

Versuch III: LQ-Regelung und Animation

Andreas Jentsch, Ali Kerem Sacakli

Praktikumsbericht – Praktikum Matlab/Simulink II



TECHNISCHE
UNIVERSITÄT
DARMSTADT

REGELUNGSTECHNIK *rtm*
UND MECHATRONIK

3.1 LQ-Regelung Funktion

Es soll eine Funktion erstellt werden die über die Eingaben von A, B, Q, R die Reglermatrix K und die Pole des geschlossenen Kreises zurückgibt.

Listing 3.1: Funktion für den LQR-Entwurf

```
1 function [ K, poleRK] = berechneLQR( A, B, Q, R )

    %Fehlerabfragen
    K      = 'Error';
    poleRK = []

6
    %Steuerbarkeit
    if length(A) == rank(ctrb(A,B))
        disp('System_list_steuerbar');
    else
11        disp('System_list_NICHT_steuerbar');
    end

    %Test auf Symmetrie von Q:

16 if all(all(Q == Q'))
        disp('Matrix_Q_list_symmetrisch');
    else
        disp('Matrix_Q_list_NICHT_symmetrisch');
    end

21
    %Test auf Symmetrie von R:

    if all(all(R == R'))
        disp('Matrix_R_list_symmetrisch');
    else
26        disp('Matrix_R_list_NICHT_symmetrisch');
    end

    %Test auf positive Definitheit von Q:

31
    if all(real(eig(Q)) > 0)
        disp('Matrix_Q_list_positiv_definit');
    else
```

```
        disp('Matrix_Q_list_NICHT_positiv_definit');
36 end

%Test auf positive Definitheit von R:
if all(real(eig(R)) > 0)
41     disp('Matrix_R_list_positiv_definit');
else
    disp('Matrix_R_list_NICHT_positiv_definit');
end

46 %Reglerberechnung:
[K, ~, poleRK] = lqr(A,B,Q,R,zeros(size(B,2)));
    %poleRK: Pole geschlossener Regelkreis

51 end %function berechneLQR
```

3.2 Simulink-Implementierung der Regelung

Es ist ein Datensatz des Doppelpendels und eine S-Function, die das nichtlineare Modell beschreibt, vorgegeben. Darauf aufbauend soll eine Regelung in Simulink implementiert werden.

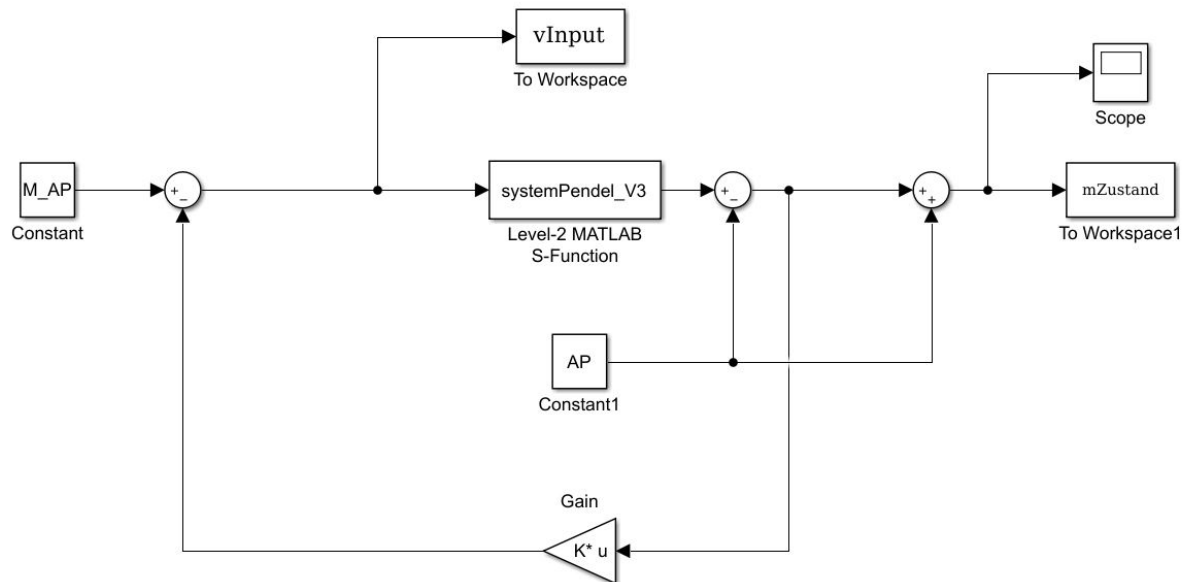


Figure 3.1: Implementierung mit Fcn-Blöcken

3.3 Funktion runPendel

Die Funktion runPendel soll über die Eingabe der Pendel­daten, der Reglermatrix und der Anfangswerte des Systems den Verlauf der Zustandsgrößen mit dem dazugehörigen Zeitvektor zurückgeben.

