Aufgabe 3.1 - El. System

Punkt 1 - Modellherleitung vom el. System

```
> restart;
```

Knoten- und Maschengleichungen:

```
gl1 := Ua/Uc2 = K ; #((K-1)*R3)/R3 ;
#Ausgangsspannungsteiler
gl2 := ic2 = i2 + is ; #Knotengleichung
gl3 := i1 = ic1 + i2 ; #Knotengleichung
gl4 := Ua = -Uc1 + Ux ;
```

Gleichung 1 in Gleichung 4 eingesetzt:

```
gl1und4 := isolate(eval(gl4, isolate(gl1, Ua)), Ux);
```

Stroeme ausgedrueckt durch Spannungen und Widerstaende:

```
gl5 := is = (Us-Uc2)/R2;
gl6 := i2=eval((Ux-Uc2)/R2, {gl1und4});
gl7 := i1=eval((Ue-Ux)/R1, {gl1und4});
```

Mit den Stroemen eingesetzt als Ausdruck von U und R ==> Gesuchte Stroeme ic2 und ic1 als Ausdruecke von U und R

```
ic2eval := isolate(eval(gl2, {gl5, gl6, gl7}), ic2);
ic1eval := isolate(eval(gl3, {gl5, gl6, gl7}), ic1);
```

$$gl1 := \frac{Ua}{Uc2} = K$$

$$gl2 := ic2 = i2 + is$$

$$gl3 := i1 = ic1 + i2$$

$$gl4 := Ua = -Uc1 + Ux$$

$$gl1und4 := Ux = K\,Uc2 + Uc1$$

$$gl5 := is = \frac{Us - Uc2}{R2}$$

$$gl6 := i2 = \frac{K\,Uc2 + Uc1 - Uc2}{R2}$$

$$gl7 := i1 = \frac{-K\,Uc2 - Uc1 + Ue}{R1}$$

$$ic2eval := ic2 = \frac{K\,Uc2 + Uc1 - Uc2}{R2} + \frac{Us - Uc2}{R2}$$

$$ic1eval := ic1 = \frac{-K\,Uc2 - Uc1 + Ue}{R1} - \frac{K\,Uc2 + Uc1 - Uc2}{R2} \qquad \textbf{(1.1.1)}$$

```
> with(VectorCalculus, Jacobian):#Achtung, Jacobian nur so
einbinden, das VectorCalculus Paket macht sonst sehr viel
Bloedsinn.
with(LinearAlgebra):
xx := <Uc1, Uc2> ;
#Zustandsvektor
uu := <Ue> ;
#Eingangsvektor
dd := <Us> ;
#Stoerungsvektor
ud := <Ue,Us>;                                    #Eingangs-
und Stoerungsvektor
ff := <1/C1*rhs(ic1eval), 1/C2*rhs(ic2eval)> ;
```

```
#Zustandsdifferentialgleichungen =xpunkt
hh := <rhs(isolate(gl1, Ua))> ;
#Ausgangsgleichungen

AA:=simplify(Jacobian(ff, convert(xx, list)));
#Systemmatrix
BBu:=simplify(Jacobian(ff, convert(uu, list)));
#Eingangsvektor nur mit dem Stelleingang
BBd:=simplify(Jacobian(ff, convert(dd, list)));
#Eingangsvektor nur mit der Störgrösse
BB:=simplify(Jacobian(ff, convert(ud, list)));
#Eingangsvektor mit Stoerung und Stelleingang
CC:=simplify(Jacobian(hh, convert(xx, list)));
#Ausgangsmatrix
DD:=simplify(Jacobian(hh, convert(ud, list)));
#Durchgangsmatrix mit Stoerung und Stelleingang
```

$$xx := \begin{bmatrix} Uc1 \\ Uc2 \end{bmatrix}$$

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

$$dd := \begin{bmatrix} Us \end{bmatrix}$$

$$ud := \begin{bmatrix} Ue \\ Us \end{bmatrix}$$

$$ff := \begin{bmatrix} \dfrac{-K\,Uc2 - Uc1 + Ue}{R1} - \dfrac{K\,Uc2 + Uc1 - Uc2}{R2} \\ \hline C1 \\ \dfrac{K\,Uc2 + Uc1 - Uc2}{R2} + \dfrac{Us - Uc2}{R2} \\ \hline C2 \end{bmatrix}$$

$$hh := \begin{bmatrix} K\,Uc2 \end{bmatrix}$$

$$AA := \begin{bmatrix} -\dfrac{R2 + R1}{C1\,R1\,R2} & -\dfrac{K\,R1 + K\,R2 - R1}{C1\,R1\,R2} \\ \dfrac{1}{C2\,R2} & \dfrac{K - 2}{C2\,R2} \end{bmatrix}$$

$$BBu := \begin{bmatrix} \dfrac{1}{C1\,R1} \\ 0 \end{bmatrix}$$

$$BBd := \begin{bmatrix} 0 \\ \dfrac{1}{C2\,R2} \end{bmatrix}$$

$$BB := \begin{bmatrix} \dfrac{1}{C1\,R1} & 0 \\ 0 & \dfrac{1}{C2\,R2} \end{bmatrix}$$

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

$$DD := \begin{bmatrix} 0 & 0 \end{bmatrix} \qquad \textbf{(1.1.2)}$$

Punkt 2 - Uebertragungsfunktionen bestimmen

```
> with(DynamicSystems):
```

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)

```
G:=(s)->simplify((CC.MatrixInverse(s*IdentityMatrix(2)-AA)
).BB+DD):
Gu:=(s)->G(s)(1):
Gd:=(s)->G(s)(2):
Gu:=Gu(s);
Gd:=Gd(s);
```

$$Gu := (K\ R2) \Big/ (C1\ C2\ R1\ R2^2\ s^2 - C1\ K\ R1\ R2\ s + 2\ C1\ R1\ R2\ s + C2\ R1\ R2\ s + C2\ R2^2\ s + R1 + 2\ R2)$$

$$Gd := (K\ (C1\ R1\ R2\ s + R1 + R2)) \Big/ (C1\ C2\ R1\ R2^2\ s^2 - C1\ K\ R1\ R2\ s + 2\ C1\ R1\ R2\ s + C2\ R1\ R2\ s + C2\ R2^2\ s + R1 + 2\ R2) \qquad \textbf{(1.2.1)}$$

Punkt 3 und 4 - Widerstaende bestimmen

```
> PT2:=V/(1+2*xi*s*T+(s*T)^2);
      #Struktur eines PT2-Gliedes lt. Angabe
  params:= {C1=1*10^(-6), C2=1*10^(-6), K=3, xi=0.8, T=10^
  (-3)} : #Parameter lt. Angabe
  V:=numer(Gu);
```

Extract Coeffs:

```
GuCoeffs:=<coeffs(eval(denom(Gu), params),s)>;
PT2Coeffs:=<coeffs(eval(denom(PT2), params),s)>;
```

Divide through first Coefficient

```
gl1:=PT2Coeffs(2)=GuCoeffs(2)/GuCoeffs(1);
gl2:=PT2Coeffs(3)=GuCoeffs(3)/GuCoeffs(1);
glsys:={gl1,gl2, R1>0, R2>0};
res:=solve(glsys, {R1,R2});

Gparam:=(s)->eval(G(s), res union params) :
Gparam:=Gparam(s) ;
```

Im Bode-Plot ist ersichtlich, dass es sich um ein Aktives Tiefpassfilter handelt.
Man kann dabei beobachten, dass die Stoergroessen mit einem Tiefpassfilter 1. Ordnung
und der normale Eingang mit einem Tiefpassfilter 2. Ordnung weggefiltert werden.

