

# Versuch VI:

# Trajektorienfolgeregelung

Andreas Jentsch, Ali Kerem Sacakli

Praktikumsbericht – Praktikum Matlab/Simulink II

11. Juli 2017



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

REGELUNGSTECHNIK **rtm**  
UND MECHATRONIK



---

## 6.1 Linearisierung

---

Ergänzend zum letzten Versuch, soll in diesem Versuch eine Trajektorienfolgeregelung entworfen werden. Hierbei soll nach Linearisierung der Trajektorie und Berechnung des Reglerparameters die Regelung in Simulink modelliert und anschließend simuliert werden.

Folgendes Listing zeigt die Funktion zur Linearisierung der Trajektorie:

**Listing 6.1:** Quellcode der Funktion `linearisierung`

---

```
1 function [ A, B, C, D] = linearisierung_XU( x,u )
   %LINEARISIERUNG Summary of this function goes here
   % Detailed explanation goes here
   syms phi1 phi2 dphi1 dphi2 ddphi1 ddphi2 M;
   [f,h] = nonlinear_model();

6
   z = [phi1;dphi1;phi2;dphi2];
   u_z = M;

   A = jacobian(f,z);
11 B = jacobian(f,u_z);
   C = jacobian(h,z);
   D = jacobian(h,u_z);

   A = subs(A,[z.',u_z],[x.',u]);
16 B = subs(B,[z.',u_z],[x.',u]);
   C = subs(C,[z.',u_z],[x.',u]);
   D = subs(D,[z.',u_z],[x.',u]);

21 A = double(A);
   B = double(B);
   C = double(C);
   D = double(D);

26 end
```

---

---

## 6.2 Berechnen von $K(t)$

---

Folgender Code implementiert die Riccati-DGL.

---

### Listing 6.2: Quellcode der Funktion RiccatiDGL

---

```
function vPdot = RiccatiDGL( t, vP, stTraj, Q, R )
%RICCATIDGL Summary of this function goes here
3 % Detailed explanation goes here

persistent RiccatiCache;

if nargin > 2
8   RiccatiCache.stTraj = stTraj;
   RiccatiCache.Q = Q;
   RiccatiCache.R = R;

elseif isempty(RiccatiCache)
13   disp('RiccatiDGL_was_not_initialized');

else
   stTraj = RiccatiCache.stTraj;
   Q = RiccatiCache.Q;
18   R = RiccatiCache.R;
end

Upol = interp1(stTraj.vT,stTraj.vU,t);
Xpol = interp1(stTraj.vT,stTraj.mX',t)';
23

[A,B,~,~] = linearisierung_XU(Xpol,Upol);

n = size(A,1);
28 P = reshape(vP,n,n);

vPdot = (P*B*inv(R)*B'*P) - (P*A) - (A'*P) - Q;

vPdot = (vPdot(:).')';
33
end
```

---

Folgender Code dient zur Berechnung des Reglerparameters  $K(t)$ .

