HIV and the Immune System - A Mathematical Model

Josef Lechner, Amstetten, Lower Austria

Abstract

We discuss a mathematical model of increasing the HI-viruses in the human body. This model gives an answer to the question why the space of time between HI-infection and outbreak of AIDS differs to a great extent. It also gives insight into the phenomenon that our immune system is generally not able to root out HI-viruses completely.
This example is interesting from different view-points:It is well suited for lessons combining subjects like mathematics and biology. It demonstrates that even extensive simulations can be well done by the TI92/TI89 or DERIVE. It emphasizes the usefullness of recursively defined sequencies which prove to be a multi-functional tool.

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

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

Against every mutated virus the immune system must create new specific killer cells, which can only fight this special kind. The resistant cells act as *specialists*. On the contrary all mutating viruses can destroy *all* kinds of resistant cells against HIV or at least impair their function. The work as *generalists*. In a tough fight against steadily appearing new kinds of AIDS-agents, which are generated in the body of the infected person, the immunse system is not able to work as efficiently as it should to maintain the balance. If a certain variety of viruses is exceeded, the immune system finally loses control of them and AIDS breaks out. In this way the number of virus-paricels increases sharply whereas the number of immune cells drastically decreases. [NOWAK, 1992]

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

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

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

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

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

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

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

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

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

\$ The number of virus mutants as well as the kind and time of their appearance shall be controlled in this simulation.

\$ The more (iterative) equations (difference equations) are used the higher the likelyhood of mistakes and the less useful the simulation for application at school. Therefore it is decisive that the model remains applicable. Supporting programs or utility files, which create the respective equations automatically, prove to be a practicable way.

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

Realisation of HIV-simulation by means of TI-92

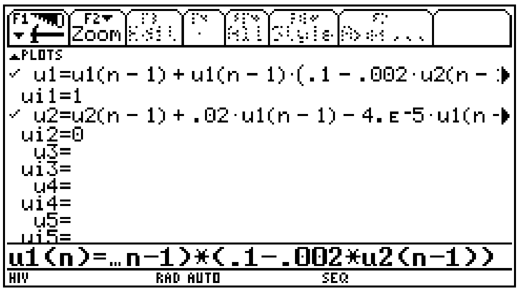
Simulation 1

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

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

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

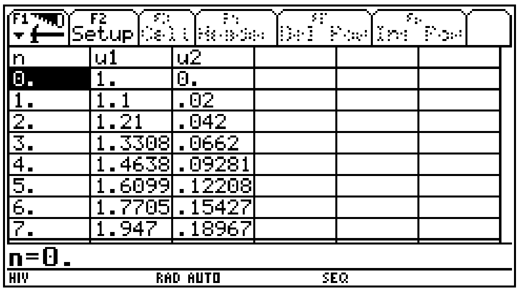
Viruses (type 1)	`u1(n)=u1(n-1)+u1(n-1)·(0.1- 0.002·u2(n-1))`
Starting value viruses	`ui1=1`
Resistant cells(type 1)	`u2(n)=u2(n-1)+0.02·u1(n-1)-0.00004·u1(n-1)·u2(n-1)`
Starting value	`ui2=0`



```
✓ u1=u1(n − 1) + u1(n − 1)·(.1 − .002·u2(n − :▶
  ui1=1
✓ u2=u2(n − 1) + .02·u1(n − 1) − 4.E-5·u1(n -▶
  ui2=0
   u3=
  ui3=
   u4=
  ui4=
   u5=
  ui5=
u1(n)=…n-1)*(.1-.002*u2(n-1))
```