Ausserdem ist ersichtlich, dass es BIBO-Stabil ist, denn die Phasenreserve bei der Durchtrittsfrequenz ist Positiv (>-180Grad)

```
BodePlot(TransferFunction(Gparam), numpoints=500); #nur
möglich mit Zahlen eingesetzt

AAparams:=eval(AA, res union params);
BBuparams:=eval(BBu, res union params);
BBdparams:=eval(BBd, res union params);
```

```
CCparams:=eval(CC, res union params);
```

$$PT2 := \frac{V}{T^2 s^2 + 2 T s \xi + 1}$$

$$V := K R2$$

$$GuCoeffs := \begin{bmatrix} R1 + 2 R2 \\ \frac{1}{1000000} R2^2 \\ \frac{1}{1000000000000} R1 R2^2 \end{bmatrix}$$

$$PT2Coeffs := \begin{bmatrix} 1 \\ 0.001600000000 \\ \frac{1}{1000000} \end{bmatrix}$$

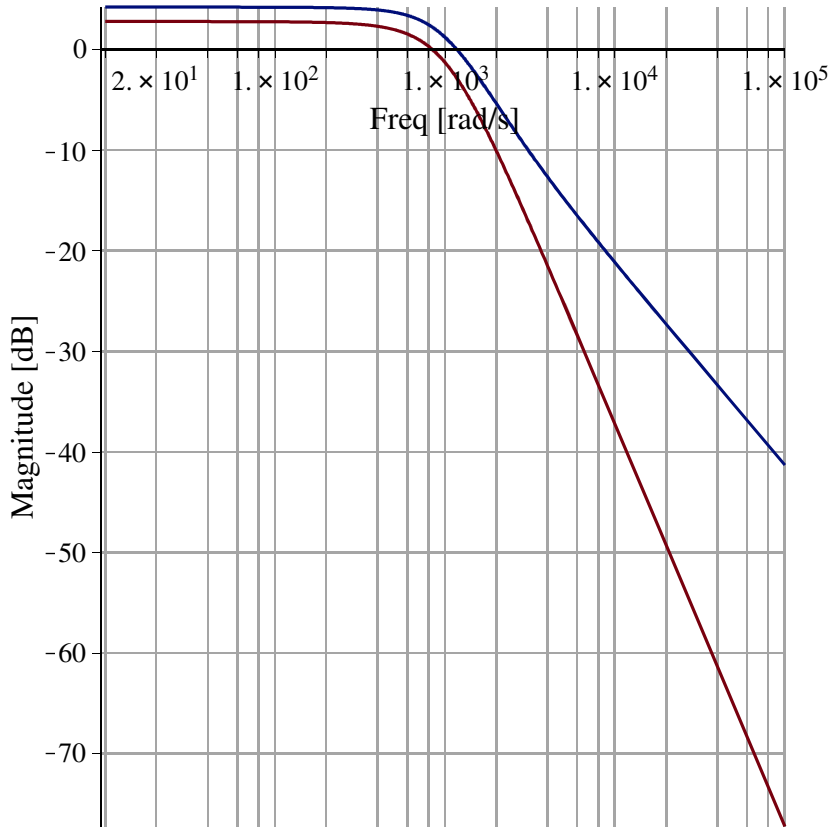
$$gl1 := 0.001600000000 = \frac{1}{1000000} \frac{R2^2}{R1 + 2 R2}$$

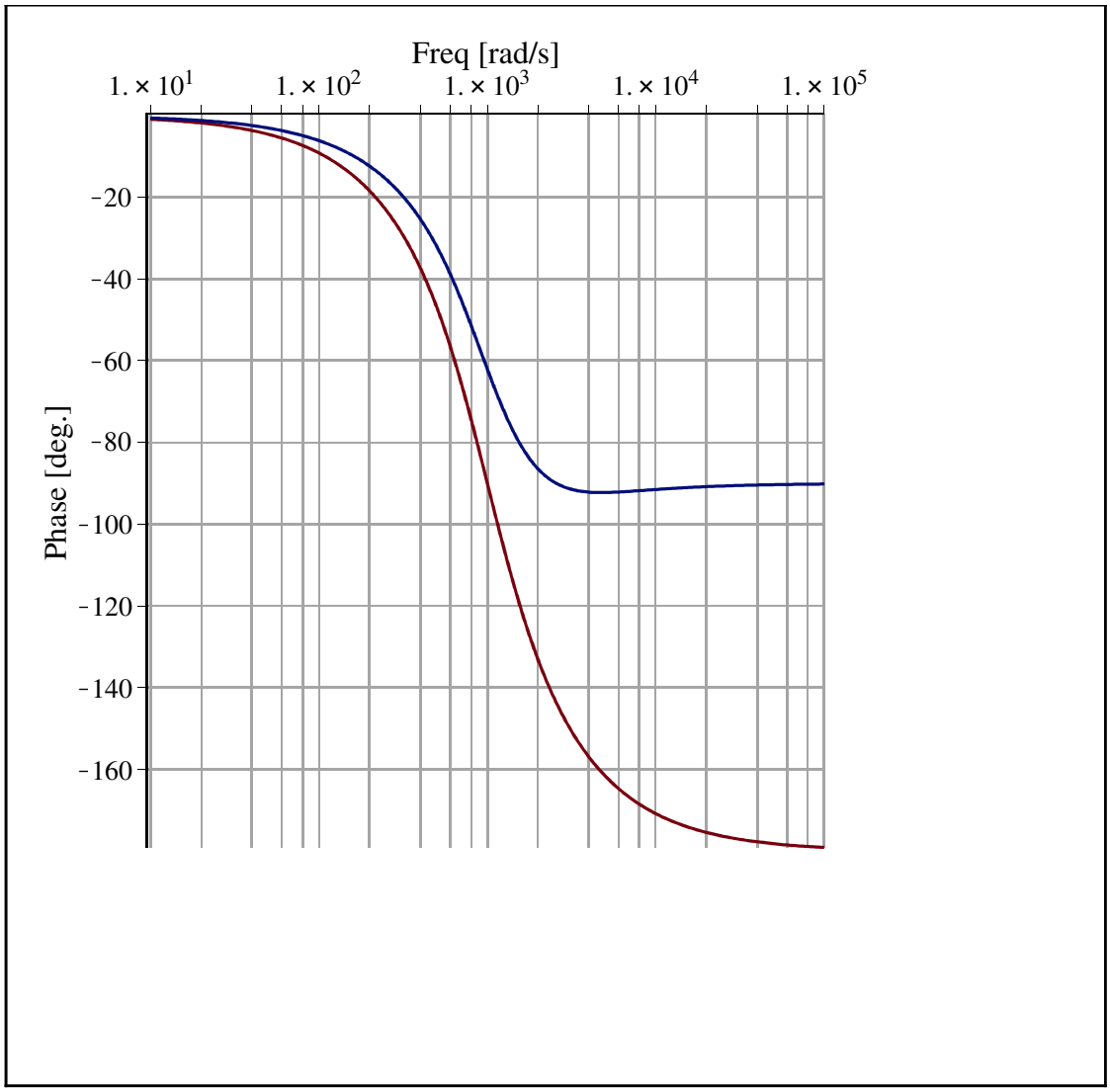
$$gl2 := \frac{1}{1000000} = \frac{1}{1000000000000} \frac{R1 R2^2}{R1 + 2 R2}$$

$$glsys := \left\{ \frac{1}{1000000} = \frac{1}{1000000000000} \frac{R1 R2^2}{R1 + 2 R2}, 0.001600000000 = \frac{1}{1000000} \frac{R2^2}{R1 + 2 R2}, 0 < R1, 0 < R2 \right\}$$

$$res := \{R1 = 625., R2 = 3486.796226\}$$

$$Gparam := \left[\frac{10460.38868}{0.007598592450 s^2 + 12.15774792 s + 7598.592452}, \frac{3 (2.179247641 s + 4111.796226)}{0.007598592450 s^2 + 12.15774792 s + 7598.592452} \right]$$


0
-10
-20
-30
-40
-50
-60
-70
Magnitude [dB]
2. × 10^1
1. × 10^2
1. × 10^3
1. × 10^4
1. × 10^5
Freq [rad/s]



$$AAparams := \begin{bmatrix} -1886.796227 & -5373.592454 \\ 286.7962264 & 286.7962264 \end{bmatrix}$$

$$BBuparams := \begin{bmatrix} 1600.000000 \\ 0 \end{bmatrix}$$

$$BBdparams := \begin{bmatrix} 0 \\ 286.7962264 \end{bmatrix}$$

$$CCparams := \begin{bmatrix} 0 & 3 \end{bmatrix} \quad \textbf{(1.3.1)}$$

Aufgabe 3.2 - El. System (Matlab)

```
> with(CodeGeneration):
  printf("Zu 1. Absatz einfuegen...\n");
  Matlab(eval(R1, res), resultname=R1):
  Matlab(eval(R2, res), resultname=R2):
  printf("\nZu 3. Absatz einfuegen...\n");
  Matlab(AA, resultname=Asys):
  Matlab(BBu, resultname=busys):
  Matlab(BBd, resultname=bdsys):
  Matlab(CC, resultname=csys):
  Matlab(DD, resultname=dsys):
```

```
  printf("\nbsys=[busys bdsys];");
Zu 1. Absatz einfuegen...
R1 = 0.625e3;
R2 = 0.3486796226e4;

Zu 3. Absatz einfuegen...
Asys = [-(R2 + R1) / C1 / R1 / R2 -(K * R1 + K * R2 - R1) /
C1 / R1 / R2; 0.1e1 / C2 / R2 (K - 2) / C2 / R2;];
busys = [0.1e1 / C1 / R1; 0;];
bdsys = [0; 0.1e1 / C2 / R2;];
csys = [0 K;];
dsys = [0 0;];

bsys=[busys bdsys];
```

