

Exercise 1

Aufgabe 1.2 - GLSys eindeutig loesbar

```
> restart;with(LinearAlgebra):  
eqs := [x[1]+2*x[2]+4*x[3] = 5, 2*x[1]+2*x[2]+x[3] = 4, 3*x  
[1]+2*x[2] = 1];  
unknown:=[x[1], x[2], x[3]];
```

Solve:

```
res := solve(eqs, unknown);
```

Linear Solve (mit erweiterter Koeffizientenmatrix):

```
koeffMatrix:=GenerateMatrix(eqs, unknown) [1];  
erwKoeffMatrix:=GenerateMatrix(eqs, unknown, augmented=true);  
resVek:=LinearSolve(erwKoeffMatrix);
```

Da der Rang der erweiterten Koeffizientenmatrix gleich der Koeffizientenmatrix ist, ist das Gleichungssystem loesbar.

```
"Rang(erwKoeffMatrix)"=Rank(erwKoeffMatrix);  
"Rang(koeffMatrix)"=Rank(koeffMatrix);
```

Wenn der Rang der beiden Matrizen auch noch gleich der Anzahl an unbekannten ist, hat das System genau eine Loesung.

(Rang wird bestimmt, indem mittels gaussischem Eliminationsverfahrens eine aequivalente Matrix in Stufenform erstellt wird. Anschliessend wird die Anzahl der unabhaengigen Zeilen bestimmt, also die Zeilenvektoren, die ungleich 0 sind.)

```
"Anzahl der Unbekannten"=nops(unknown);
```

Wie auch in der Grafik ersichtlich, besitzt das System genau eine Loesung.

```
with(plots):with(plottools):  
plt3d:=plot3d({solve(eqs[1], x[1]), solve(eqs[2], x[1]), solve  
(eqs[3], x[1])}, x[2]=-10..10, x[3]=-10..10, transparency=  
0.2):  
point3d:=pointplot3d([resVek[2], resVek[3], resVek[1]], color=  
[red], axes=boxed, symbolsize=30, symbol=diagonalcross,  
thickness=4):  
display({plt3d, point3d}, orientation=[-50,-140]);
```

$$eqs := [x_1 + 2x_2 + 4x_3 = 5, 2x_1 + 2x_2 + x_3 = 4, 3x_1 + 2x_2 = 1]$$

$$unknown := [x_1, x_2, x_3]$$

$$res := \left[\left[x_1 = -4, x_2 = \frac{13}{2}, x_3 = -1 \right] \right]$$

$$koeffMatrix := \begin{bmatrix} 1 & 2 & 4 \\ 2 & 2 & 1 \\ 3 & 2 & 0 \end{bmatrix}$$

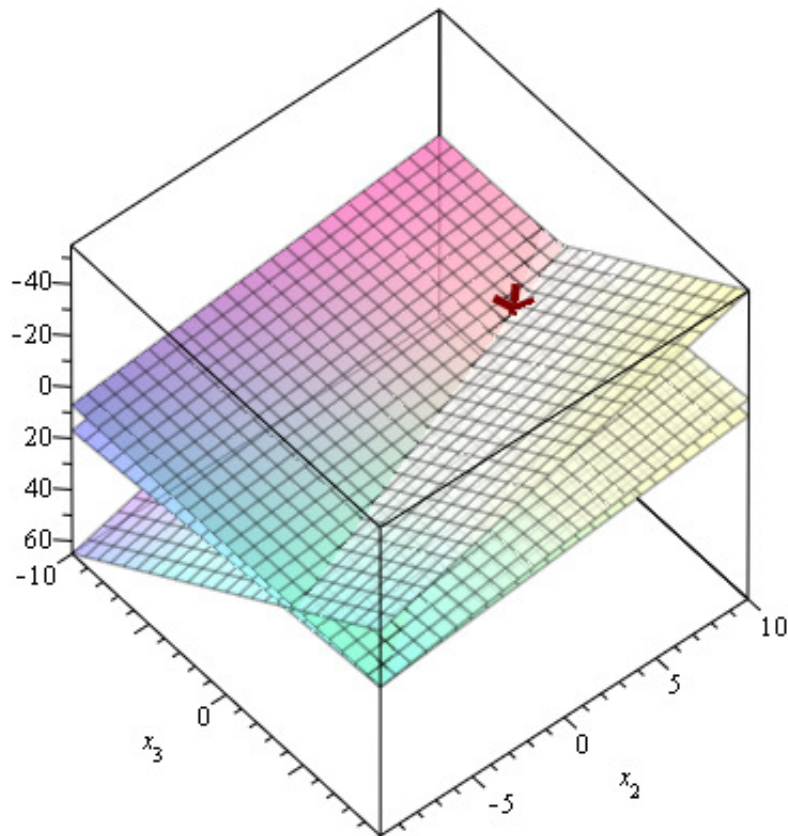
$$erwKoeffMatrix := \begin{bmatrix} 1 & 2 & 4 & 5 \\ 2 & 2 & 1 & 4 \\ 3 & 2 & 0 & 1 \end{bmatrix}$$

$$resVek := \begin{bmatrix} -4 \\ \frac{13}{2} \\ -1 \end{bmatrix}$$

"Rang(erwKoeffMatrix)" = 3

"Rang(koeffMatrix)" = 3

"Anzahl der Unbekannten" = 3



▼ Aufgabe 1.3 - GLSys mehrdeutig loesbar

```
> restart;with(LinearAlgebra):
eqs := [-x[1]-8*x[2]+2*x[3] = 1, 2*x[1]+13*x[2]-3*x[3] = 1,
3*x[2]-x[3] = -3];
unknown:=[x[1], x[2], x[3]];
```

Solve:

```
res := solve(eqs, unknown);
```

Linear Solve (mit erweiterter Koeffizientenmatrix):

```
koeffMatrix:=GenerateMatrix(eqs, unknown) [1];
erwKoeffMatrix:=GenerateMatrix(eqs, unknown, augmented=true);
resVek:=LinearSolve(erwKoeffMatrix);
```

Da der Rang der erweiterten Koeffizientenmatrix gleich der Koeffizientenmatrix ist, ist das Gleichungssystem lösbar.

```
"Rang(erwKoeffMatrix)"=Rank(erwKoeffMatrix);
"Rang(koeffMatrix)"=Rank(koeffMatrix);
```

Der Rang beider Matrizen ist jedoch nicht gleich der Anzahl an unbekannten, daher hat das System unendlich viele Lösungen.

```
"Anzahl der Unbekannten"=nops(unknown);
```

Wie auch in der Grafik ersichtlich, besitzt das System unendlich viele Lösungen.

```
with(plots):with(plottools):
plt3d:=plot3d({solve(eqs[1], x[3]),solve(eqs[2], x[3]),solve
(eqs[3], x[3])}, x[1]=-10..10, x[2]=-10..10,transparency=
0.2):
```

```
display({plt3d}, orientation=[-50,-140]);
```

$$eqs := [-x_1 - 8x_2 + 2x_3 = 1, 2x_1 + 13x_2 - 3x_3 = 1, 3x_2 - x_3 = -3]$$

$$unknown := [x_1, x_2, x_3]$$

$$res := [[x_1 = 5 - 2x_2, x_2 = x_2, x_3 = 3x_2 + 3]]$$

$$koeffMatrix := \begin{bmatrix} -1 & -8 & 2 \\ 2 & 13 & -3 \\ 0 & 3 & -1 \end{bmatrix}$$

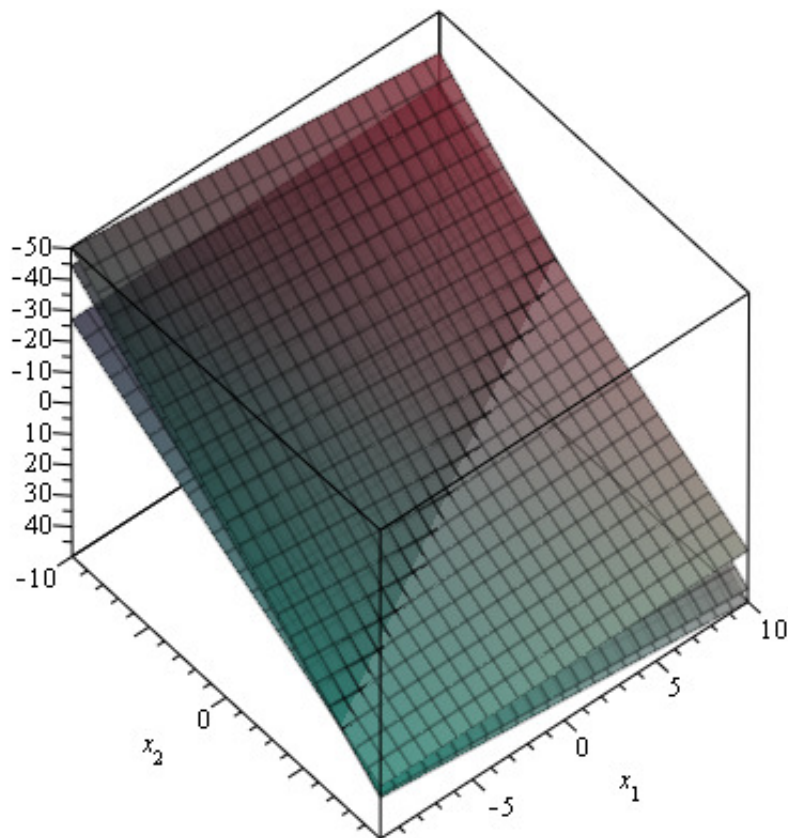
$$erwKoeffMatrix := \begin{bmatrix} -1 & -8 & 2 & 1 \\ 2 & 13 & -3 & 1 \\ 0 & 3 & -1 & -3 \end{bmatrix}$$

$$resVek := \begin{bmatrix} 5 - 2t_2 \\ -t_2 \\ 3t_2 + 3 \end{bmatrix}$$

```
"Rang(erwKoeffMatrix)"=2
```

```
"Rang(koeffMatrix)"=2
```

```
"Anzahl der Unbekannten"=3
```



▼ Aufgabe 1.4 - GLSys unloesbar

```
> restart;with(LinearAlgebra):
eqs := [-x[1]-8*x[2]+2*x[3]=2, 2*x[1]+13*x[2]-3*x[3]=1, 3*x[2]
-x[3]=3];
unknown:=[x[1], x[2], x[3]];
```

Solve:

```
res := solve(eqs, unknown);
```

Linear Solve (mit erweiterter Koeffizientenmatrix):

```
koefMatrix:=GenerateMatrix(eqs, unknown)[1];
erwKoeffMatrix:=GenerateMatrix(eqs, unknown, augmented=true);
resVek:=LinearSolve(erwKoeffMatrix);
```

Da der Rang der erweiterten Koeffizientenmatrix nicht gleich der Koeffizientenmatrix ist, ist das Gleichungssystem nicht loesbar.

```
"Rang(erwKoeffMatrix)"=Rank(erwKoeffMatrix);
"Rang(koefMatrix)"=Rank(koefMatrix);
```

Wie auch in der Grafik ersichtlich, besitzt das System keine Loesungen.

```
with(plots):with(plottools):
plt3d:=plot3d({solve(eqs[1], x[3]),solve(eqs[2], x[3]),solve
(eqs[3], x[3])}, x[1]=-10..10, x[2]=-10..10,transparency=
0.2):
```

```
display({plt3d}, orientation=[-50,-140]);
eqs := [-x1 - 8 x2 + 2 x3 = 2, 2 x1 + 13 x2 - 3 x3 = 1, 3 x2 - x3 = 3]
unknown := [x1, x2, x3]
res := [ ]
```

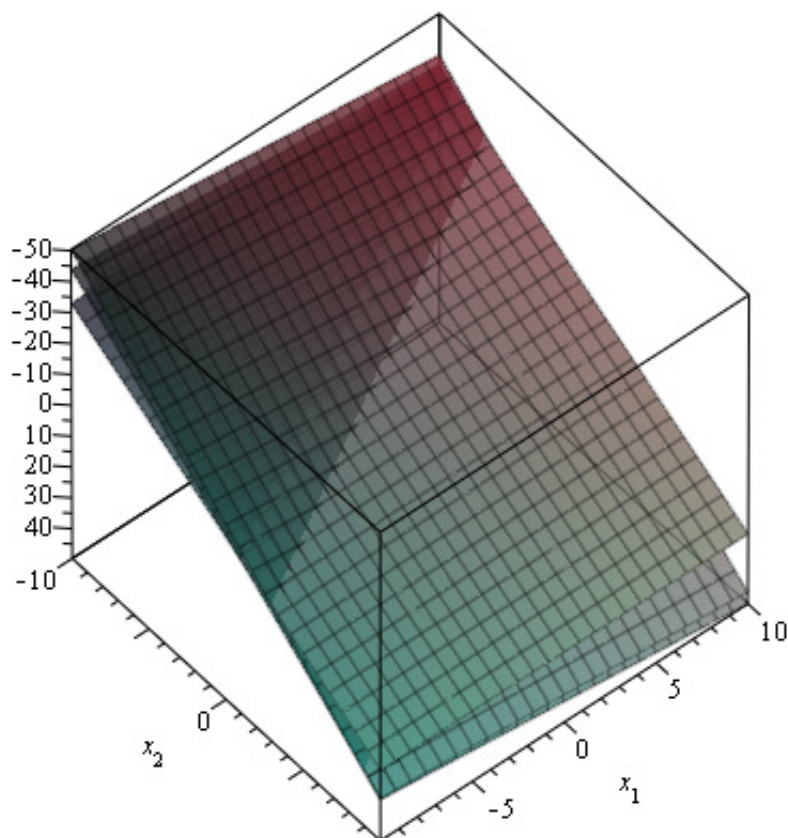
$$\text{coeffMatrix} := \begin{bmatrix} -1 & -8 & 2 \\ 2 & 13 & -3 \\ 0 & 3 & -1 \end{bmatrix}$$

$$\text{erwKoeffMatrix} := \begin{bmatrix} -1 & -8 & 2 & 2 \\ 2 & 13 & -3 & 1 \\ 0 & 3 & -1 & 3 \end{bmatrix}$$

