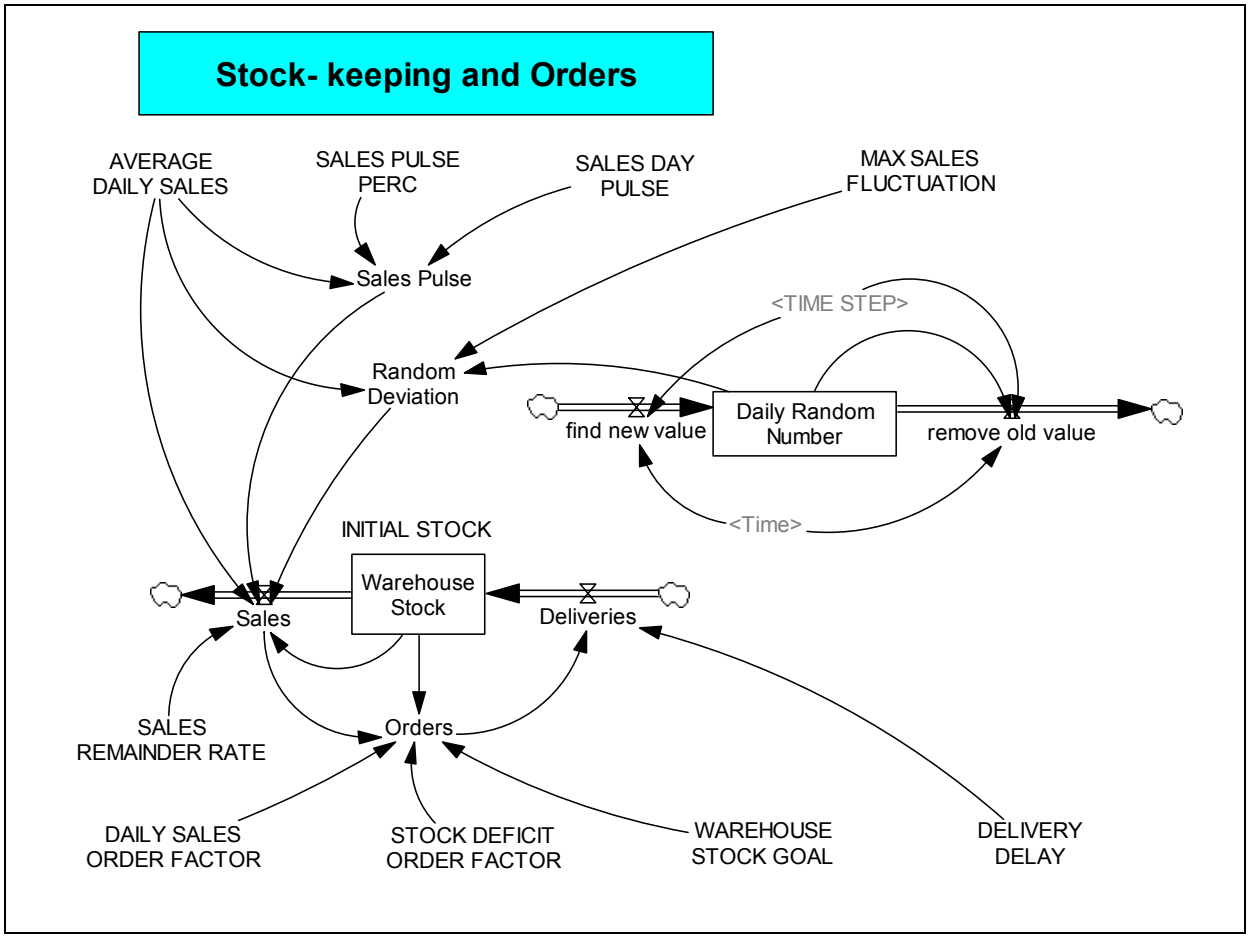
Stock-keeping is an important cost factor for many companies. A clever balance between the clients' requests – complying with orders – and keeping the stock as small as possible must be found.

I set up the model according to *Bossel* (*System Zoo 3*):

The respective *Warehouse Stock* results from time dependent quantities *Sales* and *Deliveries* (according to received *Orders*). *Orders* are based on the *Warehouse Stock* and on the actual *Sales*. Three parameters must be considered: DAILY SALES ORDER FACTOR, STOCK DEFICIT ORDER FACTOR and WAREHOUSE STOCK GOAL. An important parameter for *Deliveries* is the DELIVERY DELAY.

Varying *Sales* is defined by the *Random Deviation* of AVERAGE DAILY SALES. We use a random number (by applying a uniform distribution) for obtaining the *Daily Random Number*. Additionally given is the percentage of the sales event SALES PULSE PERC for a certain day of this event SALES DAY PULSE. Now we are able to investigate the dynamics in this system.

All other quantities which are influencing the system can be read off from the stock and flow diagram.



I show the document containing all settings and equations (including the dimensions).

Settings

FINAL TIME = 500
Units: Day
The final time for the simulation.

INITIAL TIME = 0
Units: Day
The initial time for the simulation.

SAVEPER = TIME STEP
Units: Day
The frequency with which output is stored.

TIME STEP = 0.0625
Units: Day
The time step for the simulation.

Old warehouses in Amsterdam

Parameters

AVERAGE DAILY SALES = 1000
Units: Pieces/Day

DAILY SALES ORDER FACTOR = 1
Units: 1 [0,2,0.125]

DELIVERY DELAY = 20
Units: Day [0,?]

INITIAL STOCK = 2000
Units: Pieces [0,10000,1000]

MAX SALES FLUCTUATION = 25
Units: 1 [0,50,5]

SALES DAY PULSE = 10
Units: Day

SALES PULSE PERC = 0
Units: 1 [0,50,10]
Percentage of the average daily sales

SALES REMAINDER RATE = 1
Units: 1/Day

STOCK DEFICIT ORDER FACTOR = 0.125
Units: 1/Day [0,1,0.125]

WAREHOUSE STOCK GOAL = 2000
Units: Pieces

Market Scene in Marangu, Tanzania

Dynamics (= Equations)

Daily Random Number = INTEG (+find new value – remove old value, 0)
Units: 1

Deliveries = DELAY FIXED(Orders, DELIVERY DELAY, AVERAGE DAILY SALES)
Units: Pieces/Day

find new value = IF THEN ELSE(ABS(Time – INTEGER(Time)) <= TIME STEP/2,
RANDOM UNIFORM(0, 1, 0)/TIME STEP, 0)
Units: 1/Day

Orders = IF THEN ELSE((DAILY SALES ORDER FACTOR*Sales + STOCK DEFICIT ORDER
FACTOR*(WAREHOUSE STOCK GOAL – Warehouse Stock)) > 0,
(DAILY SALES ORDER FACTOR*Sales + STOCK DEFICIT ORDER
FACTOR*(WAREHOUSE STOCK GOAL – Warehouse Stock)), 0)
Units: Pieces/Day

Random Deviation = AVERAGE DAILY SALES*(2*MAX SALES FLUCTUATION/100)*
(Daily Random Number-1/2)
Units: Pieces/Day

remove old value = IF THEN ELSE(ABS(Time + TIME STEP/2 – INTEGER(Time +
TIME STEP/2)) <= TIME STEP/2, Daily Random Number/TIME STEP, 0)
Units: 1/Day

Sales = IF THEN ELSE(AVERAGE DAILY SALES + Random Deviation + Sales Pulse <
Warehouse Stock*SALES REMAINDER RATE, (AVERAGE DAILY SALES +
Random Deviation + Sales Pulse), (SALES REMAINDER RATE*Warehouse Stock))
Units: Pieces/Day

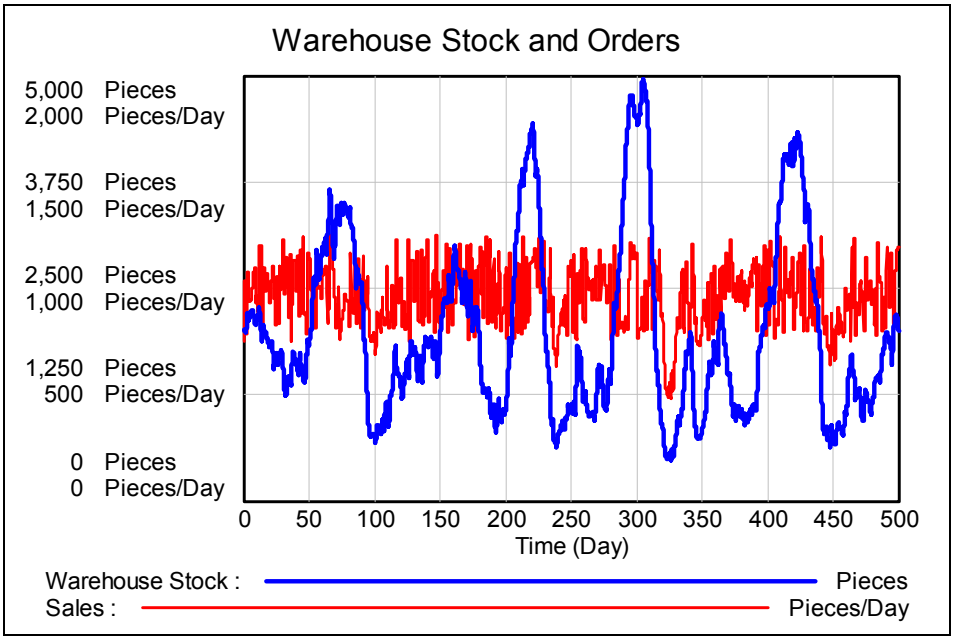
Sales Pulse = (SALES PULSE PERC/100)*AVERAGE DAILY SALES*
PULSE(SALES DAY PULSE, 1)
Units: Pieces/Day

Warehouse Stock = INTEG(+Deliveries – Sales, INITIAL STOCK)
Units: Pieces

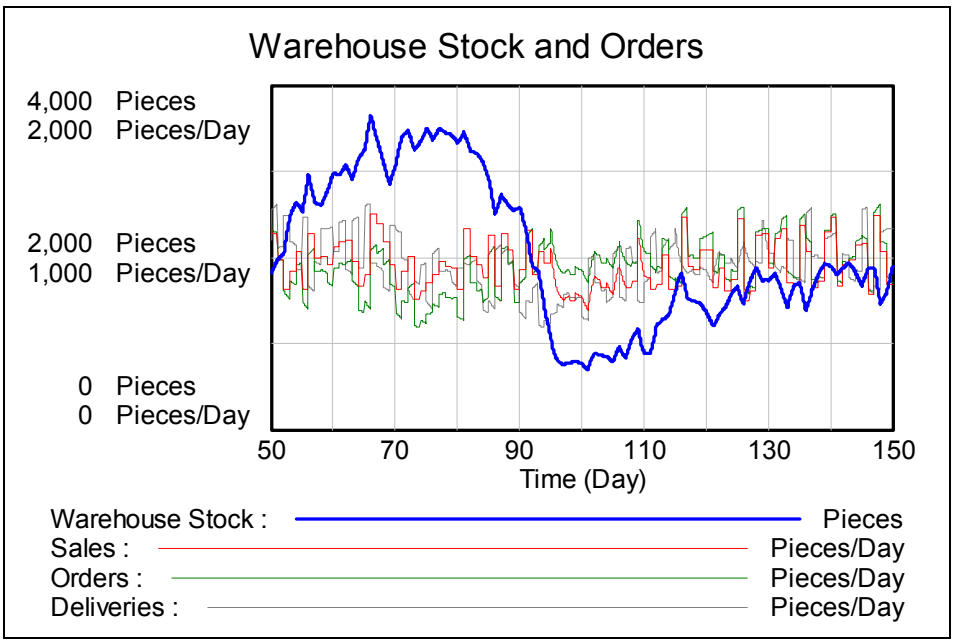
For our first simulation we set SALES PULSE PERC = 0 and observe the process which is now the result of random fluctuations only.

Please notice the command DELAY FIXED in *Deliveries*. At every delay in deliveries the value of the input must be stored for the present in order to hand it over later. (We will make use of the DELAY command in the last chapter.)