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

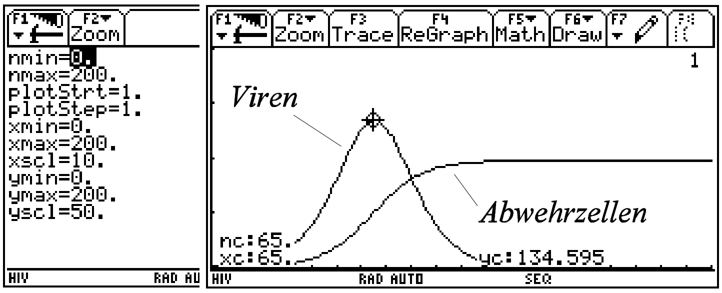
Table



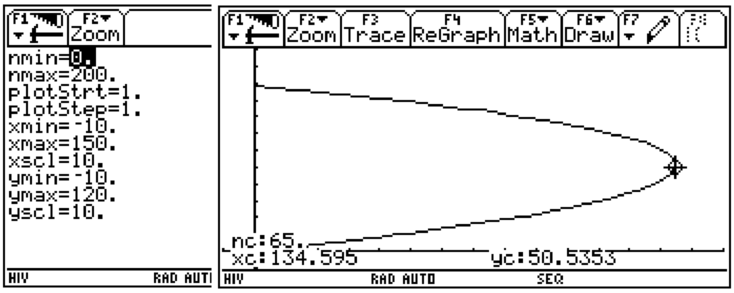
n	u1	u2			
0.	1.	0.			
1.	1.1	.02			
2.	1.21	.042			
3.	1.3308	.0662			
4.	1.4638	.09281			
5.	1.6099	.12208			
6.	1.7705	.15427			
7.	1.947	.18967			

n=0.

Graphic presentation of the development in time


```
nmin=0.
nmax=200.
plotStrt=1.
plotStep=1.
xmin=0.
xmax=200.
xscl=10.
ymin=0.
ymax=200.
yscl=50.
```

Graphic presentation of resistant cells against viruses


```
nmin=0.
nmax=200.
plotStrt=1.
plotStep=1.
xmin=-10.
xmax=150.
xscl=10.
ymin=-10.
ymax=120.
yscl=10.
```

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

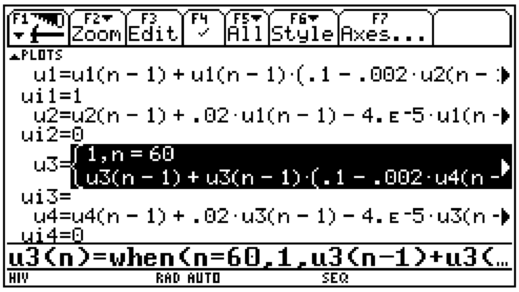
Simulation 2

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

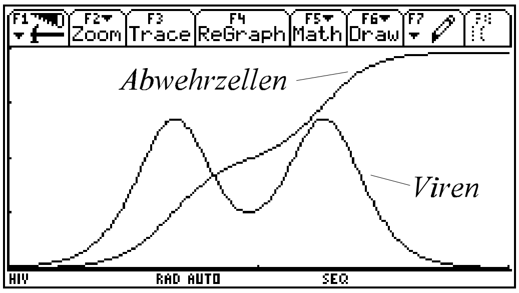
Viruses (Typ1)	`u1(n)=u1(n-1)+u1(n-1)·(0.1- 0.002·u2(n-1))`
Starting value viruses	`ui1=1`
Resistant cells (Typ 1)	`u2(n)=u2(n-1)+0.02·u1(n-1)-0.00004·u1(n-1)·u2(n-1)`
Starting value resistant cells	`ui2=0`

Viruses (Typ2)	`u3(n)= when(n=60, 1 ,`
	`u3(n-1)+u3(n-1)·(0.1- 0.002·u4(n-1)))`

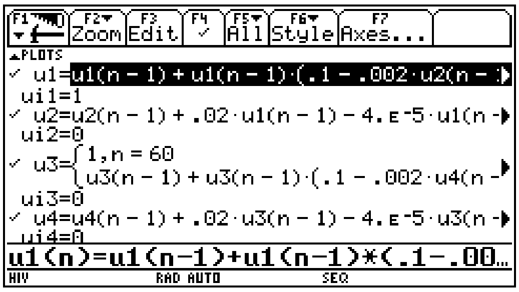
Starting value viruses	`ui3=0`
Restistant cells (Typ 2)	`u4(n)=u4(n-1)+0.02·u3(n-1)-0.00004·u3(n-1)·u4(n-1)`
Starting value resistant cells	`ui4=0`
Total number viruses	`u5(n)=u1(n-1)+u3(n-1)`
Startwert Virenzahl	`ui5(n)=0`
Total number resistant cells	`u6(n)=u2(n-1)+u4(n-1)`
Starting value resistant cells	`ui6(n)=0`


```
▲PLOTS
 u1=u1(n − 1) + u1(n − 1)·(.1 − .002·u2(n −
 ui1=1
 u2=u2(n − 1) + .02·u1(n − 1) − 4.E-5·u1(n
 ui2=0
 u3={ 1, n = 60
    { u3(n − 1) + u3(n − 1)·(.1 − .002·u4(n −
 ui3=
 u4=u4(n − 1) + .02·u3(n − 1) − 4.E-5·u3(n
 ui4=0
u3(n)=when(n=60,1,u3(n-1)+u3(…
```

