# Versuch III: LQ-Regelung und Animation

**Andreas Jentsch, Ali Kerem Sacakli**

Praktikumsbericht – Praktikum Matlab/Simulink II

TECHNISCHE UNIVERSITÄT DARMSTADT

REGELUNGSTECHNIK UND MECHATRONIK rtm

## 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

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

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

%Steuerbarkeit
if length(A) == rank(ctrb(A,B))
    disp('System ist steuerbar');
else
    disp('System ist NICHT steuerbar');
end

%Test auf Symmetrie von Q:

if all(all(Q == Q'))
    disp('Matrix Q ist symmetrisch');
else
    disp('Matrix Q ist NICHT symmetrisch');
end

%Test auf Symmetrie von R:

if all(all(R == R'))
    disp('Matrix R ist symmetrisch');
else
    disp('Matrix R ist NICHT symmetrisch');
end

%Test auf positive Definitheit von Q:

if all(real(eig(Q)) > 0)
    disp('Matrix Q ist positiv definit');
else
```

```
    disp('Matrix Q ist NICHT positiv definit');
end


%Test auf positive Definitheit von R:
if all(real(eig(R)) > 0)
    disp('Matrix R ist positiv definit');
else
    disp('Matrix R ist NICHT positiv definit');
end


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

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.

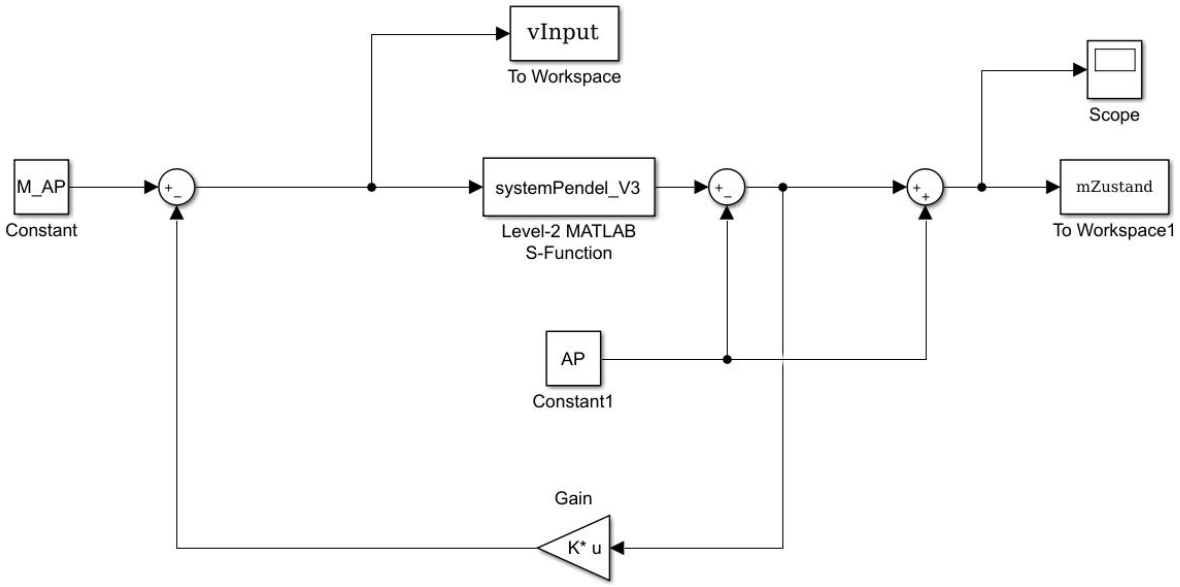


**Figure 3.1:** Implementierung mit Fcn-Blöcken

## 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 )

vT = 'error';
mX = 'error';

Tend = 10;
stOptions = simset( 'SrcWorkspace', 'current');
sim('Modell_V3', Tend, stOptions);

vT = mZustand.Time;
mX = mZustand.Data;
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
    MakeAvi = varargin{1};
end

Tpause = 1/25;


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

nBilder = length(mXAnim);

if isempty(hAxes)
    figure;
    hAxes = axes();
end
axis([-2 2 -2 2]*0.2)

hold on;
for i = 1:nBilder
    if i>1
        delete(hPendel);
    end
    P1 = [sin(mXAnim(i,1)), -cos(mXAnim(i,1))]*0.2;
    P2 = P1 + [sin(mXAnim(i,3)), -cos(mXAnim(i,3))]*0.2;
    hPendel = plot(hAxes, [0, P1(1)],[0, P1(2)],'b',...
                    [P1(1), P2(1)], [P1(2), P2(2)],'r');
    title(['t = ',num2str(vTAnim(i)),' sec']);
    xlabel('x[m]');
```