Let us observe the process over a period of 500 days:



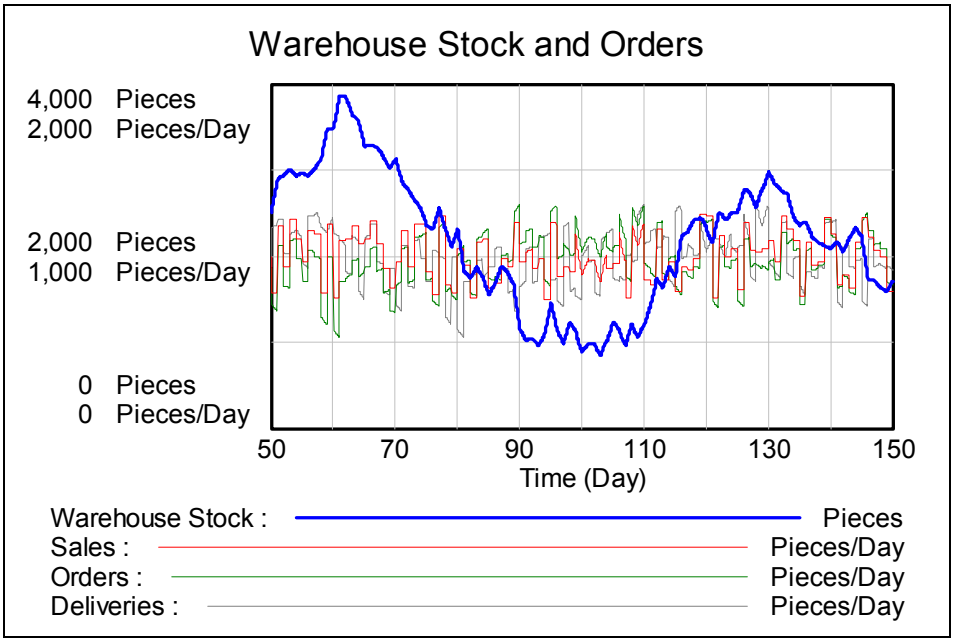
We select any space of time and spread it for a more accurate inspection, e.g. the span between days 100 and 150.



Bossel gives the interpretation in his book as follows:

> "As long as no *Delivieries* on earlier *Orders* are arriving the increasing difference from the WAREHOUSE STOCK GOAL together with ongoing *Sales* is resulting in more *Orders* which lead after the DELIVERY DELAY to *Deliveries*. Thanks them the *Warehouse Stock* is increasing and the *Orders* can be reduced. According to DELIVERY DELAY *Deliveries* will decrease after a while and the cycle starts again …"
> (This was not so easy for me to translate and I hope that I did it not too bad!)

Function RANDOM UNIFORM(0,1,0) generates a uniform distrubuted (pseudo) random number in the interval (0,1) with 0 as initial value I.e. that we will receive the same random numbers at every simulation run as long as we don't change the third parameter.
I changed to RANDOM UNIFORM(0,1,**1**) in the equation of *find new value* to run another simulation and present again the period of 100 days. Can we confirm *Bossel*'s oberservation?



After simulation with random sales fluctuations we will repeat the simulation considering one single *Sales pulse* which is then followed by constant daily sales.

Two parameters must be changed:

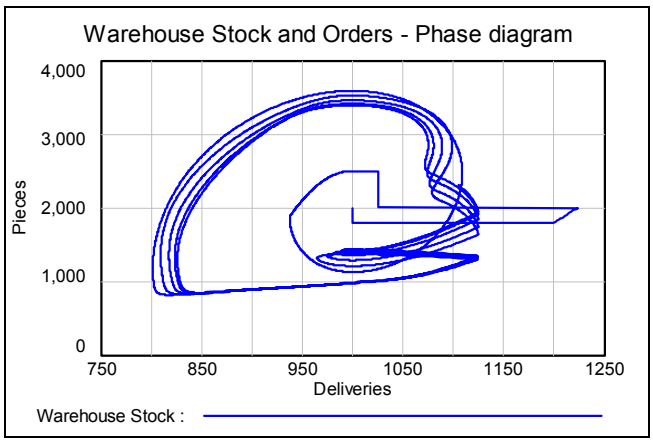
MAX SALES FLUCTUATION = 0

SALES PULSE PERC = 20

The pulse can be seen as short vertical segment on the left border.

We can observe an undamped periodic oscillation of the stock with a period length of about 80 days.

The phase diagram demonstrates this very clear.



The respective program with *DERIVE*

Ich must admit that I had some problems understanding and reproducing the dynamics manually before realizing the process in form of a *DERIVE* program. However, finally it worked. I will omit this step here. It should be possible to read off the procedure from the program.

I predefined the parameters and did not include them into the list of function arguments. I believe that this makes running the simulation more comfortable. Of course I wanted to compare my results with the *VENSIM*-data. *DERIVE* delivers other (pseudo-) random numbers. What to do?

I transferred the listof the first 500 *VENSIM* generated random numbers into a data list for *DERIVE*. This was not difficult. I named this list as `rdnrs`. Here are the first five numbers of this list:

```
rdnrs
     [1, ..., 5]

[0.4927785098, 0.56665027, 0.07161787, 0.65105546, 0.3773718198]
```

I kept the designation of the variables as short as possible – but yet understandable – in order to obtain a well readable program code.

The first lines serve for defining the parameters.

```
[ini_st := 2000, stdel_of := 0.125, dailys_of := 1, stgoal := 2000, deldel := 20]

[maxsfl := 25, avgsales := 1000, salesremrate := 1]
```

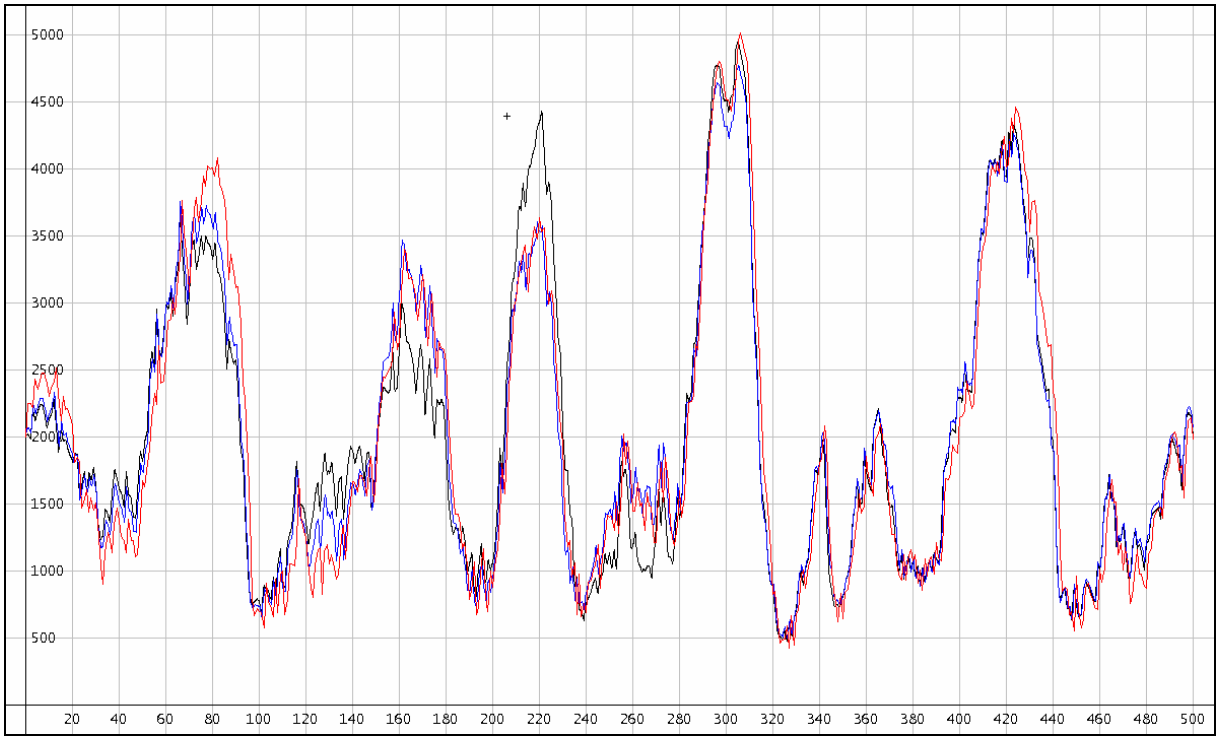
This is the program. Please note the line between quotes!

```
whouse(n, dt, i, t, rddev, sales, delivs, orders, stock, dayrd, tab) :=
     PROG(
          "rdnrs := VECTOR(RANDOM(1), i, 500)",
          n := n/dt,
          i := 1,
          [t := 0, stock := ini_st, delivs := tvm],
          rddev := avgsales·2·maxsfl/100·(−0.5),
          sales := avgsales + rddev,
          orders := dailys_of·sales,
          tab := [[t, stock, sales, orders, delivs]],
          LOOP(
               IF(i > n, RETURN tab),
               t := t + dt,
               dayrd := rdnrs↓CEILING(t),
               rddev := avgsales·2·maxsfl/100·(dayrd − 0.5),
               stock := stock + dt·(delivs − sales),
               sales := IF(avgsales + rddev < stock·salesremrate,
                    avgsales + rddev, salesremrate·stock),
               delivs := IF(i < deldel/dt, avgsales, tab↓(i − deldel/dt + 1)↓4),
               orders := IF(dailys_of·sales + stdel_of·(stgoal − stock) > 0,
                    dailys_of·sales + stdel_of·(stgoal − stock), 0),
               tab := APPEND(tab, [[t, stock, sales, orders, delivs]]),
               i :+ 1))
```

I plot the time-stock-diagram for time steps 0.0625 (*System Zoo*), 0.25 and 1

We can see that we receive the same 0.0625-diagram as given in *Bossel*'s book and further that it seems to be sufficient choosing a time step *dt* = 1.

```
(whouse(500, 0.0625))↓↓[1, 2]

(whouse(500, 0.25))↓↓[1, 2]

(whouse(500, 1))↓↓[1, 2]
```



black: dt = 0,0625; blue: dt = 0,25 and red: dt = 1

In my opinion time step 1 (day) make sense. Who will update every 30 minutes (= 1/16 of a n 8 hours' labour day) update the sales numbers, the orders etc. Usually this happens at the and of a day or even of a week!

My *DERIVE*-model does not consider the *Sales Pulse*. I realized the model with the random deviation from the average only. As mentioned above I used the (pseudo) random numbers generated by *VEN-SIM* in order to have a reference available (to check the correctness of m< program).

Let's compare the first rows of the resulting tables.

Here are the results of the first seven time steps performed with *VENSIM*:

Warehouse stock and Orders

Time (Day)	Warehouse Stock	Sales	Orders	Deliveries
0	2,000	750	750	1,000
0.0625	2,016	996.39	994.44	1,000
0.125	2,016	996.39	994.41	1,000
0.1875	2,016	996.39	994.38	1,000
0.25	2,016	996.39	994.35	1,000
0.3125	2,017	996.39	994.32	1,000
0.375	2,017	996.39	994.30	1,000

The *DERIVE* results are following:

```
whouse(0.25, 0.0625)
```

0	2000	750	750	1000
0.0625	2015.625	996.389255	994.43613	1000
0.125	2015.850671	996.389255	994.407921	1000
0.1875	2016.076343	996.389255	994.3797121	1000
0.25	2016.302014	996.389255	994.3515031	1000

Ok, the first rows maybe the same, but how will it look some days later?

I take the section around day 20 which is the time for delivery of the first order.

I start presenting the respective part of the *DERIVE*-table:

```
(whouse(500, 0.0625))
                    [315, ..., 325]
```

19.625	1838.602725	1030.294595	1050.469254	1000
19.6875	1836.709313	1030.294595	1050.70593	1000
19.75	1834.8159	1030.294595	1050.942607	1000
19.8125	1832.922488	1030.294595	1051.179283	1000
19.875	1831.029076	1030.294595	1051.41596	1000
19.9375	1829.135664	1030.294595	1051.652636	1000
20	1827.242252	1030.294595	1051.889313	750
20.0625	1809.723839	928.27672	952.06124	994.43613
20.125	1813.858803	928.27672	951.5443696	994.407921
20.1875	1817.992003	928.27672	951.0277196	994.3797121
20.25	1822.12344	928.27672	950.5112899	994.3515031

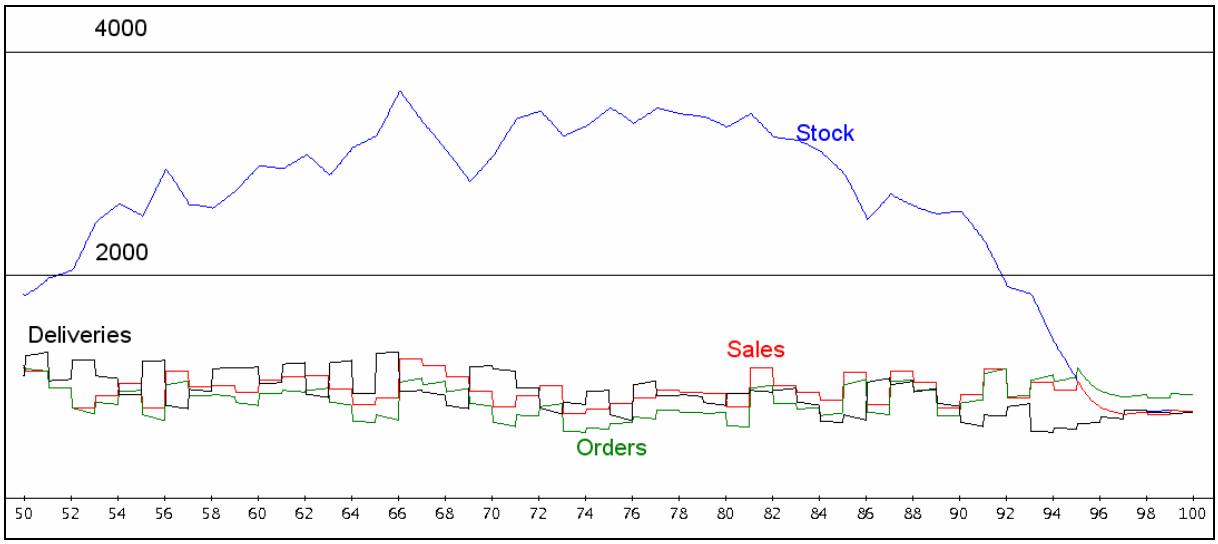
Comparison with the *VENSIM*-results shows complete correspondence.