Listing 3.2: Funktion runPendel

```
function [ vT, mX, u ] = runPendel( stPendel, AP, K, x0, M_AP )  
2  
vT = 'error';  
mX = 'error';  
  
Tend = 10;  
7 stOptions = simset( 'SrcWorkspace', 'current' );  
sim('Modell_V3', Tend, stOptions);  
  
vT = mZustand.Time;  
mX = mZustand.Data;  
12 u.Data = vInput.Data;  
u.Time = vInput.Time;  
end
```

3.5 Animation des Pendels

Für die Animation des Pendels soll eine Funktion `animierePendel` implementiert und anschließend mit Simulationsergebnissen getestet werden. Zudem soll der Code so erweitert werden, dass er optional ein Video der Animation generiert.

Der Code der Funktion sieht wie folgt aus:

Listing 3.3: Funktion `animierePendel`

```
function [ ] = animierePendel( vT, mX, stPendel, hAxes, varargin )

MakeAvi = false;
if nargin == 5
5     MakeAvi = varargin{1};
end

Tpause = 1/25;

10
vTAnim = 0:Tpause:vT(end);
mXAnim = interp1(vT,mX,vTAnim);

nBilder = length(mXAnim);

15
if isempty(hAxes)
    figure;
    hAxes = axes();
end
20 axis([-2 2 -2 2])

hold on;
for i = 1:nBilder
    if i>1
25        delete(hPendel);
    end
    P1 = [sin(mXAnim(i,1)), -cos(mXAnim(i,1))];
    P2 = P1 + [sin(mXAnim(i,3)), -cos(mXAnim(i,3))];
    hPendel = plot(hAxes, [0, P1(1)], [0, P1(2)], 'b', ...
30        [P1(1), P2(1)], [P1(2), P2(2)], 'r');
    title(num2str(vTAnim(i)));
end
```

```
    if MakeAvi
        if i == 1
35            vFrames = getframe(hAxes);
            end
            vFrames(end+1) = getframe(hAxes);
        end

40    pause(Tpause);
    end
    hold off;

    if MakeAvi
45        v = VideoWriter('animation.avi','Uncompressed_AVI');
        open(v)
        writeVideo(v,vFrames);
    end
    end
```

3.6 Simulation verschiedener LQ Regler

Im folgenden Abschnitt werden unterschiedlich ausgelegte LQ Regler an verschiedenen Arbeitspunkten unter verschiedenen Startbedingungen getestet. Die Arbeitspunkte und Anfangsbedingungen sind:

$$\begin{aligned} \mathbf{x}_{AP1} &= \begin{bmatrix} \pi/3 & 0 & 0 & 0 \end{bmatrix} & \mathbf{x}_{01} &= \begin{bmatrix} -\pi/3 & 0 & 0 & 0 \end{bmatrix} \\ \mathbf{x}_{AP2} &= \begin{bmatrix} \pi/3 & 0 & \pi & 0 \end{bmatrix} & \mathbf{x}_{02} &= \begin{bmatrix} -\pi/3 & 0 & \pi & 0 \end{bmatrix} \end{aligned}$$

Die Gewichtungsmatrizen für die unterschiedlichen Regler sind:

$$\begin{aligned} \mathbf{Q}_1 &= \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \mathbf{R}_1 &= 1 \\ \mathbf{Q}_2 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \mathbf{R}_2 &= 1 \\ \mathbf{Q}_3 &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} & \mathbf{R}_3 &= 10 \end{aligned}$$

In den nachfolgenden Grafiken kennzeichnet die gestrichelte Linie den Wert des Arbeitspunktes für die entsprechende Größe. In Figure 3.2 ist gut zu erkennen, dass die Gewichtungsmatrix \mathbf{Q}_1 Winkelabweichungen in φ_1 stark bestraft. Aus diesem Grund wird dieser Winkel schnell ausgeregelt. Die Stellgröße M hat hier allerdings anfänglich einen sehr hohen Wert, da dies in der Gewichtungsmatrix \mathbf{R} nicht stark bestraft wird.