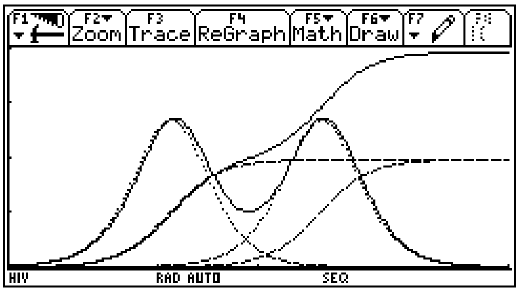
Total number of viruses and resistant cells



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



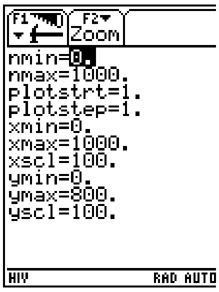
```
▲PLOTS
✓ u1=u1(n − 1) + u1(n − 1)·(.1 − .002·u2(n − ▶
 ui1=1
✓ u2=u2(n − 1) + .02·u1(n − 1) − 4.ᴇ⁻5·u1(n −▶
 ui2=0
✓ u3=⎧1,n = 60
     ⎩u3(n − 1) + u3(n − 1)·(.1 − .002·u4(n −▶
 ui3=0
✓ u4=u4(n − 1) + .02·u3(n − 1) − 4.ᴇ⁻5·u3(n −▶
 ui4=0
u1(n)=u1(n−1)+u1(n−1)*(.1−.00…
```



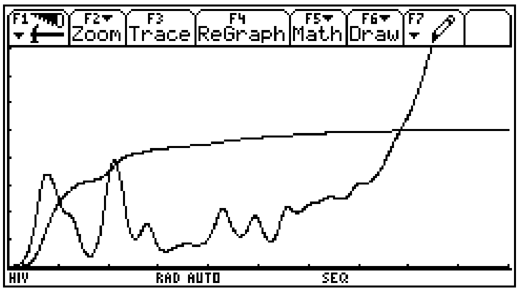
As it can be seen above, the input of the simulation equations is already extensive with only two mutants. Therefore it seems to be advisable to use a suitable program for simulations with more mutants.

Simulation 3

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

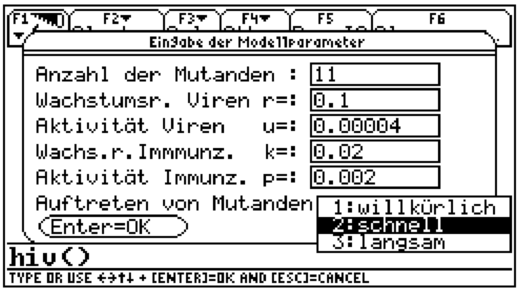


```
nmin=0.
nmax=1000.
plotstrt=1.
plotstep=1.
xmin=0.
xmax=1000.
xscl=100.
ymin=0.
ymax=800.
yscl=100.
```



How was this simulation performed?
We used the program HIV, which was started by the input of `hiv()` in Home-Screen.

The program allows the input of model parameters in the first step.



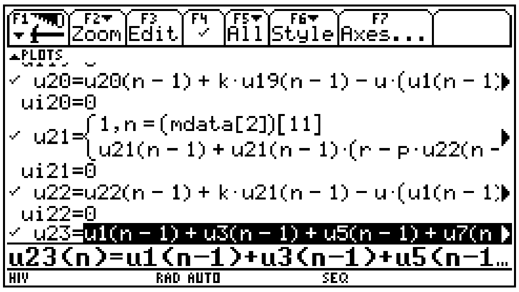
Eingabe der Modellparameter

Anzahl der Mutanden :	11
Wachstumsr. Viren r=:	0.1
Aktivität Viren u=:	0.00004
Wachs.r.Immmunz. k=:	0.02
Aktivität Immunz. p=:	0.002
Auftreten von Mutanden	1:willkürlich 2:schnell 3:langsam

hiv()

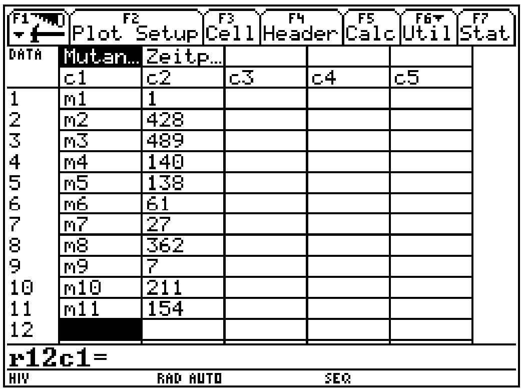
The item `Appearance of mutants" allows the use to control the time of appearance of the individual mutants on purpose or to select it quickly or slowly by accident. In the case of ˛quickly' all mutants appear within 500 steps of time (2.5 years), in the case of slowly within 4000 steps of time (20 years).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

