

**Pendulum – Quelltexte einiger Programme**

(Reference: Lindgren, Kai: Inverted Pendulum, Electrical Engineering, Helsinki Polytechnic)

**Programm mkstxt( ) - Simulation der Zahlenfolgen****(Approximation des Dgl-Systems durch Differenzgl. / Rekursionsformeln)****Das Programm generiert mit Zeichenkettenbefehlen die notwendigen Eingaben für das Zahlenfolge-Menü auf Grundlage der Matrizen aus dem Dgl.-System und startet die Tabellierung und graphische Darstellung der Zahlenfolgen.**

```
'mkstxt
'ViewWindow 0,100,1,-20,20,1
ClrText
DelVar b
[[a][b][c]] $\rightarrow$ X
```

```
SeqType "an+1a0"
'SeqType "bn+1b0"
'SeqType "cn+1c0"
'SeqSelOn an+1
'SeqSelOn bn+1
'SeqSelOn cn+1
```

```
local dm,i,j,txt
```

```
rowDim(A) $\rightarrow$ dm
```

```
(A-B $\times$ K)  $\rightarrow$ left
'print "left"
'print left
'print "aux"
For 1  $\rightarrow$  i To dm Step 1
  ExpToStr X[i,1],aux1
  StrLeft aux1,1,aux
  StrJoin aux,"n+tstep $\times$ (" ,aux
  For 1  $\rightarrow$  j To dm Step 1
    StrJoin aux,"(",aux
    ExpToStr left[i,j],aux1
    StrJoin aux,aux1,aux
    StrJoin aux," $\times$ ",aux
    ExpToStr X[j,1],aux1
    StrLeft aux1,1,aux1
    StrJoin aux,aux1,aux
    StrJoin aux,"n",aux
    If j<dm
      Then
        StrJoin aux,"+",aux
    IfEnd
```

```

'print aux
Next
StrJoin aux,")",aux
ExpToStr X[i,1],aux1
StrLeft aux1,1,aux1
StrJoin aux1,"_{n+1}",aux1
'print aux
'print aux1
aux ↵ #aux1
'1 ↵ a_0
'-1 ↵ b_0
ExpToStr X[i,1],aux1
StrLeft aux1,1,aux1
StrJoin aux1,"_0",aux1
'print aux1
init[i,1] ↵ #aux1
Next
'Stop

```

```

SeqSelOn a_{n+1}
SeqSelOn b_{n+1}
SeqSelOn c_{n+1}
0 ↵ SqStart
100 ↵ SqEnd
DispSeqTbl
Pause
DrawSeqCon

```

```

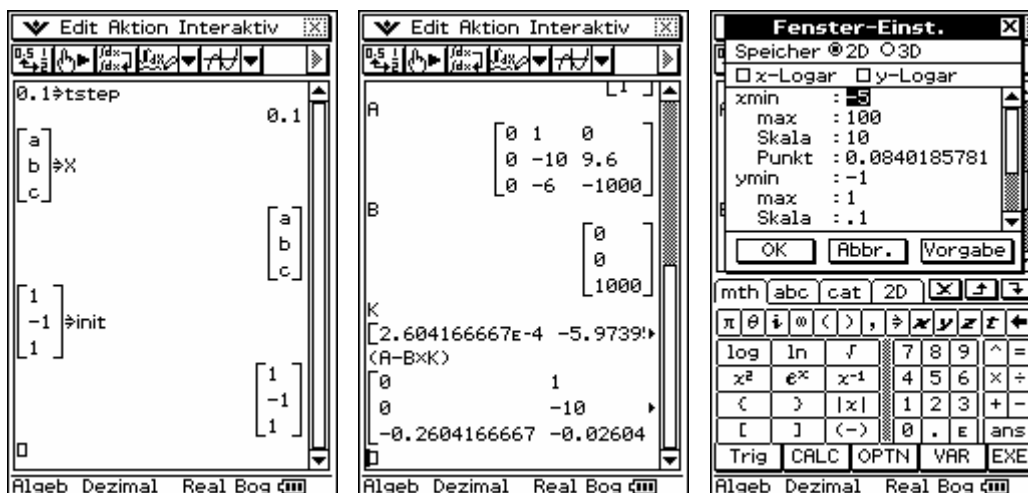
'strToExp(plambda) ↵ rest
'ExpToStr rest,tulos

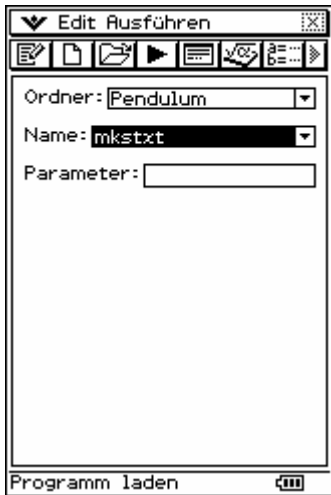
```

