Regelung mechatronischer Systeme

Hausübung, ausgegeben von: Frank Woittennek
zuletzt geändert: 17. Dezember 2022

In industriellen Bestückprozessen werden häufig sogenannte Flächenportalsysteme eingesetzt. Dabei kann eine Bestückeinheit mit Hilfe von Linearantrieben in der horizontalen Ebene positioniert werden und der Bestückkopf mit einem weiteren Antrieb in vertikaler Richtung verfahren werden. Ein Beispiel für ein Flächenportal, das auch als Laborversuch am IACE vorhanden ist, kann Abbildung 1 entnommen werden. Es besteht aus einem Gestell, auf dem hier sogar zwei Brücken (Querbalken) bewegt werden können. Auf jeder dieser Brücken befindet sich ein weiterer Schlitten, so dass im Rahmen der geometrischen Beschränkungen jeweils beliebige Punkte in der Ebene angefahren werden können. Eine Vertikalkinematik ist im abgebildeten Versuchsaufbau nicht vorhanden. Bei sehr schnellen Positioniervorgängen

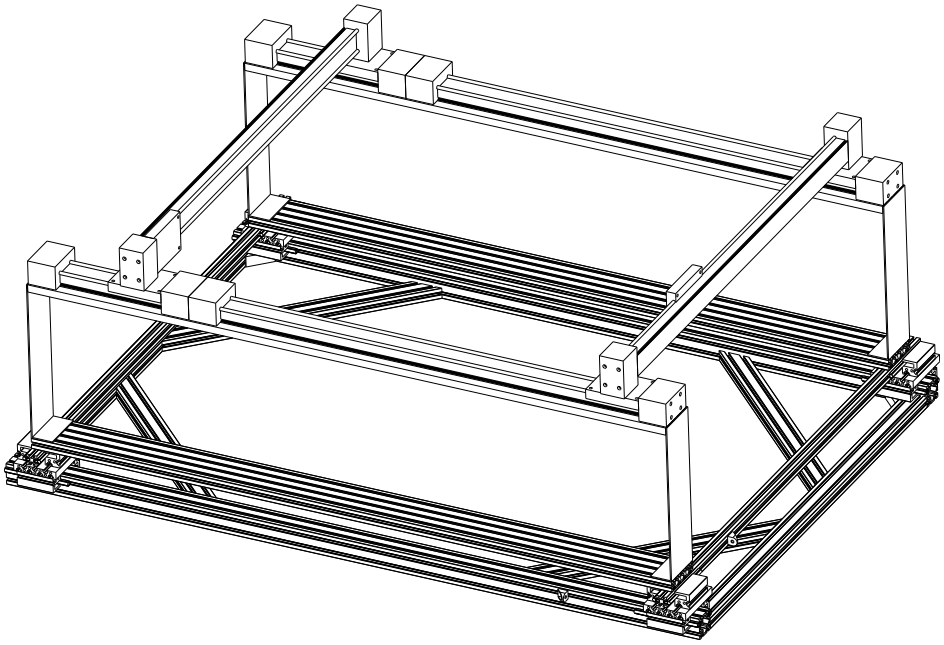

Abb. 1 – Laborversuch Flächenportal am IACE

kann das Gestell des Portalsystems zum Schwingen angeregt werden. Derartige unerwünschte Schwingungen spielen auch bei sehr massiv aufgebauten Portalen eine Rolle, wenn die Genauigkeitsanforderungen im betrachteten Bestückprozess hoch sind.

In der Hausübung wird als Abstraktion der beschriebenen Anwendung ein vereinfachter Aufbau betrachtet, in dem die Elastizität ebenfalls von Bedeutung ist. Dieser Aufbau besteht aus einer auf vier Federstäben gelagerten Platte der Masse M und Trägheitsmoment J. Die Federstäbe mit den identischen Federkonstanten k befinden sich jeweils im Abstand ℓ vom Mittelpunkt (z_1, z_2) der Platte entfernt und besitzen die identische Federkonstante k. Der Verdrehwinkel der Platte aufgrund der im System vorkommenden Elastizitäten wird mit φ bezeichnet. Auf der Platte kann sich ein Schlitten in horizontaler Richtung beliebig bewegen, seine Position in einem mit der Platte mitbewegten Koordinatensystem mit Ursprung in deren Mittelpunkt sei durch die Koordinaten (x_1, x_2) gegeben. Nachfolgend wird der Einfachheit halber davon ausgegangen, dass die Beschleunigung als Eingang $u = (u_1, u_2)$ des Systems verwendet werden kann

