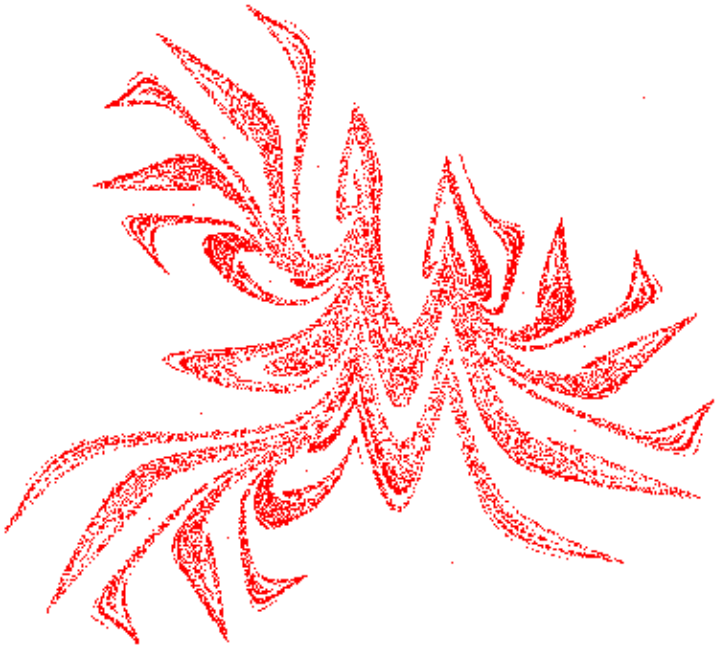
Dynamische Systeme auf verschiedenen Plattformen



Ein Streifzug
von Umwelt und Tourismus bis zu seltsamen Attraktoren

Josef Böhm

ACDCA Austrian Center for Didactics of Computer Algebra

und

DUG

Dynamische Systeme auf verschiedenen Plattformen

Inhalt

Einleitung

Mein Interesse an Fraktalen, „Chaos“ und damit auch an dynamischen Systemen besteht schon sehr lange. Die Behandlung dieser Themen ist ja eigentlich erst mit der Verfügbarkeit der Computer und entsprechender Software für Jedermann möglich geworden. Spezielle Programme wie z.B. Fractint gibt es schon lange, aber mit Hilfe von Tabellenkalkulation, Computeralgebra Systemen, Programmen zur Dynamischen Geometrie und eigener Programmierung macht es viel mehr, Spaß diese Phänomene zu untersuchen.

Durch eine Buchbesprechung bin ich auf den „*Systemzoo*“ von *Hartmut Bossel* gestoßen. Diese Bücher sind eine wahre Fundgrube für angewandte Mathematik. Dort findet man auch den wertvollen Hinweis auf die für Unterrichtszwecke freie Simulationssoftware *VENSIM*.

Ich habe dann auch den „*Systemzoo*“ auf CD erstanden und war sehr begeistert von den vielen Möglichkeiten des Einsatzes von *VENSIM*. Nun erwachte der Ehrgeiz, ein nicht zu umfangreiches Problem (Tourismus und Umwelt) auch mit anderen, an den Schulen verfügbaren Werkzeugen zu bearbeiten um jeweils deren Besonderheiten, Vor- und Nachteile bei der Behandlung von dynamischen Systemen kennen zu lernen.

Das waren dann *MS-Excel*, *DERIVE*, *WIRIS*, *TI-NspireCAS* und *GeoGebra*. Mit *Excel*, *Nspire* und *GeoGebra* konnte ich Schieberegler einsetzen, die die Simulationen durch Parametervariation noch wesentlich dynamischer machten.

Dieses eine Beispiel hat mich so fasziniert, dass ich nicht widerstehen konnte, weitere folgen zu lassen. Schließlich sind mehr als 100 Seiten daraus entstanden.

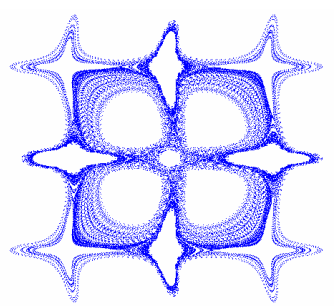
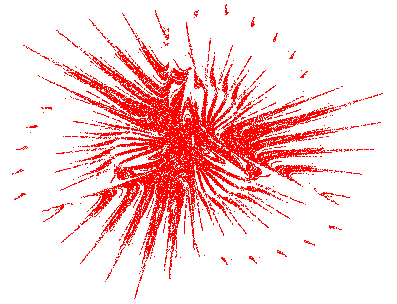
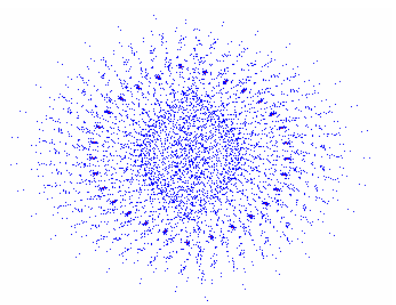
So bin ich zu Ergebnissen gekommen, die auch ästhetisch recht ansprechend geworden sind und die Lust am Entdecken und Experimentieren wohl wecken können. Mit den „schönen“ und „seltsamen“ Attraktoren mögen auch Differentialgleichungssysteme für an Mathematik nicht so interessierte Schülerinnen und Schüler an Reiz gewinnen.

Der – unbedingt notwendigen – Interpretation der entstandenen Diagramme und Tabellen konnte hier nicht der gebührende Raum gewidmet werden. Ich verweise da ausdrücklich auf die *Bossel*-Bücher und viele andere Ressourcen.

Alle in diesem Papier angesprochenen Programme können gerne von mir auf Anfrage bezogen werden. Eine kurze email genügt.

Ich wünsche viel Spaß und würde mich über Reaktionen sehr freuen.

Josef Böhm
nojo.boehm@pgv.at

1 Tourismus und Umwelt

Eine einfache Simulation – auf verschiedenen Plattformen

In *Hartmut Bossels Systemzoo* 2[1] findet sich unter vielen komplexen Systemen im Kapitel 4 Ökosysteme und Ressourcen als Beispiel Z411 *Tourismus und Umwelt*.

Alle Beispiele aus dem *Systemzoo* werden mit Hilfe der ausgezeichneten Simulationssoftware *VENSIM PLE*[2], die für schulische Zwecke frei verwendet werden darf, behandelt.

Nach einer Beschreibung des Modells werde ich hier die Durchführung dieser Simulation zuerst mit *VENSIM PLE* vorstellen. Anschließend wird ein „Nachbau" mit *MS-Excel* versucht, wobei die *VENSIM*-Ergebnisse als Referenz dienen sollen.

Das Modell kann auch mit einem Differentialgleichungssystem beschrieben werden. Die numerische Lösung mittels Runge-Kutta wird mit *DERIVE* vorgenommen. Anschließend werden die Ergebnisse mit den Ergebnissen der diskreten Modellierung verglichen.

Während mit *DERIVE* keine Schieberegler verwendet werden können, die den Einfluss der Parameter auf das Modellverhalten studieren lassen, ist dies sowohl mit *GeoGebra*[3] als auch mit *TI-Nspire*[4] möglich.

Abschließend werden die Gleichgewichtszustände wichtiger Zustandsgrößen analytisch ermittelt.

Beschreibung

„Schidorf" in der Sierra Nevada, Spanien

Zwischen Tourismus (z.B. gemessen an der Anzahl an *Touristen*/Nächtigungen) und der *Umweltqualität* besteht sicherlich eine dynamische Koppelung.

(1) Der *Zuwachs* der Umweltqualität ist bestimmt durch die UMWELTERHOLUNGSRATE und wird begrenzt durch eine logistische Wachstumsfunktion, die durch die Umweltkapazität = TRAGFÄHIGKEIT DER UMWELT limitiert ist.

(2) Bei Eintritt einer *Umweltbeanspruchung* durch die *Touristen* ergibt sich ein *Verlust* der Umweltqualität proportional zu dieser Beanspruchung nach einer UMWELTZERSTÖRUNGSRATE.

(3) Die *Umweltbeanspruchung* selbst hängt von den *Touristen* und von der *Umweltqualität* ab und ist zu beiden Größen proportional.

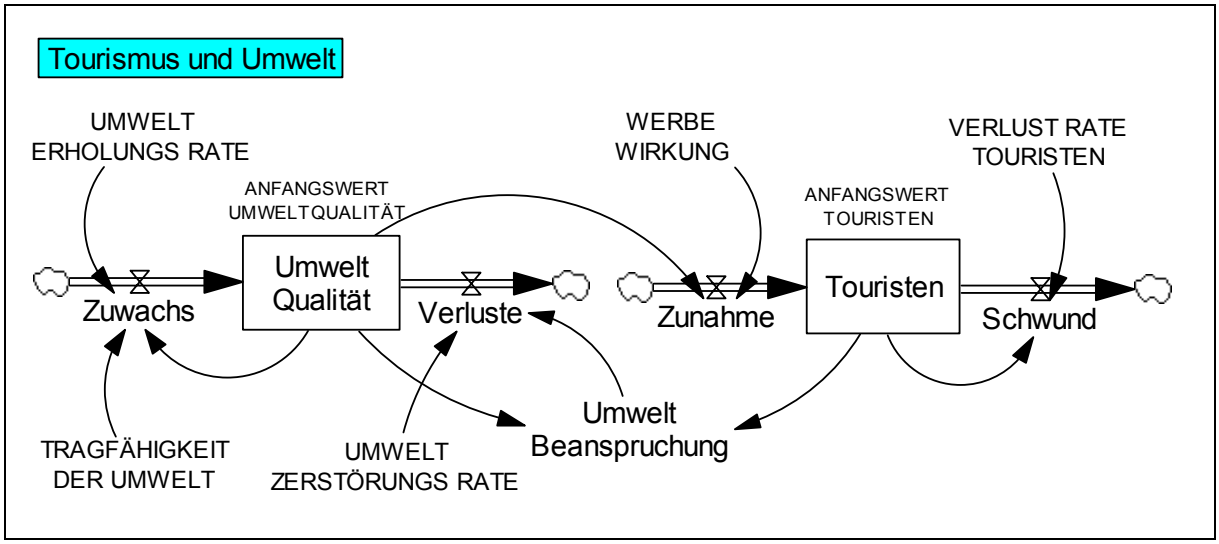
(4) Die Zunahme der *Touristen* hängt direkt von der *Umweltqualität* ab und kann durch WERBEWIRKUNG verstärkt werden.

(5) Der *Schwund* (= Rückgang) an Touristen wird durch die VERLUSTRATETOURISTEN beschrieben.

Das *VENSIM*-Modell

Das Modell kann fertig geladen werden[5,7]. Wesentlich ergiebiger aber ist es, begleitet vom Tutorial[6] und Handbuch, dieses Modell selbst zu erarbeiten.

Man beginnt mit der Darstellung des *Simulationsdiagramms*:



Die veränderlichen Parameter sind in Großbuchstaben dargestellt, die Zustandsgrößen in normaler Schrift und in den Kästchen finden sich die interessierenden Zustandsgrößen.

Die Zusammenhänge zwischen den Größen werden in Form von Gleichungen eingegeben. Die Zusammenfassung der Gleichungen findet sich in einem *VENSIM*-Dokument (im Original alphabetisch, hier nach der Art der Größen umgeordnet). Die „Units“ werde ich später besprechen.

(08) Touristen = INTEG (+ Zunahme – Schwund, ANFANGSWERT TOURISTEN)
Units: Touristen

(17) Zunahme = WERBE WIRKUNG ∗ UmweltQualität
Units: Touristen/Jahr

(06) Schwund = VERLUST RATE TOURISTEN ∗ Touristen
Units: Touristen/Jahr

(13) UmweltQualität = INTEG (+Zuwachs – Verluste, ANFANGSWERT UMWELT-QUALITÄT)
Units: Qualität

(18) Zuwachs = UMWELT ERHOLUNGS RATE * UmweltQualität ∗ (1 – UmweltQualität/TRAGFÄHIGKEIT DER UMWELT)
Units: Qualität/Jahr

(15) Verluste = UMWELT ZERSTÖRUNGS RATE ∗ UmweltBeanspruchung
Units: Qualität/Jahr

(12) UmweltBeanspruchung = Touristen ∗ UmweltQualität
Units: Qualität ∗ Touristen

Die Werte der Parameter sowie die Startwerte für *Umweltqualität* und *Touristen* werden eingetragen und können im Dokument nachgelesen bzw. dann auch ausgedruckt werden:

(01) ANFANGSWERT TOURISTEN = 0.1
Units: Touristen

(02) ANFANGSWERT UMWELTQUALITÄT = 1
Units: Qualität

(03) FINAL TIME = 20
Units: Jahr
The final time for the simulation.

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

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

(09) TRAGFÄHIGKEIT DER UMWELT = 1
Units: Qualität

(10) UMWELT ERHOLUNGS RATE = 1
Units: 1/Jahr

(11) UMWELT ZERSTÖRUNGS RATE = 1
Units: 1/(Touristen ∗ Jahr)

(14) VERLUST RATE TOURISTEN = 1
Units: 1/Jahr

(16) WERBE WIRKUNG = 5
Units: Touristen/(Qualität ∗ Jahr)

Nun kann man die erste Simulation laufen lassen. Anschließend lassen sich die Ergebnisse sowohl in Tabellen, als auch in Diagrammen ausgeben.

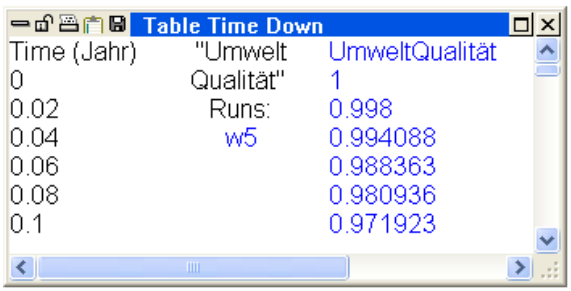


Table Time Down

Time (Jahr)	"Umwelt	UmweltQualität
0	Qualität"	1
0.02	Runs:	0.998
0.04	w5	0.994088
0.06		0.988363
0.08		0.980936
0.1		0.971923

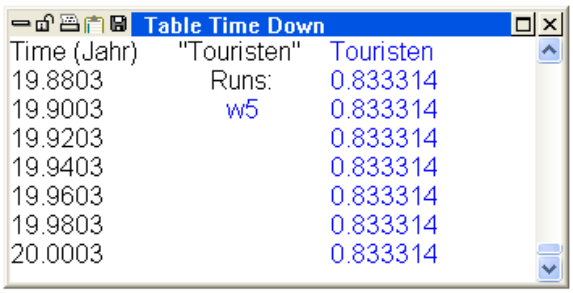
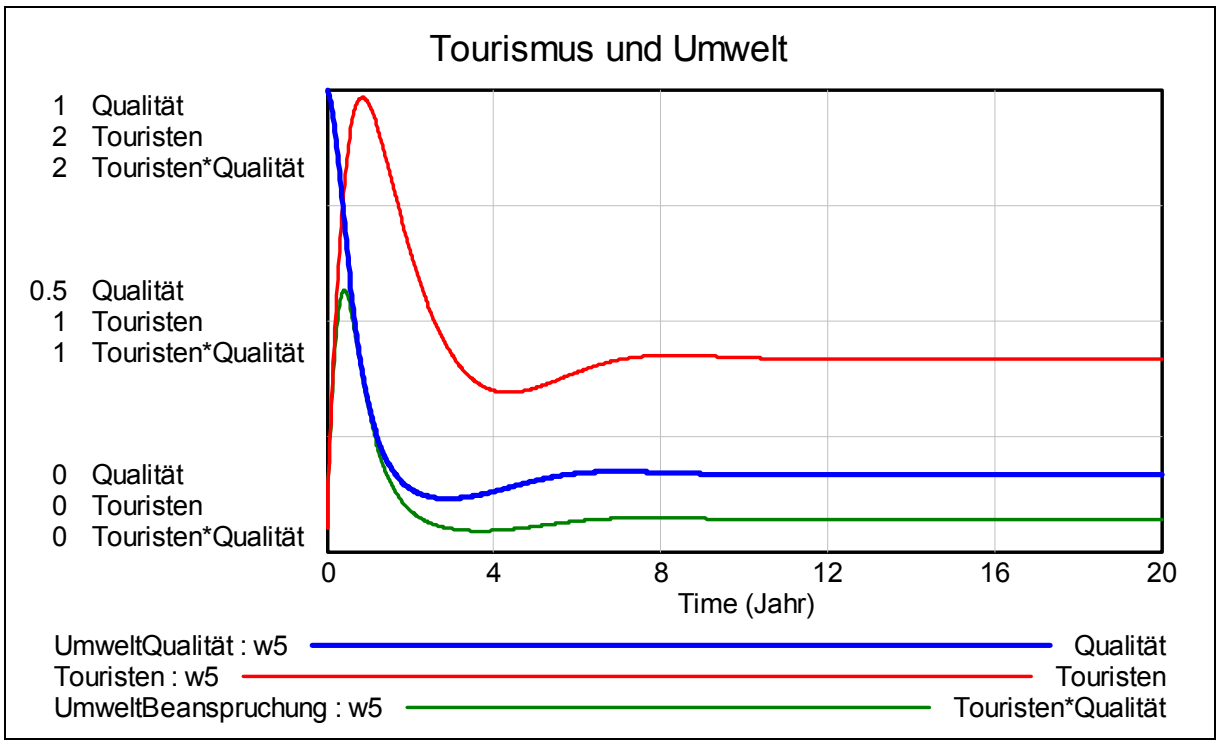


Table Time Down

Time (Jahr)	"Touristen"	Touristen
19.8803	Runs:	0.833314
19.9003	w5	0.833314
19.9203		0.833314
19.9403		0.833314
19.9603		0.833314
19.9803		0.833314
20.0003		0.833314

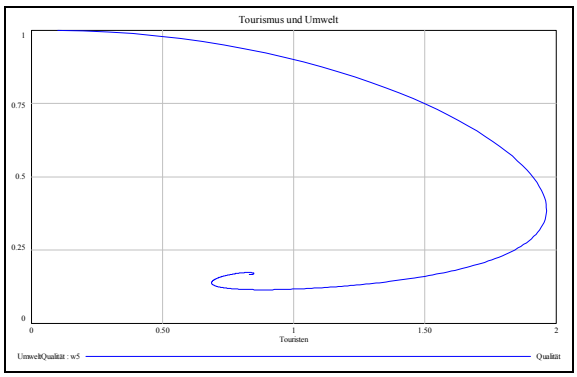
Anfang der *Umweltqualität*-Tabelle (links) und Ende der *Touristen*-Tabelle (rechts).

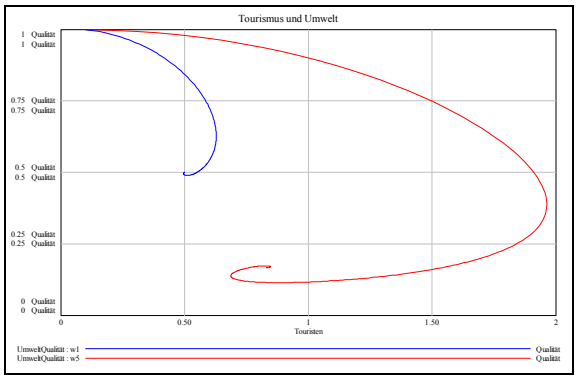
Das Diagramm zeigt die Entwicklung der *Umweltqualität*, der *Touristen* und der *Umweltbeanspruchung* für den Zeitraum von 20 Jahren.
(Beachte die unterschiedliche Skalierung auf der vertikalen Achse.)



Wenn uns der Zusammenhang zwischen *Touristen* und *Umweltqualität* interessiert, dann lassen wir das entsprechende Phasendiagramm zeichnen:

Bringt eine Verstärkung der Werbung etwas?
Wir vergleichen WERBEWIRKUNG = 5 mit WERBEWIRKUNG = 1:





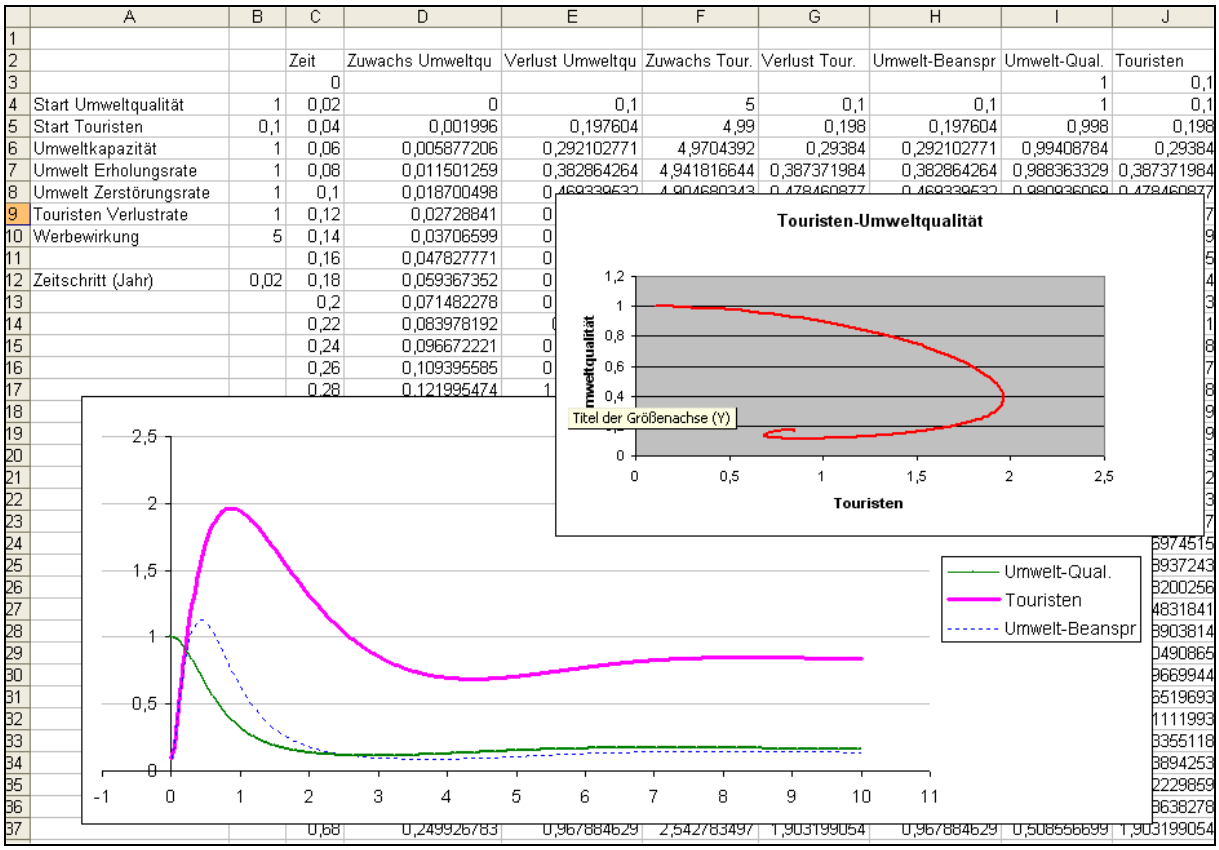
WERBEWIRKUNG = 5 (links) und vergleichsweise mit WERBEWIRKUNG = 1 (rechts).

Wir erkennen, dass die Erhöhung des Werbeeinsatzes kurzfristig eine deutliche Zunahme der *Touristen* nach sich zieht, aber eine deutliche Abnahme der *Umweltqualität* zur Folge hat.

Die Phasendiagramme für $w = 1, 2, 3, \ldots$ können auch gemeinsam dargestellt werden.

Das *MS-Excel*-Modell

Die „Gleichungen“ des *VENSIM*-Modells können einfach 1:1 in die Tabellenkalkulation übertragen werden.



	A	B	C	D	E	F	G	H	I	J
1										
2			Zeit	Zuwachs Umweltqu	Verlust Umweltqu	Zuwachs Tour.	Verlust Tour.	Umwelt-Beanspr	Umwelt-Qual.	Touristen
3			0						1	0,1
4	Start Umweltqualität	1	0,02	0	0,1	5	0,1	0,1	1	0,1
5	Start Touristen	0,1	0,04	0,001996	0,197604	4,99	0,198	0,197604	0,998	0,198
6	Umweltkapazität	1	0,06	0,005877206	0,292102771	4,9704392	0,29384	0,292102771	0,99408784	0,29384
7	Umwelt Erholungsrate	1	0,08	0,011501259	0,382864264	4,941816644	0,387371984	0,382864264	0,988363329	0,387371984
8	Umwelt Zerstörungsrate	1	0,1	0,018700498	0,469339532	4,904690343	0,478460877	0,469339532	0,980936069	0,478460877
9	Touristen Verlustrate	1	0,12	0,02728841	0					
10	Werbewirkung	5	0,14	0,03706599	0					
11			0,16	0,047827771	0					
12	Zeitschritt (Jahr)	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										6974515
25										8937243
26										8200256
27										4831841
28										8903814
29										0490865
30										9669944
31										6519693
32										1111993
33										8355118
34										8894253
35										2229859
36										8638278
37			0,68	0,249926783	0,967884629	2,542783497	1,903199054	0,967884629	0,508556699	1,903199054

Die entsprechenden Formeln sind:

Zeit	Zuwachs Umweltqu	Verlust Umweltqu	ZuwachsTour.	VerlustTour.
0				
=C3+$B$12	=$B$7*I4*(1-I4/$B$6)	=$B$8*H4	=$B$10*I4	=$B$9*J4

Umwelt-Beanspr	Umwelt-Qual.	Touristen
	=B4	=B5
=I4*J4	=I3+(D3-E3)*$B$12	=J3+(F3-G3)*$B$12

Die letzten Zeilen der Umwelt-Qual.- und Touristen-spalte sind:

0,166318761	0,83862131
0,166302328	0,838480759
0,166286419	0,838341377

Wir können diese Werte mit der Ausgabe des *VENSIM*-Modells vergleichen.

Das Differentialgleichungsmodell mit *DERIVE*

Das System der beiden Differentialgleichungen für u (Umweltqualität) und v (Touristen) ist aus der Beschreibung leicht herleitbar und gleich formuliert:

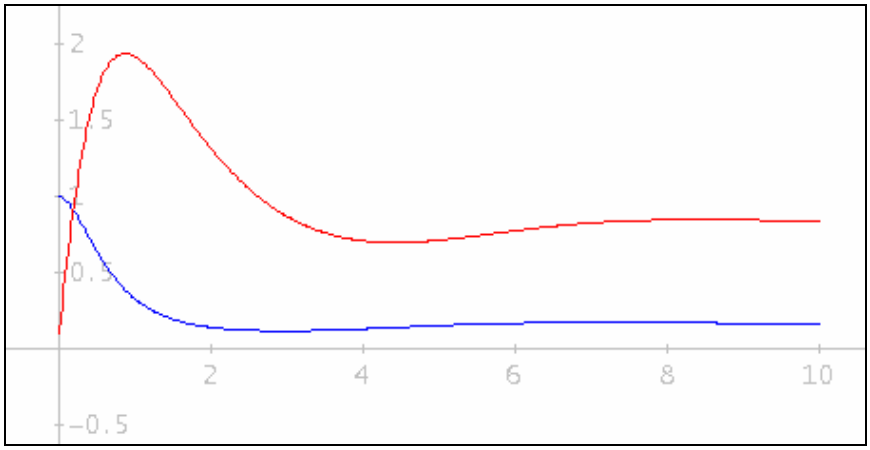
$$u' = uer \cdot u \cdot \left(1 - \frac{u}{ukap}\right) - uzr \cdot u \cdot v$$

$$v' = ww \cdot u - tvr \cdot v$$

Die Runge-Kutta-Methode zur numerischen Lösung ist in *DERIVE* implementiert, daher:

```
umw_tour(uer, uzr, tvr, ww, ukap, u_start, v_start, dx, n) :=
      RK([uer·u·(1 – u/ukap) – uzr·u·v, ww·u – tvr·v], [t, u, v],
      [0, u_start, v_start], dx, n)
```

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

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



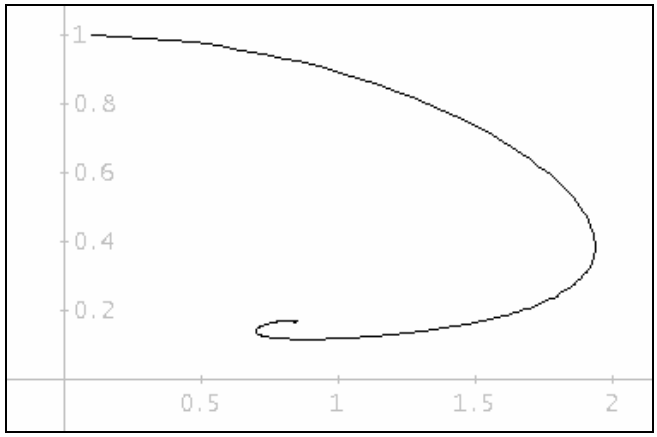
Die Graphen von *Umweltqualität* (blau) und *Touristen* (rot) sind schon bekannt.

Die letzte Zeile der Tabelle lautet:

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

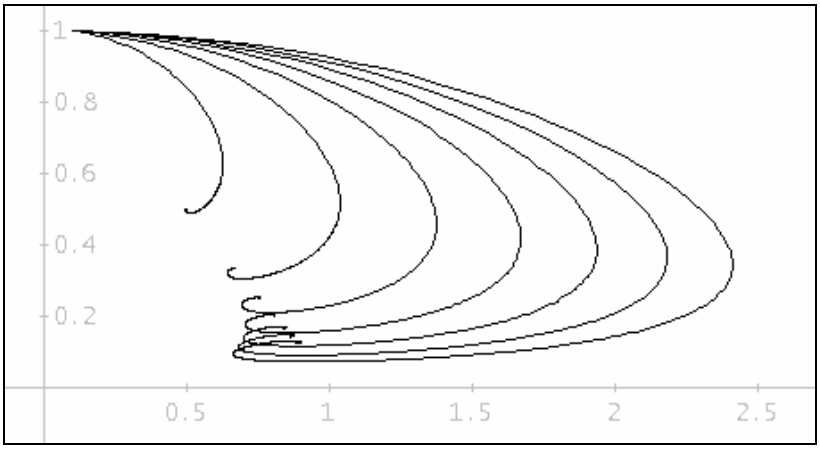
Wir zeichnen auch das Phasendiagramm für WERBEWIRKUNG $ww = 5$:

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



Mit Hilfe des VECTOR-Befehls lassen sich alle Phasendiagramme von *ww* = 1 bis *ww* = 7 in ein gemeinsames Koordinatensystem eintragen:

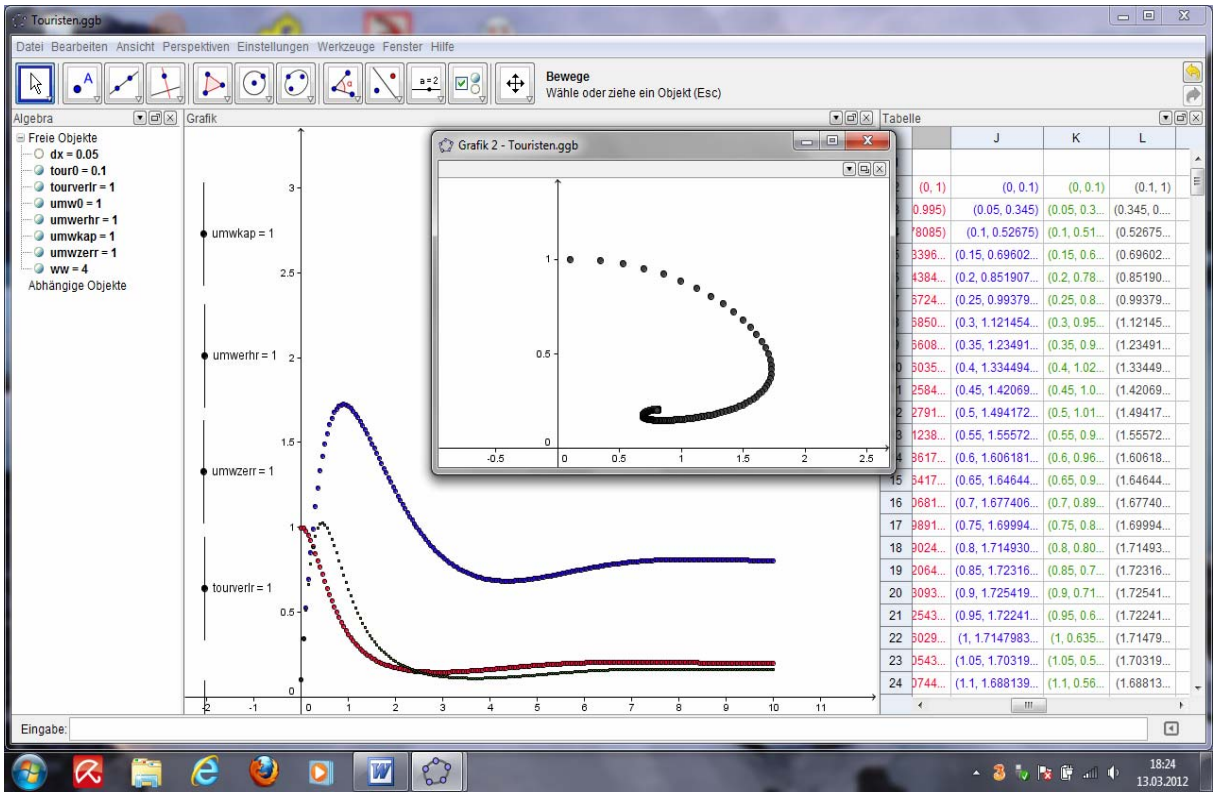
```
VECTOR((umw_tour(1, 1, 1, w, 1, 1, 0.1, 0.02, 500))↓↓[3, 2], w, 1, 7)
```



Das DGL-Modell wird uns später helfen, allfällige Fixwerte zu finden.

Die in *DERIVE* implementierten Schieberegler lassen sich – begründet durch die *DERIVE*-Programmierung - für dieses System leider nicht anwenden. Um den Einfluss der Parameter besser studieren zu können, versuche ich das Modell mit *GeoGebra* bzw. *TI-NspireCAS* zu erstellen. Es gibt aber auch mit *VENSIM* eine attraktive Möglichkeit, die Parameter zu variieren (siehe Ende dieses Kapitels).

Das Modell mit *GeoGebra*



Das Modell lässt sich erstellen, aber mit *dx* = 0,02 entstehen bei der doch für *GeoGebra* großen Tabelle unerträglich lange Rechenzeiten – bisweilen stürzt das System sogar ab. Mit *dx* = 0,05 bringt man das in den Griff.

Im zweiten Zeichenfenster kann ich das Phasendiagramm darstellen.

Die Ausarbeitung mit *TI-NspireCAS*

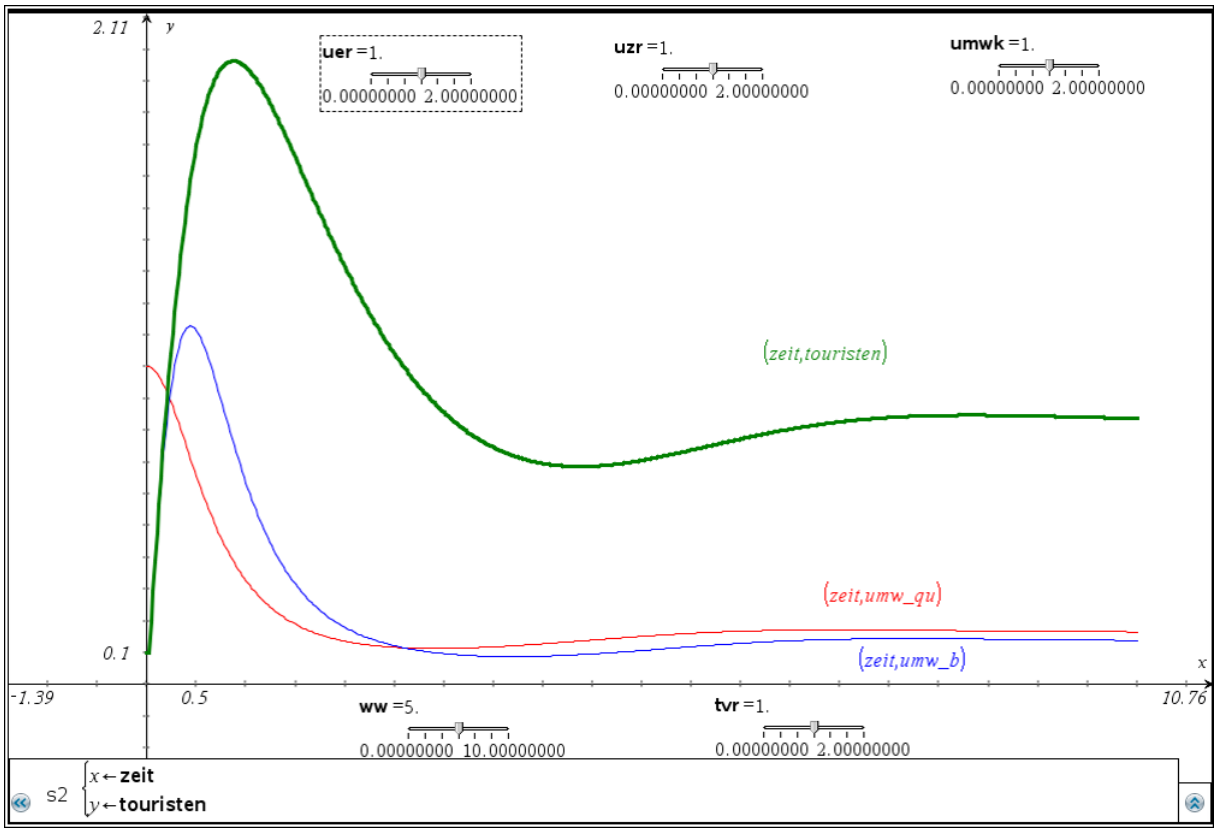
Wesentlich bessere Recheneigenschaften habe ich mit *TI-Nspire* (Version 3.0) gefunden. Hier machen 500 Zeitschritte zu je 0,02 Jahren nichts aus.

Für die fett gedruckten Parameter werden zuerst im Grafik-Fenster Schieberegler eingerichtet.
Dann kann man in der Tabellenkalkulation die Simulation durchführen. Es gelten die jeweils mit den Schiebereglern eingestellten Werte, da die Variablen verknüpft werden können.

	A	B	C	D	E	F	G	H	I	J	K	L
◆			zeit					umw_b	umw_qu	tourist...		
1	Tourismus		_	_	Verl_Umw	Zuw_Tou...	Verl_Tour	_	_	_		
2			_	_				_	_	_		
3	UmwErhRate	**1.000...**	0	0	0	0	0	0.10000...	1	0.10000...		
4	Umweltkap	**1.000...**	0.0200...	0.00000000	0.10000000	3.00000...	0.10000000	0.10000...	1.00000...	0.10000...		
5	UmwZerstRate...	**1.000...**	0.0400...	0.00199600	0.15768400	2.99400...	0.15800000	0.15768...	0.99800...	0.15800...		
6	WerbeWirkung...	**3.000...**	0.0600...	0.00508761	0.21362197	2.98465...	0.21472000	0.21362...	0.99488...	0.21472...		
7	TourVerlRate	**1.000...**	0.0800...	0.00919825	0.26761087	2.97214...	0.27011877	0.26761...	0.99071...	0.27011...		
8	dx	0.020...	0.1000...	0.01424382	0.31947435	2.95664...	0.32415933	0.31947...	0.98554...	0.32415...		
9			0.1200...	0.02013471	0.36906280	2.93832...	0.37680898	0.36906...	0.97944...	0.37680...		
10	Umw_ini	1	0.1400...	0.02677765	0.41625293	2.91739...	0.42803937	0.41625...	0.97246...	0.42803...		
11	Touristen_ini	0.100...	0.1600...	0.03407750	0.46094703	2.89402...	0.47782643	0.46094...	0.96467...	0.47782...		
12			0.1800...	0.04193883	0.50307196	2.86841...	0.52615037	0.50307...	0.95613...	0.52615...		
13			0.2000...	0.05026737	0.54257788	2.84074...	0.57299560	0.54257...	0.94691...	0.57299...		
14			0.2200...	0.05897125	0.57943675	2.81120...	0.61835056	0.57943...	0.93706...	0.61835...		
15			0.2400...	0.06796206	0.61364071	2.77997...	0.66220765	0.61364...	0.92665...	0.66220...		
16			0.2600...	0.07715570	0.64520042	2.74723...	0.70456304	0.64520...	0.91574...	0.70456...		
17			0.2800...	0.08647311	0.67414320	2.71315...	0.74541651	0.67414...	0.90438...	0.74541...		
18			0.3000...	0.09584076	0.70051129	2.67789...	0.78477126	0.70051...	0.89263...	0.78477...		
19			0.3200...	0.10519101	0.72436004	2.64161...	0.82263370	0.72436...	0.88053...	0.82263...		
20			0.3400...	0.11446235	0.74575616	2.60446...	0.85901329	0.74575...	0.86815...	0.85901...		
21			0.3600...	0.12359948	0.76477601	2.56658...	0.89392229	0.76477...	0.85552...	0.89392...		
22			0.3800...	0.13255330	0.78150400	2.52811...	0.92737556	0.78150...	0.84270...	0.92737...		

In den Spalten C, H, I, J werden in der allerersten Zeile Listennamen eingegeben, über die bequem die Streudiagramme gezeichnet werden können:

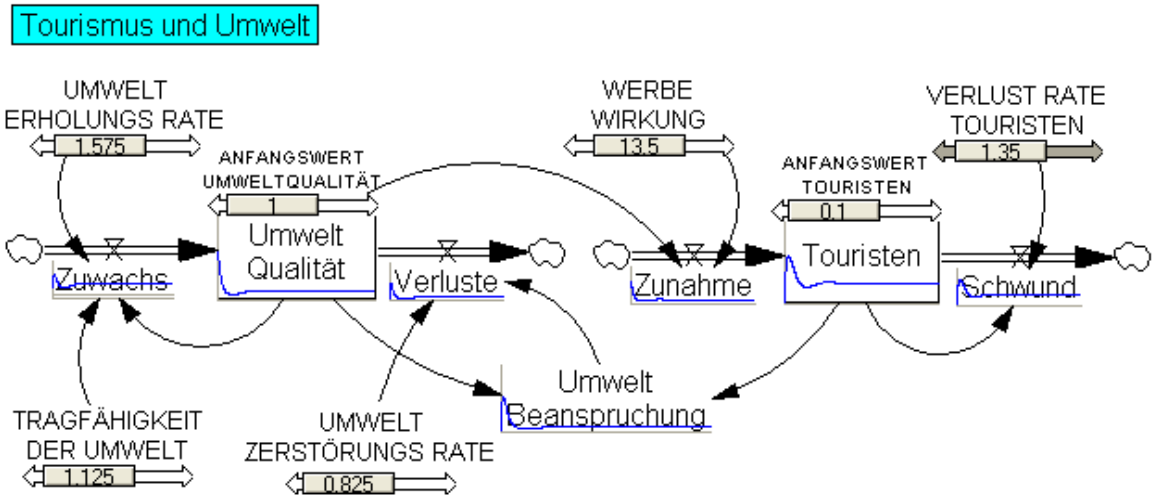
Die errechneten Punkte werden verbunden dargestellt und man erhält je nach Auswahl der Listen eine sehr ansprechende Grafik, die auch prompt auf die Schieberegler reagiert.



Hier ist das Phasendiagramm für *ww* = 2:

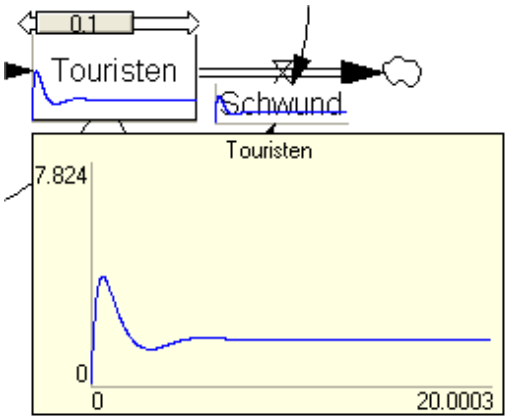
SyntheSim mit *VENSIM*

In *VENSIM* gibt es eine Art Schieberegler, die es erlaubt, alle Zustandsgrößen (Box Variable) gleichzeitig (simultan) bei Veränderung der Parameter zu beobachten. Die Beobachtungsfenster sind allerdings verhältnismäßig klein. SyntheSim wird über das Menü aktiviert und zeigt dann das folgende Bild.

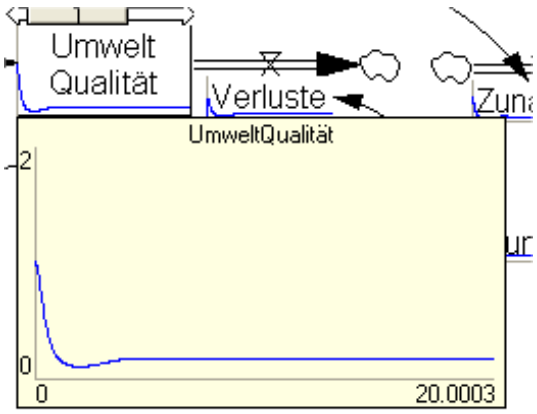


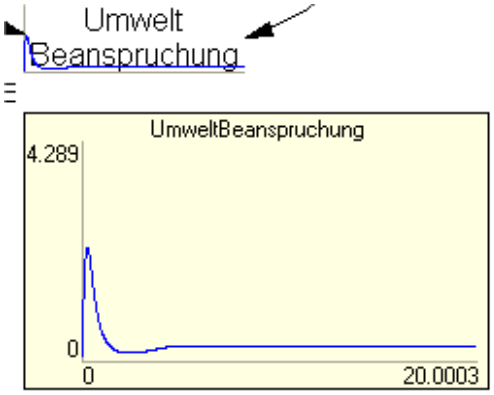
Ich habe jetzt an einigen „Parameterschrauben“ gedreht und man sieht in blau die Kurven in einer Miniaturausführung. Wenn man allerdings die Maus über eines dieser Minifenster bewegt, dann wird dieses etwas vergrößert.

Für die geänderten Parameter betrachten wir die vergrößerten Darstellungen für die *Touristen*, die *Umweltqualität* und die *Umweltbeanspruchung*.



Diese Grafiken haben wohl nicht die Qualität der *Nspire* oder *GeoGebra*-Diagramme, sie sind aber ohne zusätzlichen Aufwand leicht zu erstellen. Dies ist allerdings nur für Zeitdiagramme möglich.





Die analytische Berechnung der Fixwerte für Umweltqualität und Touristen

Die Fixwerte ergeben sich aus der Forderung, dass die Zuwächse für die Zustandsgrößen den Wert Null annehmen müssen.

Daher versuchen wir das folgende Gleichungssystem (mit *DERIVE*) zu lösen:

$$\#13:\quad \text{SOLUTIONS}\left(0 = uer \cdot u \cdot \left(1 - \frac{u}{ukap}\right) - vzr \cdot u \cdot v \;\wedge\; 0 = ww \cdot u - tvr \cdot v,\ [u,\ v]\right)$$

$$\#14:\quad \begin{bmatrix} 0 & 0 \\ \dfrac{tvr \cdot uer \cdot ukap}{tvr \cdot uer + ukap \cdot vzr \cdot ww} & \dfrac{uer \cdot ukap \cdot ww}{tvr \cdot uer + ukap \cdot vzr \cdot ww} \end{bmatrix}$$

$$\#15:\quad \left[\frac{1 \cdot 1 \cdot 1}{1 \cdot 1 + 1 \cdot 1 \cdot 5},\ \frac{1 \cdot 1 \cdot 5}{1 \cdot 1 + 1 \cdot 1 \cdot 5}\right]$$

$$\#16:\quad [0.1666666666,\ 0.8333333333]$$

Vergleiche für *ww* = 5 die letzten Zeilen der Tabellen in den vorangegangenen Modelle!

---

[1] Hartmut Bossel, *SystemZoo1, 2, 3*, Books on Demand, Norderstedt

[2] *Vensim PLE*, Simulationssoftware, für den Unterrichtsgebrauch frei downloadbar von 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
(Das ZOO MDL.zip Archiv enthält alle Computersimulationen, allerdings in englischer Sprache.)

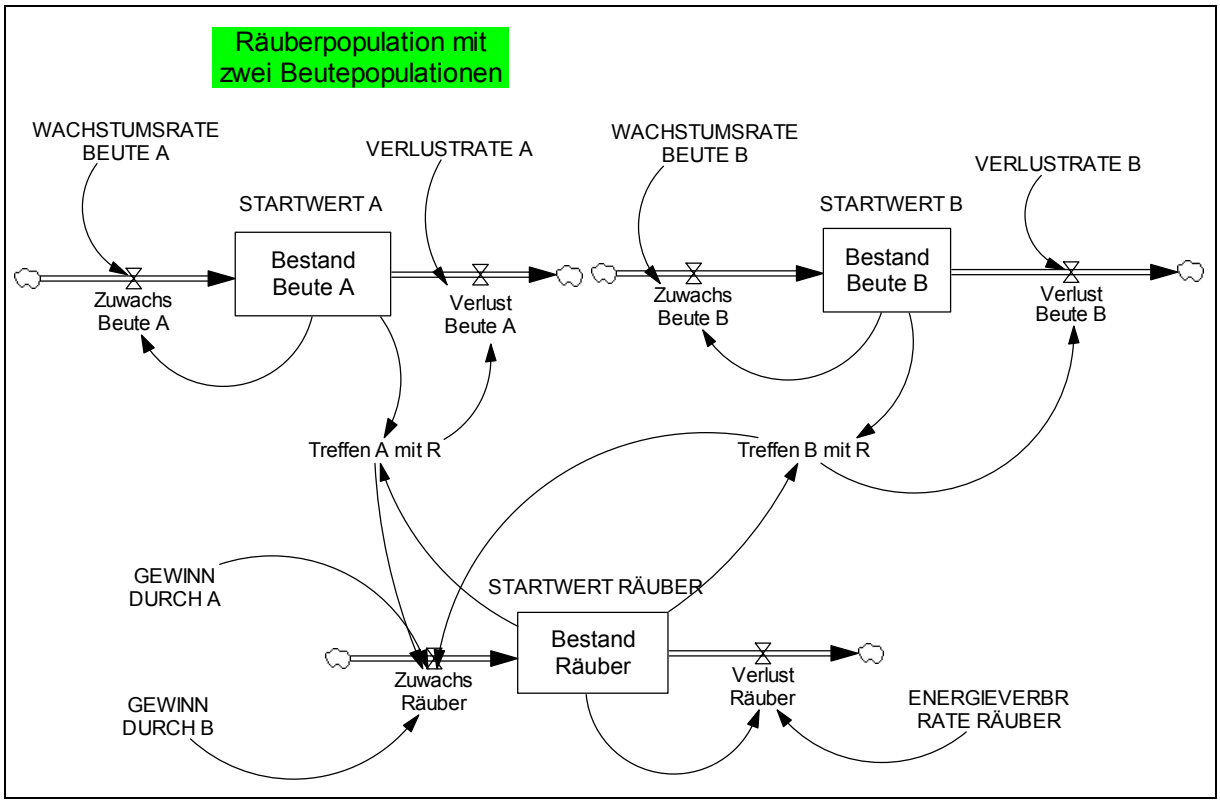
2 Räuber und Beute (mal 2)

Eine Variation der klassischen Räuber-Beute-Modellierung

Eine Räuberpopulation bezieht ihre Energie hauptsächlich aus zwei Beutepopulationen.

Der Räuberbestand ist über die Erfolge beim Zusammentreffen mit den Beständen der Beutepopulationen (mit den jeweiligen Erfolgsraten) gekoppelt. Die Bestände der Beutetiere wachsen mit – unterschiedlichen – Wachstumsraten. Den Erfolgsraten der Räuber stehen spezifische Verlustraten der Beutetiere gegenüber.

Es wird wieder mit dem *VENSIM PLE* Simulationsdiagramm und der Beschreibung der Parameter und Gleichungen begonnen.



Alle Parameter und notwendigen Gleichungen werden eingegeben und können im *VENSIM*-Dokument nachgelesen bzw. auch ausgedruckt werden. Ich habe auf die Angabe von Einheiten verzichtet. Damit wäre auch eine Dimensionsanalyse möglich. Die Nummerierung wurde auch unterdrückt.

STARTWERT A = 1

STARTWERT B = 1

STARTWERT RÄUBER = 1

GEWINN DURCH A = 0.1

GEWINN DURCH B = 0.1

VERLUSTRATE A = 0.1

VERLUSTRATE B = 0.1

Löwin in der Serengeti, Tansania

WACHSTUMSRATE BEUTE A = 0.1

WACHSTUMSRATE BEUTE B = 0.12

ENERGIEVERBR RATE RÄUBER = 0.1

Zuwachs Beute A = WACHSTUMSRATE BEUTE A ∗ Bestand Beute A

Zuwachs Beute B = WACHSTUMSRATE BEUTE B ∗ Bestand Beute B

Treffen A mit R = Bestand Beute A ∗ Bestand Räuber

Treffen B mit R = Bestand Beute B ∗ Bestand Räuber

Verlust Beute A = VERLUSTRATE A ∗ Treffen A mit R

Verlust Beute B = VERLUSTRATE B ∗ Treffen B mit R

Zuwachs Räuber = GEWINN DURCH A ∗ Treffen A mit R+
GEWINN DURCH B ∗ Treffen B mit R

Verlust Räuber = ENERGIEVERBR RATE RÄUBER ∗ Bestand Räuber

Bestand Beute A = INTEG (+Zuwachs Beute A – Verlust Beute A,
STARTWERT A)

Bestand Beute B = INTEG (+Zuwachs Beute B – Verlust Beute B,
STARTWERT B)

Bestand Räuber = INTEG (+Zuwachs Räuber – Verlust Räuber,
STARTWERT RÄUBER)

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.