Aufgabe 3.4 - GSM

Punkt 1 - Zustandstransformation von PhiGSM-PhiP=PhiGSMP

```
> restart:
  Mgleichungen := {MP = dcP + dvP*wP + dqP*wP^2 + Mext,
  MrGSM = dcGSM + dvGSM*wGSM, Mkopp = (wGSM - wP)*dGSMP +
  (phiGSM - phiP)*cGSMP} :
  Jgleichungen := {diff(apply(wGSM, t), t)=(MGSM - MrGSM -
  Mkopp)/JGSM, diff(apply(wP, t), t) = (Mkopp - MP)/JP} :
  diffgleichungen := {diff(apply(iGSM, t), t)=1/LGSM*(uGSM -
  RGSM*iGSM - kGSM*wGSM), diff(apply(phiGSM, t), t)=wGSM,
  diff(apply(phiP, t), t)=wP} :
  zusaetzlich := {MGSM = iGSM*kGSM} :
  funcs := {uGSM = um, Mext = dm} :
```

Nichtreduzierte Systembeschreibung:

```
  xm := <iGSM, phiGSM, wGSM, phiP, wP> ;
                          #Zustandsvektor
  fm := eval(map(diff, map(apply, xm, t), t),
  diffgleichungen union Jgleichungen) ;
  #Zustandsdifferentialgleichungen xpunkt
```

Reduktion mit der Matrix zstdtrans:

```
  zstdtrans:=<<1|0|0|0|0>,<0|1|0|-1|0>,<0|0|1|0|0>,
  <0|0|0|0|1>>;
  xM := eval(zstdtrans.xm, phiGSM-phiP=phiGSMP) ;
  fM := eval(zstdtrans.fm, Mgleichungen) :
  fM := eval(fM, funcs union {phiGSM-phiP=phiGSMP}) :
  fM := eval(fM, zusaetzlich) ;
```

$$xm := \begin{bmatrix} iGSM \\ phiGSM \\ wGSM \\ phiP \\ wP \end{bmatrix}$$

$$fm := \begin{bmatrix} \frac{-RGSM\,iGSM - kGSM\,wGSM + uGSM}{LGSM} \\ wGSM \\ \frac{MGSM - MrGSM - Mkopp}{JGSM} \\ wP \\ \frac{Mkopp - MP}{JP} \end{bmatrix}$$

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

$$xM := \begin{bmatrix} iGSM \\ phiGSMP \\ wGSM \\ wP \end{bmatrix}$$

$$fM := \left[\left[\frac{-RGSM\,iGSM - kGSM\,wGSM + um}{LGSM} \right], \left[wGSM - wP \right], \left[\frac{1}{JGSM} \left(iGSM\,kGSM - dvGSM\,wGSM - dcGSM - (wGSM - wP)\,dGSMP - phiGSMP\,cGSMP \right) \right], \left[\frac{1}{JP} \left((wGSM - wP)\,dGSMP + phiGSMP\,cGSMP - dqP\,wP^2 - dvP\,wP - dm - dcP \right) \right] \right] \quad \textbf{(3.1.1)}$$

Punkt 2 - Ruhelagen bestimmen, linearisieren bzgl. der Ruhelage

> Ruhezustandsmatrix mittels Gleichungen aus den Zustandsdifferentialgleichungen aufstellen (symbolisch)

```
xMruhe := <0,0,0,0> :
gl3 := isolate(fM(2)=0, wGSM):
xMruhe(3) := rhs(gl3):
gl1 := isolate(fM(1)=0, iGSM):
xMruhe(1) := rhs(eval(gl1, wGSM=xMruhe(3))):
gl2 := isolate(fM(3)=0, phiGSMP):
xMruhe(2) := rhs(eval(gl2, {iGSM=xMruhe(1), wGSM=xMruhe(3)
})):
gl4 := isolate(eval(fM(4)=0, {iGSM=xMruhe(1), phiGSMP=
xMruhe(2), wGSM=xMruhe(3)}), wP):
xMruhe(4) := rhs(gl4):

xMruhe(1) := eval(xMruhe(1), wP=xMruhe(4)) :
```

```
xMruhe(2) := eval(xMruhe(2), wP=xMruhe(4)) :
xMruhe(3) := eval(xMruhe(3), wP=xMruhe(4)) :

Ruhezustand:=xMruhe; #sehr lang...
```

$$Ruhezustand := \Bigg[\Bigg[-\frac{1}{RGSM}\Bigg(\frac{1}{2}\,\frac{1}{RGSM\,dqP}\Big(kGSM\Big(-RGSM\,dvGSM - RGSM\,dvP - kGSM^2 + \big(-4\,RGSM^2\,dcGSM\,dqP - 4\,RGSM^2\,dcP\,dqP - 4\,RGSM^2\,dm\,dqP + RGSM^2\,dvGSM^2 + 2\,RGSM^2\,dvGSM\,dvP + RGSM^2\,dvP^2 + 4\,RGSM\,dqP\,kGSM\,um + 2\,RGSM\,dvGSM\,kGSM^2 + 2\,RGSM\,dvP\,kGSM^2 + kGSM^4\big)^{1/2}\Big)\Big) - um\Bigg)\Bigg], \tag{3.2.1}$$

$$\Bigg[-\frac{1}{cGSMP}\Bigg(\frac{1}{2}\,\frac{1}{RGSM\,dqP}\Big(dvGSM\Big(-RGSM\,dvGSM - RGSM\,dvP - kGSM^2 + \big(-4\,RGSM^2\,dcGSM\,dqP - 4\,RGSM^2\,dcP\,dqP - 4\,RGSM^2\,dm\,dqP + RGSM^2\,dvGSM^2 + 2\,RGSM^2\,dvGSM\,dvP + RGSM^2\,dvP^2 + 4\,RGSM\,dqP\,kGSM\,um + 2\,RGSM\,dvGSM\,kGSM^2 + 2\,RGSM\,dvP\,kGSM^2 + kGSM^4\big)^{1/2}\Big)\Big) + \frac{1}{RGSM}\Bigg(\Bigg(\frac{1}{2}\,\frac{1}{RGSM\,dqP}\Big(kGSM\Big(-RGSM\,dvGSM - RGSM\,dvP - kGSM^2 + \big(-4\,RGSM^2\,dcGSM\,dqP - 4\,RGSM^2\,dcP\,dqP - 4\,RGSM^2\,dm\,dqP + RGSM^2\,dvGSM^2 + 2\,RGSM^2\,dvGSM\,dvP + RGSM^2\,dvP^2 + 4\,RGSM\,dqP\,kGSM\,um + 2\,RGSM\,dvGSM\,kGSM^2 + 2\,RGSM\,dvP\,kGSM^2 + kGSM^4\big)^{1/2}\Big)\Big) - um\Bigg)\,kGSM\Bigg) + dcGSM\Bigg)\Bigg],$$

$$\Bigg[\frac{1}{2}\,\frac{1}{RGSM\,dqP}\Big(-RGSM\,dvGSM - RGSM\,dvP - kGSM^2 + \big(-4\,RGSM^2\,dcGSM\,dqP - 4\,RGSM^2\,dcP\,dqP - 4\,RGSM^2\,dm\,dqP + RGSM^2\,dvGSM^2 + 2\,RGSM^2\,dvGSM\,dvP + RGSM^2\,dvP^2$$

$$+ 4\,RGSM\,dqP\,kGSM\,um + 2\,RGSM\,dvGSM\,kGSM^2 + 2\,RGSM\,dvP\,kGSM^2$$
$$+ kGSM^4)^{1/2}\Big)\Big],$$
$$\Big[\frac{1}{2}\ \frac{1}{RGSM\,dqP}\Big(-RGSM\,dvGSM - RGSM\,dvP - kGSM^2$$
$$+ \big(-4\,RGSM^2\,dcGSM\,dqP - 4\,RGSM^2\,dcP\,dqP - 4\,RGSM^2\,dm\,dqP$$
$$+ RGSM^2\,dvGSM^2 + 2\,RGSM^2\,dvGSM\,dvP + RGSM^2\,dvP^2$$
$$+ 4\,RGSM\,dqP\,kGSM\,um + 2\,RGSM\,dvGSM\,kGSM^2 + 2\,RGSM\,dvP\,kGSM^2$$
$$+ kGSM^4)^{1/2}\Big)\Big]\Big]$$