$$u_1 = \ddot{x}_1, \quad u_2 = \ddot{x}_2.$$

Es sei darauf hingewiesen, dass diese Annahme im Hinblick auf die praktische Anwendung durchaus problematisch sein kann, da dafür insbesondere eine nahezu ideale Kompensation der Reibung im System erforderlich ist. In der Praxis realistischer ist stattdessen eine Konfiguration mit der Geschwindigkeit als Eingang. Die Achsgeschwindigkeiten können dann durch einen unterlagerten Regler mit hoher Genauigkeit eingestellt werden. Derartige unterlagerte Regler können mit den aus dem Bachelorstudium bekannten Methoden ausgelegt werden, sind allerdings nicht Gegenstand dieser Projektarbeit. Darüber hinaus dienen die Modelle mit dem Stelleingriff auf Beschleunigung- oder Kraftebene später als Basis für zusätzliche Entwurfsaufgaben. Dementsprechend wird der betrachtete, stark vereinfachte Zugang und aus Gründen der Umsetzbarkeit durchaus hinterfragbare Zugang zur Modellierung beibehalten.

Verwendet man als Zustand den Vektor

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ z_1 \\ z_2 \\ \varphi \\ \dot{x}_1 \\ \dot{x}_2 \\ p_1 \\ p_2 \\ p_\varphi \end{bmatrix}$$

worin p_1, p_2, p_φ die verallgemeinerten Impulse zu den Koordinaten z_1, z_2 und φ bezeichnen, so kann das System in der folgenden Form angeschrieben werden

$$\begin{aligned}
\dot{x}_1 &= x_6 \\
\dot{x}_2 &= x_7 \\
\mathbf{M}(\boldsymbol{x}) \begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{\varphi} \end{bmatrix} &= - \begin{bmatrix} m\cos(\varphi) & -m\sin(\varphi) \\ m\sin(\varphi) & m\cos(\varphi) \\ -mx_2 & mx_1 \end{bmatrix} \begin{pmatrix} x_6 \\ x_7 \end{pmatrix} + \begin{bmatrix} p_1 \\ p_2 \\ p_\varphi \end{bmatrix} \\
\dot{x}_6 &= u_1 \\
\dot{x}_7 &= u_2 \\
\dot{p}_1 &= -4kz_1 \\
\dot{p}_2 &= -4kz_2 \\
\dot{p}_\varphi &= -4k\ell^2 \sin(\varphi) + m(-\dot{\varphi}\dot{z}_1 x_1 - \dot{\varphi}\dot{z}_2 x_2 - \dot{z}_1\dot{x}_2 + \dot{x}_1\dot{z}_2)\cos(\varphi) \\
&\quad + m(\dot{\varphi}\dot{z}_1 x_2 - \dot{\varphi}\dot{z}_2 x_1 - \dot{z}_1\dot{x}_1 - \dot{z}_2\dot{x}_1)\sin(\varphi)
\end{aligned} \tag{1}$$

mit der Massenmatrix

$$\mathbf{M}(\boldsymbol{x}) = \begin{bmatrix} m+M & 0 & -m\left(x_1\sin(\varphi) + x_2\cos(\varphi)\right) \\ 0 & m+M & m\left(x_1\cos(\varphi) - x_2\sin(\varphi)\right) \\ -m\left(x_1\sin(\varphi) + x_2\cos(\varphi)\right) & m\left(x_1\cos(\varphi) - x_2\sin(\varphi)\right) & J + m\left(x_1^2 + x_2^2\right) \end{bmatrix}$$

Als interessierende Ausgangsgröße wird die Absolutposition des Schlittens in der Ebene betrachtet:

$$\begin{aligned}
y_1 &= z_1 + \cos(\varphi)x_1 - \sin(\varphi)x_2 \\
y_2 &= z_2 + \sin(\varphi)x_1 + \cos(\varphi)x_2
\end{aligned} \tag{2}$$

Dagegen kann zur Implementierung der Regelung zunächst lediglich die Relativposition x_1, x_2 des des Schlittens in Bezug auf den Rahmen gemessen werden:

$$\begin{aligned} y_{\mathrm{m},1} &= x_1 \\ y_{\mathrm{m},2} &= x_2 \end{aligned} \tag{3}$$

Die folgenden Aufgaben können unter Zuhilfenahme der Skriptsprache Python gelöst werden. Dazu erhalten Sie ein vorbereitetes IPython Notenbook und und vorbereitete Python-Bibliotheken, mit deren Hilfe Sie Ihre Lösung erarbeiten können.