---

### Listing 6.3: Quellcode der Funktion berechneK

---

```
function [ vTK, mK ] = berechneK( stTraj, Q, R )
```

---

---

*%BERECHNEK Summary of this function goes here*

*% Detailed explanation goes here*

```
5 [A, B, ~, ~] = linearisierung_XU(stTraj.mX(:,end), stTraj.vU(end));
n = length(A);

P_End = care(A, B, Q, R, zeros(size(B)), eye(n));
P_End = (P_End(:).')';

10 Pdot_check = RiccatiDGL(stTraj.vT(end), P_End, stTraj, Q, R);

[vTK,vPt] = ode45(@RiccatiDGL,flip(stTraj.vT), P_End);
vTK = flipud(vTK);
15 vPt = flipud(vPt);

for ii = 1:length(vTK)
%     xh = interp1(stTraj.vT,stTraj.mX',stTraj.vT(ii))';
%     uh = interp1(stTraj.vT,stTraj.vU,stTraj.vT(ii));
20     xh = stTraj.mX(:,ii);
    uh = stTraj.vU(ii);
    [~,Bh,~,~] = linearisierung_XU(xh,uh);

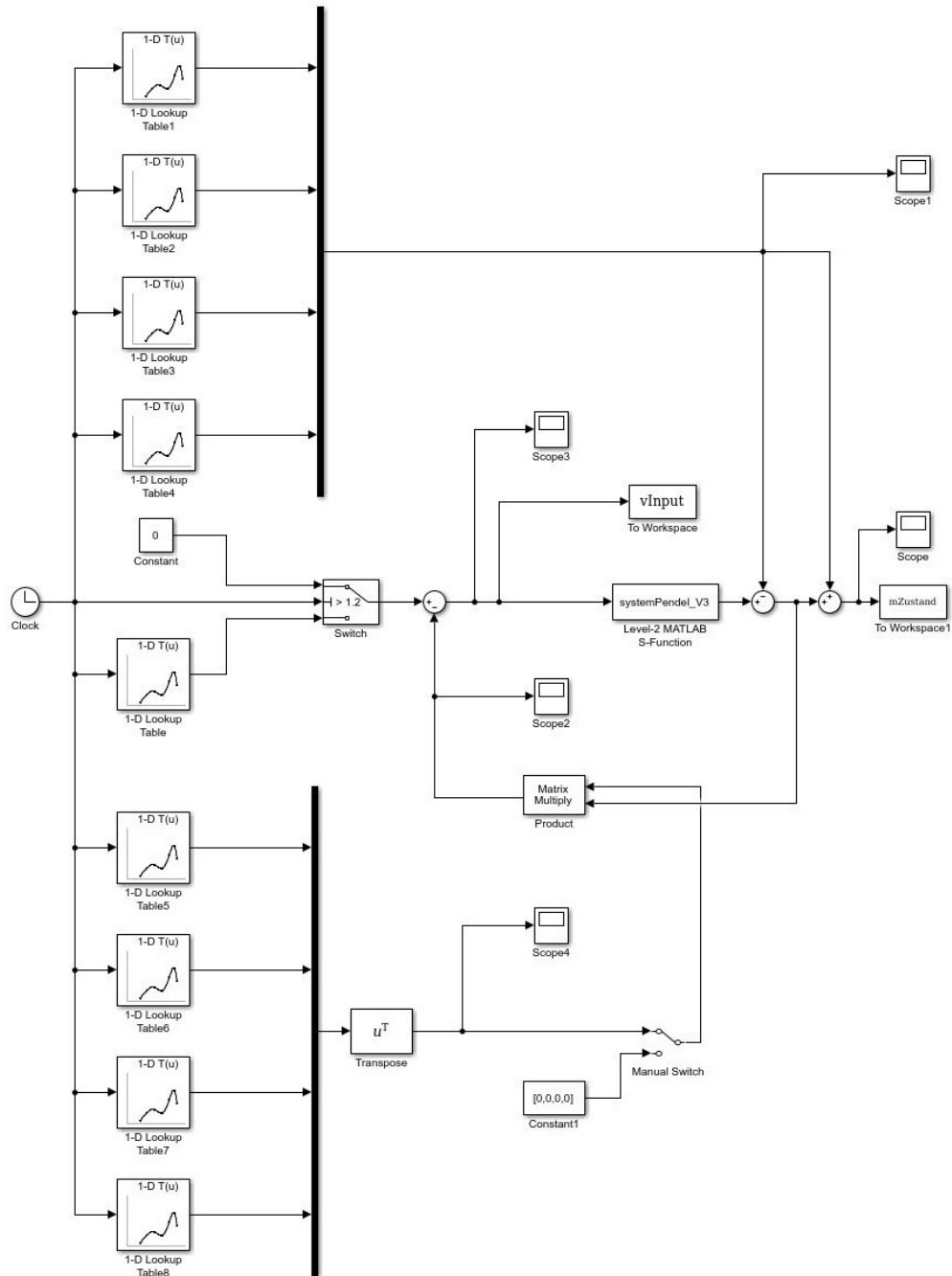
    Ph = vPt(ii,:);
25     Ph = reshape(Ph,n,n);

    mK(1:4,ii) = inv(R)*Bh'*Ph;
end
end
```