Return

### Bilder für das dreidimensionale Problem:

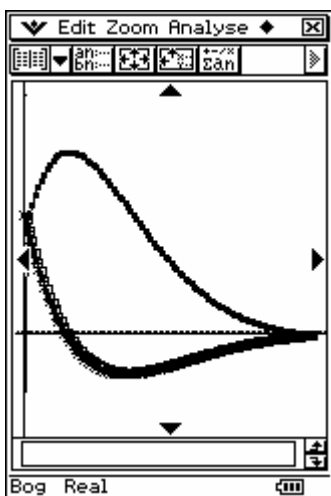
Zur Erzeugung der notwendigen Matrizen A, B, K wird zuerst das Programm **analyse3( )** gestartet. Die Schrittweite **tstep** sowie der Vektor **X** der Unbekannten und der Vektor **init** der Anfangsbedingungen sind vorzugeben, nachdem analyse3( ) gelaufen ist.





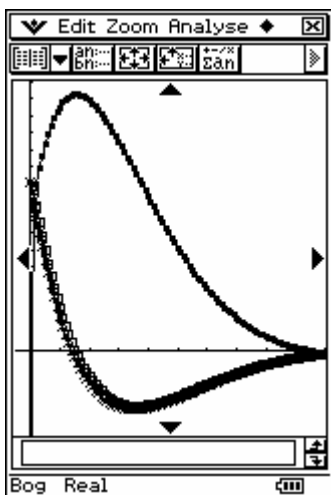
n	a <sub>n</sub>	b <sub>n</sub>
0	1	-1
1	0.9	0.96
2	0.996	0.8607
3	1.082	0.7669
4	1.1587	0.6785
5	1.2266	0.5952
6	1.2861	0.5169
7	1.3378	0.4434
8	1.3821	0.3745
9	1.4196	0.31
10	1.4506	0.2497
11	1.4756	0.1934
12	1.4949	0.1411
13	1.509	0.0924
14	1.5183	0.0473

n	b <sub>n</sub>	c <sub>n</sub>
0	-1	1
1	0.96	0.8965
2	0.8607	0.7989
3	0.7669	0.7068
4	0.6785	0.62
5	0.5952	0.5385
6	0.5169	0.4619
7	0.4434	0.3901
8	0.3745	0.3229
9	0.31	0.2601
10	0.2497	0.2015
11	0.1934	0.147
12	0.1411	0.0963
13	0.0924	0.0493
14	0.0473	5E-3



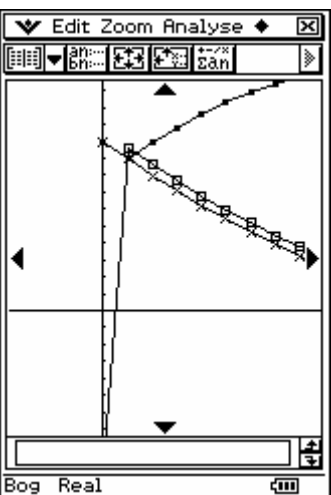
Rekursiv  Explizit  
  $a_{n+1} = a_n + \text{tstep} \cdot (0 \cdot a_n + 1)$   
 $a_0 = 1$   
  $b_{n+1} = b_n + \text{tstep} \cdot (0 \cdot a_n - 10)$   
 $b_0 = -1$   
  $c_{n+1} = c_n + \text{tstep} \cdot (-81380 / 312500)$   
 $c_0 = 1$

Rekursiv  Explizit  
  $a_{n+1} = 4 \cdot (0 \cdot a_n + 1 \cdot b_n + 0 \cdot c_n)$   
 $a_0 = 1$   
  $b_{n+1} = 4 \cdot a_n - 10 \cdot b_n + \frac{48}{5} \cdot c_n$   
 $b_0 = -1$   
  $c_{n+1} = \frac{301043}{3000051} \cdot b_n - \frac{4}{5} \cdot c_n$   
 $c_0 = 1$