```
    ylabel('y[m]');

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

    pause(Tpause);
end
hold off;

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

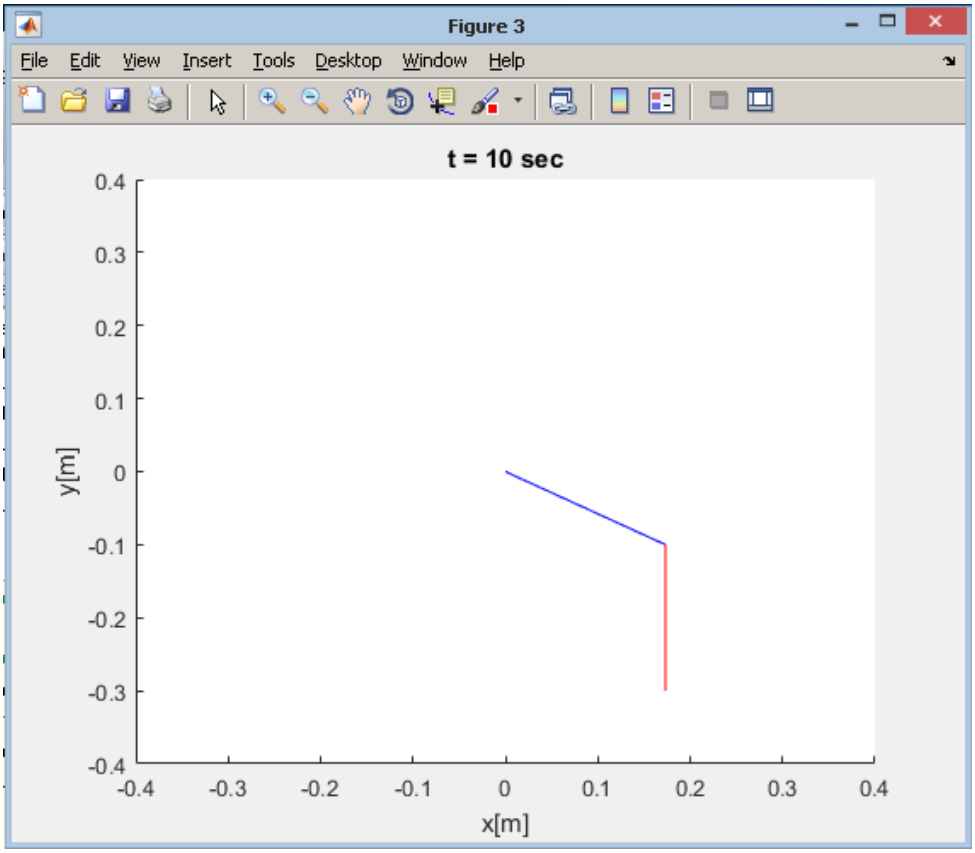


**Figure 3.2:** Bild der Animation

## 3.6 Simulation verschiedener LQ Regler

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

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

Die Gewichtungsmatrizen für die unterschiedlichen Regler sind:

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

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

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

In den nachfolgenden Grafiken kennzeichnet die gestrichelte Linie den Wert des Arbeitspunktes für die entsprechende Größe. In Figure 3.3 ist gut zu erkennen, dass die Gewichtungsmatrix $\mathbf{Q}_1$ Winkelabweichungen in $\varphi_1$ stark bestraft. Aus diesem Grund wird dieser Winkel schnell ausgeregelt. Die Stellgröße $M$ hat hier allerdings anfänglich einen sehr hohen Wert, da diese in der Gewichtungsmatrix $\mathbf{R}$ nicht stark bestraft wird.