Error, (in LinearAlgebra:-LinearSolve) inconsistent system

"Rang(erwKoeffMatrix)" = 3

"Rang(coeffMatrix)" = 2



▼ Aufgabe 1.5 - Singularitaet, Approximation

- > Das gesuchte Polynom muss an der Stelle $x=0.1$ dem Funktionswert $f(x)$ entsprechen. Aber nicht nur der Funktionswert, auch die erste und die zweite Ableitung beider Funktionen muss an dieser Stelle gleich sein.
Daher sind an Punkt $x=0.1$ drei Unbekannte, die das Polynom erfuellen muss. (Die

Funktion selbst und die erste und die zweite Ableitung)

Fuer den Punkt $x=-0.1$ gilt das Selbe, daher muss ein Gleichungssystem aufgestellt werden, das 6 Unbekannte loesen kann.

```
restart;
f:=(x)->(1-5*x^2)/x^2;
unknown:={a,b,c,d,e,h};
```

Das Polynom mit den 6 Unbekannten a,b,c,d,e,h

```
poly:=a*x^5+b*x^4+c*x^3+d*x^2+e*x+h;
```

Das Polynom muss dem Funktionswert, der ersten und der zweiten Ableitung entsprechen:
(An den Punkten $x=0.1$, $x=-0.1$)

```
glsys:={f(x)=poly,
        diff(f(x), x)=diff(poly, x),
        diff(f(x), x$2)=diff(poly, x$2)};
glsys_pa:=eval(glsys, [x=0.1]);
glsys_pb:=eval(glsys, [x=-0.1]);
replace:=solve(glsys_pa union glsys_pb, unknown);
```

Ausgabe der Funktion:

```
p:=(x)->eval(poly, replace);
print(p(x));
g:=piecewise(x<-0.1, f(x), -0.1<=x and x<0.1, p(x), x>0.1, f(x)
);
plot([f(x), g(x)], x=-2..2);
```

$$f:=x \rightarrow \frac{1-5x^2}{x^2}$$

$$\text{unknown} := \{a, b, c, d, e, h\}$$

$$\text{poly} := ax^5 + bx^4 + cx^3 + dx^2 + ex + h$$

$$\begin{aligned} \text{glsys} := & \left\{ \frac{-5x^2+1}{x^2} = ax^5 + bx^4 + cx^3 + dx^2 + ex + h, \frac{30}{x^2} + \frac{6(-5x^2+1)}{x^4} \right. \\ & = 20ax^3 + 12bx^2 + 6cx + 2d, -\frac{10}{x} - \frac{2(-5x^2+1)}{x^3} = 5ax^4 + 4bx^3 + 3cx^2 \\ & \left. + 2dx + e \right\} \end{aligned}$$

$$\begin{aligned} \text{glsys_pa} := & \{-2000.000000 = 0.0005a + 0.004b + 0.03c + 0.2d + e, 95.00000000 \\ & = 0.00001a + 0.0001b + 0.001c + 0.01d + 0.1e + h, 60000.00000 = 0.020a \\ & + 0.12b + 0.6c + 2d\} \end{aligned}$$

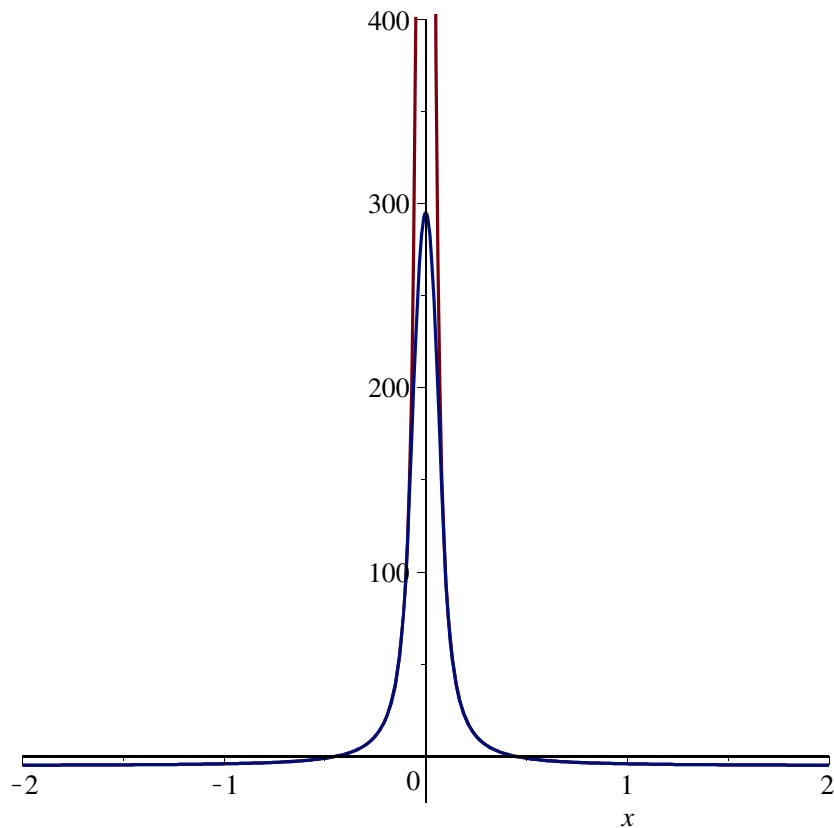
$$\begin{aligned} \text{glsys_pb} := & \{95.00000000 = -0.00001a + 0.0001b - 0.001c + 0.01d - 0.1e + h, \\ & 2000.000000 = 0.0005a - 0.004b + 0.03c - 0.2d + e, 60000.00000 = -0.020a \\ & + 0.12b - 0.6c + 2d\} \end{aligned}$$

$$\text{replace} := \{a = 0., b = 1.000000 \cdot 10^6, c = 0., d = -30000., e = 0., h = 295.\}$$

$$p := x \rightarrow \text{eval}(\text{poly}, \text{replace})$$

$$295. + 1.000000 \cdot 10^6 x^4 - 30000. x^2$$

$$g := \begin{cases} \frac{-5x^2 + 1}{x^2} & x < -0.1 \\ 295. + 1.000000 \cdot 10^6 x^4 - 30000. x^2 & -0.1 \leq x \text{ and } x < 0.1 \\ \frac{-5x^2 + 1}{x^2} & 0.1 < x \end{cases}$$



Aufgabe 1.6 - Taylorreihenentwicklung

> Die Taylorreihe ist definiert, als:

`p[f] := (x) -> sum(1/k! * diff(f(x[0]), x$k) * (x-x[0])^k, k=0..n);`

$$p_f := x \rightarrow \sum_{k=0}^n \frac{\left(\frac{\partial^k}{\partial x^k} f(x_0) \right) (x - x_0)^k}{k!}$$

(5.1)

effizient

> `restart;`

```
eff_taylor:=proc(f, n, xz_evl)
  local i:=0, res:=0, ifak:=1, xprod:=1, fdiff:=f(xz);
  for i from 0 to n do
```

```

    if i>0 then
      ifak:= ifak*i;
      xprod:=xprod*(x-xz);
      fdiff:=diff(fdiff, [xz]);
    fi;
    res := res + fdiff*xprod/(ifak);
  end:
  return eval(res, [xz=xz_evl]);
end:

```

```

a:=2:
f:=(x)->cos(a*x):

```

Taylorreihe fuer n=16:

```

res:=eff_taylor(f, 16, x[0]);
simply:=eff_taylor(f, 16, 0);

```

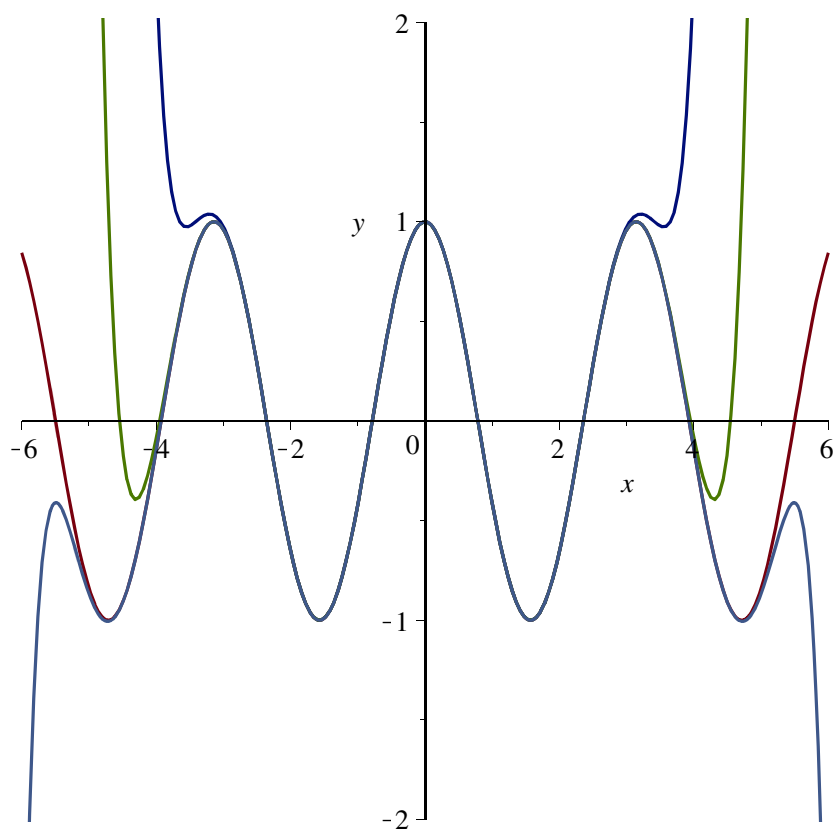
Je groesser n wird, desto besser wird f(x) approximiert.

```

plot([f(x),eff_taylor(f, 16, 0),eff_taylor(f, 20, 0),
eff_taylor(f, 26, 0)],x=-6..6,y=-2..2);

```

$$\begin{aligned}
 res := & \cos(2x_0) - 2 \sin(2x_0) (x - x_0) - 2 \cos(2x_0) (x - x_0)^2 + \frac{4}{3} \sin(2x_0) (x - x_0)^3 \\
 & + \frac{2}{3} \cos(2x_0) (x - x_0)^4 - \frac{4}{15} \sin(2x_0) (x - x_0)^5 \\
 & - \frac{4}{45} \cos(2x_0) (x - x_0)^6 + \frac{8}{315} \sin(2x_0) (x - x_0)^7 \\
 & + \frac{2}{315} \cos(2x_0) (x - x_0)^8 - \frac{4}{2835} \sin(2x_0) (x - x_0)^9 \\
 & - \frac{4}{14175} \cos(2x_0) (x - x_0)^{10} + \frac{8}{155925} \sin(2x_0) (x - x_0)^{11} \\
 & + \frac{4}{467775} \cos(2x_0) (x - x_0)^{12} - \frac{8}{6081075} \sin(2x_0) (x - x_0)^{13} \\
 & - \frac{8}{42567525} \cos(2x_0) (x - x_0)^{14} + \frac{16}{638512875} \sin(2x_0) (x - x_0)^{15} \\
 & + \frac{2}{638512875} \cos(2x_0) (x - x_0)^{16} \\
 simply := & 1 - 2x^2 + \frac{2}{3}x^4 - \frac{4}{45}x^6 + \frac{2}{315}x^8 - \frac{4}{14175}x^{10} + \frac{4}{467775}x^{12} \\
 & - \frac{8}{42567525}x^{14} + \frac{2}{638512875}x^{16}
 \end{aligned}$$



ineffizient

```
> restart:with(plots):
```

```
ineff_taylor:=proc(f, n, xz_ev1)
  local i:=0, res:=0;
  for i from 0 to n do
    res := res + diff(f(xz), [xz$i])*(x-xz)^i/(i!);
  end:
  return eval(res, [xz=xz_ev1]);
end:
```

```
a:=2:
f:=(x)->cos(a*x):
```

Taylorreihe fuer n=16:

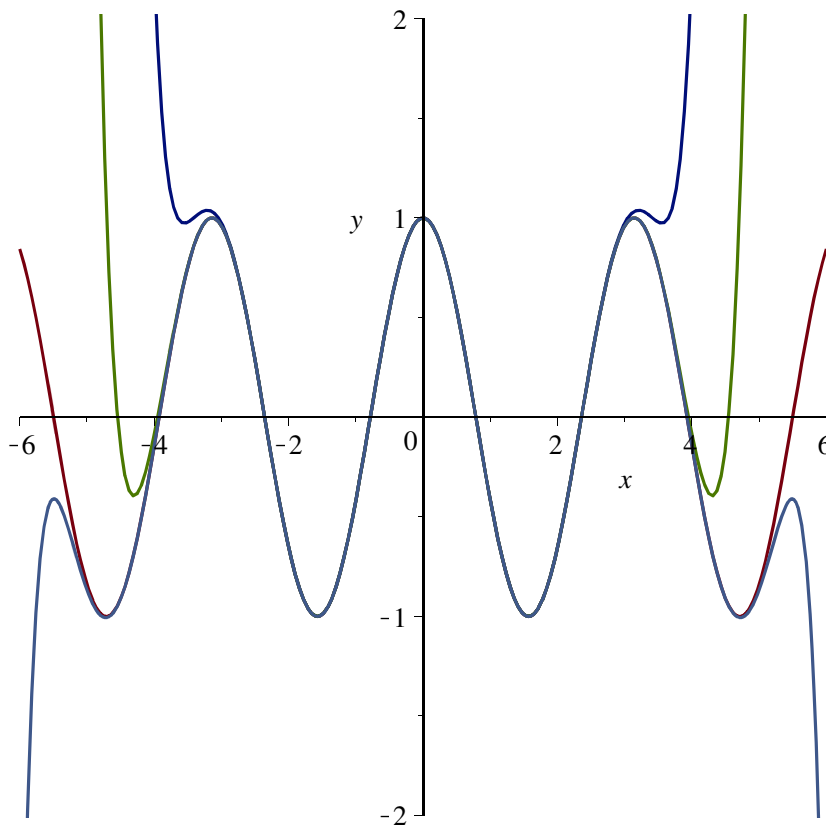
```
res:=ineff_taylor(f, 16, x[0]);
simply:=ineff_taylor(f, 16, 0);
```

Je groesser n wird, desto besser wird f(x) approximiert.

```
#animate(plot,[ineff_taylor(f, n,0), x=-4..4], n=[8, 10,
12, 14, 16],frames=2); #TODO: doesn't work
plot([f(x),ineff_taylor(f, 16, 0),ineff_taylor(f, 20, 0),
ineff_taylor(f, 26, 0)],x=-6..6,y=-2..2);
```

$$res := \cos(2x_0) - 2 \sin(2x_0) (x - x_0) - 2 \cos(2x_0) (x - x_0)^2 + \frac{4}{3} \sin(2x_0) (x - x_0)^3$$

$$\begin{aligned}
& -x_0)^3 + \frac{2}{3} \cos(2x_0) (x-x_0)^4 - \frac{4}{15} \sin(2x_0) (x-x_0)^5 \\
& - \frac{4}{45} \cos(2x_0) (x-x_0)^6 + \frac{8}{315} \sin(2x_0) (x-x_0)^7 \\
& + \frac{2}{315} \cos(2x_0) (x-x_0)^8 - \frac{4}{2835} \sin(2x_0) (x-x_0)^9 \\
& - \frac{4}{14175} \cos(2x_0) (x-x_0)^{10} + \frac{8}{155925} \sin(2x_0) (x-x_0)^{11} \\
& + \frac{4}{467775} \cos(2x_0) (x-x_0)^{12} - \frac{8}{6081075} \sin(2x_0) (x-x_0)^{13} \\
& - \frac{8}{42567525} \cos(2x_0) (x-x_0)^{14} + \frac{16}{638512875} \sin(2x_0) (x-x_0)^{15} \\
& + \frac{2}{638512875} \cos(2x_0) (x-x_0)^{16} \\
\text{simply} &:= 1 - 2x^2 + \frac{2}{3}x^4 - \frac{4}{45}x^6 + \frac{2}{315}x^8 - \frac{4}{14175}x^{10} + \frac{4}{467775}x^{12} \\
& - \frac{8}{42567525}x^{14} + \frac{2}{638512875}x^{16}
\end{aligned}$$



▼ Aufgabe 1.7 - Reihensumme

└> restart;

```
b:=(n)->(n+3)/(n^2+1);
s:=sum(b(k)-b(k+1),k=1..infinity);
```

$$b := n \rightarrow \frac{n+3}{n^2+1}$$

$$s := 2$$

(6.1)

Aufgabe 1.8 - Gewoehnliche DGL 1. Ord

> ode...ordinary differential equation

ics...initial conditions

restart;

```
odel:=diff(x(t), t)=1/2*x(t)^2*(cos(3*t)-1);
```

```
ics:=x(0)=1;
```

```
dsolve({odel, ics}, x(t));
```

$$odel := \frac{d}{dt} x(t) = \frac{1}{2} x(t)^2 (\cos(3t) - 1)$$

$$ics := x(0) = 1$$

$$x(t) = -\frac{6}{4 \cos(t)^2 \sin(t) - \sin(t) - 3t - 6}$$

(7.1)

Aufgabe 1.9 - Integro-DGL mit Laplace

Mit Faltungssatz:

```
> restart:with(intttrans):
```

```
odex := diff(x(t), t) = int(exp(a*(t-tau))*x(tau), tau = 0 .. t);
```

```
conv:=(f,g)->int(f(t-tau)*g(tau), tau=0..t):
```

Die Angabe kann in f(t) und g(t) aufgesplittet werden:

```
f:=(t)->exp(a*t);
```

```
g:=(t)->x(t);
```

```
diff(x(t), t) = conv(f,g);
```

f(t), g(t), und dx(t)/d(t) laplace transformiert:

```
lap_f:=laplace(f(t), t, s);
```

```
lap_g:=laplace(g(t), t, s);
```

```
lap_dx:=laplace(diff(x(t), t), t, s);
```

Mit dem Faltungssatz ergibt sich im Bildbereich (anschliessend umgestellt auf x(t)):

```
eqn_res:=lap_dx=lap_f*lap_g;
```

```
eqn:=solve(eqn_res, lap_g);
```

Ruecktransformiert in den Zeitbereich:

```
result:=x(t)=invlaplace(solve(eqn_res, lap_g), s, t);
```

$$odex := \frac{d}{dt} x(t) = \int_0^t e^{a(t-\tau)} x(\tau) d\tau$$

$$f := t \rightarrow e^{at}$$

$$g := t \rightarrow x(t)$$

$$\frac{d}{dt} x(t) = \int_0^t e^{a(t-\tau)} x(\tau) d\tau$$

$$lap_f := \frac{1}{s-a}$$

$$lap_g := \text{laplace}(x(t), t, s)$$

$$lap_dx := s \text{laplace}(x(t), t, s) - x(0)$$

$$eqn_res := s \text{laplace}(x(t), t, s) - x(0) = \frac{\text{laplace}(x(t), t, s)}{s-a}$$

$$eqn := \frac{x(0) (-s+a)}{a s - s^2 + 1}$$

$$result := x(t) = e^{\frac{1}{2} a t} x(0) \left(\cosh\left(\frac{1}{2} t \sqrt{a^2 + 4}\right) - \frac{a \sinh\left(\frac{1}{2} t \sqrt{a^2 + 4}\right)}{\sqrt{a^2 + 4}} \right) \quad (8.1.1)$$

Verifikation mit dsolve-laplace

> Mit dsolve-laplace kommt man auf das selbe Ergebnis.

```
odex:=diff(x(t), t)=int(exp(a*(t-tau))*x(tau), tau=0..t);
verif:=dsolve({odex}, x(t), method=laplace);
```

$$odex := \frac{d}{dt} x(t) = \int_0^t e^{a(t-\tau)} x(\tau) d\tau$$

$$verif := x(t) = e^{\frac{1}{2} a t} x(0) \left(\cosh\left(\frac{1}{2} t \sqrt{a^2 + 4}\right) - \frac{a \sinh\left(\frac{1}{2} t \sqrt{a^2 + 4}\right)}{\sqrt{a^2 + 4}} \right) \quad (8.2.1)$$

Aufgabe 1.10 - DGL 2. Ord, Zustandsraumdarstellung, Eigenwerte, Stabilität

```
> restart;with(LinearAlgebra):with(plots):
```

```
ode2:=diff(y(t), t$2)+2*alpha*diff(y(t), t)+y(t)=beta;
simpode2:=simplify(ode2) assuming 0<alpha, alpha<1;
```

$$ode2 := \frac{d^2}{dt^2} y(t) + 2 \alpha \left(\frac{d}{dt} y(t) \right) + y(t) = \beta$$

$$simpode2 := \frac{d^2}{dt^2} y(t) + 2 \alpha \left(\frac{d}{dt} y(t) \right) + y(t) = \beta \quad (9.1)$$

Punkt 1

> Allgemeine Lösung:

```
res:=dsolve(simpode2, y(t));
```

Um das charakteristische Polynom bestimmen zu können, muss die homogene Gleichung aufgestellt werden:

```
homo:=lhs(simpode2)=0;
```

```
res[homogen]:=dsolve({homo}, y(t));
```

Das charakteristische Polynom kann bestimmt werden, indem $d^2y(t)/dt^2$ durch λ^2 , $dy(t)/dt$ durch λ , usw. ersetzt wird:

```
charPoly:=subs([diff(y(t), t$2)=lambda^2, diff(y(t), t)=
lambda, y(t)=1], homo);
```

Die Nullstellen des charakteristischen Polynoms sind die Polstellen des Systems. (Da es identisch ist mit dem Nennerpolynom des Regelkreises)

Die Eigenwerte der Systemmatrix A sind genau die Nullstellen des charakteristischen Polynoms.

```
pole:=solve(charPoly, lambda);
```

Damit das System stabil ist, muss der Realteil jedes Pols kleiner als 0 sein, also in der linken Halbebene liegen. Das ist der Fall, wenn Alpha im Bereich von 0 bis unendlich liegt.

Das ist auf jeden Fall erfuehrt. Da Alpha lt. Angabe nur im Bereich von $0 < \alpha < 1$ liegt.

```
solve(Re(pole[1])<0, alpha) assuming 0<alpha, alpha<1;
```

#Wertebereich von alpha, in dem System stabil ist.

```
solve(Re(pole[2])<0, alpha) assuming 0<alpha, alpha<1;
```

#Wertebereich von alpha, in dem System stabil ist.

$$res := y(t) = e^{(-\alpha + \sqrt{\alpha^2 - 1})t} _C2 + e^{(-\alpha - \sqrt{\alpha^2 - 1})t} _C1 + \beta$$

$$homo := \frac{d^2}{dt^2} y(t) + 2\alpha \left(\frac{d}{dt} y(t) \right) + y(t) = 0$$

$$res_{homogen} := \left\{ y(t) = _C1 e^{(-\alpha + \sqrt{\alpha^2 - 1})t} + _C2 e^{(-\alpha - \sqrt{\alpha^2 - 1})t} \right\}$$

$$charPoly := 2\alpha\lambda + \lambda^2 + 1 = 0$$

$$pole := -\alpha + \sqrt{\alpha^2 - 1}, -\alpha - \sqrt{\alpha^2 - 1}$$

$$RealRange(Open(0), \infty)$$

$$RealRange(Open(0), \infty)$$

(9.1.1)

Punkt 2

> siehe http://www.isys.uni-stuttgart.de/lehre/systemdynamik/sgr/Zusatzuebungen/zueb8_loesung.pdf

de/lehre/systemdynamik/sgr/Zusatzuebungen/zueb8_loesung.pdf

Lt. dieser Loesung muss zuerst eine Integratorkette gebildet werden, deren Laenge der Ordnung der DGL um eins verringert entspricht:

Bsp: Ordnung = 4 => Laenge der Kette = 3 => (state-DGL's: $\text{diff}(x[1](t), t) = x[2](t)$, $\text{diff}(x[2](t), t) = x[3](t)$, $\text{diff}(x[3](t), t) = x[4](t)$)

In unserem Fall besteht die Kette nur aus einem Element, da wir eine DGL 2.Ordnung haben.

```
stateDGL1:=x[2](t)=diff(x[1](t), t);
```

Bildung der stateDGL fuer x[2]:

Das geschieht dadurch, dass in der ode2 das x[1] durch y und das x[2] durch y' ersetzt wird:

```
replace:=[y(t)=x[1](t), diff(y(t), t)=x[2](t), diff(y(t), t$2)
=diff(x[2](t), t)];
```

```
stateDGL2:=solve(subs(replace, ode2), diff(x[2](t), t))=diff
(x[2](t), t);
```

Nun sind die beiden Zustandsdifferentialgleichungen bekannt:

```
stateDGL:=[stateDGL1, stateDGL2];
```

Wobei hier der Zustandsvektor x definiert ist, als:

```
x_ := (t) -> [x[1](t), x[2](t)];
```

Und der Eingangsvariablenvektor u ist definiert, als:

```
u_ := [beta];
```

Bildung der Ausgangsdifferentialgleichungen: (Aus $y=x[1]$)

```
outDGL := [replace[1]];
```

Somit erhalten wir als Zustandsdifferentialgleichungsvektor (die Ableitung des Zustandsvektors), bzw. fuer den Ausgangsvariablenvektor:

```
"f(x, u, t) = dx(t)/dt" = Vector(map(rhs, stateDGL));
```

```
"h(x, u, t) = y(t)" = Vector(map(rhs, outDGL));
```

Und die Matrizen lauten:

Systemmatrix A :

```
A := GenerateMatrix(stateDGL, x_(t))[1];
```

Eingangs-Vektor:

```
B := GenerateMatrix(stateDGL, u_)[1];
```

Ausgangsvektor (transponiert):

```
C := GenerateMatrix([rhs(outDGL[1]) = lhs(outDGL[1])], x_(t))[1]; #TODO: ask, why lhs and rhs must be exchanged
```

Eigenwerte:

Aus Wikipedia: Ein System ist stabil, wenn alle Eigenwerte der Systemmatrix (bzw. Wurzeln bzw. Polstellen) einen negativen Realteil haben und damit in der linken Halbebene der komplexen Ebene (Pol-Nullstellen-Diagramm) liegen: => Daher stabil!

```
eigenwerte := Eigenvalues(A);
```

Allgemeine Loesung:

Zuerst muss die Transitionsmatrix Φ berechnet werden: ($\Phi(t) = \exp(A \cdot t)$) (mit Eqn 2.22 im Auto Skriptum)

```
Phi := (t) -> MatrixExponential(A*t);
```

```
"Phi(t)" = Phi(t);
```

Anschliessend kann damit die allgemeine Loesung bestimmt werden: (mit Satz 2.4 im Auto-Skriptum)

$x(t) = \Phi(t) \cdot x_0 + \int_0^t \Phi(t-\tau) B u(\tau) d\tau$, $\tau = 0..t$

$y(t) = C \cdot x(t) + D u(t)$

```
xAllg := (t) -> simplify(Phi(t).Vector(x_(0)) + map(int, ((Phi(t-tau).B).Vector(u_)), tau=0..t));
```

```
yAllg := (t) -> MatrixVectorMultiply(C, Vector(xAllg(t)));
```

```
"xAllg(t)" = xAllg(t);
```

```
"yAllg(t)" = yAllg(t);
```

$$stateDGL1 := x_2(t) = \frac{d}{dt} x_1(t)$$

$$replace := \left[y(t) = x_1(t), \frac{d}{dt} y(t) = x_2(t), \frac{d^2}{dt^2} y(t) = \frac{d}{dt} x_2(t) \right]$$

$$stateDGL2 := -2 \alpha x_2(t) - x_1(t) + \beta = \frac{d}{dt} x_2(t)$$

$$stateDGL := \left[x_2(t) = \frac{d}{dt} x_1(t), -2 \alpha x_2(t) - x_1(t) + \beta = \frac{d}{dt} x_2(t) \right]$$

$$x_ := t \rightarrow [x_1(t), x_2(t)]$$

$$u_ := [\beta]$$

$$outDGL := [y(t) = x_1(t)]$$

$$"f(x,u,t)=dx(t)/dt" = \begin{bmatrix} \frac{d}{dt} x_1(t) \\ \frac{d}{dt} x_2(t) \end{bmatrix}$$

$$"h(x,u,t)=y(t)" = \begin{bmatrix} x_1(t) \end{bmatrix}$$

$$A := \begin{bmatrix} 0 & 1 \\ -1 & -2\alpha \end{bmatrix}$$

$$B := \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$C := \begin{bmatrix} 1 & 0 \end{bmatrix}$$

$$eigenwerte := \begin{bmatrix} -\alpha + \sqrt{\alpha^2 - 1} \\ -\alpha - \sqrt{\alpha^2 - 1} \end{bmatrix}$$

$$\begin{aligned} "Phi(t)" = & \left[\left[\frac{1}{2} \frac{1}{\sqrt{\alpha^2 - 1}} \left(e^{-(\alpha + \sqrt{\alpha^2 - 1})t} \sqrt{\alpha^2 - 1} + \sqrt{\alpha^2 - 1} e^{(-\alpha + \sqrt{\alpha^2 - 1})t} \right. \right. \right. \\ & \left. \left. \left. - \alpha e^{-(\alpha + \sqrt{\alpha^2 - 1})t} + \alpha e^{(-\alpha + \sqrt{\alpha^2 - 1})t} \right), \right. \right. \\ & \left. \left. - \frac{1}{2} \frac{-e^{(-\alpha + \sqrt{\alpha^2 - 1})t} + e^{-(\alpha + \sqrt{\alpha^2 - 1})t}}{\sqrt{\alpha^2 - 1}} \right], \right. \\ & \left[\frac{1}{2} \frac{-e^{(-\alpha + \sqrt{\alpha^2 - 1})t} + e^{-(\alpha + \sqrt{\alpha^2 - 1})t}}{\sqrt{\alpha^2 - 1}}, \right. \\ & \left. \frac{1}{2} \frac{1}{\sqrt{\alpha^2 - 1}} \left(e^{-(\alpha + \sqrt{\alpha^2 - 1})t} \sqrt{\alpha^2 - 1} + \sqrt{\alpha^2 - 1} e^{(-\alpha + \sqrt{\alpha^2 - 1})t} \right. \right. \\ & \left. \left. \left. + \alpha e^{-(\alpha + \sqrt{\alpha^2 - 1})t} - \alpha e^{(-\alpha + \sqrt{\alpha^2 - 1})t} \right) \right] \right] \end{aligned}$$

$$\begin{aligned} "xAllg(t)" = & \left[\left[\frac{1}{2} \frac{1}{\sqrt{\alpha^2 - 1}} \left(\sqrt{\alpha^2 - 1} x_1(0) e^{-(\alpha + \sqrt{\alpha^2 - 1})t} \right. \right. \right. \\ & \left. \left. \left. + \sqrt{\alpha^2 - 1} x_1(0) e^{(-\alpha + \sqrt{\alpha^2 - 1})t} - \sqrt{\alpha^2 - 1} e^{-(\alpha + \sqrt{\alpha^2 - 1})t} \beta \right. \right. \right. \\ & \left. \left. \left. - \sqrt{\alpha^2 - 1} e^{(-\alpha + \sqrt{\alpha^2 - 1})t} \beta - x_1(0) e^{-(\alpha + \sqrt{\alpha^2 - 1})t} \alpha \right. \right. \right. \end{aligned}$$

$$\begin{aligned}
& + x_1(0) e^{(-\alpha + \sqrt{\alpha^2 - 1})t} \alpha + e^{-(\alpha + \sqrt{\alpha^2 - 1})t} \alpha \beta - e^{(-\alpha + \sqrt{\alpha^2 - 1})t} \alpha \beta \\
& + 2 \beta \sqrt{\alpha^2 - 1} - x_2(0) e^{-(\alpha + \sqrt{\alpha^2 - 1})t} + x_2(0) e^{(-\alpha + \sqrt{\alpha^2 - 1})t} \Big), \\
& \left[\frac{1}{2} \frac{1}{\sqrt{\alpha^2 - 1}} \left(\sqrt{\alpha^2 - 1} x_2(0) e^{-(\alpha + \sqrt{\alpha^2 - 1})t} \right. \right. \\
& + \sqrt{\alpha^2 - 1} x_2(0) e^{(-\alpha + \sqrt{\alpha^2 - 1})t} + x_2(0) e^{-(\alpha + \sqrt{\alpha^2 - 1})t} \alpha \\
& - x_2(0) e^{(-\alpha + \sqrt{\alpha^2 - 1})t} \alpha + \beta e^{(-\alpha + \sqrt{\alpha^2 - 1})t} + x_1(0) e^{-(\alpha + \sqrt{\alpha^2 - 1})t} \\
& \left. \left. - x_1(0) e^{(-\alpha + \sqrt{\alpha^2 - 1})t} - \beta e^{-(\alpha + \sqrt{\alpha^2 - 1})t} \right) \right] \Big] \\
\text{"yAllg(t)} &= \left[\frac{1}{2} \frac{1}{\sqrt{\alpha^2 - 1}} \left(\sqrt{\alpha^2 - 1} x_1(0) e^{-(\alpha + \sqrt{\alpha^2 - 1})t} \right. \right. \\
& + \sqrt{\alpha^2 - 1} x_1(0) e^{(-\alpha + \sqrt{\alpha^2 - 1})t} - \sqrt{\alpha^2 - 1} e^{-(\alpha + \sqrt{\alpha^2 - 1})t} \beta \\
& - \sqrt{\alpha^2 - 1} e^{(-\alpha + \sqrt{\alpha^2 - 1})t} \beta - x_1(0) e^{-(\alpha + \sqrt{\alpha^2 - 1})t} \alpha \\
& + x_1(0) e^{(-\alpha + \sqrt{\alpha^2 - 1})t} \alpha + e^{-(\alpha + \sqrt{\alpha^2 - 1})t} \alpha \beta - e^{(-\alpha + \sqrt{\alpha^2 - 1})t} \alpha \beta \\
& \left. \left. + 2 \beta \sqrt{\alpha^2 - 1} - x_2(0) e^{-(\alpha + \sqrt{\alpha^2 - 1})t} + x_2(0) e^{(-\alpha + \sqrt{\alpha^2 - 1})t} \right) \right] \quad (9.2.1)
\end{aligned}$$

▼ Punkt 3

```

> params:=[alpha=1/6, beta=-1,x[1](0)=1,x[2](0)=1];
yParam:=(t)->simplify(eval(yAllg(t), params));
xParam:=(t)->simplify(eval(xAllg(t), params));

```

```

"yParam(t)"=yParam(t);
"yParam(10)"=evalf(yParam(10));

```

Die Kurve aller Punkte im Zustandsraum (Vektor x) wird auch als Trajektorie bezeichnet.

```

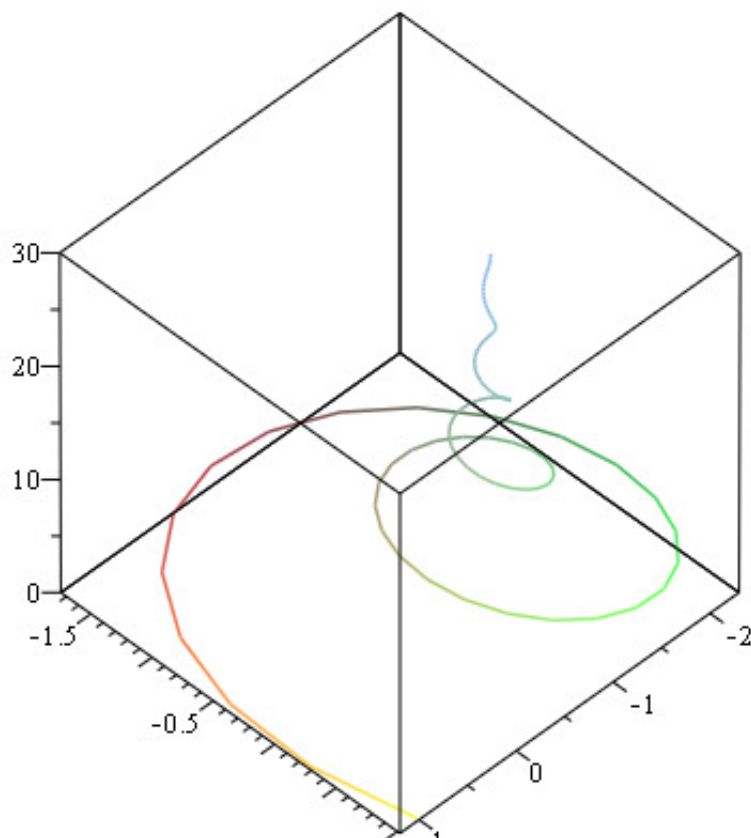
spacecurve([Re(xParam(t)[1]), Re(xParam(t)[2]), t], t=0.
.30, numpoints=100);

```

$$params := \left[\alpha = \frac{1}{6}, \beta = -1, x_1(0) = 1, x_2(0) = 1 \right]$$

$$\begin{aligned}
\text{"yParam(t)} &= \left[-\frac{1}{35} I \left(I\sqrt{35} e^{-\frac{1}{6}(I\sqrt{35}+1)t} + I\sqrt{35} e^{\frac{1}{6}(I\sqrt{35}-1)t} - I\sqrt{35} \right. \right. \\
& \left. \left. - 4 e^{-\frac{1}{6}(I\sqrt{35}+1)t} + 4 e^{\frac{1}{6}(I\sqrt{35}-1)t} \right) \sqrt{35} \right]
\end{aligned}$$

$$\text{"yParam(10)} = \left[-1.450227864 \right]$$



▼ Aufgabe 1.11 - Zweimassenschwinger, Zustandsraumdarstellung, Modellbildung

```
> restart; with(LinearAlgebra):
```

Geschwindigkeiten (lt. Angabe):

```
w1:=s1->diff(s1, t);
```

```
w2:=s2->diff(s2, t);
```

Kraefte-Gleichungen (lt. Angabe):

```
fc1:=(s1)->c1*sinh(s1/s10-1); #Federkraft
```

```
fc12:=(s1,s2)->c12*sinh((s2-s1)/s120-1);
```

```
fd1:=(w1)->d1*w1; #Daempfungskraft
```

```
fd12:=(w1,w2)->d12*(w2-w1);
```

Kraefte-Gleichungen (physikalisch):

```
ftr1:=(s1)->m1*diff(s1,t$2); #Traegheitskraft
```

```
ftr2:=(s2)->m2*diff(s2,t$2);
```

Eingangsfunktionen:

```
F1:=(t)->u(t); #Stelleingang
```

```
F2:=(t)->v(t); #Stoerung
```

Kraeftegleichgewicht: (alle Kraeftegleichungen inklusive der Eingangsfunktionen addiert)

muessen 0 ergeben).

```
fkg1:=ftr1(s1(t))=-fc1(s1(t))-fd1(w1(s1(t)))+fc12(s1(t),s2(t))+fd12(w1(s1(t)),w2(s2(t)))+F1(t);
fkg2:=ftr2(s2(t))=-fc12(s1(t),s2(t))-fd12(w1(s1(t)),w2(s2(t)))+F2(t);
```

Zustandsvektor x (lt. Angabe):

```
xL:=[s1(t), w1(s1(t)), s2(t), w2(s2(t))];
xVect:=Vector(xL);
```