In Figure 3.3 ist zu erkennen, dass der Winkel φ_2 sehr schnell ausgeregelt wird. Dies ist auf die hohe Bestrafung einer Winkelabweichung in φ_2 in der Gewichtungsmatrix \mathbf{Q}_2 zurückzuführen. Der Winkel φ_1 wird im Vergleich zu Figure 3.2 langsamer ausgeregelt, da Abweichungen hier nicht so stark gewichtet werden.

Verglichen mit den Vorangegangenen Simulationen ist in Figure 3.4 der maximale Betrag der Stellgröße M deutlich geringer. Dies ist auf die hohe bestrafung der Stellgröße in der Gewichtungsmatrix \mathbf{R}_3 zurückzuführen. Die Winkel φ_1 und φ_2 verhalten sich ähnlich wie in Figure 3.2, sie werden allerdings langsamer ausgeregelt.

Das qualitative Verhalten bezüglich ausregelzeit und betragsmaxima ist am Arbeitspunkt \mathbf{x}_{AP2} mit den Startwerten \mathbf{x}_{02} ist analog zu den mit den entsprechenden Gewichtungsmatrizen ausgelegten Reglern am Arbeitspunkt \mathbf{x}_{AP1} mit den Startwerten \mathbf{x}_{01} . Die Simulationsergebnisse zum