> linearisierte Systemmatrizen bestimmen:

```
with(VectorCalculus, Jacobian):with(LinearAlgebra):
#Achtung, Jacobian nur so einbinden, das VectorCalculus
Paket macht sonst sehr viel Bloedsinn.
ruheReplacements := {xM(1)=xMruhe(1), xM(2)=xMruhe(2), xM
(3)=xMruhe(3), xM(4)=xMruhe(4)} :
params := {LGSM=1.4*10^(-3), RGSM=0.46, kGSM=0.1, JGSM =
12.4*10^(-3), dcGSM=0.152, dvGSM=1.8*10^(-3), JP=32.5*10^
(-3), dcP=0.169, dvP=2.7*10^(-3), dqP=1*10^(-4), cGSMP=
0.6822, dGSMP=1*10^(-5), um=5.6, dm=0} :
xx := xM :
uu := <um> :
dd := <dm> :
ud := <um,dm> :
ff := fM :       #Zustandsdifferentialgleichungen
hh := <wP> :     #Ausgang

AA:=simplify(eval(Jacobian(ff, convert(xx, list)),
ruheReplacements)):#Systemmatrix
BBu:=simplify(eval(Jacobian(ff, convert(uu, list)),
ruheReplacements)):#Eingangsvektor nur mit der Stoerung
BBd:=simplify(eval(Jacobian(ff, convert(dd, list)),
ruheReplacements)):#Eingangsvektor nur mit dem
Stelleingang
BB:=simplify(eval(Jacobian(ff, convert(ud, list)),
ruheReplacements)):#Eingangsvektor mit Stoerung und
Stelleingang
CC:=simplify(eval(Jacobian(hh, convert(xx, list)),
ruheReplacements)):#Ausgangsmatrix
DD:=simplify(eval(Jacobian(hh, convert(ud, list)),
ruheReplacements)):#Durchgangsmatrix mit Stoerung und
Stelleingang

AAparam:=map(simplify, eval(AA, params));#Systemmatrix
BBuparam:=map(simplify, eval(BBu, params));#Eingangsvektor
mit Stelleingang
BBdparam:=map(simplify, eval(BBd, params));#Eingangsvektor
mit Stoerung
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
```

Eigenwerte der Dynamikmatrix A(Systemmatrix)

```
'EigenA'=Eigenvalues(AAparam);
```

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 assymptotisch stabil!

$$AAparam := \begin{bmatrix} -328.5714286 & 0 & -71.42857143 & 0 \\ 0 & 0 & 1 & -1 \\ 8.064516129 & -55.01612903 & -0.1459677419 & 0.0008064516129 \\ 0 & 20.99076923 & 0.0003076923077 & -0.2716614823 \end{bmatrix}$$

$$BBuparam := \begin{bmatrix} 714.2857143 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$BBdparam := \begin{bmatrix} 0 \\ 0 \\ 0 \\ -30.76923077 \end{bmatrix}$$

$$BBparam := \begin{bmatrix} 714.2857143 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & -30.76923077 \end{bmatrix}$$

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

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

$$EigenA = \begin{bmatrix} -326.808938596654 + 0.\,\mathrm{I} \\ -0.726627399777231 + 8.67396631587622\,\mathrm{I} \\ -0.726627399777231 - 8.67396631587622\,\mathrm{I} \\ -0.726864427991629 + 0.\,\mathrm{I} \end{bmatrix} \quad \textbf{(3.2.2)}$$

Punkt 3 - Berechnen der Uebertragungsfunktion, Sprungantwort

```
> with(inttrans):
```

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)[1,1];
```

Ruhelage des Ausgangs berechnen:

```
hhRuheRaw := eval(hh, ruheReplacements):
```

```
hhRuhe := eval(hhRuheRaw, params);
```

Sprungantwort berechnen: (h(t)=L^(-1)(G(s)/s)) --> L(h(t)) = G(s)/s (aus Wikipedia)

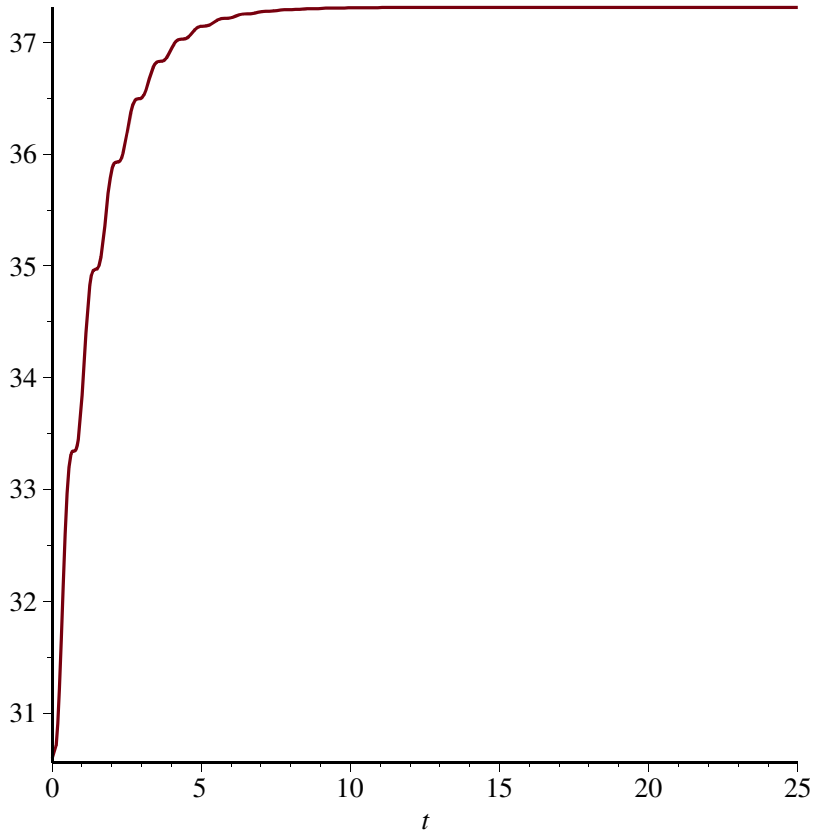
```
hhM  := unapply(invlaplace(G(s)[1,1]/s, s, t), t):
plot(hhM(t)+hhRuhe(1), t = 0..25);
```

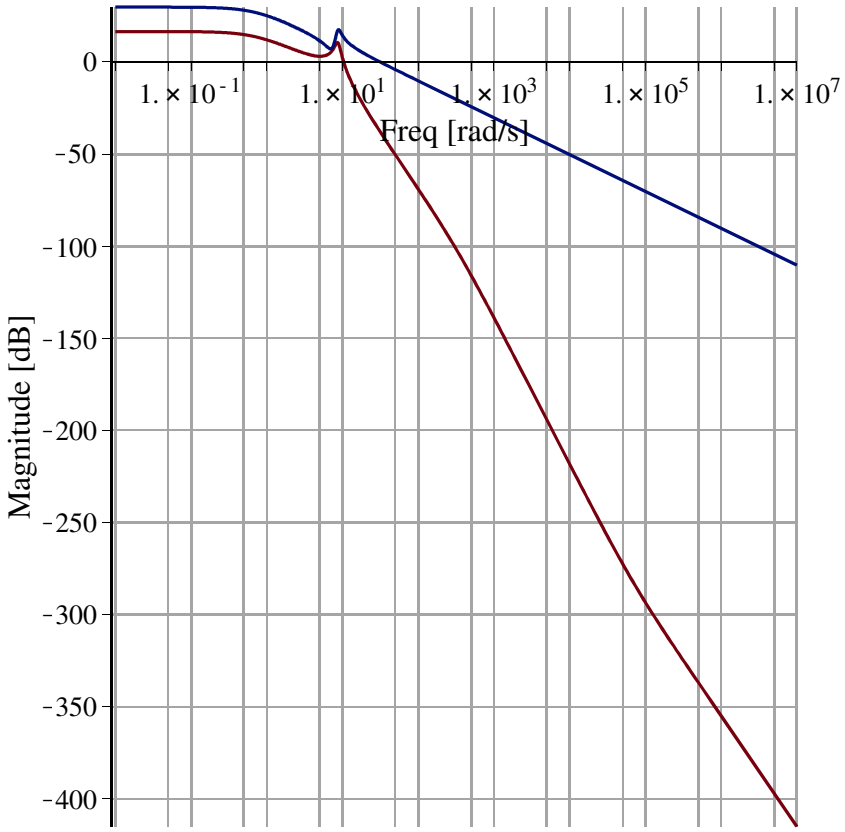
Bodediagramm:

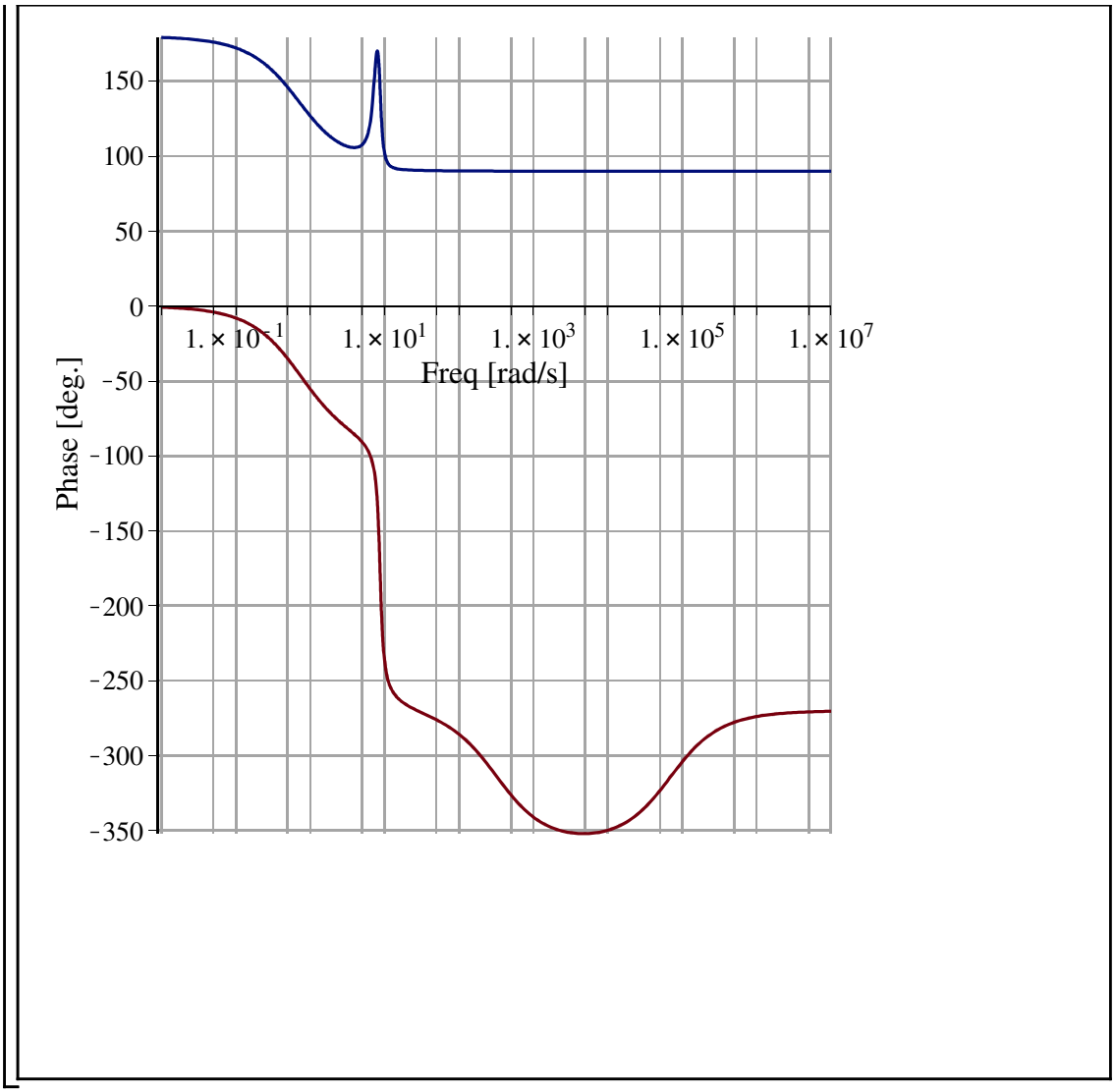
```
with(DynamicSystems):
BodePlot(TransferFunction(G(s)), numpoints=500);
```

$$(0.000009200000000\ (0.05000000000\ s + 3411.)) \Big/ (0.006530133207\ s + 0.004671009766 + 0.00008538318815\ s^3 + 2.595320000\ 10^{-7}\ s^4 + 0.0002048497622\ s^2)$$

$$hhRuhe := \left[\ 30.59499086\ \right]$$




0
-50
-100
-150
-200
-250
-300
-350
-400
Magnitude [dB]
1. × 10^-1
1. × 10^1
1. × 10^3
1. × 10^5
1. × 10^7
Freq [rad/s]



Aufgabe 3.5 - GSM Reduziert

Punkt 1, 2 - Berechnen der reduzierten GSM (Ruhelage, Linearisierung)

```
> zstdtrans2:=<<0|1|0|0>,<0|0|1|0>,<0|0|0|1>>;
  xRed := zstdtrans2.xM;
  gl := iGSM=1/RGSM*um-kGSM/RGSM*wGSM;
  fRed := eval(zstdtrans2.fM, gl);
  xRedRuhe := zstdtrans2.xMruhe:
  ruheReplacementsRed := {xRed(1)=xRedRuhe(1), xRed(2)=
  xRedRuhe(2), xRed(3)=xRedRuhe(3)}:

  xxRed := xRed :
  uuRed := <um> :
  ddRed := <dm> :
  udRed := <um,dm> :
  ffRed := fRed :
  hhRed := <wP> :

  AAred:=simplify(eval(Jacobian(ffRed, convert(xxRed, list)
  ), ruheReplacementsRed)):#Systemmatrix
  BBured:=simplify(eval(Jacobian(ffRed, convert(uuRed, list)
  ), ruheReplacementsRed)):#Eingangsvektor nur mit der
```

```
Stoerung
BBdred:=simplify(eval(Jacobian(ffRed, convert(ddRed, list)
), ruheReplacementsRed)):#Eingangsvektor nur mit dem
Stelleingang
BBred:=simplify(eval(Jacobian(ffRed, convert(udRed, list)
), ruheReplacementsRed)):#Eingangsvektor mit Stoerung und
Stelleingang
CCred:=simplify(eval(Jacobian(hhRed, convert(xxRed, list)
), ruheReplacementsRed)):#Ausgangsmatrix
DDred:=simplify(eval(Jacobian(hhRed, convert(udRed, list)
), ruheReplacementsRed)):#Durchgangsmatrix mit Stoerung
und Stelleingang

AAparamred:=map(simplify, eval(AAred, params));
#Systemmatrix
BBuparamred:=map(simplify, eval(BBured, params));
#Eingangsvektor mit Stelleingang
BBdparamred:=map(simplify, eval(BBdred, params));
#Eingangsvektor mit Stoerung
BBparamred:=map(simplify, eval(BBred, params));
#Eingangsvektor mit Stoerung und Stelleingang
CCparamred:=map(simplify, eval(CCred, params));
#Ausgangsmatrix
DDparamred:=map(simplify, eval(DDred, params));
#Durchgangsmatrix mit Stoerung und Stelleingang