Warehouse stock and Orders

Time (Day)	Warehouse Stock	Sales	Orders	Deliveries
19.625	1,839	1,030	1,050	1,000
19.6875	1,837	1,030	1,051	1,000
19.75	1,835	1,030	1,051	1,000
19.8125	1,833	1,030	1,051	1,000
19.875	1,831	1,030	1,051	1,000
19.9375	1,829	1,030	1,052	1,000
20	1,827	1,030	1,052	750
20.0625	1,810	928.28	952.06	994.44
20.125	1,814	928.28	951.54	994.41
20.1875	1,818	928.28	951.03	994.38

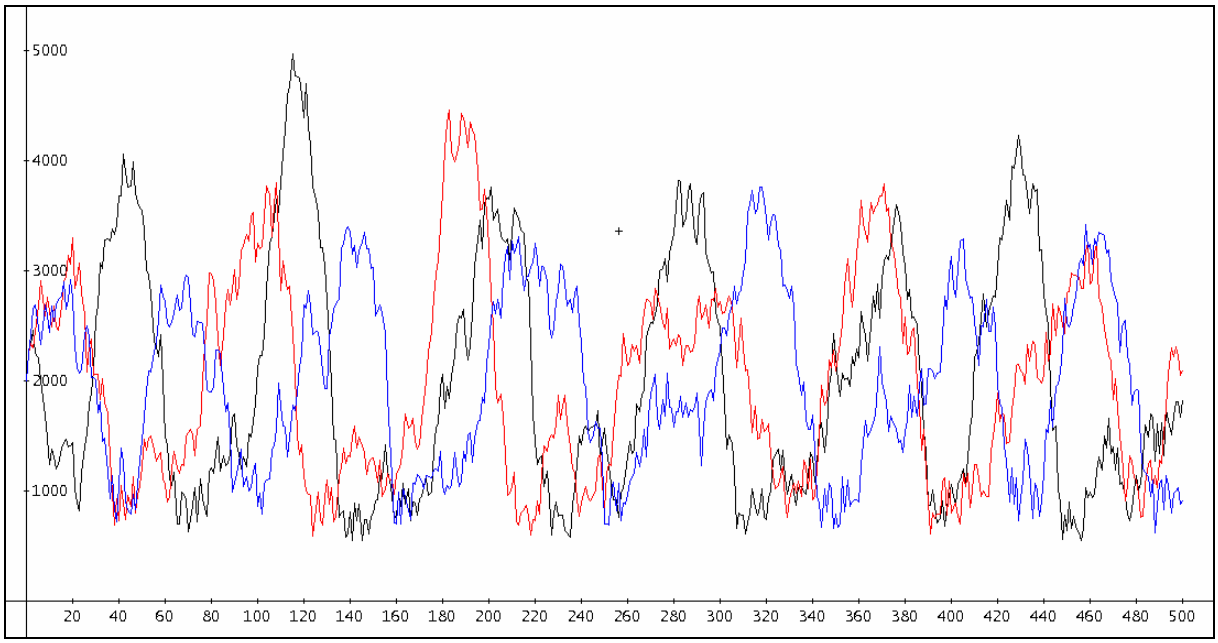
This correspondence remains for the rest of the tables.

It is easy to zoom into any section of the 500 days period.



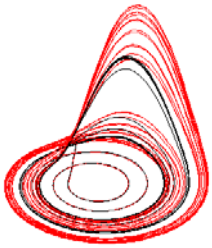
As I can be sure that my model works correctly I will activate the random number generator of *DERIVE*. What I have to do is removing the quotes in the first program line.

I run the simulation for $dt = 1$ three times and plot the stock graphs on the same axes.



Following this procedure we could also perform the simulation using *TI-Nspire*, *GeoGebra* or *MS Excel*. This would offer the possibility to introduce sliders for all parameters for better studying their influence.

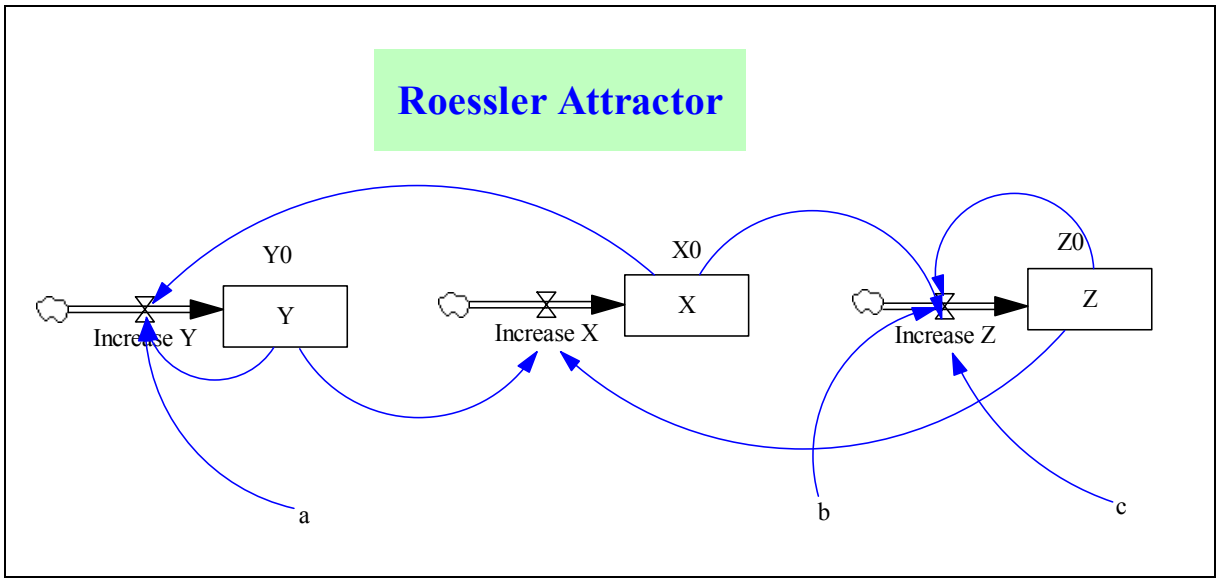
10 Rössler Attractor



If you are interested in dynamic systems, fractals and chaotic behaviour then you will very soon read about „strange attractors“, like the famous Lorenz Attractor, the Hénon Attractor, the Ikeda Attractor and others – and the Rössler Attractor.

The Rössler Attractor (Otto E. Rössler, German biochemist, born 1940 in Berlin) is described by a system of differential equations.

$$\dot{x} = -y - z$$
$$\dot{y} = x + a\,y$$
$$\dot{z} = b + z(x - c)$$



As there are only a few variables necessary, the parameters and equations for the stock variables are quickly fixed.

Parameters for the first simulation

a = 0.55

b = 2

c = 4

X0 = 1

Y0 = 0

Z0 = 0

Dynamics

Increase X = –Y – Z

Increase Y = +X + a*Y

Increase Z = b + Z*(X – c)

X = INTEG(Increase X, X0)

Y = INTEG(Increase Y, Y0)

Z = INTEG(Increase Z, Z0)

Time parameters for the first simulation

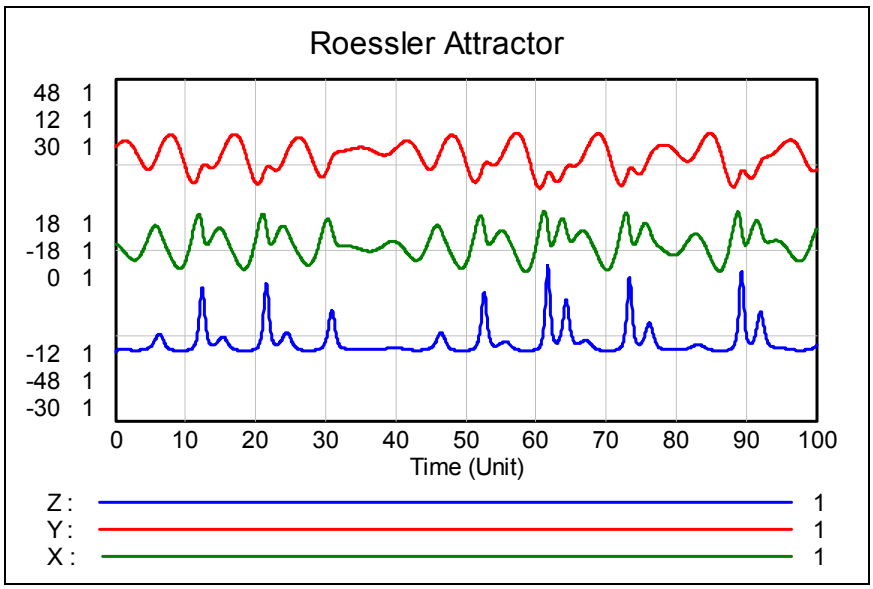
INITIAL TIME = 0

FINAL TIME = 100

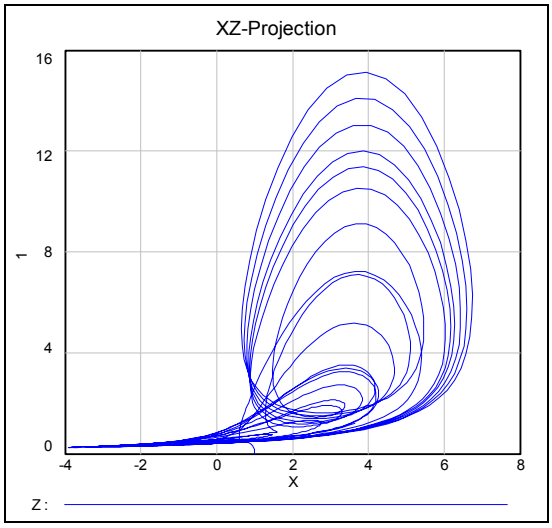
TIME STEP = 0.01

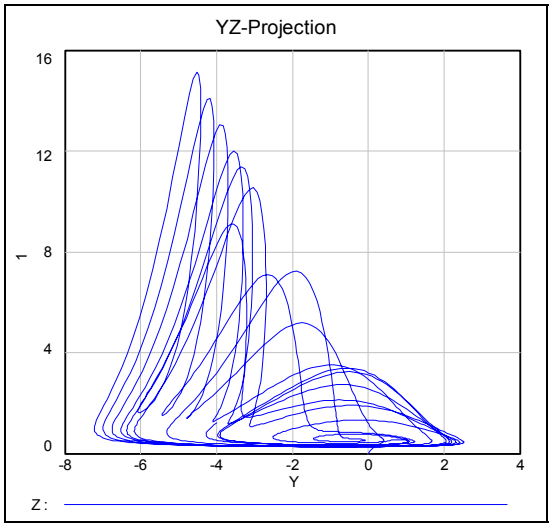
SAVEPER = 0.05

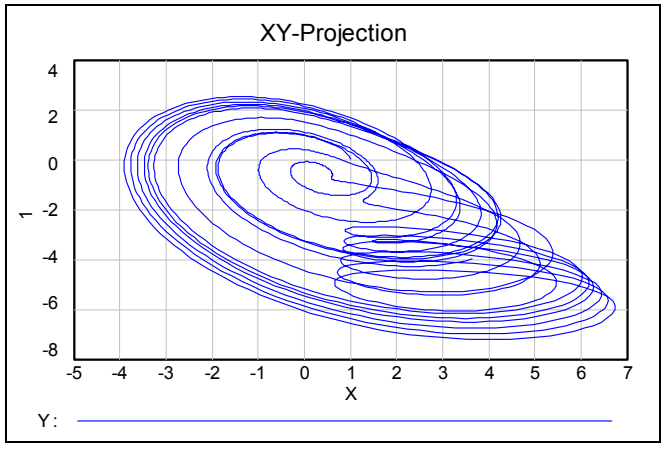
We visualize the oscillations for *X*, *Y* and *Z* for the first 100 time units. I separated the graphs using different scalings for getting a better diagram.



