Versuch IV: Beobachterentwurf - Benutzeroberflächen

Andreas Jentsch, Ali Kerem Sacakli

Praktikumsbericht – Praktikum Matlab/Simulink II

27. Juni 2017

TECHNISCHE UNIVERSITÄT DARMSTADT

REGELUNGSTECHNIK UND MECHATRONIK rtm

4.8 Verhalten des Regelkreises mit Beobachter

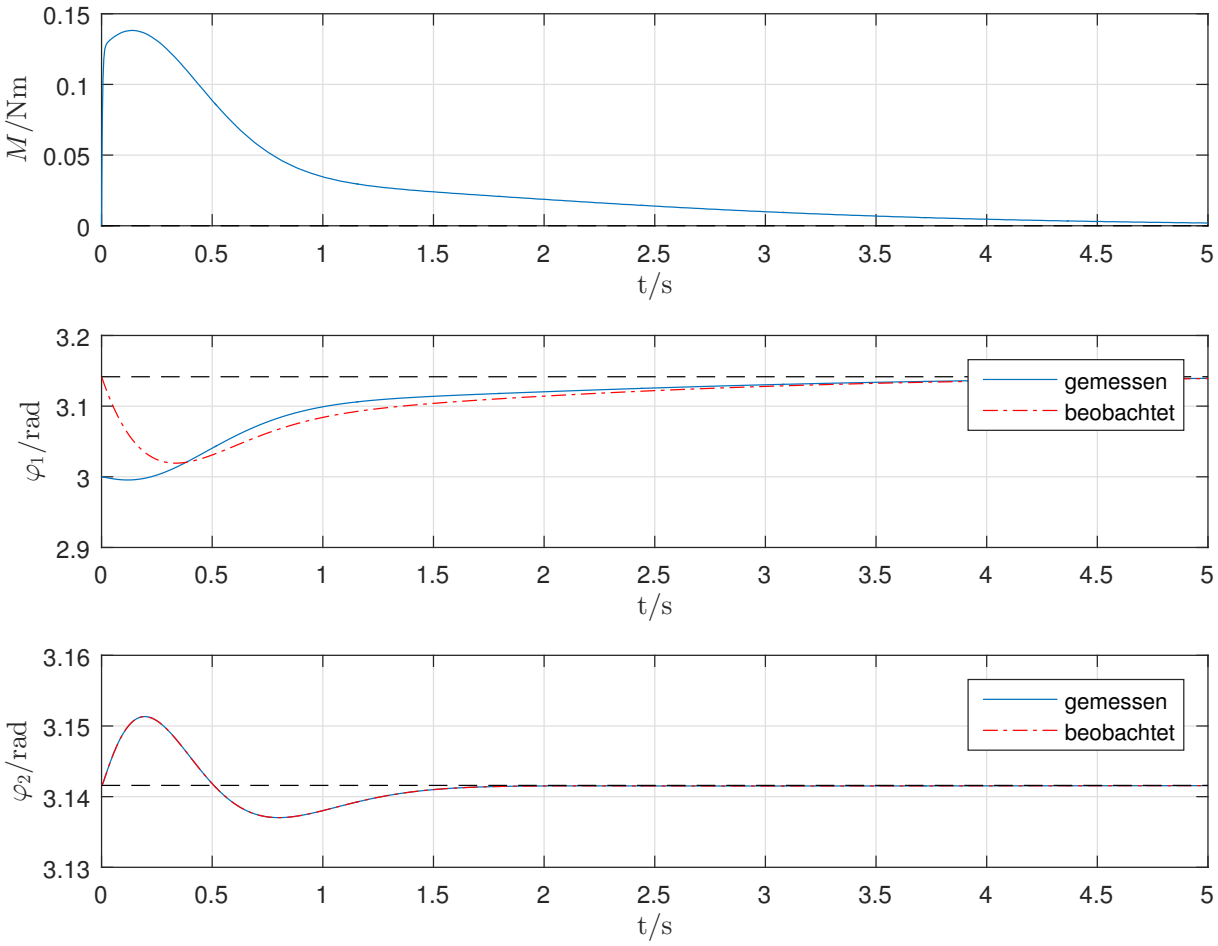

Figure 4.1: Systemverhalten mit Beobachtereigendwerten bei: [-1 -1 -5 -5]