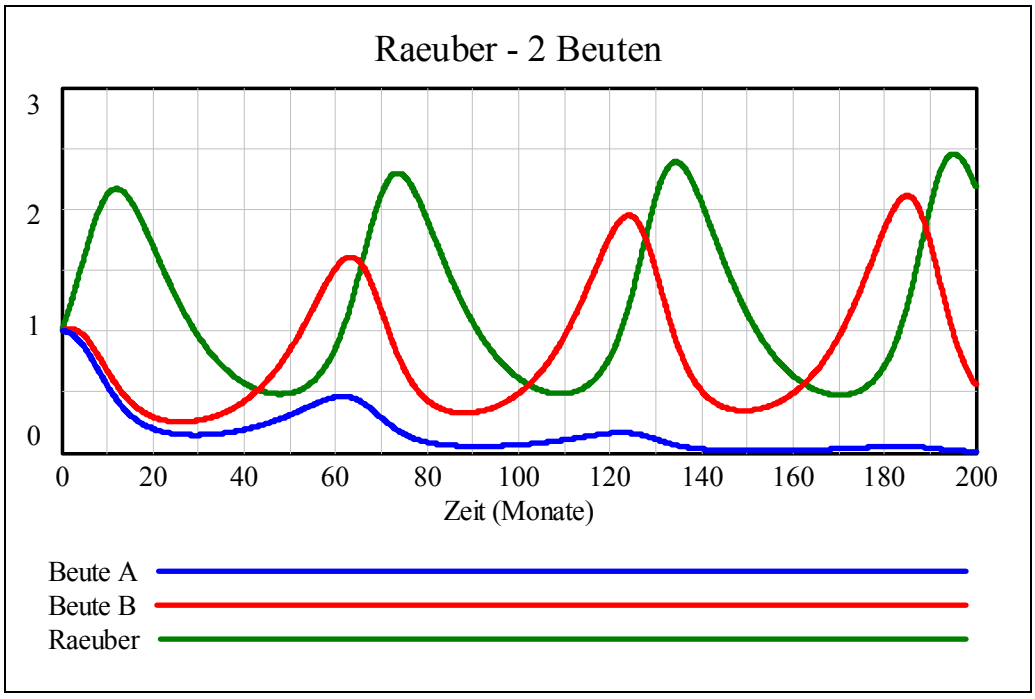
TIME STEP = 0.1
Units: Month [0,?]
The time step for the simulation.

Beutetiere in der Serengeti, Tansania

Die Reihenfolge in der Ausgabe des Dokuments wurde von mir geändert.

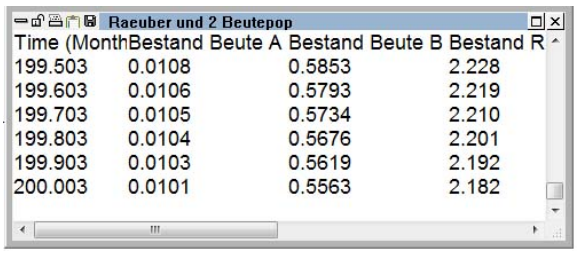
Im Original wird mit einer Schrittweite TIME STEP von 0,05 gerechnet. Die doppelte Schrittweite bringt keine erkenntliche Verschlechterung.

Im Anschluss können die Diagramme bzw. die Tabellen erzeugt und ausgegeben werden. Alle Diagramme und Tabellen lassen sich leicht über die Zwischenablage in andere Dokumente einbinden.



Hier sind die letzten Zeilen der Tabelle (zum späteren Vergleich) gefolgt vom Phasendiagramm:

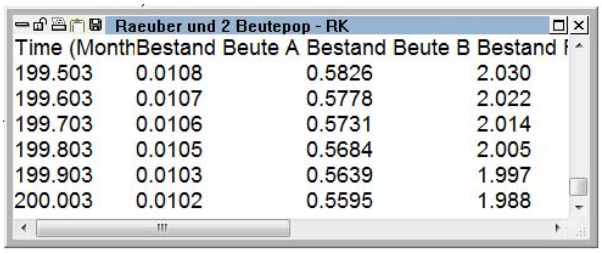
Für die Berechnung bietet *VENSIM PLE* die Wahl zwischen der Euler-Methode und der Runge-Kutta-Methode. Die hier dargestellten Graphen basieren alle auf der Euler-Methode



Raeuber und 2 Beutepop

Time (Month	Bestand Beute A	Bestand Beute B	Bestand R
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
200.003	0.0101	0.5563	2.182

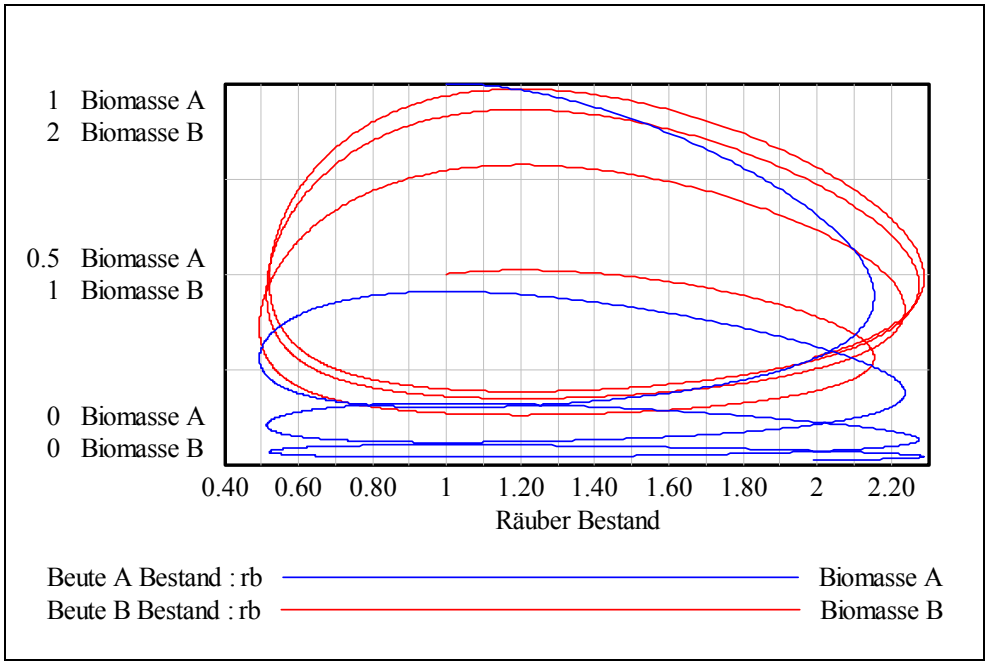
mit der Euler-Methode



Raeuber und 2 Beutepop - RK

Time (Month	Bestand Beute A	Bestand Beute B	Bestand I
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
200.003	0.0102	0.5595	1.988

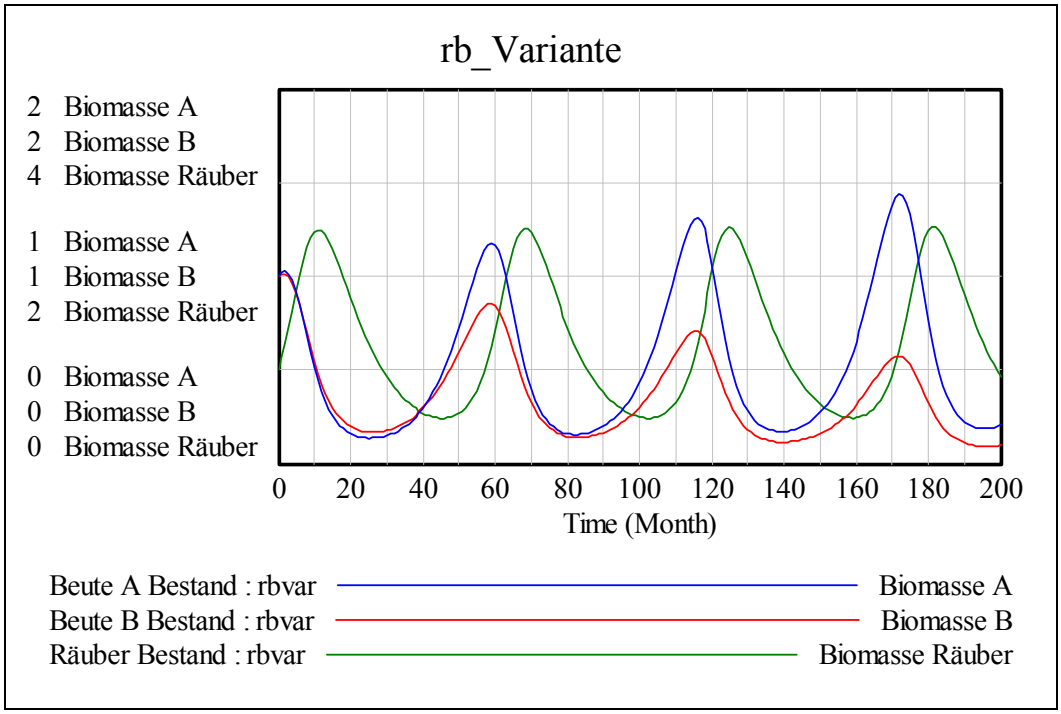
mit der Runge-Kutta-Methode



Variante:

WACHSTUMSRATE BEUTE A = 0.15
VERLUSTRATE A = 0.15
GEWINN DURCH A = 0.12

SAVEPER = 1



Hier werden nur die Werte am Ende eines jeden ganzen Monats gespeichert (SAVEPER = 1).

Das Differentialgleichungsmodell mit *DERIVE*

Für die Bearbeitung mit *DERIVE* formuliere ich das entsprechende Differentialgleichungs-system, das anschließend wieder mit Runge-Kutta näherungsweise gelöst wird. Ich verwende aber die *Systemzoo*-Schrittweite 0,05 und vergleiche die erhaltenen Werte für die letzten Monate mit denen von *VENSIM PLE*.

$$r' = r \cdot (ga \cdot ba + gb \cdot bb + vr)$$
$$ba' = wa \cdot ba - r \cdot ba \cdot va \qquad ; r(0) = ba(0) = bb(0) = 1$$
$$bb' = wb \cdot bb - r \cdot bb \cdot vb$$

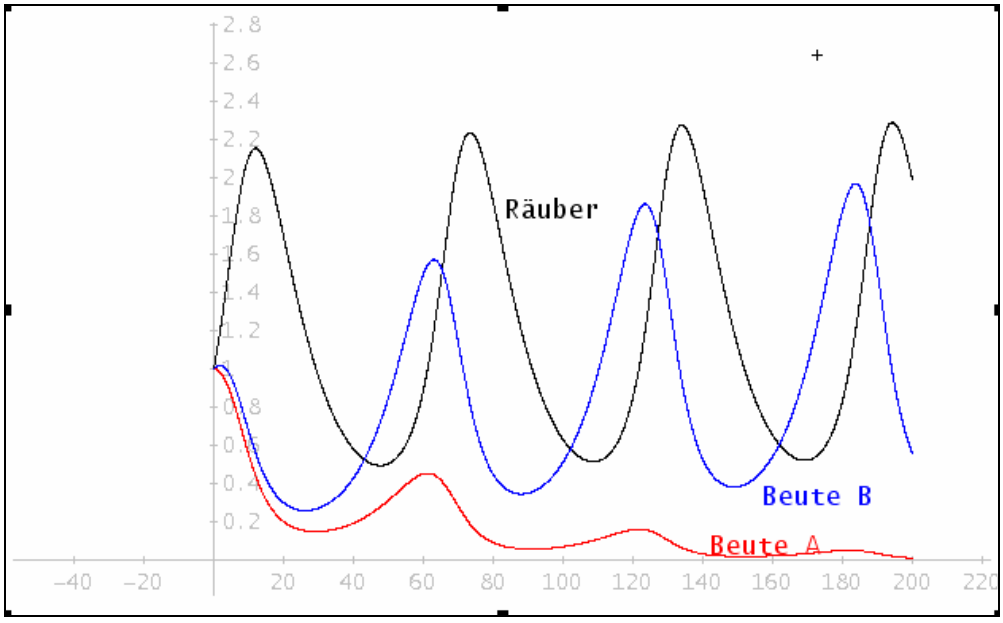
Die entsprechende Funktion wird mit allen Parametern definiert

```
rbb(wa, wb, va, vb, vr, ga, gb, ba_ini, bb_ini, rb_ini, dt, n) := RK([r·(ga·ba
  + gb·bb - vr), wa·ba - va·ba·r, wb·bb - vb·bb·r], [t, r, ba, bb], [0,
  rb_ini, ba_ini, bb_ini], dt, n)
```

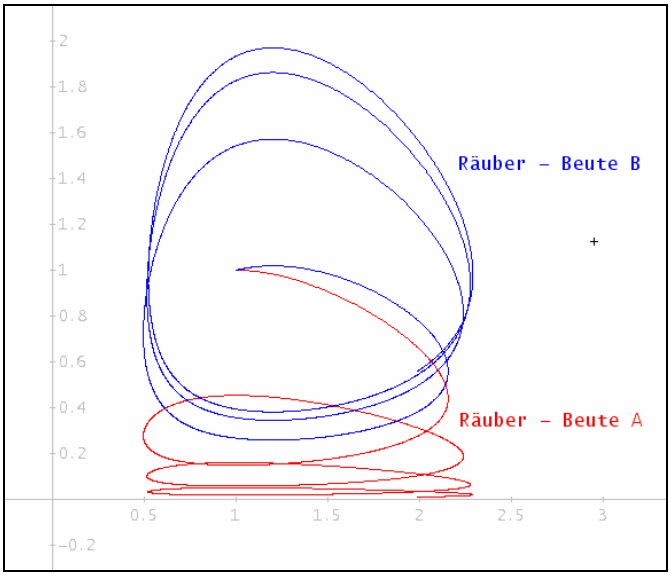
und anschließend ausgewertet (Vergleiche mit den *VENSIM*-Werten!).

$$\begin{bmatrix} 199.85 & 2.000997472 & 0.01040077792 & 0.5661621969 \\ 199.9 & 1.996753928 & 0.01034896194 & 0.5639052381 \\ 199.95 & 1.992496490 & 0.01029762295 & 0.5616692132 \\ 200 & 1.988225495 & 0.01024675712 & 0.5594539828 \end{bmatrix}$$

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




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



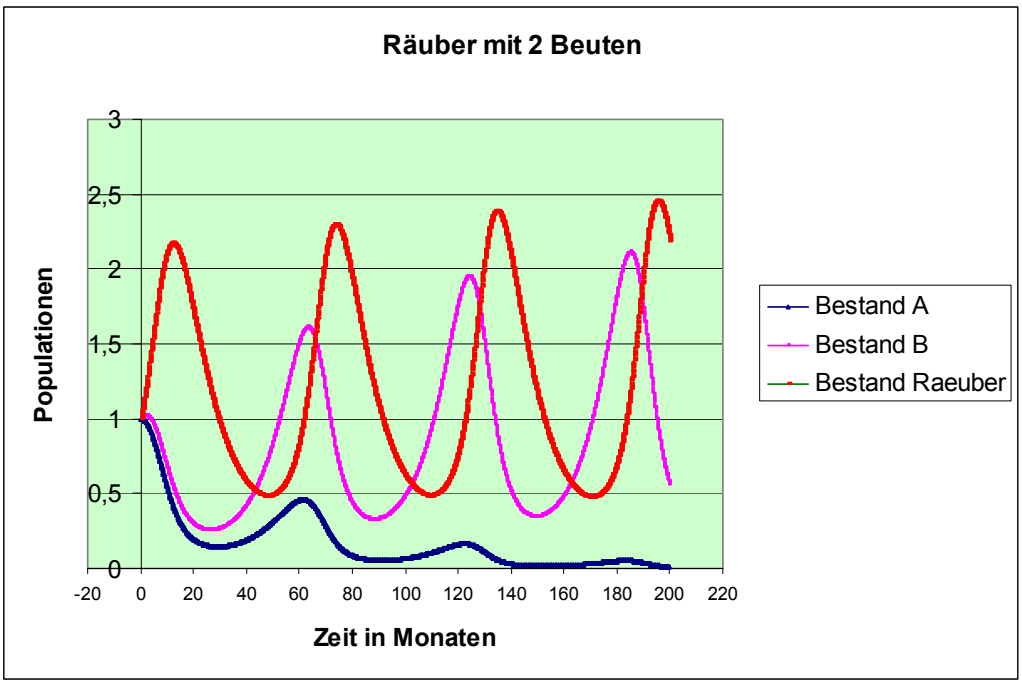
Das *MS-Excel*-Modell

Die „Gleichungen“ des *VENSIM*-Modells können einfach in die Tabellenkalkulation übertragen werden.

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

	A	B	C	D	E	F	G	H	I	J	K
1											
2			Zeit	Zuwacl	Zuwach	Verlust	Verlust	Zuwach	Bestand A	Bestand B	Bestand Raeuber
3			0						1	1	1
4	Anfangswert Beute A	1	0,1	0,1	0,12	0,1	0,1	0,1	1	1,002	1,01
5	Anfangswert Beute B	1	0,2	0,1	0,1202	0,101	0,101	0,1012	0,9999	1,0039038	1,0201202
6	Anfangswert Raeuber	1	0,3	0,1	0,1205	0,102	0,102	0,1024	0,99969882	1,00570962	1,030360205
7	Wachstum A	0,1	0,4	0,1	0,1207	0,103	0,104	0,1036	0,99939531	1,007415704	1,040719534
8	Wachstum B	0,12	0,5	0,1	0,1209	0,104	0,105	0,1048	0,99898836	1,00902032	1,051197613
9	Verlust A	0,1	0,6	0,1	0,1211	0,105	0,106	0,106	0,9984769	1,010521767	1,061793776
10	Verlust B	0,1	0,7	0,1	0,1213	0,106	0,107	0,1071	0,9978599	1,011918371	1,072507261
11	Verlust Raeuber	0,1	0,8	0,1	0,1214	0,107	0,109	0,1083	0,99713638	1,013208493	1,083337206
12	Gewinn durch A	0,1	0,9	0,1	0,1216	0,108	0,11	0,1095	0,9963054	1,01439053	1,094282648
13	Gewinn durch 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	Zeitschritt (Monat)	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

Das Diagramm ist leicht erzeugt (das Phasendiagramm wird hier nicht gezeigt).



Vergleiche auch hier die Werte der letzten Zeilen der *Excel*-Tabelle (2000 Zeilen!) mit den *VENSIM*- und den *DERIVE*-Werten:

Bestand A	Bestand B	Bestand Raeuber
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 *Systemzoo 2* wird die Frage gestellt, welche Maßnahme(n) ergriffen werden könnten, dass Population A trotz des Nachteils bei der Wachstumsrate die Population B überleben könnte?

Mit Hilfe von Schiebereglern zur Variation der Parameter könnte diese Frage leichter beantwortet werden. Schieberegler mit *MS-Excel* werden in späteren Kapitel eingeführt.

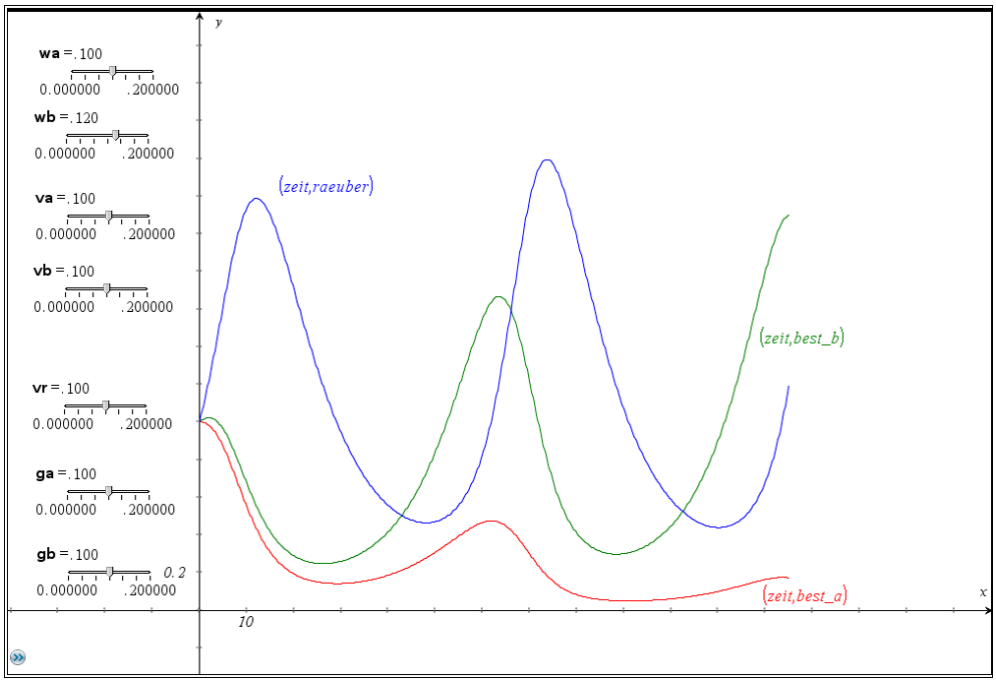
Da die Bearbeitung mit GeoGebra wohl möglich ist, aber (zu) lange Rechenzeiten verursacht, führen wir die Schieberegler mit *TI-NspireCAS* ein.

Die Ausarbeitung mit *TI-NspireCAS*

Um die Rechenzeit für eine größere Anzahl von Monaten in Grenzen zu halten, wird für den Zeitschritt der Wert 0,25 angenommen (im *Systemzoo* beträgt er 0,05!).

	A	B	C zeit	D	E	F	G	H	I best_a	J best_b	K raeuber
◆											
1			Zeit	Zuw A	Zuw B	Verl A	Verl B	Zuw R	Bestand A...	Bestand B..	Bestand ...
2			0	0.100000	0.120000	0.100000	0.100000	0.100000	1	1	1
3	Anfangswert A	1	0.25...	0.100000	0.120600	0.102500	0.103013	0.103013	1.000000	1.005000	1.025000
4	Anfangswert B	1	0.50...	0.099938	0.121128	0.105010	0.106063	0.105997	0.999375	1.009397	1.050753
5	Anfangswert Räub...	1	0.75...	0.099811	0.121580	0.107521	0.109143	0.108939	0.998107	1.013163	1.077252
6			1.00...	0.099618	0.121953	0.110027	0.112246	0.111824	0.996179	1.016272	1.104487
7	Wachstum A	**0.1000...**	1.25...	0.099358	0.122244	0.112517	0.115362	0.114635	0.993577	1.018699	1.132443
8	Wachstum B	**0.1200...**	1.50...	0.099029	0.122450	0.114982	0.118481	0.117353	0.990287	1.020419	1.161102
9	Verlust A	**0.1000...**	1.75...	0.098630	0.122569	0.117413	0.121593	0.119962	0.986299	1.021412	1.190440
10	Verlust B	**0.1000...**	2.00...	0.098160	0.122599	0.119798	0.124686	0.122441	0.981603	1.021656	1.220431
11	Verlust Räuber	**0.1000...**	2.25...	0.097619	0.122536	0.122126	0.127748	0.124770	0.976194	1.021134	1.251041
12	Gewinn durch A	**0.1000...**	2.50...	0.097007	0.122380	0.124385	0.130766	0.126928	0.970067	1.019831	1.282233
13	Gewinn durch B	**0.1000...**	2.75...	0.096322	0.122128	0.126564	0.133727	0.128894	0.963222	1.017734	1.313965
14			3.00...	0.095566	0.121780	0.128650	0.136616	0.130647	0.955662	1.014835	1.346189
15	Zeitschritt (Monat)	0.2500...	3.25...	0.094739	0.121335	0.130631	0.139419	0.132165	0.947391	1.011126	1.378851
16			3.50...	0.093842	0.120793	0.132494	0.142122	0.133427	0.938418	1.006605	1.411892

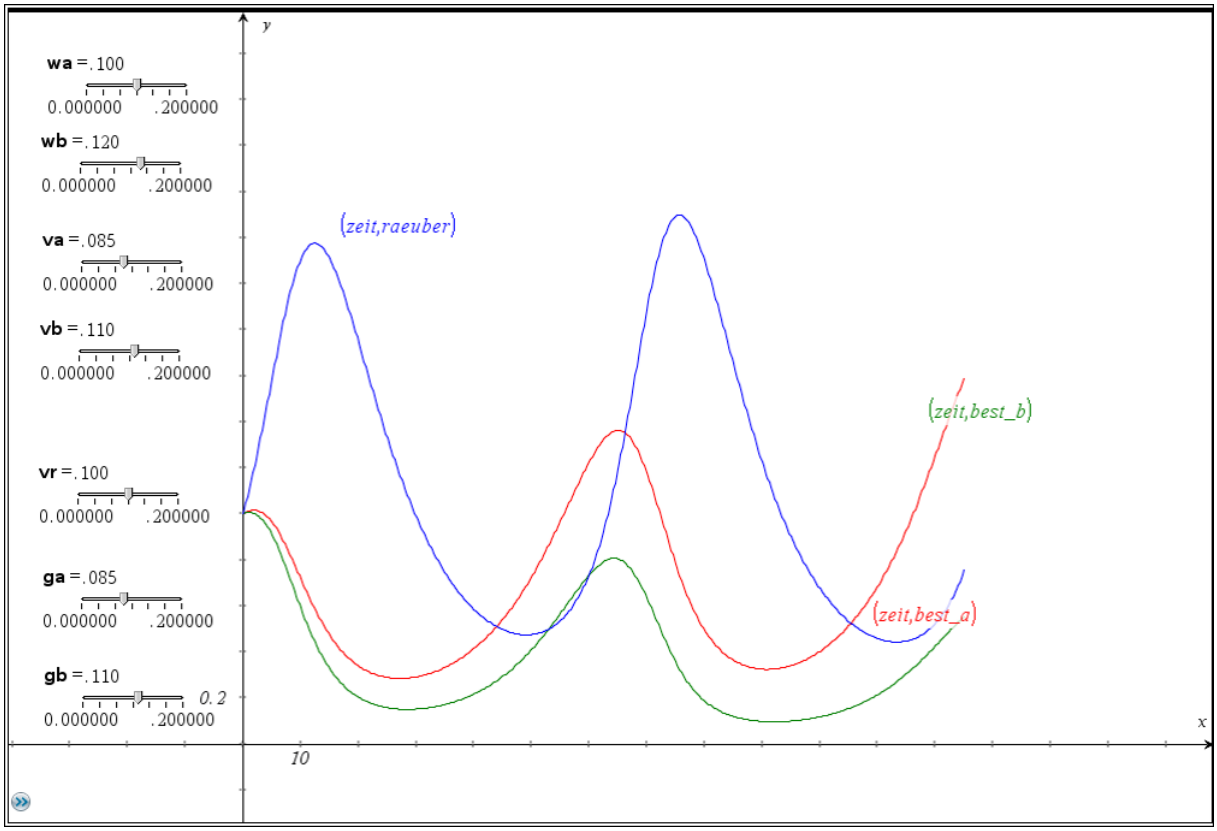
Hier sind es „nur“ 125 Monate:



Die Bestände für A, B und Räuber betragen hier: 0,1716, 2,0971 und 1,1919. In der Excel-Datei sind die entsprechenden Werte: 0,1590, 1,9387 und 1,3307.

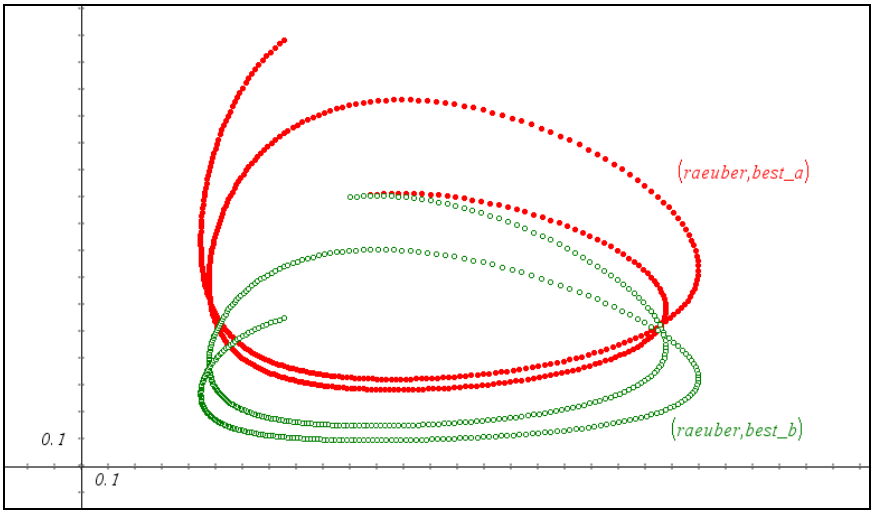
Das Diagramm sieht sehr ähnlich aus. Die doch sehr grobe Vereinfachung verfälscht offensichtlich die Modellierung nicht wesentlich.

Was könnte getan werden, dass A die Beutepopulation B überlebt?



Wie die Grafik zeigt, müssten geeignete Maßnahmen ergriffen werden, welche die Verlustrate von A verringern und die von B erhöhen. A müsste für die Räuber weniger „attraktiv“ gemacht werden. Können derartige Schutzmechanismen eingeführt werden?

Abschließend erzeugen wir das „Wunsch“-Räuber-Beute-Diagramm:



3 Zusammenbruch eines Ökosystems

Eine aufwändigere Simulation mit historischem Hintergrund

Bossel zitiert eine Quelle, die den Zusammenbruch der Weißwedelhirschpopulation im Kaibab Forest (Nordseite des Grand Canyon) als Folge des Abschusses von Räubern, die sich zum Teil von diesen Hirschen ernähren, deutlich macht.

Vor 1907 gab es auf einer Fläche von ca. 320 000 Hektar einen Bestand von etwa 4 000 Hirschen. Innerhalb von 15 bis 20 Jahren wurde der Abschuss der Räuber (Puma, Wölfe und Kojoten) forciert und etwa 8 000 Tiere wurden abgeschossen, was einen rasanten Anstieg der Hirschpopulation nach sich zog.

Weißwedelhirsch (*Odocoilus virginianus*)

Sycamore Canyon, Kaibab National Forest

1918 hatte sich der Hirschbestand mehr als verzehnfacht. Damit wurden aber auch die Futterquellen überbeansprucht. Bis 1924 hatte die Hirschpopulation eine Größe von 100 000 Tieren erreicht. Aufgrund des nun einsetzenden Nahrungsmangels sind aber während der beiden nächsten Winter 60% der Tiere verendet.

Die Vegetation wurde derart zerstört, dass dann nur etwa die Hälfte der Hirschpopulation verglichen mit der Größe vor diesem Geschehen auf Dauer existieren konnte.

Goodman hat 1974 versucht, das System mit einem Modell zu simulieren, wobei sich die Ergebnisse zufrieden stellend mit den realen Entwicklungen decken.

Zur Erklärung des Modells:

Die *Hirsche* ernähren sich auf einer FLÄCHE (320 000 ha) vom *Futter*. Der *Futterzuwachs* wird durch die *Nachwachszeit* reguliert. Die *ZuwachsrateHirsche* ist eine Funktion des *Futterangebots.* Dieses ist die für jeden Hirsch verfügbare Futtermenge. Der *Futterbedarf* richtet sich nach dem Bestand der *Hirsche* und dem TAGESBEDARF eines Hirschen (2 000 Kcal). Das *Futter* wächst entsprechend der MAXIMALEN FUTTERKAPAZITÄT (480 Mio Kcal) nach, wobei sich der *Futterzuwachs* nach der *Nachwachszeit* richtet, die ihrerseits von der *Bewuchsdichte* funktional abhängt.

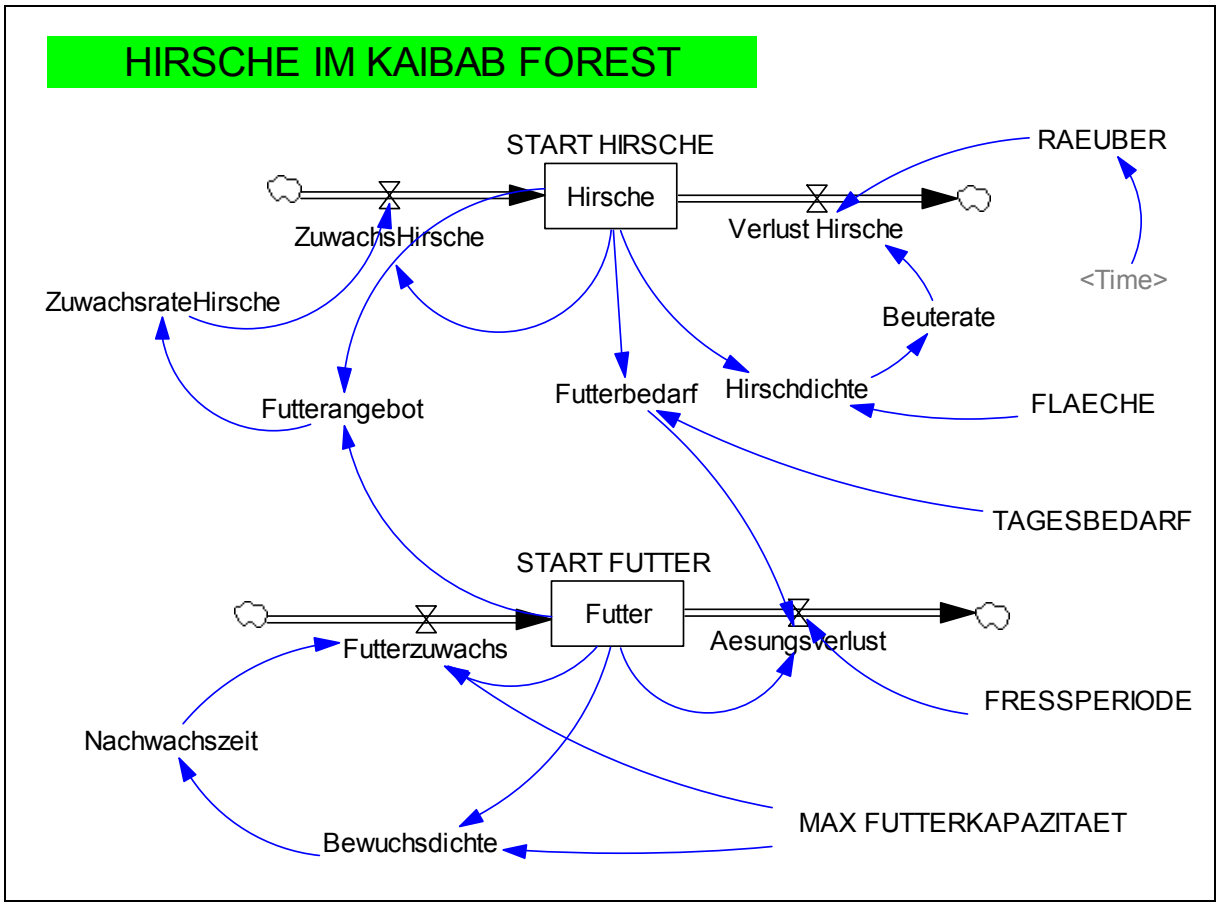
Nun kommt noch das Raubzeug ins Spiel: Die Population der *Hirsche* erleidet den *Verlust Hirsche* durch die RÄUBER, deren Menge durch die Abschusszahlen linear abnimmt. Die *Beuterate* der RÄUBER ist eine Funktion der *Hirschdichte* (= Hirsche/ha).

Im *Systemzoo* wird eine genauere Erläuterung der Parameter gegeben.

Besonders interessant ist hier der Gebrauch von funktionalen Abhängigkeiten, die durch Tabellen (= Stützpunkte der beschreibenden Funktionen) gegeben sind. Im Dokument finden wir diese Tabellen unter WITH LOOKUP.

Die Simulation läuft über 50 Jahre mit einem Zeitinkrement von 0,25 Jahren.

Beachtenswert ist hier der Einsatz der IF-Funktion, die in ihrer Syntax ganz ähnlich wie in den Computeralgebra-Systemen verwendet wird.



Das Dokument (im Original werden alle Größen alphabetisch aufgelistet) fasst die Konstanten, die Gleichungen und die Simulationszeitparameter zusammen:

(01) Aesungsverlust =
IF THEN ELSE(Futterbedarf>=(Futter/FRESSPERIODE) ,
Futter/FRESSPERIODE , Futterbedarf)

(02) Beuterate = WITH LOOKUP (Hirschdichte, ([(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)))

(03) Bewuchsdichte = Futter/MAX FUTTERKAPAZITAET

(04) FINAL TIME = 50

(05) FLAECHE = 320000

(06) FRESSPERIODE = 1

(07) Futter = INTEG (+Futterzuwachs – Aesungsverlust, START FUTTER)

(08) Futterangebot = Futter/Hirsche

(09) Futterbedarf = TAGESBEDARF * Hirsche

(10) Futterzuwachs = (MAX FUTTERKAPAZITAET – Futter)/Nachwachszeit

(11) Hirschdichte = Hirsche/FLAECHE

(12) Hirsche= INTEG (+ZuwachsHirsche – Verlust Hirsche, START HIRSCHE)

(13) INITIAL TIME = 0

(14) MAX FUTTERKAPAZITAET = 4.8e+008

(15) Nachwachszeit = WITH LOOKUP (Bewuchsdichte, ([(0,0)-(1,40)],(0,35),(0.25,15),(0.5,5),(0.75,1.5),(1,1)))

(16) RAEUBER = 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)))

(17) SAVEPER = TIME STEP

(18) START FUTTER = 4.7e+008

(19) START HIRSCHE = 4000

(20) TAGESBEDARF = 2000

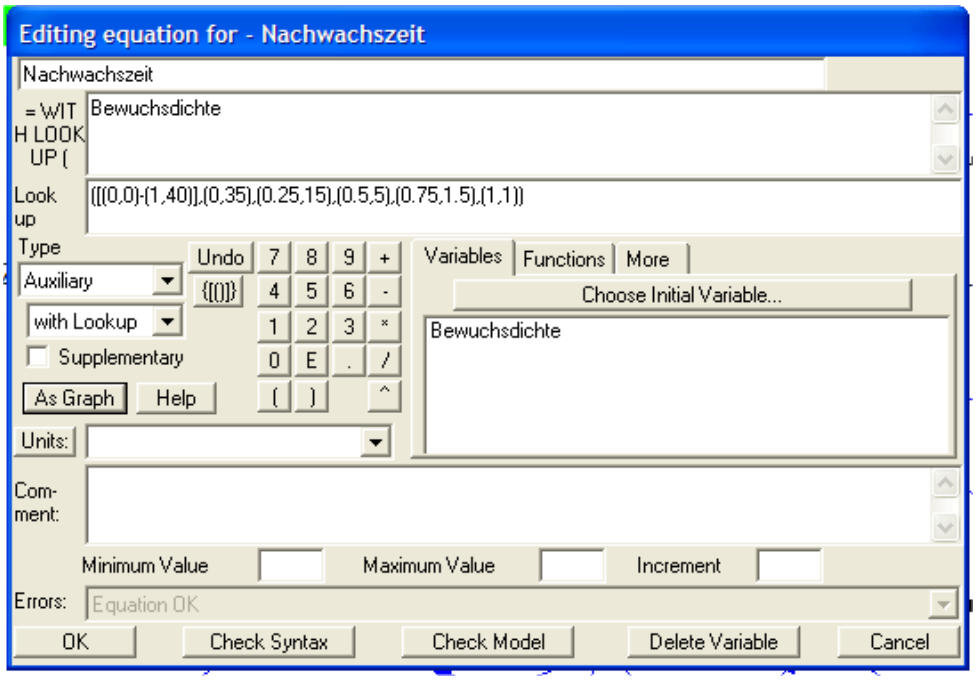
(21) TIME STEP = 0.25

(22) Verlust Hirsche = Beuterate * RAEUBER

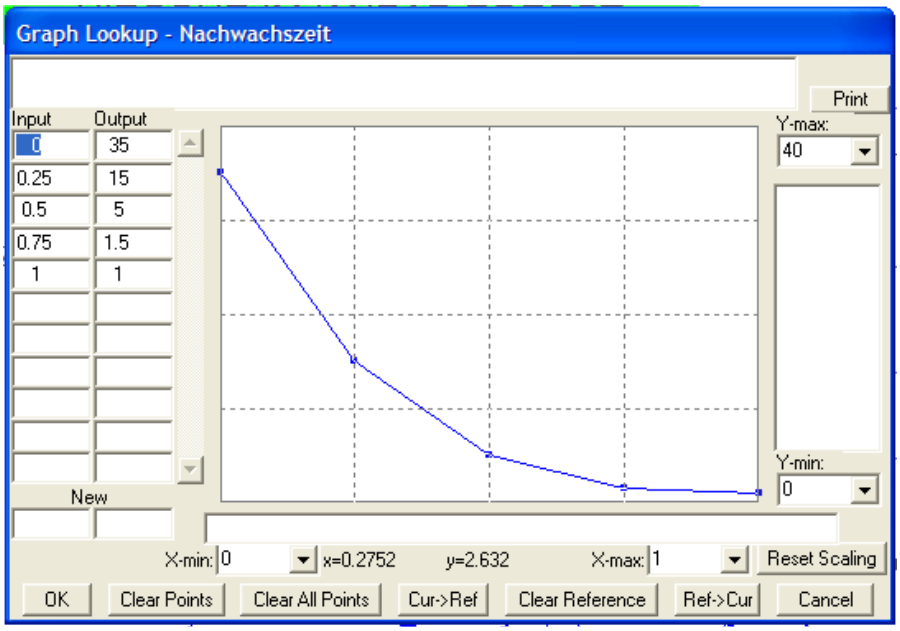
(23) ZuwachsHirsche = ZuwachsrateHirsche * Hirsche

(24) ZuwachsrateHirsche = WITH LOOKUP (Futterangebot, ([(0,-1)-(10000,1)],(0,-0.5),(500,-0.15),(1000,0),(1500,0.15),(2000,0.2),(200000,0.2)))

Betrachten wir als Beispiel die Funktion der (15) *Nachwachszeit*(*Bewuchsdichte*):



Wenn wir die Schaltfläche As Graph aktivieren, wird der Graph der Zuordnung gezeigt:



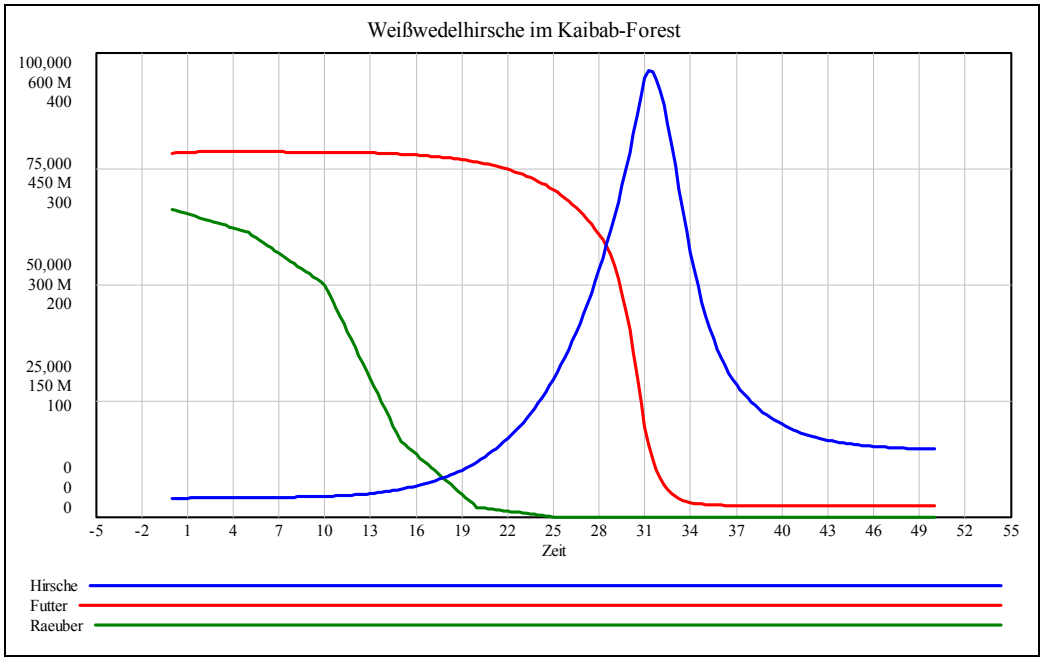
In diesem Graph ließen sich auch die Stützpunkte eingeben. Man erkennt, dass zwischen den Stützpunkten linear interpoliert wird.

Nun lassen wir die Simulation einmal laufen und betrachten die Ergebnisse.

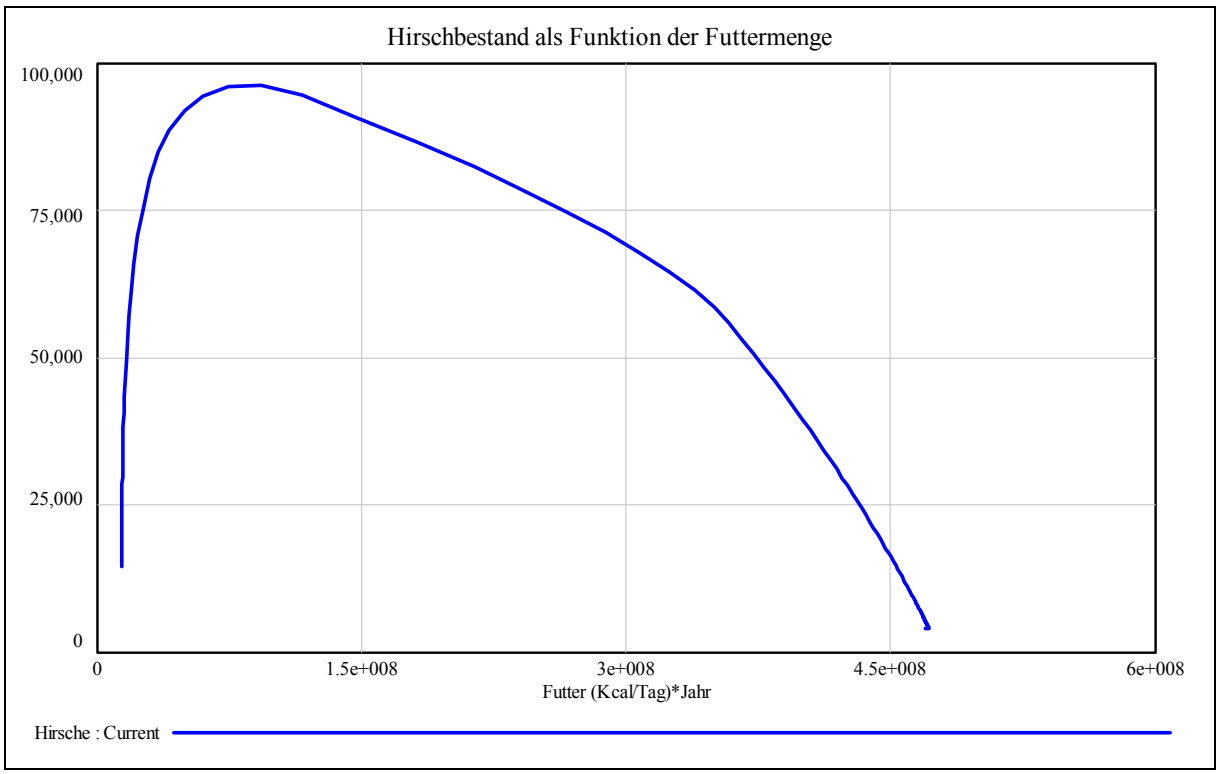
Wie ergeht es den Hirschen?

Wie entwickelt sich die verfügbare Futtermenge?

Wie ergeht es den Räubern? (Dank des menschlichen Eingreifens – schlecht!)



In einer zweiten Grafik zeigen wir die Hirschbestände als Funktion der Futtermengen:



Der Startpunkt ist rechts und die Entwicklung endet links.

Das Ergebnis der Simulation deckt sich weitgehend mit dem tatsächlichen Verlauf des Geschehens. Die Verminderung des Raubtierbestands führte zu einer explosionsartigen Vermehrung des Hirschbestands und dies weiterhin zu einer katastrophalen Überweidung der vorhandenen Futterkapazität. Eine große Anzahl der Hirsche verhungerte und schließlich stabilisierte sich der Wildbestand auf einem Niveau, das dem stark reduzierten Futterbestand entsprach.

Da ich – leider – kein *Excel*-Experte bin, weiß ich nicht, wie man auf einfache Weise die funktionalen Abhängigkeiten mit den damit verbundenen linearen Interpolationen in der Tabellenkalkulation umsetzen kann.

Es wäre schön, wenn ein Leser dieser Zeilen, das ergänzen könnte. Für eine entsprechende Nachricht wäre ich sehr dankbar.

Ich will aber später doch nochmals auf *Excel* zurückkommen.

Aber wir verfügen ja noch über weitere Werkzeuge!

Das Modell mit *DERIVE*

Dafür habe ich die Herausforderung angenommen, dieses System mit *DERIVE* zu behandeln.

Auch hier stellt sich das Problem der abschnittsweise definierten Funktionen und den damit verbundenen linearen Interpolationen.

Da hat man als DERIVIANER natürlich gleich die Idee, die in der Matrix gegebenen Punkte der Reihe nach mit Hilfe der CHI-Funktion durch Strecken zu verbinden.

Vorerst werden alle Modelldaten festgelegt. Es folgt mein erster Versuch, ein Äquivalent zur LOOKUP-Funktion zu programmieren.

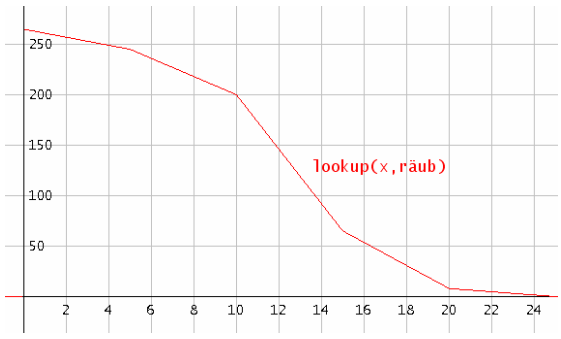
$$[\text{Fläche} := 320000,\ \text{Fressper} := 1,\ \text{Max\_Futter\_Kap} := 4.8\cdot 10^{8}]$$

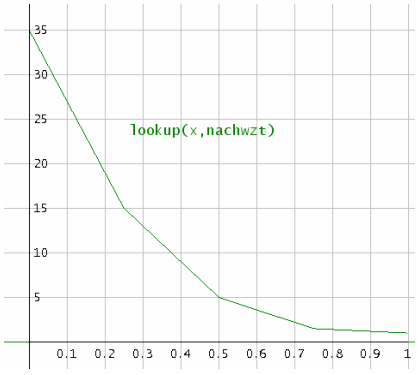
$$[\text{Tagesbed} := 2000,\ \text{Futter\_Start} := 4.7\cdot 10^{8},\ \text{Hirsch\_Start} := 4000]$$

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

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

Die Graphen sehen ja recht ordentlich aus. Ich zeige zwei Beispiele. Der Graph für die *Nachwachszeit* (Seite 25) lässt sich neben dem *VENSIM*-Graphen durchaus sehen.





Aber, das dicke Ende kommt noch. Wenn man die Funktionstabellen betrachtet (z.B. für die RÄUBER), erkennt man, das Defizit der CHI-Funktion in den Stützstellen. Dort ist die Funktion nicht definiert. Daher können wir diese Funktion hier nicht brauchen.

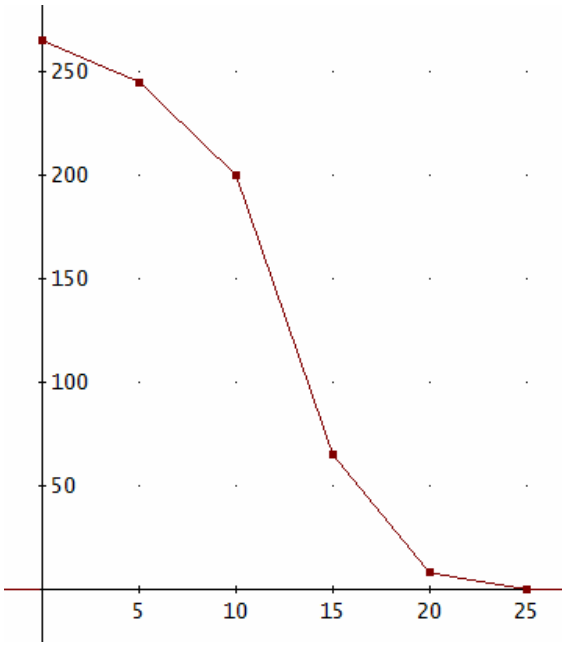
0	1	2	3	4	5	6
±132.5 + 132.5	261	257	253	249	?	236

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

Diese nächste Funktion genügt unseren Ansprüchen.

```
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_)
```

Die Graphen passen genau (Strecken und Stützpunkte) und die Tabellen zeigen keinerlei Besonderheiten, wie der Ausschnitt beweist.



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

0	1	2	3	4	5	6	7
265	261	257	253	249	245	236	227

26	27	28	29	30	31	32	33	34	35
0	0	0	0	0	0	0	0	0	0

Ich habe es immer als Vorteil empfunden, wenn ich vor dem Programmieren versuche, das System händisch – quasi als menschliche Tabellenkalkulation – zu bearbeiten. Ich würde dies auch dringend empfehlen, wenn man derartige Systeme im Unterricht behandeln will. Erst so wird die Vernetzung durchsichtig und dann die Programmierung oder der Eintrag in eine „richtige“ Tabellenkalkulation fast zum Kinderspiel. Dabei genieße ich jetzt den zusätzlichen Vorteil, dass ich die Resultate der *VENSIM*-Simulation in Form der Tabellen als Referenz verwenden kann.

Ich versuche, die händische Vorgangsweise – unterstützt durch die lu-Funktion in *DERIVE* – schrittweise zu demonstrieren.

Die Tabelle besteht aus 16 Spalten.

In die Zeile 1 kann ich für die Zeit 0 eintragen, für das FUTTER $4{,}7 \cdot 10^{8,}$ für die HIRSCHE 4000 und für die RÄUBER 265. Am Fuß der einzelnen Spalten trage ich die Reihenfolge der Berechnung ein.

Ich beginne mit dem *Futterbedarf*, setze fort mit dem *Äsungsverlust* und schließe die Zeile mit dem *Zuwachs der Hirsche*. Anschließend wird in Zeile 2 (Zeit = 0,25) die neue Futtermenge und die neue Hirschpopulation (13, 14) eingetragen und es geht wieder weiter von 1 bis 12.

Wir folgen den Formeln (Gleichungen), wie sie im Dokument aufgelistet sind.

ZeilenNr	Zeit	Futterbed.	Äsverlust	Bewuchsd	Nachwzeit	Futterzuw	FUTTER
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

RÄUBER	Hirschd	Beuterate	Verlust H	Futterang	ZuwRateH	ZuwachsH	HIRSCHE
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

$^{(*)}$ Bei den Zuwächsen (für FUTTER und HIRSCHE) ist zu beachten, dass der Zeitzuwachs *dx* zu berücksichtigen ist. Daher z.B. 4000 + (800 – 795) · 0,25 = 4001,25.

Die Werte in den Spalten für die *Nachwachszeit*, *Beute*- und *Zuwachsrate der Hirsche* wurden mit Hilfe der lu-Funktion ermittelt (analog zu WITH LOOKUP).

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

```
lu(0.25, raeub) = 264
```

Die Umsetzung in die Tabelle lässt sich 1:1 in Programmzeilen eines *DERIVE*-Programms übertragen.