'EigenAred'=Eigenvalues(AAparamred);
```

$$zstdtrans2 := \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$xRed := \begin{bmatrix} phiGSMP \\ wGSM \\ wP \end{bmatrix}$$

$$gl := iGSM = \frac{um}{RGSM} - \frac{kGSM\, wGSM}{RGSM}$$

$$fRed := \left[\left[wGSM - wP \right], \left[\frac{1}{JGSM} \left(\left(\frac{um}{RGSM} - \frac{kGSM\, wGSM}{RGSM} \right) kGSM - dvGSM\, wGSM - dcGSM - (wGSM - wP)\, dGSMP - phiGSMP\, cGSMP \right) \right], \left[\frac{1}{JP} \left((wGSM - wP)\, dGSMP + phiGSMP\, cGSMP - dqP\, wP^2 - dvP\, wP - dm - dcP \right) \right] \right]$$

$$AAparamred := \begin{bmatrix} 0 & 1 & -1 \\ -55.01612903 & -1.899123422 & 0.0008064516129 \\ 20.99076923 & 0.0003076923077 & -0.2716614823 \end{bmatrix}$$

$$BBuparamred := \begin{bmatrix} 0 \\ 17.53155680 \\ 0 \end{bmatrix}$$

$$BBdparamred := \begin{bmatrix} 0 \\ 0 \\ -30.76923077 \end{bmatrix}$$

$$BBparamred := \begin{bmatrix} 0 & 0 \\ 17.53155680 & 0 \\ 0 & -30.76923077 \end{bmatrix}$$

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

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

$$EigenAred = \begin{bmatrix} -0.722513095253796 + 8.65748526845528\,I \\ -0.722513095253796 - 8.65748526845528\,I \\ -0.725758713792401 + 0.\,I \end{bmatrix} \quad \textbf{(4.1.1)}$$

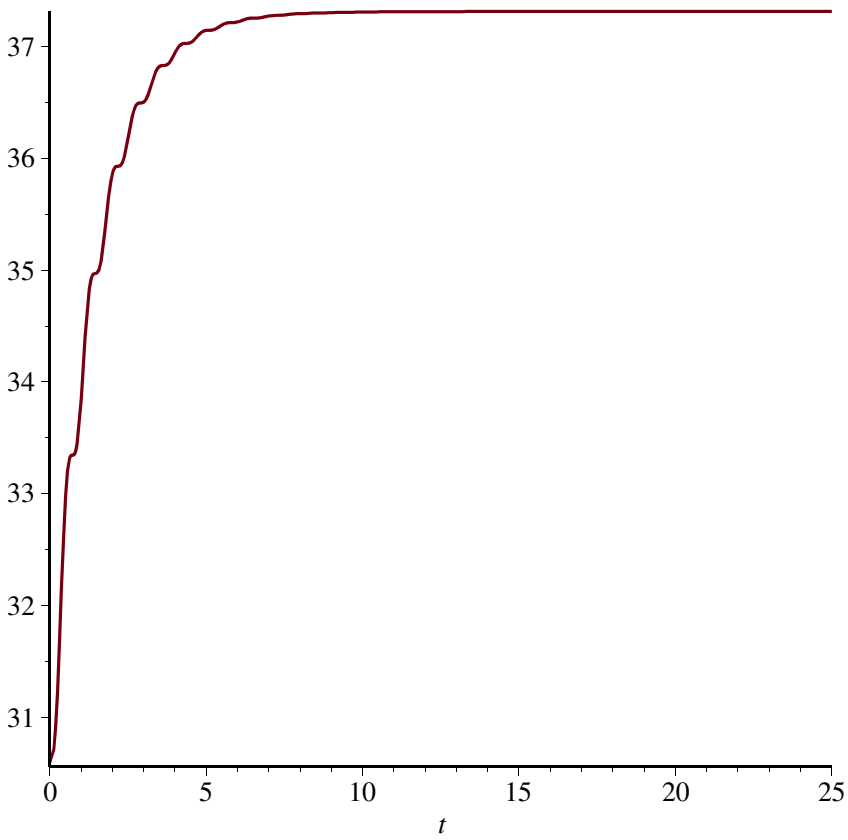
Punkt 3 - Ruhelage reduziert verglichen mit Ruhelage nicht reduziert.

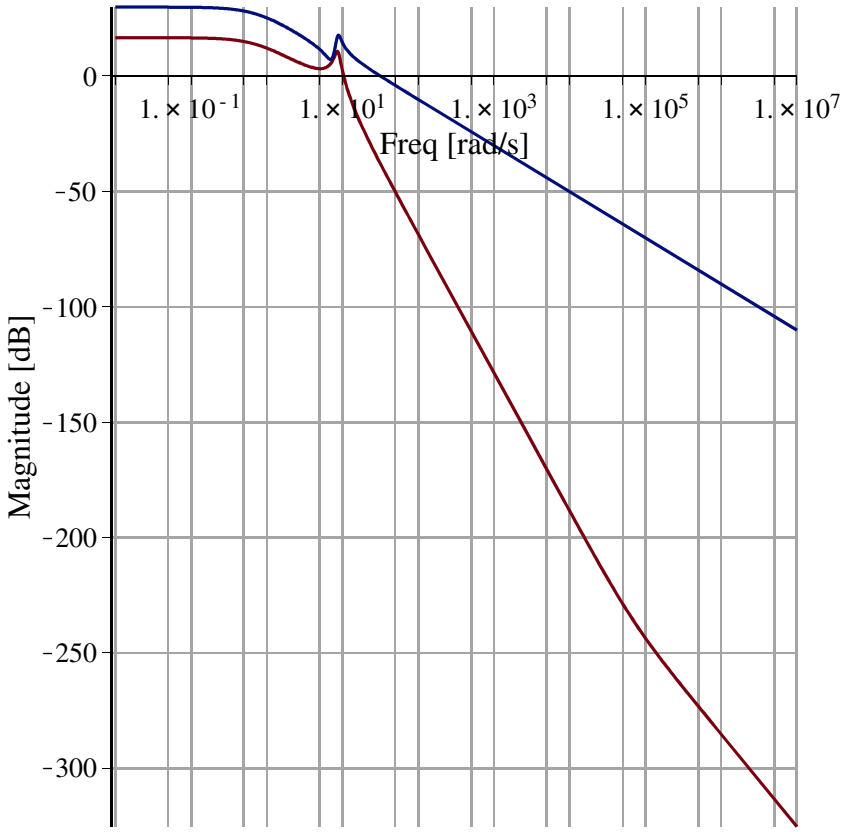
> Ja, beide Ruhelagen sind gleich, da es sich dabei um eine Abbildung eines 4D-Raums in einen 3D-Raum handelt. Der Unterraum stellt dabei nur eine Ebene dar.

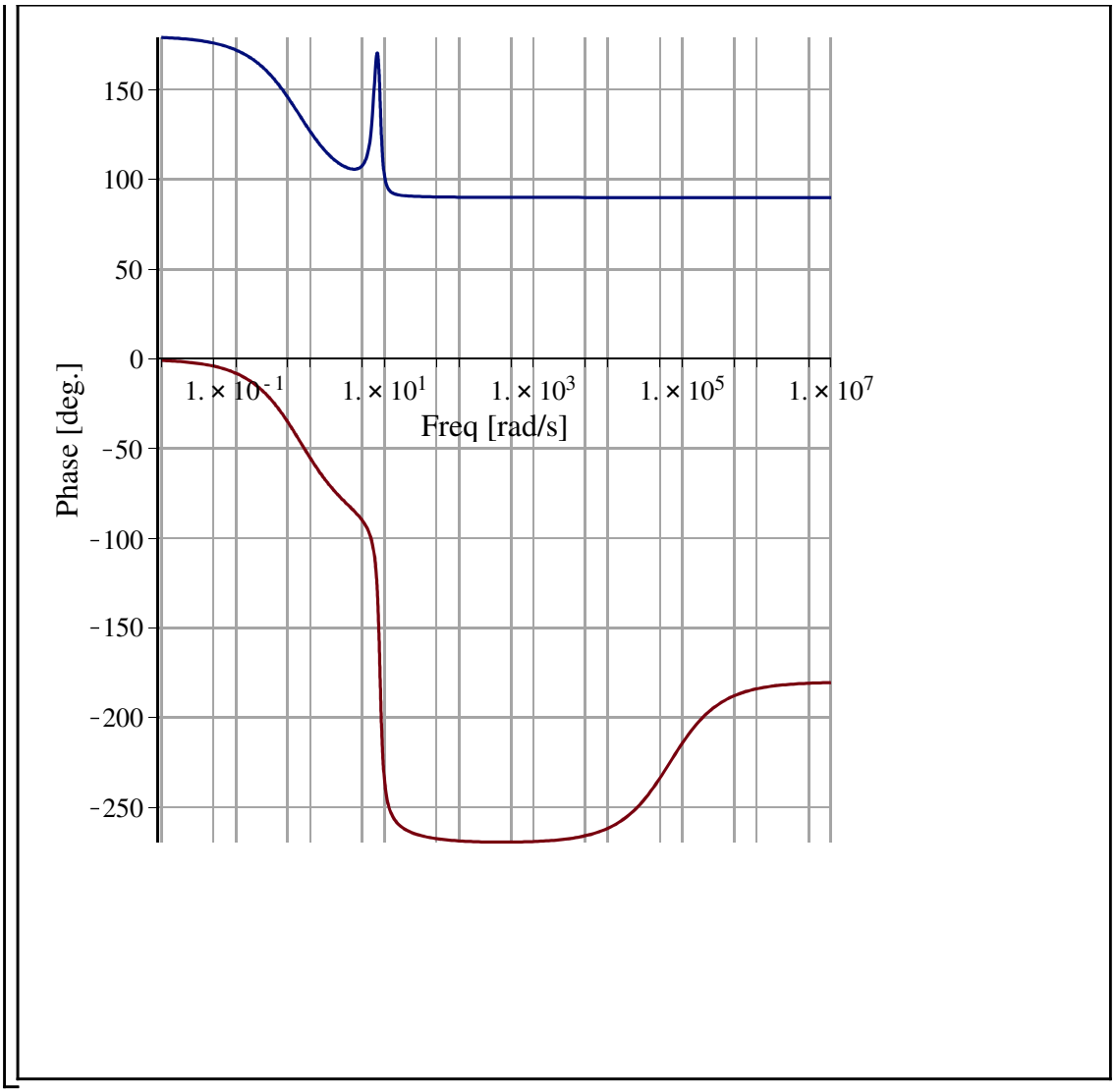
Punkt 4 - Berechnen der Uebertragungsfunktion, Sprungantwort