Aufgabe 1 (Simulationsmodelle)

a) Vervollständigen Sie das in den Methoden `model`, `position_axis` und `position_total` der Klasse `Gantry` in der Datei `gantry.py` definierte Simulationsmodell derart, dass die Ableitungen der Zustandsgrößen bzw. der jeweilige Systemausgang zurückgegeben werden. Die Methoden für den Ausgang sollten dabei auch mit vektoriellen Argumenten der Länge N für die Zeit und entsprechenden matrixwertigen Argumenten der Dimension $10 \times N$ für den Zustand umgehen können.

Alle benötigten Parameter sind entsprechend der folgenden Tabelle bereits im Konstruktor als Attribute der Klasse initialisiert.

J	M	m	ℓ	k
$\frac{5}{3}\,\mathrm{kg}\cdot\mathrm{m}^2$	$10\,\mathrm{kg}$	$0.8\,\mathrm{kg}$	$\sqrt{\frac{1}{2}}\,\mathrm{m}$	$100\,\mathrm{N\,m^{-1}}$

Prüfen Sie ihre Implementierung mit Hilfe der Methode `verify_model` der Klasse `DynamicSystem`.

b) Linearisieren Sie die Bewegungsgleichungen (1), (2) um beliebige Ruhelagen

$$\boldsymbol{x}^T(t) = \bar{\boldsymbol{x}}^T = (\bar{x}_1, \bar{x}_2, 0, 0, 0, 0, 0, 0, 0, 0), \quad \boldsymbol{u}^T(t) = \bar{\boldsymbol{u}}^T = (0, 0), \quad \boldsymbol{y}(t) = \bar{\boldsymbol{y}}^T = (\bar{x}_1, \bar{x}_2).$$

Vervollständigen Sie die Gleichungen (LaTeX-Code) in Abschnitt 1.2.1 des Notebooks entsprechend.

Vervollständigen Sie die Methode `linearize` der Klasse `Gantry` in der Datei `gantry.py` derart, dass in Abhängigkeit der übergebenen Werte für die Ruhelage die Systemmatrizen A, B, C und D der linearisierten Zustandsdarstellung

$$\dot{\tilde{\boldsymbol{x}}}(t) = A\tilde{\boldsymbol{x}}(t) + B\tilde{\boldsymbol{u}}(t), \quad \tilde{y} = C\tilde{\boldsymbol{x}}(t) + D\tilde{\boldsymbol{u}}(t)$$

für die Kleinsignalgrößen

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} - \bar{\boldsymbol{x}}, \quad \tilde{\boldsymbol{y}} = \boldsymbol{y} - \bar{\boldsymbol{y}}, \quad \tilde{\boldsymbol{u}} = \boldsymbol{u}$$

zurückgegeben werden. Prüfen Sie das Ergebnis der Linearisierung mit Hilfe der Methode `verify_linearization` der Klasse `Gantry` (Abschnitt 1.2.2 des Notebooks).

Simulieren Sie das erhaltene System für verschiedene für die Schlittenposition vorgegebene Trajektorien. Nutzen Sie dazu Abschnitt 1.1.1. des zur Verfügung gestellten Notebooks.

Aufgabe 2 (Steuerbarkeit und Reglerentwurf)

a) Prüfen Sie die Steuerbarkeit des Systems für die Ruhelagen

	1	2	3	4
x_1	0.0 m	0.0 m	0.8 m	0.8 m
x_2	0.0 m	0.3 m	0.0 m	0.3 m

mit dem Kalmanschen Kriterium. Geben Sie im Falle der Steuerbarkeit jeweils die möglichen Kronecker- Steuerbarkeits-Indizes an. Nutzen Sie dazu die bereits vorgegebene Funktion `controllability_matrix` aus der Datei `mimo.py` und vervollständigen Sie die Funktion `kronecker` in der selben Datei entsprechend.

Überlegen Sie, welche Probleme und Herausforderungen sich für den Steuerungs- und Regelungsentwurf aus den ermittelten Resultaten ableiten lassen.

In den folgenden Teilaufgaben wird das in der Ruhelage 4 linearisierte System als lineares System bezeichnet.

b) Das System soll für die Ruhelage 4 und die Kronecker-Indizes $n_1 = 4$, $n_2 = 6$ auf Regelungsnormalform transformiert und (numerisch) die Parametrierung aller Systemgrößen durch den flachen Ausgang $\boldsymbol{\eta}$ bestimmt werden. Dazu steht die Klasse `ContinuousFlatnessBasedTrajectory` zur Verfügung. Eine Instanz dieser Klasse wird durch den Aufruf der Methode `rest_to_rest_trajectory` erzeugt. Diese gehört zur Klasse `ContinuousLinearizedSystem`.

