Kontinuierliche Simulation

325.040 - Projekt 47 - *Sommersemester 2016*

FABIAN WEDENIK - 1426866
ALEXANDER WIMMER - 1328958
FELIX HOCHWALLNER - 1328839
OSKAR FÜRNHAMMER - 1329133

Studienkennzahl 033 282

E325 Institut für Mechanik und Mechatronik

Contents

List of Figures

Vorwort

Sehr geehrte Damen und Herren, liebe Leser und Leserinnen!

Das vorliegende Protkoll wurde im Rahmen der Vorlesung und Übung *Kontinuierliche Simluation (325.040/325.041)* verfasst und beschäftigt sich mit der Implementierung einer einfachen Regelung eines mechanischen Doppelpändels, sowohl in MATLAB, als auch in MalpeSim.
Dadurch soll unter anderem ein Vergleich zwischen klassischer textuelle Programmierung und grafischer, blockorientierter Modellierung gezogen werden. Betreut wurde das Projekt der Gruppe 47 von XXXXXX MISTER UNIVERSE XXXX.

Viel Spaß beim Lesen wünschen

Aufgabenstellung

Sowohl mit MATLAB als auch MapleSim soll ein mechanisches Modell eines geregelten Doppelpendels realisiert werden. Dabei soll unter anderem ein Vergleich zwischen klassischer textueller Programmierung in MATLAB und grafischer, blockorientierter Modellierung in MapleSim gezogen werden.

Implementieren Sie das Modell mit MATLAB. Führen Sie einen Simulationslauf mit den angegebenen Parametern durch, plotten Sie die Auslenkung x sowie die beiden Winkel φ_1 und φ_2 über der Zeit und interpretieren Sie die Ergebnisse. Berechnen Sie mit MATLAB auch die Eigenwerte. Ist das System stabil? Begründen Sie Ihre Aussage.

Bauen Sie das Modell mit MapleSim auf, testen Sie das Modell mit den angegebenen Parametern und vergleichen Sie die Ergebnisse mit jenen aus der MATLAB-Simulation.

Modellbildung

Eine Masse m_m gleitet reibungsfrei auf einer horizontalen Ebene. An der Masse ist ein Stab (m_1, I_1, l_1) über ein reibungsfreies Gelenk befestigt. An seinem anderen Ende ist der Stab $m_1$mit einem weiteren Stab (m_2, I_2, l_2) gelenkig verbunden.