Die entsprechenden Zustandsdifferentialgleichungen (als Vektor angeschrieben) lauten somit:

```
fL:=[w1(s1(t)), solve(fkg1, diff(xL[2],t)), w2(s2(t)), solve(fkg2, diff(xL[4],t))];
fVect:=Vector(fL);
```

Vektoren in x und dx:

```
tmpXL:=map2(subs,{s1(t)=x1(t), s2(t)=x2(t)},xL);
tmpfL:=map2(subs,{s1(t)=x1(t), s2(t)=x2(t)},fL);
unassign(w1):unassign(w2):
"xVect"=Vector(tmpXL);
"dx/dt=f(x,u,t)"=Vector(tmpfL);
```

Vektoren in s und w:

```
subsXL:=map2(subs,{diff(s1(t),t)=w1, diff(s2(t),t)=w2,s1(t)=s1, s2(t)=s2},xL);
subsFL:=map2(subs,{diff(s1(t),t)=w1, diff(s2(t),t)=w2,s1(t)=s1, s2(t)=s2},fL);
"xVect"=Vector(subsXL);
"dx/dt=f(x,u,t)"=Vector(subsFL);
```

Ausgang lt. Angabe:

```
ausgang:=y=s2-s1;
```

$$w1 := s1 \rightarrow \frac{\partial}{\partial t} s1$$

$$w2 := s2 \rightarrow \frac{\partial}{\partial t} s2$$

$$fc1 := s1 \rightarrow c1 \sinh\left(\frac{s1}{s10} - 1\right)$$

$$fc12 := (s1, s2) \rightarrow c12 \sinh\left(\frac{s2 - s1}{s120} - 1\right)$$

$$fd1 := w1 \rightarrow d1 w1$$

$$fd12 := (w1, w2) \rightarrow d12 (w2 - w1)$$

$$ftr1 := s1 \rightarrow m1 \left(\frac{\partial^2}{\partial t^2} s1 \right)$$

$$ftr2 := s2 \rightarrow m2 \left(\frac{\partial^2}{\partial t^2} s2 \right)$$

$$fkg1 := m1 \left(\frac{d^2}{dt^2} s1(t) \right) = c1 \sinh\left(-\frac{s1(t)}{s10} + 1\right) - d1 \left(\frac{d}{dt} s1(t) \right) - c12 \sinh\left(-\frac{s2(t) - s1(t)}{s120} + 1\right) + d12 \left(\frac{d}{dt} s2(t) - \left(\frac{d}{dt} s1(t) \right) \right) + u(t)$$

$$fkg2 := m2 \left(\frac{d^2}{dt^2} s2(t) \right) = cI2 \sinh \left(- \frac{s2(t) - sI(t)}{sI20} + 1 \right) - dI2 \left(\frac{d}{dt} s2(t) \right. \\ \left. - \left(\frac{d}{dt} sI(t) \right) \right) + v(t)$$

$$xVect := \begin{bmatrix} sI(t) \\ \frac{d}{dt} sI(t) \\ s2(t) \\ \frac{d}{dt} s2(t) \end{bmatrix}$$

$$fVect := \left[\left[\frac{d}{dt} sI(t) \right], \right. \\ \left[\frac{1}{mI} \left(cI \sinh \left(- \frac{sI(t)}{sI0} + 1 \right) - dI \left(\frac{d}{dt} sI(t) \right) - dI2 \left(\frac{d}{dt} sI(t) \right) \right. \right. \\ \left. \left. - cI2 \sinh \left(- \frac{s2(t) - sI(t)}{sI20} + 1 \right) + dI2 \left(\frac{d}{dt} s2(t) \right) + u(t) \right) \right], \\ \left[\frac{d}{dt} s2(t) \right], \\ \left[\frac{dI2 \left(\frac{d}{dt} sI(t) \right) + cI2 \sinh \left(- \frac{s2(t) - sI(t)}{sI20} + 1 \right) - dI2 \left(\frac{d}{dt} s2(t) \right) + v(t)}{m2} \right] \right]$$

$$"xVect" = \begin{bmatrix} xI(t) \\ \frac{d}{dt} xI(t) \\ x2(t) \\ \frac{d}{dt} x2(t) \end{bmatrix}$$

$$"dx/dt=f(x,u,t)" = \left[\left[\frac{d}{dt} xI(t) \right], \right. \\ \left[\frac{1}{mI} \left(cI \sinh \left(- \frac{xI(t)}{sI0} + 1 \right) - dI \left(\frac{d}{dt} xI(t) \right) - dI2 \left(\frac{d}{dt} xI(t) \right) \right. \right. \\ \left. \left. - cI2 \sinh \left(- \frac{x2(t) - xI(t)}{sI20} + 1 \right) + dI2 \left(\frac{d}{dt} x2(t) \right) + u(t) \right) \right], \\ \left[\frac{d}{dt} x2(t) \right], \\ \left[\frac{dI2 \left(\frac{d}{dt} xI(t) \right) + cI2 \sinh \left(- \frac{x2(t) - xI(t)}{sI20} + 1 \right) - dI2 \left(\frac{d}{dt} x2(t) \right) + v(t)}{m2} \right]$$

]]

$$\text{"xVect"} = \begin{bmatrix} s1 \\ w1 \\ s2 \\ w2 \end{bmatrix}$$

$$\text{"dx/dt=f(x,u,t)} = \begin{bmatrix} w1 \end{bmatrix},$$

$$\left[\frac{1}{m1} \left(-c1 \sinh \left(\frac{s1}{s10} - 1 \right) - d1 w1 - d12 w1 - c12 \sinh \left(-\frac{s2-s1}{s120} + 1 \right) + d12 w2 + u(t) \right) \right],$$

$$\begin{bmatrix} w2 \end{bmatrix},$$

$$\left[\frac{d12 w1 + c12 \sinh \left(-\frac{s2-s1}{s120} + 1 \right) - d12 w2 + v(t)}{m2} \right]$$

$$\text{ausgang} := y = s2 - s1$$

(10.1)

Aufgabe 1.12 - Zweimassenschwinger, Ruhelage, Uebertragungsfunktion, Bode-Diagramm

Ruhelage

```
> replace:={w1=w1R, w2=w2R, u(t)=F1R, v(t)=F2R};
```

Die Ruhelage kann berechnet werden, indem die Zustandsdifferentialgleichungen (also die erste Ableitung des Zustandsvektors) auf 0 gesetzt werden (denn die Änderung nach der Zeit soll 0 sein).

Da Maple in diesem Fall die Differentialgleichungen nicht direkt lösen kann:

```
solve(eval([subsFL[1]=0, subsFL[3]=0, subsFL[2]=0, subsFL[4]=0], replace union {s2=s2R, s1=s1R}), subsXL);
```

Muss das Gleichungssystem manuell gelöst werden:

```
w1R := solve(subsFL[1]=0, subsXL[2]);
w2R := solve(subsFL[3]=0, subsXL[4]);
s2R := solve(eval(subsFL[2]=0, replace), subsXL[3]);
s1R := solve(eval(subsFL[4]=0, replace union {s2=s2R}), subsXL[1]);
s2R := solve(eval(subsFL[2]=0, replace union {s1=s1R}), subsXL[3]);
```

Die berechneten Werte aus dem Gleichungssystem Eingesetzt:

```
xR := <s1R, w1R, s2R, w2R>; #Reihenfolge lt. Angabe
gegeben.
uR := <F1R, F2R>;
```

$$\begin{aligned}
\text{replace} &:= \{w1 = w1R, w2 = w2R, u(t) = F1R, v(t) = F2R\} \\
&[[\]] \\
w1R &:= 0 \\
w2R &:= 0 \\
s1R &:= \operatorname{arcsinh}\left(\frac{F2R + F1R}{c1}\right) s10 + s10 \\
s2R &:= \operatorname{arcsinh}\left(\frac{F2R + F1R}{c1}\right) s10 + \operatorname{arcsinh}\left(\frac{F2R}{c12}\right) s120 + s10 + s120 \\
xR &:= \begin{bmatrix} \operatorname{arcsinh}\left(\frac{F2R + F1R}{c1}\right) s10 + s10 \\ 0 \\ \operatorname{arcsinh}\left(\frac{F2R + F1R}{c1}\right) s10 + \operatorname{arcsinh}\left(\frac{F2R}{c12}\right) s120 + s10 + s120 \\ 0 \end{bmatrix} \\
uR &:= \begin{bmatrix} F1R \\ F2R \end{bmatrix}
\end{aligned} \tag{11.1.1}$$

Linearisierung

```
> with(VectorCalculus, Jacobian): #Achtung, Jacobian nur so
einbinden, das VectorCalculus Paket macht sonst sehr viel
Bloedsinn.
ruheReplacements:=replace union {s1=s1R, s2=s2R, uu=uR[1],
vv=uR[2]}:
```

Eingangsfunktionen:

```
uu:=<u>; #F2 ist die Stoerung
vv:=<v>; #F1 ist der Stelleingang.
uv:=<u, v>;
```

Zustandsvector [s1, w1, s2, w2]:

```
xx:=<s1, w1, s2, w2>;
```

Ausgang:

```
hh:=<rhs (ausgang)>;
```

rechte Seite der Zustandsdifferentialgleichung:

```
ff:=Vector(eval(subsFL, {u(t)=u, v(t)=v}));
```

Die Matrizen des Zustandsraums:

```
AA:=simplify(eval(Jacobian(ff, convert(xx, list)),
ruheReplacements));#Systemmatrix
BBu:=simplify(eval(Jacobian(ff, convert(uu, list)),
ruheReplacements));#Eingangsvektor nur mit der Stoerung
BBv:=simplify(eval(Jacobian(ff, convert(vv, list)),
ruheReplacements));#Eingangsvektor nur mit dem
Stelleingang
BB:=simplify(eval(Jacobian(ff, convert(uv, list)),
ruheReplacements));#Eingangsvektor mit Stoerung und
Stelleingang
CC:=simplify(eval(Jacobian(hh, convert(xx, list)),
ruheReplacements));#Ausgangsmatrix
```

```
DD:=simplify(eval(Jacobian(hh, convert(uv, list)),
ruheReplacements));#Durchgangsmatrix mit Stoerung und
Stelleingang
```

Die Zustandsdifferentialgleichung lautet somit:

```
lin_f:=AA.Delta*x+BBu.Delta*u+BBv.Delta*v;
```

und die Ausgangsdifferentialgleichung:

```
lin_h:=CC.Delta*x;
```

$$uu := \begin{bmatrix} u \end{bmatrix}$$

$$vv := \begin{bmatrix} v \end{bmatrix}$$

$$uv := \begin{bmatrix} u \\ v \end{bmatrix}$$

$$xx := \begin{bmatrix} s1 \\ w1 \\ s2 \\ w2 \end{bmatrix}$$

$$hh := \begin{bmatrix} s2 - s1 \end{bmatrix}$$

$$ff := \begin{bmatrix} \begin{bmatrix} w1 \end{bmatrix},$$

$$\begin{bmatrix} \frac{1}{m1} \left(-c1 \sinh\left(\frac{s1}{s10} - 1\right) - d1 w1 - d12 w1 - c12 \sinh\left(-\frac{s2 - s1}{s120} + 1\right) + d12 w2 + u \right) \end{bmatrix},$$

$$\begin{bmatrix} w2 \end{bmatrix},$$

$$\begin{bmatrix} \frac{d12 w1 + c12 \sinh\left(-\frac{s2 - s1}{s120} + 1\right) - d12 w2 + v}{m2} \end{bmatrix} \end{bmatrix}$$

$$AA := \begin{bmatrix} \begin{bmatrix} 0, 1, 0, 0 \end{bmatrix},$$

$$\begin{bmatrix}$$

$$- \frac{c1 \sqrt{\frac{F1R^2 + 2 F1R F2R + F2R^2 + c1^2}{c1^2}} s120 + c12 \sqrt{\frac{F2R^2 + c12^2}{c12^2}} s10}{s10 s120 m1},$$

$$- \frac{d1 + d12}{m1}, \frac{c12 \sqrt{\frac{F2R^2 + c12^2}{c12^2}}}{s120 m1}, \frac{d12}{m1} \end{bmatrix},$$

$$\begin{bmatrix} 0, 0, 0, 1 \end{bmatrix},$$

$$\left[\frac{c l 2 \sqrt{\frac{F 2 R^2 + c l 2^2}{c l 2^2}}}{s l 2 0 m 2}, \frac{d l 2}{m 2}, -\frac{c l 2 \sqrt{\frac{F 2 R^2 + c l 2^2}{c l 2^2}}}{s l 2 0 m 2}, -\frac{d l 2}{m 2} \right]$$

$$BBu := \begin{bmatrix} 0 \\ \frac{1}{m l} \\ 0 \\ 0 \end{bmatrix}$$

$$BBv := \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{m 2} \end{bmatrix}$$

$$BB := \begin{bmatrix} 0 & 0 \\ \frac{1}{m l} & 0 \\ 0 & 0 \\ 0 & \frac{1}{m 2} \end{bmatrix}$$

$$CC := \begin{bmatrix} -1 & 0 & 1 & 0 \end{bmatrix}$$

$$DD := \begin{bmatrix} 0 & 0 \end{bmatrix}$$

$$lin_f := \left[\begin{bmatrix} 0, 1, 0, 0 \end{bmatrix}, \right.$$

$$\left[\right.$$

$$-\frac{c l \sqrt{\frac{F l R^2 + 2 F l R F 2 R + F 2 R^2 + c l^2}{c l^2}} s l 2 0 + c l 2 \sqrt{\frac{F 2 R^2 + c l 2^2}{c l 2^2}} s l 0}{s l 0 s l 2 0 m l},$$

$$\left. -\frac{d l + d l 2}{m l}, \frac{c l 2 \sqrt{\frac{F 2 R^2 + c l 2^2}{c l 2^2}}}{s l 2 0 m l}, \frac{d l 2}{m l} \right],$$

$$\begin{aligned}
& \begin{bmatrix} 0, 0, 0, 1 \end{bmatrix}, \\
& \left[\frac{c12 \sqrt{\frac{F2R^2 + c12^2}{c12^2}}}{s120 m2}, \frac{d12}{m2}, -\frac{c12 \sqrt{\frac{F2R^2 + c12^2}{c12^2}}}{s120 m2}, -\frac{d12}{m2} \right] \Delta x \\
& + \begin{bmatrix} 0 \\ \frac{1}{m1} \\ 0 \\ 0 \end{bmatrix} \Delta u + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{m2} \end{bmatrix} \Delta v \\
& \text{lin}_h := \begin{bmatrix} -1 & 0 & 1 & 0 \end{bmatrix} \Delta x
\end{aligned} \tag{11.2.1}$$