Vervollständigen Sie im Konstruktor der Klasse `ContinuousFlatnessBasedTrajectory` in der Datei `pysim.py` insbesondere die Vorschrift für die Umrechnung zwischen den stationären Werten `eta_a` und `eta_b` des flachen Ausgangs und `ya` und `yb` des Systemausgangs.

Vervollständigen Sie weiterhin die Vorschriften zur Berechnung des Zustands aus der Trajektorie des flachen Ausgangs in der Methode `state` der selben Klasse. Orientieren Sie sich gegebenenfalls an der Methode `input` mit der der Eingang berechnet wird.

Machen Sie sich auch mit dem übrigen Teil des in der Klasse implementierten Codes vertraut und versuchen Sie die Algorithmen nachzuvollziehen.

c) Planen Sie für das lineare System flachheitsbasiert, also auf Basis der Regelungsnormalform, eine Trajektorie, die das System in der Zeit $T = 1.0\,\mathrm{s}$ aus der Ruhelage 2 in die Ruhelage 4 überführt.

Simulieren Sie dieses Szenario mit dem in der linearisierten und dem nichtlinearen Modell (Abschnitt 2.1 des Notebooks).

d) Berechnen Sie für das lineare System mit der Ackermannformel für Mehrgrößensysteme eine Zustandsrückführung $\tilde{\boldsymbol{u}} = -K\tilde{\boldsymbol{x}}$. Verschieben Sie dazu alle reellen Eigenwerte des offenen Regelkreises um -10 nach links und verändern Sie die komplexen Eigenwerte derart, dass Sie im zugehörigen harmonischen Oszillator eine Dämpfung $d = 0.5$ einführen. Wählen Sie dafür ebenfalls die die Kroneckerindizes $n_1 = 4$, $n_2 = 6$. Der Aufruf der bereits vorimplementierten Ackermannformel erfolgt über die Methode `acker`

der Klasse `LinearizedSystem`, von der auch `ContinuousLinearizedSystem` abgeleitet ist.

Simulieren Sie dieses Szenario mit dem linearen und dem nichtlinearen Modell (Abschnitt 2.2 des Notebooks). Vervollständigen dazu vorher die Methode `output` der Klasse `LinearStateFeedback` in der Datei `pysim.py`.

e) Bestimmen Sie die Rückführverstärkungen nun alternativ mit Hilfe des quadratischen Gütefunktionals

$$J(\boldsymbol{x}, \boldsymbol{u}) = \int_0^\infty \left(\boldsymbol{x}^T(t) Q \boldsymbol{x}(t) + \boldsymbol{u}^T(t) R \boldsymbol{u}(t) \right) dt.$$

Bestimmen Sie die Werte von Q derart, dass gilt

$$J(\boldsymbol{x}, \boldsymbol{u}) = \int_0^\infty 100(y_1(t)^2 + y_2(t)^2) + u_2(t)^2 + u_2(t)^2 \, \mathrm{d}t.$$

Passen Sie außerdem die Methode `lqr` der Klasse `ContinuousLinearizedSystem` derart an, dass aus den übergebenen Matrizen Q und R die gesuchten Reglerverstärkungen berechnet werden.

Simulieren Sie das Szenario mit dem linearen und dem nichtlinearen Modell (Abschnitt 2.3 des Notebooks).

Aufgabe 3 (Beobachterentwurf)

Die entworfene Zustandsrückführung erfordert die Kenntnis des vollständigen Systemzustands.

a) Machen Sie sich klar, dass der Zustand $\boldsymbol{x}$ des gegebenen Systems aus den Messgrößen $y_{\mathrm{m},1}$ und $y_{\mathrm{m},2}$ nicht rekonstruiert werden kann.