Ich lasse auch in *DERIVE* alle Werte in einer Tabelle für die Ausgabe zusammenfassen. Für die Diagramme selektiere ich dann die entsprechenden Spalten.

Zuerst muss noch für den *Äsungsverlust* eine kleine Funktion definiert werden.

```
aesverl(x, y) :=
  If x ≥ y/Fressper
    y/Fressper
    x
```

Die lu-Funktion wurde bereits vorgestellt. Nun folgt das Programm:

```
kaibab(n, dx, i, tab, t, f_bed, äs_verl, bew_d, hirsch_d, räuber,
       nachw_zeit, futter_zuw, beute_rate, hirsche_verl, futter_ang,
       hirsche_zuw, hirsche, futter) :=
PROG(
   i := 1,
   tab := [["ZNr", "Zeit", "Fbed", "Äsverl", "BewD", "NachwZt", "FZuw",
        "BeuteR", "HirschV", "FAng", "HirschZ", "FutterM", "Hirsche",
        "Räuber"]],
   [t:=0, hirsche := Hirsch_Start, futter := Futter_Start],
   LOOP(IF(i > n,
           RETURN tab),
        f_bed := hirsche·Tagesbed,
        äs_verl := äsverl(f_bed, futter),
        bew_d := futter/Max_Futter_Kap,
        hirsch_d := hirsche/Fläche,
        räuber := lu(t, räub),
        nachw_zeit := lu(bew_d, nachwzt),
        futter_zuw := (Max_Futter_Kap - futter)/nachw_zeit,
        beute_rate := lu(hirsch_d, beuterate),
        hirsche_verl := beute_rate·räuber,
        futter_ang := futter/hirsche,
        hirsche_zuw := lu(futter_ang, zuw_r_h)·hirsche,
        tab := APPEND(tab, [[i, t, f_bed, äs_verl, bew_d, nachw_zeit,
          futter_zuw, beute_rate, hirsche_verl, futter_ang, hirsche_zuw,
          futter, hirsche, räuber]]),
          hirsche := hirsche + (hirsche_zuw - hirsche_verl)·dx,
          futter := futter + (futter_zuw - äs_verl)·dx,
        t :+ dx, i :+ 1))
```

Der Ausdruck der ersten 4 Zeilen zeigt – erfreulicherweise – die vollständige Übereinstimmung mit der händisch ermittelten Tabelle und mit den *VENSIM*-Ergebnissen.

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

ZNr	Zeit	Fbed	Äsverl	BewD	NachwZt	FZuw	BeuteR
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

HirschV	FAng	HirschZ	FutterM	Hirsche	Räuber
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

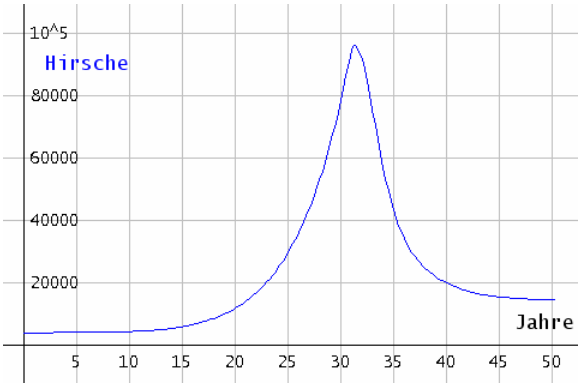
Und dies sind die Werte für FUTTER und HIRSCHE für die drei letzten Vierteljahre:

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

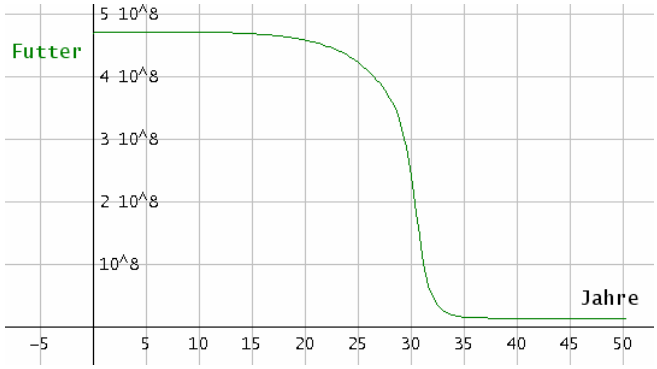
49.5	14277004.0601	14566.1776742
49.75	14277004.0192	14544.4896531
50	14277003.988	14524.4282306

Nun folgen die Diagramme, die mit den *VENSIM*-Diagrammen auf den Seiten 25 und 26 verglichen werden können.

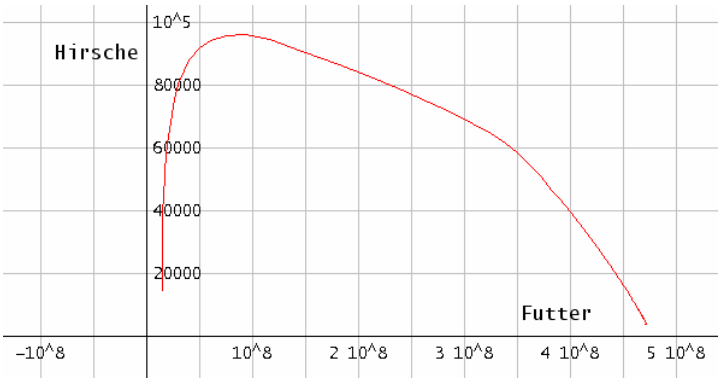
```
DELETE((kaibab(202, 0.25))↓↓[2, 13], 1)
```




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




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



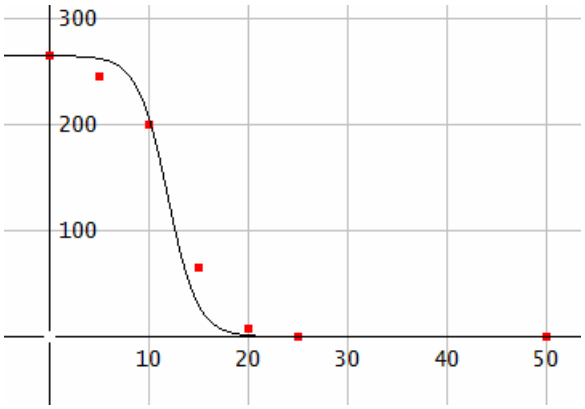
Vielleicht versuche ich später noch die Umsetzung mit Hilfe von Differentialgleichungen!

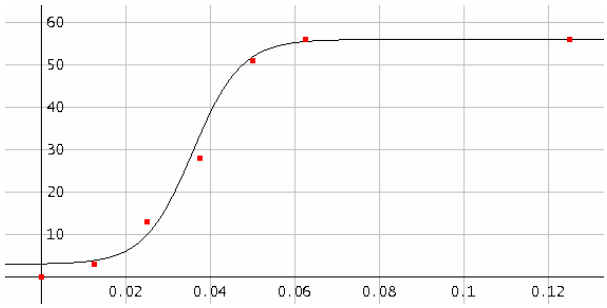
Für die Ausarbeitung mit Hilfe einer Tabellenkalkulation wäre es nützlich, die „Tabellenfunktionen“ für die RÄUBER, *Beuterate*, *Nachwachszeit* und *Hirschzuwachsrate* durch eine „gewöhnliche“ Funktion zu approximieren.

Dies ist für sich alleine schon eine „nette“ Aufgabe. Schieberegler und sinnvolle Überlegungen führen zu geeigneten Funktionen:

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

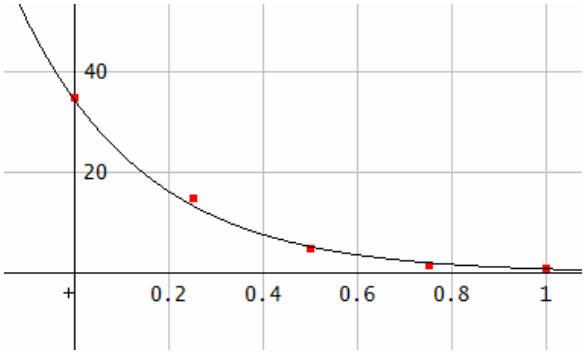
$$beuter\_f(x) := 3 + \frac{53}{1 + 529 \cdot e^{-174.9 \cdot x}}$$

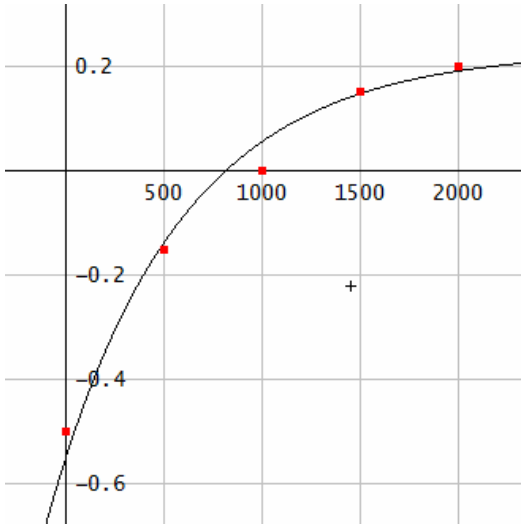




$$nachwzt\_f(x) := 34.4089 \cdot e^{-3.7653 \cdot x}$$

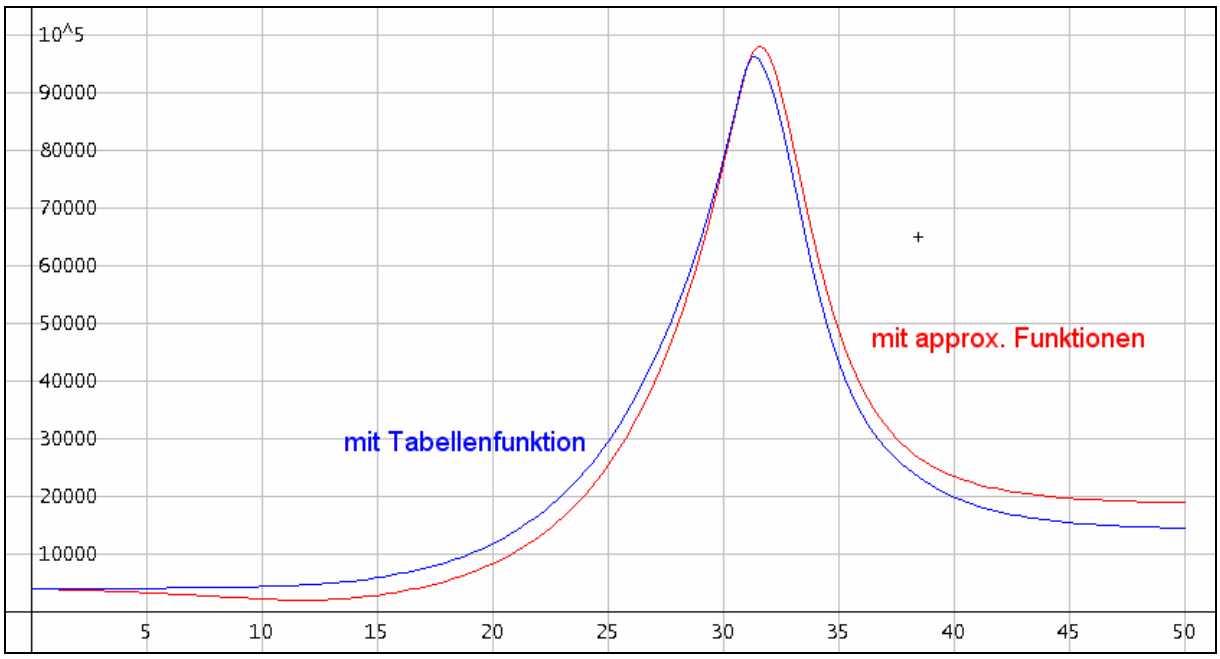
$$zuwrate\_f(x) := 0.23 - 0.78 \cdot e^{-0.0015 \cdot x}$$





Wie gut diese Approximationen sind, zeigt sich erst, wenn man die lu-Funktionen im Programm kaibab durch diese Funktionen ersetzt und damit das Programm kaibab_f gewinnt.

In der nächsten Grafik kann man die Entwicklung der Hirschpopulation mit beiden Modellen vergleichen.

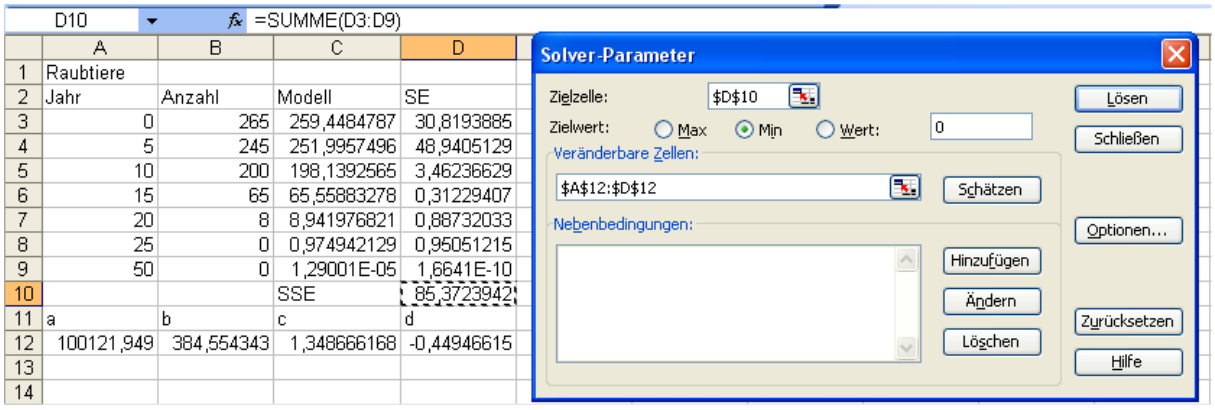


Das Ergebnis ist beeindruckend. Die Grafiken für die Futtermengen sind ebenfalls fast identisch.

Diese Funktionen machen eine Modellierung mit einer Tabellenkalkulation natürlich wesentlich einfacher und damit kann ich das nun auch mit *MS-Excel* versuchen.

Das Modell mit *MS-Excel*

Wir könnten die Funktionen von *DERIVE* übernehmen, aber mit dem SOLVER steht uns auch in der Tabellenkalkulation ein sehr brauchbares Werkzeug zu Verfügung. Wir müssen uns „nur“ von der Gestalt des Streudiagramms inspirieren lassen, für welchen Funktionstyp wir uns entscheiden wollen.



	A	B	C	D
1	Raubtiere			
2	Jahr	Anzahl	Modell	SE
3	0	265	259,4484787	30,8193885
4	5	245	251,9957496	48,9405129
5	10	200	198,1392565	3,46236629
6	15	65	65,55883278	0,31229407
7	20	8	8,941976821	0,88732033
8	25	0	0,974942129	0,95051215
9	50	0	1,29001E-05	1,6641E-10
10			SSE	85,3723942
11	a	b	c	d
12	100121,949	384,554343	1,348666168	-0,44946615
13				
14				

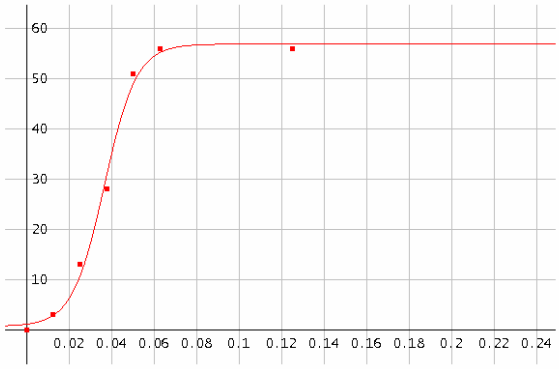
„Inspiriert“ von *DERIVE* wähle ich für die Räuberfunktion die Form $\frac{a}{b+c\cdot e^{-d\cdot x}}$ und editiere die Zelle C3 wie folgt: =$A$12/($B$12+$C$12*EXP(-$D$12*A3)).

In die Zellen A12 bis D12 gebe ich geeignete Startwerte ein – und das ist sicherlich der heiklere Teil der Aufgabe. Aber auch hier kann ich auf frühere Erkenntnisse zurückgreifen. Am schwierigsten gestaltet sich die Suche nach der Funktion für die Zuwachsrate der Hirsche.

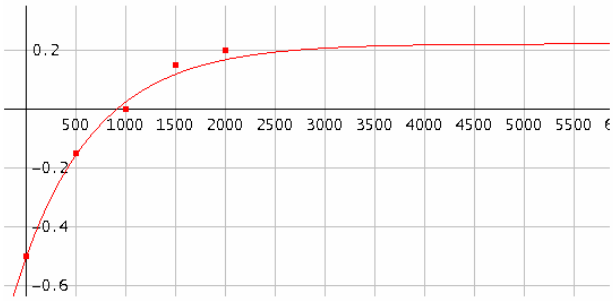
In der SE-Spalte finden sich die quadrierten Abweichungen der Modellwerte von den realen Werten (SE = *squared error*). In Zelle D10 steht die Summe dieser Werte, die minimiert werden soll.

Nun stellt sich heraus, dass mit dem SOLVER doch deutlich bessere Näherungsfunktionen gefunden werden können, wie schon allein die grafische Gegenüberstellung zeigt. Vergleiche mit den Funktionsgraphen auf Seite 32!

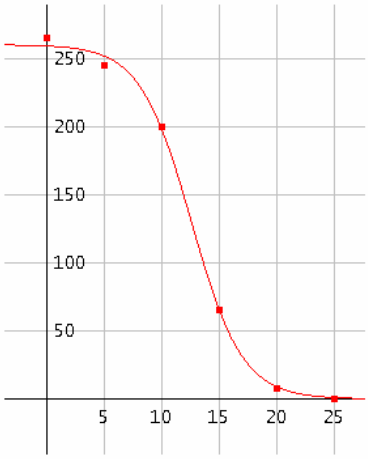
$$0.733 + \frac{317.477}{5.654 + 731.755 \cdot e^{-132.95 \cdot x}}$$



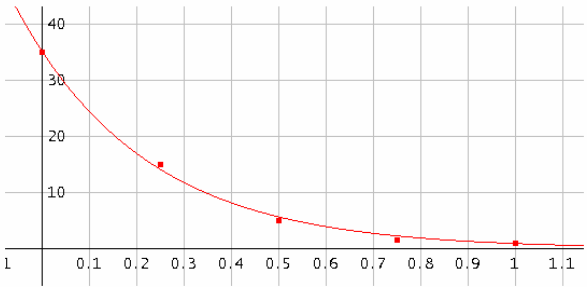
$$0.222 - 0.726 \cdot e^{-0.0013 \cdot x}$$



$$\frac{100122}{384.55 + 1.3487 \cdot e^{0.4495 \cdot x}}$$



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

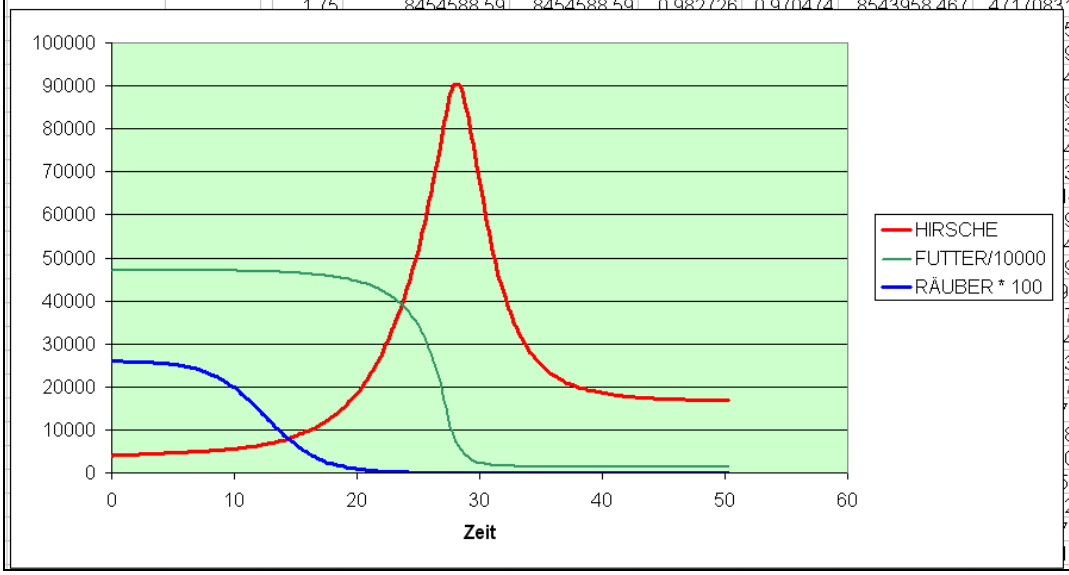


Mit diesen Funktionen wird die Exceltabelle nach dem Muster von Seite 29 erstellt. In Excel kann man es sich leisten mit einem Zeitintervall von 0,25 Jahren zu rechnen. Die Rechnung erfolgt sehr schnell.

Durch die etwas andere Räuberfunktion verschiebt sich die Spitze der Hirschpopulation, aber die Charakteristik der Aussagen bleibt völlig erhalten.

Ein Ausschnitt der Tabelle und die Grafik sind auf der nächsten Seite abgebildet.

Parameter:									
		Zeit	FutterBed	Äsungsverl	BewuchsD	NachwZt	FutterZuw	FUTTER	RÄUBER
Fläche	320000	0	8000000	8000000	0,979167	0,983177	10171106,17	470000000	259,44848
Fressperiode	1	0,25	8064887,293	8064887,293	0,980297	0,979123	9658870,723	470542776,5	259,34069
MaxFKap	4,8E+08	0,5	8129828,993	8129828,993	0,981128	0,976157	9279987,698	470941272,4	259,22019
HirschStart	4000	0,75	8194800,688	8194800,688	0,981727	0,974023	9005115,196	471228812,1	259,0855
FutterStart	4,7E+08	1	8259780,262	8259780,262	0,982149	0,972522	8810711,041	471431390,7	258,93495
Tagesbedarf	2000	1,25	8324748,316	8324748,316	0,982436	0,971503	8678181,298	471569123,4	258,76671
dx	0,25	1,5	8389688,618	8389688,618	0,98262	0,970849	8593009,36	471657481,6	258,57871
		1,75	8454588,59	8454588,59	0,982726	0,970474	8543958,467	471708311,8	258,36869
								54,3	258,13409
								90,8	257,8721
								49,7	257,57958
								97,4	257,25305
								38,2	256,88868
								44,8	256,48219
								32,4	256,02889
								1492	255,52358
								91,2	254,96053
								49,6	254,33346
								96,8	253,63546
								9615	252,85898
								71,4	251,99577
								42,2	251,03687
								30,6	249,97255
								79,7	248,79231
								7282	247,48485
								86,1	246,0381
								01,2	244,43922
								5700	242,67465
								20,6	240,73019
								7967	238,59111
								1509	236,24226



Bossel stellt in den Arbeitsvorschlägen die interessante Frage:

> *Wie hätte die Abschussstrategie (für die Raubtiere) aussehen müssen, um einen hohen stabilen Hirschbestand ohne Zusammenbruch der Weidekapazität zu erhalten?*

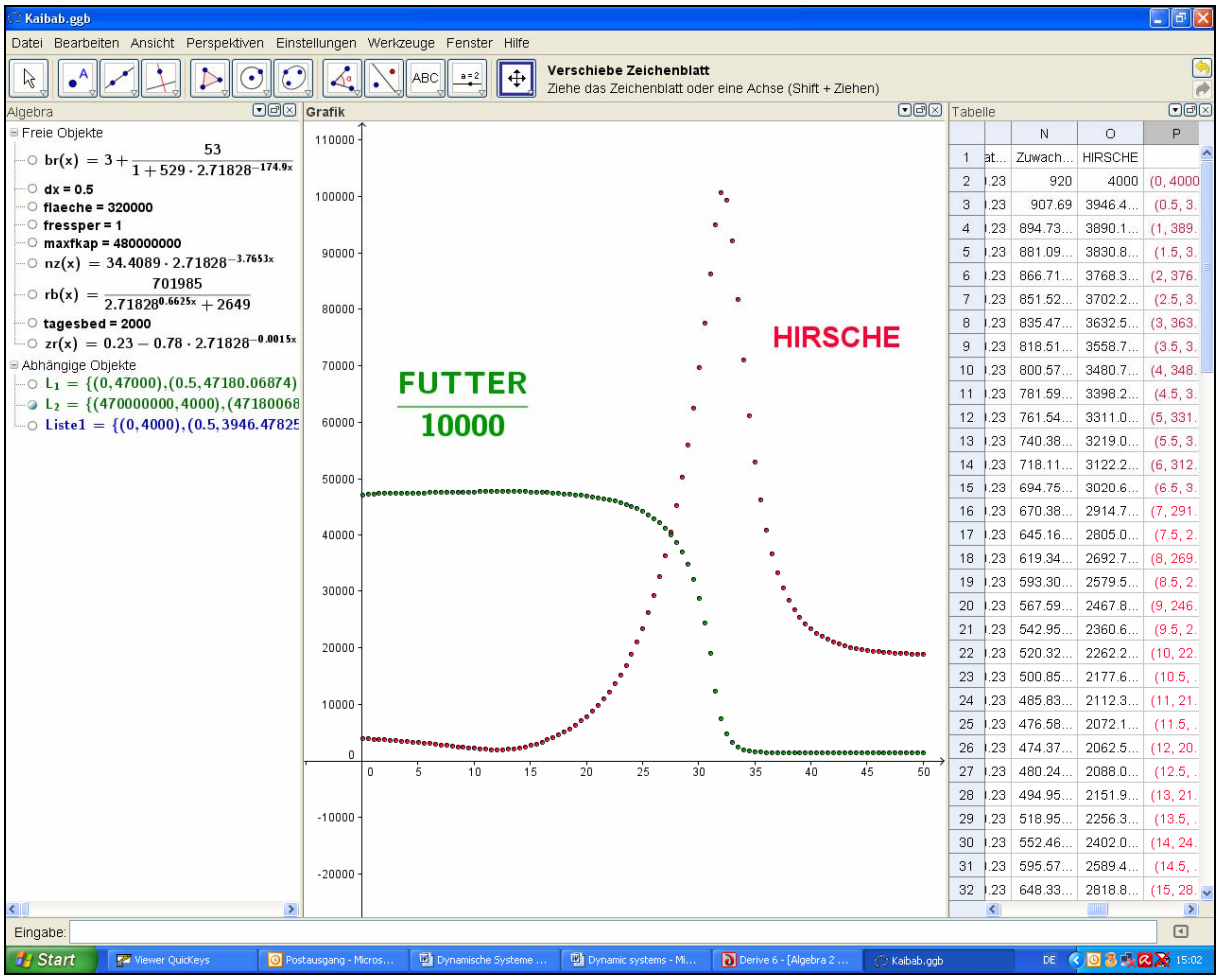
Für die Beantwortung von Fragen dieser Art ist der Einsatz von Schiebereglern ideal. Sowohl *GeoGebra* als auch *TI-Nspire* stellen dieses Werkzeug zur Verfügung.

Das Modell mit *GeoGebra*

Es werden die mit *DERIVE* ermittelten Näherungsfunktionen eingesetzt. Das erste Modell wird mit der vorgegebenen Räuberfunktion erstellt – um zu testen, ob die Ergebnisse den Erwartungen entsprechen.

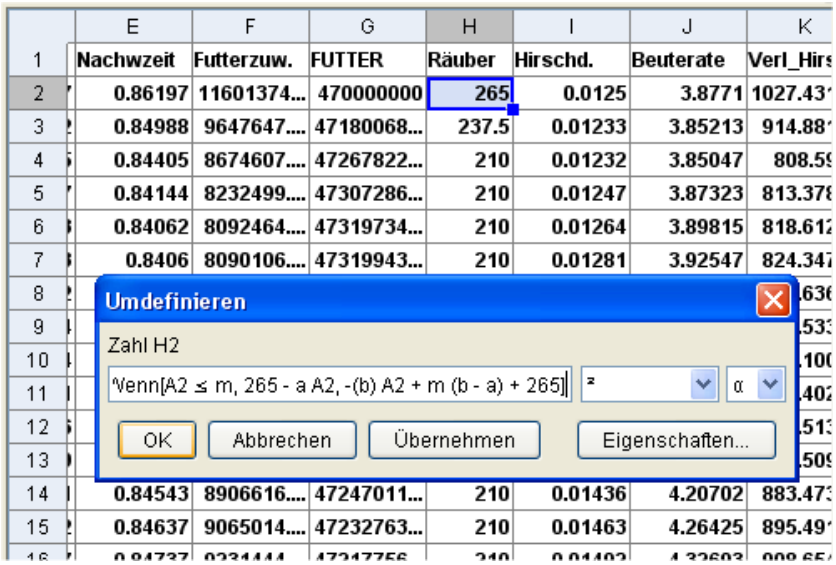
Der nächste Bildschirm ist eine Kopie des *GeoGebra*-Schirms mit der Grafik des Hirschbestands und der umskalierten Futtermenge (Futtermenge / 10 000).

Da in der *GeoGebra*-Tabellenkalkulation längere Rechenzeiten entstehen können, wurde der Zeitschritt auf 0,5 erhöht, was aber keine wesentliche Änderung in der Aussage nach sich zieht, wie die Grafik zeigt.

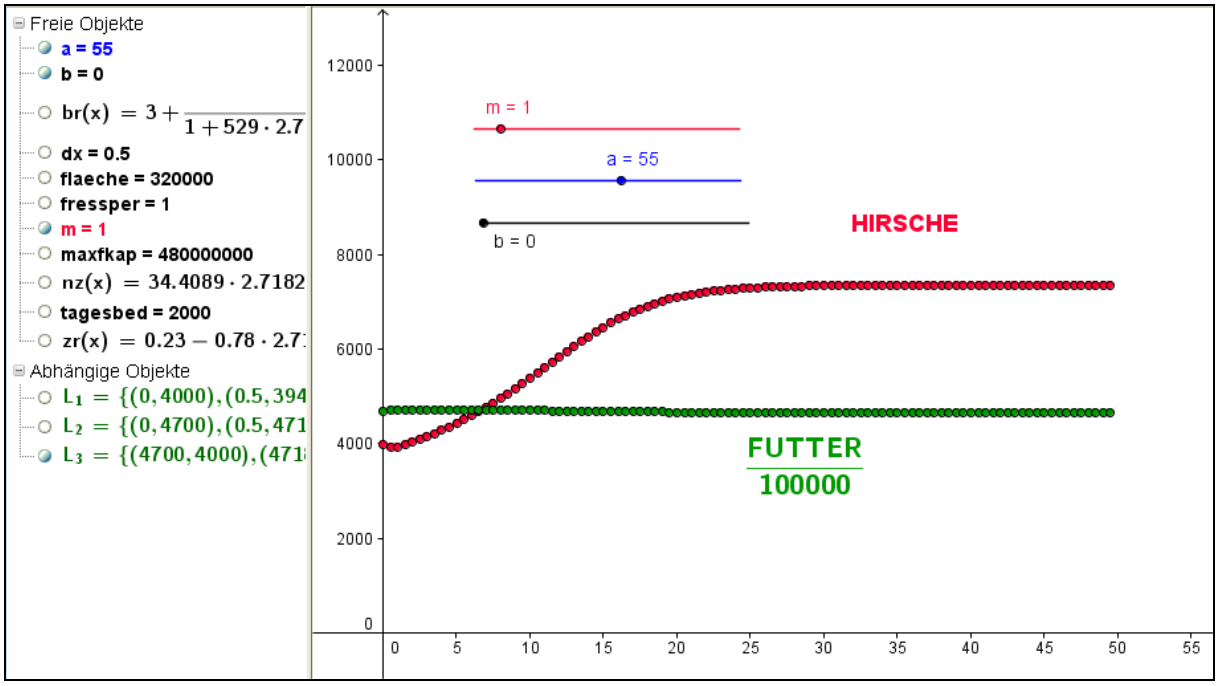


Ich habe nun versucht, eine Antwort auf die oben gegebene Aufgabenstellung zu finden. Nach einigen – spannenden – Versuchen, habe ich die folgende Abschussstrategie überlegt:

In den ersten *m* Jahren erfolgt ein radikaler Abschuss von *a* Tieren pro Jahr, anschließend verringern wir den Abschuss auf *b* Tiere pro Jahr. Die entsprechende „Räuberfunktion“ steht in Zelle H2, während in A2 die jeweilige Zeit zu finden ist. Sonst muss sonst in der Tabelle von vorhin nichts geändert werden.



Die Berechnung der ersten kompletten Tabelle dauert etwas, dafür reagiert die Grafik auf die Veränderung der Parameter durch die Schieberregler sehr rasch.
Bei einem anfänglichen Abschuss von 55 Raubtieren im ersten Jahr kann offensichtlich anschließend der Bestand auf 210 gehalten werden. Die Futtermenge (hier skaliert als FUTTER/100 000) bleibt konstant und die Anzahl der Hirsche pendelt sich bei 7360 ein.



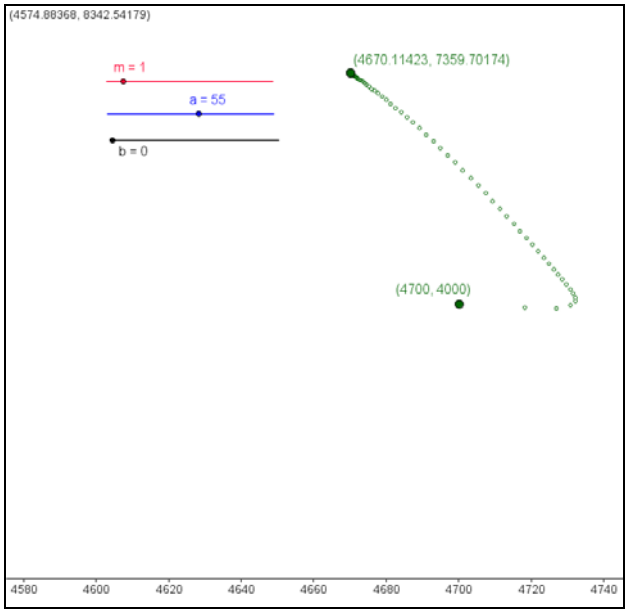
Das Phasendiagramm Futter-Hirsche zeigt auch eine deutliche Konvergenz.

Das ist aber nicht die einzige Möglichkeit, das Ziel einer stabilen hohen Hirschpopulation zu erreichen.

Man kann jetzt richtig probieren auf einen halbwegs konstanten Hirschbestand auf einem niedrigeren oder höheren Niveau zu kommen.

Das lässt sich auch mit einem moderaten Abschuss des Raubzeugs erreichen. Neben dieser „exogenen“ Regulierung der Räuber kann man natürlich auch „endogene“ Faktoren wie natürliche Verlustrate usw. berücksichtigen und in die Simulation einbeziehen.

Waidmanns Heil!



Die Ausarbeitung mit *TI-NspireCAS*

Mit der Tabellenkalkulation hatte ich zuerst erhebliche Schwierigkeiten. Es hat aber dann doch sehr ordentlich funktioniert. Die Datei kann von mir bezogen werden. Die Grafik sieht genau so aus wie das *GeoGebra*-Programm. Die Berechnung der Tabelle läuft allerdings wesentlich rascher und daher sind auch kleinere Zeitinkremente möglich.

Die Übertragung des *DERIVE*-Programms in die TI-NspireCAS-Sprache ist leicht gelungen.

Dabei ist luf(x,pk) die Tabellenfunktion, die der 1u-Funktion in *DERIVE* entspricht:

```
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
```

Es folgt das Programm, das die entsprechenden Listen, die zur grafischen Darstellung nötig sind, erzeugt.

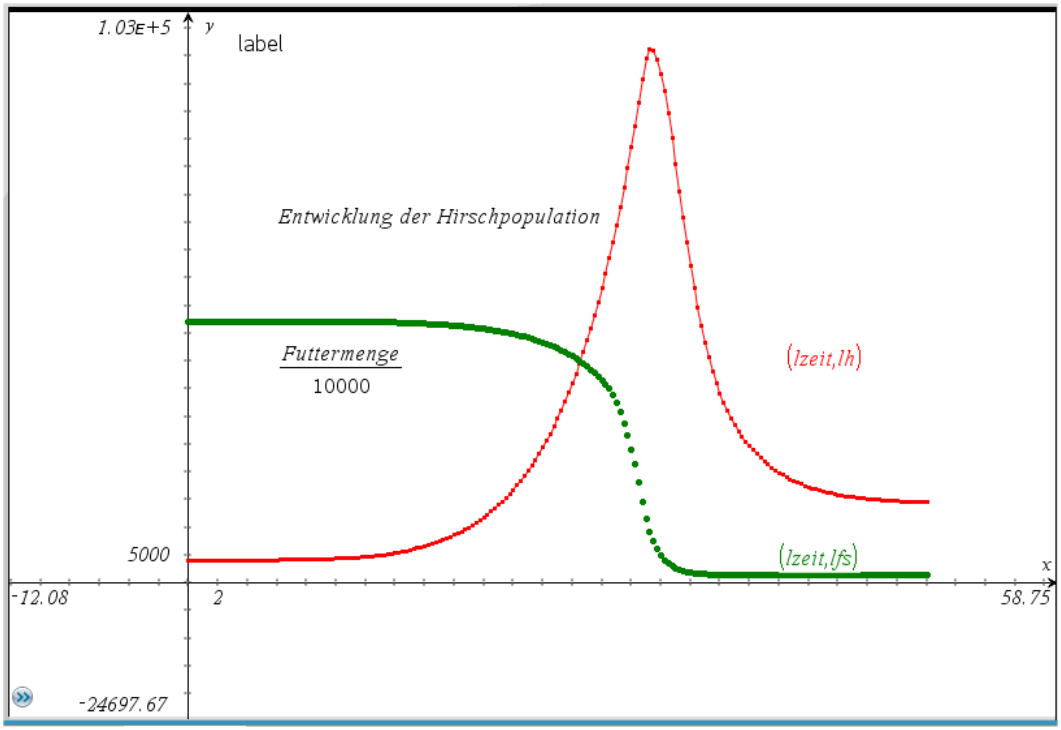
```
Define kaibab(n,dx)=
Prgm
:Local i,t,hirsche,futter,f_bed,aes_verl
:Local bew_d,hirsch_d,raeuber
:Local nachw_zeit,futter_zuw,beute_rate
:Local hirsche_verl,futter_ang,hirsche_zuwr
:Local hirsche_zuw
:i:=1: t:=0
:hirsche:=hirsch_start:futter:=futter_start
:lh:={hirsche}:lf:={futter}:lzeit:={t}
:While i≤n
:  f_bed:=hirsche*tagesbed
:  aes_verl:=when(f_bed≥((futter)/(fressper)),((futter)/(fressper)),f_bed)
:  bew_d:=((futter)/(max_futter_kap))
```

```
:  hirsch_d:=((hirsche)/(flaeche))*1.
:  raeuber:=luf(t,raeub_lu)
:  nachw_zeit:=luf(bew_d,nachwachszeit_lu)
:  futter_zuw:=((max_futter_kap−futter)/(nachw_zeit))
:  beute_rate:=luf(hirsch_d,beuterate_lu)
:  hirsche_verl:=beute_rate*raeuber
:   futter_ang:=((futter)/(hirsche))
:  hirsche_zuwr:=luf(futter_ang,zuwachsrate_h_lu)
:  hirsche_zuw:=hirsche*hirsche_zuwr
:  hirsche:=hirsche+(hirsche_zuw−hirsche_verl)*dx
:  futter:=futter+(futter_zuw−aes_verl)*dx
:  t:=t+dx
:  lh:=augment(lh,{hirsche})
:  lf:=augment(lf,{futter})
:  lzeit:=augment(lzeit,{t})
:  i:=i+1
:EndWhile
:Disp "Hirsche in lh, Futter in lf, Zeit in lzeit"
:EndPrgm
```

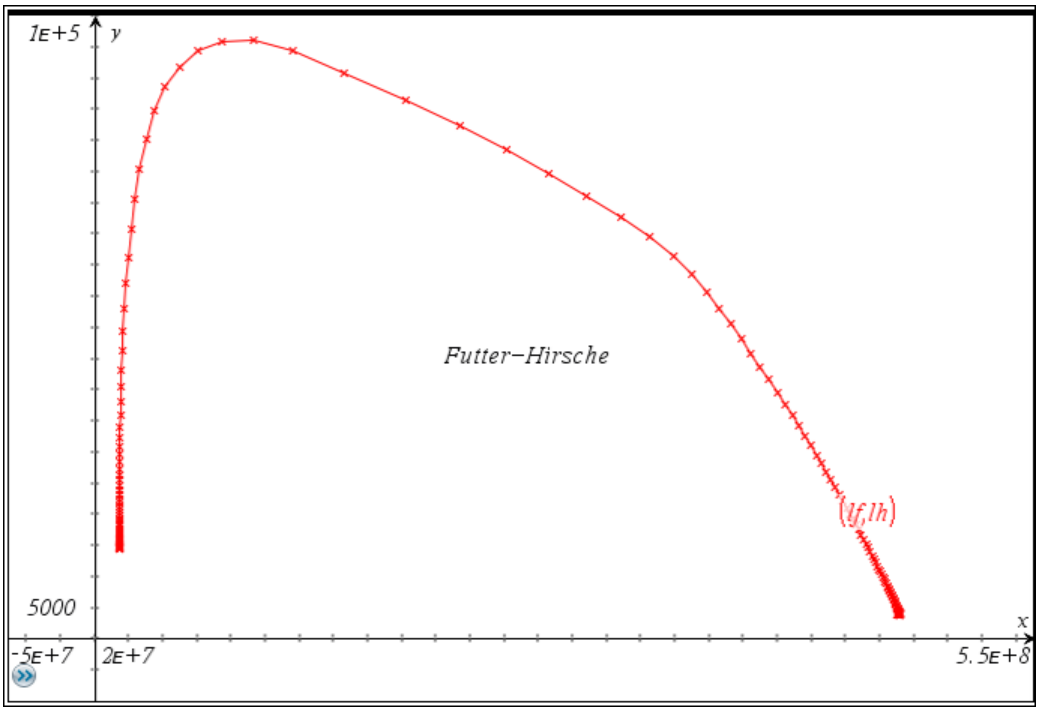
Das Calculator-Fenster enthält die Angaben und den Aufruf des Programms.

$nachwachszeit\_lu:=\begin{bmatrix}0 & 35\\ 0.25 & 15\\ 0.5 & 5\\ 0.75 & 1.5\\ 1 & 1\end{bmatrix}$	$\begin{bmatrix}0.000000 & 35.000000\\ 0.250000 & 15.000000\\ 0.500000 & 5.000000\\ 0.750000 & 1.500000\\ 1.000000 & 1.000000\end{bmatrix}$
$zuwachsrate\_h\_lu:=\begin{bmatrix}0 & -0.5\\ 500 & -0.15\\ 1000 & 0\\ 1500 & 0.15\\ 2000 & 0.2\\ 200000 & 0.2\end{bmatrix}$	$\begin{bmatrix}0.000000 & -0.500000\\ 500.000000 & -0.150000\\ 1000.000000 & 0.000000\\ 1500.000000 & 0.150000\\ 2000.000000 & 0.200000\\ 200000.000000 & 0.200000\end{bmatrix}$
$flaeche:=320000:fressper:=1:max\_futter\_kap:=4.8\cdot10^{8}$	480000000.000
$tagesbed:=2000:futter\_start:=4.7\cdot10^{8}:hirsch\_start:=4000$	4000.000000
$kaibab(200,0.25)$	
	Hirsche in lh, skal. Futter in lfs, Zeit in lzeit
	Done

Die Listen lzeit, lh und lfs bilden die Grundlage der Streudiagramme in der Graph-Applikation. Die nächste Abbildung zeigt die bereits bekannte Entwicklung der Hirschpopulation gemeinsam mit der Darstellung einer passend skalierten Futtermenge.



Auch das Phasendiagramm haben wir in dieser Form schon kennen gelernt.

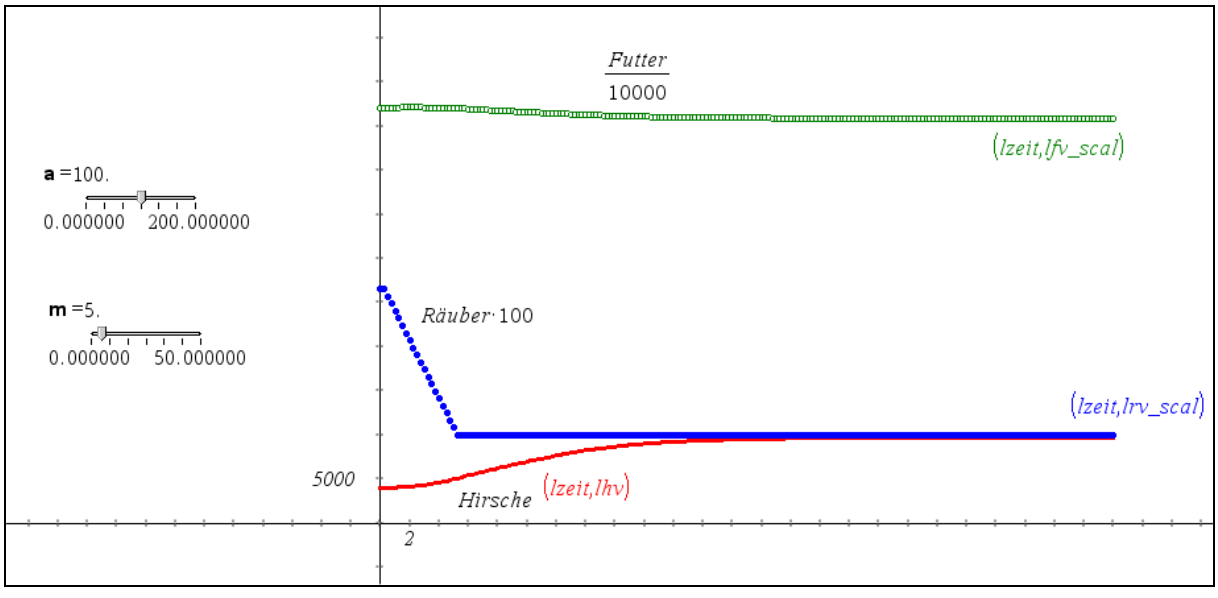


Jetzt will ich aber auch noch mit diesem Werkzeug den Schieberegler zu Ehren kommen lassen.

Ich beschreibe die Abschussstrategie etwas anders ein als vorhin. Für die ersten *m* Jahre sollen die Abschusszahlen konstant bleiben bis der Bestand an Raubzeug den vorgegebenen Wert *a* erreicht hat. Dieser Wert soll dann gehalten werden. Für *m* und *a* werden Schieberegler eingeführt.

```
"kaibab_var" stored successfully
lrv:={rb_start}
lzeit:={t}
While i≤n
 f_bed:=hirsche·tagesbed
 aes_verl:=when(f_bed≥futter/fressper, futter/fressper, f_bed)
 bew_d:=futter/max_futter_kap
 hirsch_d:=hirsche/flaeche·1.
 raeuber:=when(x≤m, (a−265)/m·x+265, a)|x=t
 nachw_zeit:=luf(bew_d,nachwachszeit_lu)
 futter_zuw:=(max_futter_kap−futter)/nachw_zeit
 beute_rate:=luf(hirsch_d,beuterate_lu)
 hirsche_verl:=beute_rate·raeuber
```

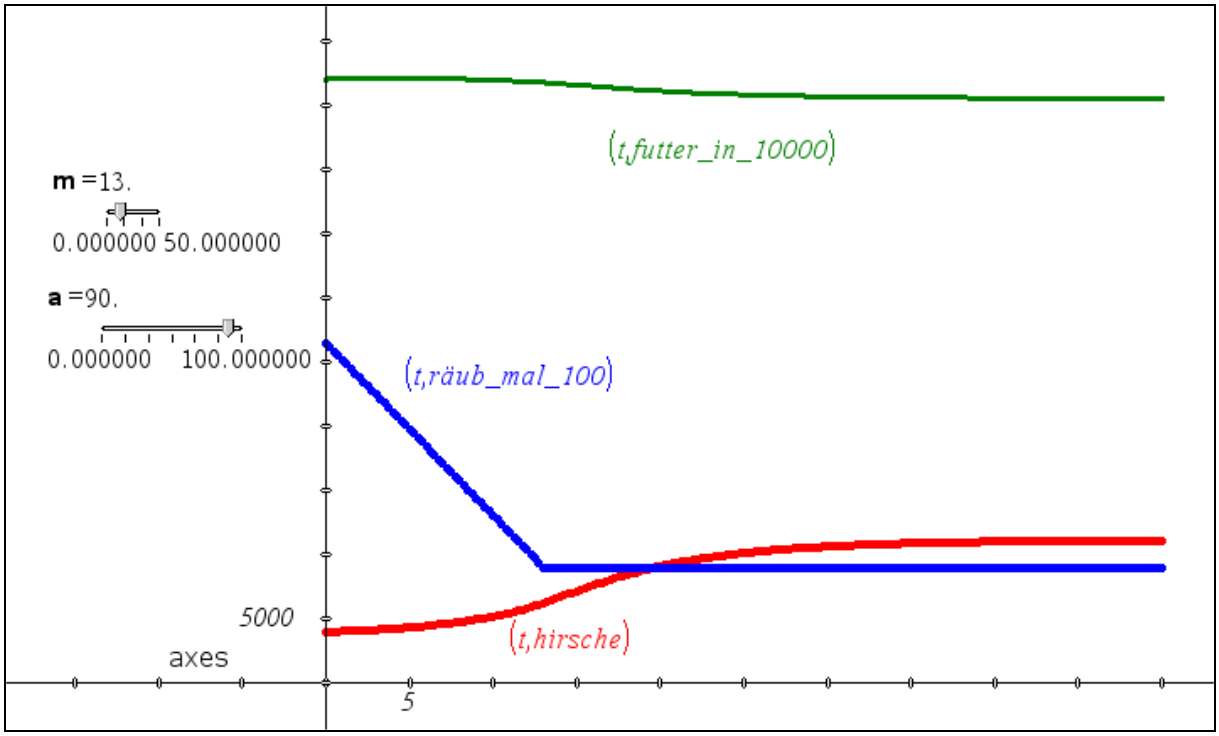
Die Definition ist im Programm unter *raeuber*:= zu sehen.



Die Arbeit mit Schiebereglern über das Programm hat den Nachteil, dass nach jeder Änderung der Parameter durch die Schieberegler das Programm neuerlich aufgerufen werden muss, um die Listen zu aktualisieren. Die Grafik passt sich dann allerdings augenblicklich an.

Wenn man aber auch hier die Tabellenkalkulation einsetzt, lässt sich die Grafik sofort über die Schieberegler manipulieren. (Die Tabellenkalkulation ist hier ausgeblendet.)

Hier wollen wir innerhalb der ersten 13 Jahre den Bestand der Räuber gleichmäßig auf 90 herabsetzen und diesen Stand dann halten. Man sieht, dass damit das von *Bossel* angestrebte Ziel erreicht werden kann.



Ich hatte schon vorher die Absicht erklärt, möglicherweise auch den Zugang über die numerische Lösung des entsprechenden Differentialgleichungssystems zu versuchen. Es ist gelungen, wie man gleich sehen kann.

Die ökologische Katastrophe als Differentialgleichungssystem

Die Formulierung der beiden Differentialgleichungen ergibt sich direkt aus den *VENSIM*-Gleichungen:

$$\frac{dh}{dt} = h \cdot zuwrate\_f\left(\frac{f}{h}\right) - beuter\_f\left(\frac{h}{Flaeche}\right) \cdot raeuber\_f(t)$$

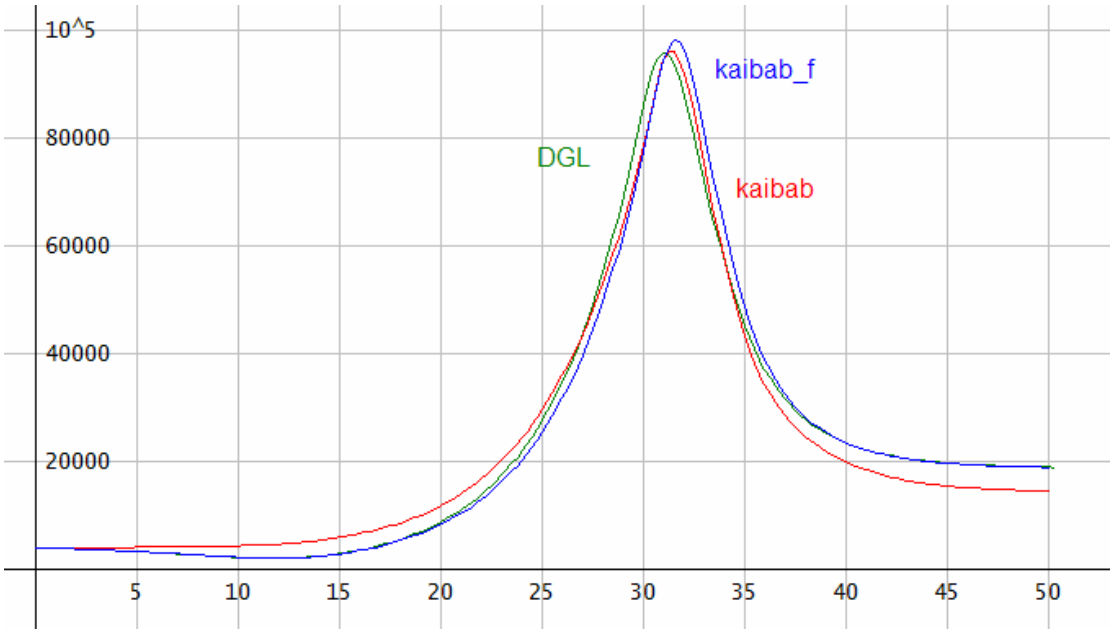
$$\frac{df}{dt} = \frac{Max\_Futter\_Kap - f}{nachwzt\_f\left(\frac{f}{Max\_Futter\_Kap}\right) - aesverl(Tagesbed \cdot h)}$$

Ich verwende wieder die Runge-Kutta-Routine von *DERIVE*.

```
(RK([[zuwrate_f(f/h)·h - beuter_f(h/Flaeche)·raeuber_f(t), (Max_Futter_Kap - f)/nachwzt_f(f/Max_Futter_Kap) -
  aesverl(Tagesbed·h, f)], [t, h, f], [0, Hirsch_Start, Futter_Start], 0.25, 201])↓↓[1, 2]
```

Es lassen sich auch die WITH LOOKUP-Routinen verwenden, allerdings bewältigt RK dann nicht alle 201 Zeilen der Tabelle am Stück.

Die Auswahl der 1. und 2. Spalte zeigt die Entwicklung der Hirschpopulation:



Hier sind alle Ergebnisse gegenüber gestellt.