---

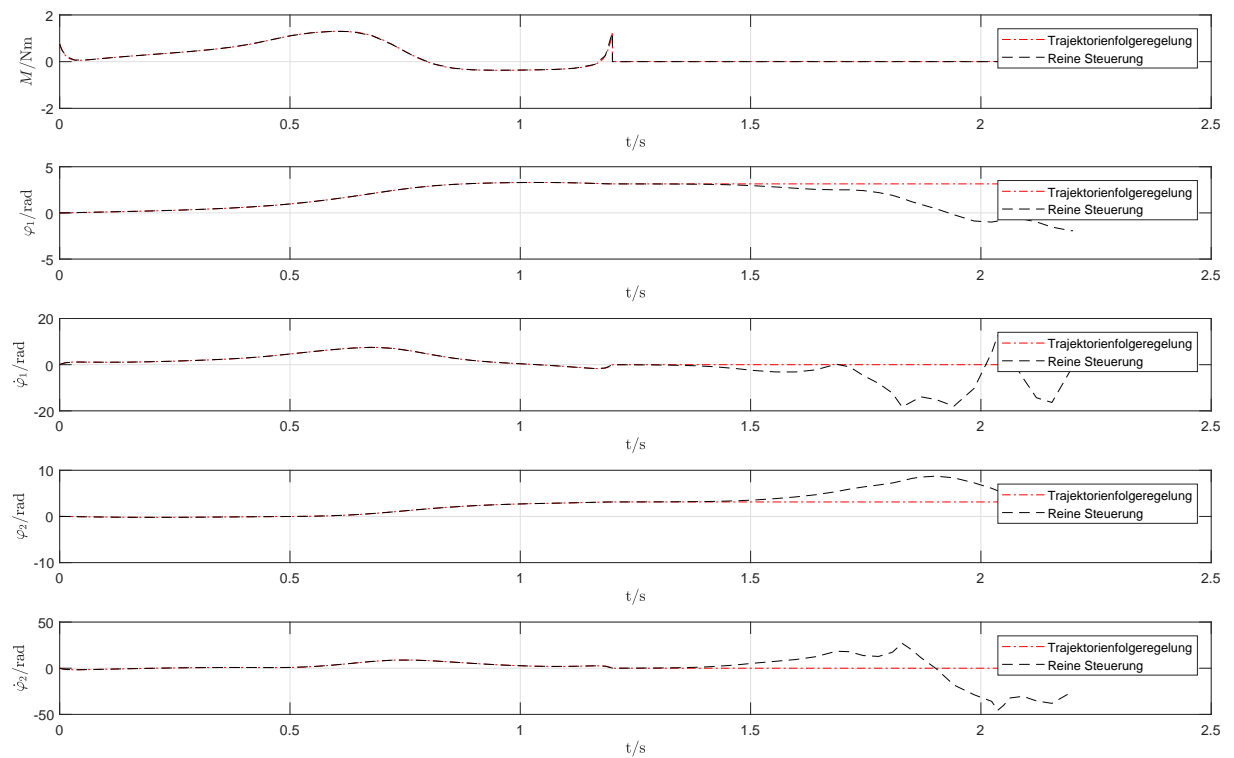
### 6.3 Folgeregelung unter Simulink

Die Trajektorienfolgeregelung wird unter Ausnutzung der getätigten Berechnungen in Simulink implementiert. Das folgende Bild zeigt das gesamte Simulink-Modell.