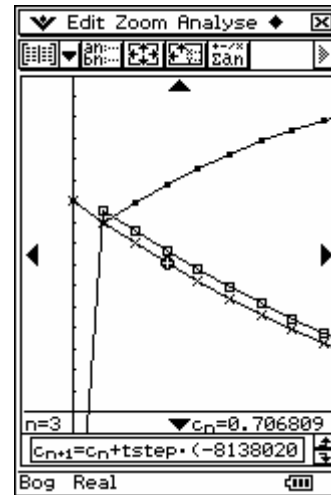
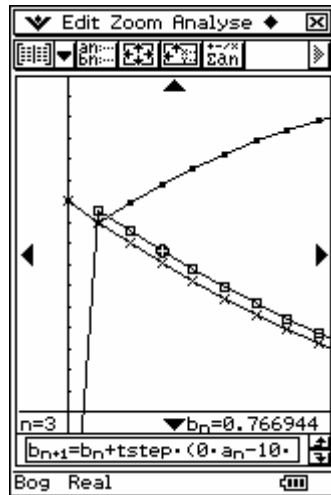
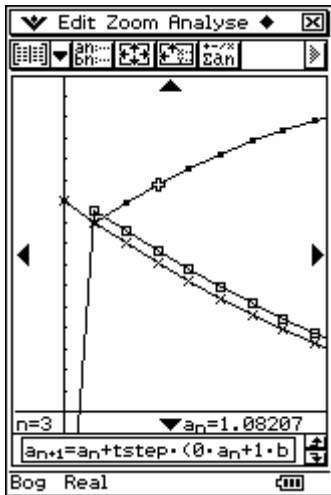


**Fenster-Einst.**  
 Speicher @2D @3D  
 x-Logar  y-Logar  
 xmin : 5  
 max : 100  
 Skala : 10  
 Punkt : 0.6818181818  
 ymin : -0.5  
 max : 1.6  
 Skala : 0.1  
 OK Abbr. Vorgabe

mth abc cat 2D  
 π θ i ∅ (< >), ÷ × √ ±  
 log ln √ 7 8 9 ^ =  
 x² eˣ x⁻¹ 4 5 6 × +  
 ( ) |x| 1 2 3 + -  
 [ ] (-) 0 . E ans  
 Trig CALC OPTN VAR EXE



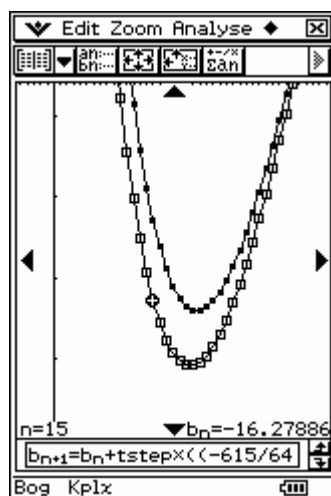
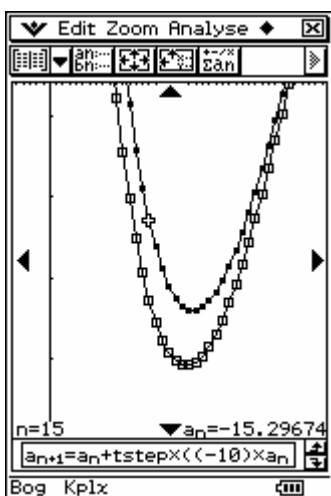
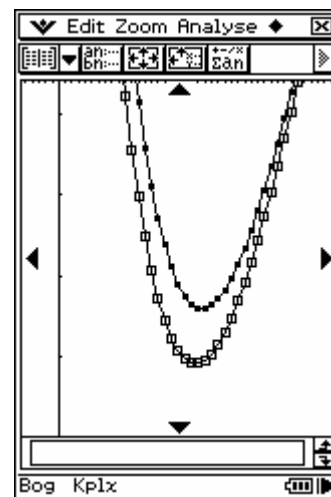
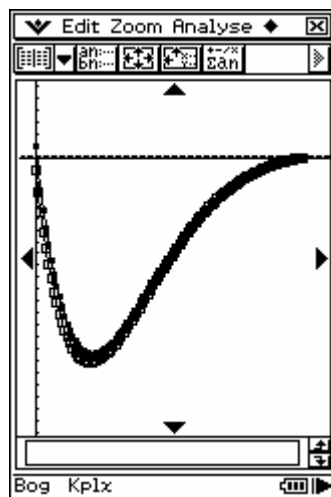
Vergrößerung: Es werden drei Zahlenfolgen dargestellt!