After this first step the calculator is busy for some time - the respective equations are generated. Finally one can see that the equations have been compiled correctly.



```
✓ u20=u20(n - 1) + k·u19(n - 1) - u·(u1(n - 1)
  ui20=0
✓ u21= { 1,n =(mdata[2])[11]
       { u21(n - 1) + u21(n - 1)·(r - p·u22(n -
  ui21=0
✓ u22=u22(n - 1) + k·u21(n - 1) - u·(u1(n - 1)
  ui22=0
✓ u23=u1(n - 1) + u3(n - 1) + u5(n - 1) + u7(n
u23(n)=u1(n-1)+u3(n-1)+u5(n-1…
```

In a second step the appearace of mutants is controlled. If one has decided to choose one of the two random methods, one can see in the data-matrix-editor, when new mutants will appear.



	Mutan…	Zeitp…			
	c1	c2	c3	c4	c5
1	m1	1			
2	m2	428			
3	m3	489			
4	m4	140			
5	m5	138			
6	m6	61			
7	m7	27			
8	m8	362			
9	m9	7			
10	m10	211			
11	m11	154			
12					

r12c1=

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

Two final remarks:

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

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

Program HIV: Source-text

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

¨ Sequence-Bereich lÔschen
RclGDB leer

¨ Lokale Variablen
Local i,mlist,tlist,asum,astr,aisum,aistr,
      vsum,vstr,visum,vistr,modus,aktiv,mz

¨ Definitionen

"1"»mz:2»modus
"0.1"»r:"0.002"»p
"0.02"»k:"0.00004"»u

Define vhsum(mz)=Func
Local i,vhsstr
""»vhsstr:1»i
While iœmz
 If i=1 Then
  vhsstr&"u1(n-1)"»vhsstr
 Else
  vhsstr&"+u"&string(2*i-1)&"(n-1)"»vhsstr
 EndIf
 i+1»i
EndWhile
 vhsstr
EndFunc

Define ahsum(mz)=Func
Local i,ahsstr
""»ahsstr:1»i
While iœmz
 If i=1 Then
  ahsstr&"u2(n-1)"»ahsstr
 Else
ahsstr&"+u"&string(2*i)&"(n-1)"»ahsstr
 EndIf
 i+1»i
EndWhile
 ahsstr
EndFunc

"Define u"&string(i)&"(n)=when(n=(mdata[2])["&
string((i+1)/2)&"],1,u"&string(i)&"(n-1)+u"&
string(i)&"(n-1)*(r-p*u"&string(i+1)&"(n-1)))"»vstr(i)
"Define u"&string(i+1)&"(n)=u"&string(i+1)&
"(n-1)+k*u"&string(i)&"(n-1)-u*("&vhsum(mz)&")*u"&
string(i+1)&"(n-1)"»astr(i,mz)
"Define ui"&string(i)&"=0"»vistr(i)
"Define ui"&string(i+1)&"=0"»aistr(i)
"Define u"&string(2*mz+1)&"(n)="&vhsum(mz)»vsum(mz)
"Define u"&string(2*mz+2)&"(n)="&ahsum(mz)»asum(mz)
"Define ui"&string(2*mz+1)&"=0"»visum(mz)
"Define ui"&string(2*mz+2)&"=0"»aisum(mz)
```

```
¨ Modell spezifizieren
Dialog
 Title "Eingabe der Modellparameter"
 Request "Anzahl der Mutanden ",mz
 Request "Wachstumsr. Viren r=",r
 Request "Aktivitàt Viren   u=",u
 Request "Wachs.r.Immmunz.  k=",k
 Request "Aktivitàt Immunz. p=",p
 DropDown "Auftreten von
 Mutanden",{"willkÙrlich","schnell","langsam"},modus
EndDlog

¨ Iterationsgleichungen schreiben
expr(mz)»mz
expr(r)»r:expr(u)»u
expr(k)»k:expr(p)»p
For i,1,mz
 expr(vstr(2*i-1))
 expr(vistr(2*i-1))
 expr(astr(2*i-1,mz))
 expr(aistr(2*i-1))
EndFor
expr(vsum(mz)):expr(visum(mz))
expr(asum(mz)):expr(aisum(mz))