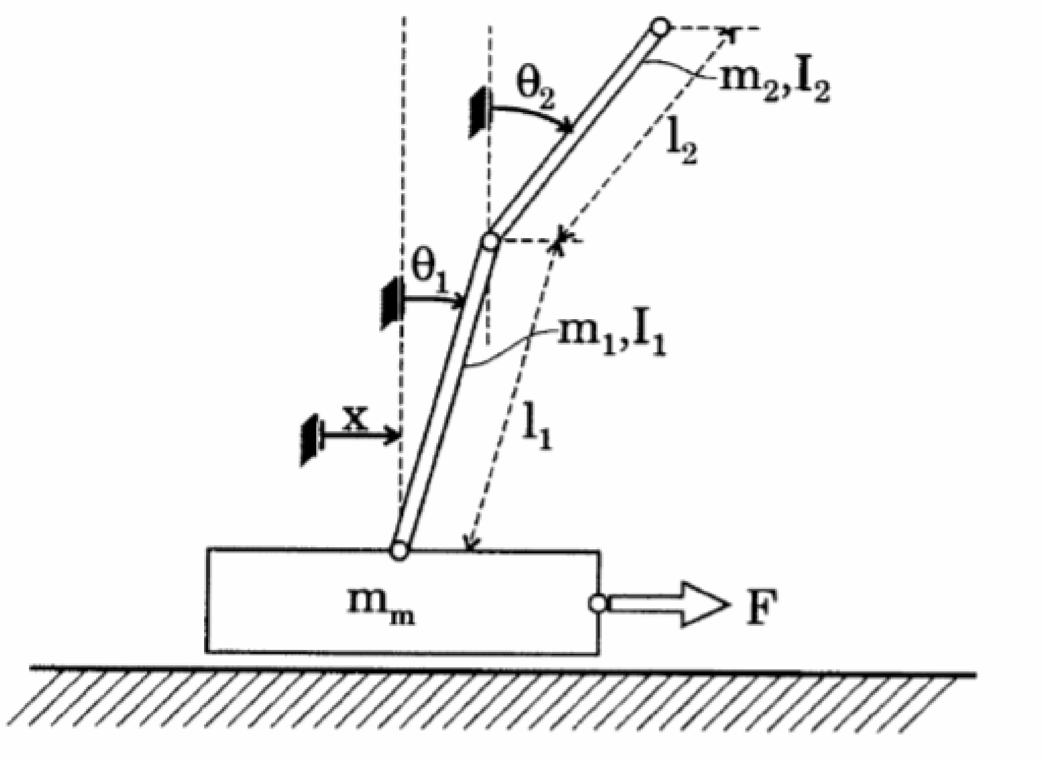

Figure 2.1: Mechanisches Modell eines stehenden Doppelpendels

Da wir bei der Berechnung der Matrizen, welche für eine Zustandsraumdarstellung erfoderlich sind, einige Probleme hatte entschlossen wir uns sicherheitshalber mittels Euler-Lagrange-Formalismen auch die Bewegungsgleichungen neu aufzustellen und in MATLAB linearisieren zu lassen.

Auf die Modellbildung in MATLAB wird in Kapitel 3.1 näher eingegangen. Für die Implementierung in MapleSim wurde das gegebene mechanische Modell so übernommen, da eine

Implementierung in MATLAB

MATLAB ist eine numerische Programmiersprache, welche für die schnelle Manipulation und Berechnung von Matrizen entwickelt wurde. Programmiert wird unter Matlab in einer proprietären Programmiersprache, die auf der jeweiligen Maschine interpretiert wird. Die Programmierung erfolgt hierbei textuell.

3.1 Variablendifinition und Modellbildung

Bevor wir unser System simulieren lassen können, müssen wir unser mechanisches (Ersatz-)System in ein digitales Modell übersetzen. Dazu müssen dem Programm einige Parameter übergeben werden.
Zuerst werden Systemvariablen deklariert, sowie die Anzahl der Freiheitsgrade und Körper festgelegt. Außerdem wird ein Minimalkoordinatenvektor mit zugehörigen zeitlichen Ableitungen bestimmt.

```
%Definition der Systemvariablen
syms l1 l2 phi_1 phi_2 phi_p1 phi_p2 phi_pp1 phi_pp2
syms a a_p a_pp mm m1 m2 g I_1 I_2 F xc

frg=3;                              %Anzahl der Freiheitsgrade
n=3;                                %Anzahl der Koerper

q=[a ; phi_1 ; phi_2];              %Minimalkoordinaten
q_p=[a_p ; phi_p1 ; phi_p2];        %zeitliche Ableitungen
q_pp=[a_pp ; phi_pp1 ; phi_pp2];
```

work in progress...

3.2 Simulation

3.3 Plot und Ausgabe

3.4 sourcecode

```
%---- Ermitteln der Bewegungsgleichungen
%     definieren der Systemvariablen
syms l1 l2 phi_1 phi_2 phi_p1 phi_p2 phi_pp1 phi_pp2
syms a a_p a_pp mm m1 m2 g I_1 I_2 F xc

frg=3;                                   %Anzahl der Freiheitsgrade
n=3;                                     %Anzahl der Koerper

q=[a ; phi_1 ; phi_2];                   %Minimalkoordinaten
q_p=[a_p ; phi_p1 ; phi_p2];             %zeitliche Ableitungen
q_pp=[a_pp ; phi_pp1 ; phi_pp2];

%---- Drehmatrix Stab 1
T_IK1 = [cos(phi_1) sin(phi_1) 0;
        -sin(phi_1) cos(phi_1) 0;
              0          0     1];
%---- Drehmatrix Stab 2
T_IK2 = [cos(phi_2) sin(phi_2) 0;
        -sin(phi_2) cos(phi_2) 0;
              0          0     1];

%---- Ortsvektoren
I_r_Sm = [a;0;0];
I_r_S1 = [a+l1/2*sin(phi_1) ; l1/2*cos(phi_1) ; 0];
I_r_Q2 = [a+l1*sin(phi_1) ; l1*cos(phi_1) ; 0];
K1_r_Q1S1 = [0; l1/2; 0];
K2_r_Q2S2 = [0; l2/2; 0];
I_r_S2 = I_r_Q2 + T_IK2 * K2_r_Q2S2;

%---- Traegheitstensoren in den koerperfesten Koordinatensystemen
K1_I_S1 = diag([0 0 I_1]);
K2_I_S2 = diag([0 0 I_2]);

%---- Winkelgeschwindigkeitsvektoren der Staebe
K_om1 = [0 ; 0 ; -phi_p1];
K_om2 = [0 ; 0 ; -phi_p2];

%---- JACOBI-Matrizen der Translation
J_Tm = jacobian(I_r_Sm , q);
J_T1 = jacobian(I_r_S1 , q);
J_T2 = jacobian(I_r_S2 , q);