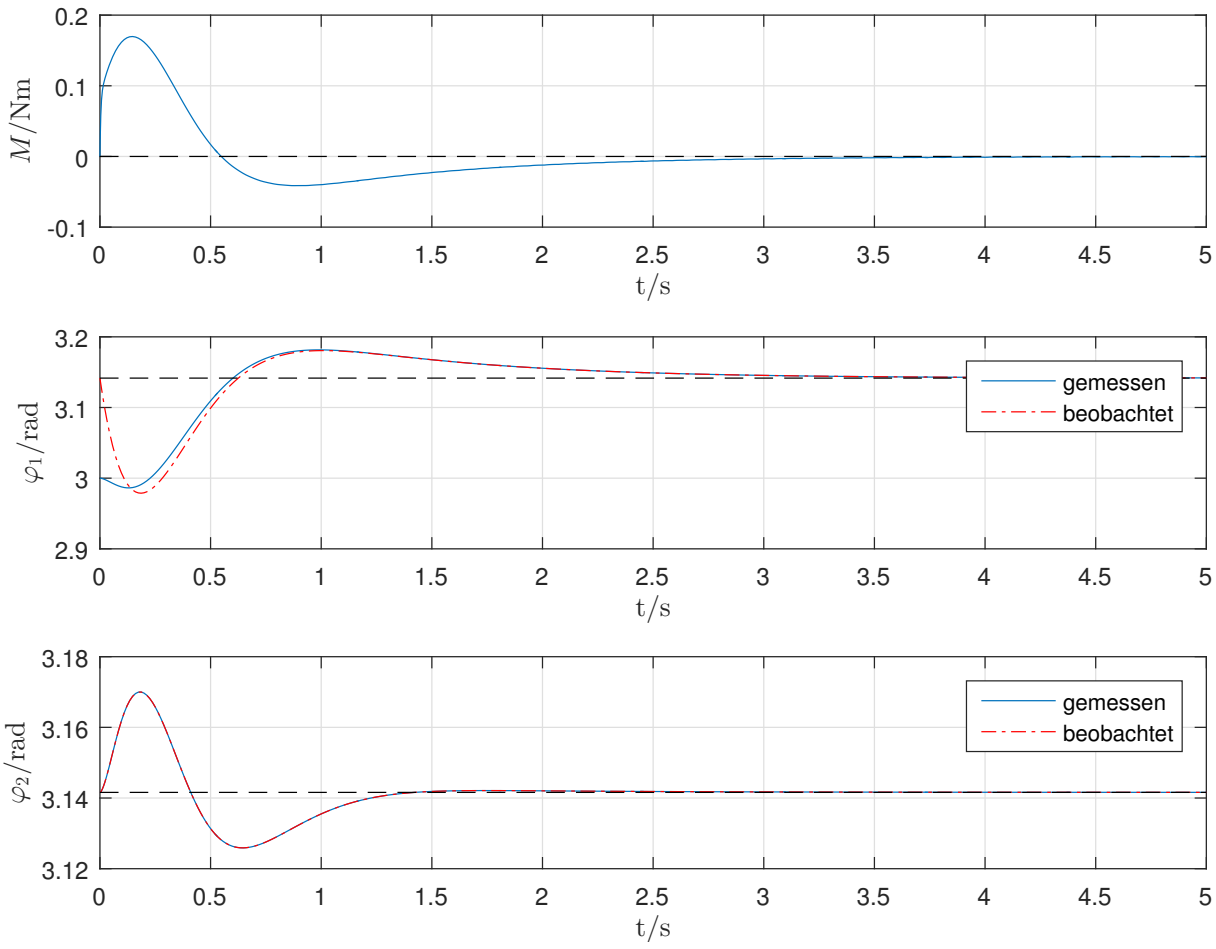

Figure 4.2: Systemverhalten mit Beobachtereigendwerten bei: [-5 -5 -10 -10]