```
> rawGred:=(s)->simplify((CCred.MatrixInverse(s*
  IdentityMatrix(3)-AAred)).BBred+DDred):
  Gred:=(s)->simplify(eval(rawGred(s), params)):

  Ruhelage des Ausgangs berechnen:
  hhRedRuheRaw := eval(hhRed, ruheReplacements):
  hhRedRuhe := eval(hhRedRuheRaw, params);

  Sprungantwort berechnen: (h(t)=L^(-1)(G(s)/s))  --> L(h(t)) = G(s)/s (aus Wikipedia)
  hhRedM  := unapply(invlaplace(Gred(s)[1,1]/s, s, t), t):
  plot(hhRedM(t)+hhRedRuhe(1), t = 0..25);

  Bode Diagramm:
  with(DynamicSystems):
  BodePlot(TransferFunction(Gred(s)), numpoints=500);
```

$$hhRedRuhe := \begin{bmatrix} 30.59499086 \end{bmatrix}$$


37
36
35
34
33
32
31
0
5
10
15
20
25
t


0
-50
-100
-150
-200
-250
-300
Magnitude [dB]
1. × 10 - 1
1. × 10 1
1. × 10 3
1. × 10 5
1. × 10 7
Freq [rad/s]



Aufgabe 3.6 - GSM (Matlab)

```
> with(CodeGeneration):
  Matlab(fM(1), resultname=fM1):
  Matlab(fM(2), resultname=fM2):
  Matlab(fM(3), resultname=fM3):
  Matlab(fM(4), resultname=fM4):
  map(Matlab, params):
  Matlab(eval(xMruhe, {um=umRuhe, dm=dmRuhe}), resultname=
  xMruheMatlab);
  Matlab(eval(xRedRuhe, {um=umRuhe, dm=dmRuhe}), resultname=
  xRedRuheMatlab);
fM1 = 0.1e1 / LGSM * (-RGSM * iGSM - kGSM * wGSM + um);
fM2 = wGSM - wP;
fM3 = (iGSM * kGSM - dvGSM * wGSM - dcGSM - (wGSM - wP) *
dGSMP - phiGSMP * cGSMP) / JGSM;
fM4 = ((wGSM - wP) * dGSMP + phiGSMP * cGSMP - dqP * wP ^ 2
- dvP * wP - dm - dcP) / JP;
cg = JGSM == 0.1240000000e-1;
cg0 = JP == 0.3250000000e-1;
cg1 = LGSM == 0.1400000000e-2;
cg2 = RGSM == 0.46e0;
cg3 = cGSMP == 0.6822e0;
cg4 = dGSMP == 0.1e1 / 0.100000e6;
cg5 = dcGSM == 0.152e0;
```

```
cg6 = dcP == 0.169e0;
cg7 = dm == 0;
cg8 = dqP == 0.1e1 / 0.10000e5;
cg9 = dvGSM == 0.1800000000e-2;
cg10 = dvP == 0.2700000000e-2;
cg11 = kGSM == 0.1e0;
cg12 = um == 0.56e1;
xMruheMatlab = [-(kGSM / RGSM / dqP * (-(RGSM * dvGSM) -
(RGSM * dvP) - (kGSM ^ 2) + sqrt((-4 * RGSM ^ 2 * dcGSM *
dqP - 4 * RGSM ^ 2 * dcP * dqP - 4 * RGSM ^ 2 * dmRuhe * dqP
+ RGSM ^ 2 * dvGSM ^ 2 + 2 * RGSM ^ 2 * dvGSM * dvP + RGSM ^
2 * dvP ^ 2 + 4 * RGSM * dqP * kGSM * umRuhe + 2 * RGSM *
dvGSM * kGSM ^ 2 + 2 * RGSM * dvP * kGSM ^ 2 + kGSM ^ 4))) /
0.2e1 - umRuhe) / RGSM -(dvGSM / RGSM / dqP * (-(RGSM *
dvGSM) - (RGSM * dvP) - (kGSM ^ 2) + sqrt((-4 * RGSM ^ 2 *
dcGSM * dqP - 4 * RGSM ^ 2 * dcP * dqP - 4 * RGSM ^ 2 *
dmRuhe * dqP + RGSM ^ 2 * dvGSM ^ 2 + 2 * RGSM ^ 2 * dvGSM *
dvP + RGSM ^ 2 * dvP ^ 2 + 4 * RGSM * dqP * kGSM * umRuhe +
2 * RGSM * dvGSM * kGSM ^ 2 + 2 * RGSM * dvP * kGSM ^ 2 +
kGSM ^ 4))) / 0.2e1 + (kGSM / RGSM / dqP * (-(RGSM * dvGSM)
- (RGSM * dvP) - (kGSM ^ 2) + sqrt((-4 * RGSM ^ 2 * dcGSM *
dqP - 4 * RGSM ^ 2 * dcP * dqP - 4 * RGSM ^ 2 * dmRuhe * dqP
+ RGSM ^ 2 * dvGSM ^ 2 + 2 * RGSM ^ 2 * dvGSM * dvP + RGSM ^
2 * dvP ^ 2 + 4 * RGSM * dqP * kGSM * umRuhe + 2 * RGSM *
dvGSM * kGSM ^ 2 + 2 * RGSM * dvP * kGSM ^ 2 + kGSM ^ 4))) /
0.2e1 - umRuhe) / RGSM * kGSM + dcGSM) / cGSMP 0.1e1 / RGSM
/ dqP * (-(RGSM * dvGSM) - (RGSM * dvP) - (kGSM ^ 2) + sqrt(
(-4 * RGSM ^ 2 * dcGSM * dqP - 4 * RGSM ^ 2 * dcP * dqP - 4
* RGSM ^ 2 * dmRuhe * dqP + RGSM ^ 2 * dvGSM ^ 2 + 2 * RGSM
^ 2 * dvGSM * dvP + RGSM ^ 2 * dvP ^ 2 + 4 * RGSM * dqP *
kGSM * umRuhe + 2 * RGSM * dvGSM * kGSM ^ 2 + 2 * RGSM * dvP
* kGSM ^ 2 + kGSM ^ 4))) / 0.2e1 0.1e1 / RGSM / dqP * (-
(RGSM * dvGSM) - (RGSM * dvP) - (kGSM ^ 2) + sqrt((-4 * RGSM
^ 2 * dcGSM * dqP - 4 * RGSM ^ 2 * dcP * dqP - 4 * RGSM ^ 2
* dmRuhe * dqP + RGSM ^ 2 * dvGSM ^ 2 + 2 * RGSM ^ 2 * dvGSM
* dvP + RGSM ^ 2 * dvP ^ 2 + 4 * RGSM * dqP * kGSM * umRuhe
+ 2 * RGSM * dvGSM * kGSM ^ 2 + 2 * RGSM * dvP * kGSM ^ 2 +
kGSM ^ 4))) / 0.2e1];
xRedRuheMatlab = [-(dvGSM / RGSM / dqP * (-(RGSM * dvGSM) -
(RGSM * dvP) - (kGSM ^ 2) + sqrt((-4 * RGSM ^ 2 * dcGSM *
dqP - 4 * RGSM ^ 2 * dcP * dqP - 4 * RGSM ^ 2 * dmRuhe * dqP
+ RGSM ^ 2 * dvGSM ^ 2 + 2 * RGSM ^ 2 * dvGSM * dvP + RGSM ^
2 * dvP ^ 2 + 4 * RGSM * dqP * kGSM * umRuhe + 2 * RGSM *
dvGSM * kGSM ^ 2 + 2 * RGSM * dvP * kGSM ^ 2 + kGSM ^ 4))) /
0.2e1 + (kGSM / RGSM / dqP * (-(RGSM * dvGSM) - (RGSM * dvP)
- (kGSM ^ 2) + sqrt((-4 * RGSM ^ 2 * dcGSM * dqP - 4 * RGSM
^ 2 * dcP * dqP - 4 * RGSM ^ 2 * dmRuhe * dqP + RGSM ^ 2 *
dvGSM ^ 2 + 2 * RGSM ^ 2 * dvGSM * dvP + RGSM ^ 2 * dvP ^ 2
+ 4 * RGSM * dqP * kGSM * umRuhe + 2 * RGSM * dvGSM * kGSM ^
2 + 2 * RGSM * dvP * kGSM ^ 2 + kGSM ^ 4))) / 0.2e1 -
umRuhe) / RGSM * kGSM + dcGSM) / cGSMP 0.1e1 / RGSM / dqP *
(-(RGSM * dvGSM) - (RGSM * dvP) - (kGSM ^ 2) + sqrt((-4 *
RGSM ^ 2 * dcGSM * dqP - 4 * RGSM ^ 2 * dcP * dqP - 4 * RGSM
^ 2 * dmRuhe * dqP + RGSM ^ 2 * dvGSM ^ 2 + 2 * RGSM ^ 2 *
dvGSM * dvP + RGSM ^ 2 * dvP ^ 2 + 4 * RGSM * dqP * kGSM *
umRuhe + 2 * RGSM * dvGSM * kGSM ^ 2 + 2 * RGSM * dvP * kGSM
^ 2 + kGSM ^ 4))) / 0.2e1 0.1e1 / RGSM / dqP * (-(RGSM *
dvGSM) - (RGSM * dvP) - (kGSM ^ 2) + sqrt((-4 * RGSM ^ 2 *
dcGSM * dqP - 4 * RGSM ^ 2 * dcP * dqP - 4 * RGSM ^ 2 *
dmRuhe * dqP + RGSM ^ 2 * dvGSM ^ 2 + 2 * RGSM ^ 2 * dvGSM *
dvP + RGSM ^ 2 * dvP ^ 2 + 4 * RGSM * dqP * kGSM * umRuhe +
```

