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.

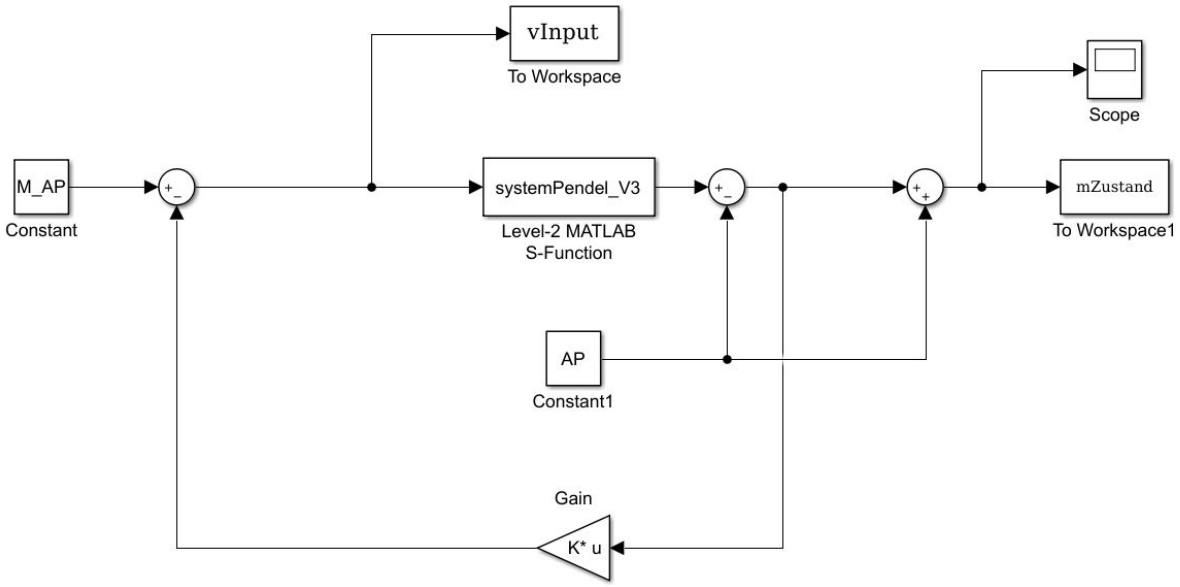

Figure 3.1: Implementierung mit Fcn-Blöcken

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])

hold on;
for i = 1:nBilder
    if i>1
        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',...
                    [P1(1), P2(1)], [P1(2), P2(2)],'r');
    title(num2str(vTAnim(i)));
```

```
    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
```

3.6 Tests