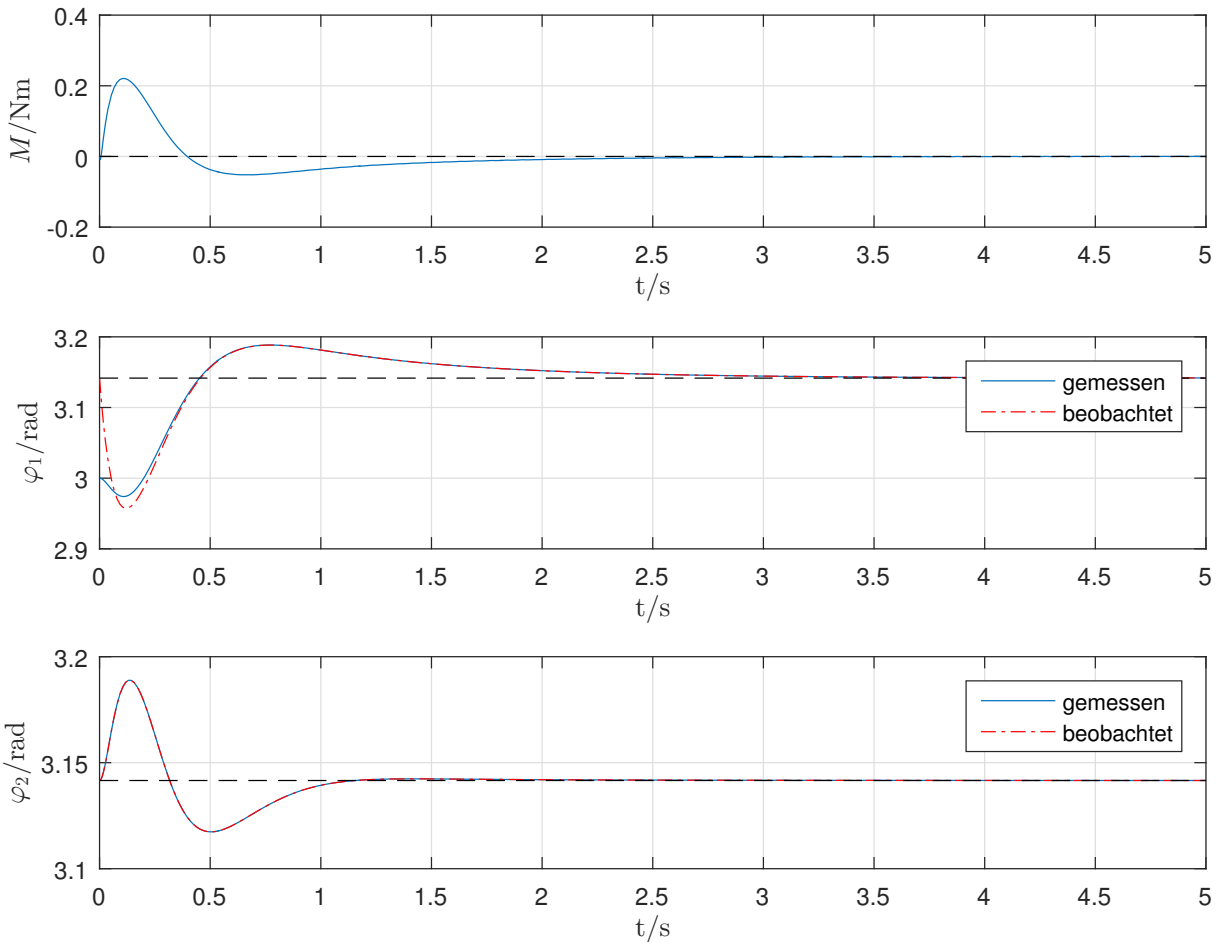

Figure 4.3: Systemverhalten mit Beobachtereigendwerten bei: [-10 -10 -20 -20]