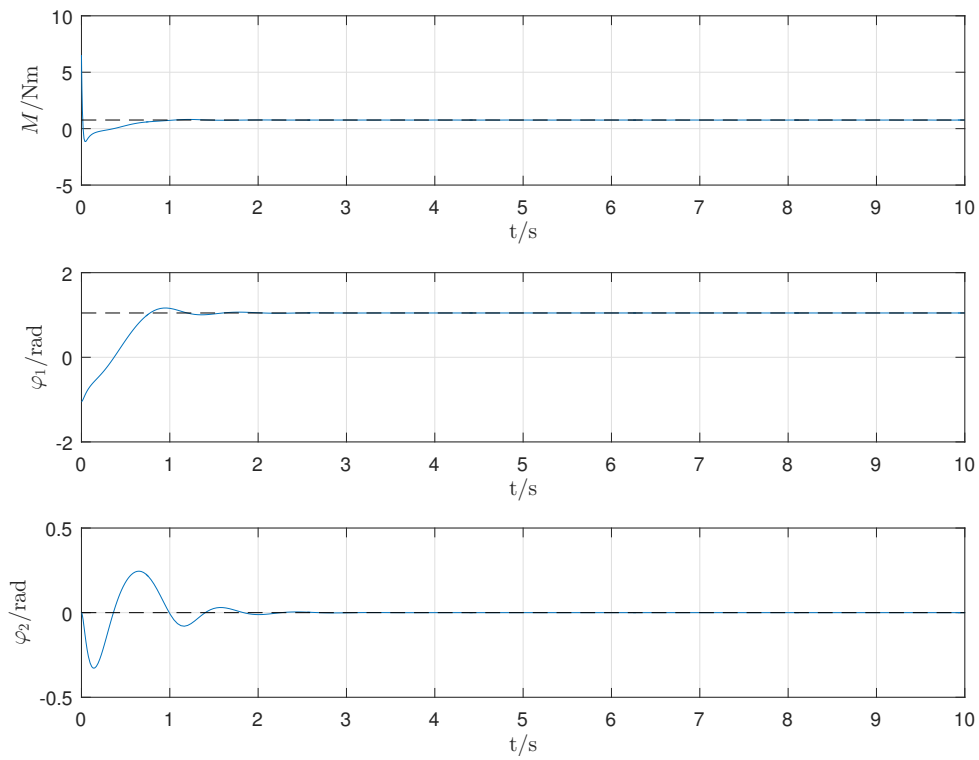


Figure 3.2: Simulationsergebnisse für $x_{AP1}, x_{01}, Q_1, R_1$

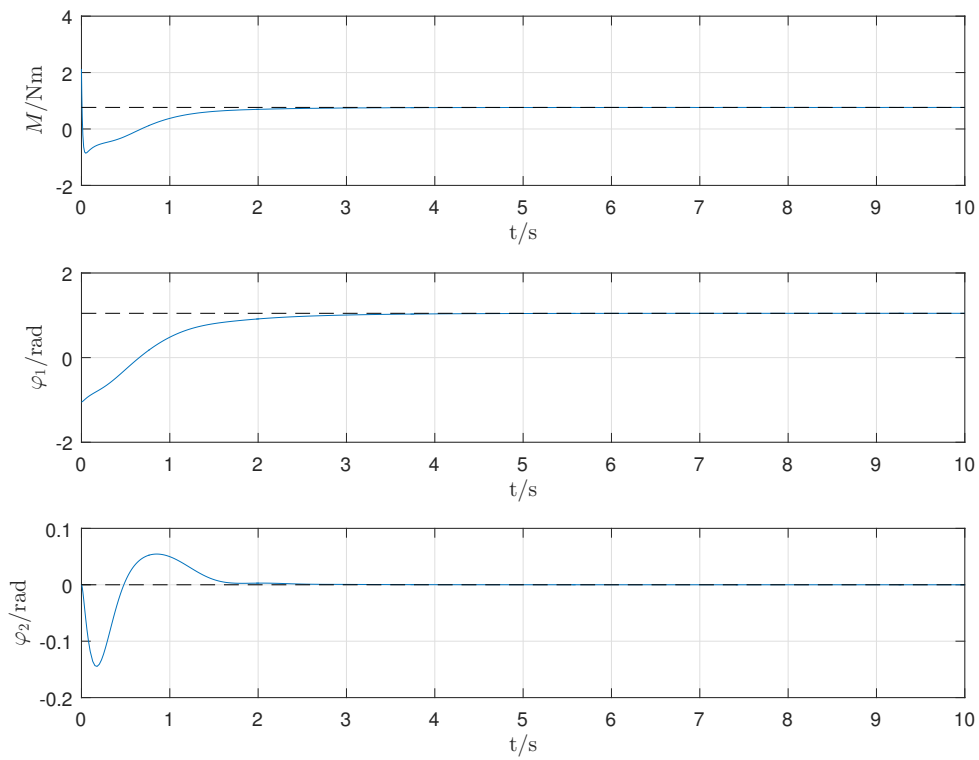


Figure 3.3: Simulationsergebnisse für $x_{AP1}, x_{01}, Q_2, R_2$

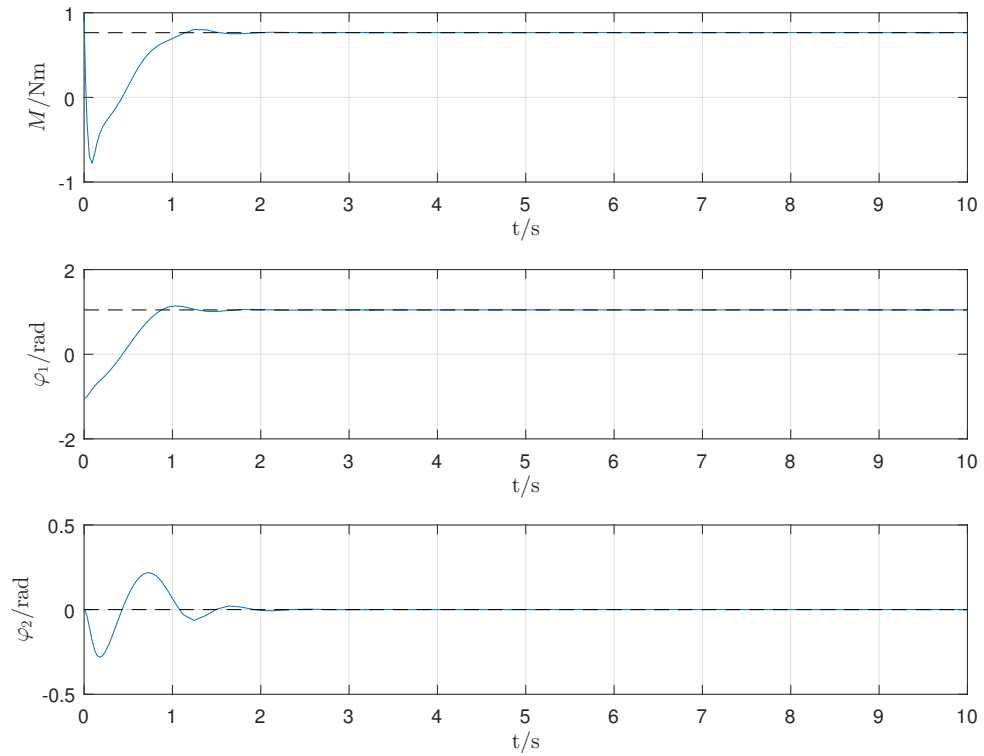


Figure 3.4: Simulationsergebnisse für \mathbf{x}_{AP1} , \mathbf{x}_{01} , \mathbf{Q}_3 , \mathbf{R}_3