These diagrams can also be displayed with SyntheSim for observing the influence of the parameters. Unfortunately this is not possible for the phase diagrams. It is a pity that we cannot admire the beauty of the attractor in its full 3D-appearance with *VENSIM*. We must be content with projections of the object into the coordinate planes (front view, side view and top view).







Some parameter combinations lead to limit situations with doubling periods occuring. We will show this on the next page and then again in the frame of the *TI-Nspire*-modelling..

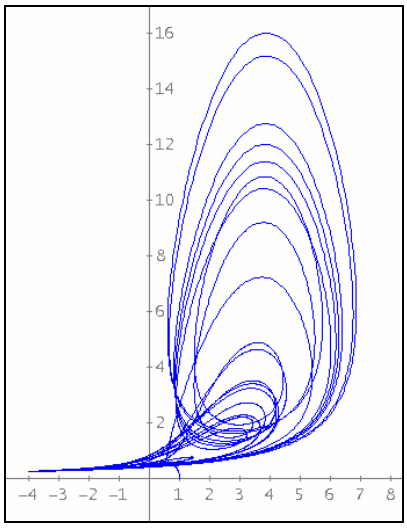
Again I must regret that despite the many features of *VENSIM* – which we cannot fully exploit at all – there is no way to produce a 3D-presentation and there are no more extended sliders.

DERIVE offers both features. So I will turn to this CAS. But here is also some reason for regret. The spatial presentation needs a huge number of points calculated by the Runge-Kutta-method and we cannot introduce sliders because of reasons which we mentioned earlier.

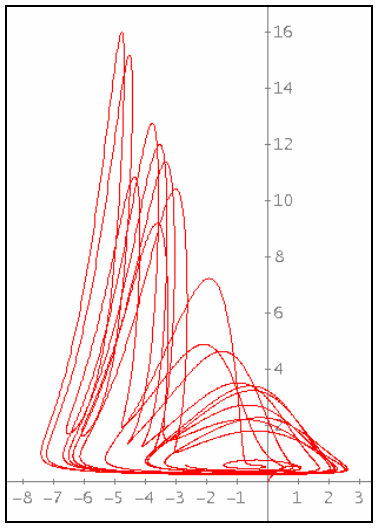
DERIVE and the Roessler Attractor

The projections into one of the coordinate planes need only one command line. Within a few seconds we can admire the result of the calculation in form of 10 000 points.

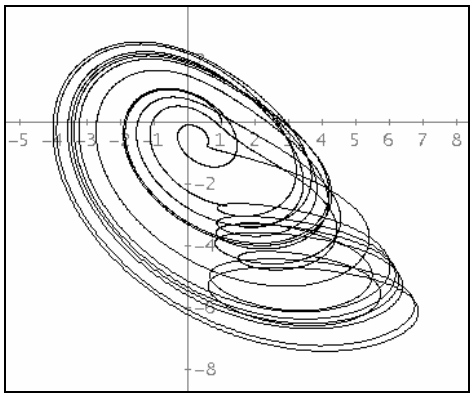
```
(RK([-y - z,x + 0.55·y,2 + z·(x - 4)],[t,x,y,z],[0,1,0,0], 0.01, 10000))↓↓[2,4]
(RK([-y - z,x + 0.55·y,2 + z·(x - 4)],[t,x,y,z],[0,1,0,0], 0.01, 10000))↓↓[3, 4]
(RK([-y - z,x + 0.55·y,2 + z·(x - 4)],[t,x,y,z],[0,1,0,0], 0.01, 10000))↓↓[2, 3]
```



XZ-Projection



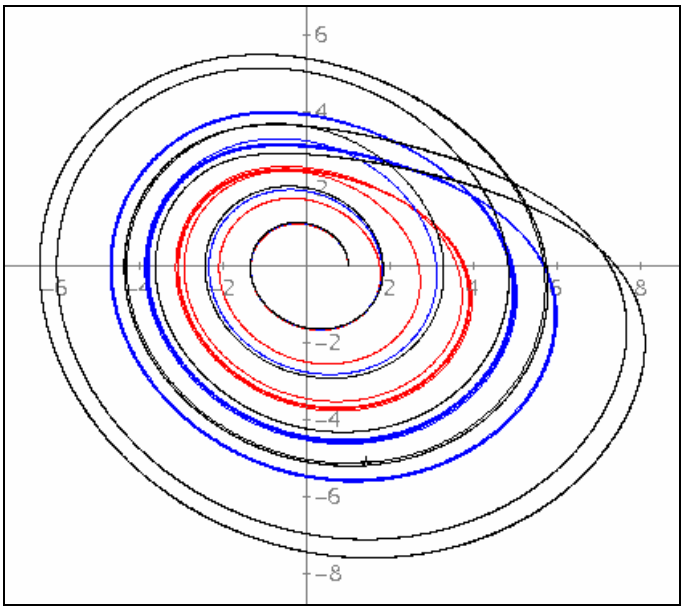
YZ-Projection



XY-Projection

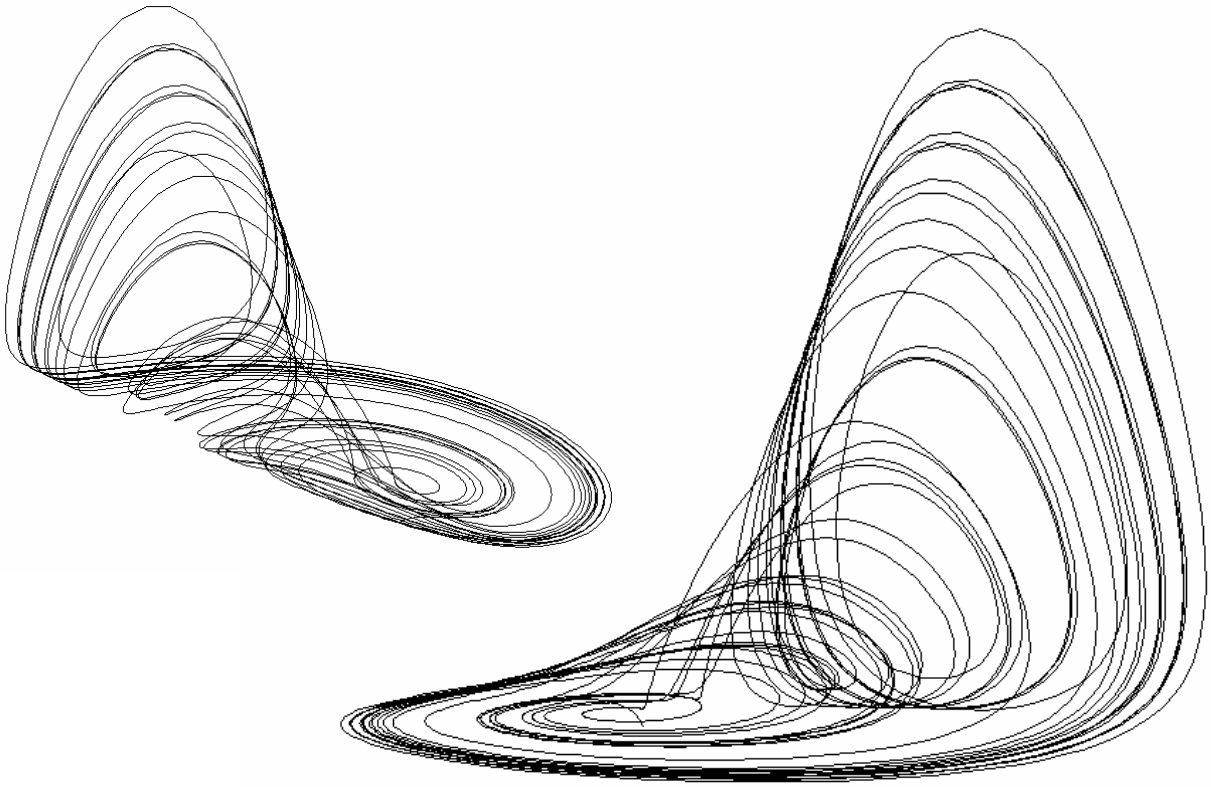
Taking the parameters as follows we have limit cycles for c = 2, 3 and 4, which let us recognize the period doubling.

```
VECTOR((RK([-y - z, x + 0.2·y, 0.2 + z·(x - c)], [t, x, y, z], [0, 1, 0, 0],
      0.01, 10000))↓↓[2, 3], c, 2, 4)
```



DERIVE enables displaying this beautiful attractor in three dimensions. 4000 points are sufficient for producing a fine graph.

```
(RK([-y - z,x + 0.55·y,2 + z·(x - 4)],[t,x,y,z],[0,1,0,0], 0.05, 4000))↓↓[2,3,4]
```

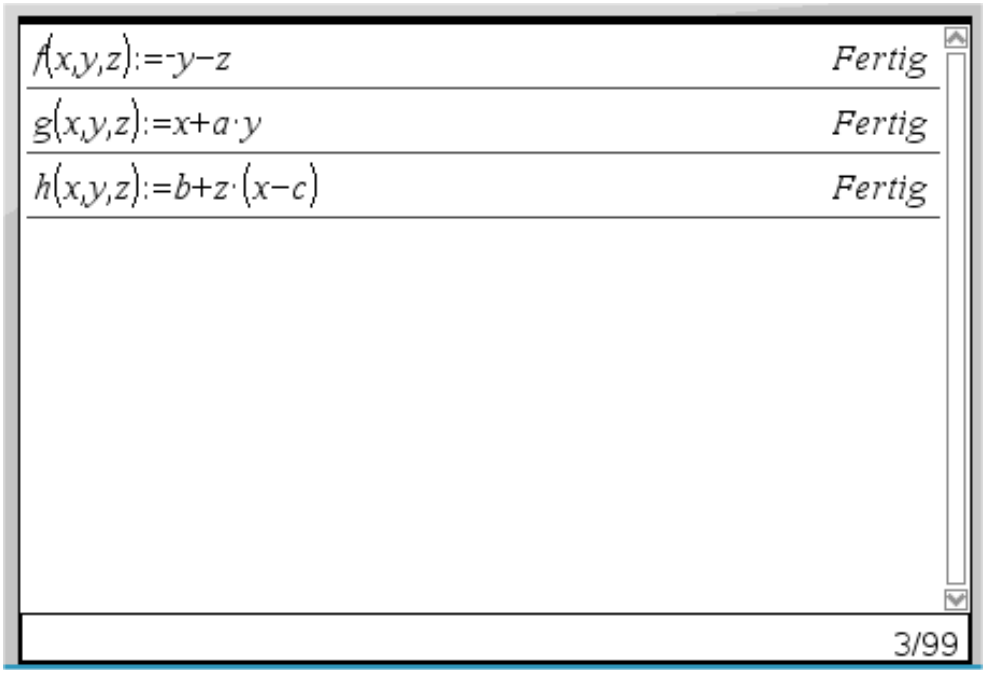


Rössler Attractor with Sliders and *TI-Nspire*

I find it boring entering the command line again and again when changing one or the other parameter and then waiting some seconds for inspecting the resulting graph. I'd like to observe the influence of the parameters continuously. That is: we need sliders. *TI-Nspire* (and *GeoGebra* and *MS-Excel* as well) offer them.

I start with *TI-NspireCAS*. As I did in earlier models I do not use a program but I rely upon the spreadsheet. I will apply the *Euler*-method in my first attempt. In order to obtain a reasonable result I must keep the step width very small which provokes a large number of steps, i.e. a (possibly too) large number of rows in the spreadsheet for *TI-Nspire*. Runge-Kutta can be done, but this procedure is very costly. I choose the middle course and will work applying the "*Improved Euler method*" for solving systems of differential equations.

I install sliders for the parameters *a*, *b* and *c*, for the initial values *x0, y0*, z0, and for the step width *dt*. The right sides of the system are defined as independent functions.



Now I am ready to insert the spreadsheet application.