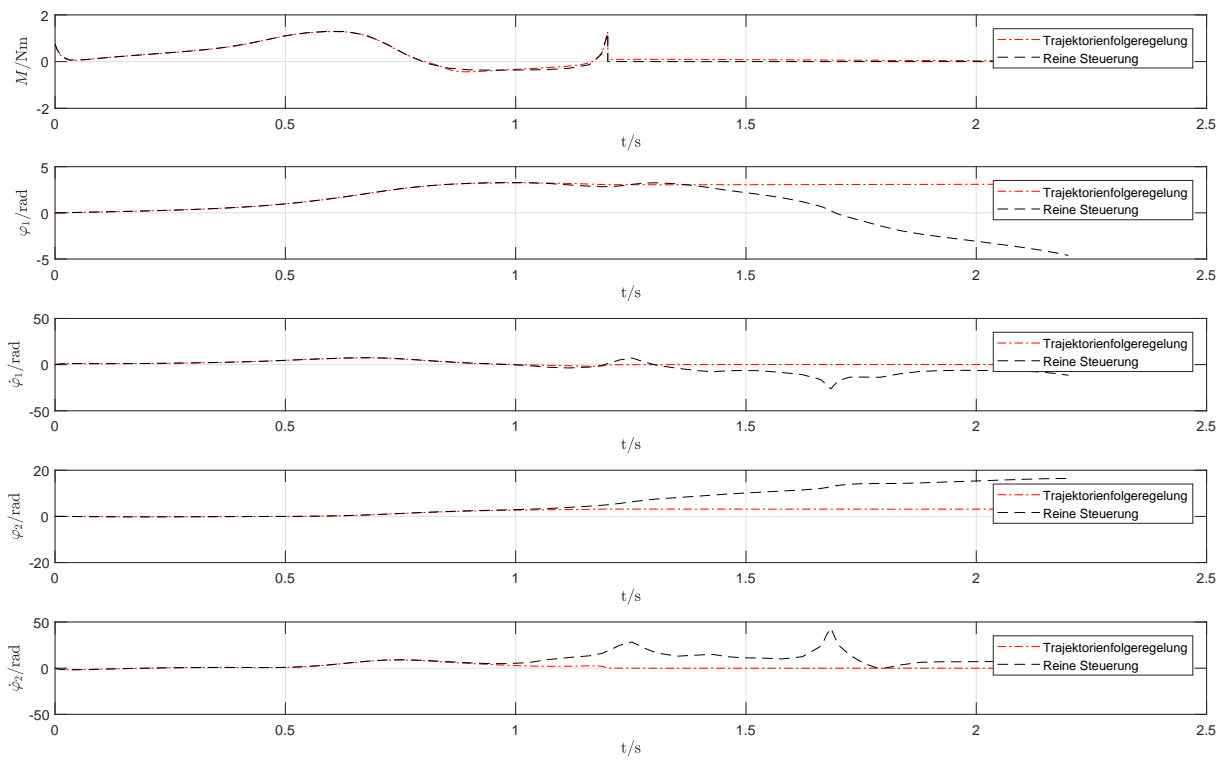


**Figure 6.1:** Modell der gesamten Trajektorienfolgeregelung

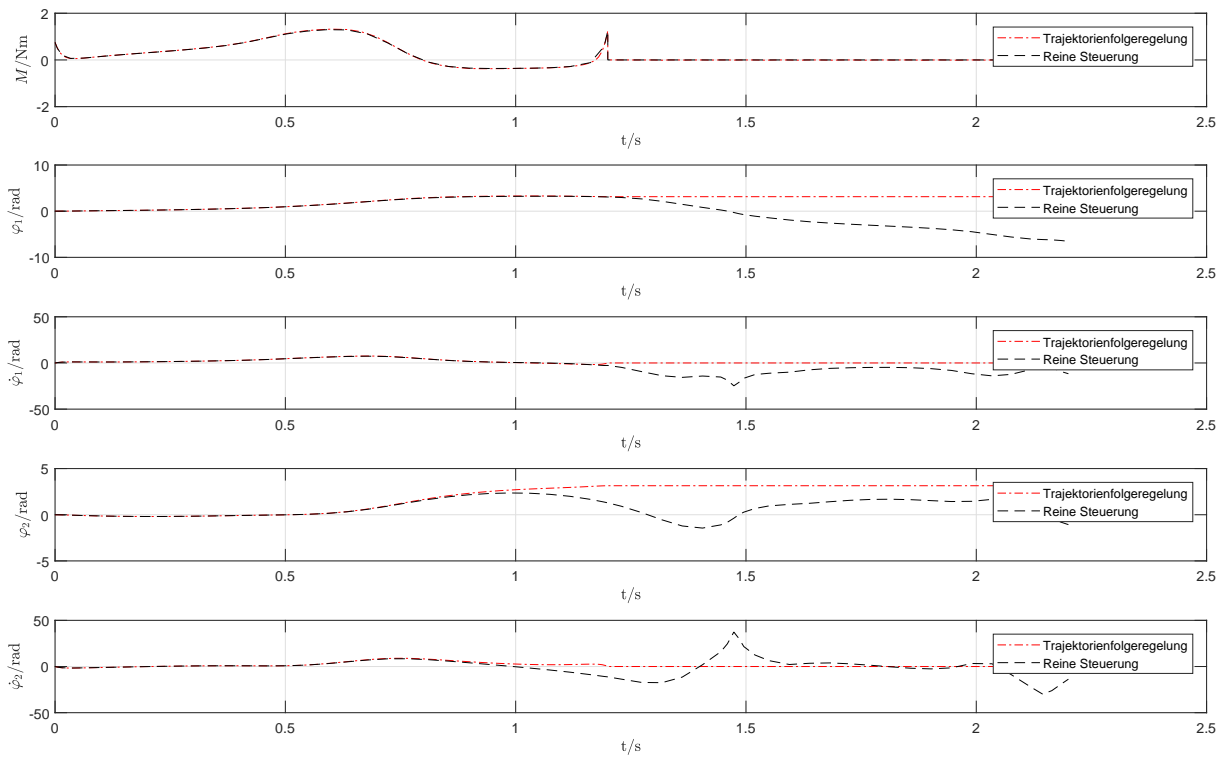
Um den Unterschied zwischen der reinen Steuerung und der Trajektorienfolgeregelung aufzuzeigen, werden im Nachfolgenden Plots gezeigt, wobei im zweiten und dritten Plot Parameteränderungen durchgenommen worden sind.



**Figure 6.2:** Plot der Zustände ohne Parameteränderung



**Figure 6.3:** Plot der Zustände mit 5% kürzerem zweiten Pendel



**Figure 6.4:** Plot der Zustände mit schwererem ersten Pendel



---

Welche Unterschiede stellen Sie beim Vergleich der reinen Steuerung und der Steuerung mit Folgeregelung fest?

- Ohne Parameteränderung: bis Ende der Übergangszeit stimmen Steuerung und Regelung übereinander. Danach ist Steuerung instabil, jedoch Regelung stabil (stationär genau)
- mit Parameteränderungen: Unterschiede sind schon während der Übergangszeit (besonders gegen Ende der Übergangszeit) zu sehen. Nach der Übergangszeit so wie ohne Parameteränderung (Steuerung instabil, Regelung stabil, stationär genau)

---

## 6.4 Einfluss des Antriebs

---

Motor modellieren

Vergleichen mit System vorher, verschiedene  $T_m$  ausprobieren → Wertebereich von  $T_m$  bestimmen