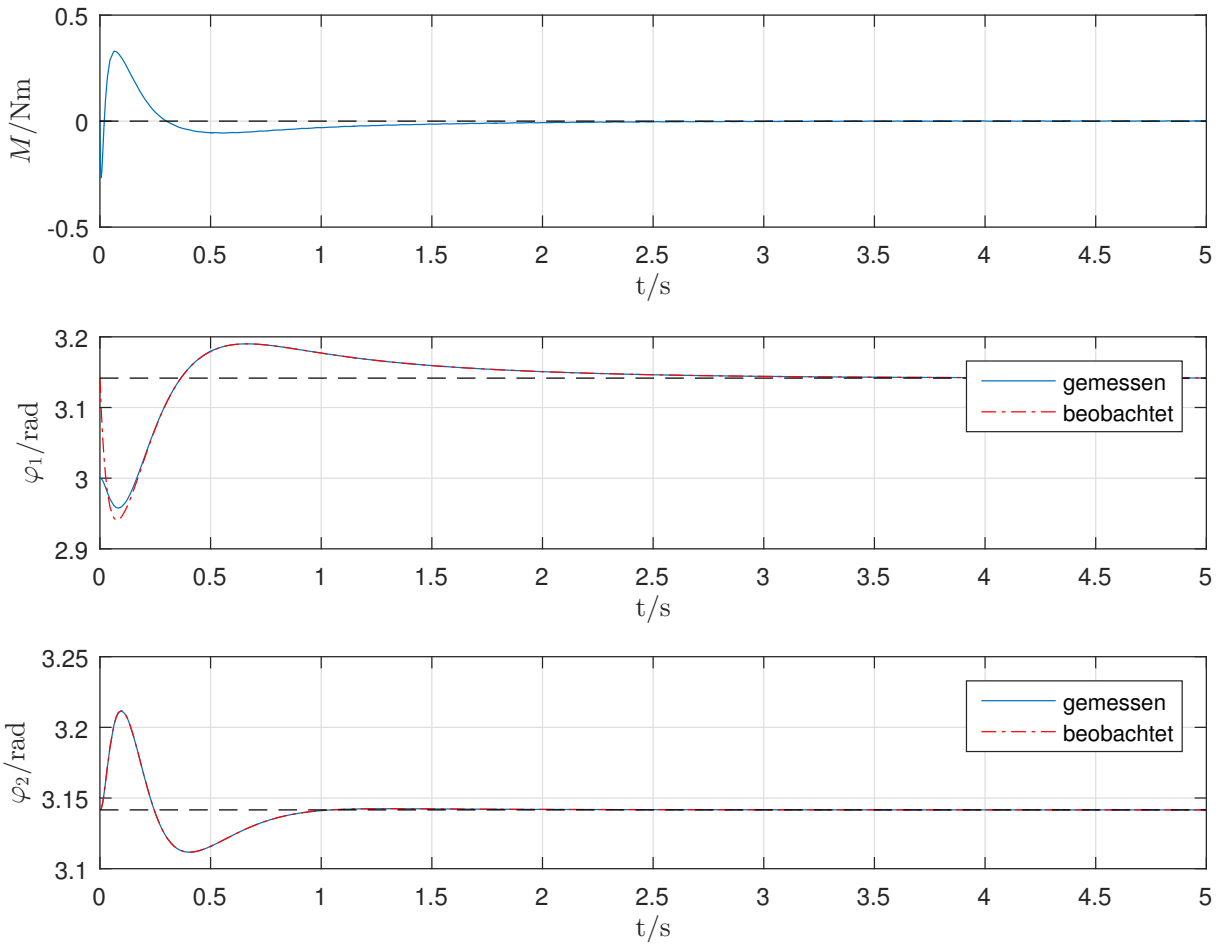

Figure 4.4: Systemverhalten mit Beobachtereigendwerten bei: [-20 -20 -40 -40]

4.9 GUI Entwurf

Unter Ausnutzung der schon erstellten Funktionen soll in diesem Versuch das Modell durch eine grafische Benutzeroberfläche (Figure 4.5)und einen Luenberger-Beobachter erweitert werden. Die wichtigsten Callback-Funktionen sind in Listing 4.1 bis 4.3 aufgeführt.