The situation is given by the following positions of the sliders: *x0* = -0.1, *y0* = 2, *z0* = 0, *a* = *b* = 0.2, *c* = 2, *dt* = 0.1

	A t	B xw	C yw	D zw	E	F	G	H	I	J
◆										
1	0	-0.100000	2.000000	0.000000	-2.000000	0.300000	0.200000	-2.050000	0.106000	0.154000
2	0.100000	-0.302500	2.020300	0.017700	-2.038000	0.101560	0.159246	-2.064081	-0.100209	0.115727
3	0.200000	-0.507604	2.020368	0.031449	-2.051816	-0.103531	0.121139	-2.053577	-0.310783	0.081824
4	0.300000	-0.712874	1.999652	0.041597	-2.041249	-0.312943	0.087153	-2.018670	-0.523327	0.053240
5	0.400000	-0.915870	1.957838	0.048616	-2.006455	-0.524302	0.058241	-1.959849	-0.735433	0.030335
6	0.500000	-1.114185	1.894852	0.053045	-1.947897	-0.735214	0.034807	-1.877856	-0.944708	0.012957

The first row is filled in according to the improved Euler method as follows:

The contents of the first row cells from A1 to J1 are:

A1: 0 B1: = x0 C1: = y0 D1: = z0

E1: = f(b1,c1,d1) F1:= g(b1,c1,d1) G1: = h(b1,c1,d1)

H1: = f(b1 + dt e1, c1 + dt f1, d1 + dt g1) I1: = g(b1 + dt e1, c1 + dt f1, d1 + dt g1)

J1: = h(b1 + dt e1, c1 + dt f1, d1 + dt g1)

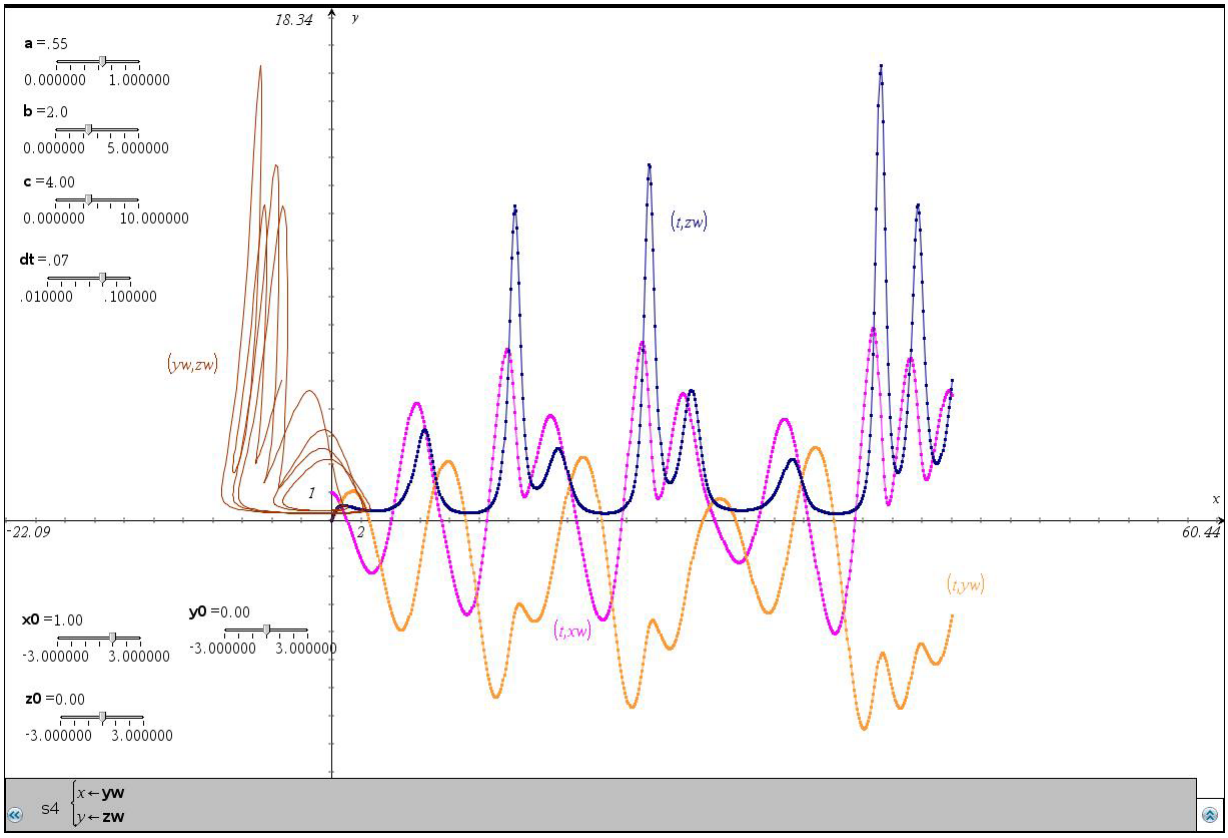
The second row is following:

A2: = a1 + dt B2: = dt/2 (e1 + h1) C2, D2 are copies of B2

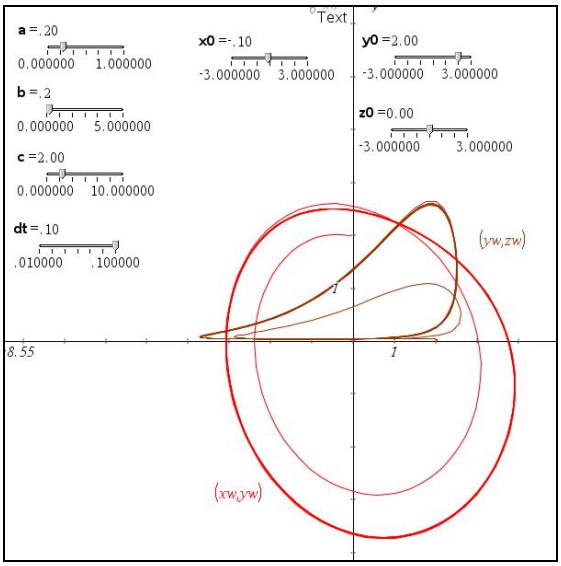
F2 : J2 are copies of F1 : J1.

This second row is to copied down in order to produce more points. It is recommended to perform this process in more steps to not overstress the system, at first until row 201, then proceed to row 401 and finally until row 601. This gives 600 points which is sufficient for fine graphs. Columns A to D are denominated and the columns (= lists) are used to plot the scatter diagrams which can be formatted as you like – and what is most important, they react immediately on the sliders.

The first screen shot shows all time-diagrams together with one projection (YZ-projection).

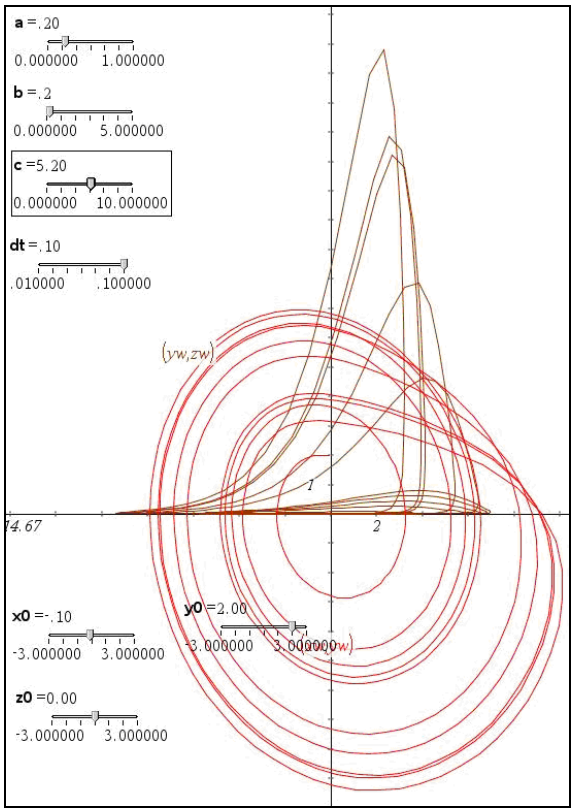


Small movements with the sliders let us receive the representation of the limit cycles.



XY-Projection – Limit Cycles

Take $c = 5.2$ – and the limit cycles tend to become chaotic (right screen shot).



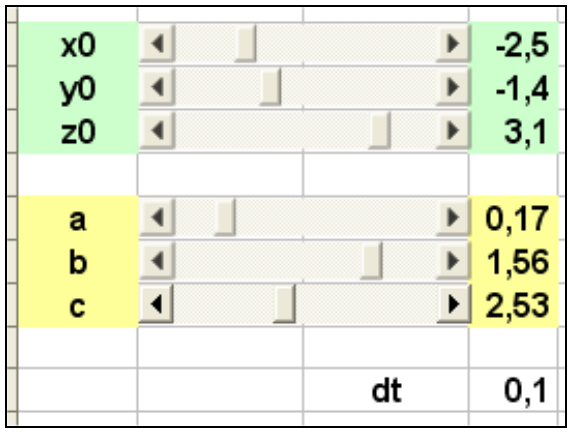
This works really excellent and can be reproduced in classroom – at least in my opinion.

I still have the hope to produce a three dimensional representation of the attractor together with sliders. We expect a 3D-*GeoGebra* in the near future (and possibly there will follow a 3D-*Nspire,* too), then we will be able, but now?? There is indeed another way, come and look!

The Rössler Attractor in three Dimensions with Sliders

The spreadsheet programs implemented in *Nspire* and *GeoGebra* are quite nice but for really many data I like to turn back to good old *MS-Excel.* (I don't forget that *Nspire* has and *GeoGebra* will have in the near future CAS features available in the spreadsheet, too!)

I introduce for all parameters – except for *dt* – sliders (which are called in the German version "*Schieberegler*" = "slide controller").



Entering the formulae is some work, but it is not too much to do.

2000 iterations and more can be performed without any problems. So we can have a step width of 0.05 to reach $t = 100$.

It is really fascinating to observe how the three projections are reacting simultaneously on every change of the parameters.

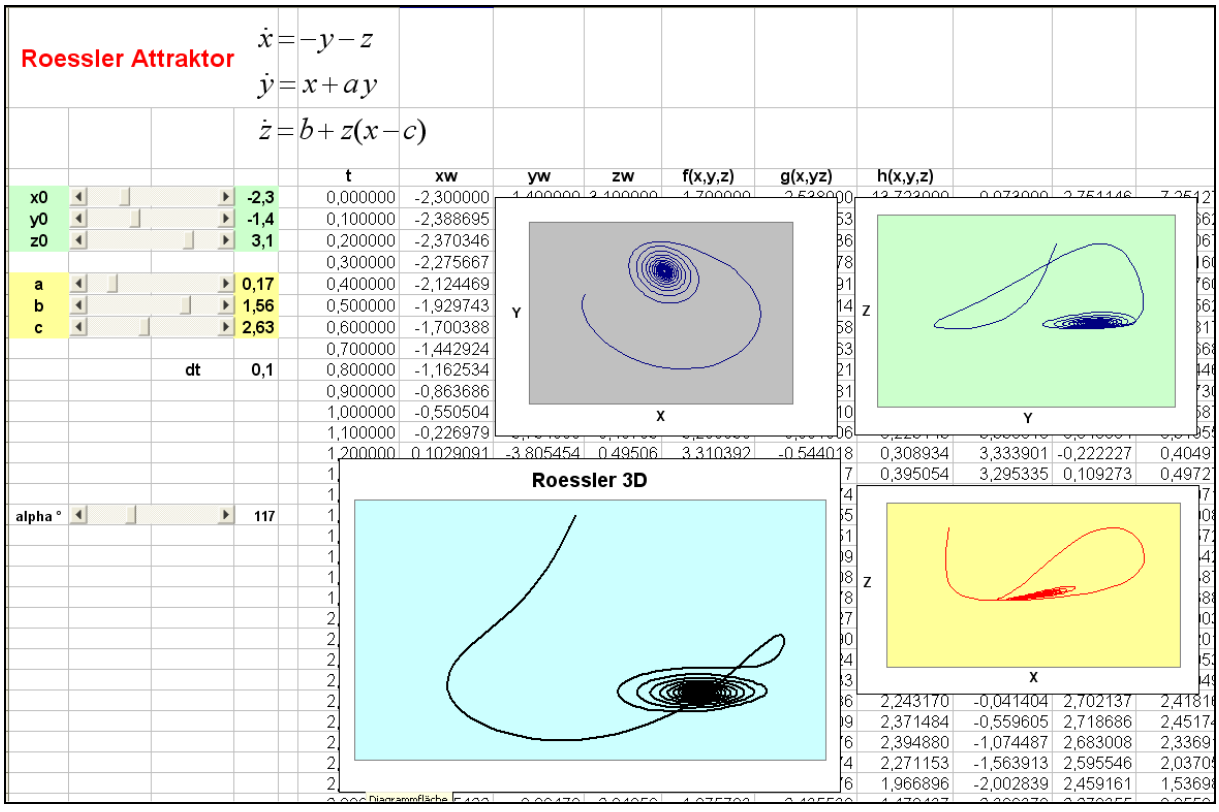
This is not new, but what about a three dimensional picture? We could produce a parallel projection or even a central projection of the 3D-object into the 2D-plane.