### Bilder für das zweidimensionale Problem:

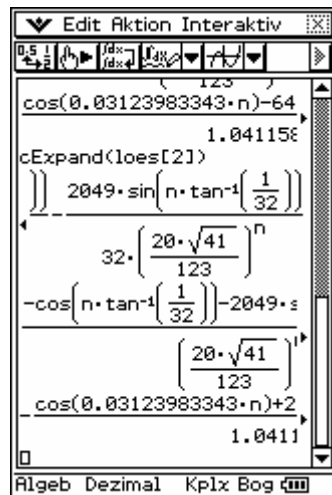
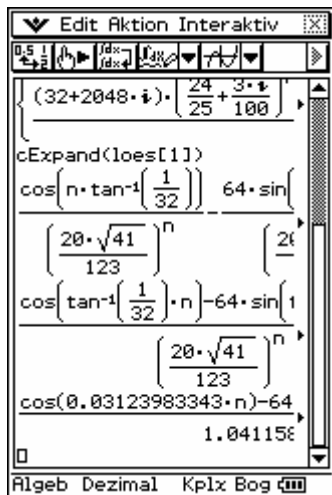
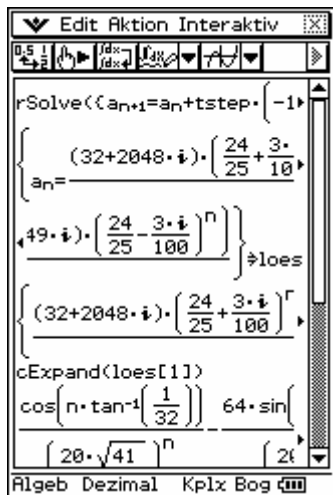
Zur Erzeugung der notwendigen Matrizen A, B, K wird zuerst das Programm `analyse2()` gestartet. Die Schrittweite `tstep` sowie der Vektor `X` der Unbekannten und der Vektor `init` der Anfangsbedingungen sind vorzugeben, nachdem `analyse2()` gelaufen ist.

n	a <sub>n</sub>	b <sub>n</sub>
0	1	-1
1	-0.96	-2.88
2	-2.765	-4.608
3	-4.424	-6.191
4	-5.943	-7.635
5	-7.33	-8.949
6	-8.591	-10.13
7	-9.732	-11.21
8	-10.76	-12.17
9	-11.68	-13.02
10	-12.5	-13.78
11	-13.23	-14.44
12	-13.86	-15.02
13	-14.42	-15.51
14	-14.89	-15.93



Rekursiv  Explizit   
  $a_{n+1}=a_n+tstep \cdot (-10 \cdot a_n)$   
 $a_0=1$   
  $b_{n+1}=b_n+tstep \cdot (-\frac{615}{64} \cdot a_n)$   
 $b_0=-1$   
  $c_{n+1}=c_n+tstep \cdot (-\frac{81380}{312500})$   
 $c_0=1$

Mit dem `rSolve`-Befehl können  $a_n$  und  $b_n$  explizit ausgerechnet werden:



$$a_n = (\cos(0.03123983343117*n)-64*\sin(0.03123983343117*n)) / 1.041158412591^n$$

$$b_n = -(\cos(0.03123983343117*n)+2049*\sin(0.03123983343117*n)) / 1.041158412591^n$$

Mithilfe des Programms **mkdtx()** wird der **dSolve**-Befehl generiert, um das dazu gehörende Dgl-System zu lösen. Das Programm **mkdtx()** basiert ebenfalls auf Zeichenketten-Befehlen zur Erzeugung des **dSolve**-Befehls, einschließlich der kompletten Syntax des Befehls. Zuvor muss wieder **analyse2()** gestartet werden, um die benötigten Matrizen bereitzustellen.

### Programm **mkdtx()** - Symbolische Lösung des Dgl-Systems

```
'mkdtx
ClrText
[[1],[-1]] => init

[[x],[y]] => X
local dm,i,txt

rowDim(A) => dm
"" => tulos
(A-B*X)*X => aux
'print "aux"
'print aux

StrJoin tulos,"{",tulos
For 1 => i To dm Step 1
  ExpToStr X[i,1],txt
  StrJoin tulos,txt,tulos
  StrJoin tulos,"=",tulos
  ExpToStr aux[i,1],txt
  StrJoin tulos,txt,tulos
  If i<dm
    Then
      StrJoin tulos,",",tulos
  IfEnd
```

```

    'print "tulos"
    'print tulos
Next
'Pause
StrJoin tulos," },t,{",tulos
'print "tulos"
'print tulos
'Pause
For 1 ⇨i To dm Step 1
    ExpToStr X[i,1],txt
    StrJoin tulos,txt,tulos
    If i<dm
        Then
            StrJoin tulos,",",tulos
        IfEnd
    'print "tulos"
    'print tulos
Next

StrJoin tulos," }," ,tulos
'print "tulos"
'print tulos
'Pause
For 1 ⇨i To dm Step 1
    StrJoin tulos,"t=0," ,tulos
    ExpToStr X[i,1],txt
    StrJoin tulos,txt,tulos
    StrJoin tulos,"=",tulos
    ExpToStr init[i,1],txt
    StrJoin tulos,txt,tulos
    If i<dm
        Then
            StrJoin tulos,",",tulos
        IfEnd
Next
'print "tulos"
'print tulos
'Pause
StrJoin "dSolve(",tulos,tulos
StrJoin tulos,")",tulos
'print "tulos"
'print tulos

'strToExp(plambda) ⇨rest
'ExpToStr rest,tulos
Return