Dieses Modell hat mich richtig fasziniert, da es so viele Möglichkeiten der Behandlung bietet. Die Beschreibung der abschnittsweise definierten Funktionen durch jeweils eine Funktion verlangen einige Phantasie und Kenntnisse über mögliche Funktionsformen. „Die“ richtige Form gibt es nicht, und das lässt sich ja von den meisten Modellierungsaufgaben behaupten.

Der Vergleich zwischen Programm und Tabellenkalkulation ist reizvoll und herausfordernd zugleich.

Der Gebrauch von Schiebereglern bringt noch eine wichtige zusätzliche Qualität.

Neben dem mathematischen Aspekt macht dieses Modell einmal mehr deutlich wie ein Eingriff in die Natur das Gleichgewicht zerstören und nicht vorherzusehende Folgen nach sich ziehen kann.

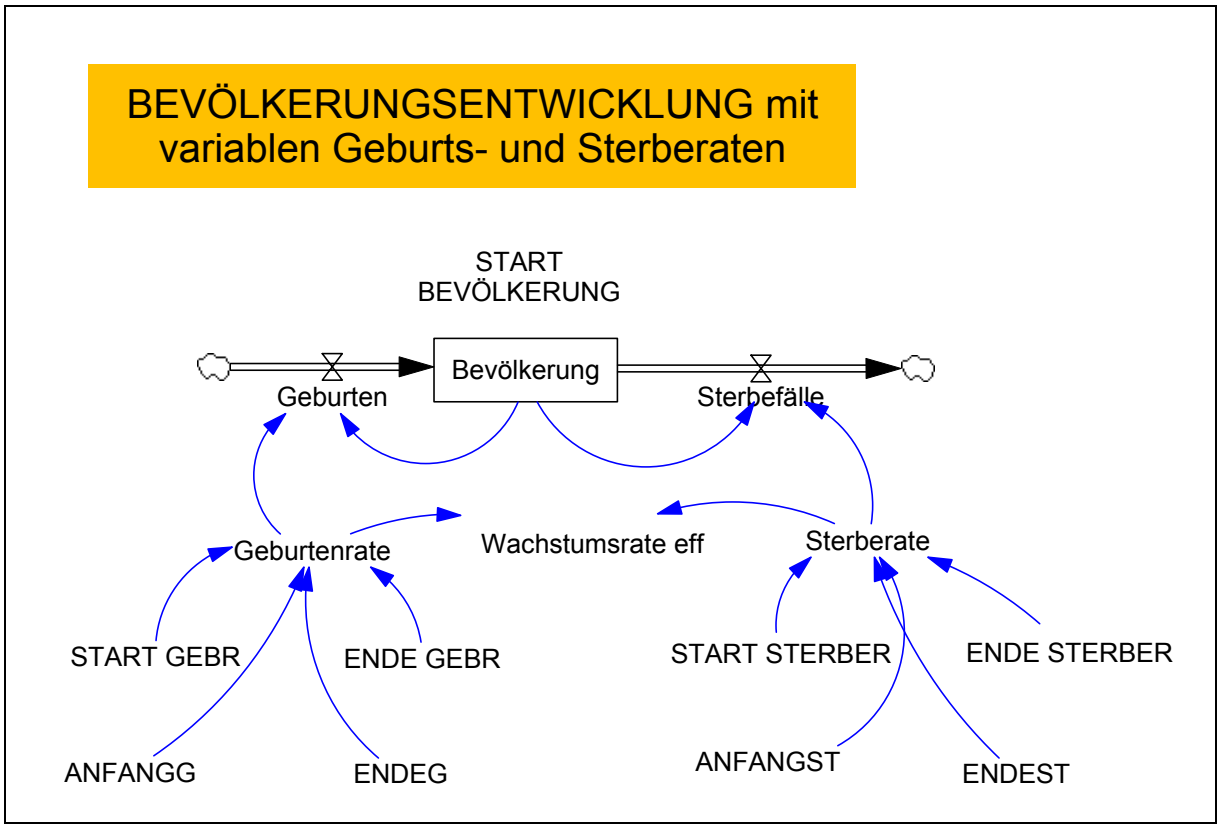
4 Bevölkerungsdynamik

mit variablen Geburts- und Sterberaten

Hier wird die Entwicklung einer Population (Startwert = 1000) beschrieben, bei der sich sowohl die Geburten- als auch die Sterberate mit der Zeit linear ändert. Beide Raten fallen innerhalb eines gewissen Zeitraums gleichmäßig vom Anfangswert zu ihrem Endwert.

Ich beginne wieder mit dem *Simulationsdiagramm* in *VEN-SIM*.

R*Volksschule in Hellville, Nosy Be,Madagaskar*



Hier gibt es mehr Konstante und nur wenig Gleichungen. Neu ist hier die RAMP-Funktion, die die gleichmäßige Veränderung der beiden Raten beschreibt.

Das Dokument listet wieder in alphabetischer Reihenfolge alle gemachten Eingaben auf:

(01) ANFANGG = 2010

(02) ANFANGST = 2010

(03) Bevölkerung = INTEG(Geburten – Sterbefälle, START BEVÖLKERUNG)

(04) ENDE GEBR = 0.01

(05) ENDE STERBER = 0.012

(06) ENDEG = 2060

(07) ENDEST = 2030

(08) FINAL TIME = 2100

(09) Geburten = Geburtenrate ∗ Bevölkerung

(10) Geburtenrate = START GEBR + RAMP((ENDE GEBR – START GEBR)/
(ENDEG – ANFANGG), ANFANGG, ENDEG)

(11) INITIAL TIME = 2000
Units: Jahr

(12) SAVEPER = TIME STEP
Units: Jahr [0,?]

(13) START BEVÖLKERUNG = 1000

(14) START GEBR = 0.04

(15) START STERBER = 0.015

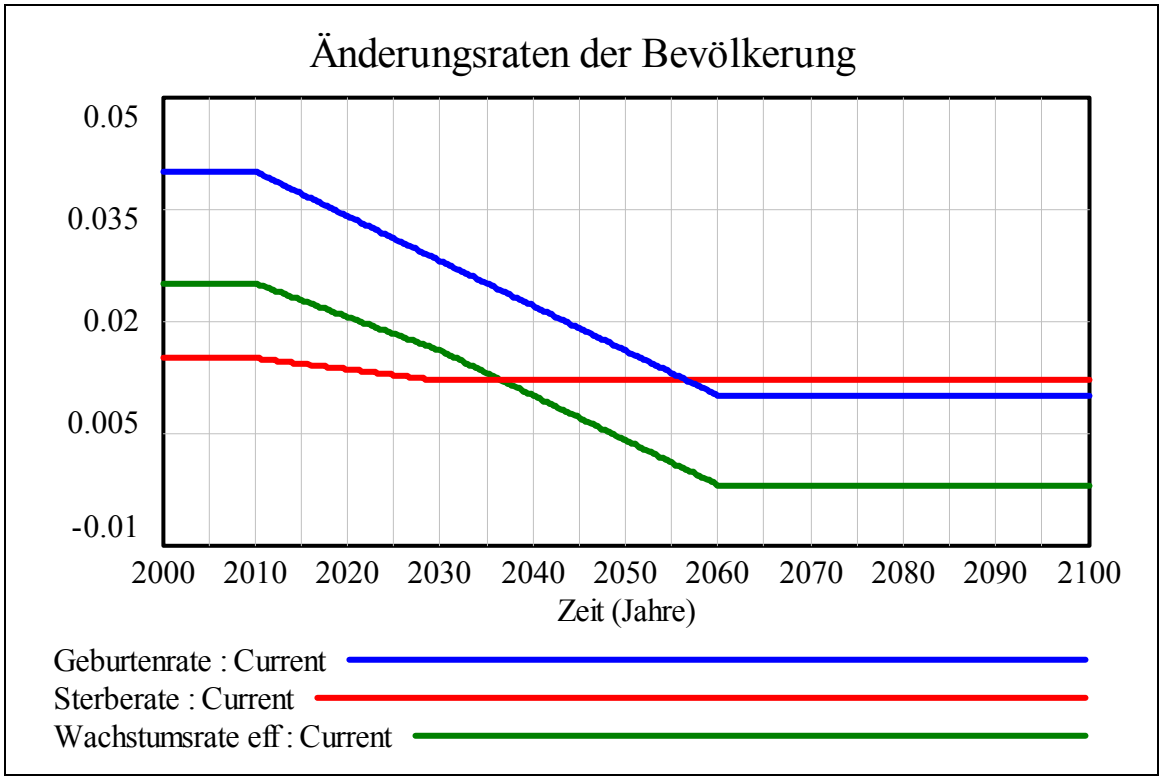
(16) Sterbefälle = Sterberate ∗ Bevölkerung

(17) Sterberate = START STERBER + RAMP((ENDE STERBER – START STERBER)/
(ENDEST – ANFANGST), ANFANGST, ENDEST)

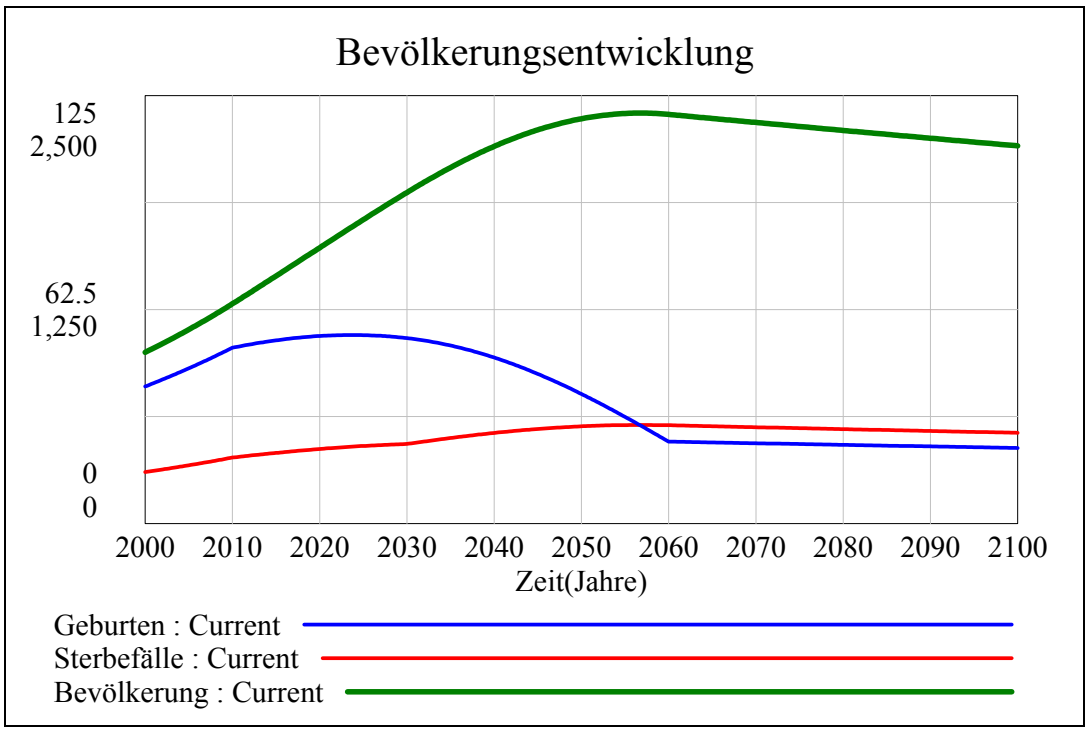
(18) TIME STEP = 0.1
Units: Jahr [0,?]

(19) Wachstumsrate eff = Geburtenrate – Sterberate

Die Graphen der Raten erklären den Namen der Funktion RAMP:



Das nächste Diagramm zeigt die Entwicklung der *Bevölkerung*. Hier nimmt die *Geburtenrate* stärker ab als die *Sterberate*.



Die Entwicklung lässt sich auch in Tabellenform ausgeben. Hier sieht man die letzten Zeilen.

Im *Systemzoo* wird als Zeitinkrement der Wert 0,1 angenommen. Das Ergebnis ändert sich nicht wesentlich wie die beiden folgenden Ausschnitte zeigen.

Entwicklung in Tabelle

Time (Jahr)	Geburten	Sterbefälle	Bevölkerung
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

Entwicklung in Tabelle

Time (Jahr)	Geburten	Sterbefälle	Bevölkerung
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

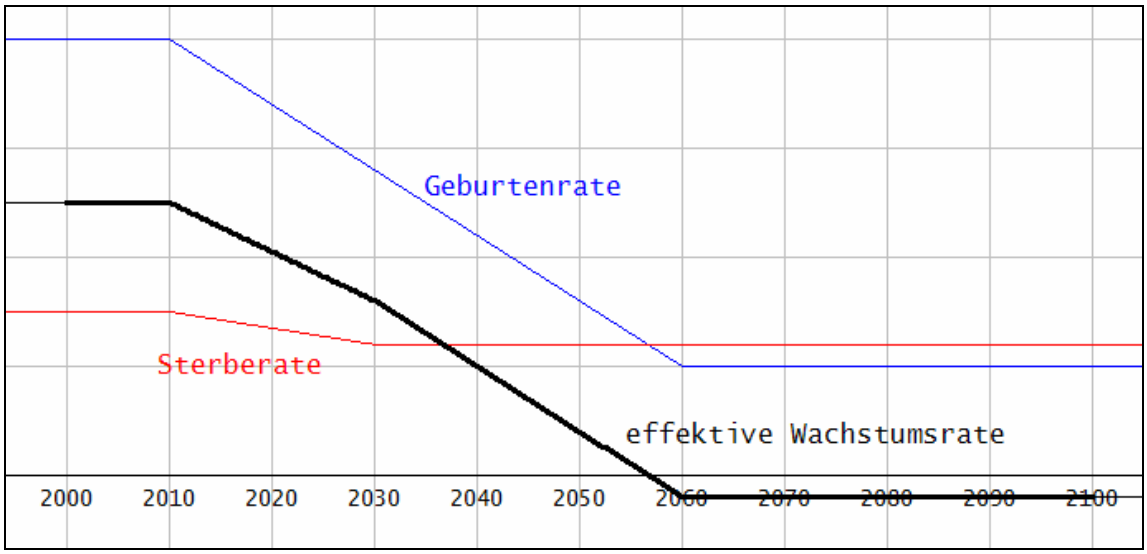
Das Modell mit ITERATES in *DERIVE*

Für rekursive Modelle eignet sich ausgezeichnet die ITERATES-Funktion von *DERIVE*. Sie ist für viele Schüler und Anwender ein wenig gewöhnungsbedürftig, aber sie ist sehr effizient.
Seit der Programmierbarkeit von *DERIVE* kann man natürlich an Stelle dessen ein kleines Programm mit einer Schleife schreiben.

```
#1:   InputMode := Word

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

#3:   bev_entw(dt, n) := 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)
```

Die „Rampenfunktion“ wurde oben definiert und damit lassen sich alle Wachstumsraten darstellen:

```
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)
```



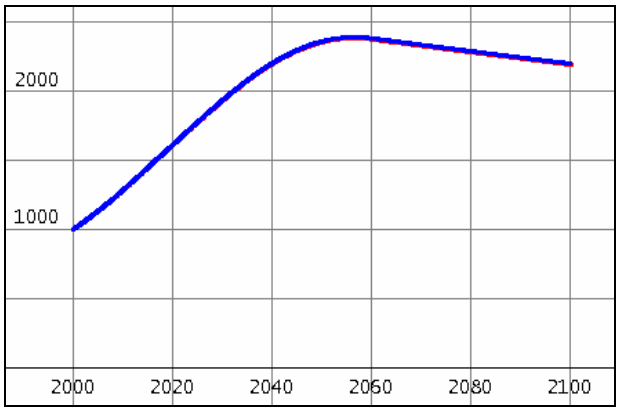
Wir iterieren 401 mal mit einem Zeitintervall von 0,25 Jahren. Damit erreichen wir genau die 100 Jahre von 2000 bis 2100. Die Syntax des ITERATES-Befehls ist eine Spezialität von *DERIVE*.

Die Ausgabe erfolgt in einer Matrix, die genau die Punktkoordinaten (Zeit, Bevölkerungsgröße) enthält. Die Punkte können sofort gezeichnet werden.

Es zeigt sich, dass die Verkleinerung der Schrittweite auf 0,1 keine Veränderung der Grafik nach sich zieht.

```
bev_entw(0.25, 401)

bev_entw(0.1, 1001)
```

Beide Entwicklungen wurden übereinander gezeichnet. Es ist praktisch kein Unterschied festzustellen.



Die nächste Tabelle zeigt die drei ersten und letzten Zeilen der Matrix für den Zeitschritt von 0,25 Jahren, wobei zuerst die ITERATES-Konstruktion von oben verwendet wurde. Daneben ist die gleiche Tabelle bei Einsatz des Programms pop gezeigt, das anschließend entwickelt wird.

```
      (bev_entw(0.25, 401))
#9:                        [1, 2, 3, 399, 400, 401]

      ⎡  2000       1000     ⎤
      ⎢ 2000.25    1006.25   ⎥
      ⎢ 2000.5   1012.539062 ⎥
#10:  ⎢ 2099.5   2193.478612 ⎥
      ⎢ 2099.75  2192.381873 ⎥
      ⎣  2100    2191.285682 ⎦
```

```
(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 ⎦
```

```
pop(n, dt, b, d, 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
```

Die mit pop erzeugten Werte stimmen mit den *VENSIM*-Werten genau überein. Ich habe keine Erklärung dafür, dass die mit ITERATES berechneten Werte gegen Ende der Tabelle nicht übereinstimmen, obwohl das Programm auch die Iteration durchführt.

Die Ausarbeitung mit *TI-NspireCAS*

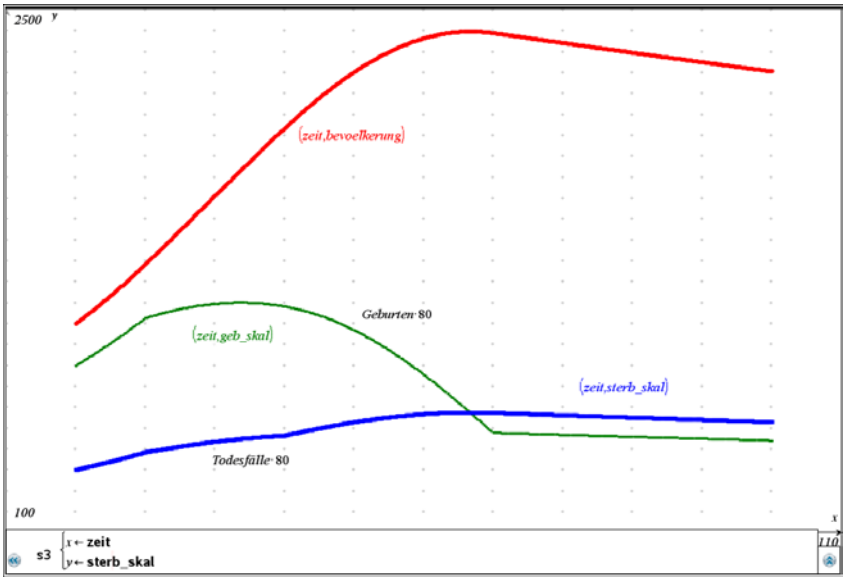
$ramp(x,ax,ex,ay,ey):=when\left(x\le ax,ay,when\left(x\le ex,ay+\frac{ey-ay}{ex-ax}\cdot(x-ax),ey\right)\right)$	Fertig
$gebr(x):=ramp(x,2010,2060,0.04,0.01)$	Fertig
$sterbr(x):=ramp(x,2010,2030,0.015,0.012)$	Fertig
$bev\_start:=1000$	1000
$dx:=0.25$	0.25

Das Spreadsheet liefert die Listen für *Geburten*, *Sterbefälle* und für die *Gesamtbevölkerung*.

	A zeit	B geburten	C sterbef	D bevoelkerung	E geb_skal	F sterb_skal
◆					=geburten*80	=sterbef*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

B2 | =d2·gebr(a2)·dx

Um alle drei Größen gemeinsam darstellen zu können, habe ich die *Geburten* und die *Sterbefälle* umskaliert, indem ich ihre Anzahlen mit 80 multipliziert habe. Damit ergibt sich eine Grafik, die wir schon kennen.



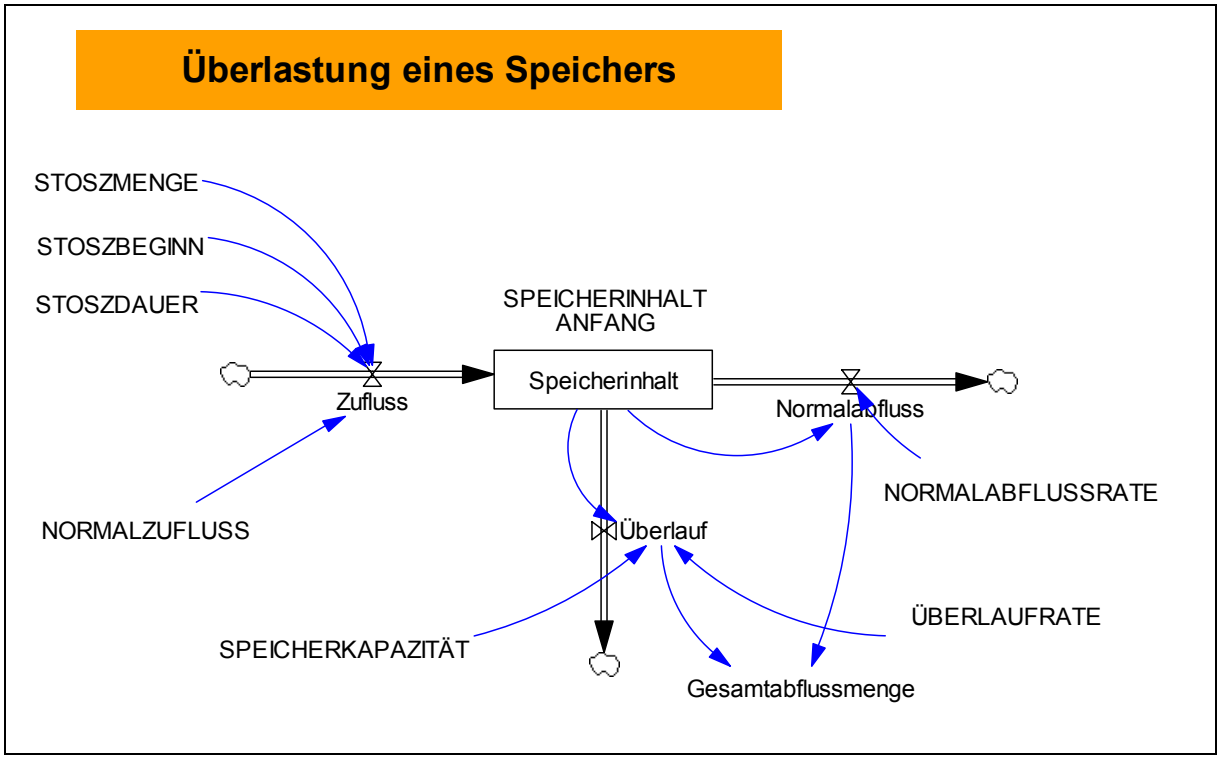
5 Der Speicher läuft über!

Wir simulieren die Dynamik eines *Speicherinhalts*, dessen SPEICHERKAPAZITÄT durch einen konstanten NORMALZUFLUSS und überdies durch einen pulsförmigen (= stoßförmigen) Zufluss überschritten werden kann (z.B. bei Schneeschmelze, Starkregen, …). Wir nehmen an, dass der *Normalabfluss* proportional zum jeweiligen *Speicherinhalt* mit einer NORMALABFLUSSRATE erfolgt.

Speicherkraftwerk Kaprun, Salzburg

Bei Überlastung des Speichers fließt der Überschuss als *Überlauf* mit einer ÜBERLAUFRATE ab, die über der NORMALABFLUSSRATE liegt. Die Stoßbelastung wird durch die STOSZMENGE beschrieben, die zu STOSZBEGINN einsetzt und eine gewisse STOSZDAUER aufweist.

Das *Simulationsdiagramm* ist mit *VENSIM* leicht erstellt.



Das Dokument - in eine gewisse Ordnung gebracht - stellt alle Daten des Modells zusammen:

NORMALABFLUSSRATE = 0.5

NORMALZUFLUSS = 0.25

SPEICHERINHALT ANFANG = 0.3

SPEICHERKAPAZITÄT = 1

STOSZBEGINN = 5

STOSZDAUER = 0.2

STOSZMENGE = 10

ÜBERLAUFRATE = 10

Gesamtabflussmenge = Normalabfluss + Überlauf

Normalabfluss = NORMALABFLUSSRATE * Speicherinhalt

Speicherinhalt = INTEG (+Zufluss – Normalabfluss – Überlauf, SPEICHERINHALT ANFANG)

Überlauf = IF THEN ELSE(Speicherinhalt > SPEICHERKAPAZITÄT,
ÜBERLAUFRATE * (Speicherinhalt – SPEICHERKAPAZITÄT) ,0)

Zufluss = NORMALZUFLUSS + STOSZMENGE * PULSE(STOSZBEGINN, STOSZDAUER)

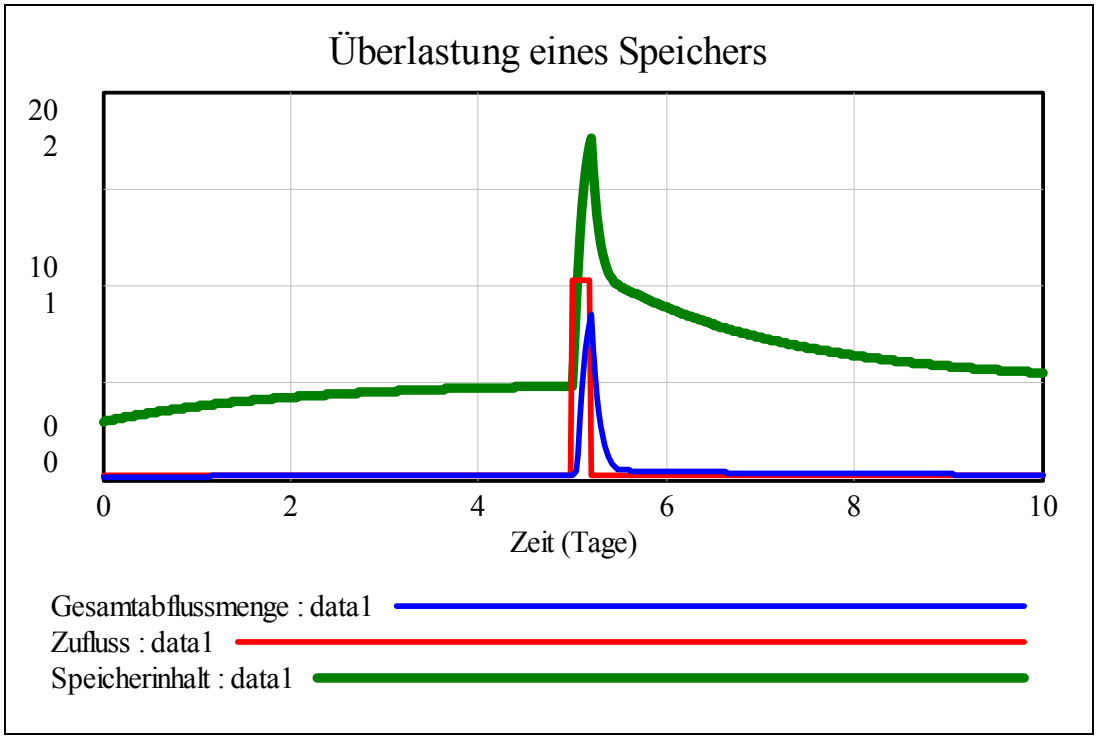
INITIAL TIME = 0
Units: Tag

FINAL TIME = 20
Units: Tag

TIME STEP = 0.02
Units: Tag [0,?]

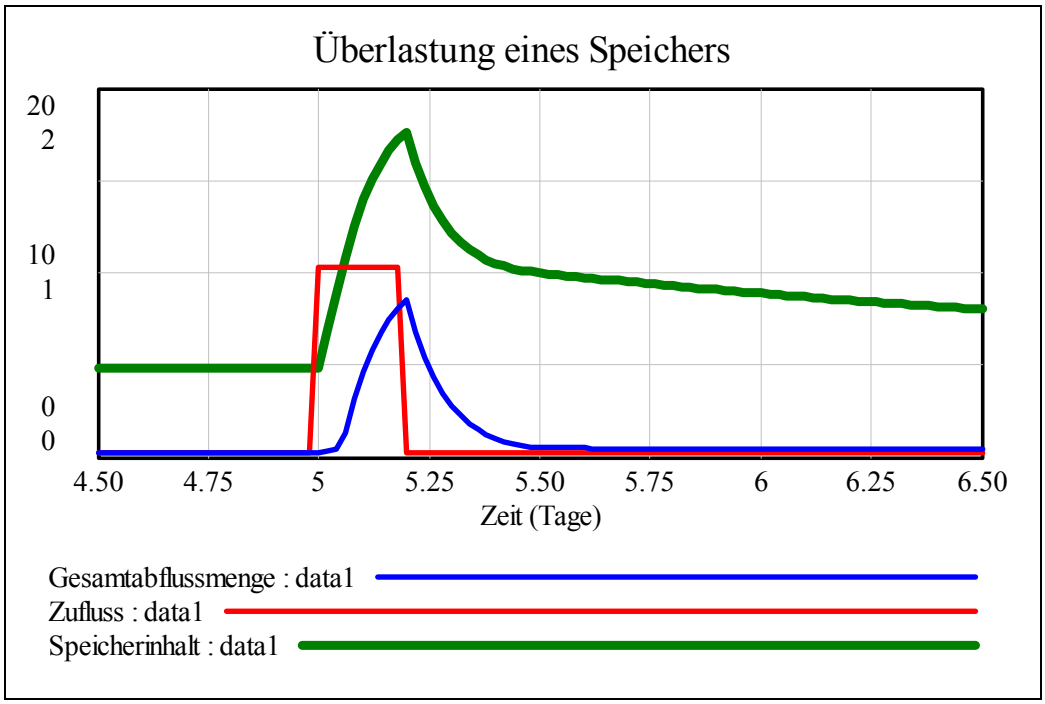
SAVEPER = TIME STEP
Units: Tag [0,?]

Wir betrachten das Diagramm für die vorgeschlagenen Parameter:



Skalierung: $0 \le$ *Speicherinhalt* ≤ 2, $0 \le$ *Zufluss*, *Gesamtabflussmenge* ≤ 20

Den spannenden Abschnitt zwischen den Tagen 4 und 6 können wir aufspreizen:

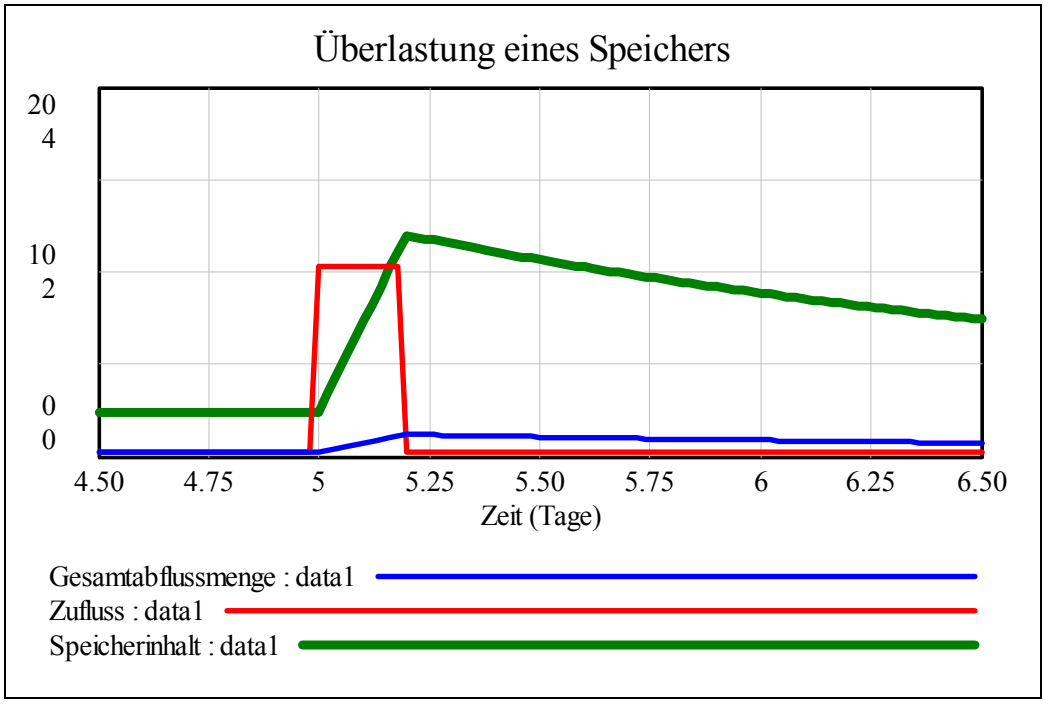


Wie lässt sich dieses Diagramm interpretieren?

Der Speicher füllt sich langsam und hat nach etwa 4,5 Tagen den Gleichgewichtswert von 0,5 Einheiten erreicht. Dann kommt der stoßartige *Zufluss*, der den Speicher rasch zum Überlaufen bringt. Dieser *Überlauf* hält solange an, bis der *Speicherinhalt* wieder auf seine KAPAZITÄT von einer Einheit gesunken ist. Dann stellt sich nach etwa 10 Tagen wieder der Normalzustand her.

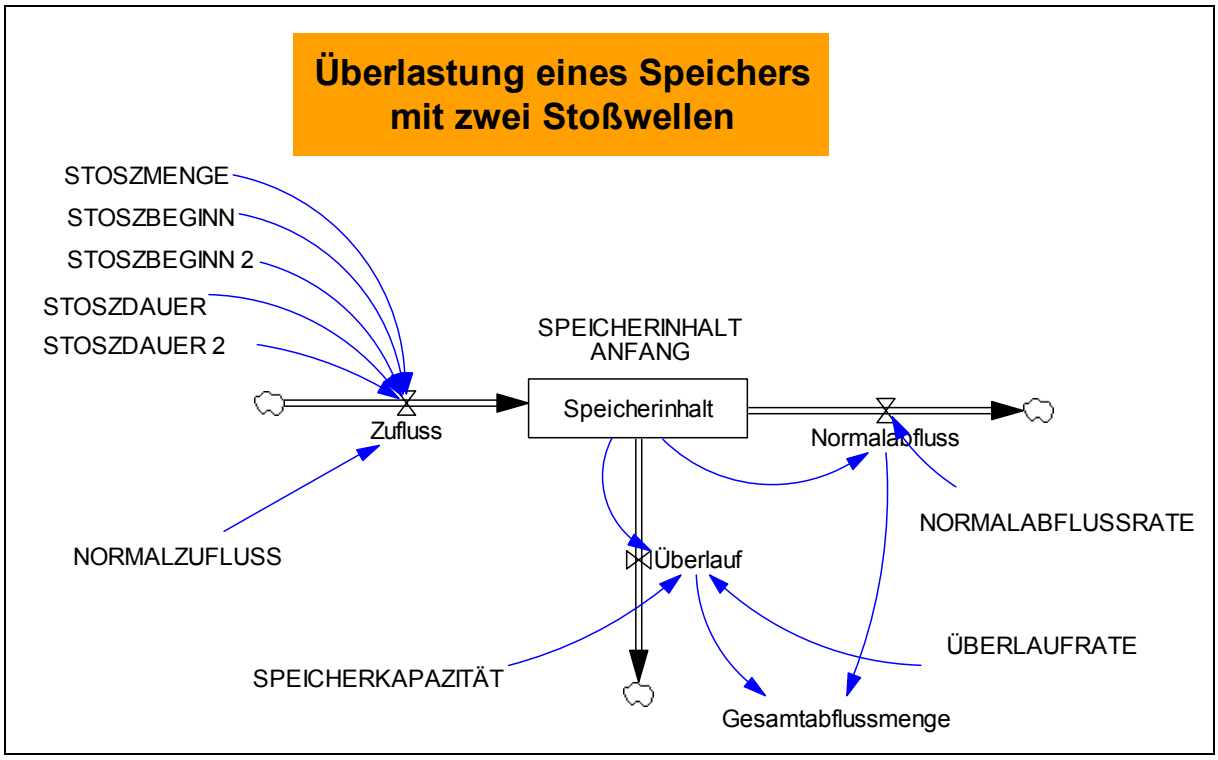
Natürlich hängt das System vor allem von der SPEICHERKAPAZITÄT und von der NORMALABFLUSSRATE ab. So lange diese hinreichend groß sind, kommt es nicht leicht zu einem *Überlauf*.

Wir simulieren den Verlauf bei einer Vergrößerung des Speichers auf 2,5 Einheiten:



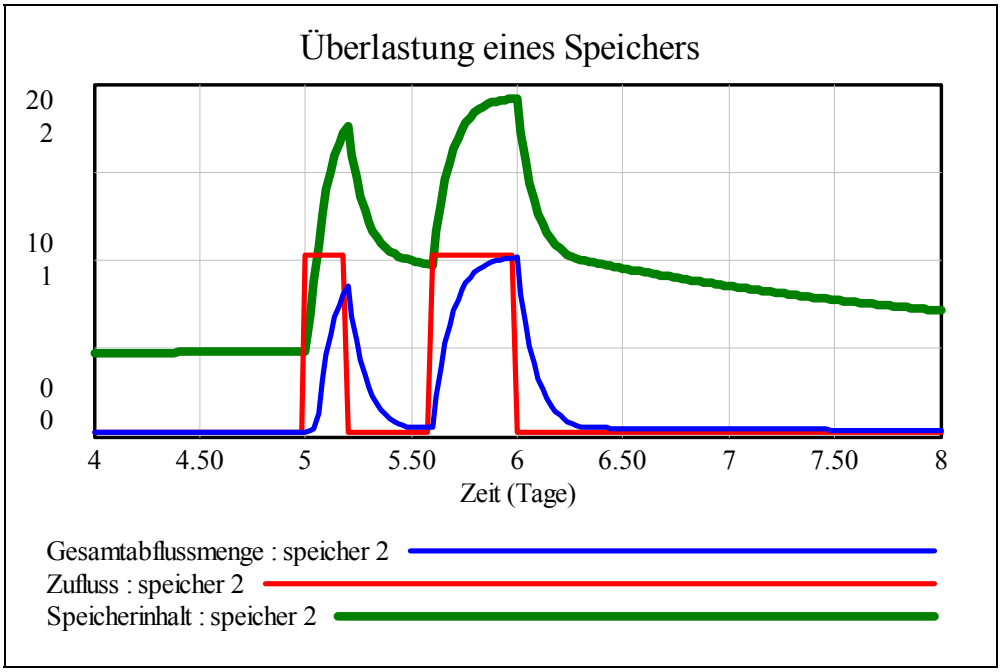
Man muss auf die Änderung der senkrechten Skalierung achten: für den *Speicherinhalt* gilt nun der Bereich von 0 bis 4. Die größere Kapazität des Speichers nimmt dem plötzlichen Zustrom doch die Spitze.

Wenn wir schon beim Modellieren sind, dann wollen wir auch untersuchen, was passiert, wenn bald nach der ersten Stoßwelle noch eine zweite folgt – das Unwetter kommt zurück. Wir fügen einen zweiten PULSE ein.



Die STOSZMENGE soll dieselbe sein, der zweite Stoß beginnt zum Zeitpunkt 5,6 und hat die Dauer von 0,4 Zeiteinheiten. Es ändert sich nur die Gleichung für den *Zufluss*.

Zufluss = NORMALZUFLUSS + STOSZMENGE ∗ PULSE(STOSZBEGINN,
STOSZDAUER) + STOSZMENGE ∗ PULSE(STOSZBEGINN 2, STOSZDAUER 2)



Zum Vergleich für die nächsten Realisierungen des Modells zeigen wir interessante Zeilen aus der Tabelle:

Time (Tag)	Gesamtabflussmenge	Zufluss	Speicherinhalt
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

Jetzt kommt die erste Welle in den Speicher:

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

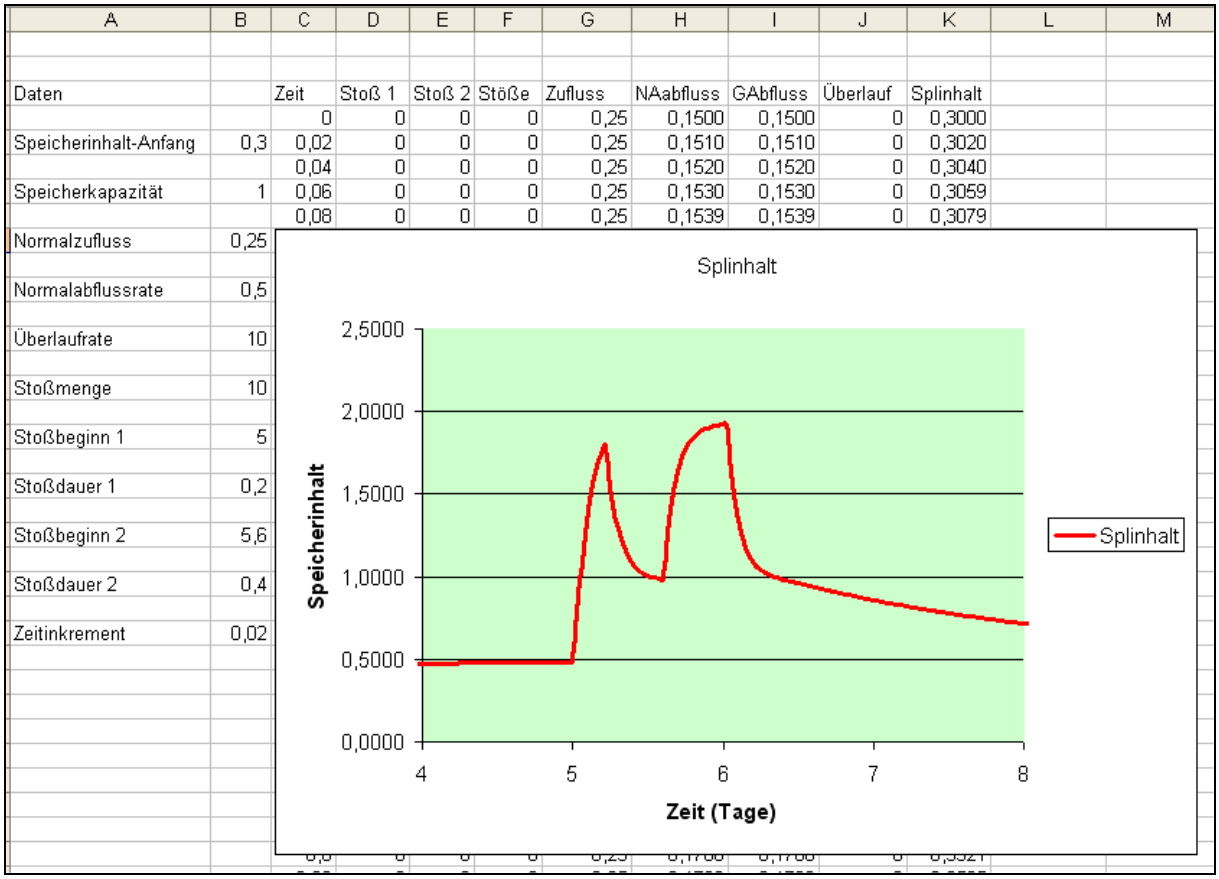
Zum Zeitpunkt 5.6 kommt die zweite Welle und ergießt sich in den Speicher.

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

Nun herrscht schon längere Zeit wieder Normalität.

Mit *MS-Excel* kann das Modell so aussehen:

Im ersten Modell sind die Daten von oben eingesetzt (mit zwei Stoßwellen).

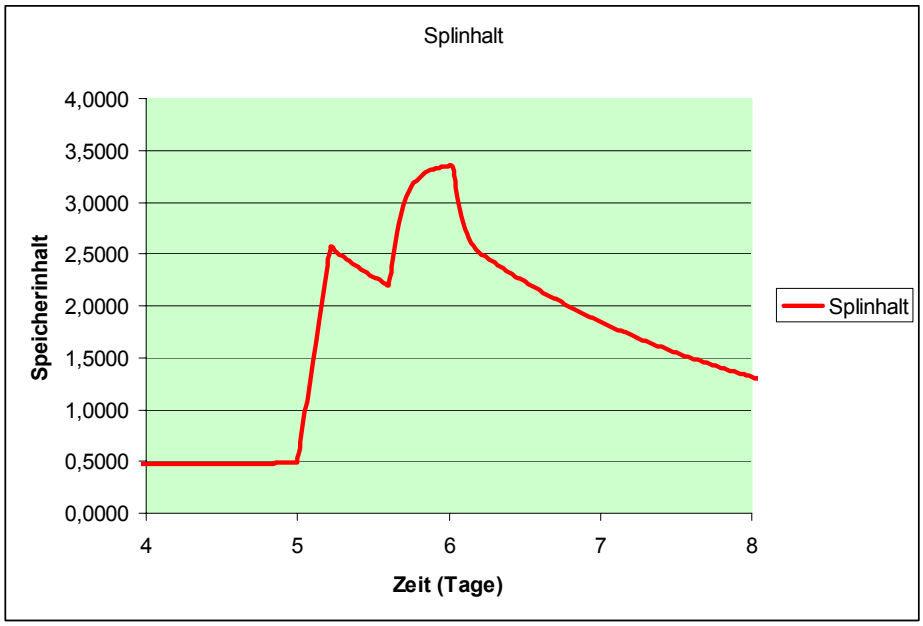


	A	B	C	D	E	F	G	H	I	J	K	L	M
	Daten		Zeit	Stoß 1	Stoß 2	Stöße	Zufluss	NAabfluss	GAbfluss	Überlauf	Splinhalt		
			0	0	0	0	0,25	0,1500	0,1500	0	0,3000		
	Speicherinhalt-Anfang	0,3	0,02	0	0	0	0,25	0,1510	0,1510	0	0,3020		
			0,04	0	0	0	0,25	0,1520	0,1520	0	0,3040		
	Speicherkapazität	1	0,06	0	0	0	0,25	0,1530	0,1530	0	0,3059		
			0,08	0	0	0	0,25	0,1539	0,1539	0	0,3079		
	Normalzufluss	0,25											
	Normalabflussrate	0,5											
	Überlaufrate	10											
	Stoßmenge	10											
	Stoßbeginn 1	5											
	Stoßdauer 1	0,2											
	Stoßbeginn 2	5,6											
	Stoßdauer 2	0,4											
	Zeitinkrement	0,02											

Nach 20 Tagen sieht es aus wie bei *VENSIM*:

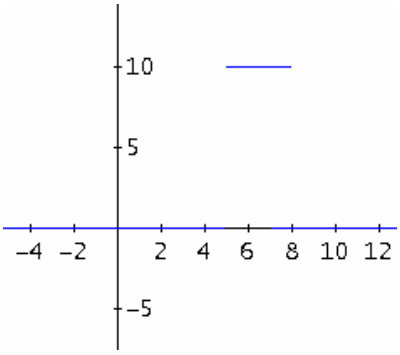
20	**0**	**0**	**0**	**0,25**	**0,2502**	**0,2502**	**0**	**0,5005**

Bei einer auf 2,5 Einheiten erhöhten Speicherkapazität ändert sich das Bild:



Da wir das Modell schon zweimal erfolgreich umgesetzt haben, ist es nun ein Leichtes, das Speicherverhalten auch in ein ***DERIVE*-Programm** zu gießen. Die PULSE- (= Stoß-) Funktion wird vordefiniert:

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

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



Das Simulationsprogramm ist relativ kurz:

```
speicher(nar, nz, spa, spk, stm1, stb1, std1, stm2, stb2, std2, uelr, t_start, t_end,
  Prog
    tab := []
    t := t_start
    Loop
      If t > t_end
         RETURN tab
      gzufl := nz + puls(stm1, stb1, std1, t) + puls(stm2, stb2, std2, t)
      uebfl := IF(spa > spk, uelr·(spa − spk), 0)
      gabfl := nar·spa + uebfl
      tab := APPEND(tab, [[t, gabfl, gzufl, spa]])
      spa := spa + dt·(gzufl − gabfl)
      t :+ dt

  dt, tab, t, spi, gzufl, uebfl, gabfl) :=
```

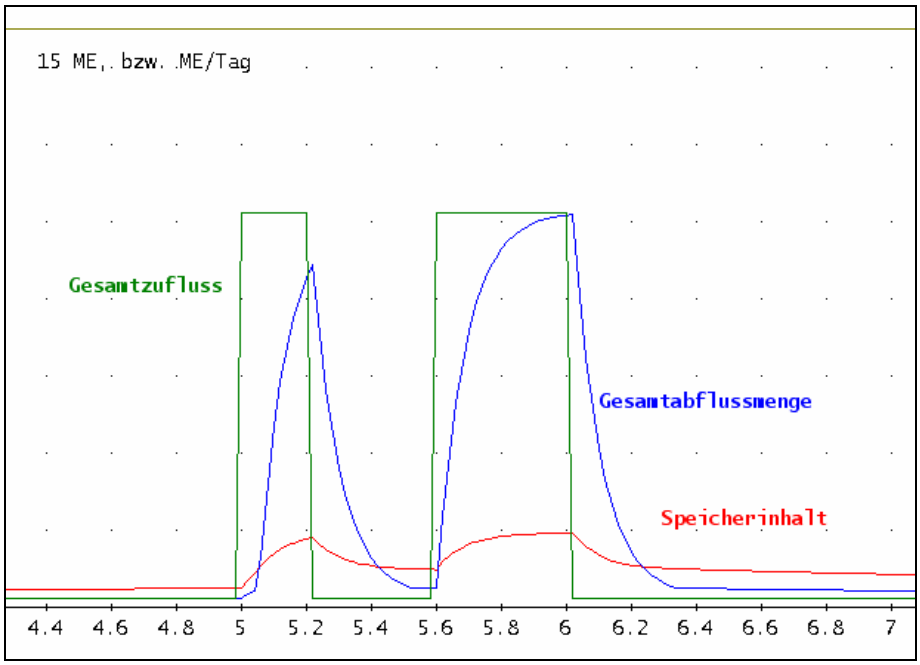
Ich gebe einige Tabellenwerte aus und vergleiche mit *VENSIM* bzw. *MS-Excel*:

```
speicher(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  ⎦

(speicher(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 ⎦
```

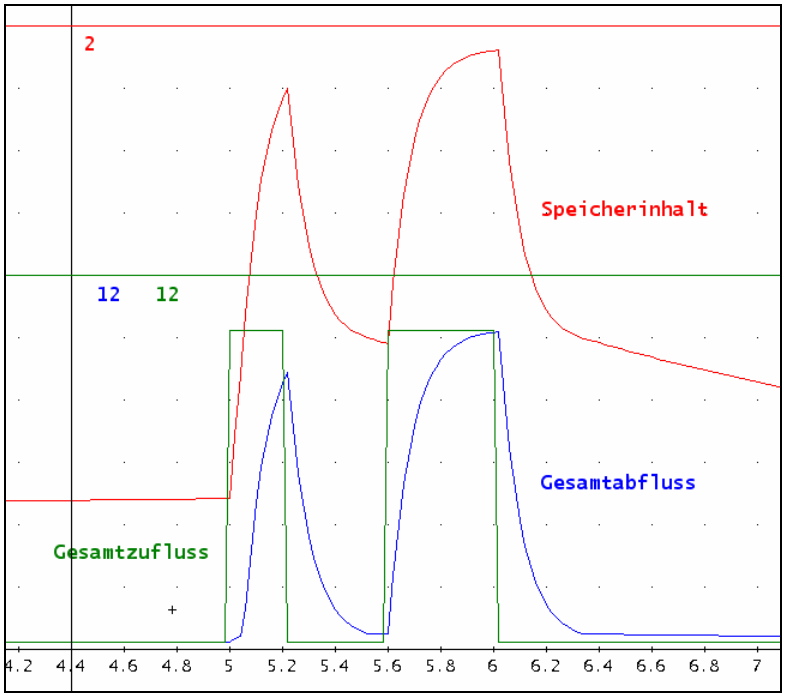
Das sieht ja recht gut aus. In der Grafik wähle ich für alle Größen die gleiche *y*-Skalierung:

```
(speicher(0.5, 0.25, 0.3, 1, 10, 5, 0.2, 10, 5.6, 0.4, 10, 0, 10, 0.02))↓↓[1, 4]
(speicher(0.5, 0.25, 0.3, 1, 10, 5, 0.2, 10, 5.6, 0.4, 10, 0, 10, 0.02))↓↓[1, 2]
(speicher(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
```

Zur besseren Orientierung ziehe ich eine Horizontale in der Höhe 15.



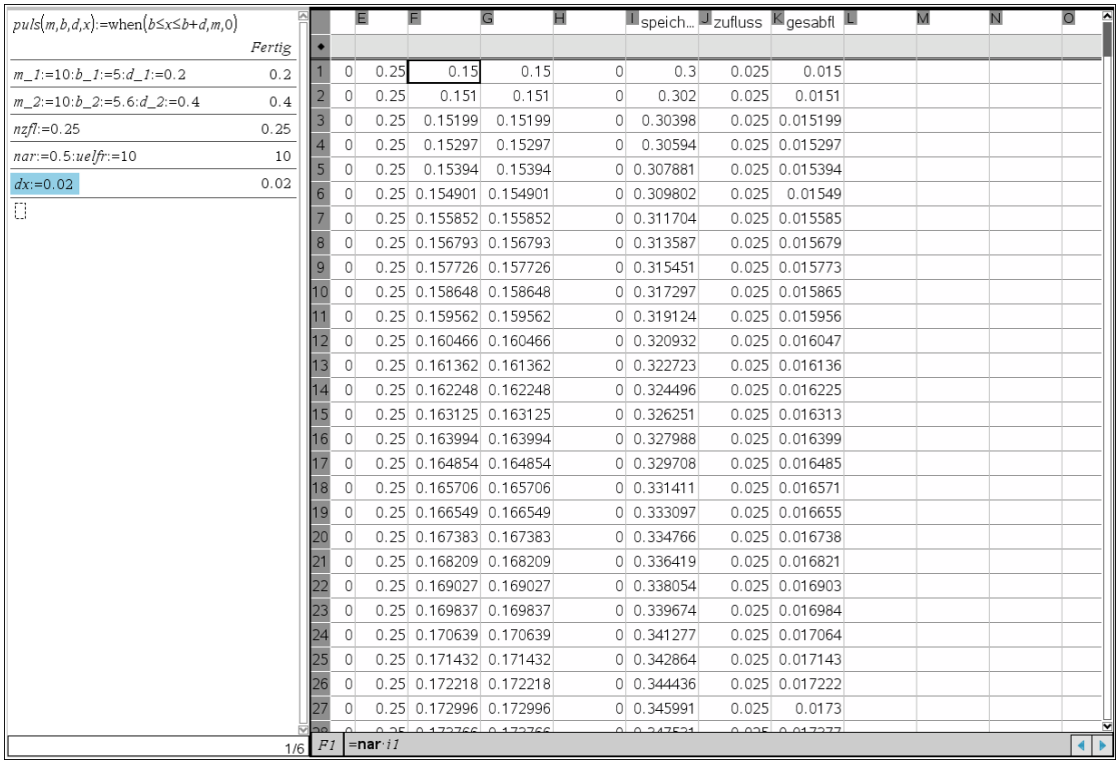
Leider können auch hier keine Schieberegler für die unterschiedlichen Parameter eingeführt werden. Aber mit einem kleinen Trick lässt sich die gleiche Grafik erzielen wie mit *VENSIM*.