```

```
%---- JACOBI-Matrizen der Rotation
J_R1 = jacobian(K_om1, q_p);
J_R2 = jacobian(K_om2, q_p);

%---- Geschwindigkeitsvektoren
I_v_Sm = J_Tm*q_p ;
I_v_S1 = J_T1*q_p ;
I_v_S2 = J_T2*q_p ;

%---- kinetische Energie
T = 1/2*(mm*(I_v_Sm.'*I_v_Sm)+m1*(I_v_S1.'*I_v_S1)+m2*(I_v_S2.'*I_v_S2) ...%Translation
    +K_om1.'*K1_I_S1*K_om1+K_om2.'*K2_I_S2*K_om2);                              %Rotation
T = simplify(T);                                                                 %Vereinfachung

%---- potentielle Energie
V=-(m1*I_r_S1.'+m2*I_r_S2.')*[0 ; -g ; 0];

%---- Ableitungen fuer LAGRANGEsche Gleichung 2. Art
dTdv = simplify(jacobian(T,q_p).');          %mit transponieren zu Spaltenvektor gemacht
dTdq = simplify(jacobian(T,q).');
dVdq = simplify(jacobian(V,q).');

%---- Elemente der Bewegungsgleichung M(q)*q_pp + f(q,q_p) = 0
disp('System-Massenmatrix M')
M = simplify(jacobian(dTdv,q_p))
disp('System-Vektorfunktion f')
f = simplify(jacobian(dTdv,q)*q_p+dVdq-dTdq-[F;0;0])

%==========================================================================
%---- Linearisierung um die Gleichgewichtslage:
%      phi_1 = 0, phi_2 = 0, a = 0

disp(' ')
disp('Elemente der linearisierten Bewegungsgleichung')
disp('System-Massenmatrix M0')
M0 = subs(M,{phi_1, phi_2, a},{0, 0, 0})
f0 = subs(f,{a, phi_1, phi_2, a_p, ...
    phi_p1, phi_p2},{0, 0, 0, 0, 0, 0});
disp('Auslenkungs-proportionaler Anteil')
Q = subs(jacobian(f,q),{a, phi_1, phi_2, a_p, ...
    phi_p1, phi_p2},{0, 0, 0, 0, 0, 0})
disp('Steifigkeitsmatrix K')
K = 1/2*(Q+Q.')
disp('Matrix der nichtkonservativen Kraefte')
N = 1/2*(Q-Q.')
disp('gesschw.-proportionaler Anteil')
P = subs(jacobian(f,q_p),{a, phi_1, phi_2, a_p, ...
    phi_p1, phi_p2},{0, 0, 0, 0, 0, 0})
```

```
disp('Daempfungsmatrix')
D = 1/2*(P+P.')
disp('gyroskopischer Anteil')
G = 1/2*(P-P.')

%
    ==================================================================

%----Erstellen und Simulieren der Zustandsraumdarstellung
syms x th1 th2 x_p th1_p th2_p
syms x_pp th1_pp th2_pp

y = [q.',q_p.'].';
y_p = [q_p.',x_pp , th1_pp , th2_pp].';

A = [zeros(3),eye(3);
    -M0^(-1)*Q, -M0^(-1)*P];
A = double(subs(A,{mm, m1, m2, l1 , l2 , g, I_1 , I_2}, ...
    {0.2, 0.01, 0.01, 0.5, 0.7, 9.81, 2.0833e-04, 4.0833e-04})
        );
A(7,7) = 0;
A(7,1) = -1

B = [zeros(3,1);M0^(-1)*[1;0;0]];
B = double(subs(B,{mm, m1, m2, l1 , l2 , g, I_1 , I_2}, ...
    {0.2, 0.01, 0.01, 0.5, 0.7, 9.81, 2.0833e-04, 4.0833e-04})
        );
B(7,1) = 0
Bxc = [0; 0; 0; 0; 0; 0; 1]

C = [1 0 0 0 0 0 0;
   0 1 0 0 0 0 0;
   0 0 1 0 0 0 0]

D = [0; 0; 0]


Q=eye(7);
r=1;

%----lqr Regelungsentwurf
k = lqr(A,B,Q,r)

%----neue Zustandsraumsystemmatrizen nach
    Parameterruekfuehrung
Ac = [(A-B*k)];
Bc = [Bxc];
Cc = [C];
Dc = [D];

states = {'x' 'th1' 'th2' 'x_p' 'th1_p' 'th2_p' 'in'};
inputs = {'F'};
outputs = {'x' 'th1' 'th2'};