Top-, front- and side view could be seen earlier. I will refer to a presentation which we all know from our time as student: the oblique view. This representation form is determined by a dilatation in *x*- or *y*-direction and the angle which is formed by this dilatated axis and and the horizontal line. I fix the dilatation with 0.75 and make the angle variable using another slider, of course.

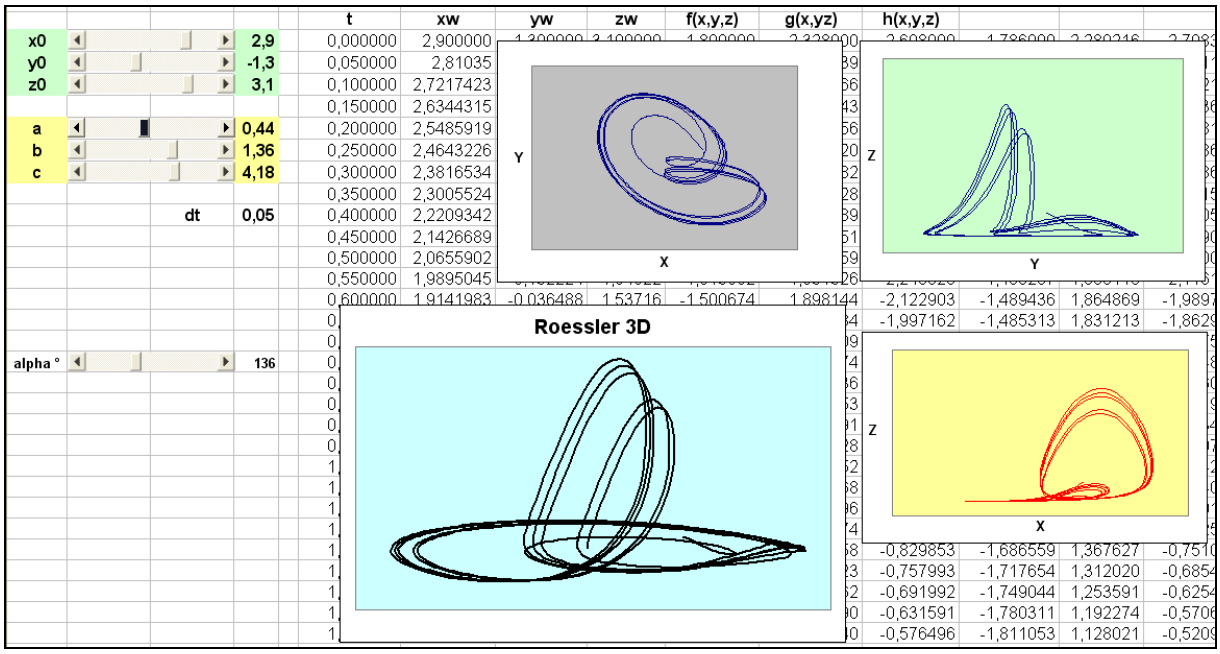
The transformation equations which give the coordinates *x*' and *y*' of the mapping resulting of the space coordinates of a point (*x*, *y*, *z*) are:

$$x' = -0,75\,x\cos\alpha + y$$
$$y' = -0,75\,x\sin\alpha + z$$

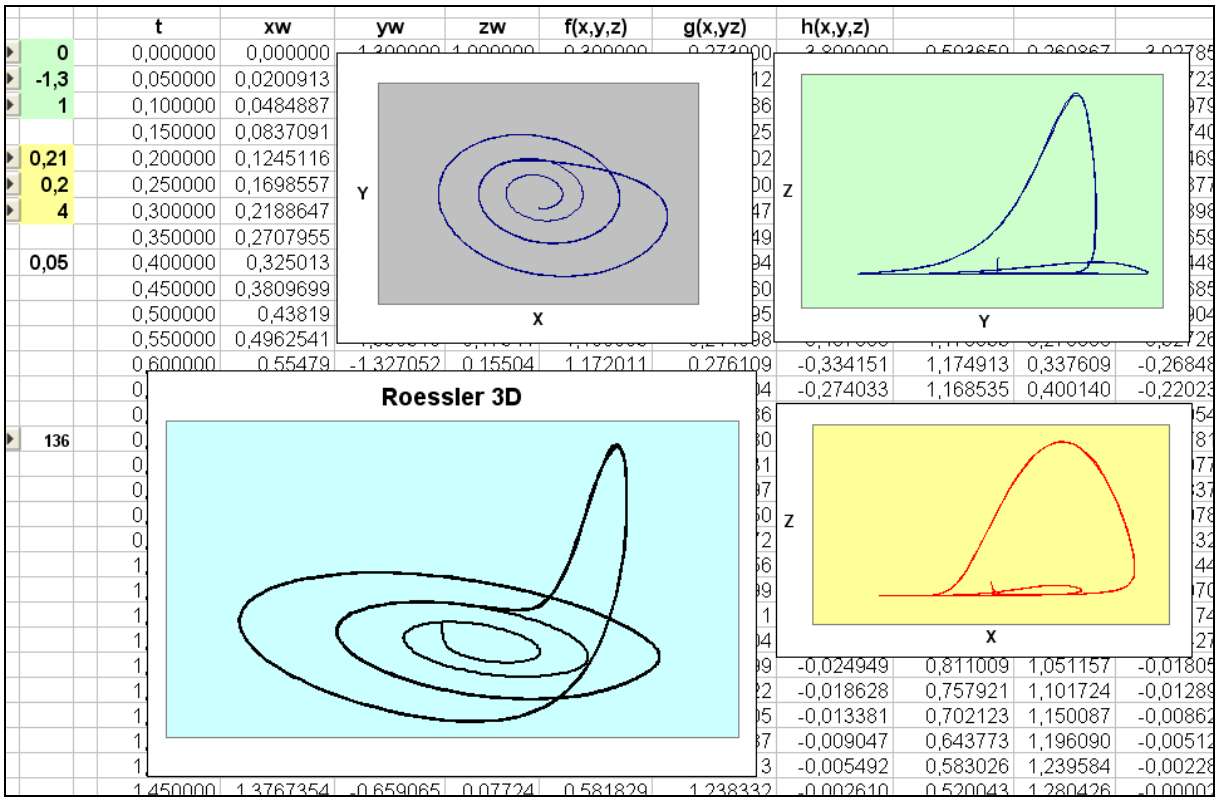
These formulae are the entries for the next two columns and copied down. The two columns are used to create the diagram which represents the oblique view of the attractor (bottom left).



The next page shows another configuration of the parameters. The diagrams are based on a step width *dt* = 0.05.



Here is another screen shot presenting limit cycles which are addressed on pages 98 and 99.



One can proceed in the same way with *TI-Nspire* and it works. The reactions are not as smooth as with *Excel*.

11 Gumowski-Mira – and another "attractive" Attractor

My intention was to stop with chapter 10. But then I became ambitious enough to add something which cannot be found in the *System Zoo*.

The world of the „strange attractors" is a wonderworld of forms and ideas behind the forms. In one of my *Chaos-books* I found a note on the *Gumowski-Mira-Attractor*. This attractor was discovered by two physicists, I. Gumowski and C. Mira in the frame of their work at CERN in Geneva in 1980.

The original model is described by

$$x_{n+1} = b \cdot y_n + f(x_n)$$
$$y_{n+1} = -x_n + f(x_{n+1})$$

with $f(x) = a \cdot x + \frac{2\,(1-a)\,x^2}{1+x^2}$; a, b are constants.

Exploring Gumowski-Mira with a Computer Algebra System

I was interested in this fascinating and manifold class of attractors prior to my knowledge of *VENSIM*. My tools were *DERIVE* and *WIRIS*.

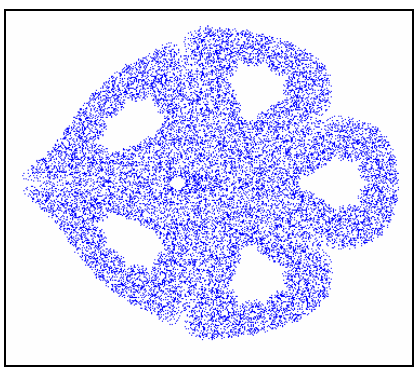
The *DERIVE*-Code is easy to follow. Some 1000 points can be generated and plotted in a short time.

The task is to find "attractive" values for the parameters *a* and *b*.

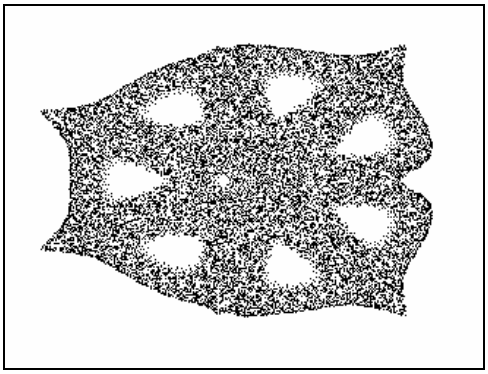
Searching in the Internet I came across great websites containing rich selections of exciting graphs. [11, 12, 13,14]

Of course, real fun makes own experimenting, researching and discovering.

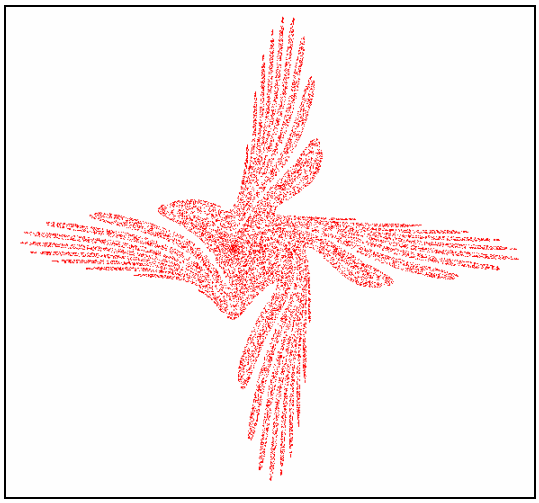
```
#1:   f(u, a) := a·u + 2·(1 - a)·u^2 / (1 + u^2)

      gum0(x0, y0, a, b, n, xn, yn, i, pts) :=
        Prog
          pts := [[x0, y0]]
          i := 1
          Loop
            If i > n
#2:           RETURN pts
            xn := b·y0 + f(x0, a)
            yn := -x0 + f(xn, a)
            pts := APPEND(pts, [[xn, yn]])
            x0 := xn
            y0 := yn
            i :+ 1
```



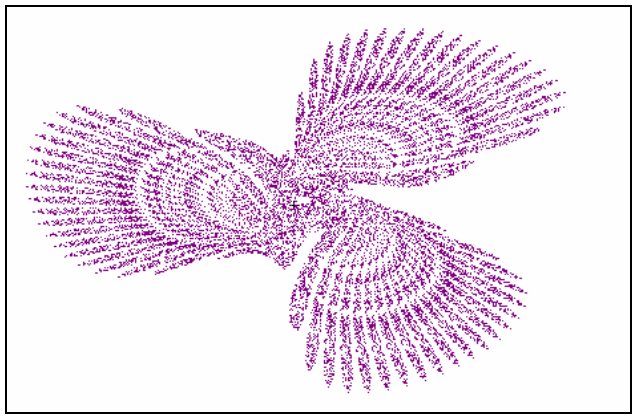
gum0(1, 1, 0.245, 1, 20000)



gum0(1, 1, -0.245, 1, 20000)



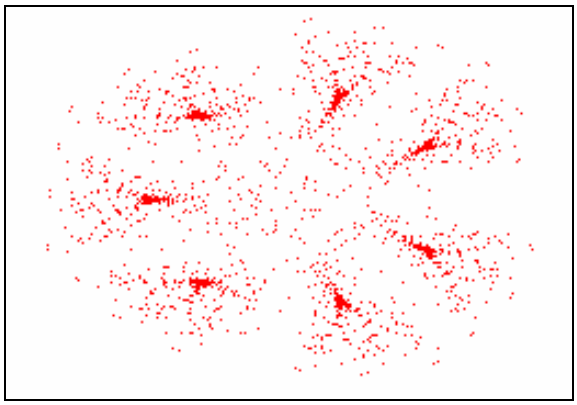
gum0(1, 1, 0.01, 0.978, 20000)



gum0(1, 1, −0.48, 0.9924, 20000)

This is an “own“ creation::

Comment: I am missing my beloved sliders!