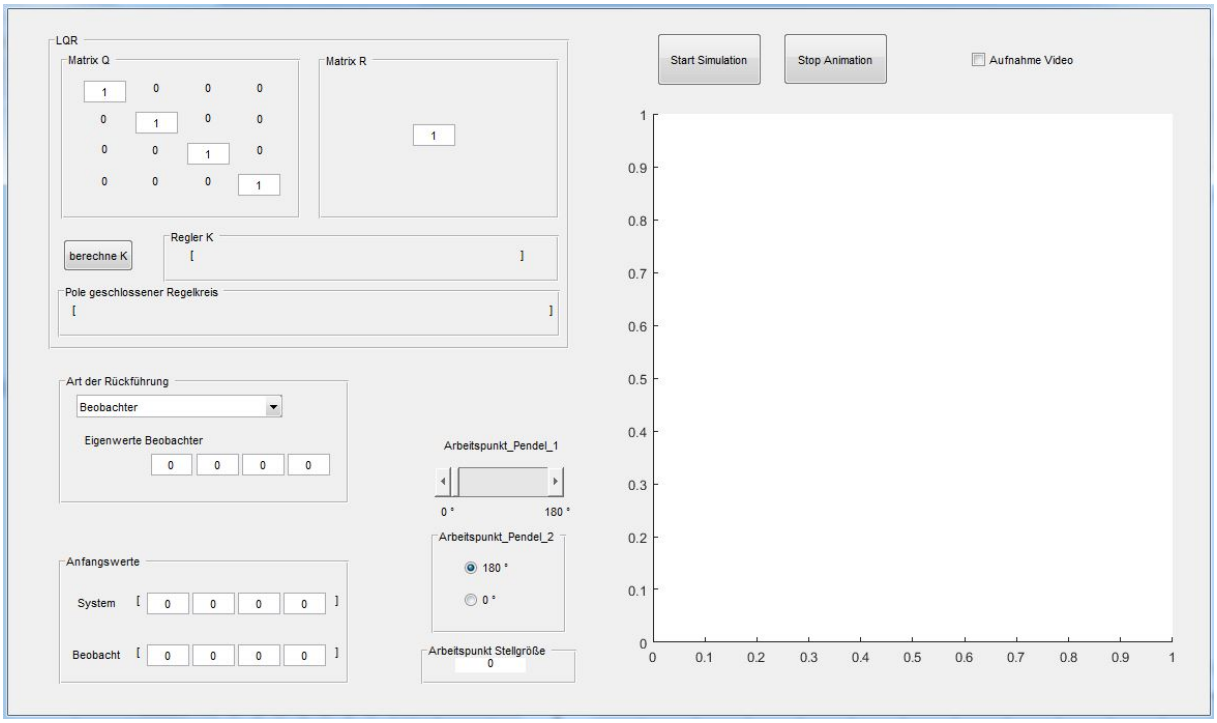

Figure 4.5: Finale grafische Benutzeroberfläche

Listing 4.1: Quellcode der Callback-Funktion zur Reglerberechnung

```
% --- Executes on button press in berechneK.
function berechneK_Callback(hObject, eventdata, handles)
    % hObject    handle to berechneK (see GCBO)
    % eventdata  reserved - to be defined in a future version of →
        ←MATLAB
    % handles    structure with handles and user data (see GUIDATA)

    % Struktur mit den Handles aller Objekte der GUI erzeugen
    h = guihandles();

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Auslesen der Matrix Q
    q11 = str2num(get(h.Q11,'String'));
    q22 = str2num(get(h.Q22,'String'));
    q33 = str2num(get(h.Q33,'String'));
```

```
    q44 = str2num(get(h.Q44,'String'));

    Q = diag([q11 q22 q33 q44]);
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Auslesen von R
    R = str2num(get(h.R,'String'));
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % Auslesen des Arbeitpunkts
    % !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    % Ggf. an eigene Codierung des Arbeitspunktes anpassen!
    % !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
    AP = [0 0 0 0];
    value1 = get(h.slider_AP,'Value');
        AP(1) = value1*pi;
    value2 = get(h.AP_2_1,'Value');
    if (value2 == 1)
        AP(3) = pi;
    else % (value == 0)
        AP(3) = 0;
    end
    %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    [f_m, h_m] = nonlinear_model();

    [A, B, C, D, M_AP] = linearisierung(f_m, h_m, AP);

    stObs = getappdata(h.figure1,'stObs');
    stObs.A = A;
    stObs.B = B;
    stObs.C = C;
    setappdata(h.figure1,'stObs',stObs);

    [K, poleRK] = berechneLQR(A, B, Q, R);
    % Anzeigen des Vektors 'K' im Textfeld 'reglerK'

    set(h.reglerK, 'String', num2str(K));
    set(h.poleRK, 'String', num2str(poleRK'));
```

```
56      set(h.M_AP,'String',num2str(M_AP));
% end function berechneK_Callback
```