```

**Erzeugte Zeichenkette, abgespeichert unter tulos:**

"dSolve({x'=-10\*x+48\*y/5,y'=-615\*x/64+46\*y/5},t,{x,y},t=0,x=1,t=0,y=-1)"

**Lösung des Dgl-Systems im Main-Menü:**

$$\left\{ \begin{aligned} x &= e^{((-2t)/5)} \cdot \cos((3t)/10) - 64 \cdot e^{((-2t)/5)} \cdot \sin((3t)/10), \\ y &= -e^{((-2t)/5)} \cdot \cos((3t)/10) - ((2049 \cdot e^{((-2t)/5)}) \cdot \sin((3t)/10)) / (32) \end{aligned} \right\}$$

▼ Edit Aktion Interaktiv

tulos  
 "dSolve((x'=-10\*x+48\*y/5, y'  
 strToExp(tulos)⇒dgl

$$\left( \frac{3 \cdot t}{10}, y = -e^{-\frac{2 \cdot t}{5}} \cdot \cos\left(\frac{3 \cdot t}{10}\right) - 64 \cdot e^{-\frac{2 \cdot t}{5}} \cdot \sin\left(\frac{3 \cdot t}{10}\right) \right)$$

Define y1(x)= $e^{-\frac{2 \cdot x}{5}} \cdot \cos\left(\frac{3 \cdot x}{10}\right)$  done

Define y2(x)= $-e^{-\frac{2 \cdot x}{5}} \cdot \cos\left(\frac{3 \cdot x}{10}\right) - \frac{2049 \cdot e^{-\frac{2 \cdot x}{5}} \cdot \sin\left(\frac{3 \cdot x}{10}\right)}{32}$  done

Algeb Dezimal Kplx Bog

▼ Edit Aktion Interaktiv

Define y1(x)= $e^{-\frac{2 \cdot x}{5}} \cdot \cos\left(\frac{3 \cdot x}{10}\right)$  done

Define y2(x)= $-e^{-\frac{2 \cdot x}{5}} \cdot \cos\left(\frac{3 \cdot x}{10}\right) - 64 \cdot \sin\left(\frac{3 \cdot x}{10}\right)$  done

simplify(y1(x))  
 $e^{-\frac{2 \cdot x}{5}} \cdot \left( \cos\left(\frac{3 \cdot x}{10}\right) - 64 \cdot \sin\left(\frac{3 \cdot x}{10}\right) \right)$

simplify(y2(x))  
 $-e^{-\frac{2 \cdot x}{5}} \cdot \left( 32 \cdot \cos\left(\frac{3 \cdot x}{10}\right) + 2049 \cdot \sin\left(\frac{3 \cdot x}{10}\right) \right)$

Algeb Dezimal Kplx Bog

▼ Edit Aktion Interaktiv

Define y1(x)= $e^{-\frac{2 \cdot x}{5}} \cdot \cos\left(\frac{3 \cdot x}{10}\right)$  done

Define y2(x)= $-e^{-\frac{2 \cdot x}{5}} \cdot \cos\left(\frac{3 \cdot x}{10}\right) - 64 \cdot \sin\left(\frac{3 \cdot x}{10}\right)$  done

simplify(y1(x))  
 $e^{-\frac{2 \cdot x}{5}} \cdot \left( \cos\left(\frac{3 \cdot x}{10}\right) - 64 \cdot \sin\left(\frac{3 \cdot x}{10}\right) \right)$

simplify(y2(x))  
 $-e^{-\frac{2 \cdot x}{5}} \cdot \left( 32 \cdot \cos\left(\frac{3 \cdot x}{10}\right) + 2049 \cdot \sin\left(\frac{3 \cdot x}{10}\right) \right)$

Algeb Dezimal Kplx Bog