Uebertragungsfunktion

> Zum Berechnen der Uebertragungsfunktion sind die folgenden Parameter lt. Angabe gegeben:

params := {m1=5, m2=1, s10=0.35, s120=0.5, d1=3, d12=0.15, c1=50, c12=5, F1R=30, F2R=0};

Eingesetzt in die Formel $G(s) = c^T (sE - A)^{-1} b + d$ (Satz 3.5) im Automatisierungsskriptum folgt: (Wobei E...Einheitsmatrix und $()^{(-1)}$ die Inverse Matrix darstellt)

rawG := (s) -> simplify((CC.MatrixInverse(s*IdentityMatrix(4) - AA)) . BB + DD) :

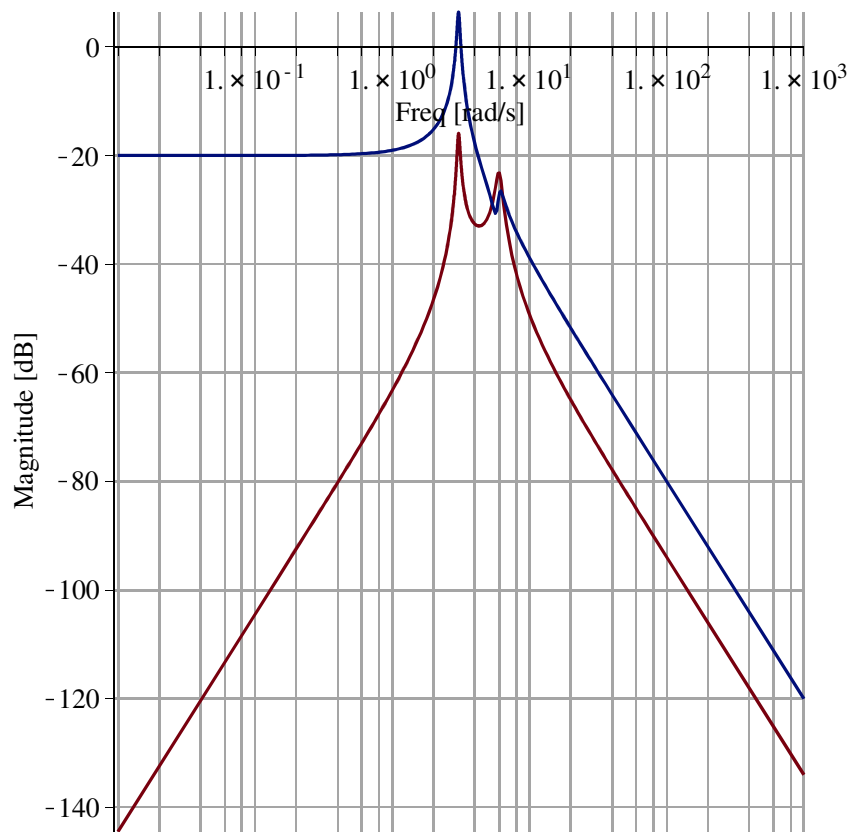
G := (s) -> simplify(eval(rawG(s), params)) :
G(s) ;

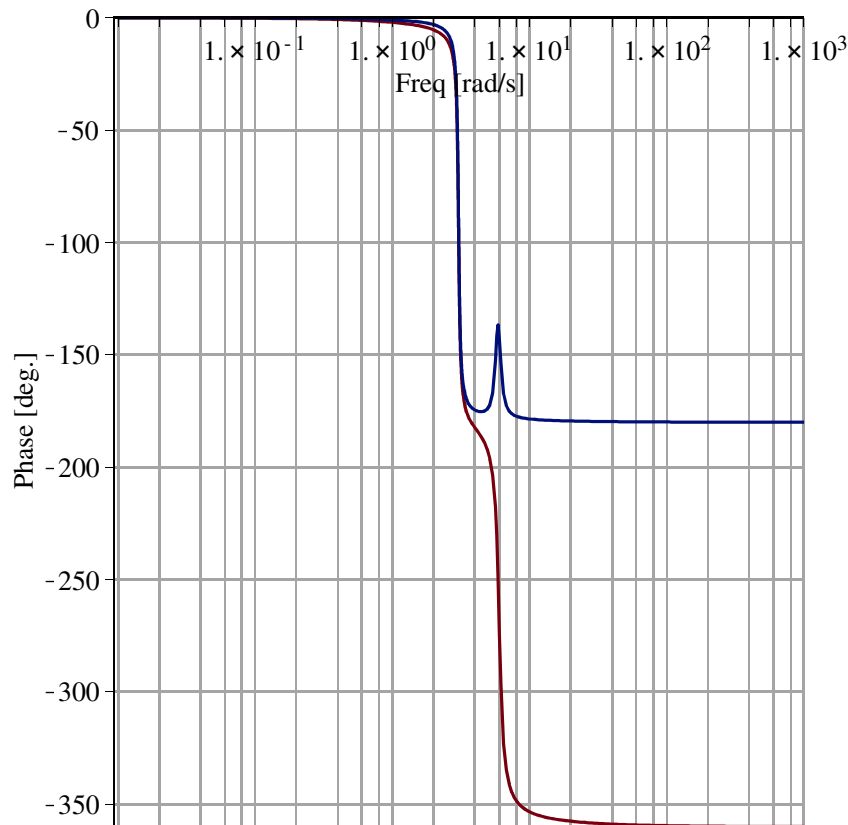
params := {F1R=30, F2R=0, c1=50, c12=5, d1=3, d12=0.15, m1=5, m2=1, s10=0.35, s120=0.5}

$$\begin{aligned}
& \left[-\frac{1.400 s^2}{7. s^4 + 5.460000000 s^3 + 317.8680758 s^2 + 76.98571137 s + 2332.380758}, \right. \\
& \left. \frac{7.000000000 (s^2 + 0.6000000000 s + 33.31972512)}{7. s^4 + 5.460000000 s^3 + 317.8680758 s^2 + 76.98571137 s + 2332.380758} \right]
\end{aligned} \tag{11.3.1}$$

Bode-Diagramm

> **with(DynamicSystems) :**
BodePlot(TransferFunction(G(s)), numpoints=500) ;





▼ Aufgabe 1.13 - Zweimasseschwinger, Endwert

> Allgemeine Loesung:

Die Parameter von 1.12 eingesetzt in die Matrizen:

```
AAparam:=map(simplify, eval(AA, params)):#Systemmatrix
BBparam:=map(simplify, eval(BB, params)):#Eingangsvektor mit
Stoerung und Stelleingang
CCparam:=map(simplify, eval(CC, params)):#Ausgangsmatrix
DDparam:=map(simplify, eval(DD, params)):#Durchgangsmatrix
mit Stoerung und Stelleingang
```

Anschliessend muss die Transitionsmatrix Phi berechnet werden: $\Phi(t) = \exp(A \cdot t)$

```
Phi:=(t)->MatrixExponential(AAparam, t):
```

Die Ruhelage mit den Parameter von 1.12 eingesetzt:

```
xRparam:=eval(xR, params):
uRparam:=eval(uR, params):
```

Anfangswerte: Das System befindet sich fuer $t \leq 0$ lt. Angabe in der Ruhelage:

```
x0:=xRparam:
dx0:=x0-xRparam:
```

▼ Punkt 1

```
> uEingang1 := (t) -> eval(<F1R, 0.6>, params) :
```

Anschliessend kann damit die allgemeine Loesung bestimmt werden: (mit Satz 2.4 im Auto-Skriptum)

$$x(t) = \Phi(t) \cdot x[0] + \int (\Phi(t-\tau) B u(\tau)) d\tau, \tau=0..t$$

$$y(t) = C \cdot x(t) + D u(t)$$

Wobei natuerlich noch die Ruhelage an $x(t)$ dazuaddiert werden muss, denn das System startet bei der Ruhelage:

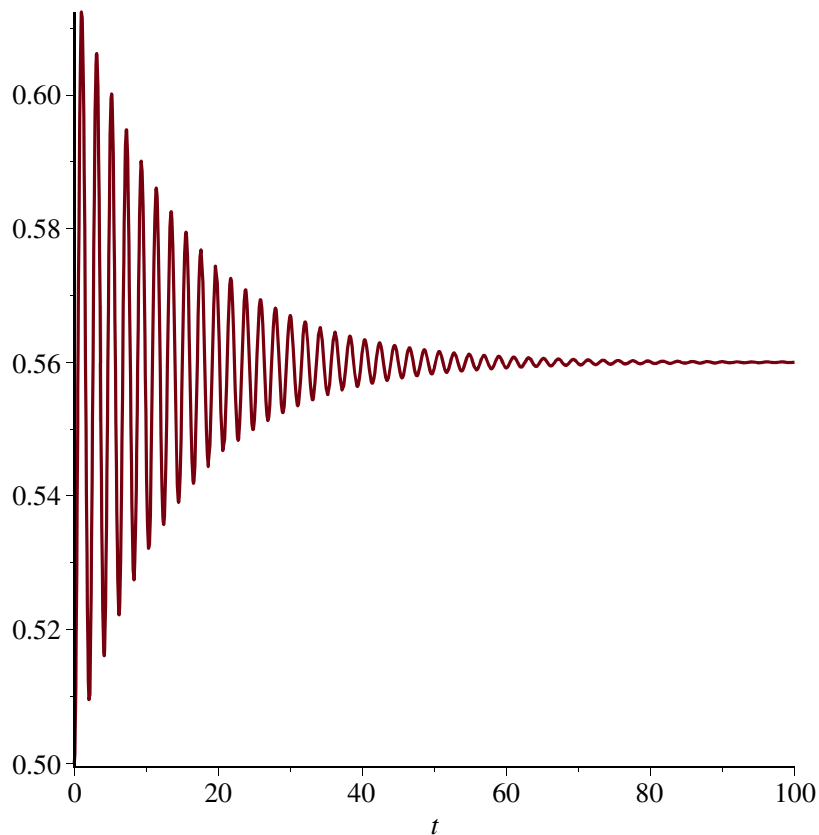
```
xAllg1 := (t) -> simplify(xRparam + Phi(t) . dx0 + map(int, (Phi(t-  
tau) . BBparam) . (uEingang1(tau) - uRparam), tau=0..t)) :
```

```
yAllg1 := (t) -> CCparam.xAllg1(t) + DDparam.(uEingang1(t) -  
uRparam) :
```

```
plot(yAllg1(t), t=0..100);
```

Endwert (Eingeschwungener Zustand):

```
endwert := limit(yAllg1(t)[1], t=infinity);
```



$$\text{endwert} := 0.5600000000 + 6.295274910 \cdot 10^{-14} i$$

(12.1.1)

▼ Punkt 2

```
> uEingang2 := (t) -> eval(<F1R, 0.3*sin(2*t+convert(10*degrees,  
radians))>, params) :
```

Anschliessend kann damit die allgemeine Loesung bestimmt werden: (mit Satz 2.4 im

Auto-Skriptum)

$x(t) = \Phi(t) * x[0] + \int (\Phi(t-\tau) B u(\tau) d\tau, \tau=0..t)$

$y(t) = C * x(t) + D u(t)$

Wobei natuerlich noch die Ruhelage an $x(t)$ dazuaddiert werden muss, denn das System startet bei der Ruhelage:

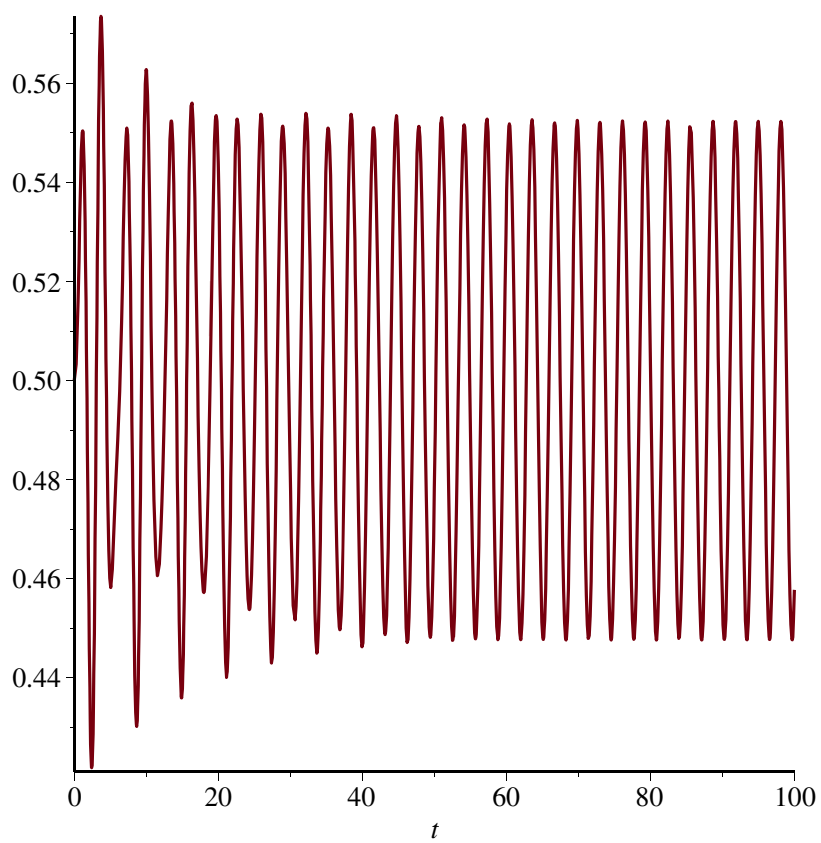
```
xAllg2 := (t) -> simplify (xRparam + Phi (t) . dx0 + map (int, (Phi (t -  
tau) . BBparam) . (uEingang2 (tau) - uRparam), tau=0..t)) :  
yAllg2 := (t) -> CCparam.xAllg2 (t) + DDparam. (uEingang2 (t) -  
uRparam) :  
plot (yAllg2 (t), t=0..100);
```

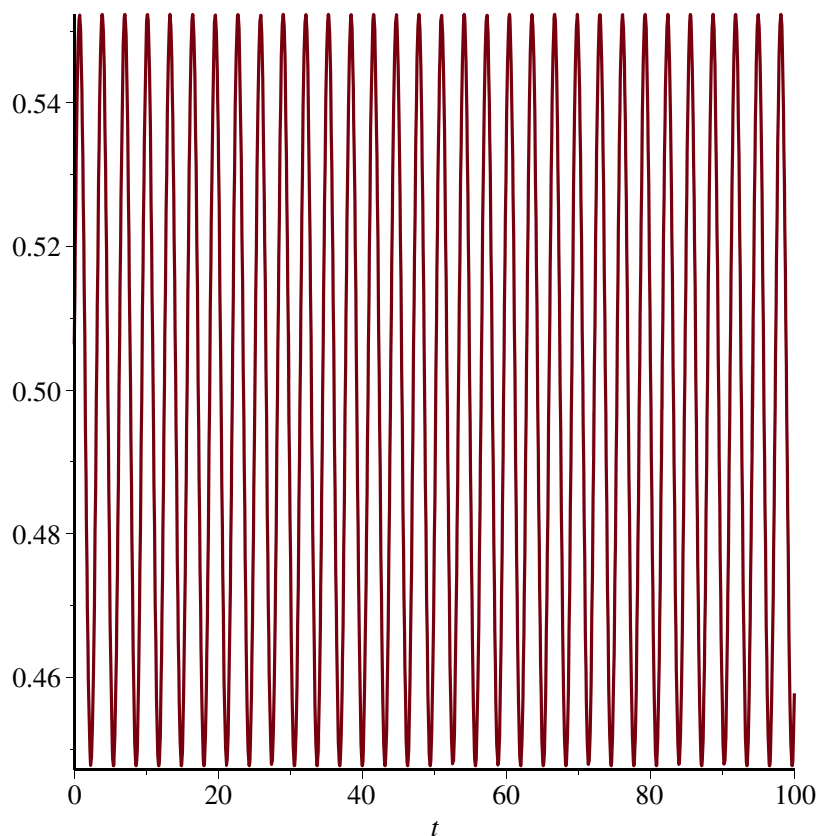
Wird am Eingang eine harmonische Eingangsgroesse der Form $u(t) = A[0] * \sin(w[0] * t)$ geschaltet (und das System ist BIBI-Stabil), dann erhaelt man fuer die Ausgangsgroesse im eingeschwungenen Zustand das folgende $y(t)$: (siehe s.69, eqn 3.103)

$y(t) = A[0] \text{abs}(G(Iw[0]) * \sin(w[0] * t + \arg(G(Iw[0])))$;

Da unser $G(Iw)$ eine Matrix ist, muessen beide Teile der Matrix betrachtet werden:

```
yEndwert := (t) -> uEingang2 (t) [1] * abs (G (I*0) [1] [1]) * cos (0*t +  
argument (G (I*0) [1] [1]))  
+ 0.3 * abs (G (I*2) [1] [2]) * sin (2*t + convert (10*degrees,  
radians) + argument (G (I*2) [1] [2]))  
+ eval (s120, params): #Auch hier muss die Ruhelage  
von y (das definiert ist, als y=s2-s1) addiert werden.  
plot (yEndwert (t), t=0..100);  
endwert := limit (yEndwert (t), t=infinity);
```





endwert := 0.4476921445 ..0.5523078555

(12.2.1)

Aufgabe 1.14 - Potentielle Energie

> Im folgenden muessen die Parameter von 1.12 angewendet werden:

```
fc1param:=(s1)->eval(fc1(s1),params):
fc12param:=(s12)->eval(eval(fc12(s1, s2), s2-s1=s12),params)
:
s10param:=eval(s10, params): #Entspannte lage von s1=s10
s120param:=eval(s120, params): #Entspannte lage von s2-s1=
s120
```

Pot. Energie Formeln: (geg. lt. Angabe)

```
v1:=(s)->int(fc1param(xi), xi=s10param..s):
v12:=(s)->int(fc12param(xi), xi=s120param..s):
```

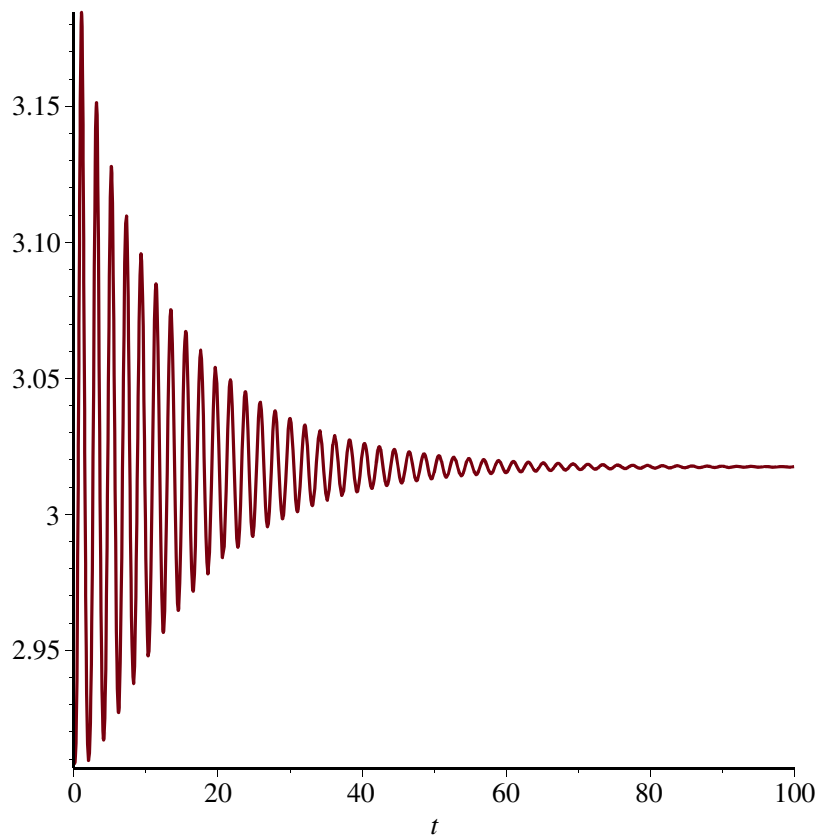
Plots berechnen (mit Stoerung von Aufgabe 1.13, Unterpunkt 1)

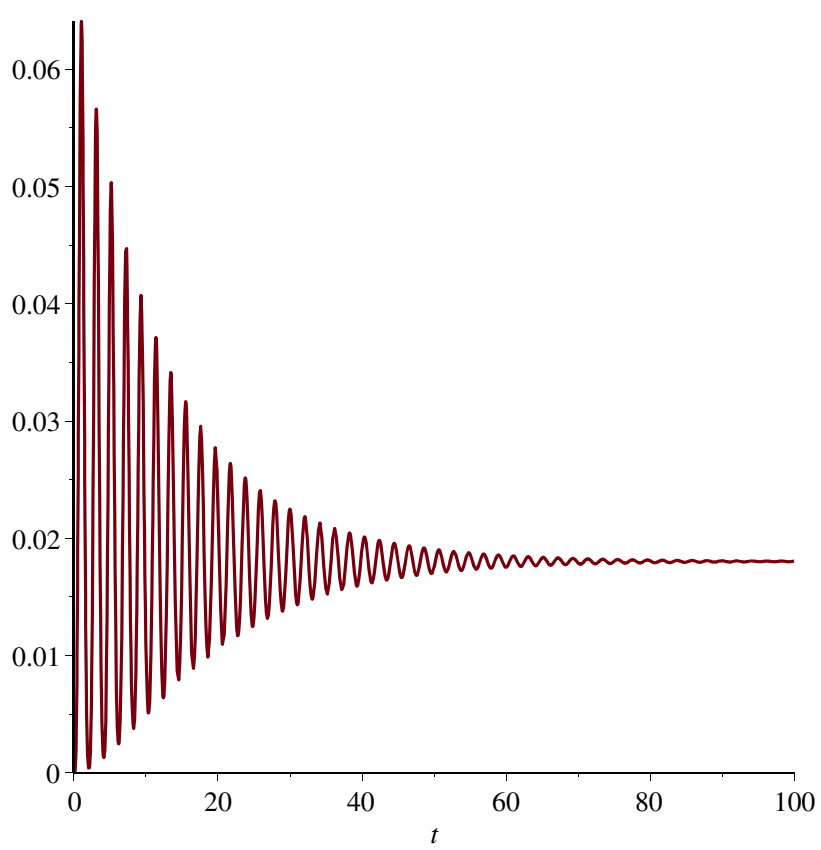
```
p1x:=(t)->v1(xAllg1(t)[1]):
plot(p1x(t), t=0..100);
p1y:=(t)->v12(yAllg1(t)[1]):
plot(p1y(t), t=0..100);
```

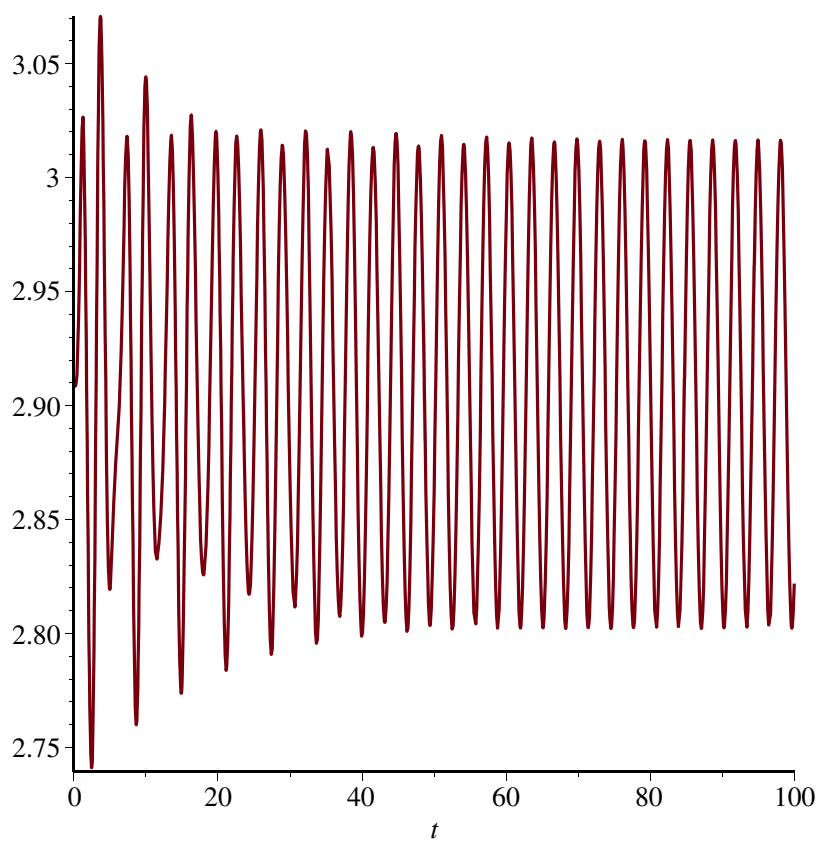
Plots berechnen (mit Stoerung von Aufgabe 1.13, Unterpunkt 2)