Bossel stellt in seinem Buch die Aufgabe, den Einfluss der SPEICHERKAPAZITÄT auf die Vermeidung von Flutspitzen im Speicherabfluss zu untersuchen. Da wäre natürlich ein Schieberegler ideal.

In *GeoGebra* lässt sich das verwirklichen, aber das System ist zurzeit nicht stabil genug und die Rechenzeiten sind zu groß – bis zum „Aufhängen“ des Programms. Hingegen kann man das mit *TI-NspireCAS* sehr ordentlich mit relativ wenig Aufwand erreichen.

Die Ausarbeitung mit *TI-NspireCAS*

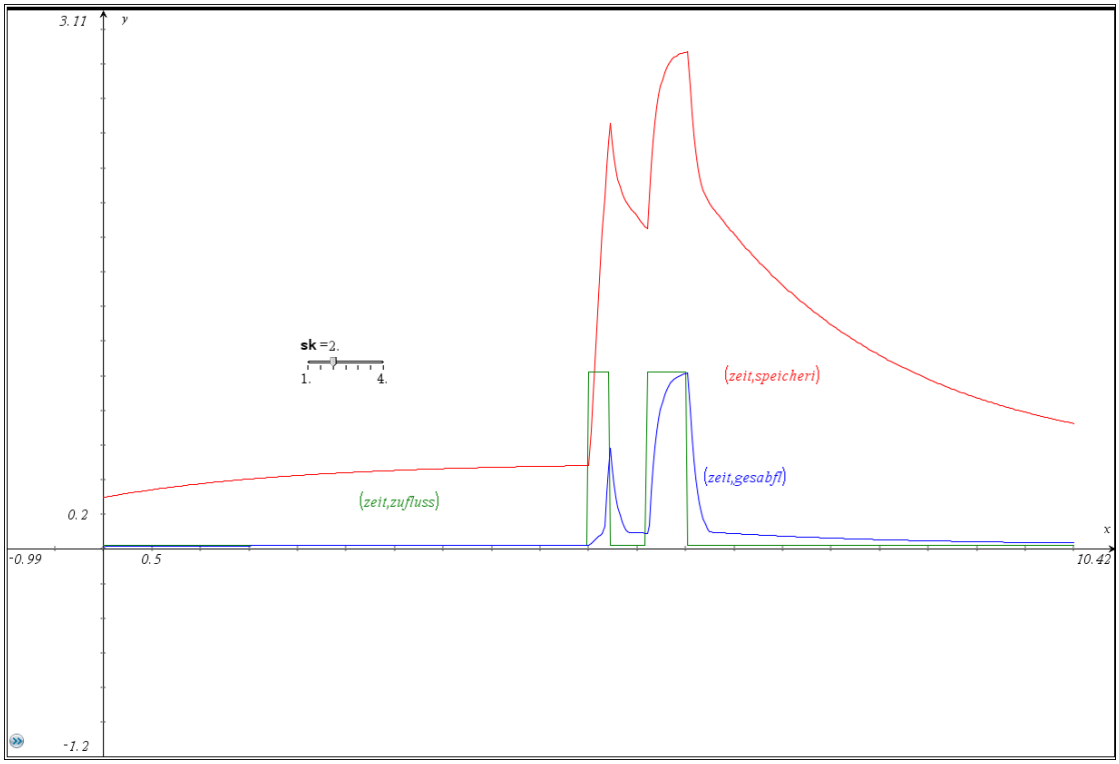


puls(m,b,d,x):=when(b≤x≤b+d,m,0) — Fertig

m_1:=10:b_1:=5:d_1:=0.2	0.2
m_2:=10:b_2:=5.6:d_2:=0.4	0.4
nzfl:=0.25	0.25
nar:=0.5:uelfr:=10	10
dx:=0.02	0.02

	E	F	G	H	I speich...	J zufluss	K gesabfl	L	M	N	O
1	0	0.25	0.15	0.15	0	0.3	0.025	0.015			
2	0	0.25	0.151	0.151	0	0.302	0.025	0.0151			
3	0	0.25	0.15199	0.15199	0	0.30398	0.025	0.015199			
4	0	0.25	0.15297	0.15297	0	0.30594	0.025	0.015297			
5	0	0.25	0.15394	0.15394	0	0.307881	0.025	0.015394			
6	0	0.25	0.154901	0.154901	0	0.309802	0.025	0.01549			
7	0	0.25	0.155852	0.155852	0	0.311704	0.025	0.015585			
8	0	0.25	0.156793	0.156793	0	0.313587	0.025	0.015679			
9	0	0.25	0.157726	0.157726	0	0.315451	0.025	0.015773			
10	0	0.25	0.158648	0.158648	0	0.317297	0.025	0.015865			
11	0	0.25	0.159562	0.159562	0	0.319124	0.025	0.015956			
12	0	0.25	0.160466	0.160466	0	0.320932	0.025	0.016047			
13	0	0.25	0.161362	0.161362	0	0.322723	0.025	0.016136			
14	0	0.25	0.162248	0.162248	0	0.324496	0.025	0.016225			
15	0	0.25	0.163125	0.163125	0	0.326251	0.025	0.016313			
16	0	0.25	0.163994	0.163994	0	0.327988	0.025	0.016399			
17	0	0.25	0.164854	0.164854	0	0.329708	0.025	0.016485			
18	0	0.25	0.165706	0.165706	0	0.331411	0.025	0.016571			
19	0	0.25	0.166549	0.166549	0	0.333097	0.025	0.016655			
20	0	0.25	0.167383	0.167383	0	0.334766	0.025	0.016738			
21	0	0.25	0.168209	0.168209	0	0.336419	0.025	0.016821			
22	0	0.25	0.169027	0.169027	0	0.338054	0.025	0.016903			
23	0	0.25	0.169837	0.169837	0	0.339674	0.025	0.016984			
24	0	0.25	0.170639	0.170639	0	0.341277	0.025	0.017064			
25	0	0.25	0.171432	0.171432	0	0.342864	0.025	0.017143			
26	0	0.25	0.172218	0.172218	0	0.344436	0.025	0.017222			
27	0	0.25	0.172996	0.172996	0	0.345991	0.025	0.0173			

1/6 — F1 =nar:i1



Die Berechnung wird im Spreadsheet durchgeführt. Die Listen können leicht im Diagramm dargestellt werden. Da wären auch Schieberegler für weitere Parameter denkbar.

6 Dichte abhängiges Wachstum: Michaelis-Menten-Kinetik

Eine Wachstumsfunktion, die dem logistischen Wachstum ähnlich ist, basiert auf der Michaelis-Menten-Kinetik.

Leonor Michaelis, 1875 – 1949, deutscher Biochemiker und Maud Leonora Menten, 1879 – 1960, kanadische Medizinwissenschafterin

Diese Funktion eignet sich besonders zur Beschreibung von Sättigungsprozessen in der Chemie.

Die Reaktionsgeschwindigkeit v ist u.a. natürlich abhängig von der Konzentration des jeweiligen Bestands und lässt sich beschreiben durch die Gleichung

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

Dabei ist S die jeweilige Substratkonzentration. c ist die Dissoziationskonstante oder Michaelis-Menten-Konstante oder auch Halbsättigungskonstante. Die letzte Bezeichnung leitet sich davon ab, dass sich für $S = c$ der halbe Sättigungseffekt ergibt.

Ein schöne Beschreibung der Michaelis-Menten-Kinetik findet sich in

[8] http://www.isitech.com/fileadmin/pb/pdf-Dateien/Michaelis_Menten_Kinetik.pdf

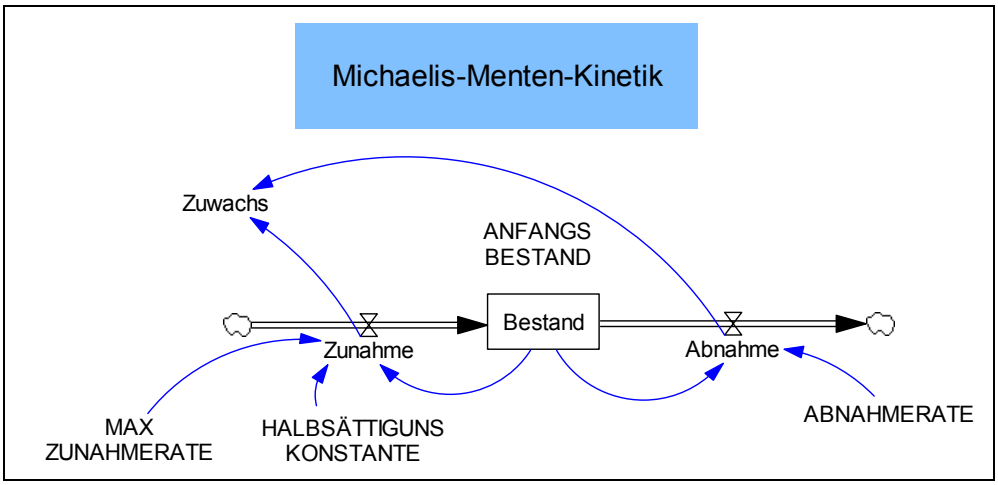
Besonders reizvoll ist, dass die Simulationen in diesem Papier auch mit *VENSIM* durchgeführt werden.

Während beim logistischen Wachstum der Bestandszuwachs beschrieben wird durch $r \cdot B \cdot \left(1 - \frac{B}{K}\right)$,

gilt hier für den Zuwachs: $r \cdot B \cdot \left(1 - \frac{B}{B+c}\right)$. Dabei ist r die MAXIMALE WACHSTUMSRATE. Für die Abnahme des Bestands wird mit e eine ABNAHMERATE eingeführt.

Damit lautet die Differentialgleichung für diese Form des Wachstums: $B' = r \cdot B \cdot \left(1 - \frac{B}{B+c}\right) - e \cdot B.$

In diesem Fall ist das *VENSIM*-Simulationsdiagramm nicht sehr aufwändig:



Es folgt die Modellbeschreibung (wieder in alphabetischer Reihenfolge).

Hier habe ich erstmals auch die Dimensionen der einzelnen Größen eingegeben. Ich kann *VENSIM* veranlassen, eine Dimensionsanalysis durchzuführen und die Konsistenz aller Größen zu überprüfen. Hier erhalte ich die Antwort: *Units are O.K*. Eine Überprüfung des ganzen Modells endet mit der Feststellung: *Zuwachs is not used in the model*. Das stimmt, da diese Größe nur einen Zwischenwert darstellt, den ich ausgeben könnte.
(Das hat sich auch für *Wachstumsrate eff* bei der Bevölkerungsdynamik in Kapitel 4 ergeben.)

(01) Abnahme = ABNAHMERATE ∗ Bestand
Units: Menge/Tag

(02) ABNAHMERATE = 0.5
Units: 1/Tag

(03) ANFANGSBESTAND = 0.02
Units: Menge

(04) Bestand = INTEG (Zuwachs – Ernte, ANFANGSBESTAND)
Units: Menge

(05) FINAL TIME = 50
Units: Tag

(06) HALBSÄTTIGUNSKONSTANTE = 1
Units: Menge

(07) INITIAL TIME = 0
Units: Tag

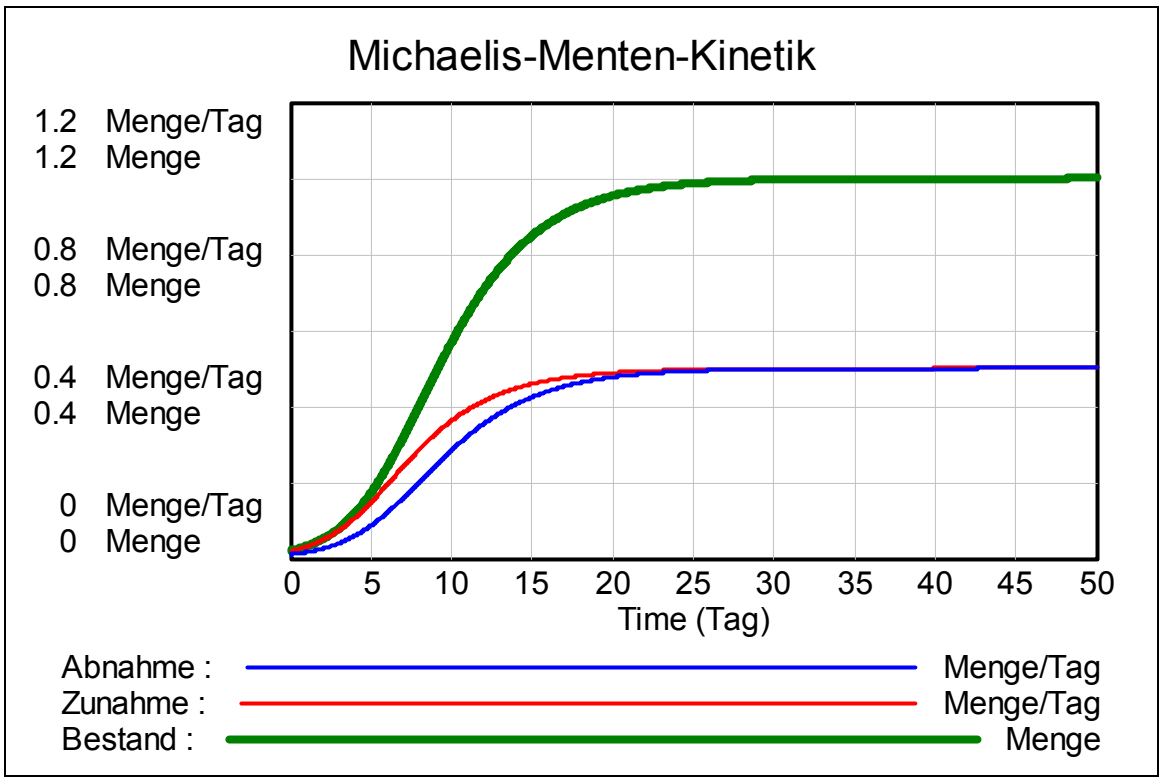
(08) MAX ZUNAHMERATE = 1
Units: 1/Tag

(09) SAVEPER = TIME STEP
Units: Tag [0,?]

(10) TIME STEP = 0.02
Units: Tag [0,?]

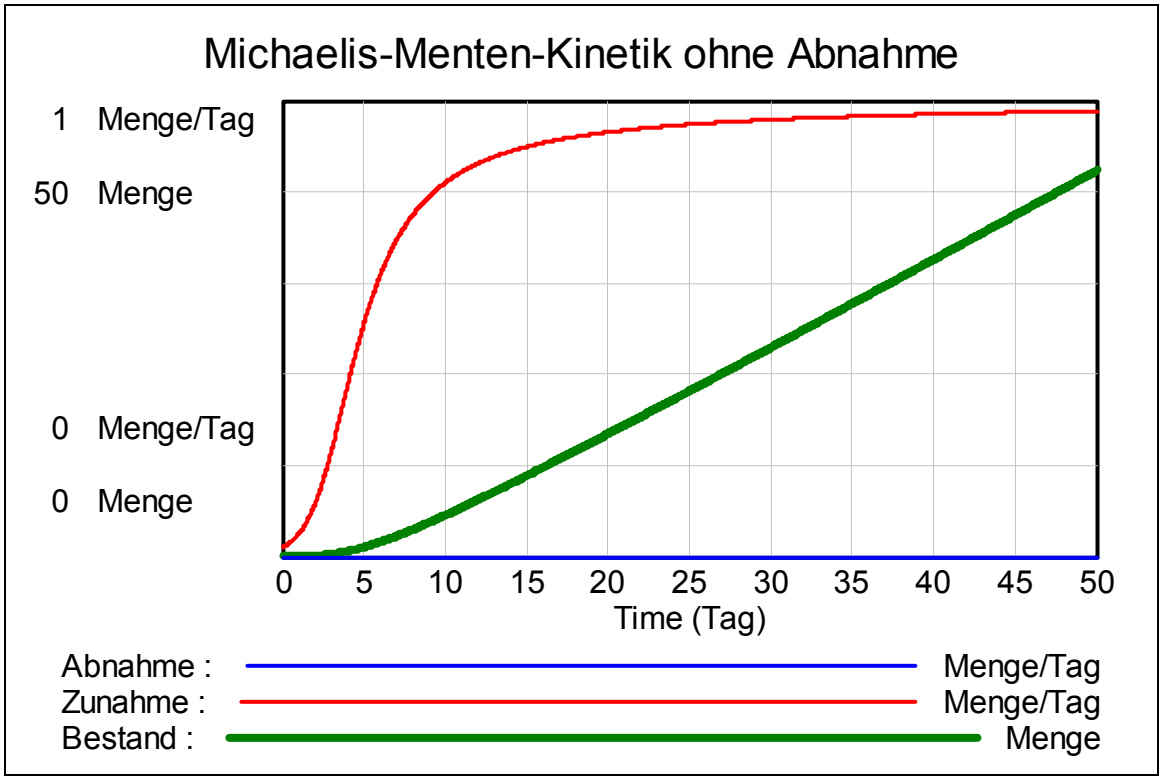
(11) Zunahme = MAX WACHSTUMSRATE ∗ Bestand ∗
(1 – Bestand/(HALBSÄTTIGUNSKONSTANTE + Bestand))
Units: Menge/Tag

(12) Zuwachs = Zunahme – Abnahme
Units: Menge/Tag



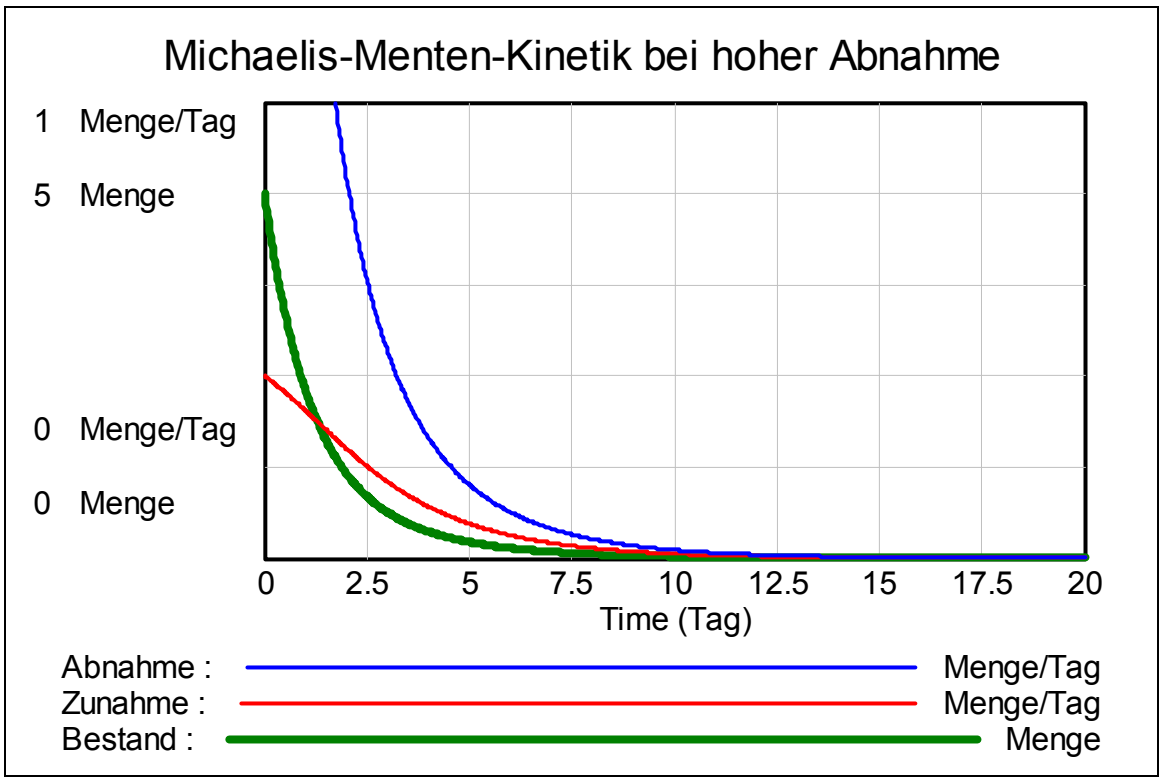
Bossel zeigt noch zwei Extremfälle:

(1) Was geschieht, wenn keine Abnahme (z.B. Ernte) eintritt ($e = 0$)?



(2) Was geschieht bei einer (zu) hohen Abnahmerate? (z.B. bei $r = 0{,}5$, $e = 0{,}9$ und einem Anfangsbestand = 4)

Wie die nächste Grafik zeigt, zerfällt der Bestand natürlich sehr rasch.



Bossel bietet einige Arbeitsvorschläge im Zusammenhang mit diesem Modell:

Untersuchen Sie das Verhalten des Systems für

(a) verschiedene MAX WACHSTUMSRATEN r bei konstanter ABNAHMERATE $e > 0$,

(b) verschiedene ABNAHMERATEN e bei konstanter MAX WACHSTUMSRATE r,

(c) verschiedene Werte für die HALBSÄTTIGUNGSKONSTANTE c.

Dafür wäre natürlich wieder der Einsatz von Schiebereglern ideal. Aber auch die simultane Darstellung von mehreren Bestandskurven für eine Reihe von Parametern ist sehr aussagekräftig. Dazu eignet sich der VECTOR-Befehl von *DERIVE* ausgezeichnet.

Bearbeitung mit *DERIVE* als Differentialgleichung

Wir verwenden das CAS auch gleich dazu, den Gleichgewichtszustand (= Sättigungswert) zu berechnen.

$$B^* = \frac{c \cdot (r - e)}{e}$$

$$B = B + r \cdot B \cdot \left(1 - \frac{B}{c + B}\right) - e \cdot B$$

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

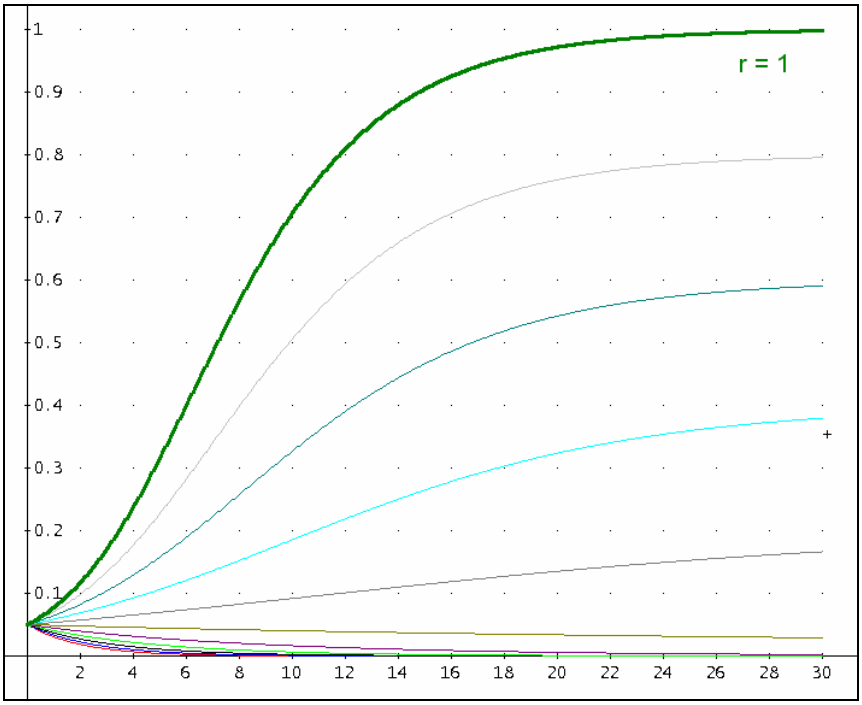
$$B = \frac{c \cdot (r - e)}{e} \vee B = 0$$

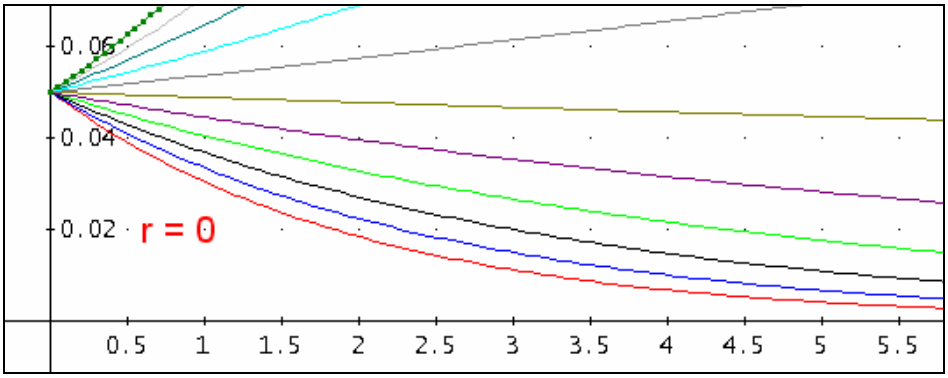
Für die numerische Lösung der DGL verwenden wir wieder die Runge-Kutta-Methode:

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

$$\text{VECTOR}(\text{MMK}(r,\ 0.5,\ 1,\ 0.05,\ 30),\ r,\ 0,\ 1,\ 0.1)$$

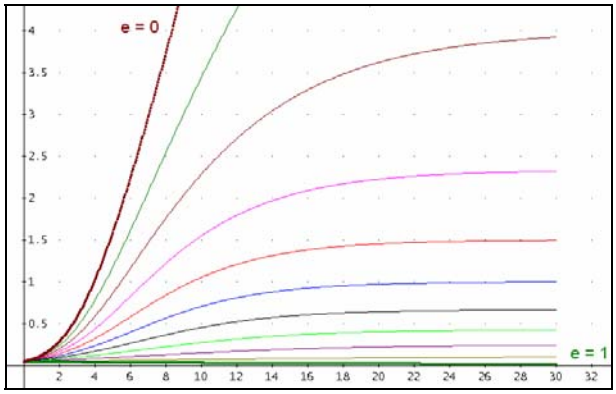
Die erste Grafik zeigt den Bestand bei veränderlicher Wachstumsrate r für $0 \le r \le 1$ – mit einer Vergrößerung des Bereichs der ersten Jahre ($e = 0{,}5$ und $c = 1$).





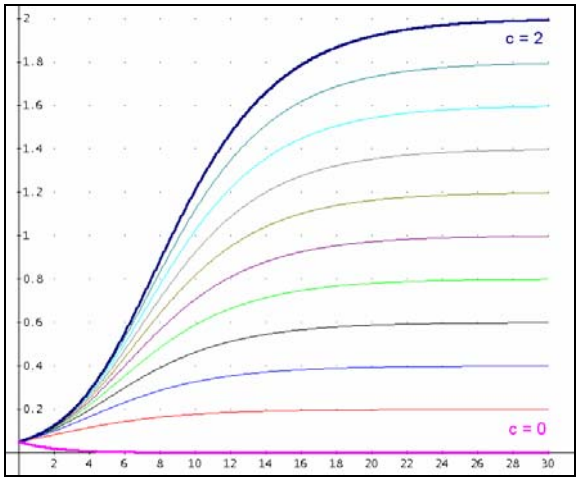
Jetzt variiere ich die Abnahmerate e von 0 bis 1 ($r = c = 1$):

```
VECTOR(MMK(1, e, 1, 0.05, 30), e, 0, 1, 0.1)
```



Nun fehlt noch die Untersuchung des Einflusses der Halbsättigungskonstante c mit $0 \le c \le 2$ ($r = 1$, $e = 0{,}5$).

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



Jetzt versuche ich noch aus „Jux und Tollerei“ die Differentialgleichung exakt zu lösen – und mache dies mit *WIRIS*[9].

$$\text{solve}\left(y'(x)=r\cdot y(x)\cdot\left(1-\frac{y(x)}{y(x)+sk}\right)-e\cdot y(x)\right)$$

$$\rightarrow \left\{\left\{-\frac{\ln(y(x))}{e-r}+\frac{r\cdot\ln(e\cdot y(x)+(e\cdot sk-r\cdot sk))}{e^2-e\cdot r}=c+x\right\}\right\}$$

$$\text{solve}\left(y'(x)=1\cdot y(x)\cdot\left(1-\frac{y(x)}{y(x)+1}\right)-0.5\cdot y(x),y(0)=0.02\right)$$

$$\rightarrow \{\{4.\cdot\ln(-6.\cdot y(x)+6.)-2.\cdot\ln(y(x))+(x-14.91)=0\}\}$$

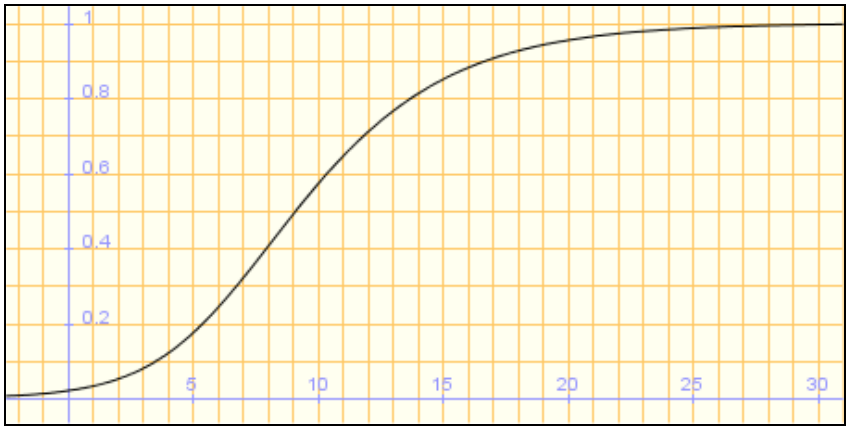
$$mm:=4.\cdot\ln(-6.\cdot y(x)+6.)-2.\cdot\ln(y(x))+(x-14.91)=0;$$

$$T1=\text{plotter}(\text{point}(20,1),45,3);$$

$$\text{plot}(T1,mm)\rightarrow \text{plotter1}$$

Dabei ist c die Integrationskonstante.

Die letzte Programmzeile erzeugt den Graph der gefundenen Integralkurve.



Da geht offensichtlich was, daher werde ich das auch mit *DERIVE* durchführen. Vielleicht komme ich so doch noch zu meinen heiß geliebten Schiebereglern?

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

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

Damit habe ich einmal die allgemeine Lösung in impliziter Darstellung.
Dann rechne ich die Integrationskonstante k heraus und setze $B(t = 0) = i$ ein:

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

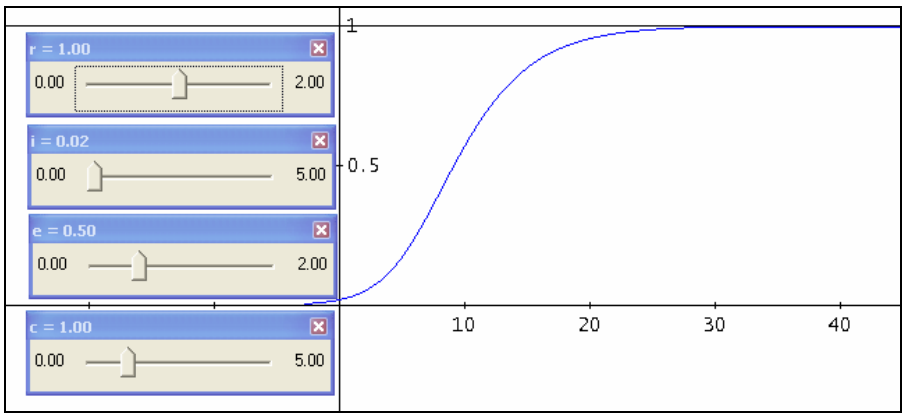
$$\frac{r\cdot \text{LN}(B\cdot e + c\cdot(e - r))}{e\cdot(e - r)} + \frac{\text{LN}(B)}{r - e} - t = \frac{r\cdot \text{LN}(c\cdot(e - r) + e\cdot i)}{e\cdot(e - r)} + \frac{\text{LN}(i)}{r - e}$$

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

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

Die vorletzte Zeile ist die Lösung (in impliziter Form) und der letzte Ausdruck ist der Gleichgewichtszustand.

Ich führe für r, c, e und i Schieberegler ein und zeichne Funktion und Gleichgewichtsgerade:



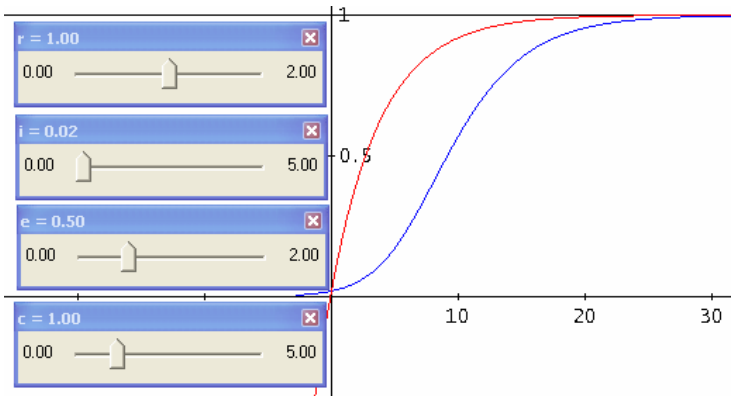
Der Aufwand – und er war ja gar nicht so groß - hat sich gelohnt!

Das war ja sehr schön. Aber dann kam ich auf recht merkwürdige Ergebnisse. Ich versuchte die Lösung in expliziter Form zu finden:

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

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

Und ich kam auch zu einem Ergebnis – aber der Graph dieser Funktion stimmt nicht mit dem Graph der impliziten Lösung überein.



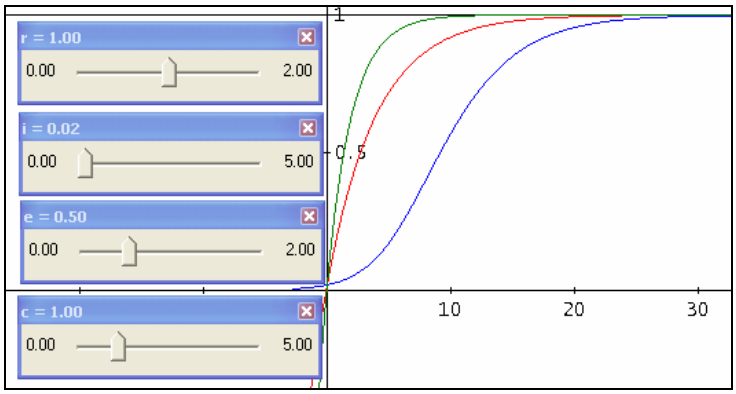
Ich führte meine Versuche – mit Papier und Bleistift – fort und brachte die implizite Darstellung in eine „schönere“ Form und löste wieder nach *y* auf.

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

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

Während sich die implizite Form noch mit der Lösung von oben deckt, zeigt die explizite Form wieder eine – nochmals verschiedene – Form.

Wenn ich allerdings in die implizite Form die Parameter einsetze und dann nach *y* auflöse …



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

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

… dann gibt es zwei Lösungen. Der Graph einer der beiden Lösungsfunktionen deckt sich genau mit dem Graph der impliziten Form.

Diese Gleichung lässt sich allerdings nur für besondere Werte für *r* und *e* lösen.

Ich weiß nicht, warum hier so viele Widersprüchlichkeiten auftreten. Vielleicht kann ein interessierter Leser dieser Zeilen weiter helfen.

Weitere schöne Beschreibungen und Erklärungen zum Miachaelis-Menten-Wachstum finden sich in:

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

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

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

---

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

7 Was ist ein Brusselator?

Ja, was ist nun wirklich ein *Brusselator*? Da mir dieser Begriff noch nie untergekommen ist, habe ich ein wenig im Internet recherchiert.

In [10] habe ich gefunden, dass dies eine „einfache Simulation einer chemischen Reaktion mit oszillierender Dynamik“ ist. Ob mich das viel klüger machte?

Eine schöne Beschreibung findet sich in [11].

Und dann habe ich auch noch den Ursprung des Namens *Brusselator* ausfindig gemacht:

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)

Der Brusselator wird beschrieben durch ein Differentialgleichungssystem:

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

Dabei sind A und B konstante vorgegebene Konzentrationen und x und y Zwischenprodukte, deren Verhalten bei der chemischen Reaktion untersucht bzw. simuliert wird.

Dieses Mal will ich als erstes das Differentialgleichungssystem numerisch lösen.

Die Lösung mit *WIRIS*

```
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
```

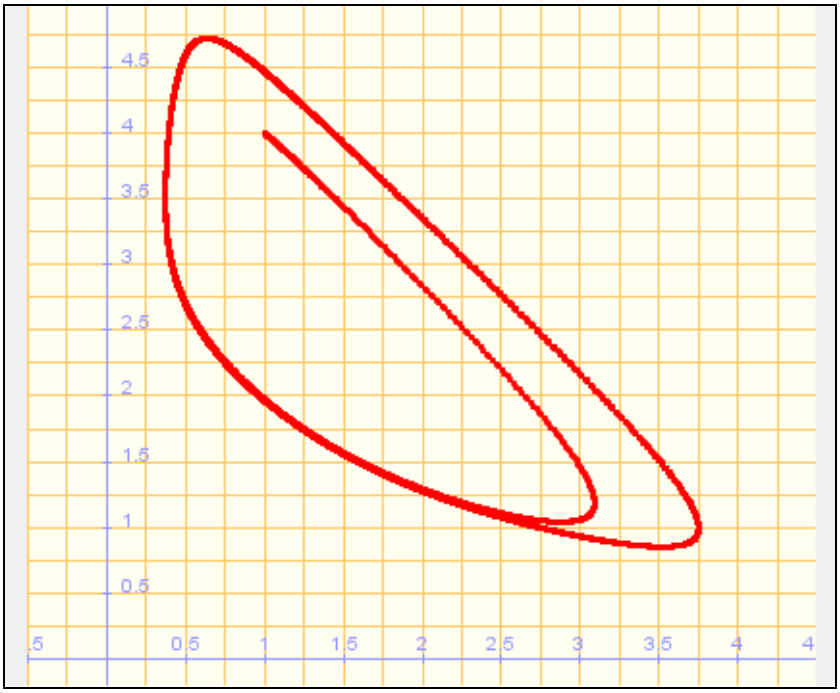
In der ersten – nicht aufgeklappten –„library“ ist mein Programm zur Runge-Kutta-Methode zu finden.

WIRIS hat diesen numerischen Algorithmus – im Gegensatz zu *DERIVE* – nicht standardmäßig implementiert.
In der zweiten „library“ habe ich die beiden ersten Ableitungen definiert und in der dritten die Parameterwerte, die auch *Bossel* zum Start verwendet. Die Inhalte der „libraries“ sind global in der ganzen *WIRIS*-Sitzung zu verwenden.

Ich habe es nur bis zu 6000 Punkten mit einer Schrittweite von 0,004 geschafft. Bei mehr Punkten gibt es keinen Plot mehr. Wir werden dann sehen, ob *DERIVE* mehr vermag?

Im ersten Plot sieht man die Entwicklung von *y* für die ersten 24 Zeiteinheiten (Sekunden).

Der zweite Plot zeigt das Phasendiagramm zwischen den *x*- und *y*-Werten.

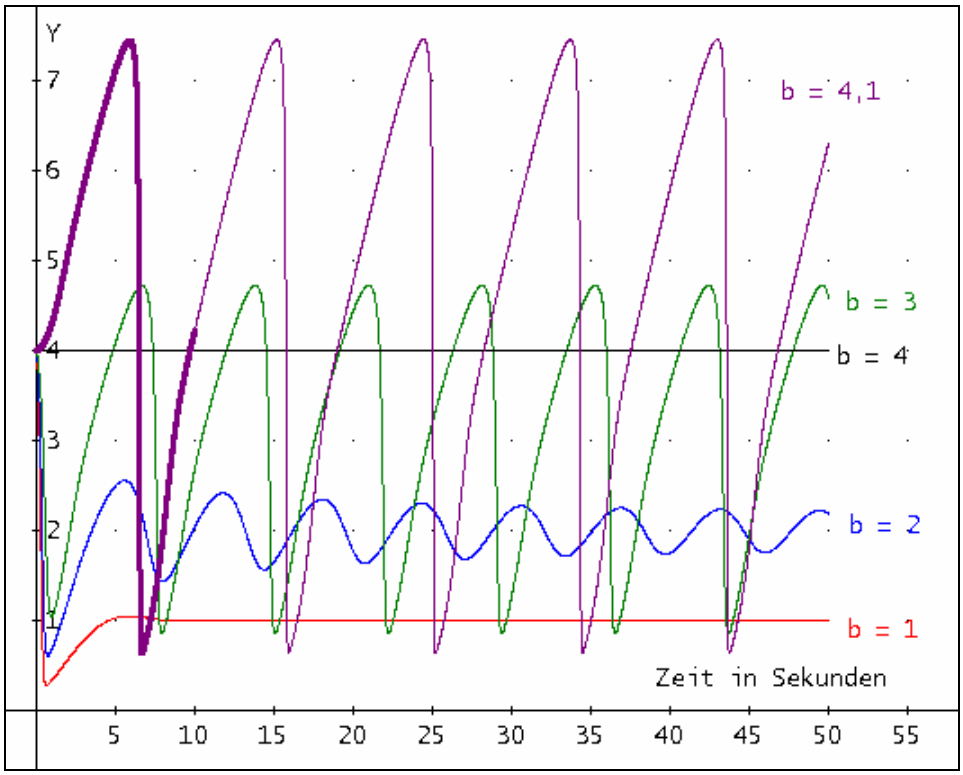


Die Bearbeitung mit *DERIVE* und dann mit *TI-NspireCAS*

Ich verwende meine eigene Runge-Kutta-Routine rk4 und zeichne für verschiedene Werte für B das t-Y-Diagramm. (rk4 arbeitet hier rascher als 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]
```

Der Ausdruck #3 wird aufgerufen (gilt für $B = 1$), dann definiere ich $B = 2$, lasse #3 wieder zeichnen, u.s.w. Zum Unterschied von *Bossel* verwende ich eine Schrittweite von 0,01. *Bossel* arbeitet mit *VENSIM* mit einer Schrittweite von 0,002.



Ich habe dann im letzten Term eine Schrittweite von 0,001 eingesetzt, um die Genauigkeit zu vergleichen. Wie man deutlich sieht, weicht die starke Linie ($h = 0{,}001$) praktisch nicht von der Kurve ab, die mit $h = 0{,}01$ erzeugt wurde.
Die *VENSIM*-Bilder (später in diesem Abschnitt) haben das gleiche Aussehen!

Wir erkennen, dass sich die Kurve von einer gedämpften Schwingung zu deutlichen periodischen Oszillationen verändert. Für $B = 4$ bleibt $y = 4$ konstant.

Als nächstes zeichnen wir mit den gleichen Einstellungen die Phasendiagramme.

#9: b := 1

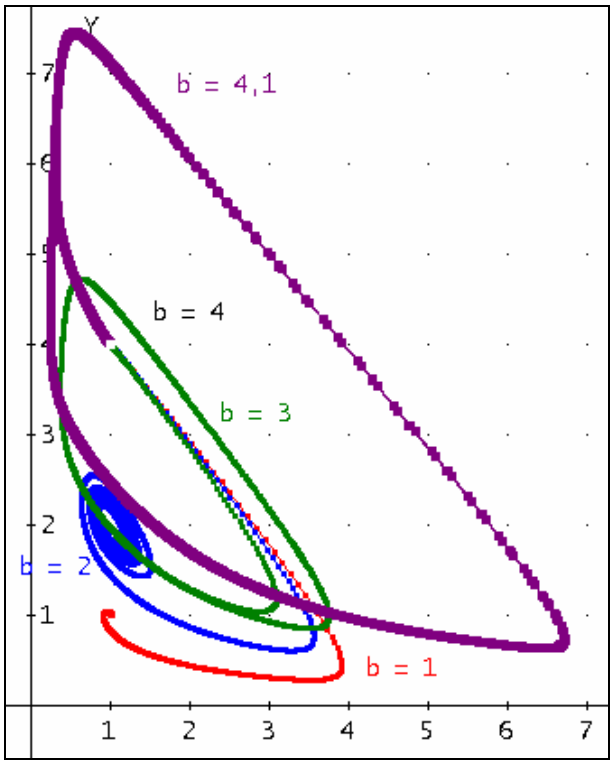
#10: (RK([a - (b + 1)·x + x^2·y, b·x - x^2·y], [t, x, y], [0, x_, y_], 0.01, 5000))↓↓[2, 3]

#11: b := 2

#12: b := 3

#13: b := 4

#14: b := 4.1



Um den Punkt für $B = 4$ hervorzuheben (1,4) habe ich ihn weiß eingezeichnet.

Bossel erzeugt dann mit *VENSIM PLE* ein wenig aufwändiger noch die Herstellung von „Zustandsbildern“, in denen er die Anfangswerte für x und y einen ganzen Bereich durchlaufen lässt und alle Phasendiagramme gemeinsam darstellt.

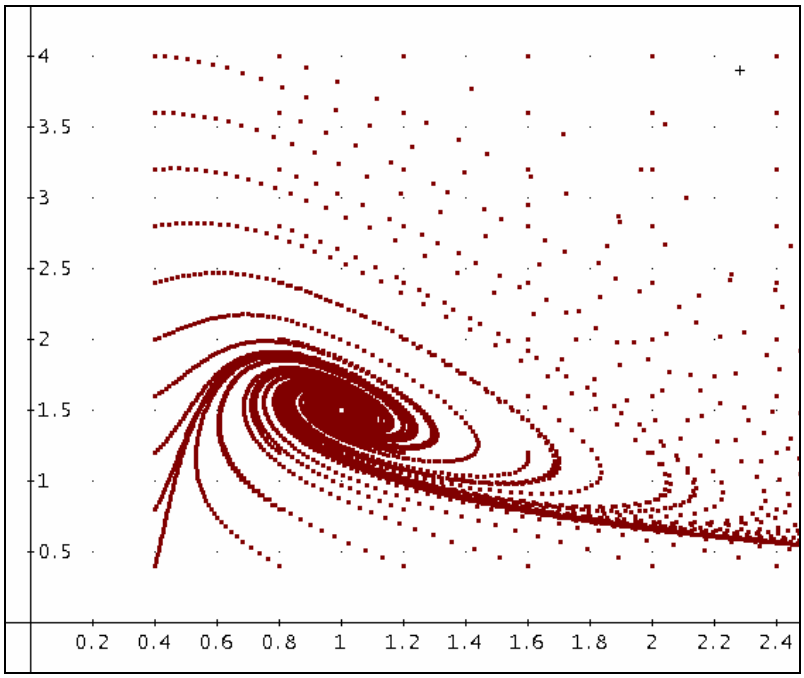
Wir können das mit *DERIVE* mit einer geschachtelten VECTOR-Anweisung – mit ein wenig Rechenzeit auch erledigen lassen ($0{,}4 \le x, y \le 4$).

#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]

Die Punkte werden hier vorerst nicht verbunden dargestellt.

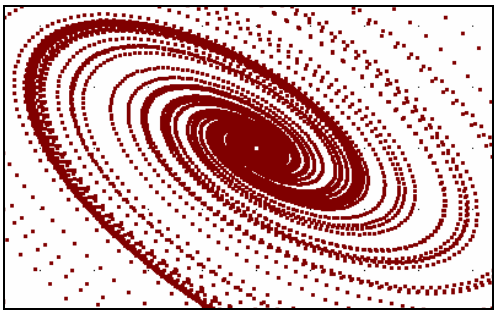


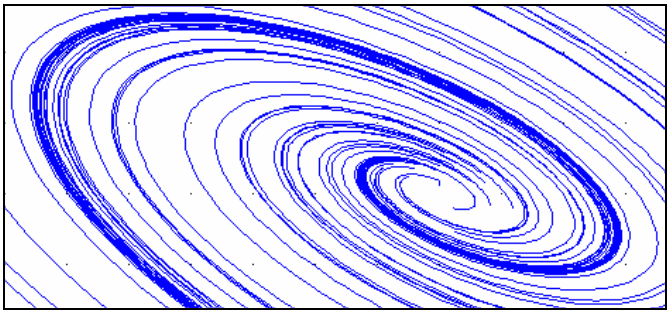
Der Gleichgewichtpunkt ergibt sich indem man im Differentialgleichungssystem die Ableitungen gleich Null setzt und das verbleibende System nach x und y auflöst:

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

$$0 = B \cdot x - x^2 \cdot y$$

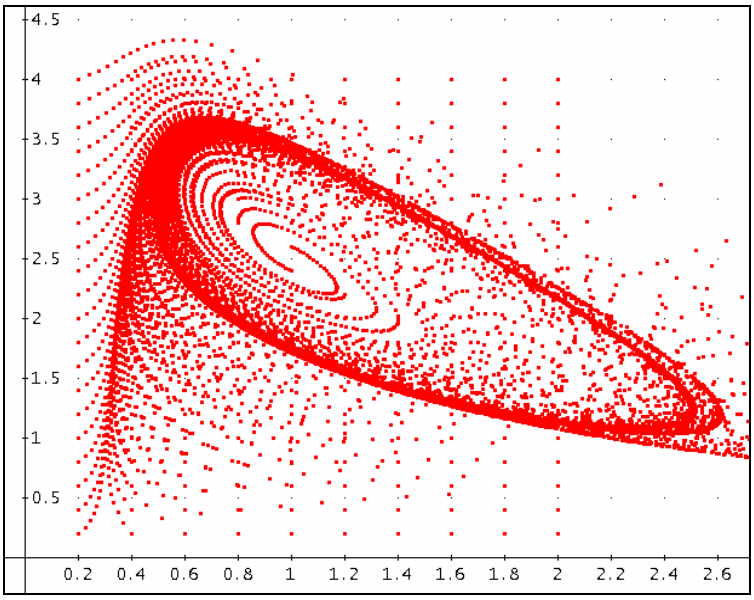
Auch ohne CAS findet man hier bequem den Gleichgewichtspunkt mit $\left(A, \frac{B}{A} \right)$. Nach *Bossel* lässt sich auch leicht zeigen, dass für $B < A^2 + 1$ dieser Punkt ein stabiler Strudel ist, wie er im obigen Zustandsbild deutlich zu sehen ist. (Ich habe den Gleichgewichtspunkt (1|1,5) als weißen Punkt aufgesetzt.) Die nächsten beiden Bilder zoomen in das Innenleben des Strudels, wobei das rechte Bild die Kurven zeigt.





Für $B > A^2 + 1$ entsteht ein Grenzzyklus, der in seinem Innersten einen instabilen Gleichgewichtspunkt enthält, hier (1|2,5) für $A = 1$, $B = 2{,}5$.