```

```
sys_cl = ss(Ac,Bc,Cc,Dc,'statename',states,'inputname',inputs,'outputname',outputs);

%----definieren des Simulationszeitraums
t = 0:0.01:8;

%----definition des konstanten 0.2m offsets als Input
u =0.2*ones(size(t));

%----Simulation des erstellten Systems ueber gegebene Zeit mit bekanntem
%Input
[y,t,x]=lsim(sys_cl,u,t);


%----Drei einzelne Diagramme in einem Fenster
figure(1);
ax(1) = subplot(3,1,1);
    plot(ax(1),t,y(:,1),'b');
    title(ax(1),'cart position');      %Titel, Beschriftungen, Kommentare,
    ylim([-0.1,0.25]);                  %andere Farben, andere skalierungen,
    grid on                             %da kann man sich noch frei austoben.
ax(2) = subplot(3,1,2);                 %relativ einfach verstaendliche
    plot(ax(2),t,y(:,2),'r');           %loesung. Ws nicht Laufzeit optimiert
    title(ax(2),'angle theta 1');
    grid on
ax(3) = subplot(3,1,3);
    plot(ax(3),t,y(:,3),'g');
    title(ax(3),'angle theta 2');
    grid on

%----Plotten der Ausgangsgroessen
% [AX,H1,H2] = plotyy(t,y(:,1),t,y(:,2),'plot');
% hold on
% line(t,y(:,3),'parent',AX(2),'color','g')
% hold off
% set(get(AX(1),'Ylabel'),'String','cart position (m)')
% set(get(AX(2),'Ylabel'),'String','pendulum angles (radians)')
% title('Step Response with LQR Control')

%----Berechnung der Eigenwerte
Eigenwerte = eig(Ac)
disp('Das System ist stabil, da der Realteil aller Eigenwerte negativ ist!')
```

Implementierung in MalpeSim

Nachdem das vorherige Kapitel ausschließlich der Implementierung in MATLAB gewidment wurde, beschäftigt sich dieses Kapitel nun mit der Umsetzung in einer *nicht klassischen*, blockorientierten, grafischen Programmierung in MapleSim.
Wie bereits erwähnt funktioniert die Programmierung in MapleSim grafisch. Es wird zuerst das Modell (in unserem Fall das geregelte mechanische Doppelpendel) im MapleSim GUI nachgebildet. Anschließend können Signale direkt an diesem Model abgegriffen und ins System rückgeführt werden.. Dadurch lassen sich selbst komplexe dynamische Systeme aus allen Bereichen der Natur- und Ingeneurswissenschaften vergleichsweise einfach modellieren.

4.1 Modellbildung

Da hier keine mathematischen Transformationen mehr nötig waren um das Doppelpendel in MapleSim modellieren zu können wurden direkt die Größen aus der Angabe verwendet. Die Materialparamter sind selbstverständlich die selben, wie die, die auch schon in den anderen Kapiteln verwendet wurden.

what what