Da die Schwingungen des Gestells aus den Messungen der Relativ-Position des Schlittens in Bezug auf den Rahmen nicht rekonstruiert werden können, muss dass betrachtete System mit weiteren Sensoren ausgestattet werden. Eine einfache Möglichkeit besteht in der Nutzung einer sogenannten inertialen Messeinheit (*engl.* inertial measurement unit, IMU). Eine IMU kann neben den Linear-Beschleunigungen in allen Richtungen auch die Winkelgeschwindigkeiten messen. Nachfolgend wird angenommen, dass die IMU die Beschleunigungen $\bar{u}_1 = \ddot{z}_1$ und $\bar{u}_2 = \ddot{z}_2$ sowie die Winkelgeschwindigkeit $y_{\mathrm{m},3} = \dot{\varphi}$ liefert.

b) Leiten Sie die Messgleichungen für die IMU auf der Basis des nichtlinearen Modells her und implementieren Sie diese in den Methoden `_gyro_frame_scalar` (Winkelgeschwindigkeit $y_{\mathrm{m},3}$) und `_accell_frame_scalar` (Beschleunigungen $\bar{u}_1$ und $\bar{u}_2$) der Klasse `Gantry`. Sie müssen an dieser Stelle lediglich Argumente auswerten, die zu einem bestimmten Zeitpunkt gegeben sind, d.h. $t \in \mathbb{R}$ und $\boldsymbol{x} \in \mathbb{R}^{10}$.

   Prüfen Sie ihre Implementierung, indem Sie sie mit den entsprechenden Differenzenquotienten vergleichen.

Mit den zusätzlichen Messgrößen soll nun ein Beobachter entworfen werden. Dazu sollen zunächst der originale Eingang $\boldsymbol{u}$ aus dem System eliminiert und stattdessen die gemessenen Beschleunigungen $\bar{u}_1$ und $\bar{u}_2$ als neue Eingangsgrößen verwendet werden. Dies bietet den Vorteil, dass die Aufgrund von Reibung und möglichen Fehlern im unterlagerten Motormodell nur ungenau einstellbaren Beschleunigungen $u_1 = \ddot{x}_1$ und $u_2 = \ddot{x}_2$ im Beobachter nicht benötigt werden.

c) Leiten Sie ein nichtlineares Zustandsraummodell mit dem Zustand

$$\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ z_1 \\ z_2 \\ \varphi \\ \dot{x}_1 \\ \dot{x}_2 \\ \dot{z}_1 \\ \dot{z}_2 \\ \dot{\varphi} \end{bmatrix}$$

   und dem Eingang $\bar{\boldsymbol{u}} = (\bar{u}_1, \bar{u}_2)^T$ her und vervollständigen Sie die Methode `model` der Klasse `GantryObserverModel` entsprechend.

   Prüfen Sie ihre Implementierung, indem Sie die Sensorwerte $\bar{u}_1$ und $\bar{u}_2$ des ursprünglichen Modells als Eingang des modifizierten Modells nutzen und die Ausgangsgrößen miteinander vergleichen.

d) Linearisieren Sie die erhaltenen Gleichungen und ergänzen Sie die Methode `linearize` der Klasse `GantryObserverModel` entsprechend.

e) Prüfen Sie die Beobachtbarkeit des modifizierten Systems für jede der Ruhelagen

	1	2	3	4
x_1	0.0 m	0.0 m	0.8 m	0.8 m
x_2	0.0 m	0.3 m	0.0 m	0.3 m

Gehen Sie einerseits vom der Messung $(y_{\mathrm{m},1}, y_{\mathrm{m},2})^T$ und andererseits von der Messung $(y_{\mathrm{m},1}, y_{\mathrm{m},2}, y_{\mathrm{m},3})^T$ aus.

Nutzen Sie dazu die bereits vorgegebene Funktion `controllability_matrix` aus der Datei `mimo.py`.

Welche Schlussfolgerungen lassen sich in Bezug auf die Umsetzbarkeit des im vergangenen Abschnitt entworfenen Regelungskonzeptes ziehen?

f) Gehen Sie nun von der Messung $(y_{\mathrm{m},1}, y_{\mathrm{m},2}, y_{\mathrm{m},3})^T$ aus und bestimmen Sie für alle angegebenen Ruhelagen die Kronecker-Indizes. Nutzen Sie dazu die bereits implementierte Funktion `kronecker` zur Bestimmung der Kronecker-Steuerbarkeits-Indizes.

g) Entwerfen Sie auf der Basis des um die Ruhelage 1 linearisierten Modells einen Beobachter. Dazu wird das nichtlineare Modell als Simulator genutzt. Die Beobachterverstärkungen werden per Eigenwertvorgabe auf der Basis des um die Ruhelage 1 linearisierten Modells mit den Kronecker-Indizes $(4, 4, 2)$ bestimmt. Implementieren Sie den Beobachter, indem Sie die Methode `model` der Klasse `GantryObserver` entsprechend vervollständingen.

h) Testen Sie den Beobachter im offenen und geschlossenen Regelkreis anhand der Szenarien aus Aufgabe 2. Spielen Sie auch mit Modellfehlern, indem Sie die Parameter des Simulationsmodells etwas variieren.