¨ Auftreten von Mutanden steuern
{}»mlist:{}»tlist
If modus=2 Then
   500»aktiv
EndIf
If modus=3 Then
   4000»aktiv
EndIf
For i,1,mz
augment(mlist,{expr("m"&string(i))})»mlist
If i=1 Then
   augment(tlist,{1})»tlist
Else
   If modus=1 Then
    augment(tlist,{0})»tlist
   EndIf
   If modus=2 Then
    augment(tlist,{rand(aktiv)})»tlist
   EndIf
EndIf
EndFor
NewData mdata,mlist,tlist

EndPrgm
```

Realisation of the HIV-Simulation by means of DERIVE

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

Simulation 1

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

```
"HIV-Simulation"

"(c) by Josef Lechner, Mai 1999"

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

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

```
NV(v, a) := v + v·(r - p·a)

NA(v, a) := a + k·v - u·a·v
```

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

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

„      1            0       †
|     1.1         0.02      |
|   1.20995   0.0419991     |
…   1.33084   0.0661961     ‡
```

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

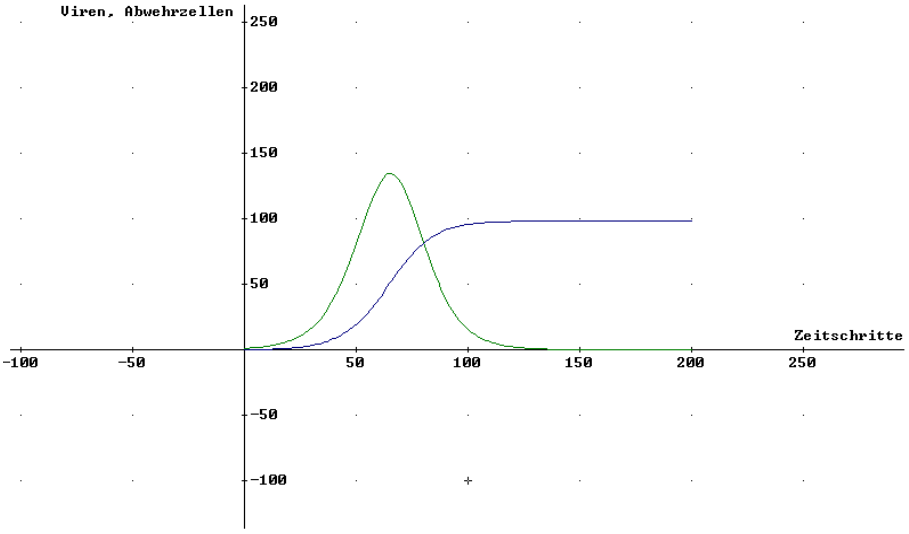
```
ITERATES([t + 1, NV(v, a), NA(v, a)], [t, v, a], [0, 1, 0], 3)

„ 0      1            0       †
| 1     1.1         0.02      |
| 2   1.20995   0.0419991     |
… 3   1.33084   0.0661961     ‡
```

To be able to show the time-development of the system, you must create two tables: one table time-viruses, the other time-resistant cells. The output is restricted (Version 4.11) to the desired columns for plotting by using the command SUB SUB [column1, column2].

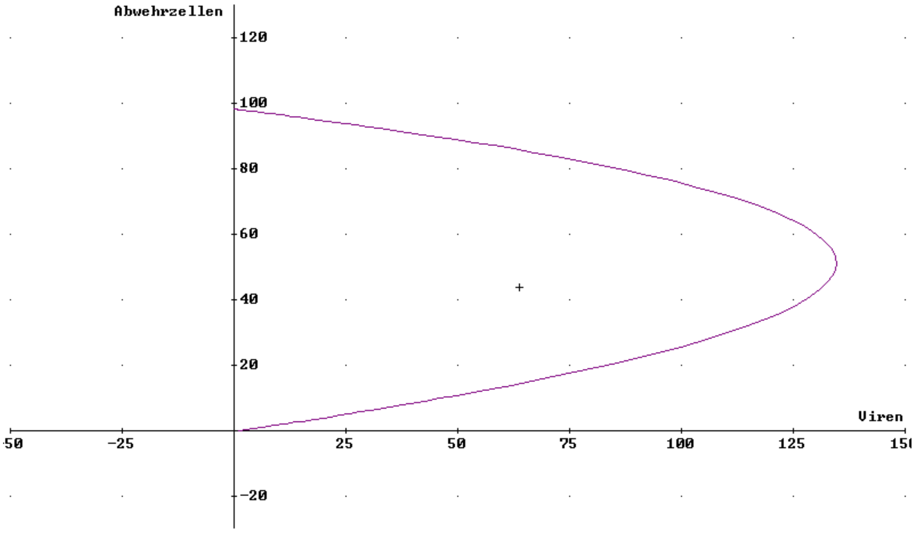
```
(ITERATES([t + 1, NV(v, a), NA(v, a)], [t, v, a], [0, 1, 0], 200))™™[1, 2]
(ITERATES([t + 1, NV(v, a), NA(v, a)], [t, v, a], [0, 1, 0], 200))™™[1, 3]
```

The result of the simulation can be reviewed in the graphic window



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

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



Simulation 2
If you want to work with more mutants, the shown method is only partly suitalbe, because you need at least the double number of iterative equations than there are mutants.So that the number of mutants does not play a role on the one hand and to create a clearly structured simulation it is recommended to express the current condition of the system in the form of a list (a vector) z. It shall contain the step of time, for each mutant and its type of resistant cell the respective equations and finally the total numbers of viruses and resistant cells. Additionally a second list (a second vector) mdata shall indicate the steps of time of the appearing mutants. This vector can be - if you want to - generated at random.

