Prof. Dr. L. Paditz 31.07.2006

Pendulum – Quelltexte einiger Programme

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

Programm mkstxt( ) - Simulation der Zahlenfolgen

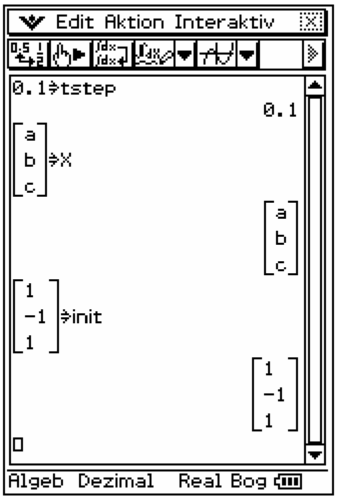
(Approximation des Dgl-Systems durch Diffenrenzengln. / Rekursionsformeln)

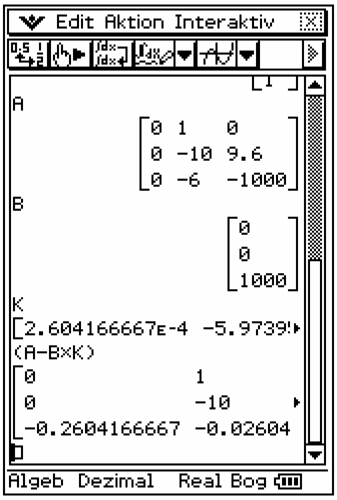
Das Programm generiert mit Zeichenkettenbefehlen die notwendigen Eingaben für das Zahlenfolge-Menü auf Grundlage der Matrizen aus dem Dgl.-System und startet die Tabellierung und graphische Darstellung der Zahlenfolgen.

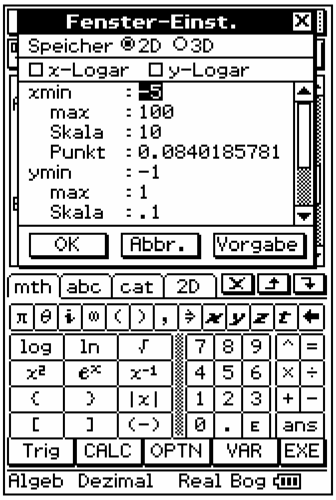
```
'mkstxt
'ViewWindow 0,100,1,-20,20,1
ClrText
DelVar b
[[a][b][c]]⇛X

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)⇛dm

(A-B×K) ⇛left
'print "left"
'print left
'print "aux"
For 1 ⇛ i To dm Step 1
   ExpToStr X[i,1],aux1
   StrLeft aux1,1,aux
   StrJoin aux,"n+tstep×(",aux
   For 1 ⇛ j To dm Step 1
     StrJoin aux,"(",aux
     ExpToStr left[i,j],aux1
     StrJoin aux,aux1,aux
     StrJoin aux,")×",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
```

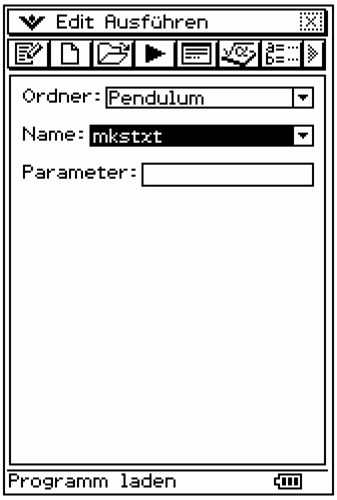
```
      'print aux
    Next
    StrJoin aux,")",aux
    ExpToStr X[i,1],aux1
    StrLeft aux1,1,aux1
    StrJoin aux1,"n+1",aux1
    'print aux
    'print aux1
    aux ⇒#aux1
    '1 ⇒a0
    '-1 ⇒b0
    ExpToStr X[i,1],aux1
    StrLeft aux1,1,aux1
    StrJoin aux1,"0",aux1
    'print aux1
    init[i,1]⇒#aux1
Next
'Stop

SeqSelOn an+1
SeqSelOn bn+1
SeqSelOn cn+1
0 ⇒SqStart
100 ⇒SqEnd
DispSeqTbl
Pause
DrawSeqCon

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

Return
```