zweiten Arbeitspunkt sind in Figure 3.5 bis 3.7 abgebildet. Generell werden am ersten Arbeitspunkt alle Größen schneller ausgeregelt. Dies ist plausibel, da es leichter ist ein hängendes Pendel zu stabilisieren, als ein inverses Pendel auszuregeln.

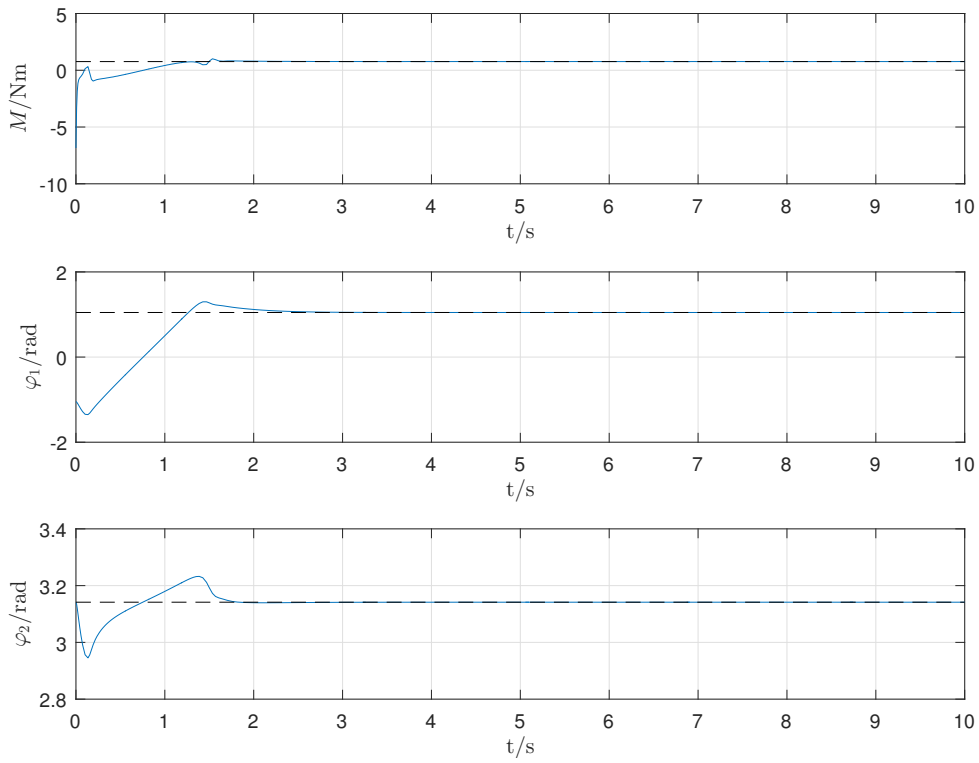


Figure 3.5: Simulationsergebnisse für $\mathbf{x}_{AP2}, \mathbf{x}_{02}, \mathbf{Q}_1, \mathbf{R}_1$

