

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.

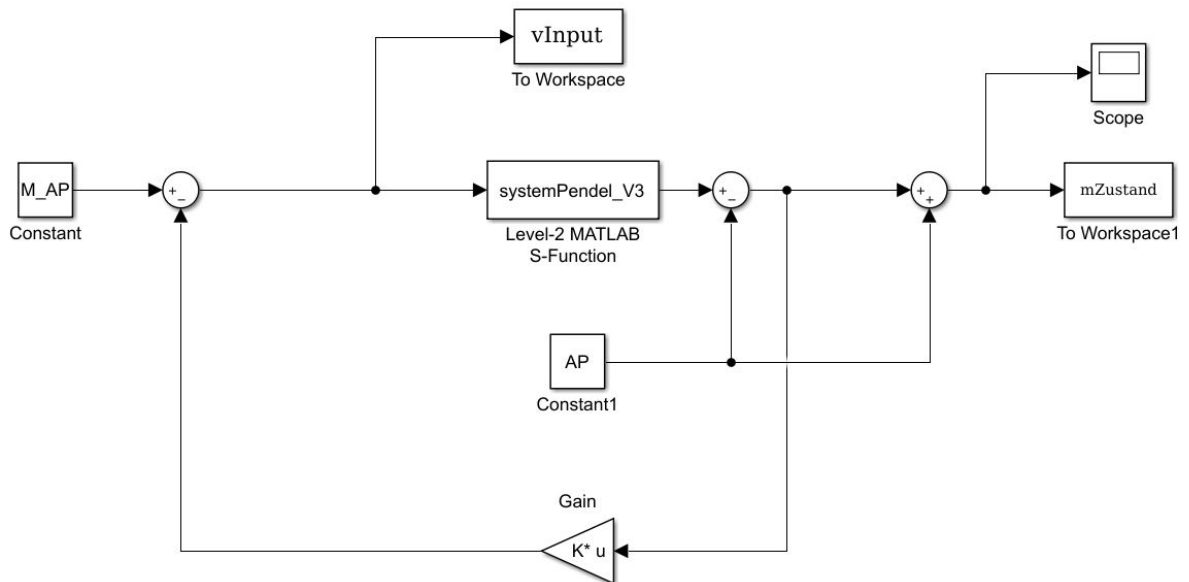


Figure 3.1: Implementierung mit Fcn-Blöcken

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
```

3.4 Animation des Pendels

Für die Animation des Pendels soll eine Funktion `animierePendel` implementiert und anschließend mit Simulationsergebnissen getestet werden.

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([0, P1(1)],[0, P1(2)],'b',...
30                [P1(1), P2(1)], [P1(2), P2(2)],'r');
    title(num2str(vTAnim(i)));

    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
```