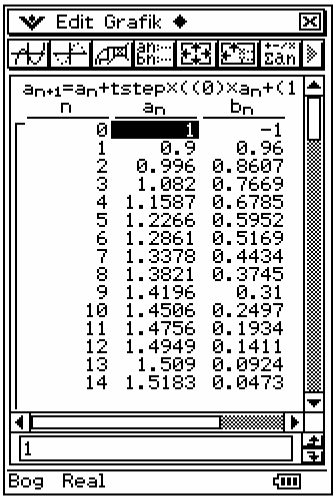
Bilder für das dreidimensionale Problem:
Zur Erzeugung der notwendigen Matrizen A, B, K wird zuerst das Programm **analyse3( )** gestartet. Die Schrittweite **tstep** sowie der Vektor **X** der Unbekannten und der Vektor **init** der Anfangsbedingungen sind vorzugeben, nachdem analyse3( ) gelaufen ist.

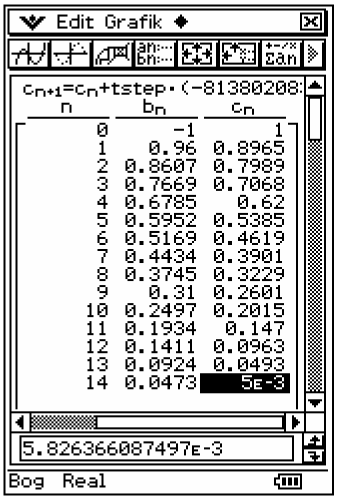


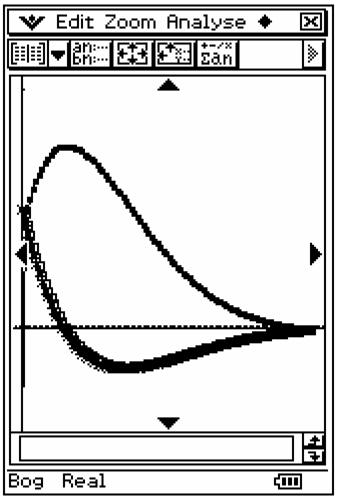