```
"HIV-Simulation"

"(c) by Josef Lechner, Mai 1999"

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

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

```
mdata := [1, 5, 10]
```

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

```
z0 := VECTOR(0, i, 1, 2·mz + 3)
```

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

```
          mz
SV(z) :=  ¤   z
          k=1  2·k

          mz
SA(z) :=  ¤   z
          k=1  2·k + 1
```

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

```
NV(z, i) := IF(z  = mdata   , 1, z  + z ·(r - p·z     ))
                1          i/2     i    i        i + 1

NA(z, i) := z      + k·z  - u·SV(z)·z
             i + 1      i             i + 1

NZ(z) := APPEND(„z  + 1†,
                 … 1    ‡

          APPEND(VECTOR([NV(z, i), NA(z, i)], i, 2, 2·mz, 2)),[SV(z), SA(z)])
```

Again the calculation is carried out by means of the ITERATES-command. This can be done in a surprisingly simple manner. The first function *NZ(z)* calculates the next using the current condition of the system, z contains the respective current condition, z_0 the initial condition and the last parameter indicates the number of iterations.

```
ITERATES(NZ(z), z, z0, 15)
```

provides the following table:

```
„  0     0         0          0          0          0          0          0          0       †
|                                                                                          |
|  1     0         0          0          0          0          0          0          0       |
|                                                                                          |
|  2     1         0          0          0          0          0          0          0       |
|                                                                                          |
|  3    1.1       0.02        0          0          0          0          1          0       |
|                                                                                          |
|  4  1.20995  0.0419991      0          0          0          0         1.1       0.02      |
|                                                                                          |
|  5  1.33084  0.0661961      0          0          0          0       1.20995  0.0419991   |
|                                                                                          |
|  6  1.46375  0.0928096      1          0          0          0       1.33084  0.0661961   |
|                                                                                          |
|  7  1.60986  0.122075      1.1       0.02         0          0       2.46375  0.0928096   |
|                                                                                          |
|  8  1.77045  0.154259   1.20995  0.0419978      0          0       2.70986  0.142075    |
|                                                                                          |
|  9  1.94695  0.189650   1.33084  0.0661919      0          0       2.98041  0.196257    |
|                                                                                          |
| 10  2.14091  0.228564   1.46375  0.0928002      0          0       3.27780  0.255842    |
|                                                                                          |
| 11  2.35402  0.271350   1.60986  0.122062       1          0       3.60467  0.321364    |
|                                                                                          |
| 12  2.58815  0.318376   1.77045  0.154235      1.1       0.02      4.96388  0.393412    |
|                                                                                          |
| 13  2.84531  0.370070   1.94695  0.189610   1.20995  0.0419956  5.45860  0.492611    |
|                                                                                          |
| 14  3.12774  0.426887   2.14091  0.228504   1.33084  0.0661846  6.00222  0.601676    |
|                                                                                          |
… 15  3.43784  0.489329   2.35402  0.271262   1.46375  0.0927841  6.59950  0.721576    ‡
```