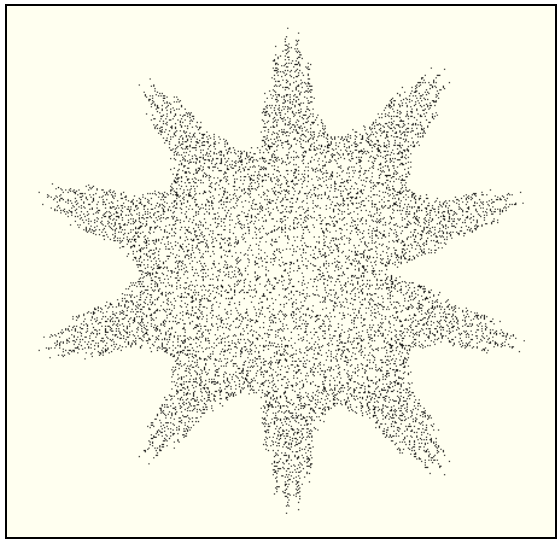


gum0(1, 1, −0.27, 0.995, 30000)

Please compare the *WIRIS*-program with the *DERIVE*-program.

$$g(x,a) := a \cdot x + \frac{2 \cdot (1-a) \cdot x^2}{1+x^2};$$

```
gum0(x0,y0,a,b,n) := begin
                       local list,xn,yn
                       list :={point(x0,y0)}
                       for i in [1..n] do
                          xn=b·y0+g(x0,a)
                          yn=−x0+g(xn,a)
                          list = list+{point(xn,yn)}
                          x0=xn
                          y0=yn
                       end
                       plot(list,{point_size=1})
                     end ;
gum0(1,1,−0.305,1,10000) → plotter1
```

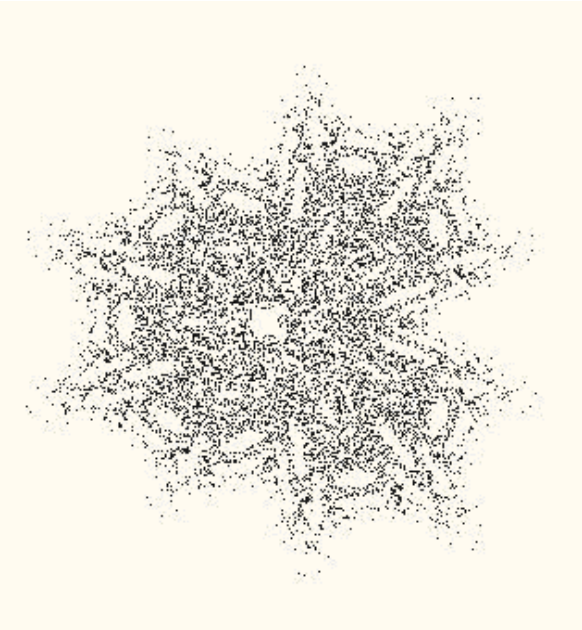


There are only slight changes in the syntax. The plot of 10 000 (!) points is convincing.

Little change – Big effect: Change of sign creates a very different plot.

$$g(x,a) := a \cdot x + \frac{2 \cdot (1+a) \cdot x^2}{1+x^2};$$

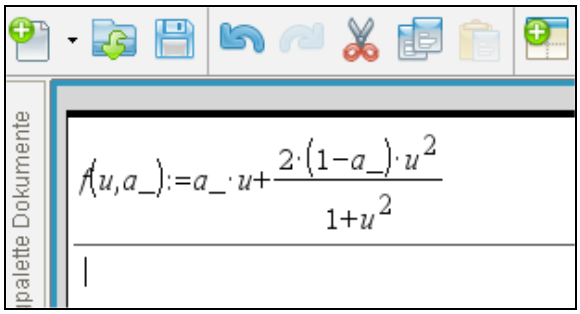
gum0(1,1,−0.305,1,10000) → plotter1



Hunting for "beautiful" attractors supported by sliders

When working with *TI-NspireCAS* we cannot calculate (and plot) thousands of points but in most cases we will get an impression plotting the first hundreds of points whether there could an „attractive" attractor be hidden or not.

This is reason enough for me to use this tool as a vehicle for my search. The procedure is very similar to that one which I used investigating the Roessler attractor.



Only two columns are needed. In this case the time-diagrams are without any value for us.

	A x	B y	C	D	E
◆					
1	1.	1.			
2	2.	0.316			
3	1.632	-0.672273			
4	0.655454	-1.06962			
5	-0.507232	0.404533			
6	1.46452	1.79992			
7	3.09261	-0.564006			

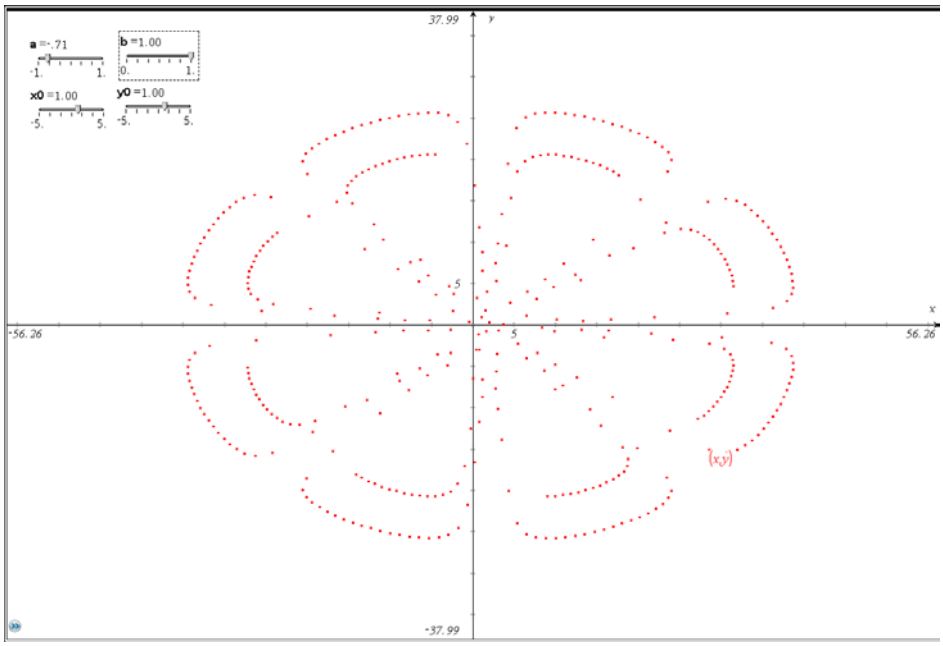
A2 =b·b1+f(a1,a)

	A x	B y	C	D	E
◆					
1	1.	1.			
2	2.	0.316			
3	1.632	-0.672273			
4	0.655454	-1.06962			
5	-0.507232	0.404533			
6	1.46452	1.79992			
7	3.09261	-0.564006			

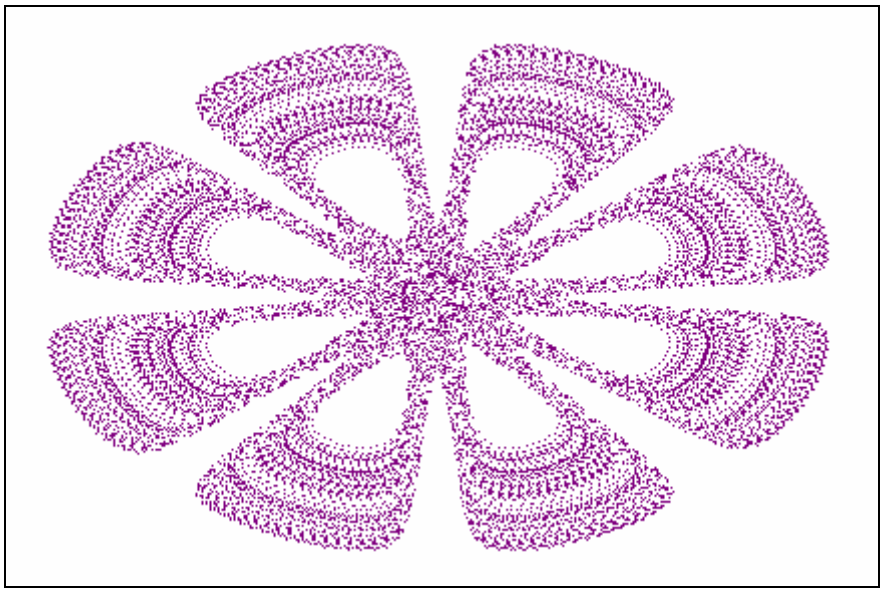
B2 =-a1+f(a2,a)

The sliders installed for *a*, *b*, *x0* and *y0* allow free experimenting. Additionally we can vary the definition of function *f* at any time.

It does not need much time to discover a promising scatter plot:



I switch to *DERIVE* or *WIRIS* and would like to observe the result of calculating and plotting not less than 20 000 points.

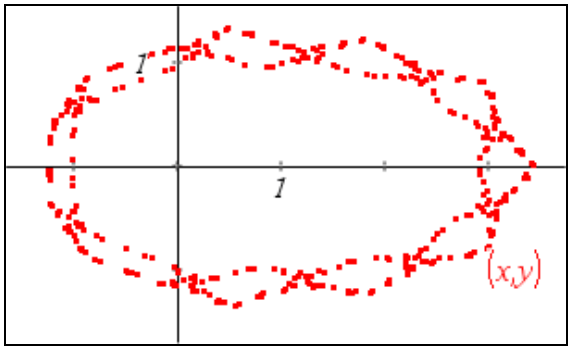


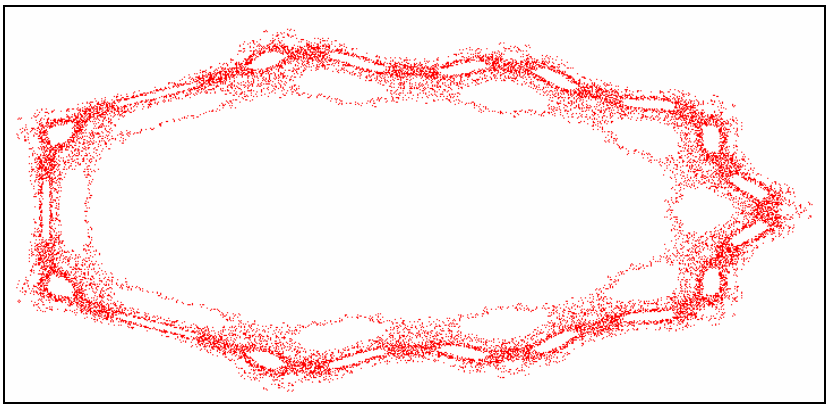
`gum0(1, 1, –0.71, 1, 20000)`

Taking $a = 0.71$ instead of $a = –0.71$ and plotting the first hundreds of points I receive just another – but even interesting – picture.

This makes me curious again and I insist seeing the attractor in its whole beauty.

The next page will unveil the secret.





gum0(1, 1, 0.71, 1, 20000)

Can we do this with *VENSIM,* too?

Unfortunately I mus admit that I – as a *VENSIM*-novice – was unable to set up an appropriate model. I had problems addressing x_n and x_{n+1} as well in the definition for y_{n+1}. I sent a request to *VENSIM* via `http://www.vensim.com`. I was very much surprised recieving an answer after some minutes. The answer was not very helpful for the moment. It read as follows:

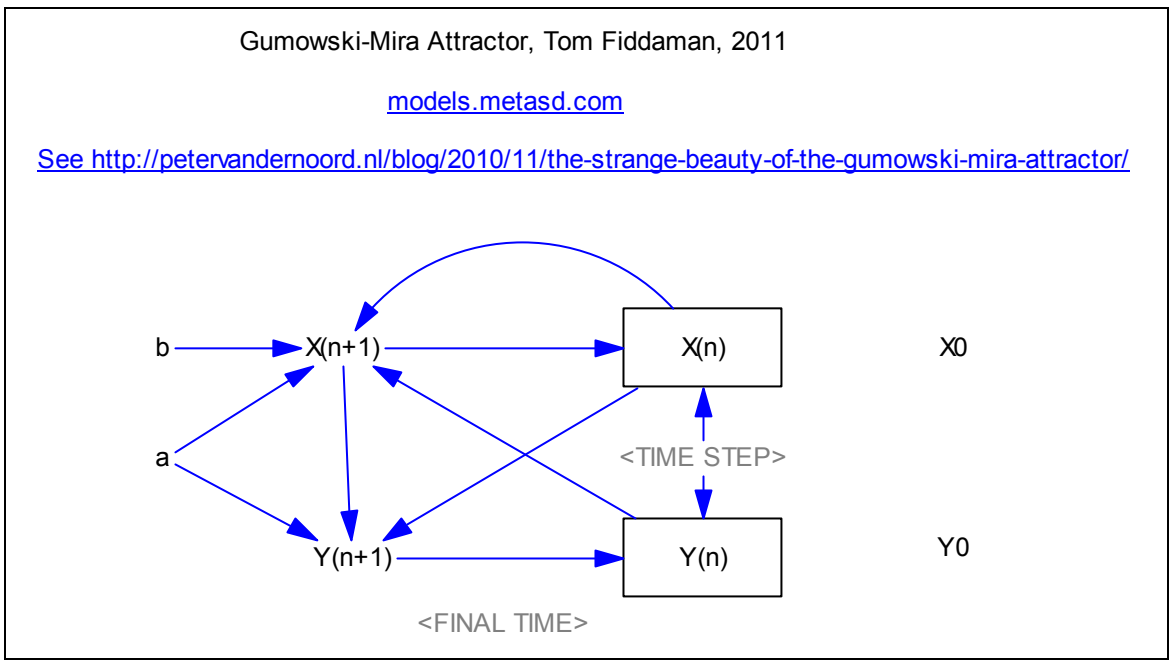
I don't see why you cannot do it. To get the old values of x and y use DELAY FIXED with a delay time of one time step.