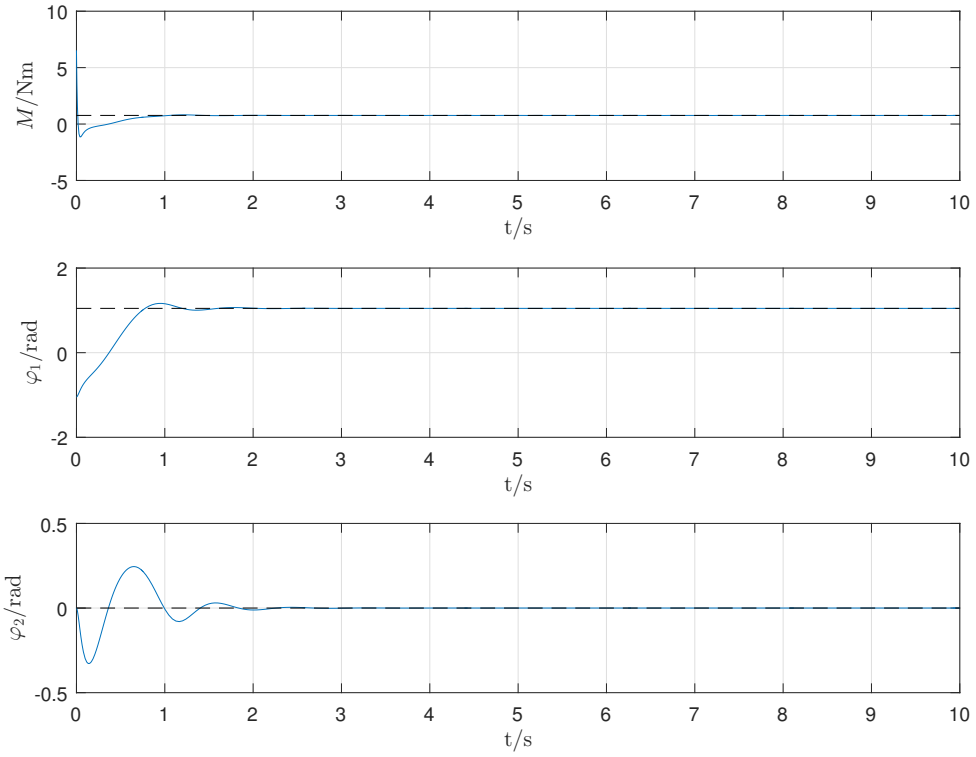


**Figure 3.3:** Simulationsergebnisse für $\mathbf{x}_{\mathrm{AP1}}, \mathbf{x}_{01}, \mathbf{Q}_1, \mathbf{R}_1$

In Figure 3.4 ist zu erkennen, dass der Winkel $\varphi_2$ sehr schnell ausgeregelt wird. Dies ist auf die hohe Bestrafung einer Winkelabweichung in $\varphi_2$ in der Gewichtungsmatrix $\mathbf{Q}_2$ zurückzuführen.

Der Winkel $\varphi_1$ wird im Vergleich zu Figure 3.3 langsamer ausgeregelt, da Abweichungen hier nicht so stark gewichtet werden.