```
#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)
```



Instabiler Gleichgewichtspunkt in (1|2,5)

Diesen Zustandsbildern kann ein großer ästhetischer Reiz nicht abgesprochen werden.

Wir werden später sehen, dass diese auch mit *VENSIM* – mit etwas Aufwand – hergestellt werden können.

Natürlich wäre auch hier der Einsatz von Schiebereglern von Vorteil, ließen sich doch die Einflüsse aller Parameter bequem untersuchen.

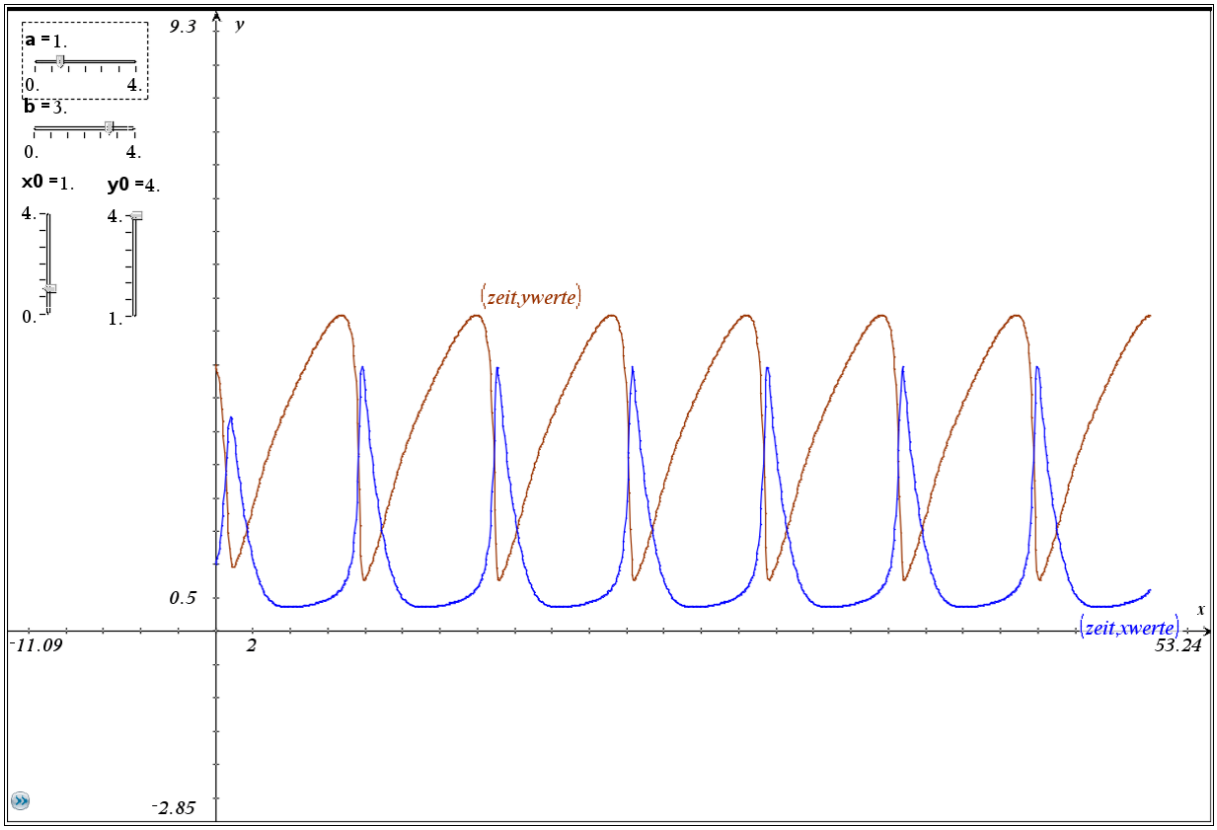
Schieberegler mit *TI-NspireCAS*:

	B	C zeit	D xwerte	E ywerte	F	G
◆						
1	lt 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		

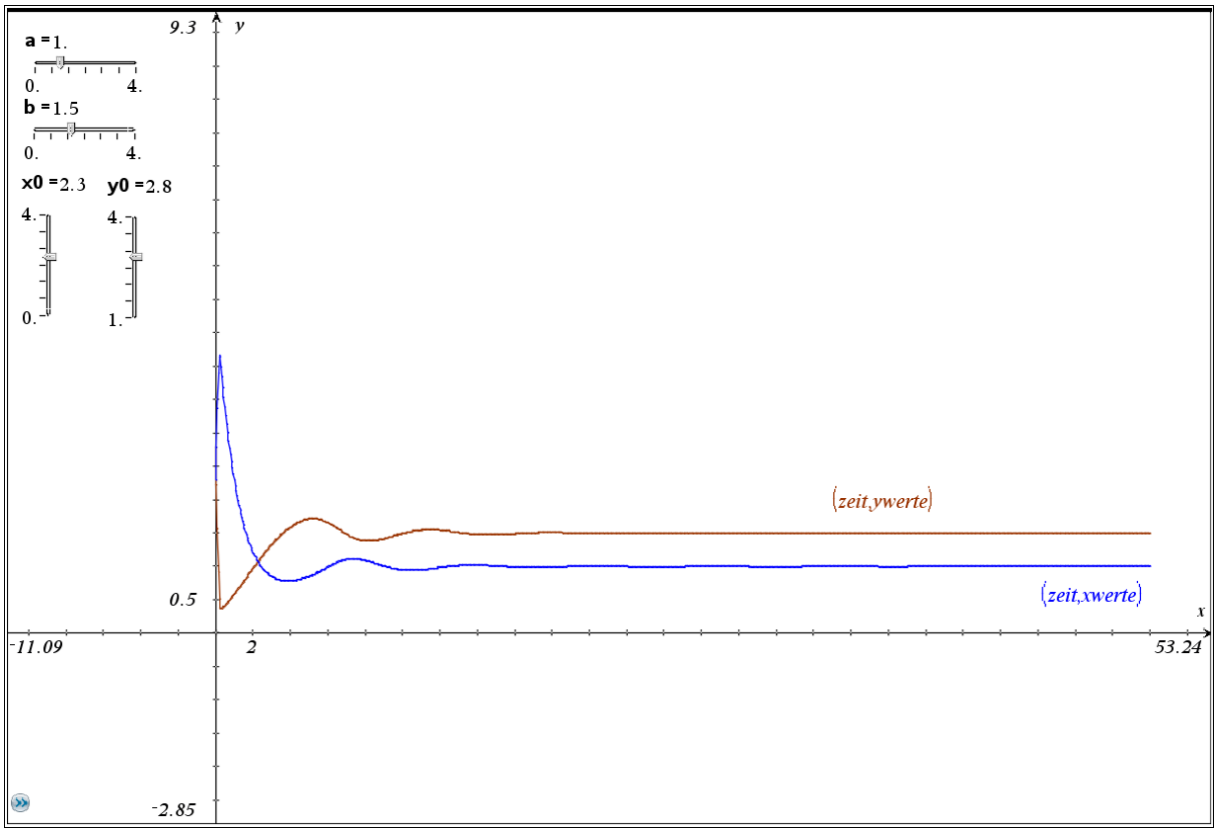
D3 $=d2+\$b\$1\cdot\left(a-\left(b+1\right)\cdot d2+d2^2\cdot e2\right)$

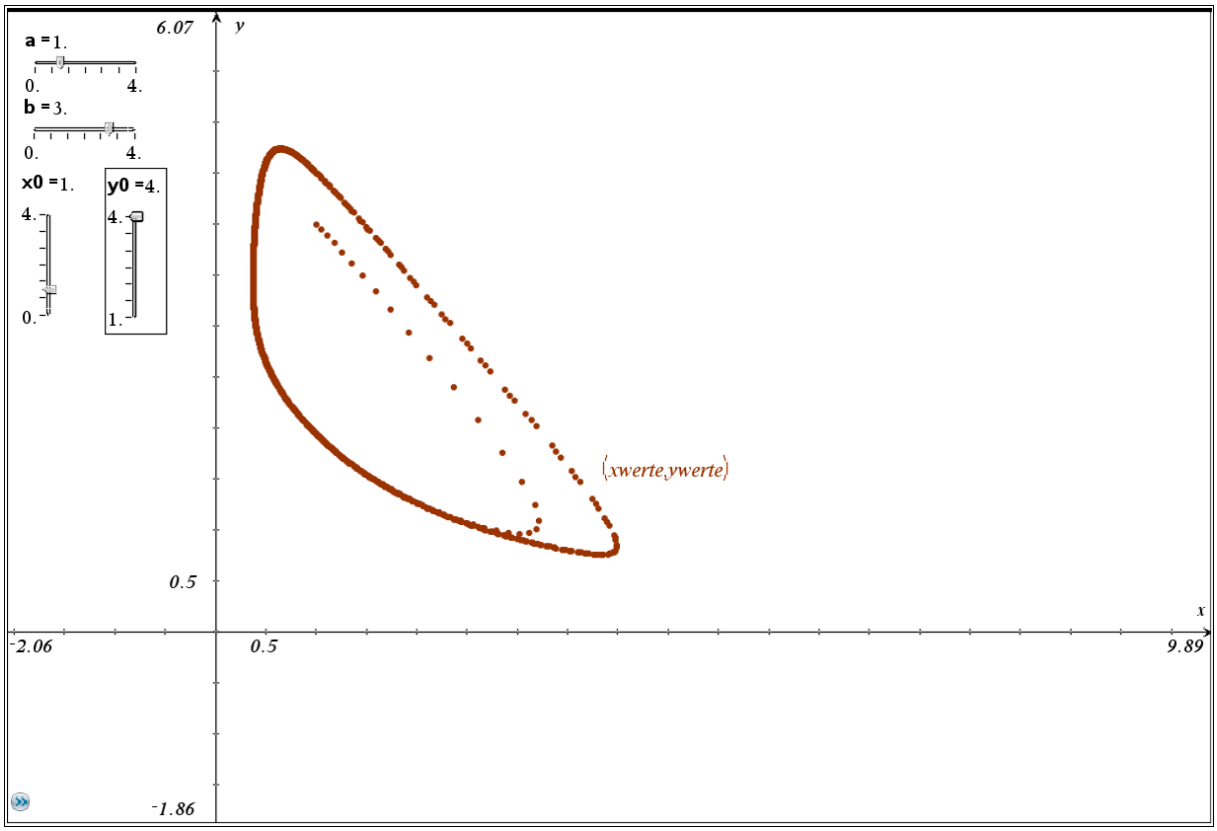
Die Tabelle ist rasch erstellt. Für *a*, *b*, *x0* und *y*0 wurden vorher im Zeichenblatt Schieberegler eingeführt.

Die erste Grafik zeigt die Entwicklung von *x* und *y* während der ersten 50 Sekunden für die von *Bossel* gewählten Werte für die Systemparameter.

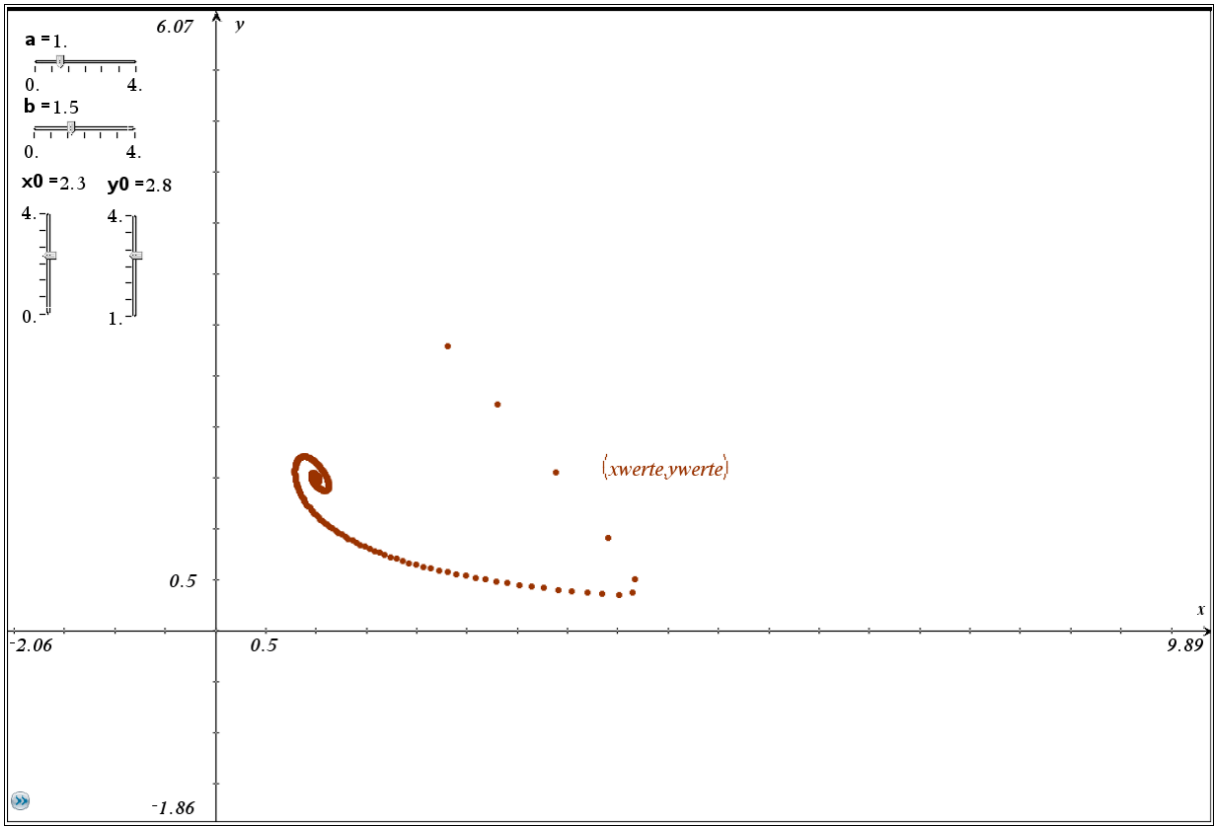


Die Grafik reagiert unmittelbar auf jede Veränderung der Einstellung der Schieberegler.





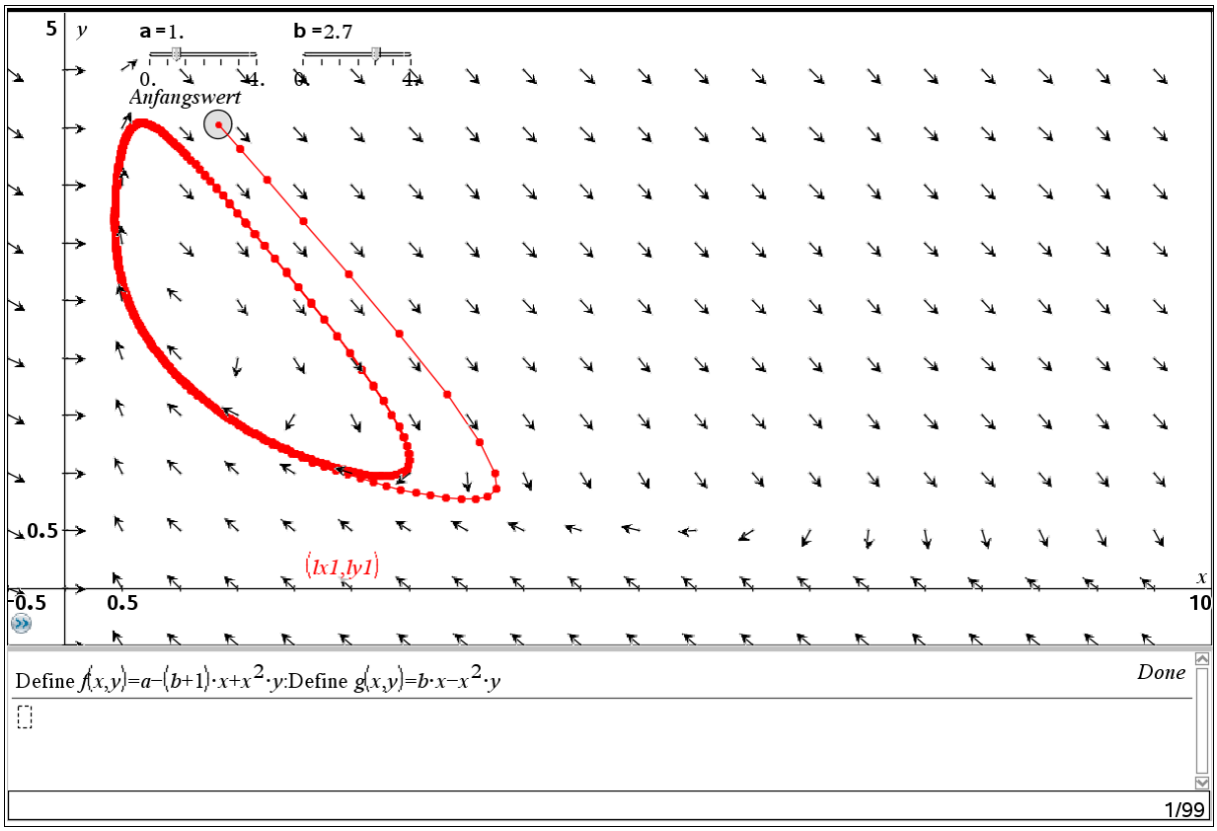
Das Phasendiagramm für die „*Systemzoo-Werte*“ und dann noch für eine andere Zusammenstellung der Parameter.



Dank einer Kommunikation mit Philippe Fortin konnte ich noch eine andere Darstellung des Systems erzeugen. Wir benutzten das Programm rk4syst (Runge-Kutta) und anstelle des Einsatzes von von Schiebereglern bewegen wir den Anfangspunkt mit der Maus. Zusätzlich tragen wir auch noch das Richtungsfeld des DGL-systems ein. In einem weiteren Geometriefenster könnten wir die Zeitgraphen darstellen.

	A	B	C	D	E lx1	F ly1	G dt	H	I
◆				=rk4syst(x0,y0,tr	=left(d[],dim((	=right(d[],dir			
1	x0	**1.33621**	Anfangswert x, für t=tmin	1.52154	1.52154	3.82091	0		
2	y0	**4.02749**	Anfangswert für y, für t=tmin	1.76317	1.76317	3.54744	0.05		
3	xmin	**-0.5**	Fenstereinstellung	2.07669	2.07669	3.18825	0.1		
4	xmax	**10**	Fenstereinstellung	2.46829	2.46829	2.73333	0.15		
5	ymin	**-0.5**	Fenstereinstellung	2.91126	2.91126	2.20595	0.2		
6	ymax	**5**	Fenstereinstellung	3.32647	3.32647	1.68447	0.25		
7	xg	**0.5**	erste x–Markierung	3.61813	3.61813	1.26854	0.3		
8	yg	**0.5**	erste y–Markierung	3.75029	3.75029	1.00153	0.35		
9	numpoin...	**800**	Anzahl der mit RK4 ber. Pun...	3.75593	3.75593	0.857797	0.4		
10				3.68574	3.68574	0.791705	0.45		
11	tmin	**0**		3.57695	3.57695	0.768816	0.5		
12	tmax	**40**		3.45087	3.45087	0.769155	0.55		
13				3.31858	3.31858	0.782192	0.6		
14	lv	**0.3**	Länge der Vektoren	3.18567	3.18567	0.802498	0.65		
15				3.05492	3.05492	0.827249	0.7		
16				2.92769	2.92769	0.854936	0.75		
17				2.8046	2.8046	0.884728	0.8		
18				2.68593	2.68593	0.916152	0.85		
19				2.57174	2.57174	0.948925	0.9		
20				2.46197	2.46197	0.982872	0.95		

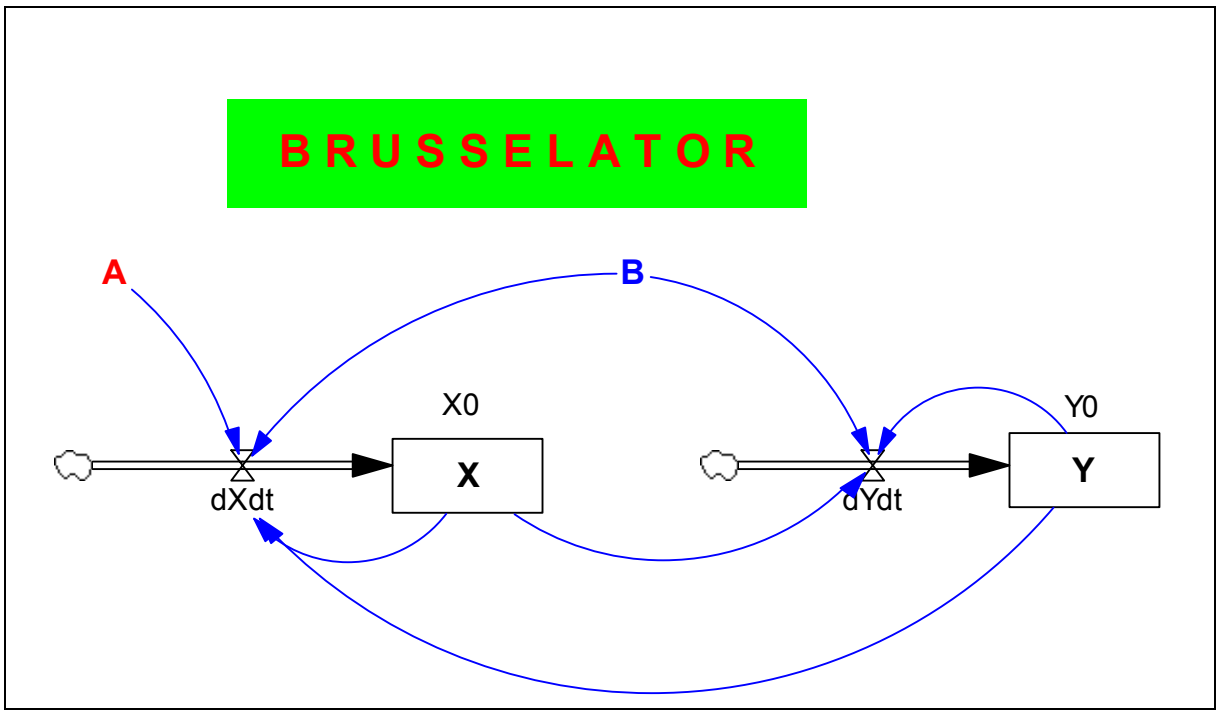
B21



Zum Schluss wollen wir das aber doch auch mit *Vensim PLE* durchführen.

Das Modell ist ja sehr einfach und daher auch rasch hergestellt. (Die Erzeugung der Zustandsbilder ist allerdings ein wenig kniffelig!)

Der Brusselator mit *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

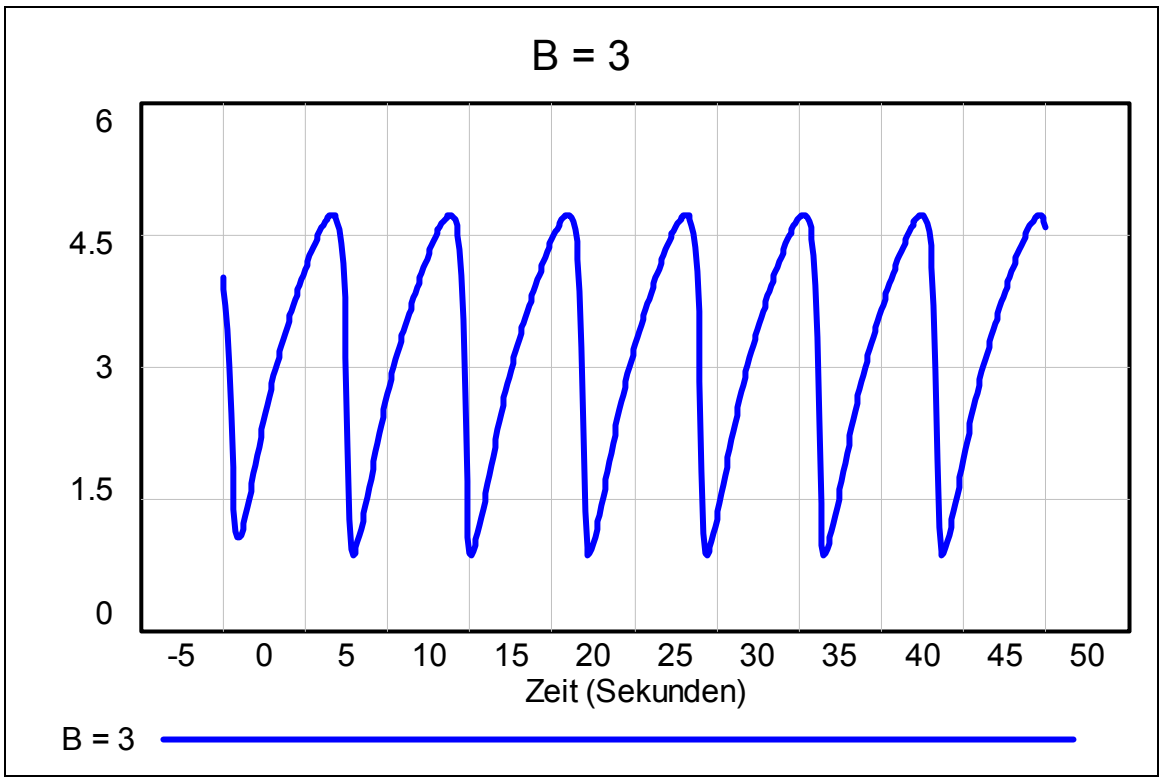
(08) TIME STEP = 0.05

(09) X= INTEG (dXdt, X0)

(10) X0 = 1

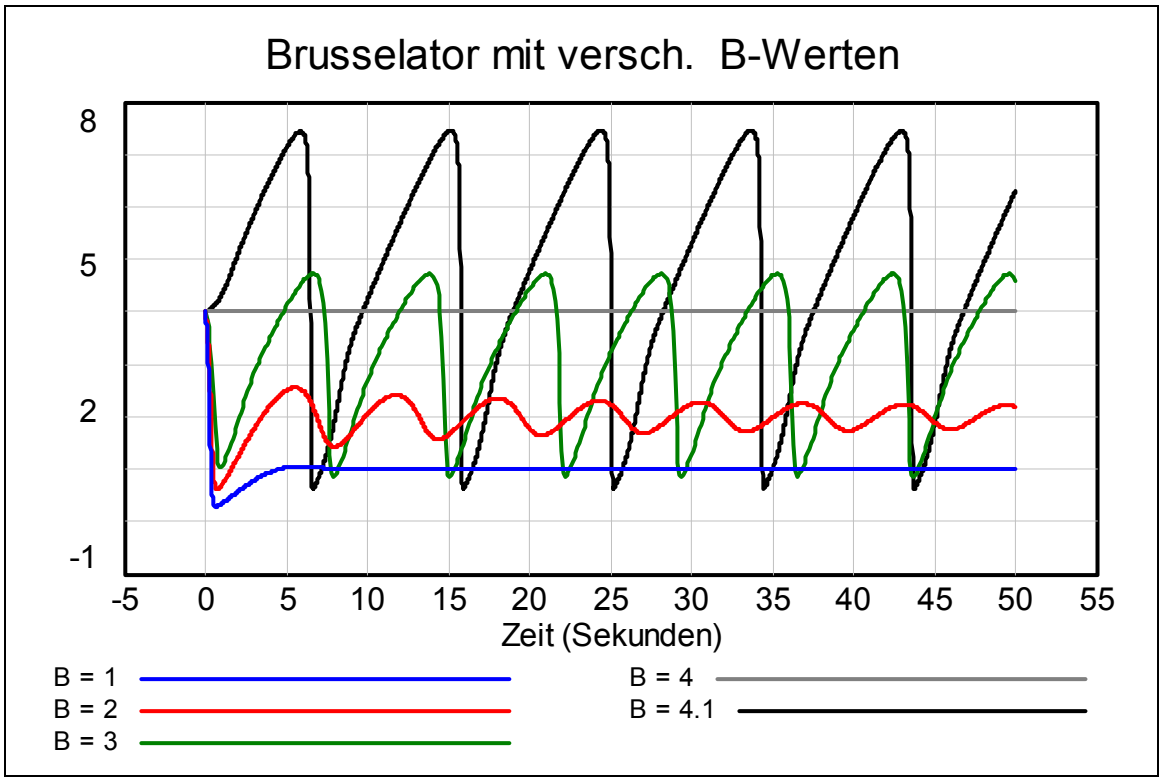
(11) Y = INTEG (dYdt, Y0)

(12) Y0 = 4



Bossel lässt mit dem Euler-Verfahren bei einer Schrittweite von 0,001 rechnen (Abbildung oben). Ich habe die Schrittweite auf 0,05 hinaufgesetzt und praktisch keine Veränderung im Diagramm bemerkt. Damit ist auch meine Vereinfachung in der *TI-Nspire*-Version einigermaßen gerechtfertigt.

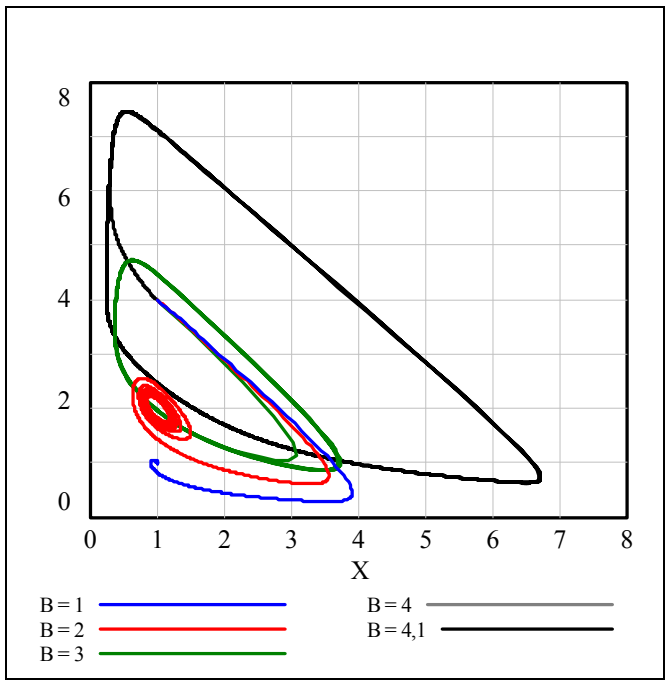
Das nächste Diagramm zeigt den Vergleich für verschiedene Werte für *B*.



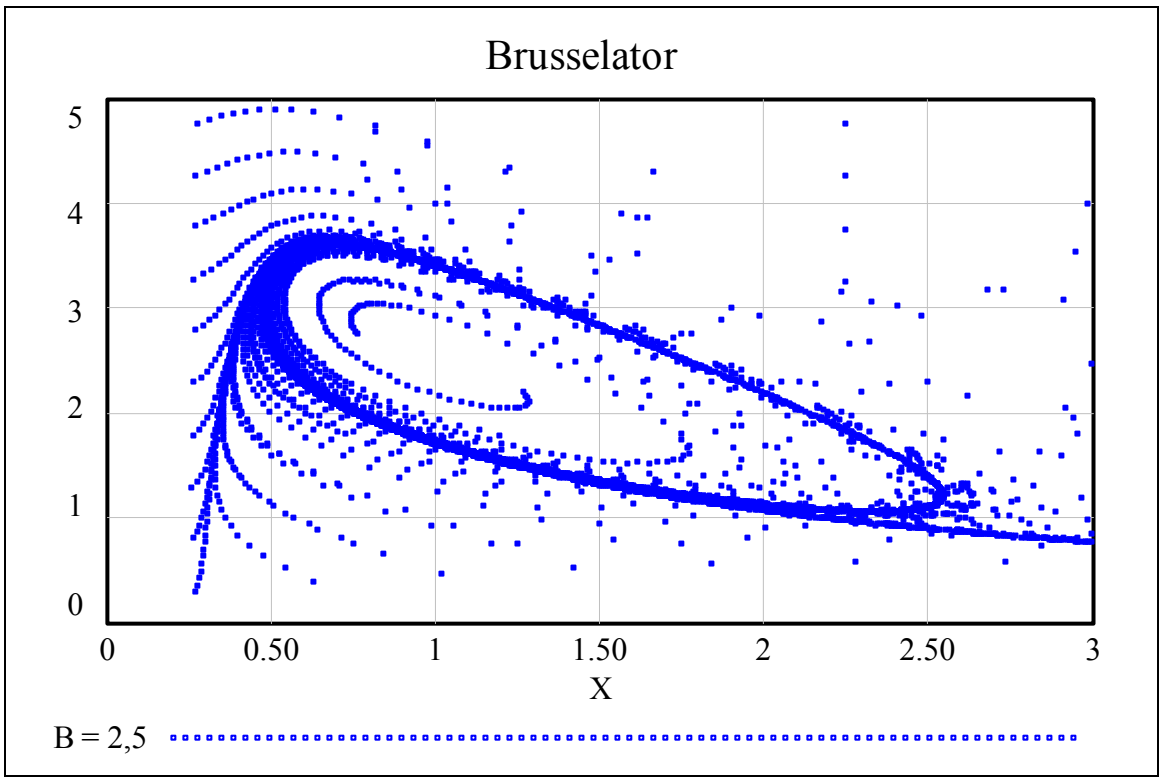
Die rechte Grafik zeigt die Phasendiagramme.

Mit Hilfe eines Tricks (zusätzliches Modul) zaubert *Bossel* auch die doppelte Schleife für die *X*- und *Y*-Werte für einen Raster von 10 mal 10 Anfangszuständen für *X* und *Y*.

Das will ich hier nicht genau erklären, sondern lade ein, in *Systemzoo 1* nachzulesen.



Ein ganz ähnliches Bild wie das nächste (allerdings in rot) sollten Sie weiter oben schon gesehen haben!



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

8 Der bistabile Schwinger

Bevor der „*bistabile Schwinger*“ behandelt wird, will ich kurz den *linearen Schwinger* erläutern, da dieser die Grundform des Schwingers bildet.

So kann ein bistabiler Schwinger als elektroscher Baustein aussehen

Die allgemeine Form des linearen Schwingers wird beschrieben durch das Differentialgleichungssystem

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

Ein bekannter Fall ist die *Federgleichung*:

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

Für die Nichtphysiker – wie mich: x ist die Auslenkung, y ist die Geschwindigkeit und k ist die Federkonstante. Es lässt sich noch ein Dämpfungsparameter d berücksichtigen. Dann lautet das System:

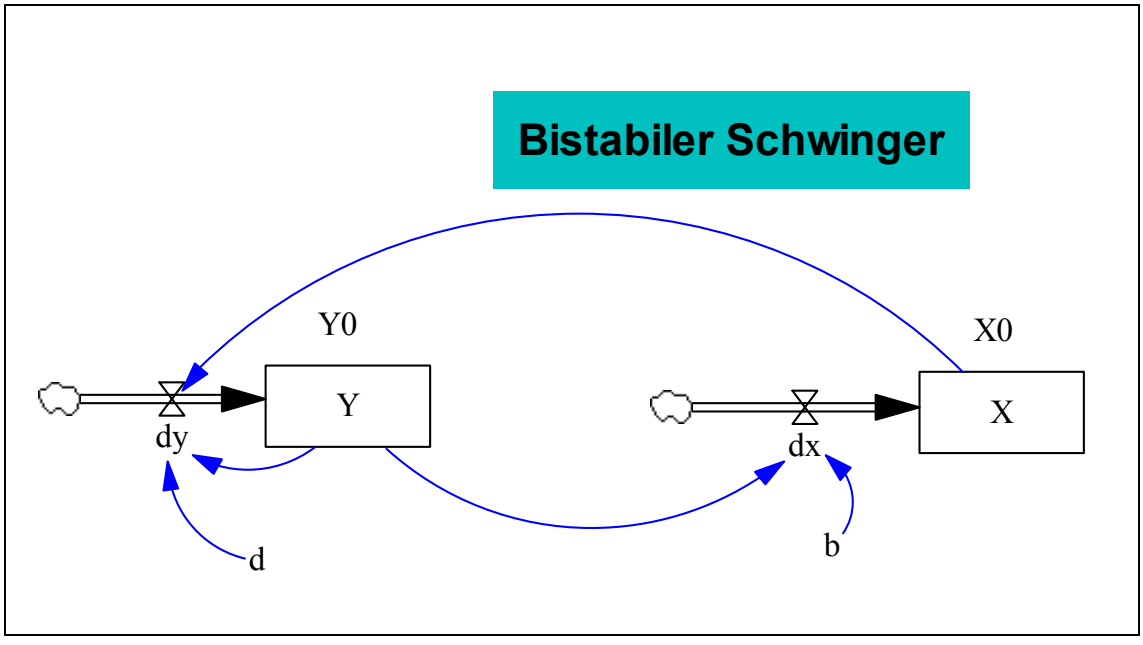
$$\dot{x} = y$$
$$\dot{y} = -k \cdot x - d \cdot y$$

Wenn man nun eine zusätzliche nichtlineare Verkopplung einführt gelangt man z.B. zu

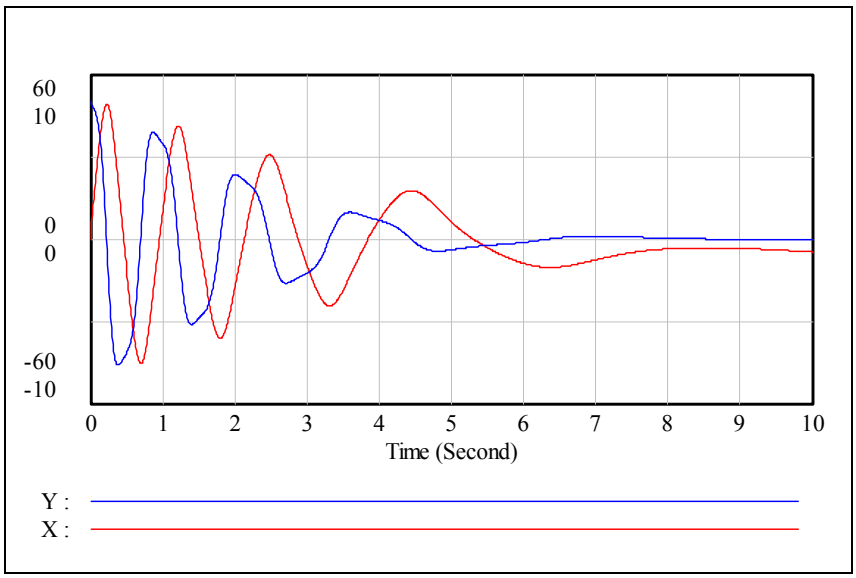
$$\dot{x} = b \cdot y$$
$$\dot{y} = x - x^3 - d \cdot y$$

mit dem Kopplungsparameter b und dem Dämpfungsparameter d. Damit steht bereits unser bistabiler Schwinger. Der Name wird sich später erklären.

Ich beginne mit der Modellierung mit *VENSIM PLE* – mit einer sehr puristischen Bezeichnung der Variablen.



Für die ersten 10 Sekunden zeigt die Grafik das Verhalten der Auslenkung X und der Geschwindigkeit Y bei einer Anfangsgeschwindigkeit $Y0 = 50$



Das beschreibende Dokument ist hier natürlich sehr kurz:

(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

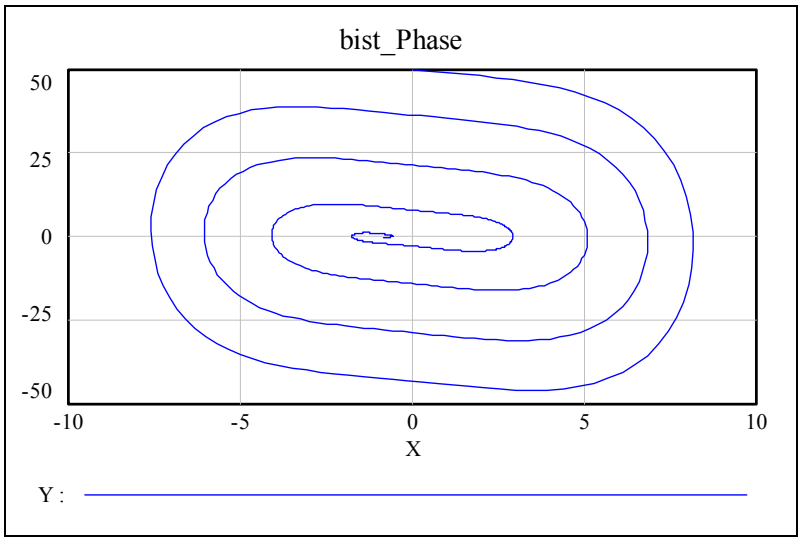
(09) X = INTEG (dx,X0)

(10) X0 = 0

(11) Y = INTEG (dy,Y0)

(12) Y0 = 50

Wir betrachten das Phasendiagramm an. Welchen Eindruck vom das Endverhalten des Schwingers erhalten wir?



Wenn wir den Tabellenausdruck dieses Diagramms betrachten, verstärkt sich unsere Annahme, dass es hier einen Gleichgewichtspunkt bei ($X = -1; Y = 0$) gibt.

bist Phase

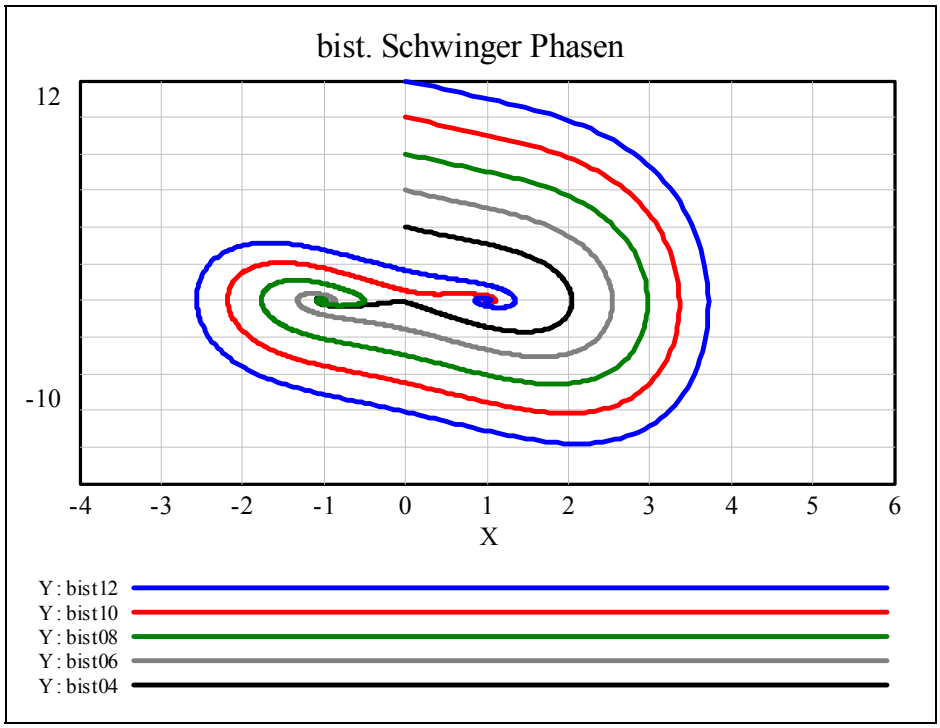
Time (Second)	Y	X
19.7704	-0.0024	-0.9991
19.7804	-0.0024	-0.9991
19.7904	-0.0024	-0.9992

Die analytische Suche nach möglichen Gleichgewichtspunkten gestaltet sich auch unschwierig: Wir lösen das Gleichungssystem

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

nach x und y auf und erhalten drei Lösungen: $x_1 = 0$, $x_2 = 1$ und $x_3 = -1$. Die y-Koordinate ist überall 0.

Das System schwingt in Abhängigkeit von der Anfangsgeschwindigkeit $Y0$ nach (+1,0) oder (–1,0). Das können wir leicht sehen, indem wir Y eine Folge von Werten durchlaufen lassen und alle Phasendiagramme in einer Grafik sammeln.



Das globale Verhalten für -2,50 ≤ X ≤ 2,50 und -2,50 ≤ Y ≤ 2,50 zeige ich später wieder mit *DERIVE* und dessen VECTOR-Befehl.

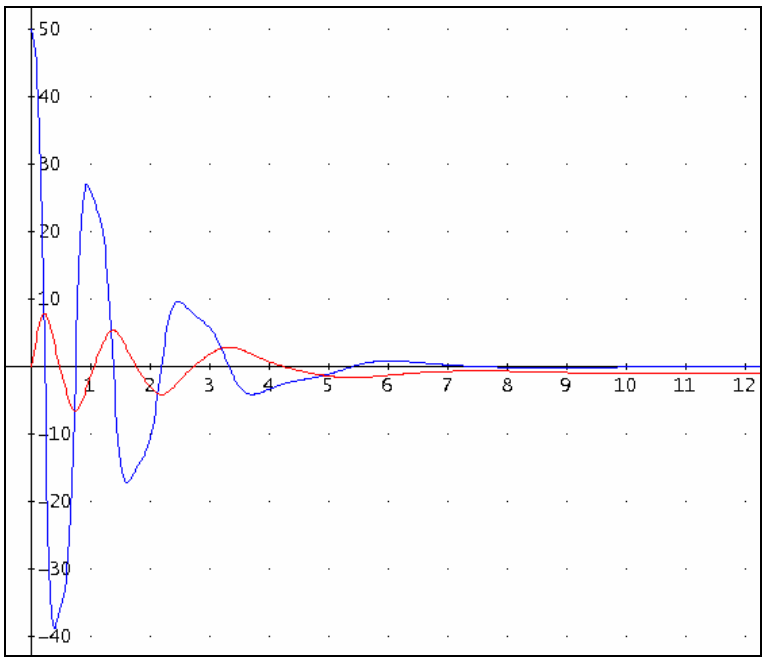
Wenn nun schon von *DERIVE* gesprochen wird, dann wenden wir uns gleich diesem CAS-Programm zu und versuchen wir dort diese nicht zu aufwändige Simulation nachzubauen.

Der bistabile Schwinger mit *DERIVE*

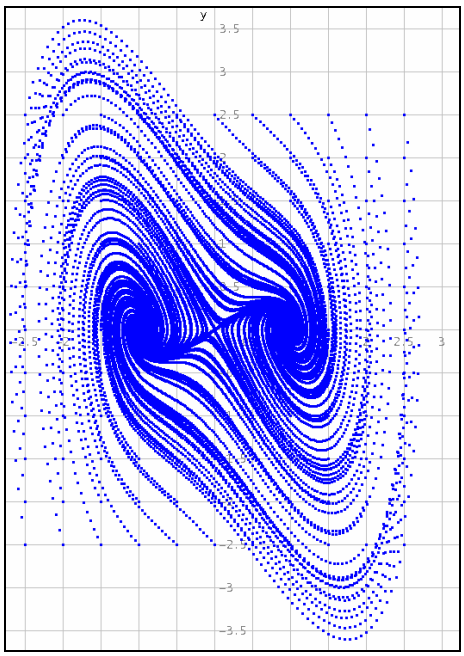
5 Zeilen mit *DERIVE* führen uns zu gelungenen Grafiken:

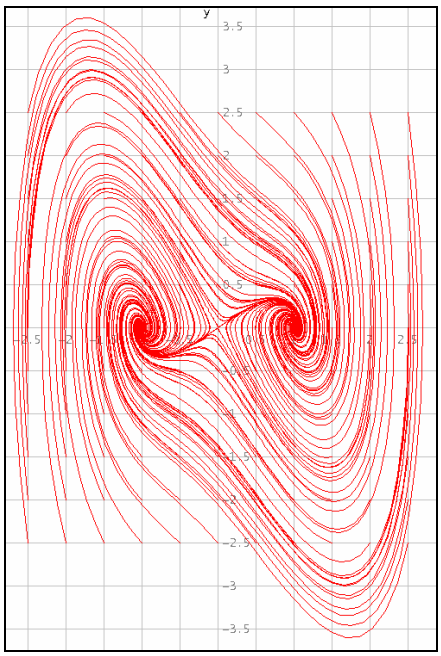
```
bist_Schw(b, d, x0, y0, dt, n) := RK([b·y, x - x^3 - d·y], [t, x, y], [0, x0, y0], dt, n)
(bist_Schw(1, 1, 0, 50, 0.02, 1000))↓↓[1, 3]
(bist_Schw(1, 1, 0, 50, 0.02, 1000))↓↓[1, 2]
VECTOR(VECTOR((bist_Schw(1, 1, x0, y0, 0.02, 500))↓↓[2, 3], x0, -2.5, 2.5, 0.5), y0, -2.5, 2.5, 0.5)
VECTOR(VECTOR((bist_Schw(1,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

In der ersten Zeile definiere ich eine Funktion zur numerischen Lösung des DGLsystems. Zeile 2 und 3 ergeben die zeitliche Entwicklung von *X* und *Y* (in den gleichen Farben wie in der *VENSIM*-Bearbeitung).



Die nächsten beiden Zeilen erzeugen die auf der vorigen Seite angesprochenen Zustandsbilder (siehe Abbildungen unten).





Man kann hier deutlich in den Stellen (± 1,0) *stabile Strudel* und in (0,0) einen instabilen Sattel erkennen (die drei Gleichgewichtspunkte). Natürlich lässt sich auch mit den beiden Parametern *b* und *d* experimentieren. Das wollen wir anschließend mit Schiebereglern durchführen.

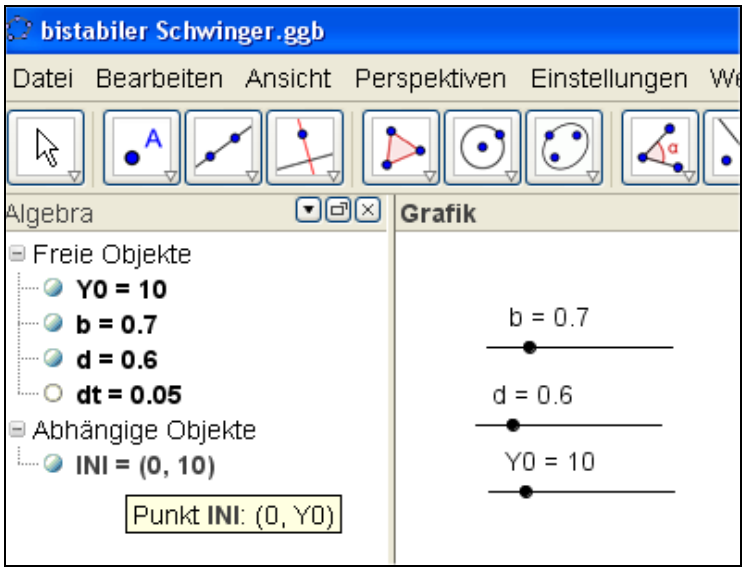
Das *GeoGebra*-Modell

Ich habe schon längere Zeit nicht *GeoGebra* verwendet. Dieses dynamische System ist so einfach, dass es sich auch mit *GeoGebra* (und seinen Schiebereglern) problemlos modellieren lässt.

ch definiere für *b*, *d* und *Y*0 Schieberegler.

Der Punkt INI hat die variablen Koordinaten (0, *Y*0).

In der Tabelle erzeuge ich die entsprechenden Listen.



Um die folgende Tabelle zu erhalten, fülle ich die Zellen der ersten beiden Zeilen wie folgt:

A1:	0	Zeit
D1:	0	X0 (bleibt konstant 0)
E1:	Y0	Startwert für Y (veränderlich über einen Schieberegler)
F1:	(D1,E1)	erster Punkt des Phasendiagramms
G1:	(A1,D1)	erster Punkt des Zeit-X-Diagramms
H1:	(A1,E1)	erster Punkt des Zeit-Y-Diagramms

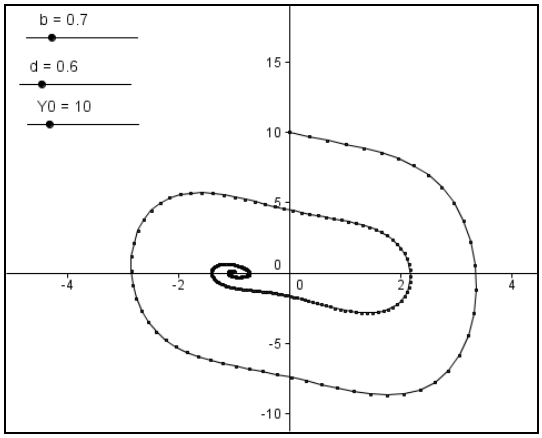
A2:	A + dt	
B2:	dt * b * E1	Zuwachs von X
C2:	dt * (–d * E1 + D1 – D1^3)	Zuwachs von Y
D2:	D1 + B2	neuer X-Wert
E2:	E1 + C2	neuer Y-Wert
F2:	(D2,E2)	
G2:	(A2,D2)	
H2:	(A2,E2)	
I2:	Strecke(F1,F2)	

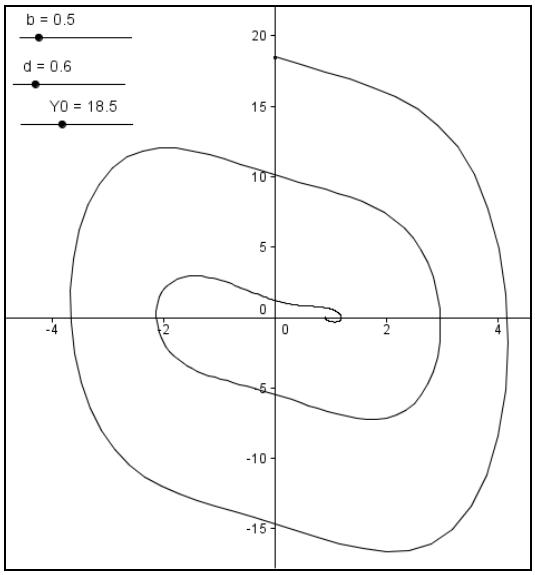
Diese zweite Zeile wird nun zumindest bis in Zeile 201 kopiert.

Tabelle

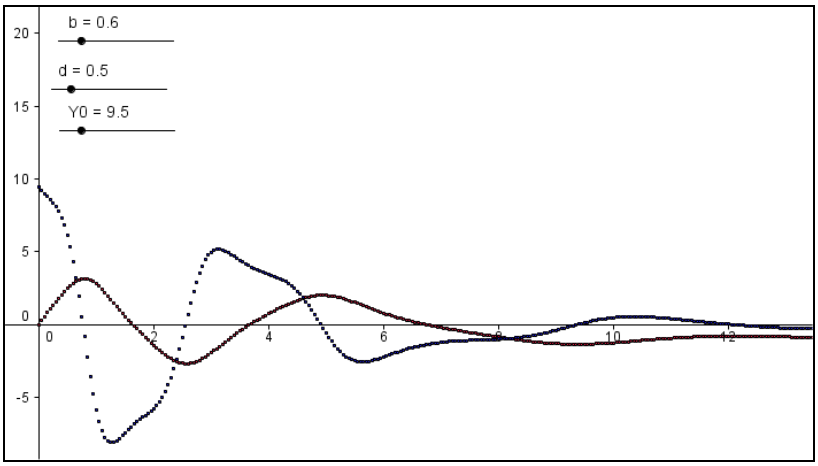
	A	B	C	D	E	F	G	H	I
1	0			0	10	(0, 10)	(0, 0)	(0, 10)	
2	0.05	0.35	-0.3	0.35	9.7	(0.35, 9.7)	(0.05, 0.35)	(0.05, 9.7)	0.461
3	0.1	0.34	-0.276	0.69	9.424	(0.69, 9.424)	(0.1, 0.69)	(0.1, 9.424)	0.437
4	0.15	0.33	-0.265	1.019	9.16	(1.019, 9.16)	(0.15, 1.019)	(0.15, 9.16)	0.423
5	0.2	0.321	-0.277	1.34	8.883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

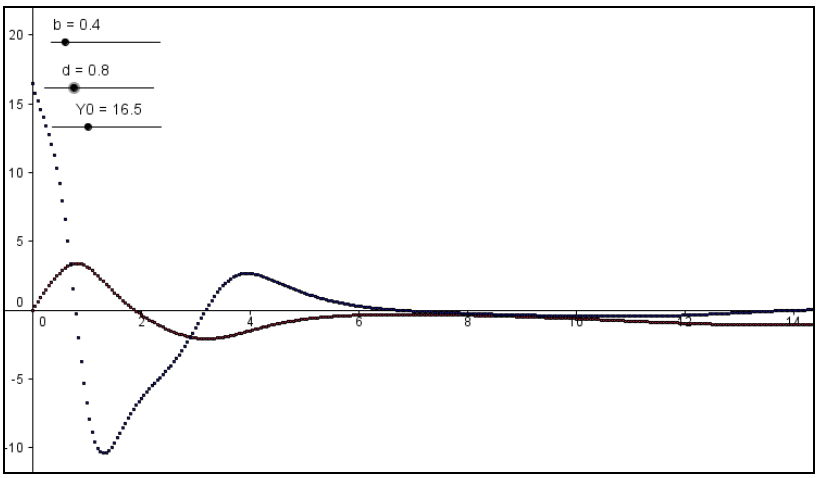
Die nächsten beiden Diagramme geben die Zustände von *X* und *Y* für unterschiedliche Parameterwerte. Man sieht hier auch deutlich die „Bistabilität“. Im rechten Bild werden nur die Verbindungsstrecken (Spalte I) dargestellt.





Im Folgenden sind für zwei Konstellationen die Zeitdiagramme abgebildet. Dem Ursprung entspringt jeweils der *t*-X-Graph.





Mit *MS-Excel* und *TI-Nspire* ist im Grunde genau so zu verfahren, wobei die Einführung der Schieberegler in *Excel* natürlich aufwändiger ist.

Und so wird der Schwinger chaotisch:

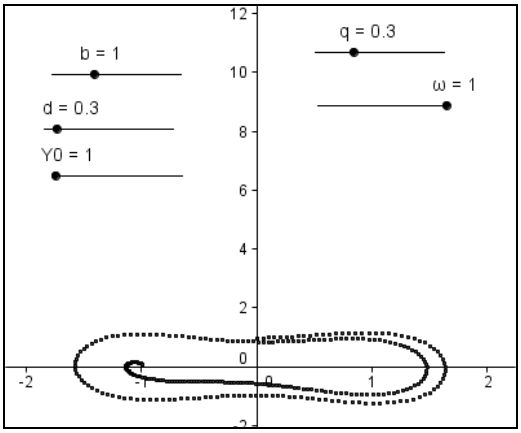
Wenn der Schwinger, der sich bis jetzt ziemlich vorhersehbar verhalten hat von außen durch eine ebenfalls regelmäßige periodische Schwingung zusätzlich angeregt wird, reagiert dieser interessanterweise sehr vielfältig hin bis zu chaotisch.