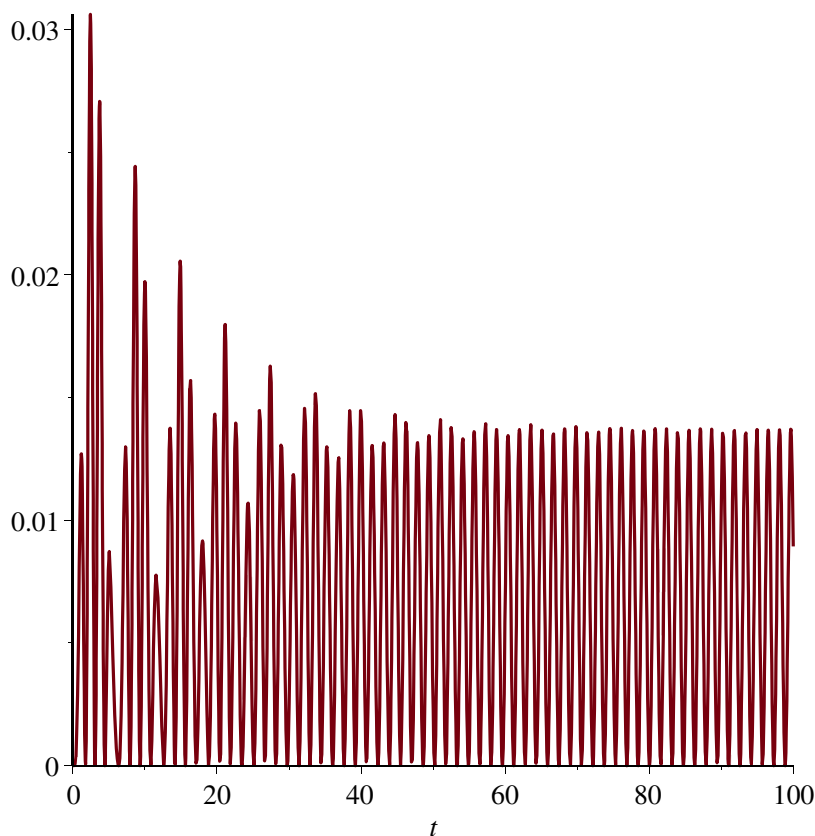
```
p1x:=(t)->v1(xAllg2(t)[1]):
plot(p1x(t), t=0..100);
p1y:=(t)->v12(yAllg2(t)[1]):
```

```
plot(ply(t), t=0..100);
```









▼ Aufgabe 1.15 - Diskretes System

▼ Punkt 1

```
> restart:with(LinearAlgebra):with(intttrans):
params:=[c=0.5, T1=1*10^(-3), a=0.4, Ta=1*10^(-6)]:
G:=(s)->c/(s*T1+1);
```

in die Beobachtbarkeitsnormalform bringen: (eqn 3.82)

Dabei muss der Koeffizient der groessten Potenz im Nenner den Wert 1 annehmen.

Aus der Uebertragungsfunktion wird die folgende Variablen-Zerlegung durchgefuehrt:

(Bei dieser Zerlegung muss a[n] immer den Wert 1 annehmen.)

```
zerlegung:=[a0=1/T1, a1=1, b0=c/T1, b1=0]:
```

Fuer die Zerlegung umgeformt:

```
G[zerl]:=(s)->eval((b0+b1*s)/(a0+a1*s), zerlegung):
"G-zerlegt "=G[zerl](s); #identify AA, BB, ... from this
form
```

Die Systemmatrix, der Eingangsvektor, die Ausgangsmatrix und der Durchgangsvektor:

```
AA[Proc]:=eval(<-a0>>, zerlegung); #siehe 3.5: 3.91a page
64
```

```
BB[Proc]:=eval(<b0-a0*b1>, zerlegung); #b0-a0*b1...siehe
```

3.86 page 63

```
CC[Proc]:=eval(<a1>, zerlegung);
DD[Proc]:=eval(<b1>, zerlegung);
```

Berechnung der Dynamikmatrix Phi: (siehe 2.21, 6.19c)

```
Phi[Proc]:=MatrixExponential(AA[Proc], Ta); #Ta...
Abtastrate
```

Berechnung des Eingangsvektors Gamma (siehe 6.19d)

```
Gamma[Proc]:=map(simplify, map(int, MatrixExponential(AA
[Proc], tau),tau=0..Ta).BB[Proc]);
```

Das Abtastsystem bildet sich zu: (Siehe 6.19a, b)

```
x1(k+1):=Phi[Proc][1][1].x1(k) + Gamma[Proc][1].u1(k);
y1(k):=CC[Proc][1].x1(k)+DD[Proc].u1(k);
```

$$G := s \rightarrow \frac{c}{s Tl + 1}$$

$$\text{"G-zerlegt"} = \frac{c}{Tl \left(s + \frac{1}{Tl} \right)}$$

$$AA_{Proc} := \begin{bmatrix} -\frac{1}{Tl} \end{bmatrix}$$

$$BB_{Proc} := \begin{bmatrix} \frac{c}{Tl} \end{bmatrix}$$

$$CC_{Proc} := \begin{bmatrix} 1 \end{bmatrix}$$

$$DD_{Proc} := \begin{bmatrix} 0 \end{bmatrix}$$

$$\Phi_{Proc} := \begin{bmatrix} e^{-\frac{Ta}{Tl}} \end{bmatrix}$$

$$\Gamma_{Proc} := \begin{bmatrix} -\left(e^{-\frac{Ta}{Tl}} - 1\right) c \end{bmatrix}$$

$$xI(k+1) := e^{-\frac{Ta}{Tl}}.xI(k) - \left(\left(e^{-\frac{Ta}{Tl}} - 1\right) c\right).uI(k)$$

$$yI(k) := xI(k) + \begin{bmatrix} 0 \end{bmatrix}.uI(k)$$

(14.1.1)

▼ Punkt 2

> geeignete Zustaende: dc und S

```
xx:=<dc(k), S(k)>;
#dp(k)->a*S(k); #doesn't work, but you get the idea
```

Aufstellen der Gleichungen, mit denen das System beschrieben werden kann.

Glsys:=

```
[Phi[Proc][1][1]*dc(k) + Gamma[Proc][1]*a*S(k)=dc
(k+1), #Transitionsmatrix+Eingangsvektor = Ausgang vom
Prozessor
```

```
u(k)*Ta + dc(k)*Ta + (1-a*Ta)*S(k)=S(k+1)]:
#Eingang Pro Abtastzeit + Ausgang vom Prozessor - Entnomme
Daten vom Speicher = S
```

Transitionsmatrix des Gesamtsystems:

```

Phi[Ges]:=GenerateMatrix(Glsys, convert(xx, list))[1];
Eingangsvektor des Gesamtsystems:
Gamma[Ges]:=GenerateMatrix(Glsys, [u(k)])[1];
Ausgangsvektor des Gesamtsystems:
CC[Ges]:=GenerateMatrix([S(k)=y2(k)], convert(xx, list))
[1];
Durchgangsvektor des Gesamtsystems:
DD[Ges]:=GenerateMatrix([S(k)=y2(k)], [u(k)])[1];

```

Das Abtastsystem bildet sich zu: (Siehe 6.19a, b)

```

StateGL:=x2(k+1)=Phi[Ges].x2(k) + Gamma[Ges].u2(k); #F=x
[k+1]

```

```

OutGL:=y2(k)=CC[Ges].x2(k) + DD[Ges].u2(k); #y[k]

```

$$xx := \begin{bmatrix} dc(k) \\ S(k) \end{bmatrix}$$

$$\Phi_{Ges} := \begin{bmatrix} e^{-\frac{Ta}{Tl}} & -\left(e^{-\frac{Ta}{Tl}} - 1\right) c a \\ Ta & -Ta a + 1 \end{bmatrix}$$

$$\Gamma_{Ges} := \begin{bmatrix} 0 \\ Ta \end{bmatrix}$$

$$CC_{Ges} := \begin{bmatrix} 0 & 1 \end{bmatrix}$$

$$DD_{Ges} := \begin{bmatrix} 0 \end{bmatrix}$$

$$StateGL := x2(k+1) = \begin{bmatrix} e^{-\frac{Ta}{Tl}} & -\left(e^{-\frac{Ta}{Tl}} - 1\right) c a \\ Ta & -Ta a + 1 \end{bmatrix} x2(k) + \begin{bmatrix} 0 \\ Ta \end{bmatrix} u2(k)$$

$$OutGL := y2(k) = \begin{bmatrix} 0 & 1 \end{bmatrix} x2(k) + \begin{bmatrix} 0 \end{bmatrix} u2(k) \quad (14.2.1)$$

Punkt 3

- > Ruhelage bedeutet: $x3(k)=x3(k+1)=x3R$, denn in der Ruhelage aendert sich der Zustand nicht.

Aus der StateGl von vorhin folgt:

```

folgt1:=eval(StateGL, [x2(k)=x3R, u2(k)=u2R, x2(k+1)=x3R])
;

```

Das Einfuehren einer zusaetzlichen Einheitsmatrix fuehrt nicht zu einer veraenderung der Gleichung:

```

folgt2:=eval(StateGL, [x2(k)=x3R, u2(k)=u2R, x2(k+1)=
IdentityMatrix(2).x3R]);

```

Durch Weitere Umformungen kommt man schliesslich auf das folgende:

```

folgt3:=isolate(folgt2, x3R);
folgt4:=(IdentityMatrix(2)-Phi[Ges]).x3R=Gamma[Ges].u3R;
folgt5:=x3R=MatrixInverse(IdentityMatrix(2)-Phi[Ges]).
Gamma[Ges].u3R;

```

```

x3RParam:=eval(folgt5, params);

```

$$\begin{aligned}
\text{folgt1} &:= x3R = \begin{bmatrix} e^{-\frac{Ta}{Tl}} & -\left(e^{-\frac{Ta}{Tl}} - 1\right) c a \\ Ta & -Ta a + 1 \end{bmatrix} x3R + \begin{bmatrix} 0 \\ Ta \end{bmatrix} u2R \\
\text{folgt2} &:= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x3R = \begin{bmatrix} e^{-\frac{Ta}{Tl}} & -\left(e^{-\frac{Ta}{Tl}} - 1\right) c a \\ Ta & -Ta a + 1 \end{bmatrix} x3R + \begin{bmatrix} 0 \\ Ta \end{bmatrix} u2R \\
\text{folgt3} &:= \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} x3R - \begin{bmatrix} e^{-\frac{Ta}{Tl}} & -\left(e^{-\frac{Ta}{Tl}} - 1\right) c a \\ Ta & -Ta a + 1 \end{bmatrix} x3R = \begin{bmatrix} 0 \\ Ta \end{bmatrix} u2R \\
\text{folgt4} &:= \begin{bmatrix} 1 - e^{-\frac{Ta}{Tl}} & \left(e^{-\frac{Ta}{Tl}} - 1\right) c a \\ -Ta & a Ta \end{bmatrix} x3R = \begin{bmatrix} 0 \\ Ta \end{bmatrix} u3R \\
\text{folgt5} &:= x3R = \begin{bmatrix} -\frac{c}{c-1} \\ -\frac{1}{(c-1)a} \end{bmatrix} u3R \\
x3RParam &:= x3R = \begin{bmatrix} 1.000000000 \\ 5.000000000 \end{bmatrix} u3R
\end{aligned} \tag{14.3.1}$$

▼ Punkt 4

> Die Gleichung aus Punkt3:

```
x4R:=eval(MatrixInverse(IdentityMatrix(2)-Phi[Ges]).Gamma
[Ges]*u4R, params);
```

Gleichung 6.33a geplotted:

```
darstX:=x[4](k)=Phi^k.x[4[0]] + sum(Phi^(k-j-1).Gamma.u[4]
(j), j=0..k-1);
```

Damit folgt fuer den Sprung und fuer die Normale Gleichung:

```
x4:=(k)->map(simplify,MatrixPower(Phi[Ges], k)).x4Zero +
map(simplify,sum(map(simplify,MatrixPower(Phi[Ges], k-j-1)
) .Gamma[Ges]*u4(j), j=0..k-1)):
x4Sprung:=(k)->map(simplify,MatrixPower(Phi[Ges], k)).
x4SprungZero + map(simplify,sum(map(simplify,MatrixPower
(Phi[Ges], k-j-1)) .Gamma[Ges]*u4Sprung(j), j=0..k-1)):
```

mit den gegebenen Parametern (Datenrate geht sprungfoermig auf 1000*uR)

```
x4SprungParam:=(k)->evalf(eval(x4Sprung(k), convert
(params, set) union {u4Sprung(j)=1000*u4R, x4SprungZero=
x4R})):
```

Der Nachfolgende x4-Teil beginnt erst, wenn am Eingang 5ms die Datenrate angestiegen ist.

```
x4Param:=(k)->eval(evalf(eval(x4(k), convert(params, set)
union {u4(j)=u4R, x4Zero=x4Max})), [k=k-5000]):
sprungDannKonst:=(k)->piecewise(k<=5000, x4SprungParam(k),
k>5000, x4Param(k)): #5000...5ms=5000us
```

```
x4Max:=eval(x4SprungParam(k), k=5000);
plot([eval(sprungDannKonst(k)[1][1], u4R=1), eval
(sprungDannKonst(k)[2][1], u4R=1)], k=0..4*10^5);
```

$$x4R := \begin{bmatrix} 1.000000000 \, u4R \\ 5.000000000 \, u4R \end{bmatrix}$$

$$darstX := x_4(k) = \Phi^k x_{4_0} + \sum_{j=0}^{k-1} \Phi^{k-j-1} \Gamma u_4(j)$$

$$x4Max := \begin{bmatrix} 1.799603196 \, u4R \\ 9.989560417 \, u4R \end{bmatrix}$$