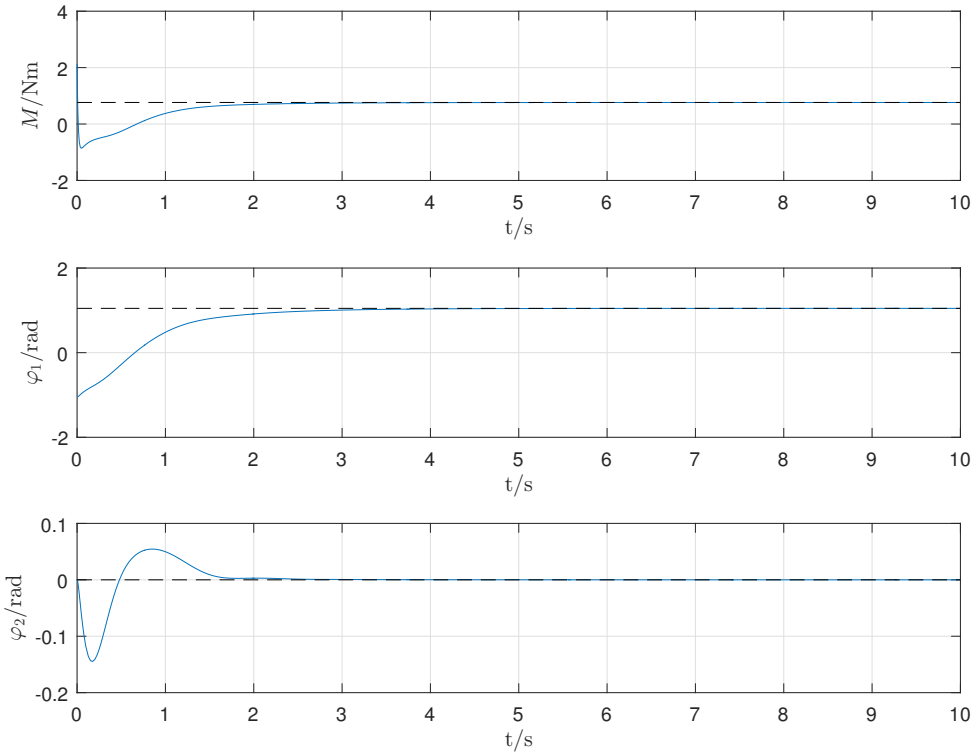


**Figure 3.4:** Simulationsergebnisse für $\mathbf{x}_{\mathrm{AP1}}, \mathbf{x}_{01}, \mathbf{Q}_2, \mathbf{R}_2$

Verglichen mit den vorangegangenen Simulationen ist in Figure 3.5 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 $\varphi_1$ und $\varphi_2$ verhalten sich ähnlich wie in Figure 3.3, sie werden allerdings langsamer ausgeregelt.

Das qualitative Verhalten bezüglich Ausregelzeit und Betragsmaxima ist am Arbeitpunkt $\mathbf{x}_{\mathrm{AP2}}$ mit den Startwerten $\mathbf{x}_{02}$ analog zu den mit den entsprechenden Gewichtungsmatrizen ausgelegten Reglern am Arbeitspunkt $\mathbf{x}_{\mathrm{AP1}}$ mit den Startwerten $\mathbf{x}_{01}$. Die Simulationsergebnisse zum zweiten Arbeitspunkt sind in Figure 3.6 bis 3.8 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.