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

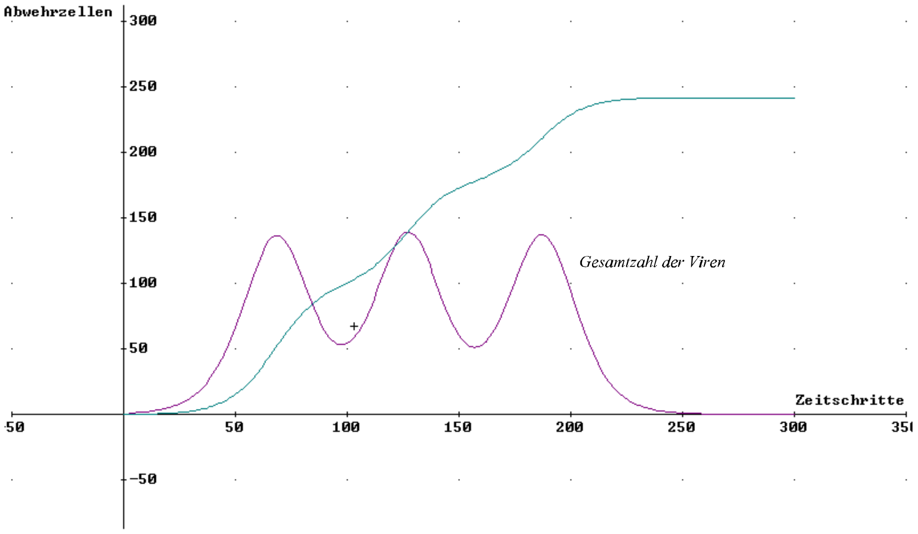
Example 1: 3 Mutants

Choosing model parameters

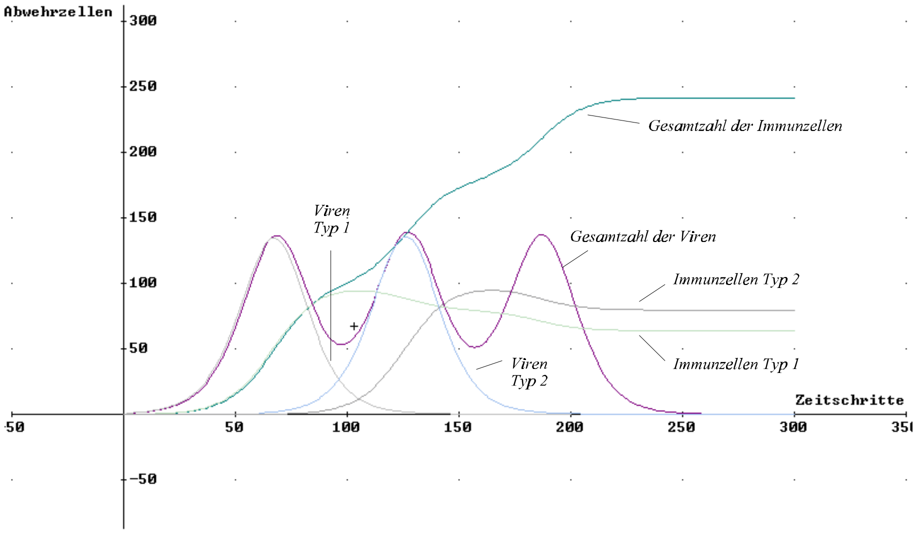
```
[r := 0.1, p := 0.002, k := 0.02, u := 0.00004, mz := 3]
mdata := [1, 60, 120]
```

Calculating the model

```
(ITERATES(NZ(z), z, z0, 300))™™[1, 8]
(ITERATES(NZ(z), z, z0, 300))™™[1, 9]
```

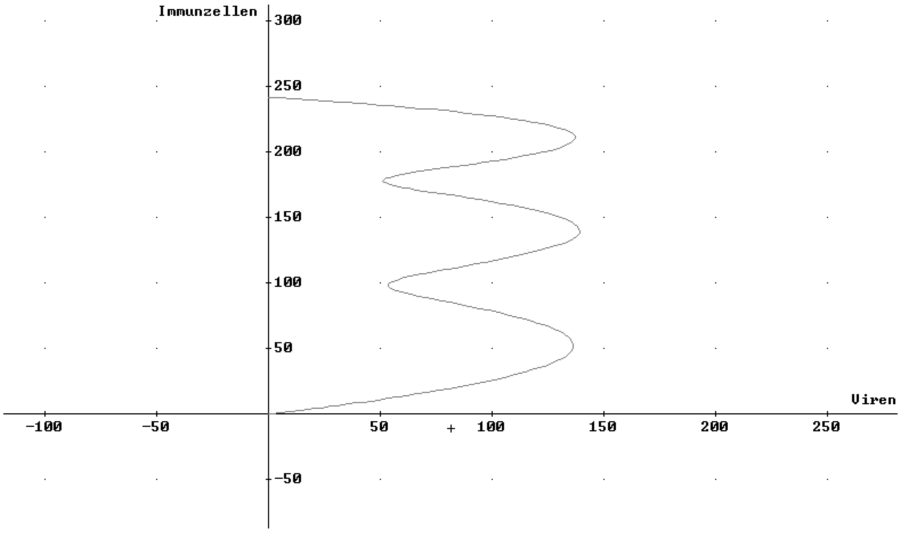


The development of the individual mutants and the immune cells respectively can be observed pretty well:



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

```
(ITERATES(NZ(z), z, z0, 300))™™[8, 9]
```



Example 2 - 11 Mutants

Choosing model parameters:

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

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

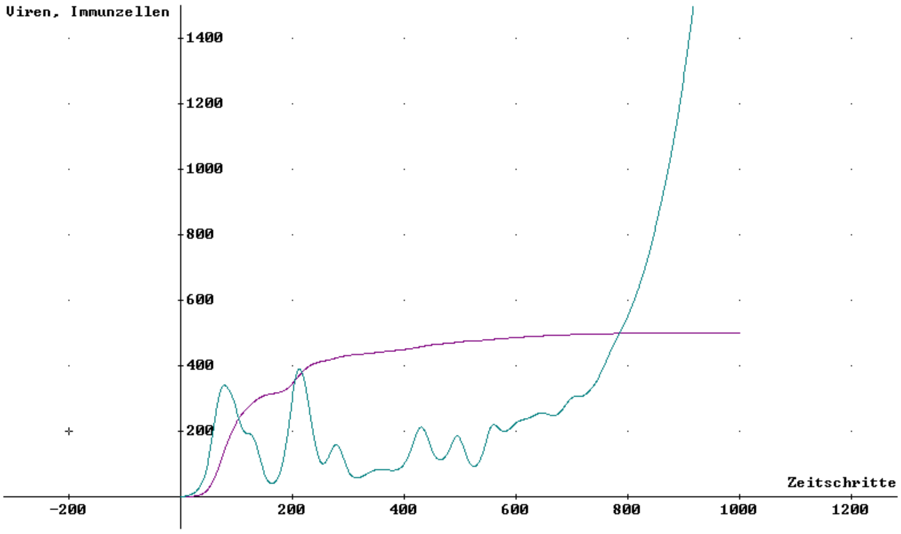
This list should be compiled at random, which can be achieved in the following way:

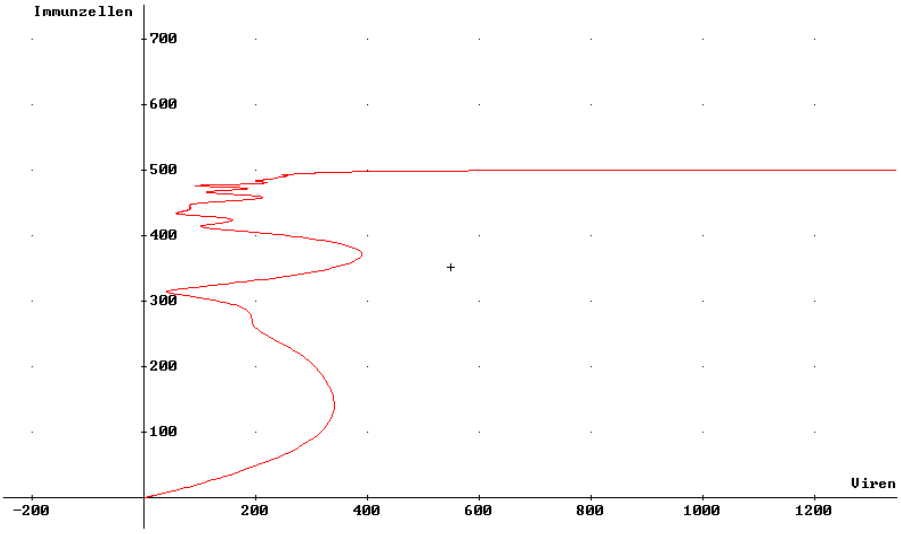
```
"Vector of mutants generated at random"
MUTANTS(mz, periods) :=
            APPEND([1], VECTOR(RANDOM(periods) + 1, i, 1, mz - 1))
mdata := MUTANTS(3, 200)
mdata := [1, 89, 160]
mdata := [1, 106, 185]
```

The modul mutants creates a list wíth the starting times, when you provide the number of mutants and the time span within which they shall appear. Every evaluation leads to a new random list.

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

```
v := 2·mz + 2
a := 2·mz + 3
(ITERATES(NZ(z), z, z0, 1000))™™[1, v]
(ITERATES(NZ(z), z, z0, 1000))™™[1, a]
(ITERATES(NZ(z), z, z0, 1000))™™[v, a]
"Calculation time ca.270s using PC486,120MHz)"
"Calculation time ca.60s using Pentium II PC,333MHz)"
```





Literature

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

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

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

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