```
2 * RGSM * dvGSM * kGSM ^ 2 + 2 * RGSM * dvP * kGSM ^ 2 +
kGSM ^ 4))) / 0.2e1];
> Matlab(eval(AA, {um=umRuhe, dm=dmRuhe}), resultname=Asys);
  Matlab(eval(BB, {um=umRuhe, dm=dmRuhe}), resultname=Bsys);
  Matlab(eval(CC, {um=umRuhe, dm=dmRuhe}), resultname=Csys);
  Matlab(eval(DD, {um=umRuhe, dm=dmRuhe}), resultname=Dsys);

  Matlab(eval(AAred, {um=umRuhe, dm=dmRuhe}), resultname=
  AredSys);
  Matlab(eval(BBred, {um=umRuhe, dm=dmRuhe}), resultname=
  BredSys);
  Matlab(eval(CCred, {um=umRuhe, dm=dmRuhe}), resultname=
  CredSys);
  Matlab(eval(DDred, {um=umRuhe, dm=dmRuhe}), resultname=
  DredSys);
Asys = [-0.1e1 / LGSM * RGSM 0 -0.1e1 / LGSM * kGSM 0; 0 0 1
-1; kGSM / JGSM -cGSMP / JGSM -(dvGSM + dGSMP) / JGSM dGSMP
/ JGSM; 0 cGSMP / JP dGSMP / JP -((dGSMP * RGSM) - (RGSM *
dvGSM) - (kGSM ^ 2) + sqrt((-4 * RGSM ^ 2 * dcGSM * dqP - 4
* RGSM ^ 2 * dcP * dqP - 4 * RGSM ^ 2 * dmRuhe * dqP + RGSM
^ 2 * dvGSM ^ 2 + 2 * RGSM ^ 2 * dvGSM * dvP + RGSM ^ 2 *
dvP ^ 2 + 4 * RGSM * dqP * kGSM * umRuhe + 2 * RGSM * dvGSM
* kGSM ^ 2 + 2 * RGSM * dvP * kGSM ^ 2 + kGSM ^ 4))) / RGSM
/ JP;];
Bsys = [0.1e1 / LGSM 0; 0 0; 0 0; 0 -0.1e1 / JP;];
Csys = [0 0 0 1;];
Dsys = [0 0;];
AredSys = [0 1 -1; -cGSMP / JGSM -((dGSMP * RGSM) + (RGSM *
dvGSM) + (kGSM ^ 2)) / RGSM / JGSM dGSMP / JGSM; cGSMP / JP
dGSMP / JP -((dGSMP * RGSM) - (RGSM * dvGSM) - (kGSM ^ 2) +
sqrt((-4 * RGSM ^ 2 * dcGSM * dqP - 4 * RGSM ^ 2 * dcP * dqP
- 4 * RGSM ^ 2 * dmRuhe * dqP + RGSM ^ 2 * dvGSM ^ 2 + 2 *
RGSM ^ 2 * dvGSM * dvP + RGSM ^ 2 * dvP ^ 2 + 4 * RGSM * dqP
* kGSM * umRuhe + 2 * RGSM * dvGSM * kGSM ^ 2 + 2 * RGSM *
dvP * kGSM ^ 2 + kGSM ^ 4))) / RGSM / JP;];
BredSys = [0 0; kGSM / RGSM / JGSM 0; 0 -0.1e1 / JP;];
CredSys = [0 0 1;];
DredSys = [0 0;];
> Matlab(eval(hhRuheRaw, {um=umRuhe, dm=dmRuhe}), resultname=
  hhRuheMatlab);
  Matlab(eval(hhRedRuheRaw, {um=umRuhe, dm=dmRuhe}),
  resultname=hhRedRuheMatlab);
hhRuheMatlab = [0.1e1 / RGSM / dqP * (-(RGSM * dvGSM) -
(RGSM * dvP) - (kGSM ^ 2) + sqrt((-4 * RGSM ^ 2 * dcGSM *
dqP - 4 * RGSM ^ 2 * dcP * dqP - 4 * RGSM ^ 2 * dmRuhe * dqP
+ RGSM ^ 2 * dvGSM ^ 2 + 2 * RGSM ^ 2 * dvGSM * dvP + RGSM ^
2 * dvP ^ 2 + 4 * RGSM * dqP * kGSM * umRuhe + 2 * RGSM *
dvGSM * kGSM ^ 2 + 2 * RGSM * dvP * kGSM ^ 2 + kGSM ^ 4))) /
0.2e1];
hhRedRuheMatlab = [0.1e1 / RGSM / dqP * (-(RGSM * dvGSM) -
(RGSM * dvP) - (kGSM ^ 2) + sqrt((-4 * RGSM ^ 2 * dcGSM *
dqP - 4 * RGSM ^ 2 * dcP * dqP - 4 * RGSM ^ 2 * dmRuhe * dqP
+ RGSM ^ 2 * dvGSM ^ 2 + 2 * RGSM ^ 2 * dvGSM * dvP + RGSM ^
2 * dvP ^ 2 + 4 * RGSM * dqP * kGSM * umRuhe + 2 * RGSM *
dvGSM * kGSM ^ 2 + 2 * RGSM * dvP * kGSM ^ 2 + kGSM ^ 4))) /
0.2e1];
```