Listing 4.2: Quellcode Callback-Funktion zum Start der Simulation

```
% --- Executes on button press in startSim.
function startSim_Callback(hObject, eventdata, handles)
% hObject    handle to startSim (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Initialisiert variable zum stoppen der Animation
global stopAnimation;
stopAnimation = false;

h = guihandles();
cla(h.axes1);

% Startwerte aus der GUI auslesen
x0(1,1) = str2num(get(h.x01,'String'));
x0(2,1) = str2num(get(h.x02,'String'));
x0(3,1) = str2num(get(h.x03,'String'));
x0(4,1) = str2num(get(h.x04,'String'));

% Arbeitspunkt aus der GUI auslesen
AP = [0 0 0 0];
value1 = get(h.slider_AP,'Value');
    AP(1) = value1*pi;
value2 = get(h.AP_2_1,'Value');
if (value2 == 1)
    AP(3) = pi;
else % (value == 0)
    AP(3) = 0;
end
M_AP = str2num(get(h.M_AP,'String'));

% Regler aus der GUI auslesen
K = str2num(get(h.reglerK,'String'));
stPendel = ladePendel();
```

```
% Reglerpole aus der GUI auslesen
stObs = getappdata(h.figure1,'stObs');
stObs.pole(1) = str2num(get(h.lam_b_1,'String'));
stObs.pole(2) = str2num(get(h.lam_b_2,'String'));
stObs.pole(3) = str2num(get(h.lam_b_3,'String'));
stObs.pole(4) = str2num(get(h.lam_b_4,'String'));

% Reglerstartwerte aus der GUI auslesen
stObs.x0(1) = str2num(get(h.x01b,'String'));
stObs.x0(2) = str2num(get(h.x02b,'String'));
stObs.x0(3) = str2num(get(h.x03b,'String'));
stObs.x0(4) = str2num(get(h.x04b,'String'));

% Fragt ab ob mit oder ohne Beobachter
logic = get(h.popupmenu2,'Value');
if logic == 1
    stObs.switch  = true;
else
    stObs.switch = false;
end

% Beobachter berechnen
stObs.L = berechneBeobachter(stObs.A,stObs.C,stObs.pole);
setappdata(h.figure1,'stObs',stObs);

% Simulation des Modells
[vT, mX, mXobs, u] = runPendel(stPendel, AP, K, x0, M_AP, stObs);

% Variablen zum plotten in den Base Workspace schreiben
assignin('base','vT',vT);
assignin('base','mX',mX);
assignin('base','mXobs',mXobs);
assignin('base','u',u);
assignin('base','M_AP',M_AP);
assignin('base','x0',x0);

% Abfragen ob die animation aufgezeichnet werden soll
if get(h.aufnahme,'Value') == 1
    record = true;
else
```

```
    record = false;
end

%Animation des Pendels
animierePendel(vT,mX,stPendel,h.axes1,record);
```

Listing 4.3: Quellcode der Callback-Funktion zum Stoppen der Animation

```
% --- Executes on button press in stopAnimation.
function stopAnimation_Callback(hObject, eventdata, handles)
% hObject    handle to stopAnimation (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
global stopAnimation;
stopAnimation = true;
```

4.10 Beobachterentwurf

Zur Berechnung der Beobachter-Matrix L soll die Funktion `berechneBeobachter` implementiert werden.

Listing 4.4: Quellcode der Funktion berechneBeobachter

```
function L = berechneBeobachter(A, C, poleBeobachter)
MB = obsv(A,C);

if rank(MB)==length(A)
L = place(A', C', poleBeobachter)';
else
 disp('System nicht vollständig beobachtbar');
end

end
```

Für die Erweiterung soll zudem die Funktion `runPendel` erweitert werden:

Listing 4.5: Quellcode der Funktion runPendel

```
function [ vT, mX, mXobs, u ] = runPendel( stPendel, AP, K, x0, M_AP→
   ←, stObs )

vT = 'error';
```

```
mX = 'error';
mXobs = [];
u = [];

if ~isempty(stObs)
    stObs.switch = true;
else
    stObs.switch = false;
    stObs.A = eye(4);
    stObs.B = [0;1;0;1];
    stObs.C = [1 0 0 0; 0 0 1 0];
    stObs.L = stObs.C';
    stObs.x0 = [0 0 0 0];
end

Tend = 10;
stOptions = simset( 'SrcWorkspace', 'current');
sim('Modell_V4', Tend, stOptions);

vT = mZustand.Time;
mX = mZustand.Data;
mXobs = mBeobacht.Data;
u = vInput.Data;

end
```

Das Zugrundeliegende Simulink Modell mit Beobachter ist in Figure 4.6 zu sehen.