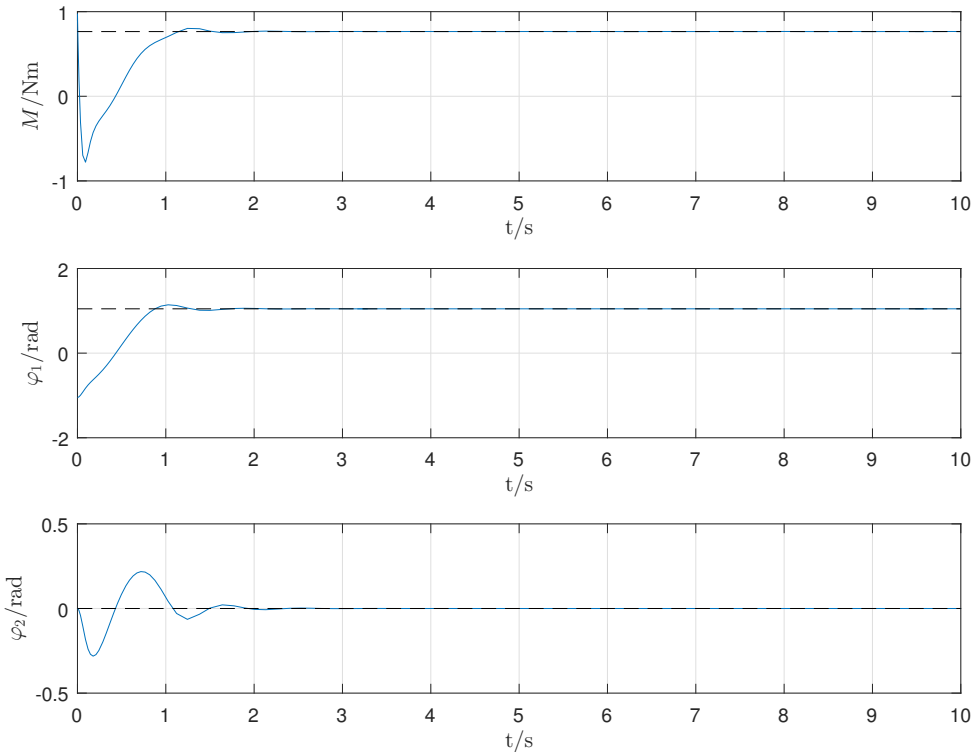


**Figure 3.5:** Simulationsergebnisse für $\mathbf{x}_{\mathrm{AP1}}, \mathbf{x}_{01}, \mathbf{Q}_3, \mathbf{R}_3$

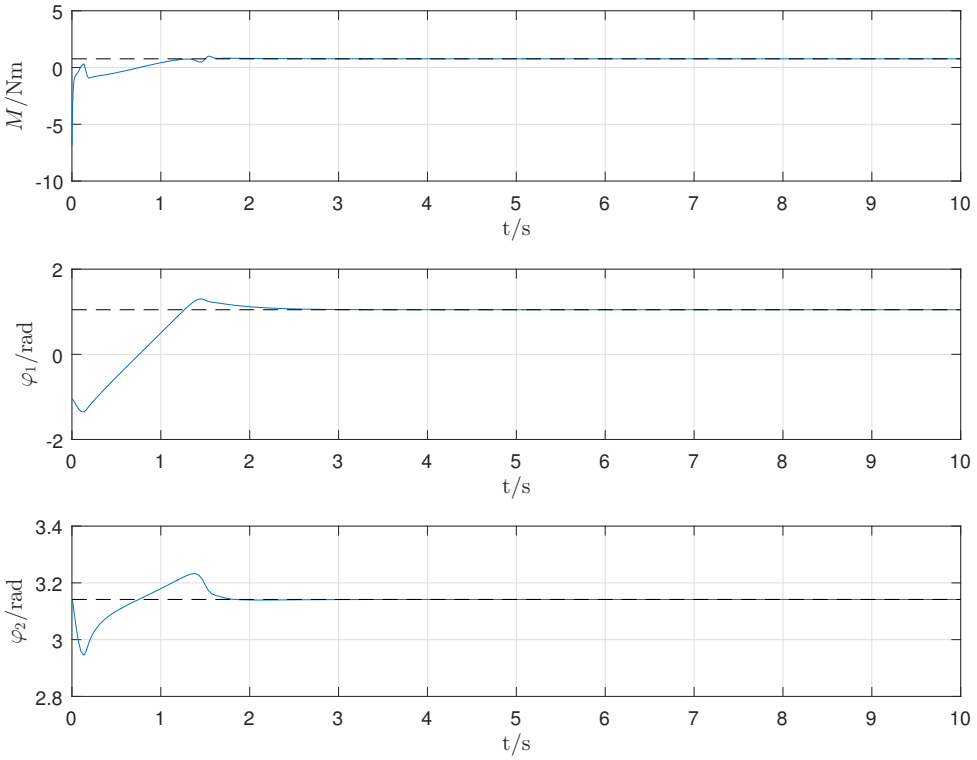


**Figure 3.6:** Simulationsergebnisse für $\mathbf{x}_{\mathrm{AP2}}, \mathbf{x}_{02}, \mathbf{Q}_1, \mathbf{R}_1$

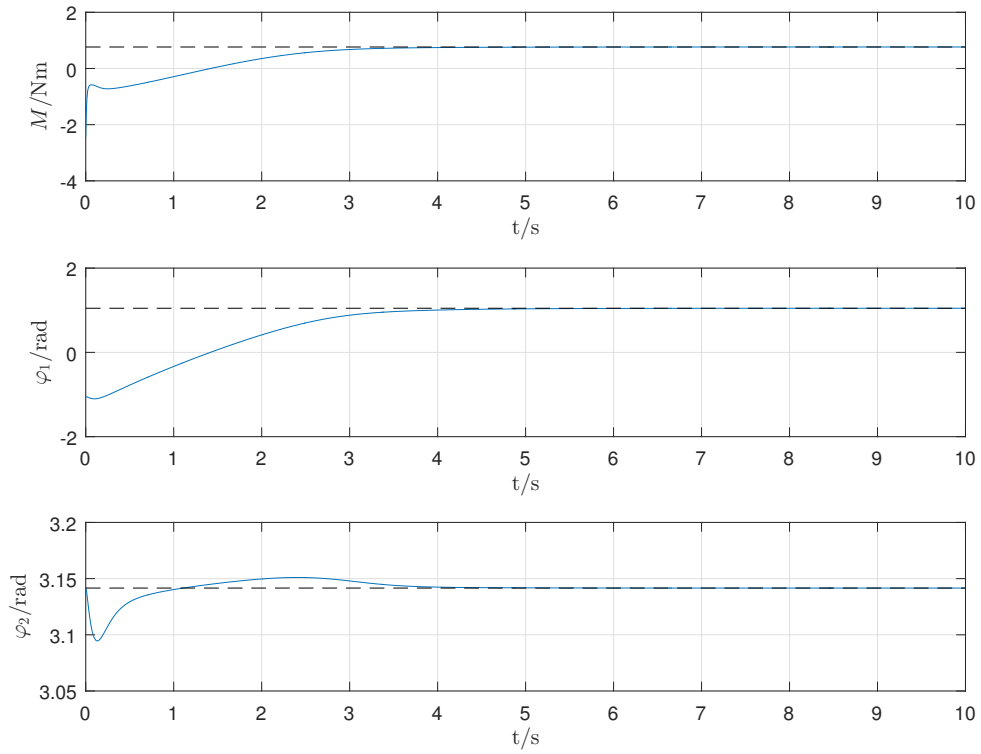


**Figure 3.7:** Simulationsergebnisse für $\mathbf{x}_{\mathrm{AP2}}, \mathbf{x}_{02}, \mathbf{Q}_2, \mathbf{R}_2$

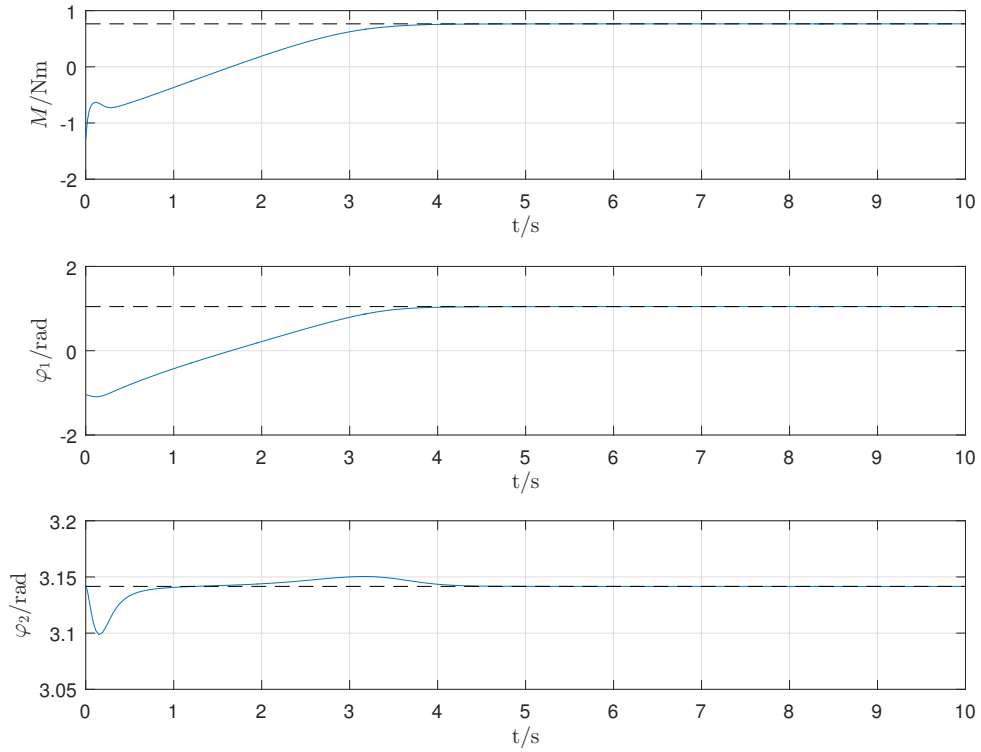


**Figure 3.8:** Simulationsergebnisse für $\mathbf{x}_{\mathrm{AP2}}, \mathbf{x}_{02}, \mathbf{Q}_3, \mathbf{R}_3$

## 3.7 Numerische Probleme

Werden zu große Werte für die Gewichtungsmatrizen gewählt, verlassen die Zustände des linearen Modells das Einzugsgebiet. Das nichtlineare Modell wird somit instabil. Die Simulationswerte divergieren ins Unendliche. Anders ausgedrückt wird das System durch die starke Gewichtung eines Parameters zu steif, sodass der Konvergenzbereich des angewandten Runge-Kutte-Verfahrens zu klein ausfällt. Das Gear-Verfahren, das einen größeren Konvergenzbereich aufweist, könnte hierbei Abhilfe schaffen.