I had to admit my inability and I received an invitation:

Can I ask you to post the question on our forum so that others can benefit from the answers as well?

I followed this invitation and – the next surprise – there arrived an answer after a short while together with the respective *VENSIM*-file. As I noticed later my request started an interesting discussion in the *VENSIM*-Forum[15].

See first the *VENSIM*-model of the Gumowski-Mira attractor:



The *VENSIM* - document offers full details:

(02) a = 0.6

(03) b = 0.995

(04) FINAL TIME = 2000

(05) INITIAL TIME = 0

(06) SAVEPER = TIME STEP

(07) TIME STEP = 1

(08) "X(n)" = DELAY FIXED ("X(n+1)",TIME STEP,X0)

(09) "X(n+1)" = b*"Y(n)"+a*"X(n)"+2*(1-a)*"X(n)"^2/(1+"X(n)"^2)

(10) X0 = 2.25

(11) "Y(n)"= DELAY FIXED ("Y(n+1)",TIME STEP,Y0)

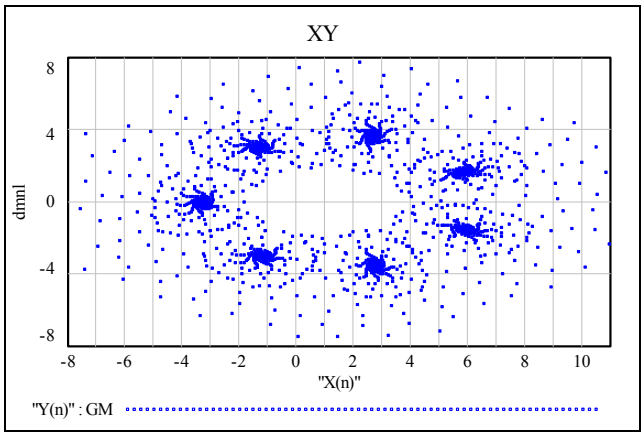
(12) "Y(n+1)"= -"X(n)"+a*"X(n+1)"+2*(1-a)*"X(n+1)"^2/(1+"X(n+1)"^2)

(13) Y0 = 7.75

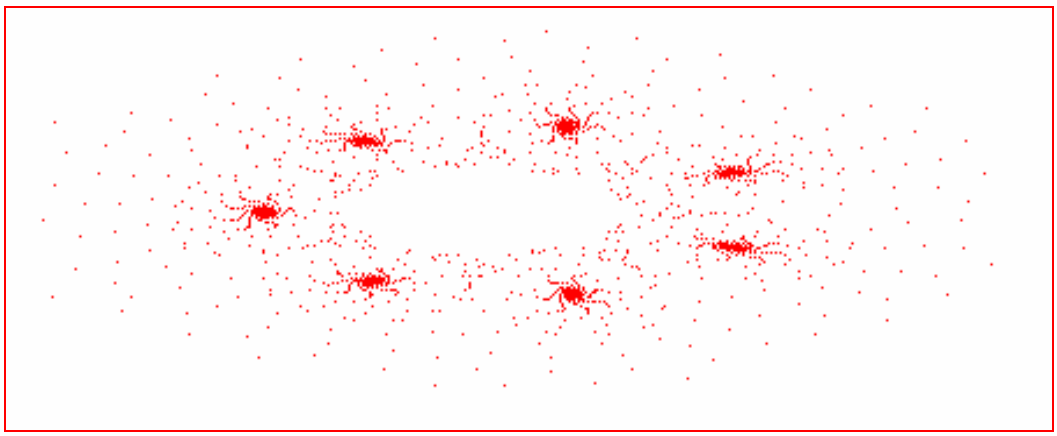
Here we find the DELAY FIXED-command which makes possible the one step delay. When using the parameters from above we are presented a “Seven Cluster of Stars”.

Tom Fiddaman added a short comment:

The behavior is really amazing.



See below the “Clusters” consisting of 20 000 “stars” generated with *DERIVE*.



This was not the end of the discussion. Another member of the *VESNIM*-Forum sent a contribution:

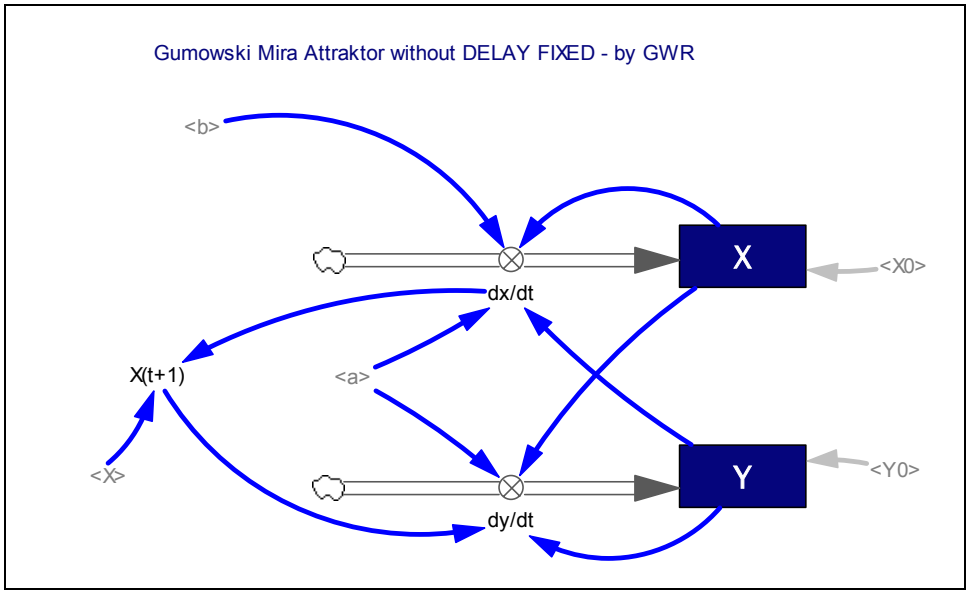
Hi Tom,

I do not see any reason for using a DELAY FIXED here. You can handle discrete systems - e.g. difference equations' systems - using the regular System Dynamics notation. What you need to do is:

1. *Set the time step to 1.*
2. *Convert the Stock Equations into difference equations, e.g. Delta X = X(t+1) – X(t) = dx/dt with dt = 1.*
3. *In the case of the Gumowski-Mira-System a bit of algebra will convince you that no delays are needed.*

See the model enclosed.

Cheers, Guido



(01) a = 0.6

(02) b = 0.995

(03) "dx/dt" = (b*Y+a*X+2*(1-a)*X^2/(1+X^2))-X

(04) "dy/dt" = (-X+a*"X(t+1)"+2*(1-a)*"X(t+1)"^2/(1+"X(t+1)"^2))-Y

(05) FINAL TIME = 2000

(06) INITIAL TIME = 0

(07) SAVEPER = TIME STEP

(08) TIME STEP = 1

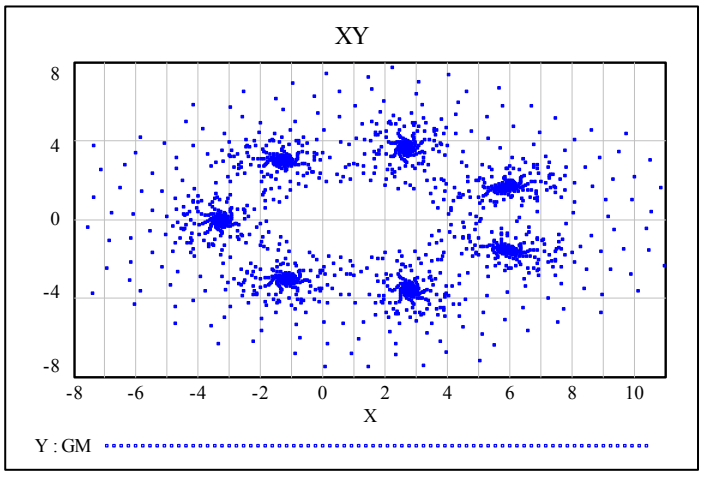
(09) X = INTEG ("dx/dt",X0)

(12) "X(t+1)" = "dx/dt"+X

(13) X0 = 2.25

(14) Y = INTEG ("dy/dt",Y0)

(17) Y0 = 7.75



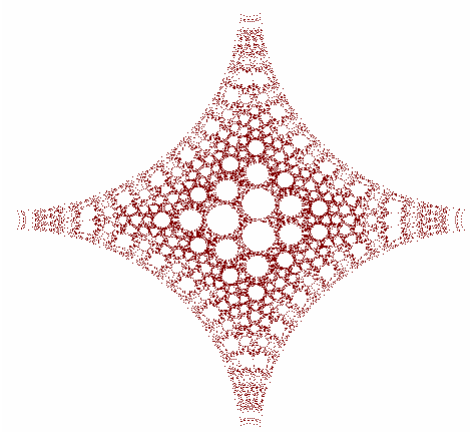
It is interesting that both graphs do not correspond exactly. Comparing the values for X we observe a correspondence until X_{34}. Then the values become different from another. It seems to be that two different algorithms are used in the background. What is Tom's opinion?

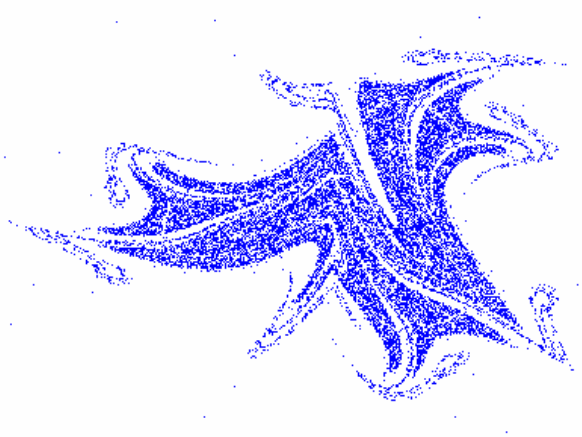
Guido,

You're entirely correct. The INTEG notation that you used is probably the nicest way to do this for formal correspondence with discrete derivative notation. However, as long as TIME STEP=1 and Euler or Diff integration is used, the results will be identical whether INTEG, SMOOTH, or DELAY FIXED is used. (Diff integration is the same as Euler, but the rates and levels are stored differently, which makes it easier to see the initial values of the rates - sometimes useful for discrete systems like this.)

Tom

For function $f(x)$ many forms can be chosen – including trig- and exp-functions. Marvellous plots are our reward.[16]





$$\text{gum\_gen}\left(-2.8,\ 17,\ 0.04,\ 1,\ 20000,\ a\cdot x + \frac{3 - a}{a + e^{b\cdot x}}\right)$$

$$\text{gum\_gen}\left(12,\ 9,\ -0.96,\ 0.96,\ 20000,\ a\cdot x - \left|\text{ATAN}(a - x) + \frac{b\cdot x^2}{1 + x^2}\right|\right)$$

This is the end of my excursion in the world of the dynamic systems. There is a huge number of interesting Internet resources. Some links are given below.

---

[11] http://www.maplesoft.com/applications/view.aspx?SID=87666

[12] http://elif-erdine.com/?p=283

[13] http://petervandernoord.nl/blog/2010/11/the-strange-beauty-of-the-gumowski-mira-attractor/

[14] http://www.generativeart.com/on/cic/papersGA2007/19.pdf

[15] http://www.ventanasystems.co.uk/forum/index.php

[16] http://math.cmaisonneuve.qc.ca/alevesque/chaos_fract/Attracteurs/Mira_exemples.pdf

[17] http://models.metasd.com

[18] http://demonstrations.wolfram.com/StrangeAttractorOfGumowskiMira/