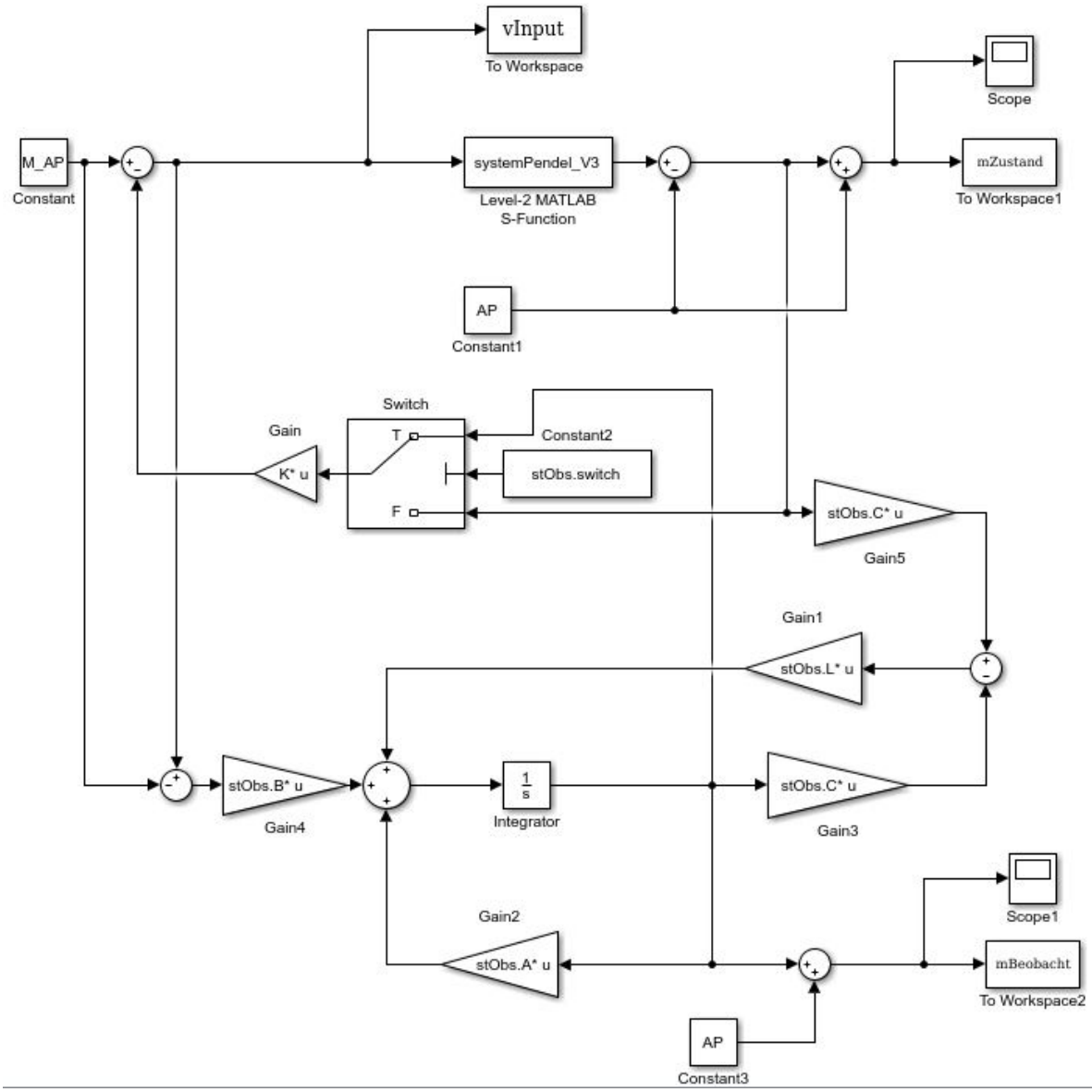

Figure 4.6: Simulink-Modell mit Luenberger-Beobachter