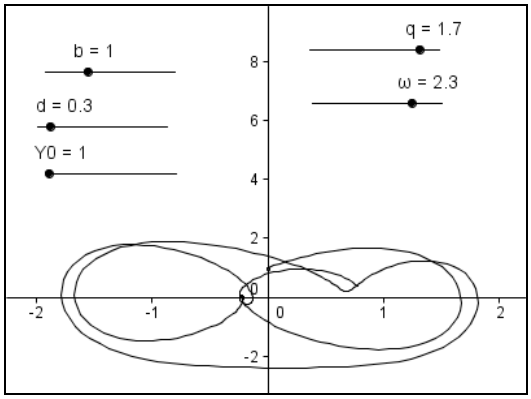
Das Gleichungssystem wird geändert:

$$\dot{x} = b \cdot y$$

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

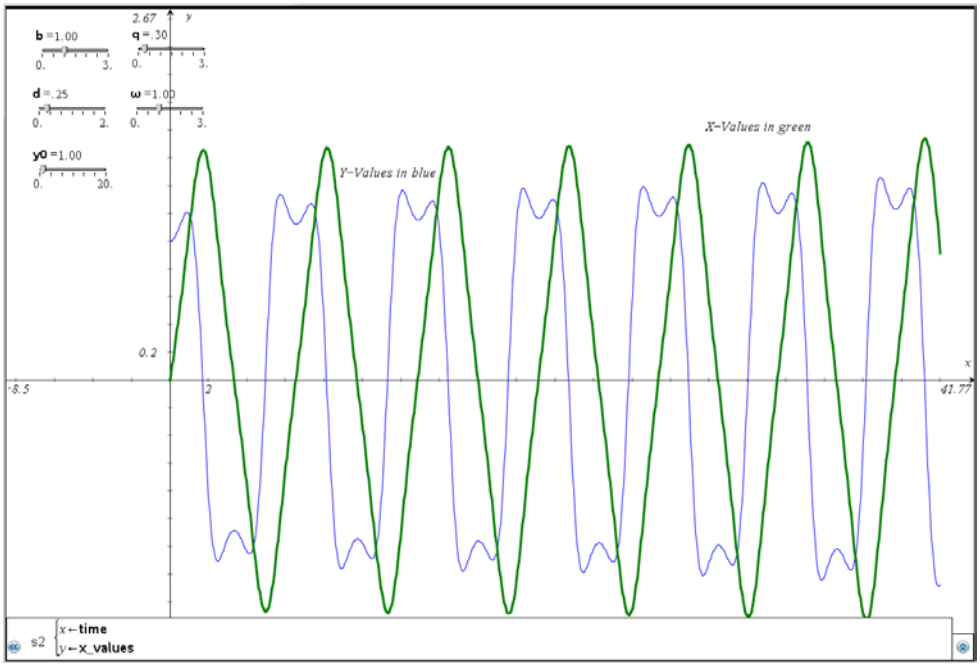
Die momentan verfügbare Version von *GeoGebra* ist für einen größeren Tabellenbereich (> 200 Zeilen) damit überfordert. Mit langen Rechenzeiten lassen sich noch Diagramme erzielen, aber nicht in einer vernünftigen Arbeitszeit. (Zelle C2 ist entsprechend abzuändern.)





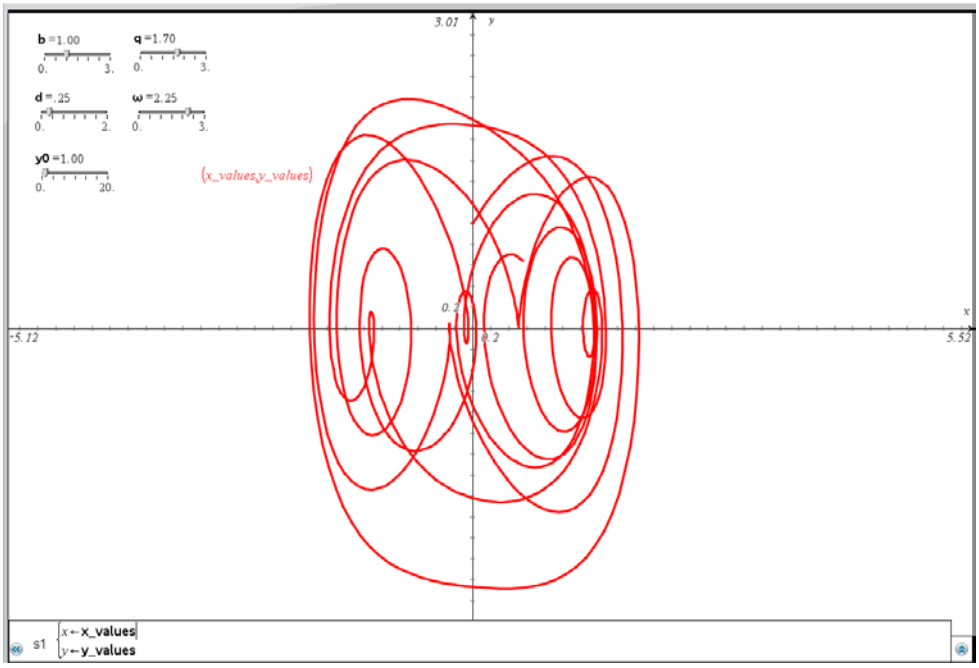
Schieberegler mit *TI-NSpireCAS*

Im Spreadsheet wird genau so verfahren wie bei *GeoGebra*. Die Schrittweite *dt* wurde mit 0,05 vordefiniert.



Entwicklung von *X* und *Y* für die ersten 50 Zeiteinheiten

Das Phasendiagramm sieht schon recht interessant aus.

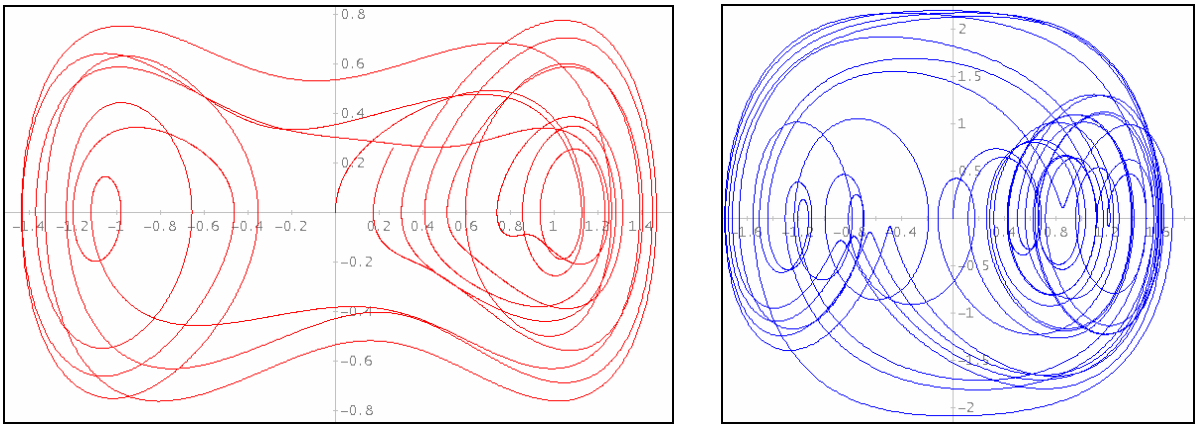


Besonders attraktiv wird die Darstellung, wenn man einen der Schieberegler – egal welchen – animiert. Es läuft dann ein spannendes Filmchen mit sehr unterschiedlichen Diagrammen ab.

Statische Diagramme mit *DERIVE*

Mit *DERIVE* können wir die Runge-Kutta-Methode für das Differentialgleichungssystem einsetzen – aber dafür gibt es keine Schieberegler. Wir genießen aber den Vorteil, dass wir mit einer kleinen Schrittweite $dt = 0{,}01$ in einer vernünftigen Rechenzeit (~ 45 Sekunden) 10 000 Punkte berechnen und darstellen können.

```
(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]
```



Phasendiagramme für die oben angegebenen Parametersätze

9 Lagerhaltung – mit Zufallszahlen

Modellierungen mit Zufallsereignissen sind nicht nur besonders attraktiv sondern auch wirklichkeitsnahe, schwanken doch viele Parameterwerte zufällig innerhalb gewisser Grenzen.

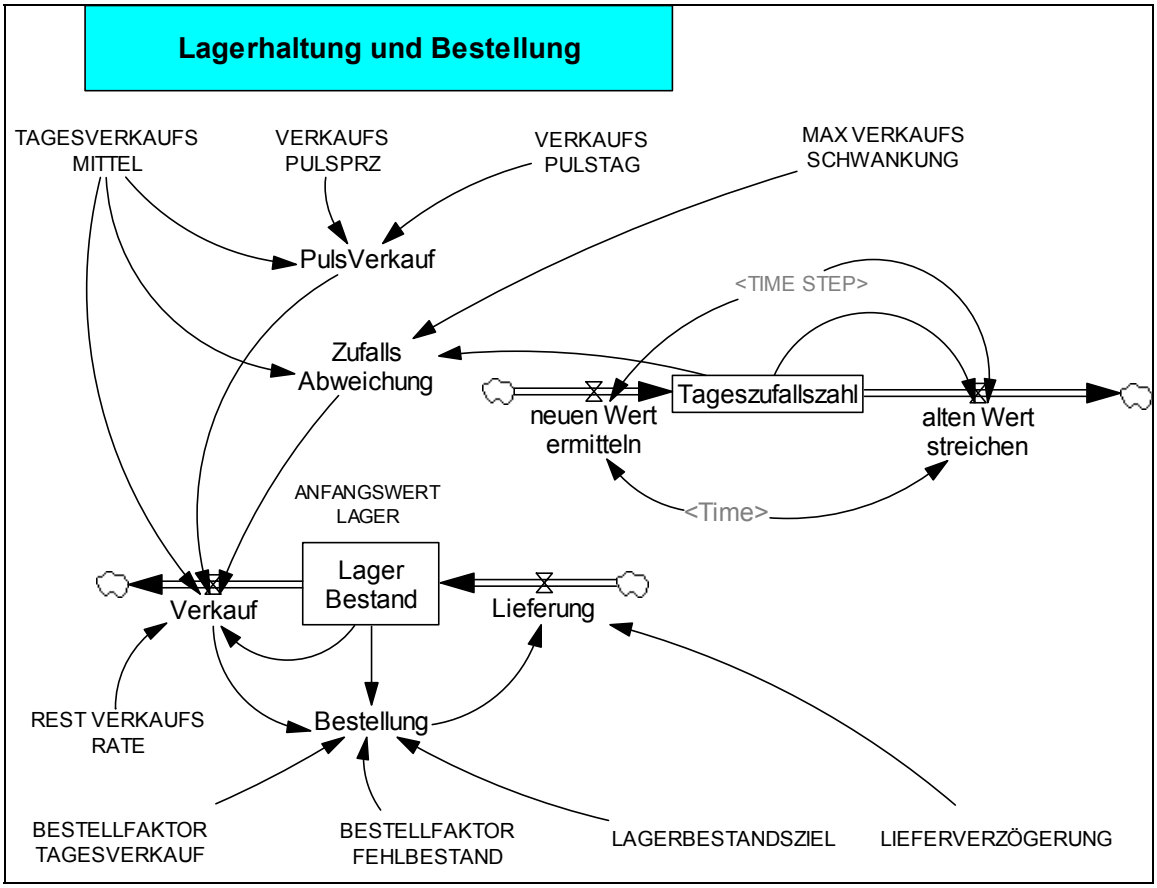
Die Lagerhaltung stellt für jeden Betrieb einen nicht unwesentlichen Kostenfaktor dar. Es muss ein vernünftiger Mittelweg zwischen den Wünschen, einerseits allen Liefererfordernissen nachkommen zu können und andererseits den Lagerbestand möglichst klein zu halten, gefunden werden.

Ich formuliere das Modell nach *Bossel* (aus *Systemzoo* 3):

Der jeweilige *Lagerbestand* ergibt sich aus den zeitabhängigen Größen *Verkauf* und *Lieferung* (gemäß bereits erfolgter *Bestellung*). Die *Bestellung* orientiert sich am *Lagerbestand* sowie am momentanen *Verkauf*. Dabei sind drei Parameter zu berücksichtigen: BESTELLFAKTORTAGESVERKAUF, BESTELLFAKTORFEHLBESTAND und LAGERBESTANDSZIEL. Für die *Lieferung* ist die LIEFERVERZÖGERUNG eine entscheidende Größe.

Der schwankende Verkauf wird bestimmt durch die *Zufallsabweichung* vom TAGESVERKAUFSMITTEL. Dazu wird (über eine Gleichverteilung) eine *TagesZufallszahl* bestimmt. Außerdem wird Prozentsatz des Verkaufsereignisses VERKAUFSPULSPRZ für einen bestimmten Tag dieses Ereignisses VERKAUFSPULSTAG vorgegeben. Damit lässt sich die dadurch initiierte Dynamik im System untersuchen.

Alle weiteren Einflussgrößen sind dem Simulationsdiagramm zu entnehmen.



Ich zeige sofort das Dokument, dem alle Einstellungen und Gleichungen entnommen werden können:

Voreinstellungen

Alte Lagerhäuser in Amsterdam

ANFANGSWERT LAGER = 2000
Units: Stück

BESTELLFAKTORFEHLBESTAND = 0.125
Units: 1/Tag

BESTELLFAKTORTAGESVERKAUF = 1
Units: 1

LAGERBESTANDSZIEL = 2000
Units: Stück

LIEFERVERZÖGERUNG = 20
Units: Tag

MAX VERKAUFS SCHWANKUNG = 25
Units: 1

TAGESVERKAUFSMITTEL = 1000
Units: Stück/Tag

VERKAUFSPULSPRZ = 0
Units: 1
Angabe in Prozent vom mittleren Tagesverkauf

VERKAUFSPULSTAG = 10
Units: Tag

REST VERKAUFS RATE = 1
Units: 1/Tag

Marktleben in Marangu, Tansania

Dynamik

alten Wert streichen = IF THEN ELSE(ABS(Time + TIME STEP/2 –
INTEGER(Time + TIME STEP/2))<=TIME STEP/2,
Tageszufallszahl/TIME STEP, 0)
Units: 1/Tag

Bestellung = IF THEN ELSE((BESTELLFAKTORTAGESVERKAUF * Verkauf +
BESTELLFAKTORFEHLBESTAND * (LAGERBESTANDSZIEL –
LagerBestand)) > 0, (BESTELLFAKTORTAGESVERKAUF * Verkauf +
BESTELLFAKTORFEHLBESTAND * (LAGERBESTANDSZIEL –
LagerBestand)), 0)
Units: Stück/Tag

LagerBestand = INTEG (+Lieferung – Verkauf, ANFANGSWERT LAGER)
Units: Stück

Lieferung = DELAY FIXED(Bestellung, LIEFERVERZÖGERUNG,
TAGESVERKAUFSMITTEL)
Units: Stück/Tag

neuen Wert ermitteln = IF THEN ELSE(ABS(Time – INTEGER(Time))<=TIME STEP/2,
RANDOM UNIFORM(0, 1, 0)/TIME STEP, 0)
Units: 1/Tag

PulsVerkauf = (VERKAUFSPULSPRZ/100) ∗ TAGESVERKAUFSMITTEL ∗
PULSE(VERKAUFSPULSTAG, 1)
Units: Stück/Tag

Tageszufallszahl = INTEG (+neuen Wert ermitteln – alten Wert streichen, 0)
Units: 1

Verkauf = IF THEN ELSE(TAGESVERKAUFSMITTEL + ZufallsAbweichung +
PulsVerkauf < LagerBestand ∗ REST VERKAUFS RATE,
(TAGESVERKAUFSMITTEL + ZufallsAbweichung + PulsVerkauf),
(REST VERKAUFS RATE ∗ LagerBestand))
Units: Stück/Tag

ZufallsAbweichung = TAGESVERKAUFSMITTEL ∗ (2 ∗ MAX VERKAUFS
SCHWANKUNG/100) ∗ (Tageszufallszahl – 1/2)
Units: Stück/Tag

Zeiteinstellungen

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

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

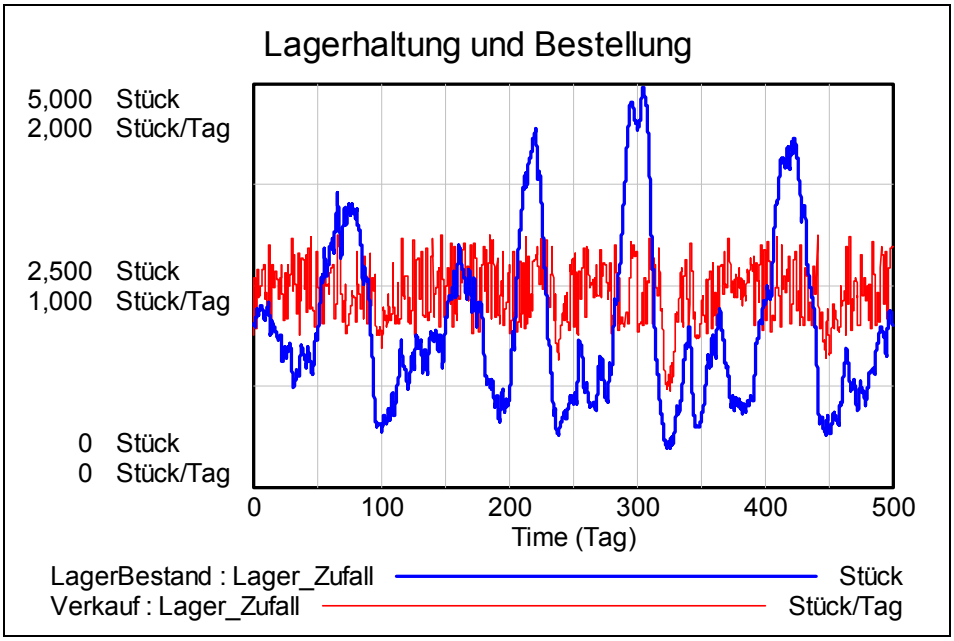
SAVEPER = TIME STEP
Units: Tag
The frequency with which output is stored.

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

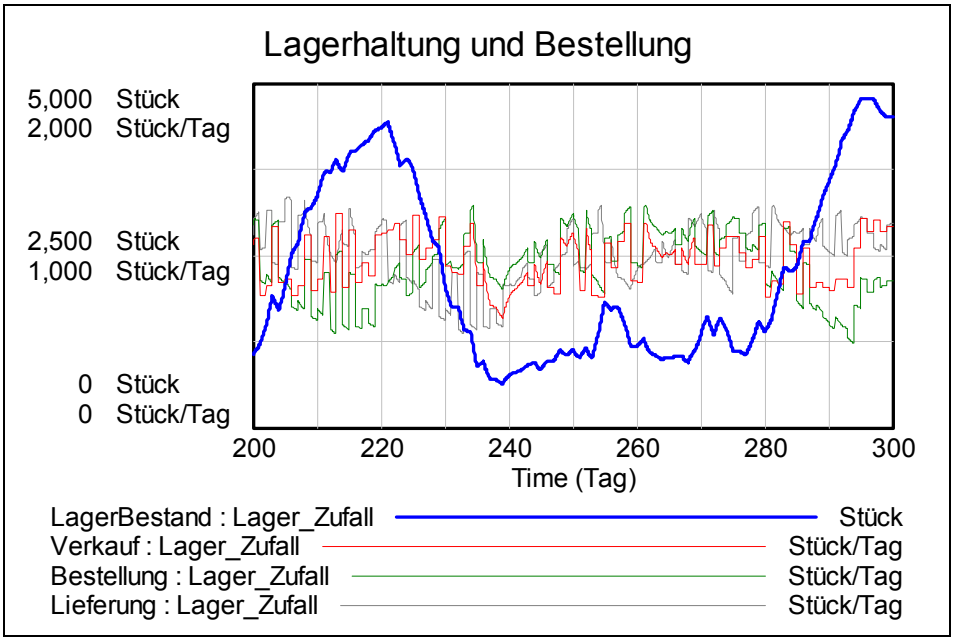
In der ersten Simulation setzen wir den VERKAUFSPULSPRZ auf 0 und beobachten so den Verlauf, der sich nur aus den zufälligen Schwankungen ergibt.

Zu beachten ist die Anweisung DELAY FIXED in der *Anlieferung*. Jede Verzögerung muss den Wert der Eingangsgröße vorerst speichern. (DELAY werden wir im letzten Beispiel auch noch gut verwenden können.)

Betrachten wir die Entwicklung über den Zeitraum von 500 Tagen:



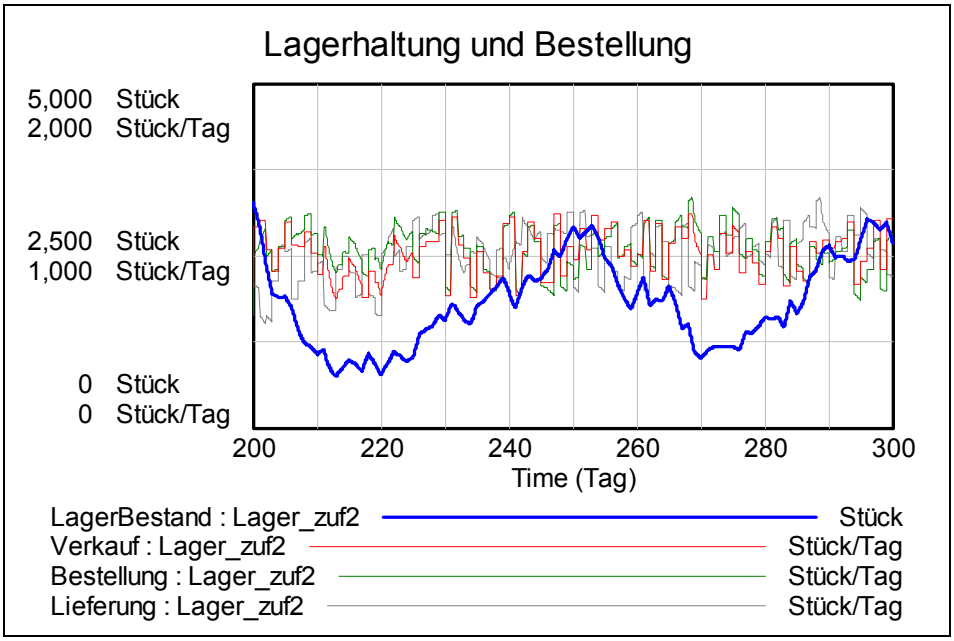
Wir können einen beliebigen Zeitraum herausgreifen und diesen aufspreizen, wie z.B. den Ausschnitt zwischen den Tagen 200 und 250.



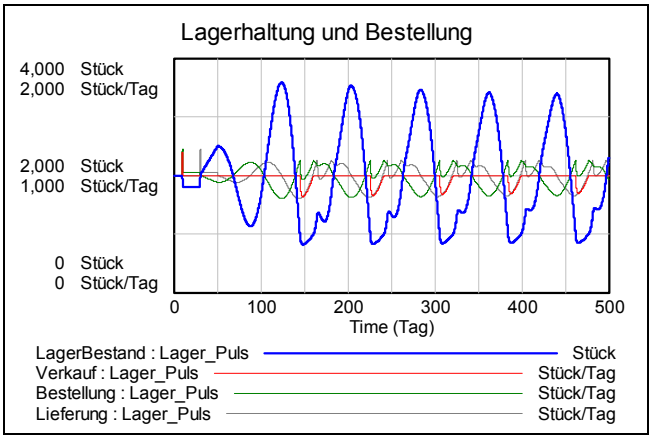
Ich zitiere hier die Interpretation von *Bossel*:

> Solange noch keine Lieferreaktionen auf frühere Bestellungen eintreten, führen die zunehmende Abweichung vom LAGERBESTANDSZIEL und weiterer *Verkauf* zu weiteren Bestellungen, die erst nach der LIEFERVERZÖGERUNG zur *Lieferung* führen. Mit diesen steigt der *LagerBestand* zunächst an, woraufhin die *Bestellungen* wieder zurück genommen werden. Entsprechend der LIEFERVERZÖGERUNG sinkt danach die *Lieferung* wieder nach einiger Zeit, woraufhin wieder etwas mehr bestellt wird usw.

Die Funktion RANDOM UNIFORM(0, 1, 0) erzeugt eine gleich verteilte Zufallszahl in Intervall (0,1) mit der Zahl 0 als „Initialzündung“ für den Zufallsgenerator. Das heißt, dass wir solange bei jedem Simulationslauf die gleichen Pseudozufallszahlen erhalten, solange wir diesen Parameter nicht ändern. Ich habe nun in der Gleichung für *neuen Wert ermitteln* RANDOM UNIFORM(0, 1, **1**) eingesetzt und zeige hier den Verlauf dieser 50 Tage. Lässt sich hier die Beobachtung von *Bossel* bestätigen?



Nach der Simulation mit zufälligen Verkaufsschwankungen wiederholen wir die Simulation mit einem einmaligen „Verkaufspuls“ und danach konstantem Tagesverkauf.



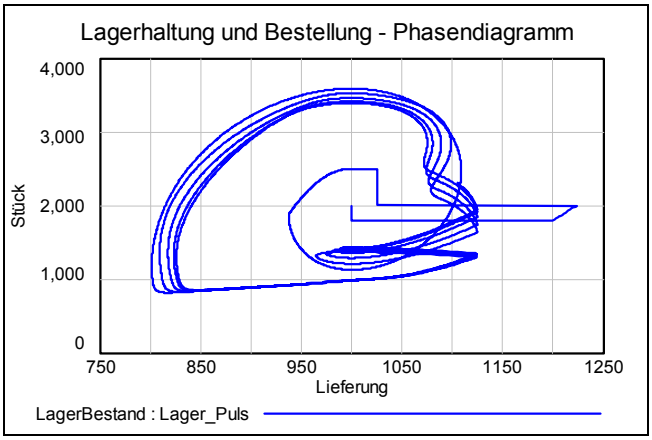
Zwei Parameter sind zu ändern:

MAX VERKAUFS SCHWANKUNG = 0

VERKAUFSPULSPRZ = 20

Hier kommt es zu einer ungedämpften periodischen Schwingung des Lagerbestands mit einer Periode von ca. 80 Tagen.

Das Phasendiagramm zeigt das sehr schön.



Ein entsprechendes Programm mit *DERIVE*

Ich muss zugeben, dass ich vor der Umsetzung in ein *DERIVE*-Programm einige Mühe hatte, das Modell händisch nachzuvollziehen. Die Mühe hat sich aber schließlich gelohnt. Ich werde diesen Schritt hier auslassen, denn aus dem Programm sollte man die Vorgangsweise deutlich heraus lesen können.

Die Parameter habe ich vordefiniert und nicht als Funktionsargumente verwendet. Ich denke, dass ihre Eingabe so deutlicher ist. Natürlich wollte ich meine Ergebnisse mit den *VENSIM*-Daten vergleichen. *DERIVE* liefert andere (Pseudo-) Zufallszahlen, daher: was tun?

Zu diesem Zweck habe ich die Liste der ersten 500 von *VENSIM* gelieferten Zufallszahlen relativ einfach in eine Liste für *DERIVE* übertragen. Diese Liste habe ich mit zufl bezeichnet. Hier sind die ersten 5 Zahlen dieser Liste im *DERIVE*-Format:

```
      zufl
#1:       [1, ..., 5]

#2:   [0.4927785098, 0.5666502700, 0.07161787000, 0.6510554600, 0.3773718198]
```

Die Variablenbezeichnungen wurden möglichst kurz – aber hoffentlich noch verständlich – gehalten, um eine gut lesbare Wiedergabe zu erreichen.

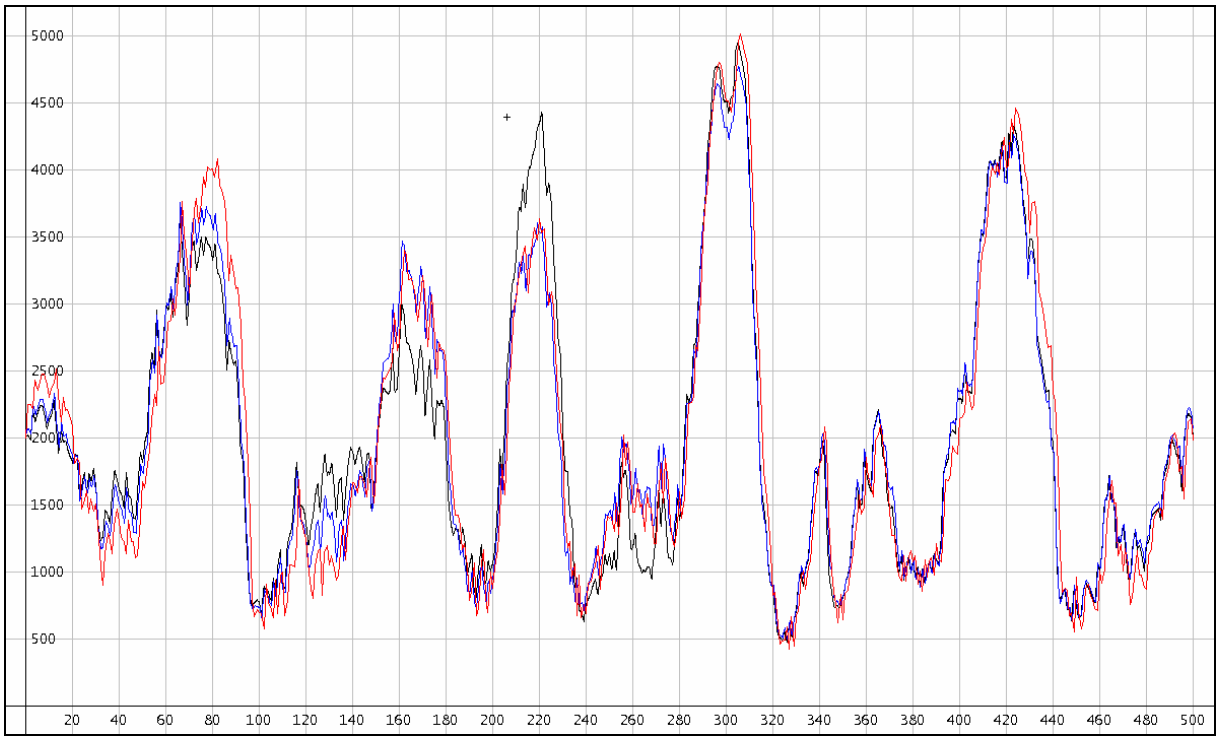
```
#3:   [0.4927785098, 0.56665027, 0.07161787, 0.65105546, 0.3773718198]

#4:   [l0 := 2000, bff := 0.125, bft := 1, lbz := 2000, lv := 20]

#5:   [mvs := 25, tvm := 1000, rvr := 1]

      lager(n, dt, i, z, za, verk, lief, best, lag, tz, tab, dummy) :=
        Prog
          "zufl := VECTOR(RANDOM(1), i, 500)"
          [n := n/dt, i := 1]
          [z := 0, lag := l0, lief := tvm, dummy := RANDOM(0)]
          za := tvm·2·mvs/100·(−0.5)
          verk := tvm + za
          best := bft·verk
          tab := [[z, lag, verk, best, lief]]
          Loop
#6:         If i > n
              RETURN tab
            z := z + dt
            tz := zufl↓CEILING(z)
            za := tvm·2·mvs/100·(tz − 0.5)
            lag := lag + dt·(lief − verk)
            verk := IF(tvm + za < lag·rvr, tvm + za, rvr·lag)
            lief := IF(i < lv/dt, tvm, tab↓(i − lv/dt + 1)↓4)
            best := IF(bft·verk + bff·(lbz − lag) > 0, bft·verk + bff·(lbz − lag), 0)
            tab := APPEND(tab, [[z, lag, verk, best, lief]])
            i :+ 1
```

In den Zeilen #3 bis #5 werden die Parameter definiert. Zeile #6 stellt das Programm dar.

Ich lasse nun das Zeit – Lagerbestand – Diagramm für die Zeitschritte 0,0625 (*Bossel*), 0,25 und 1 zeichnen.

Wir werden sehen, dass sich die gleiche Grafik (0,0625) wie bei *Bossel* ergibt und, dass es ausreicht, einen Zeitschritt von *dt* = 1 zu wählen.

```
#7:   (lager(500, 0.0625))↓↓[1, 2]

#8:   (lager(500, 0.25))↓↓[1, 2]

#9:   (lager(500, 1))↓↓[1, 2]
```



schwarz: *dt* = 0,0625; blau: *dt* = 0,25 und rot: *dt* = 1

Mir erscheint die Schrittweite von 1 (Tag) auch sinnvoll. Wer wird schon alle 30 Minuten (= 1/16 Arbeitstag von 8 Stdn) die Verkaufszahlen und die Bestellmenge usw. aktualisieren. Dies passiert üblicherweise am Ende des Tages - oder gar erst am Ende der Woche!

Ich habe in meinem Modell nur die Variante mit der Zufallsabweichung vom Mittelwert realisiert. Der *Pulsverkauf* wurde nicht eingebaut.

Wie oben erwähnt, habe ich die Zufallszahlen von *VENSIM* verwendet, um eine Referenz verfügbar zu haben. Optisch sieht das ja ganz gut aus, aber stimmen auch die Zahlen überein.

Vergleichen wir einen Ausschnitt aus den Tabellen mit den Resultaten:

Betrachten wir gleich die ersten paar Zeilen. Zuerst zeige ich *VENSIM*:

Lagerhaltung und Bestellung

Time (Day)	LagerBestand	Verkauf	Bestellung	Lieferung
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

Und anschließend *DERIVE*:

```
#10:  lager(0.25, 0.0625)

      ⎡    0        2000          750          750       1000 ⎤
      ⎢                                                       ⎥
      ⎢ 0.0625   2015.625    996.3892550  994.4361300   1000 ⎥
      ⎢                                                       ⎥
#11:  ⎢  0.125  2015.850671  996.3892550  994.4079210   1000 ⎥
      ⎢                                                       ⎥
      ⎢ 0.1875  2016.076343  996.3892550  994.3797121   1000 ⎥
      ⎢                                                       ⎥
      ⎣  0.25   2016.302014  996.3892550  994.3515031   1000 ⎦
```

Na ja, in den ersten Zeilen wird sich möglicherweise nichts tun, aber wie sieht es später aus.

Ich nehme nun den Ausschnitt um den Tag 20, da sollte ja die erste Bestellung termingemäß geliefert werden.

Nun zeige ich zuerst den Ausschnitt aus der *DERIVE*-Tabelle:

```
      (lager(500, 0.0625))
#12:                    [315, ..., 325]

      ⎡  19.625   1838.602725  1030.294595  1050.469254     1000     ⎤
      ⎢                                                              ⎥
      ⎢ 19.6875   1836.709313  1030.294595  1050.705930     1000     ⎥
      ⎢                                                              ⎥
      ⎢  19.75    1834.815900  1030.294595  1050.942607     1000     ⎥
      ⎢                                                              ⎥
      ⎢ 19.8125   1832.922488  1030.294595  1051.179283     1000     ⎥
      ⎢                                                              ⎥
      ⎢ 19.875    1831.029076  1030.294595  1051.415960     1000     ⎥
      ⎢                                                              ⎥
#13:  ⎢ 19.9375   1829.135664  1030.294595  1051.652636     1000     ⎥
      ⎢                                                              ⎥
      ⎢   20      1827.242252  1030.294595  1051.889313      750     ⎥
      ⎢                                                              ⎥
      ⎢ 20.0625   1809.723839   928.27672   952.0612400  994.4361300 ⎥
      ⎢                                                              ⎥
      ⎢ 20.125    1813.858803   928.27672   951.5443696  994.4079210 ⎥
      ⎢                                                              ⎥
      ⎢ 20.1875   1817.992003   928.27672   951.0277196  994.3797121 ⎥
      ⎢                                                              ⎥
      ⎣  20.25    1822.123440   928.27672   950.5112899  994.3515031 ⎦
```

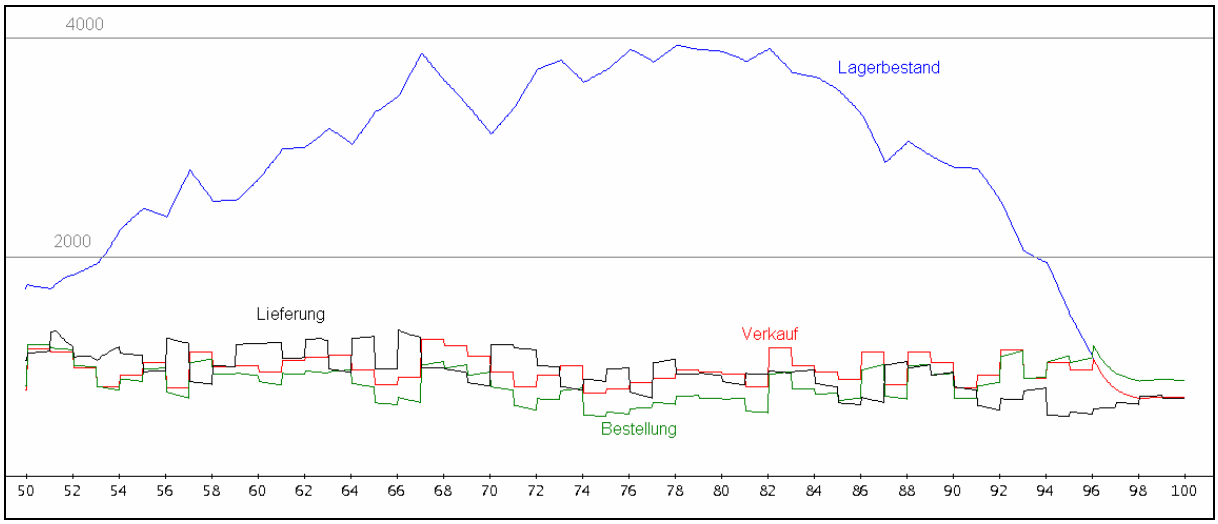
Der Vergleich mit *VENSIM* zeigt tatsächlich volle Übereinstimmung.

Lagerhaltung und Bestellung

Time (Day)	LagerBestand	Verkauf	Bestellung	Lieferung
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

Diese Übereinstimmung bleibt auch für andere Zeitschritte erhalten!

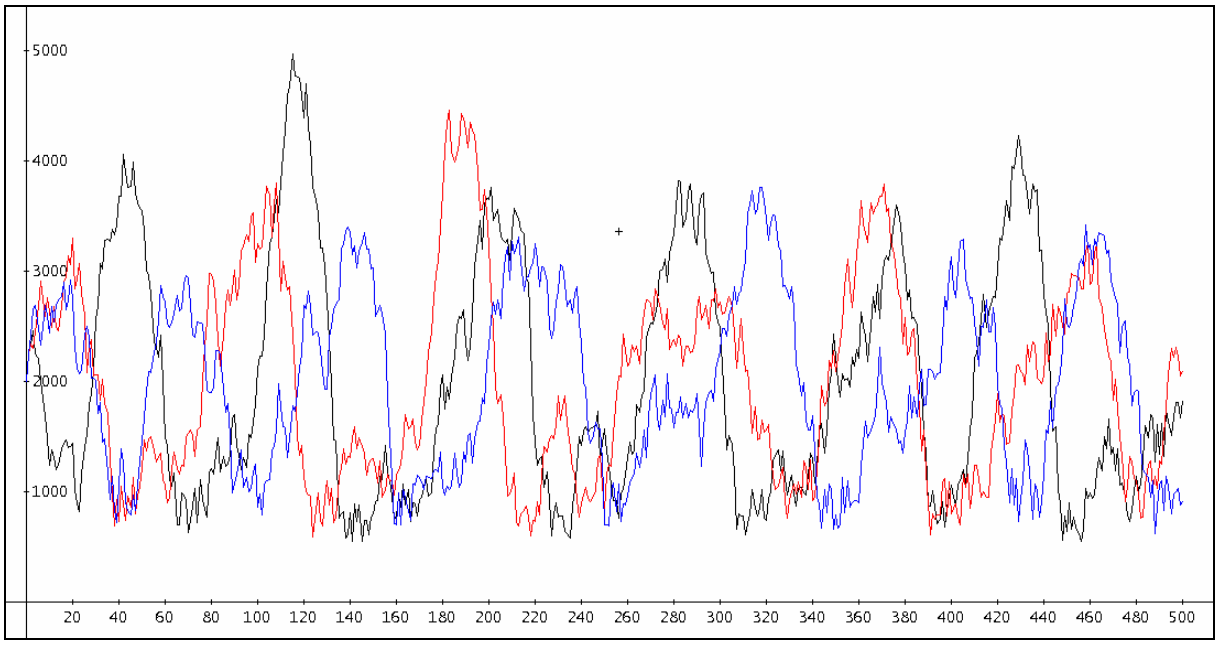
Die Aufspreizung für 50 Tage (mit gleicher Skalierung) ist leicht darzustellen.



Damit bin ich sicher, dass mein Modell richtig funktioniert und ich kann den Zufallsgenerator von *DERIVE* aktivieren.

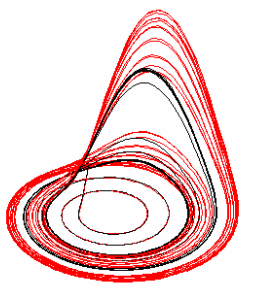
Dazu muss ich nur die Anführungszeichen (″) in der ersten Programmzeile löschen:

In der Grafik sehen Sie drei Simulationen für $dt = 1$:



Nach diesem Muster lässt sich natürlich auch die Simulation mit *TI-Nspire* und *GeoGebra* durchführen. Da hätten wir auch die Gelegenheit, mit Schiebereglern den Einfluss der Parameter zu beobachten.

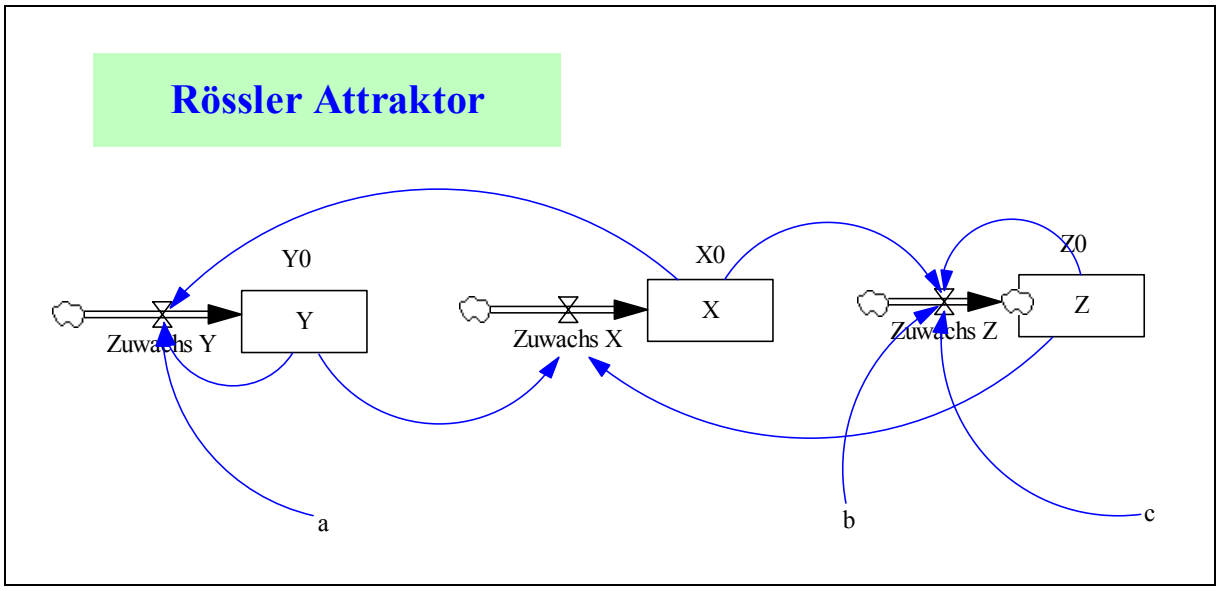
10 Der Rössler-Attraktor



Wenn man sich mit dynamischen Systemen, Fraktalen und Chaos beschäftigt, dann stößt man sicher früher oder später auf die „seltsamen Attraktoren“, wie z.B. den berühmten Lorenz-Attraktor, den Hénon-Attraktor, den Ikeda-Attraktor u.a. – oder eben den Rössler-Attraktor.

Der Rössler-Attraktor (benannt nach Otto Rössler, deutscher Biochemiker, geb. 1940 in Berlin) wird durch ein Differential-gleichungssystem beschrieben.

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



Hier treten recht wenige Größen auf, daher sind die Parameter und die Gleichung für die Änderungen der Zustandsgrößen rasch festgelegt:

Parameter für die erste Simulation

a = 0.55

b = 2

c = 4

X0 = 1

Y0 = 0

Z0 = 0

Dynamik

Zuwachs X = –Y – Z

Zuwachs Y = +X + a*Y

Zuwachs Z = b + Z*(X – c)

X = INTEG (Zuwachs X, X0)

Y = INTEG (Zuwachs Y, Y0)

Z = INTEG (Zuwachs Z, Z0)

Zeitparameter für die Simulation

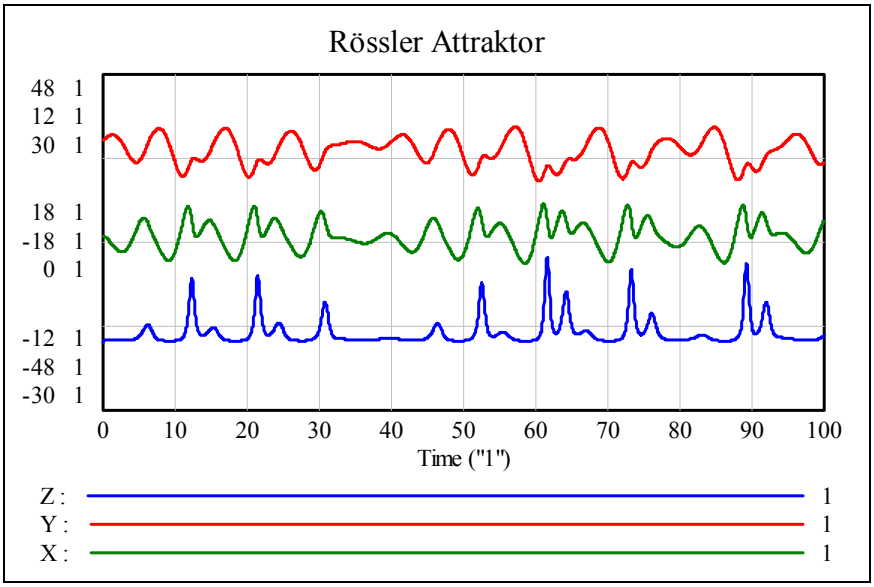
INITIAL TIME = 0

FINAL TIME = 100

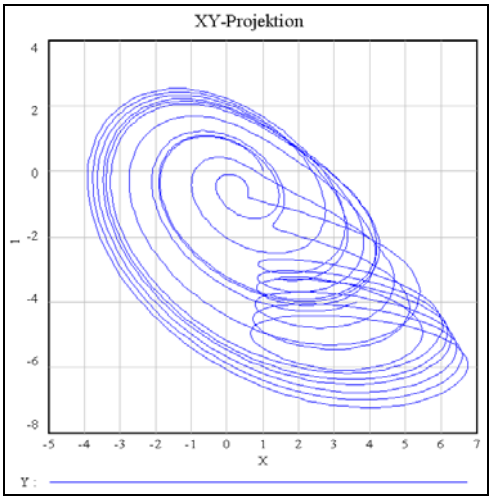
TIME STEP = 0.01

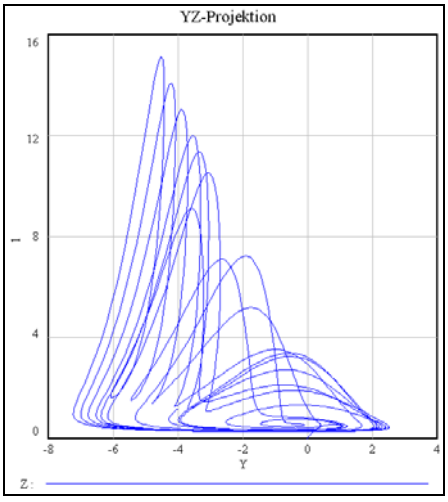
SAVEPER = 0.05

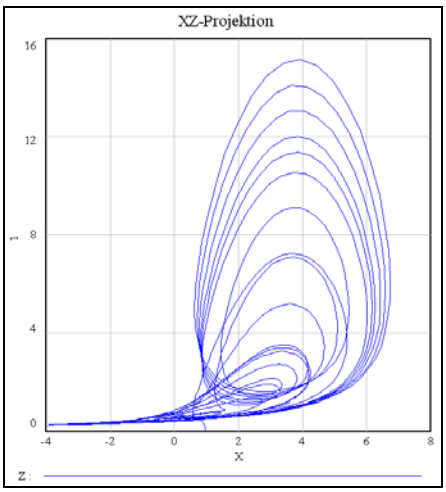
Für die ersten 100 Zeiteinheiten machen wir Schwingungen für *X*, *Y* und *Z* sichtbar. Die Grafiken habe ich der Deutlichkeit halber mit unterschiedlicher Skalierung etwas auseinander gezogen.



Diese Diagramme können auch mit SyntheSim bei Parametervariation betrachtet werden, die Zustandsdiagramme (Phasendiagramme) leider nicht. Leider lässt sich in *VENSIM* der Attraktor auch nicht in seiner 3D-Schönheit bewundern. So können nur Projektionen in die Koordinatenebenen dargestellt werden (Grundriss, Aufriss und Kreuz- oder Seitenriss).







Für manche Parameterkonstellationen ergeben sich Grenzlagen, wobei auch Periodenverdopplungen auftreten können. Das zeigen wir auf der nächsten Seite und dann später im Rahmen der *TI-Nspire*-Modellierung.

Wir bedauern nochmals, dass trotz der Vielseitigkeit von *VENSIM* (dessen Möglichkeiten wir hier überhaupt nicht ausschöpfen können) die räumliche Darstellung und Schieberegler nicht vorgesehen sind.

Diese beiden Features bietet *DERIVE*. Daher wende ich mich nun der Modellierung mit diesem CAS-Programm zu. Auch hier gibt es einen Wermutstropfen: Die räumliche Darstellung ist durch die hohe Anzahl der mit der Runge-Kutta-Methode berechneten Bildpunkte ausgezeichnet – aber aus bereits früher genannten Gründen kann ich auch hier die Variation der Parameter nicht durchführen.

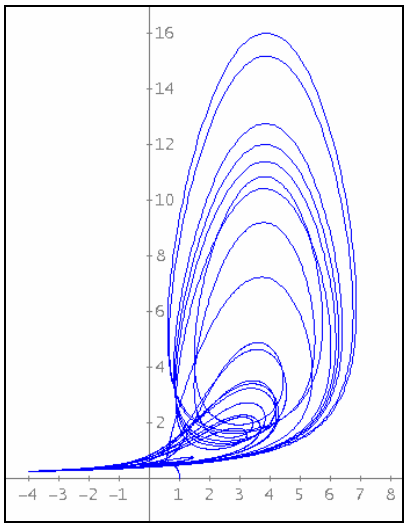
DERIVE und der Rössler Attraktor

Die Projektionen auf die Koordinatenebenen sind mit je einer Befehlszeile erzeugt. Nach wenigen Sekunden sieht man das Ergebnis der Berechnung von 10 000 Punkten mit der Runge-Kutta-Methode.