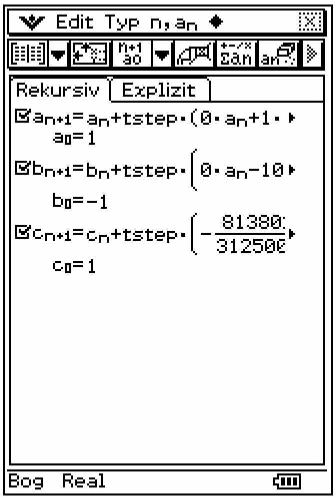


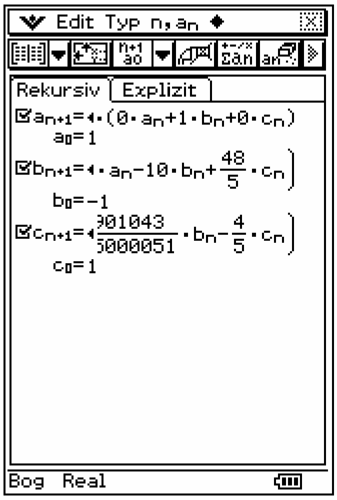


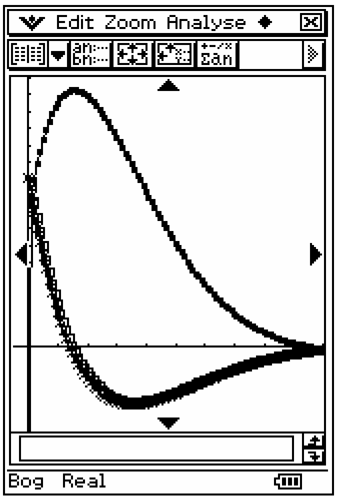


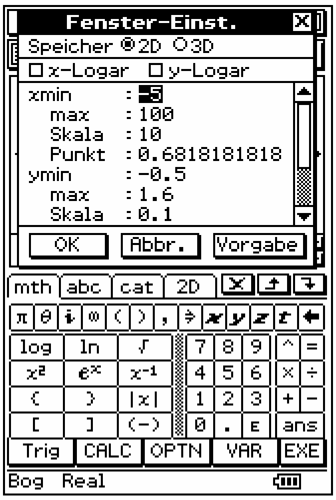


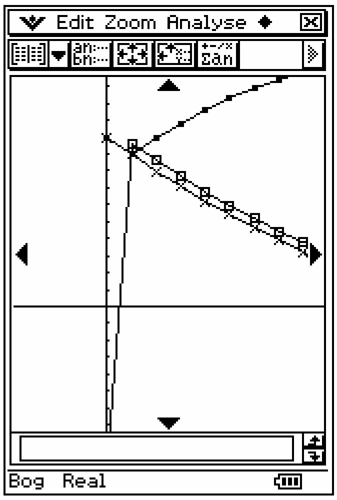




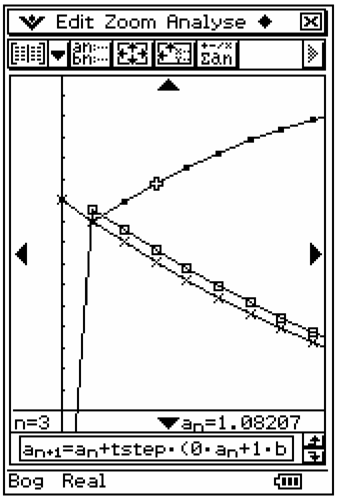


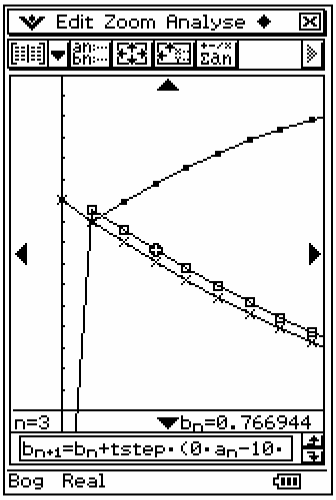


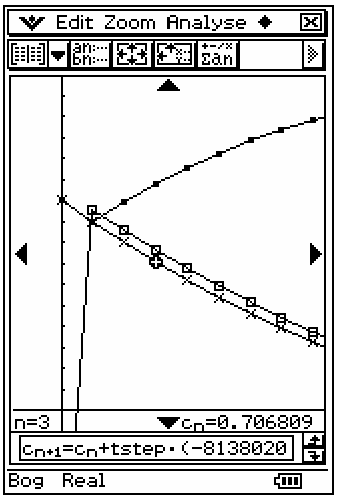




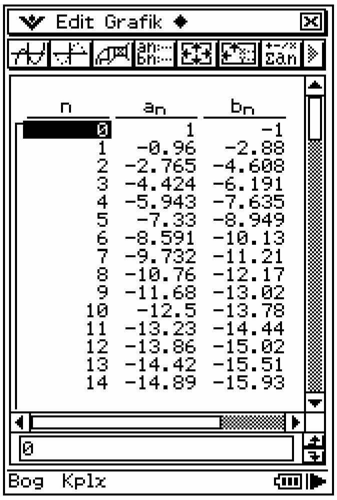
Vergrößerung: Es werden drei Zahlenfolgen dargestellt!