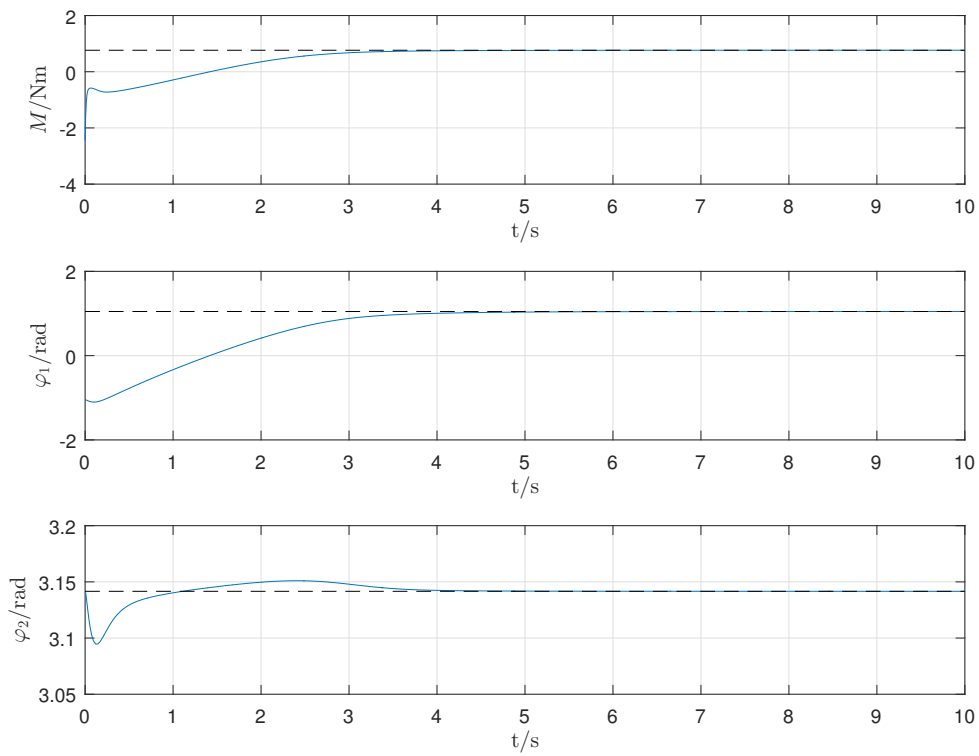


Figure 3.6: Simulationsergebnisse für $\mathbf{x}_{AP2}, \mathbf{x}_{02}, \mathbf{Q}_2, \mathbf{R}_2$

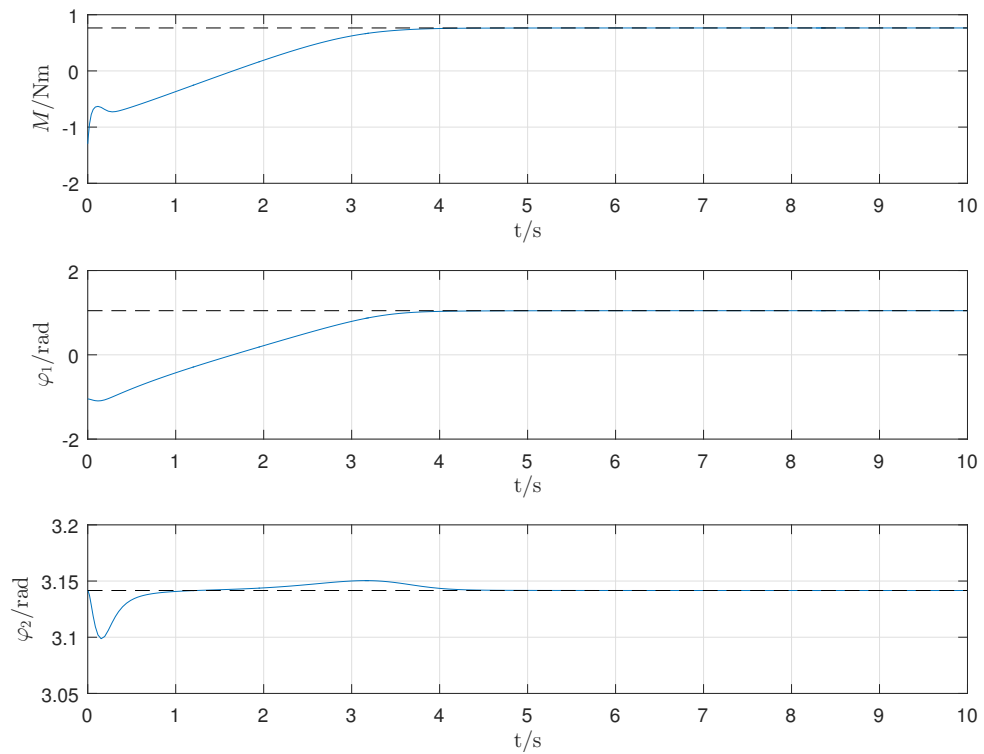


Figure 3.7: Simulationsergebnisse für \mathbf{x}_{AP2} , \mathbf{x}_{02} , \mathbf{Q}_3 , \mathbf{R}_3

3.7 Numerische Probleme

Werden zu große Werte für die Gewichtungsmatrizen gewählt, verlassen die Zustände des linearen Modells das Einzugsgebiet. Eine das nichtlineare Modell wird somit instabil. Die Simulationsergebnisse divergieren ins Unendliche.