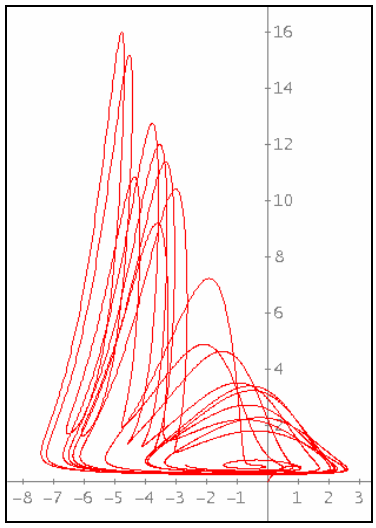
(RK([-y - z, x + 0.55·y, 2 + z·(x - 4)], [t, x, y, z], [0, 1, 0, 0], 0.01, 10000))↓↓[2, 4]

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

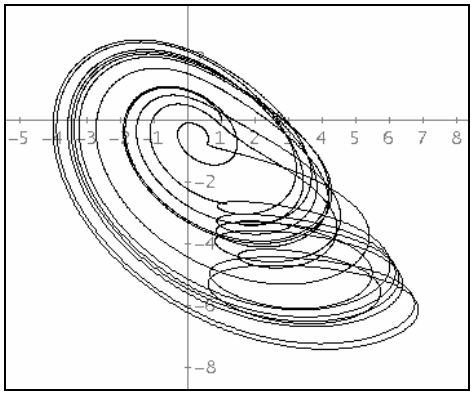
(RK([-y - z, x + 0.55·y, 2 + z·(x - 4)], [t, x, y, z], [0, 1, 0, 0], 0.01, 10000))↓↓[2, 3]



xz-Projektion



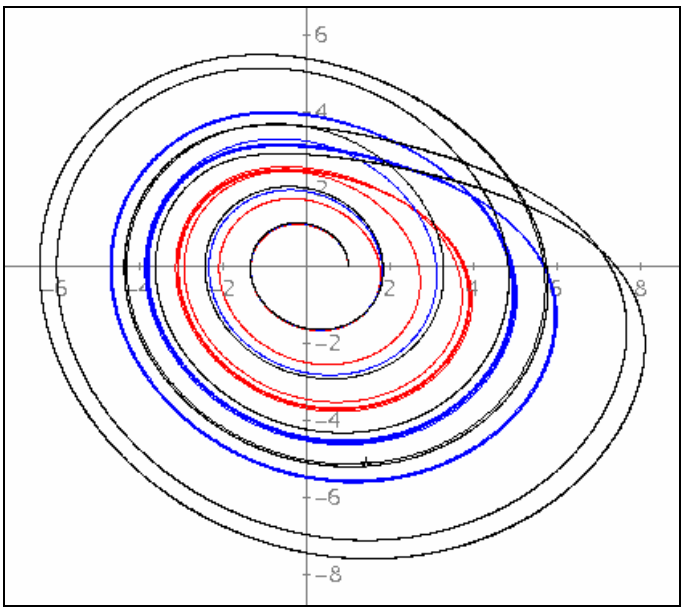
yz-Projektion



xy-Projektion

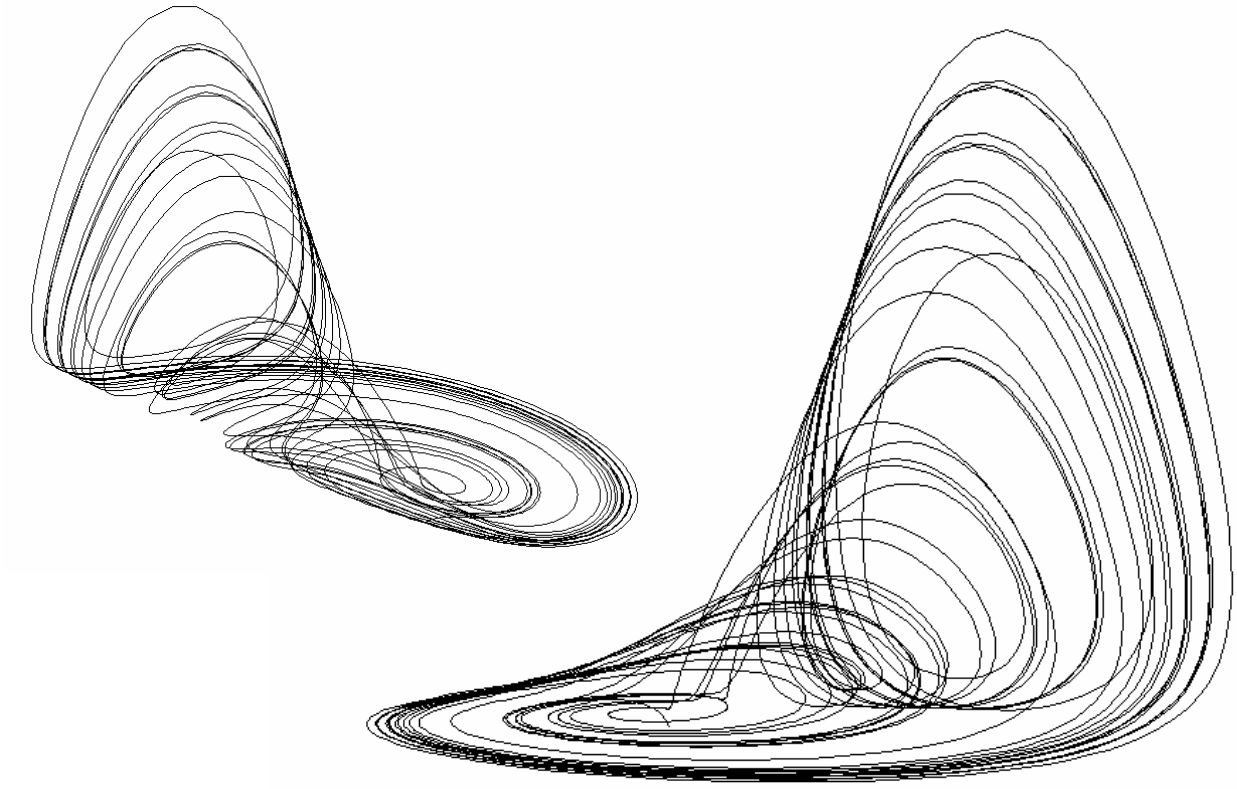
Bei dieser Wahl der Parameter ergeben sich für c = 2, 3 und 4 Grenzzyklen, die die Periodenverdopplung erkennen lassen.

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)



Mit *DERIVE* kann der schöne Attraktor auch dreidimensional dargestellt werden. Schon 4 000 Punkte reichen für eine ansprechende Darstellung im 3D-Zeichenfenster.

(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]

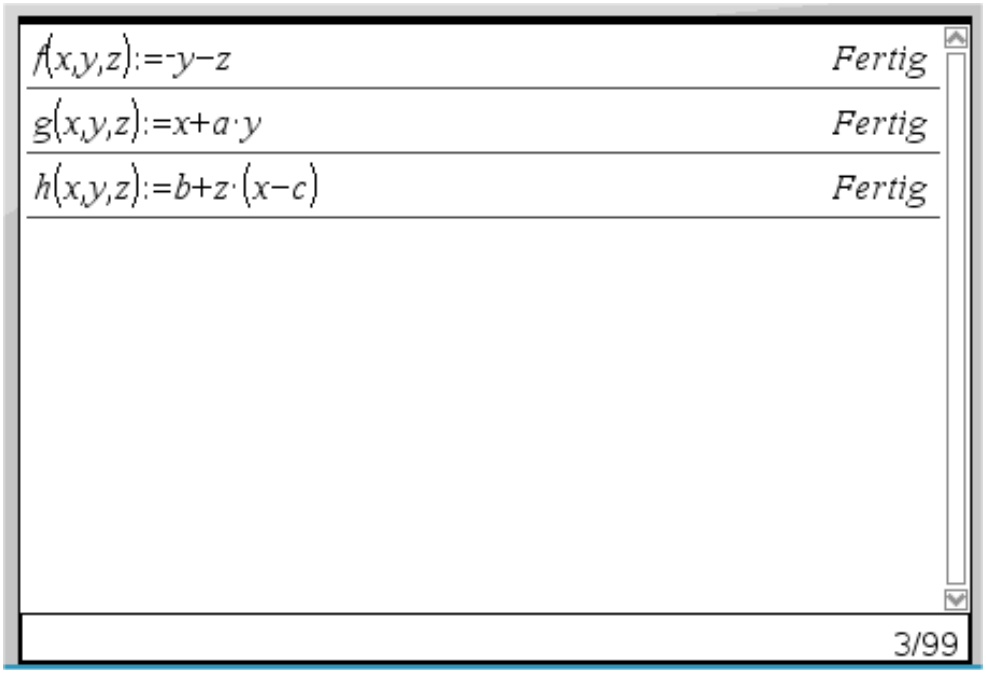


Der Rössler Attraktor mit Schiebereglern und *TI-Nspire*

Ich möchte nicht immer wieder für Änderungen der Parameter die Befehlszeile neu eingeben und neue Grafiken zeichnen (lassen), sondern den Einfluss der Parameter fließend beobachten. Das geht nur mit Schiebereglern. Diese stehen bei *TI-NspireCAS* (und bei *GeoGebra* sowie *MS-Excel*) zur Verfügung.

Ich versuche es einmal mit *TI-NspireCAS*. Wie schon bei früheren Modellen arbeite ich mit keinem Programm, sondern verlasse mich auf die Tabellenkalkulation. Meinen ersten Versuch machte ich mit der *Euler*-Methode. Um ein vernünftiges Ergebnis zu erhalten, musste ich die Schrittweite so klein halten, dass die Anzahl der Schritte (und damit der Zeilen in der Tabellenkalkulation) für *Nspire* zu groß war. Runge-Kutta ist machbar, aber sehr aufwändig. Ich wähle einen Mittelweg und arbeite mit der „verbesserten Euler-Methode“ zur Lösung von Differentialgleichungssystemen.

Ich richte für die Parameter *a*, *b* und *c*, für die Anfangswerte *x0, y0* und *z*0 und für die Schrittweite *dt* Schieberegler ein und definiere anschließend die rechten Seiten des Systems als eigenständige Funktionen.



Jetzt kann ich in die Spreadsheet-Applikation wechseln.

Für diese Situation gilt: *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

Die erste Zeile wird nach der verbesserten Euer-Methode Zelle für Zelle gefüllt:

Die erste Zeile lautet von A1 bis J1:

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)

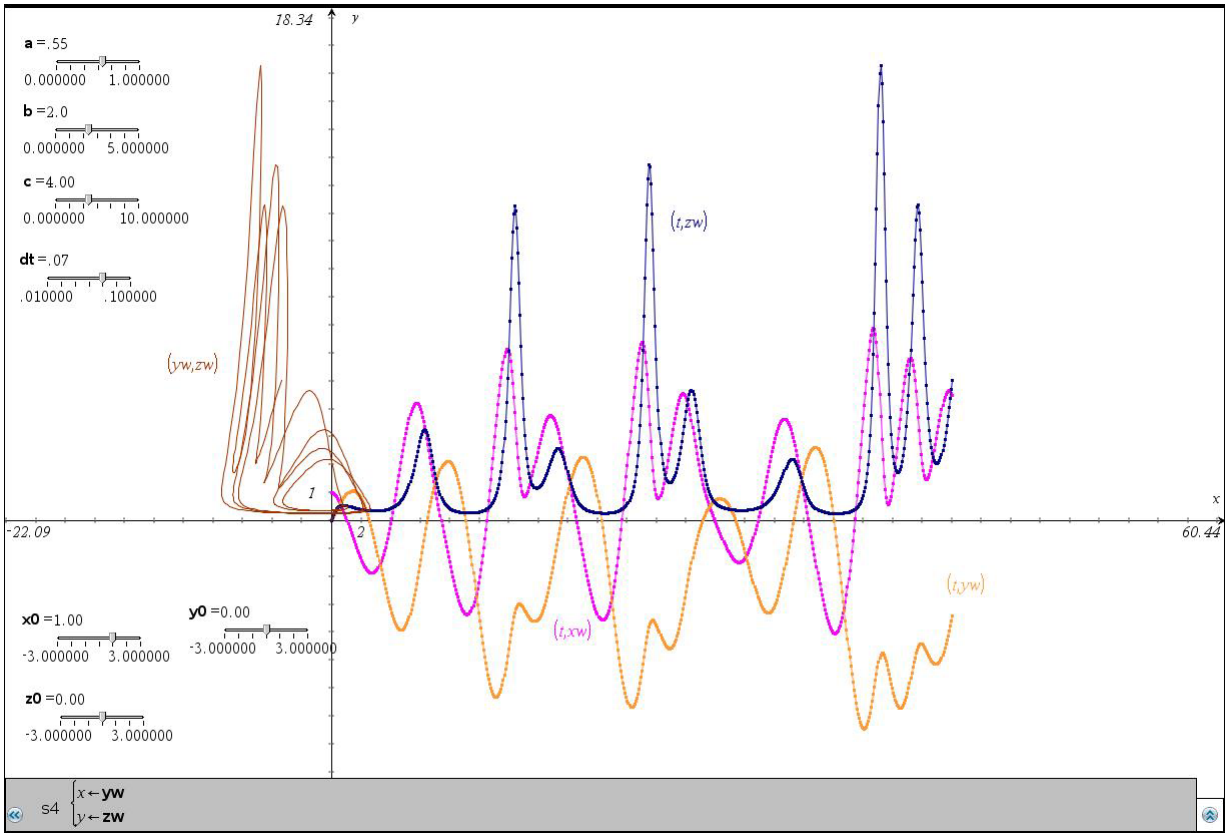
Nun folgt die zweite Zeile:

A2: = a1 + dt B2: = dt/2 (e1 + h1) C2, D2 sind Kopien von B2

F2 : J2 sind Kopien von F1 : J1.

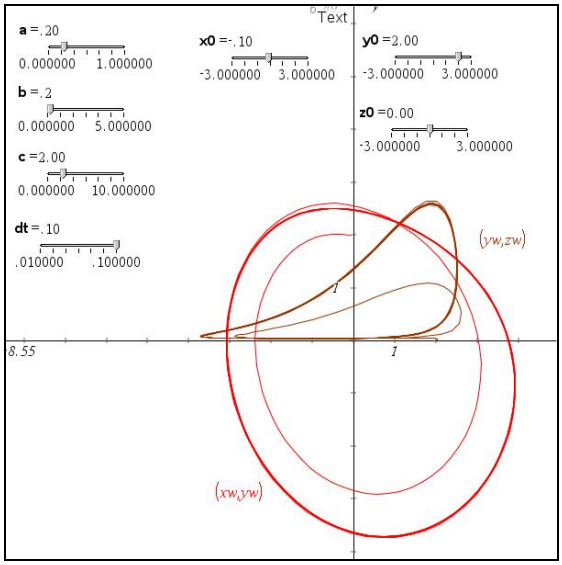
Diese zweite Zeile ist nach unten zu kopieren, um weitere Punkte zu erzeugen. Am besten geschieht das portionsweise um das System nicht zu überfordern zuerst bis Zeile 201, dann bis 401, und schließlich bis 601. Damit haben wir 600 Punkte und das reicht schon für ordentliche Grafiken. Die Spalten A bis D werden benannt und mit diesen Listen lassen sich die Streudiagramme zeichnen, die dann nach Belieben formatiert werden können – und vor allem, sie reagieren auf die Schieberegler.

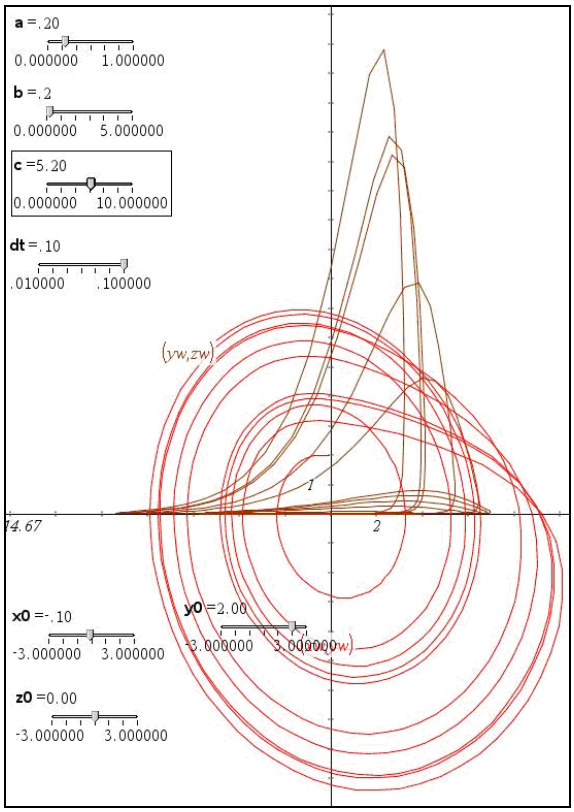
Der erste Screenshot zeigt alle Zeitdiagramme und eine Projektion.



Ich rücke die Schieberegler ein wenig und erhalte die Darstellung der Grenzzyklen:

XY-Projektion – Grenzzyklen





Für $c = 5{,}2$ lösen sich die Grenzzyklen ins Chaos auf (rechtes Bild).

Das funktioniert hier wirklich recht elegant und kann sicherlich auch in der Schule nachvollzogen werden.

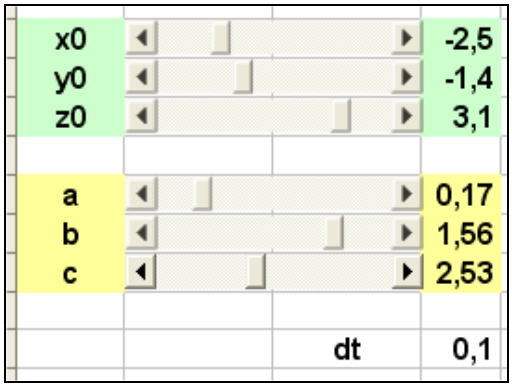
Ich gebe aber die Hoffnung nicht auf, auch ein 3D-Bild des Attraktors mit Schiebereglern zustande zu bringen. Wenn es einmal *GeoGebra* (oder auch *TI-Nspire*) mit einer 3D-Ansicht geben wird, dann könnte das möglich sein. Es gibt aber vielleicht noch eine andere Möglichkeit!!

Der Rössler-Attraktor dreidimensional und mit Schieberreglern

Die bei *Nspire* und *GeoGebra* implementierten Tabellenkalkulationen sind ja recht ordentlich, aber für richtig viele Daten greife ich doch wieder auf *MS-Excel* zurück.

Ich führe für alle Parameter – mit Ausnahme von *dt* – Schieberegler (hier: Bildlaufleisten) ein.

Die Eingabe der Formeln ist in *Excel* – zumindest für mich – etwas aufwändiger, weil ich keine Funktion mit mehreren Variablen definieren und verwenden kann. Aber der Aufwand hält sich wirklich in Grenzen.



Ich habe hier ohne Probleme 2000 Iterationen durchführen können und kann daher mit einer Schrittweite von 0,05 bis $t = 100$ bequem arbeiten.

Es ist dann schon faszinierend zu beobachten, wie alle drei Projektionen sofort auf jede Änderung der Parameter reagieren.

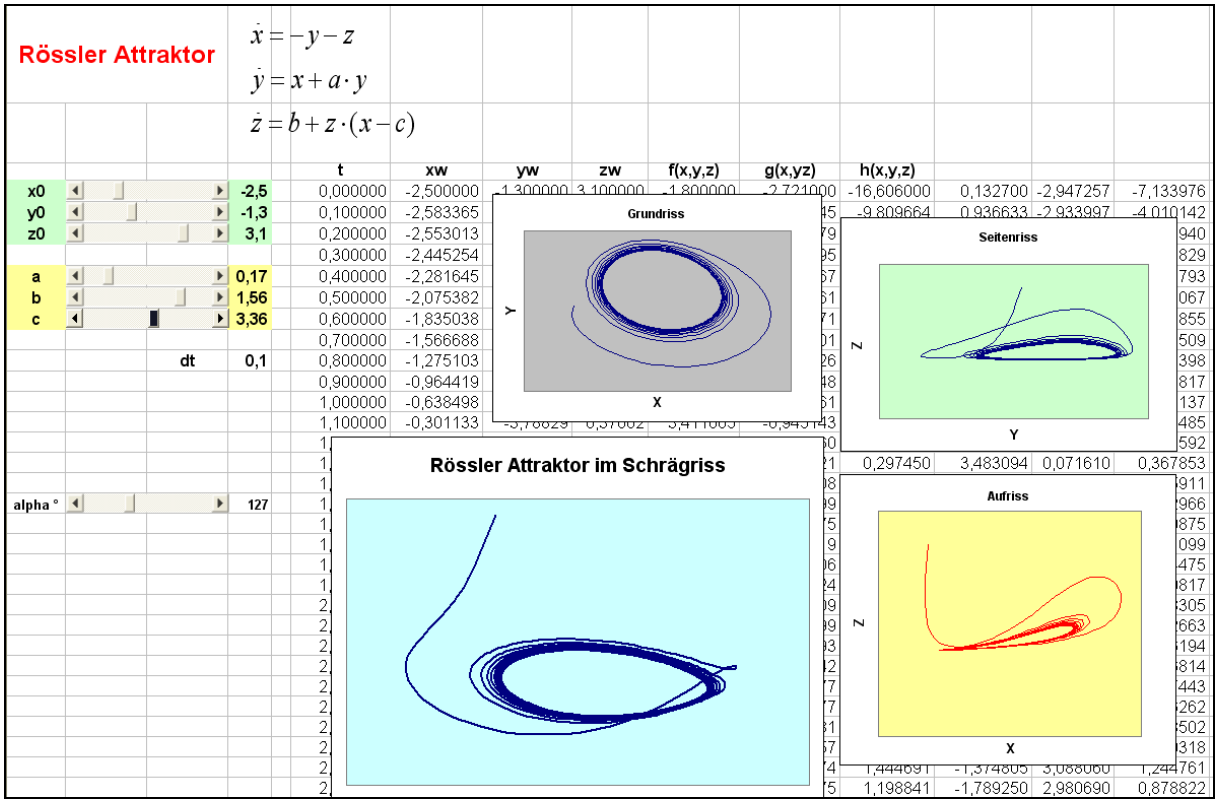
Nun, das hatten wir ja schon. Wie gelangen wir aber zu einem dreidimensionalen Bild? Mit einer geeigneten Transformation können wir eine Parallelprojektion oder auch eine Zentralprojektion (perspektivische Abbildung) des räumlichen Objekts auf die Bildebene erzeugen.

Grund- Auf- und Kreuz-(Seiten-)riss haben wir ja schon. Ich bleibe bei einer Darstellung, die wir alle in der Schule kennen gelernt haben: beim Schrägriss. Für diesen brauchen wir in einer Richtung (x- oder y-Achse) eine passende Verkürzung und einen Winkel, den diese verkürzte Achse mit der Horizontalen einschließt. Den Verkürzungsfaktor lege ich mit 0,75 fest, aber den Winkel mache ich auch – natürlich über eine „Laufbildleiste“ – variabel.

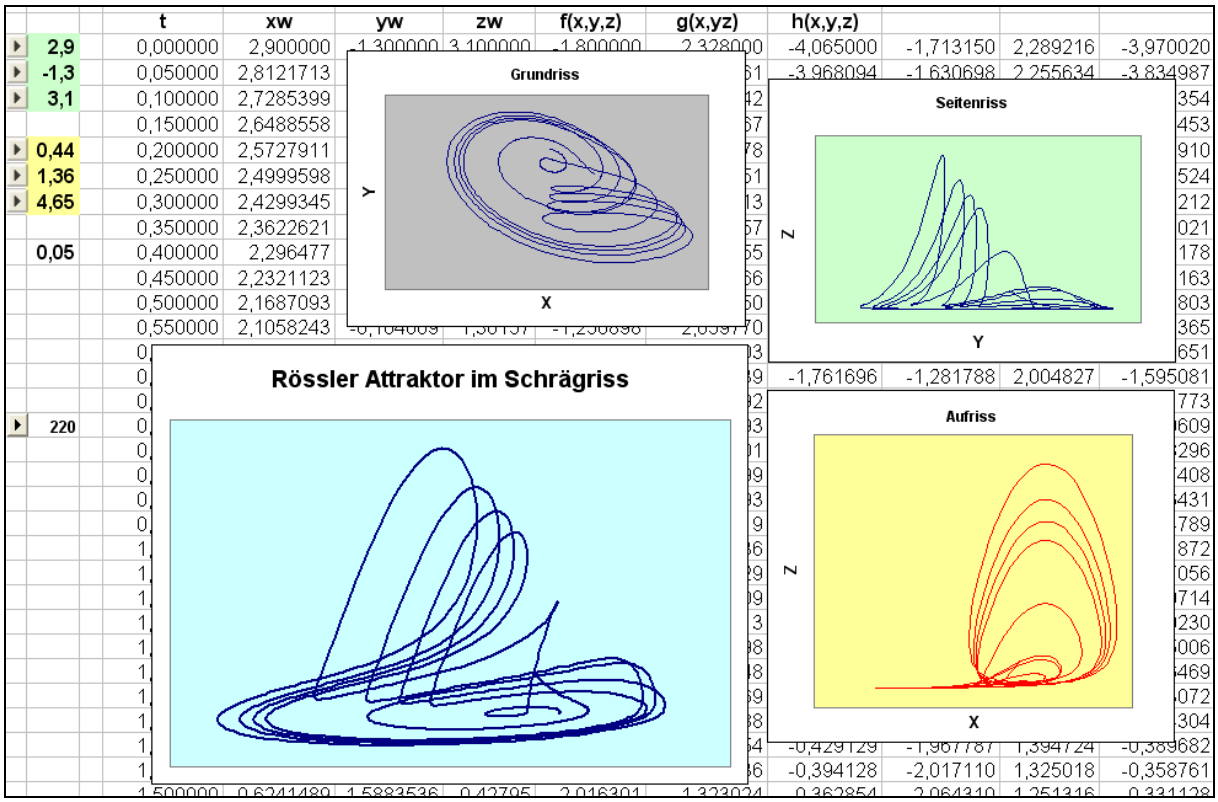
Die Transformationsgleichungen, die aus den drei Raumkoordinaten eines Punktes x, y, z die Bildkoordinaten x' und y' berechnen, lauten:

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

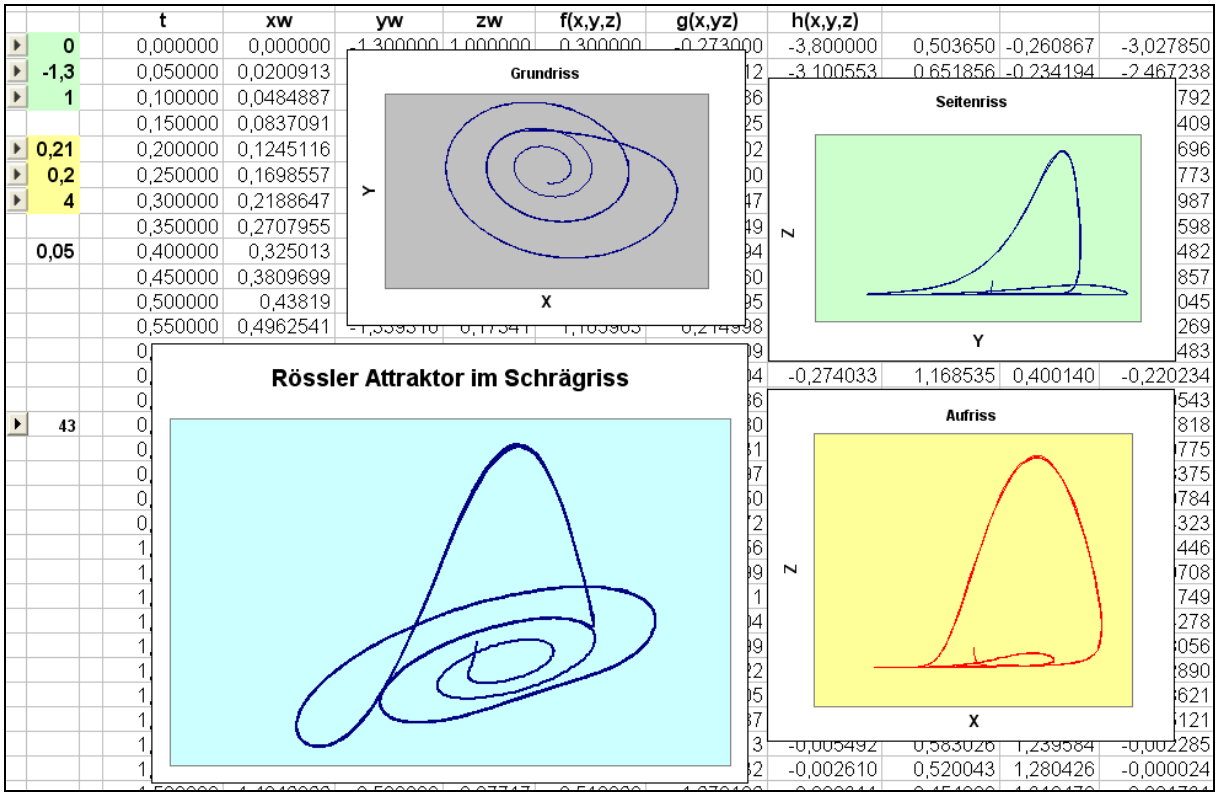
Diese Formeln werden sofort in die nächsten beiden Spalten eingetragen, nach unten kopiert und führen uns zum Diagramm, das nun neben Grund-, Auf- und Kreuzriss auch das Schrägbild des Attraktors zeigt (links unten).



Auf der nächsten Seite sieht man eine andere Konfiguration der Parameter und die entstehenden Grafiken basierend auf einer Schrittweite von $dt = 0,05$.



Dies ist dann noch das Bild mit den Grenzzyklen, die auf den Seiten 94 und 95 angesprochen wurden.



11 Gumowski-Mira - und noch ein „attraktiver“ Attraktor

Eigentlich sollte es mit dem 10. Kapitel Schluss sein. Dann hat mich jedoch der Ehrgeiz gepackt, noch etwas hinzu zu nehmen, das nicht im *Systemzoo* zu finden ist.

Die Welt der „seltsamen Attraktoren“ ist eine Wunderwelt von Formen und den dahinter stehenden Ideen. In einem meiner „Chaos“-Bücher aus meinem Bücherschrank habe ich einen Hinweis auf den *Gumowski-Mira-Attraktor* gefunden. Dieser wurde von zwei Physikern, I. Gumowski und C. Mira 1980 im Rahmen ihrer Tätigkeit am CERN in Genf gefunden.

Das Originalmodell wird beschrieben durch

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

mit $f(x) = a \cdot x + \frac{2(1-a)x^2}{1+x^2}$; a, b sind Konstante.

Gumowski-Mira mit einem CAS

Ich habe mich schon vor der Bekanntschaft mit *VENSIM* mit dieser so vielfältigen Klasse von Attraktoren beschäftigt. Dies geschah – für mich selbstverständlich – mit *DERIVE*, aber auch mit *WIRIS*.

Der *DERIVE*-Code ist einfach und es lassen sich mehrere tausend Punkte in rascher Zeit generieren und plotten.

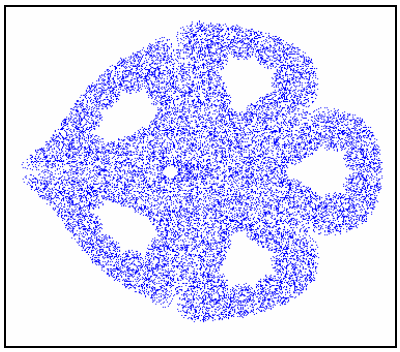
Nun gilt es, interessante Werte für a und b zu finden.

Bei meinen Internetrecherchen bin ich auf sehr schöne Websites gestoßen, auf denen ich für den Anfang eine große Auswahl an spannenden Grafiken gefunden habe [12, 13, 14,15].

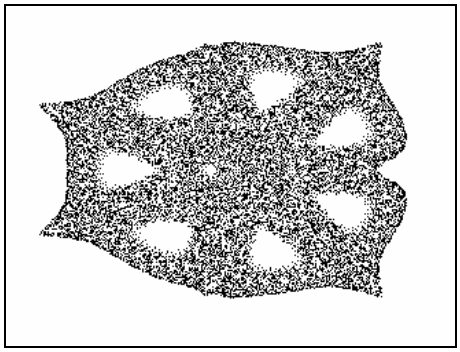
Richtigen Spaß bereitet aber natürlich das eigene Experimentieren und Forschen.

#1:
$$f(u, a) := a \cdot u + \frac{2 \cdot (1 - a) \cdot u^2}{1 + u^2}$$

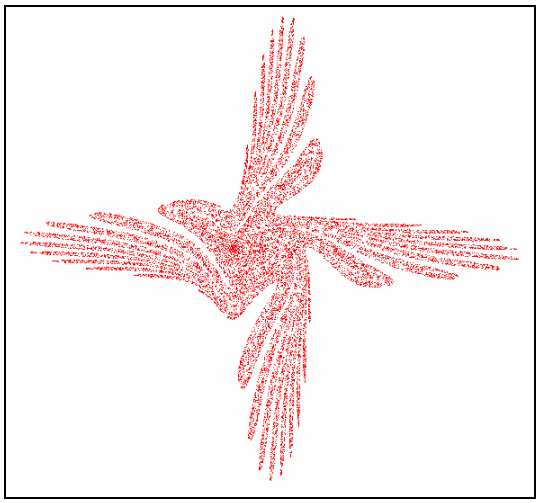
#2:
```
gum0(x0, y0, a, b, n, xn, yn, i, pts) :=
  Prog
    pts := [[x0, y0]]
    i := 1
    Loop
      If i > n
        RETURN pts
      xn := b·y0 + f(x0, a)
      yn := -x0 + f(xn, a)
      pts := APPEND(pts, [[xn, yn]])
      x0 := xn
      y0 := yn
      i :+ 1
```



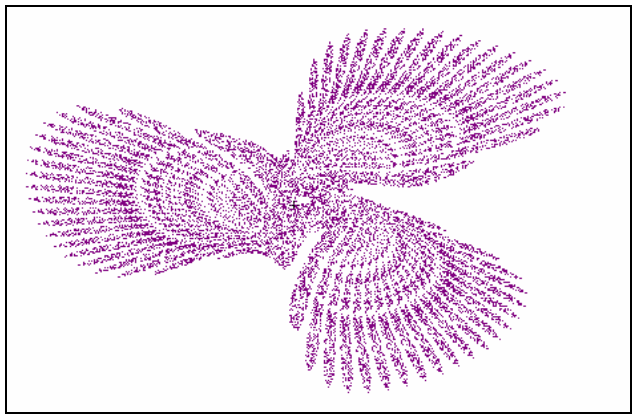
gum0(1, 1, 0.245, 1, 20000)



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



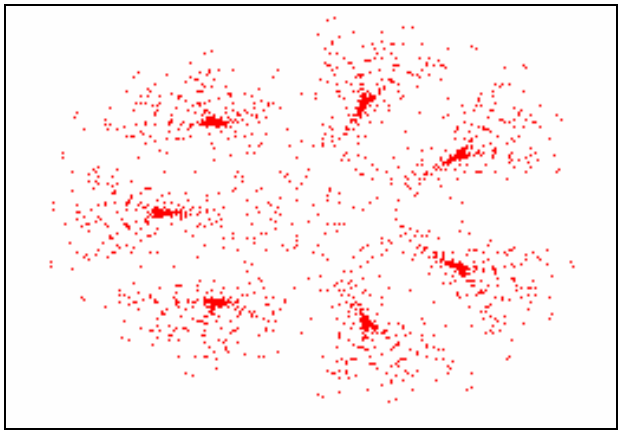
gum0(1, 1, 0.01, 0.978, 20000)



gum0(1, 1, -0.48, 0.9924, 20000)

Der rechte Attraktor ist noch eine „Eigenkreation“:

Bemerkung: Da wären meine geliebten Schieberegler wieder eine gute Hilfe!

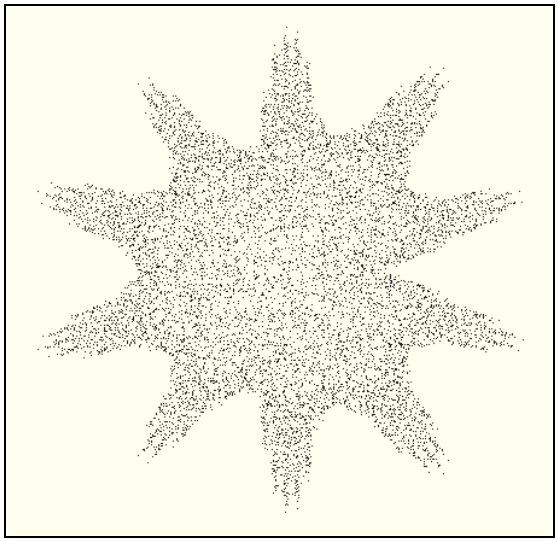


gum0(1, 1, -0.27, 0.995, 30000)

Mit dem CAS *WIRIS* gehen wir sehr ähnlich vor:

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

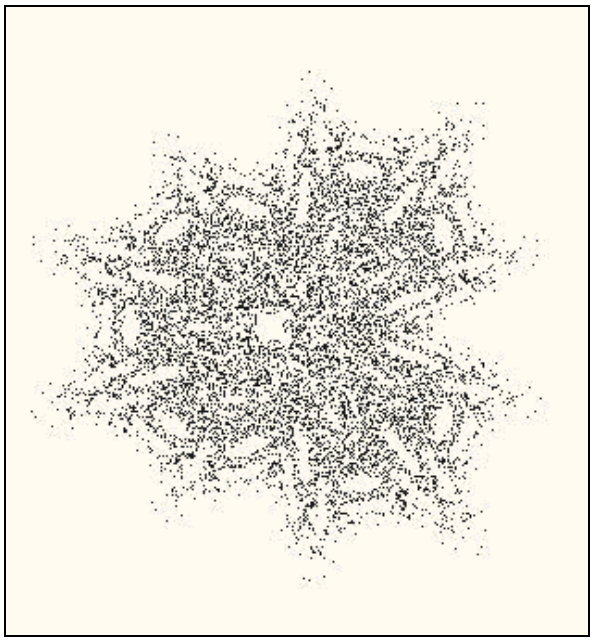
```
gum0(x0,y0,a,b,n) := begin
                       local list,xn,yn
                       list :={point(x0,y0)}
                       for i in [1..n] do
                          xn=b·y0+g(x0,a)
                          yn=-x0+g(xn,a)
                          list = list+{point(xn,yn)}
      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



Änderungen im Vorzeichen haben auch hier große Wirkung.

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

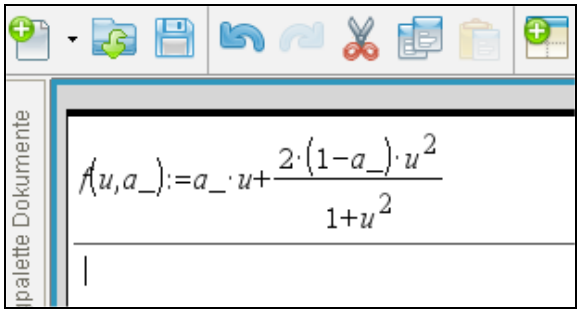
gum0(1,1,−0.305,1,10000) → plotter1



Mit Schiebereglern auf der Suche nach „schönen“ Attraktoren

Mit *TI-NspireCAS* lassen sich zwar keine Tausende von Punkten berechnen, aber es lässt sich meist schon aus den ersten paar hundert Punkten erahnen, ob da ein interessantes Bild entstehen könnte.

Daher werde ich nun dieses Werkzeug als Vehikel für meine Suche verwenden. Die Vorgangsweise ist ganz ähnlich zu der, mit der der Rössler-Attraktor behandelt wurde.



Hier werden nur zwei Spalten benötigt. Das Zeitdiagramm sagt überhaupt nichts aus und wird daher vernachlässigt.

	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			

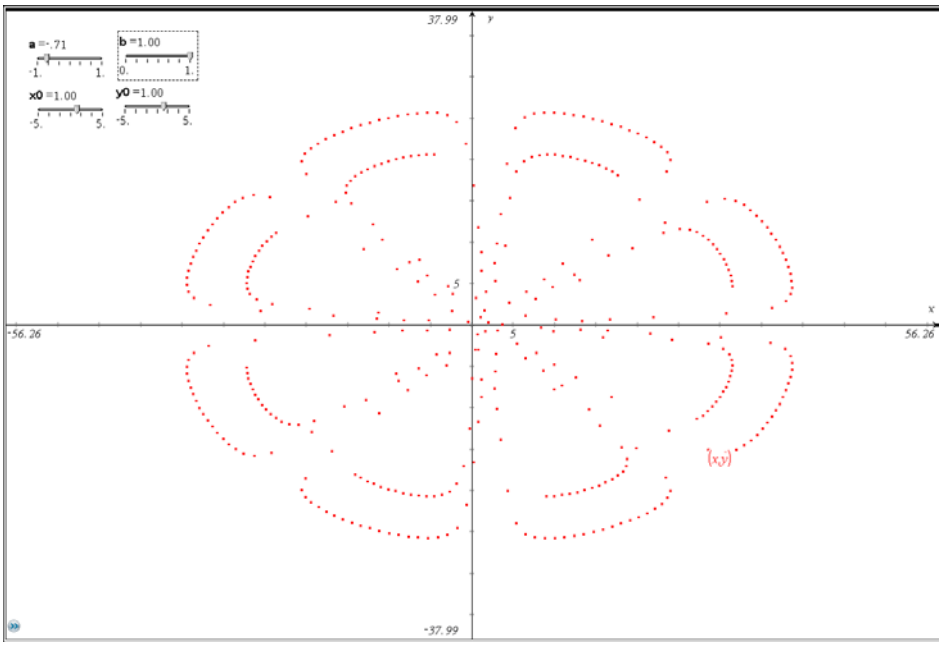
A2 =b·b1+f(a1,a)

	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)

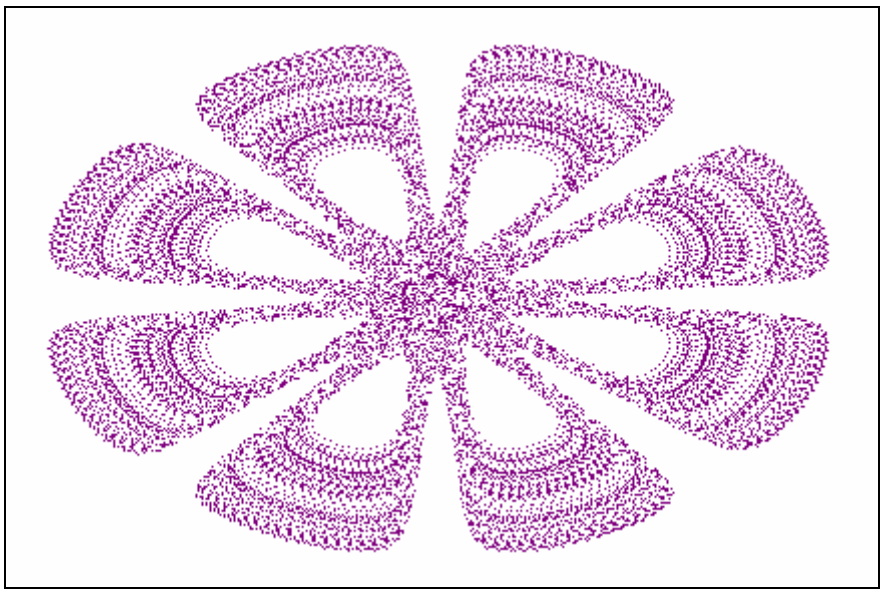
Mit den für *a*, *b*, *x0* und *y0* eingerichteten Reglern lässt sich nun ungehemmt experimentieren.

Und schon bald werde ich fündig. Diese Rosette verspricht einiges:

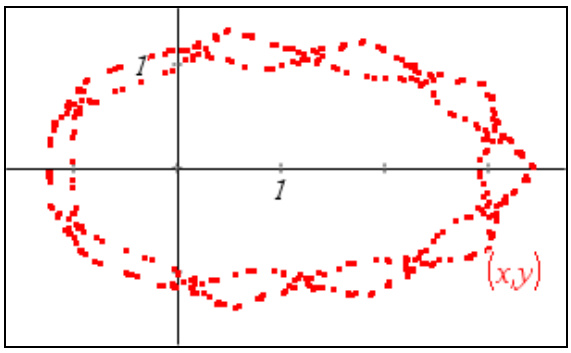


Ich wechsle zu *DERIVE* oder *WIRIS* und betrachte dann gleich 20 000 Punkte.

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

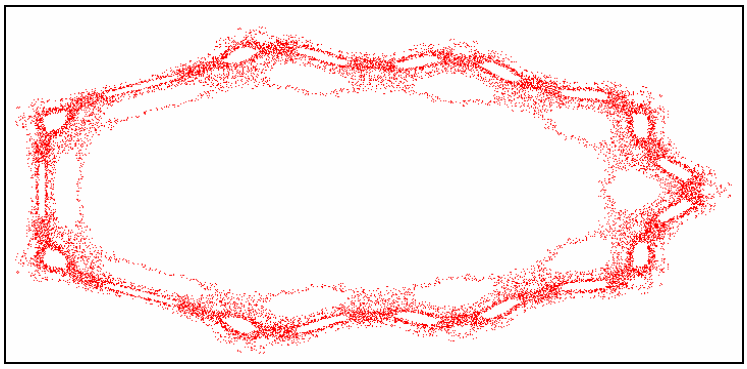


Wenn ich anstelle von $a = -0{,}71$ aber $a = 0{,}71$ wähle, dann zeigen die ersten paar hundert Punkte wiederum ein interessantes – ganz anderes – Bild.



Das macht mich wieder neugierig, wie der Attraktor in seiner vollen Schönheit wohl aussehen mag.

Na, bitte, auf der nächsten Seite ist er zu bewundern.



Und wie geht das nun mit *VENSIM*?

Leider muss ich zugeben, dass ich –als *VENSIM*-Novize – nicht im Stande war, ein geeignetes Modell zu aufzustellen. Der Zugriff auf dieselbe Indexebene in der Formel für y_{n+1} machte mir zu schaffen. So habe ich eine Anfrage an *VENSIM* über die Webseite `http://www.vensim.com` gestellt. Zu meiner nicht geringen Überraschung habe ich sofort eine Antwort erhalten, die allerdings fürs Erste nicht sehr hilfreich war:

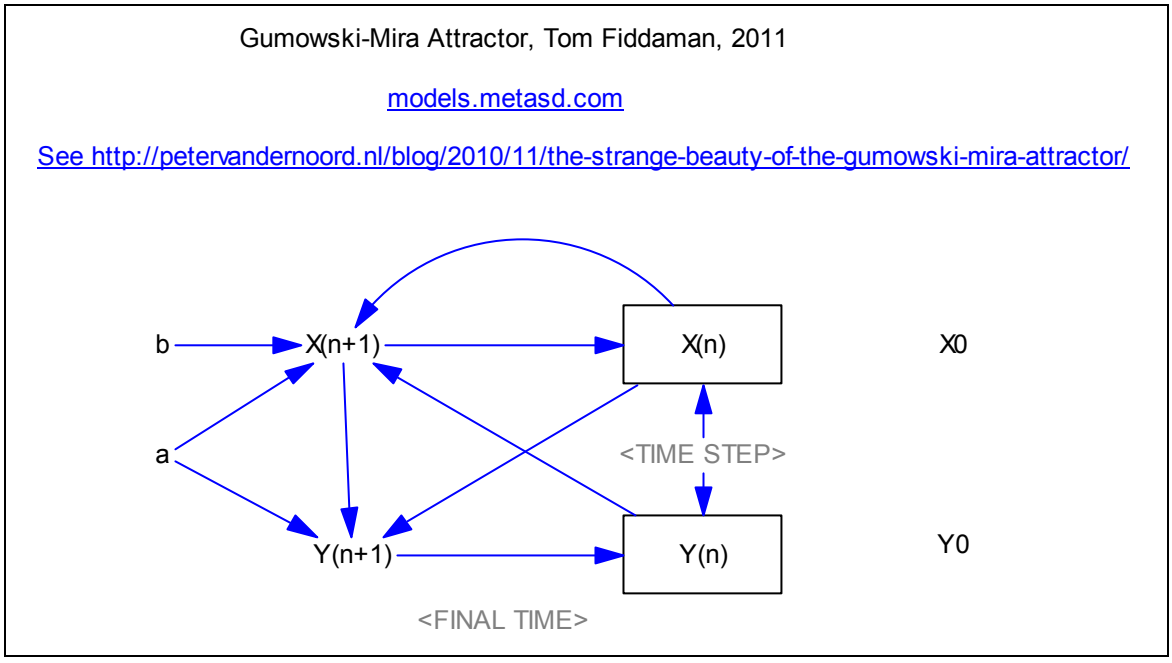
I don't see why you cannot do it. To get the old values of x, y and G, use DELAY FIXED with a delay time of one time step.

Ich musste mein Unvermögen eingestehen und erhielt eine Aufforderung:

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

Ich habe das sofort gemacht und – nächste Überraschung – da war gleich eine Antwort mit der entsprechenden *VENSIM*-Datei. Wie sich später herausgestellt hat, hat meine Anfrage auch eine interessante Diskussion im *VENSIM*-Forum[16] in Gang gebracht.

Vorerst die *VENSIM*-Umsetzung des Gumowski-Mira-Attraktors:



Das *VENSIM* - Protokoll gibt uns genauere Auskunft:

(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)

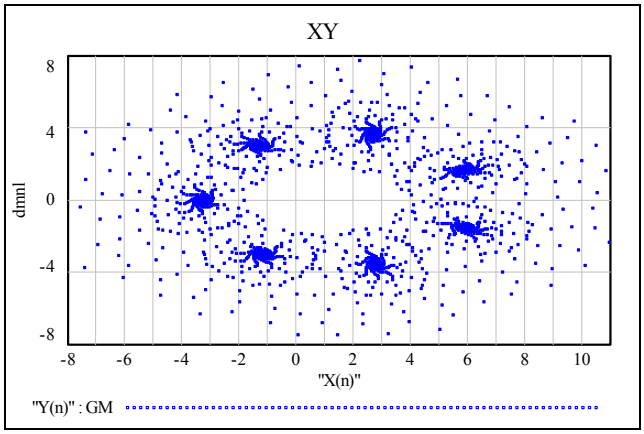
(12) "Y(n+1)"= -"X(n)"+a*"X(n+1)"+2*(1-a)*"X(n+1)"^2/(1+"X(n+1)"^2)

(13) Y0 = 7.75

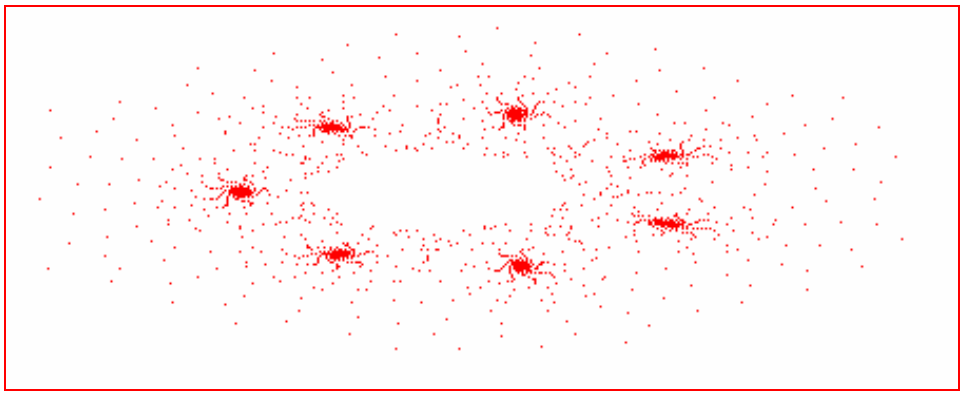
Wie man sieht, ist es die DELAY FIXED-Anweisung, welche die Verzögerung um einen Schritt ermöglicht. Für die hier gegebenen Parameter zeigt *VENSIM* die folgende Grafik.

Tom Fiddaman hatte noch einen kurzen Kommentar:

"The behavior is really amazing."



Und so sieht dieser „7-Sterne Haufen" bestehend aus 20 000 Punkten in *DERIVE* aus:



Das war aber doch nicht das Ende der Diskussion. Ein anderes Mitglied des *VENSIM* Forums meldete sich ebenfalls:

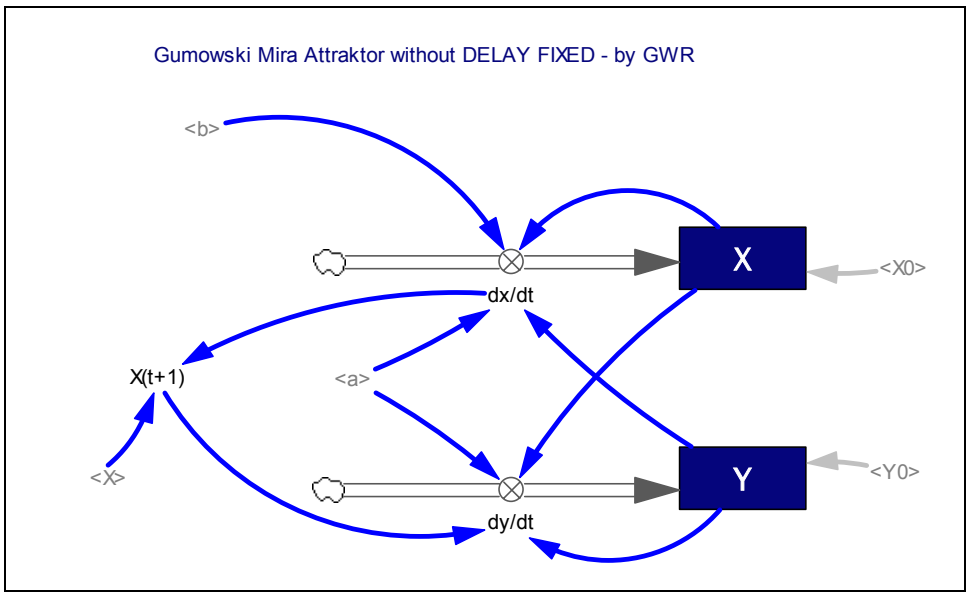
Hi Tom,

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

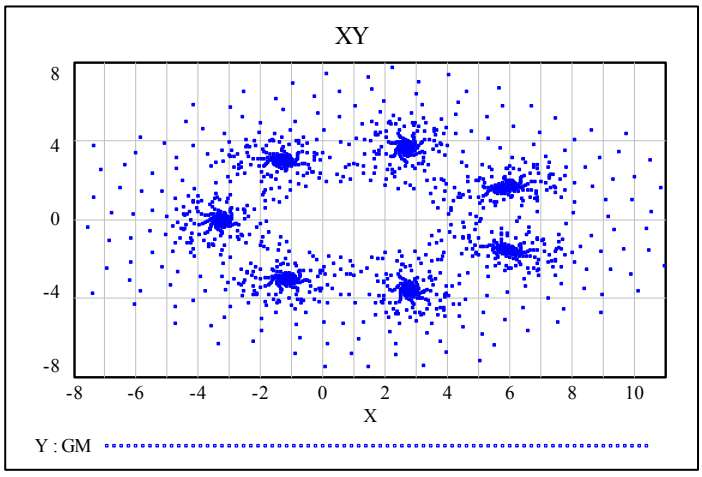
(09) X = INTEG("dx/dt",X0)

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

(13) X0 = 2.25

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

(17) Y0 = 7.75



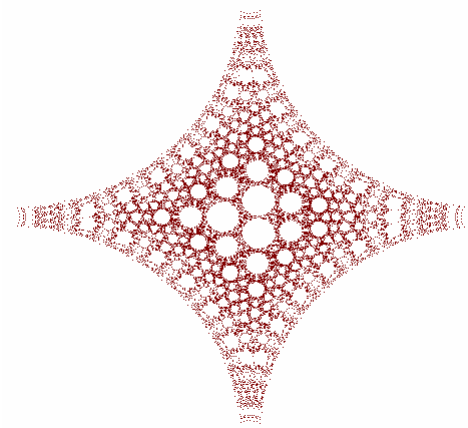
Es ist interessant, dass die beiden Graphen nicht genau übereinstimmen. Wenn man die z.B. die Werte für X vergleicht, kann man eine Übereinstimmung bis zu X_{34} erkennen, dann driften die Werte leicht auseinander. Offensichtlich werden hier zwei unterschiedliche Algorithmen im Hintergrund verwendet. Was meint Tom dazu?

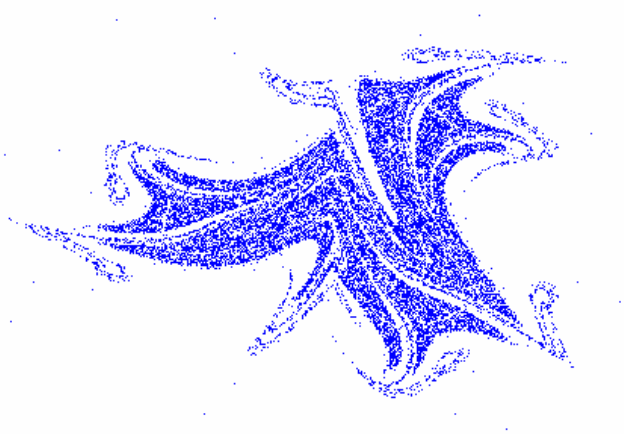
Guido,

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

Für die Funktion $f(x)$ können viele Varianten gewählt werden. Hier entstehen ganz wunderbare Grafiken.[17]





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

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

Damit will ich meinen kleinen Streifzug durch die Welt der dynamischen Systeme beenden, nicht ohne auf einige weitere sehr ergiebige Internet-Links zu verweisen.

---

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

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

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

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

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

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

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

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