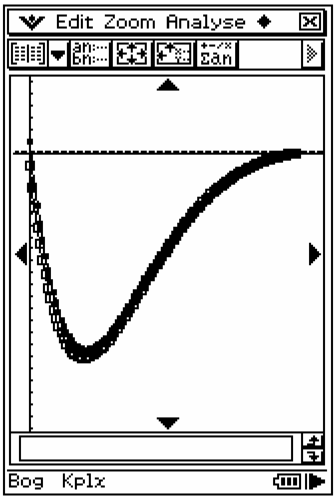


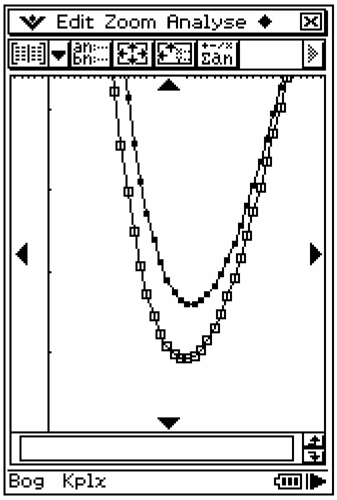


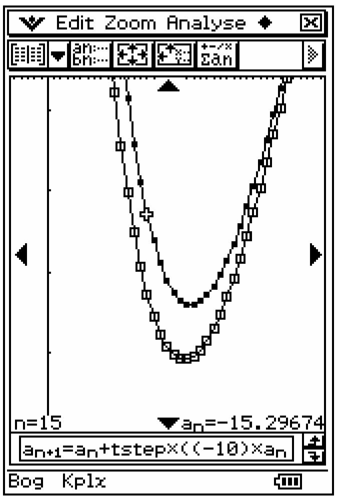


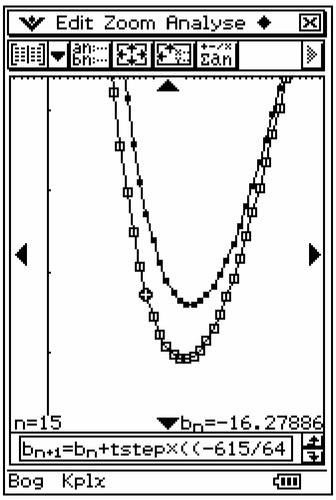
Bilder für das zweidimensionale Problem:
Zur Erzeugung der notwendigen Matrizen A, B, K wird zuerst das Programm **analyse2( )** gestartet. Die Schrittweite **tstep** sowie der Vektor **X** der Unbekannten und der Vektor **init** der Anfangsbedingungen sind vorzugeben, nachdem analyse2( ) gelaufen ist.

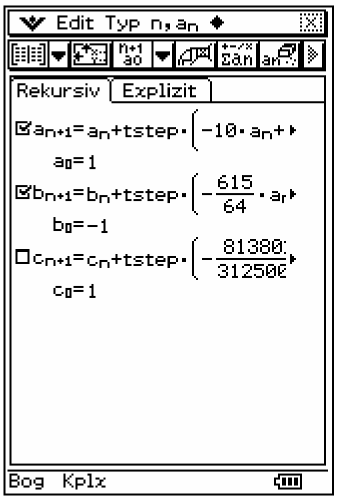




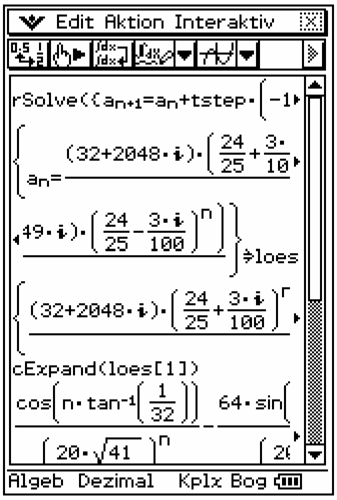


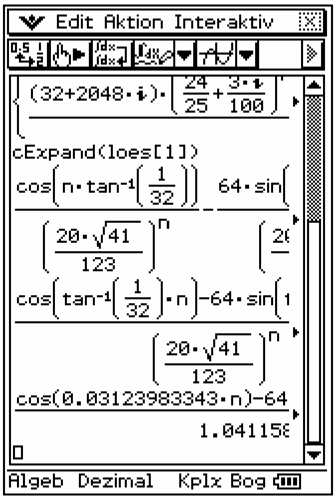


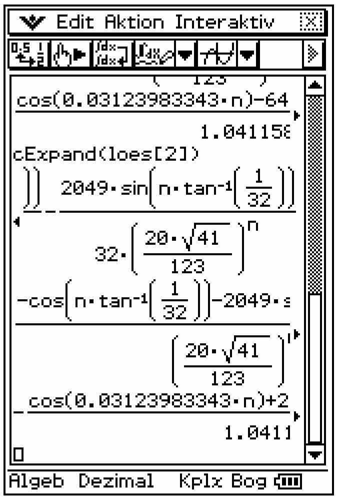




Mit dem **rSolve**-Befehl können a_n und b_n explizit ausgerechnet werden:







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 **mkdtxt( )** wird der **dSolve**-Befehl generiert, um das dazu gehörende Dgl-System zu lösen. Das Programm mkdtxt( ) 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 mkdtxt( ) - Symbolische Lösung des Dgl-Systems

```
'mkdtxt

ClrText
[[1],[-1]] ⇒init

[[x],[y]] ⇒X
local dm,i,txt

rowDim(A) ⇒dm
"" ⇒tulos
(A-B×K)×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:

{ x = *e*^(((-2*t)/(5)))*cos(((3*t)/(10))) - 64*_e_^(((-2*t)/(5)))*sin(((3*t)/(10))),
y = -*e*^(((-2*t)/(5)))*cos(((3*t)/(10))) - ((2049*_e_^(((-2*t)/(5)))*sin(((3*t)/(10))))/(32)) }