Optimale Steuerung und Regelung

2. Übung, ausgegeben von: Prof. Frank Woittennek
zuletzt geändert am: 5. Juni 2023

Der Lösungsweg sowie der Python-Code sind am Ende der Übungseinheit zur Bewertung abzugeben. Der Lösungsweg soll mit einem geeignetes Textsatzsystem wie LibreOffice Writer oder Latex dokumentiert werden und ist als PDF-Datei abzugeben. Sie können den Lösungsweg auch zusammen mit dem Code in Form eines Jupyter-Notebooks dokumentieren und als HTML-Datei + IPYNB-Datei abgeben. Während der Übung haben Sie die Möglichkeit die verbleibenden Fragen und Probleme zu klären bzw. zu lösen und Ihre Dokumentation entsprechend anzupassen.

Aufgabe 1

Es wird das Wagen-Pendel-System aus Abb. 1 betrachtet, vergleiche [Ste20, Übung 3]. Die

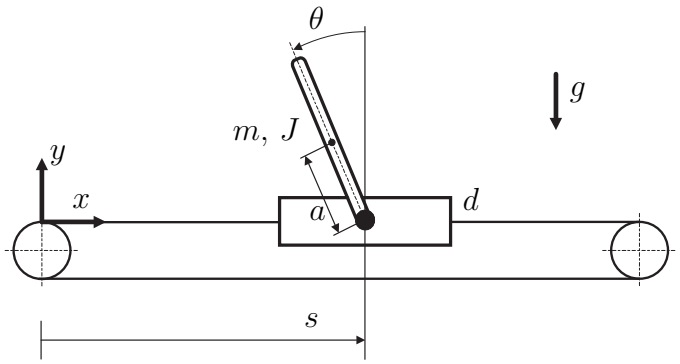

Abb. 1 – Wagen-Pendel-System (Abbildung aus [Ste20])

Zustandsdarstellung für den Zustand $\boldsymbol{x}^\top = (\theta, \omega, s, v)$ lautet

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), u(t)) = \begin{pmatrix} \omega(t) \\ \frac{m\,g\,a\,\sin\theta(t) - d\,\omega(t) + m\,a\,u(t)\,\cos\theta(t)}{J + m\,a^2} \\ v(t) \\ u(t) \end{pmatrix}, \tag{1}$$

mit der Wagenposition s, der Wagengeschwindigkeit v, dem Pendelwinkel θ, der Pendelgeschwindigkeit ω, dem Massenträgheitsmoment $J = 0.0361\,\mathrm{kg\,m^2}$, dem Schwerpunktabstand $a = 0.42\,\mathrm{m}$ und der Masse $m = 0.3553\,\mathrm{kg}$ des Pendelstabes sowie der Erdbeschleunigung $g = 9.81\,\mathrm{m\,s^{-2}}$ und dem Reibungskoeffizienten $d = 0.005\,\mathrm{N\,m\,s}$. Die Stellgröße u ist als die Beschleunigung des Wagens $u = \ddot{s} = \dot{v}$ festgelegt .

Für dieses System soll das Optimalsteuerungsproblem

$$\min_{u(\cdot)} \quad J(u(\cdot)) = \varphi(\boldsymbol{x}(t_1)) + \int_{t_0}^{t_1} l(\boldsymbol{x}(t), u(t))\,\mathrm{d}t$$

$$\text{u.B.v.} \quad \dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}, u), \qquad \boldsymbol{x}(t_0) = \boldsymbol{x}_0, \qquad \boldsymbol{\psi}(\boldsymbol{x}(t_1)) = 0$$

mit den Endkosten

$$\varphi(\boldsymbol{x}(t_1)) = \frac{1}{2}\boldsymbol{x}^\top(t_1)\boldsymbol{S}\boldsymbol{x}(t_1), \qquad \boldsymbol{S} \geq 0,$$

den integralen Kosten

$$l(\boldsymbol{x}, u) = 1 + \frac{1}{2}\left(\boldsymbol{x}^\top \boldsymbol{Q}\boldsymbol{x} + Ru^2\right), \qquad \boldsymbol{Q}, R > 0,$$

der Endbedingung

$$\boldsymbol{\psi}(\boldsymbol{x}(t_1)) = \boldsymbol{M}\boldsymbol{x}(t_1) = 0 \tag{2}$$

und der Anfangsbedingung

$$\boldsymbol{x}_0 = (\pi, 0, -0.5, 0)^\top \tag{3}$$

gelöst werden. Die Gewichtungsmatrizen sind mit

$$\boldsymbol{M} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \boldsymbol{Q} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \quad \boldsymbol{S} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad R = 0.1$$

gegeben.

Zur Lösung soll das **direkte Verfahren** der **Volldiskretisierung** zum Einsatz kommen. Lesen Sie sich dafür [Gra19, Abschnitt 5.7] durch. Bei diesem Verfahren werden die Trajektorien zu N diskreten Zeitpunkten

$$t_0 = t^1 < t^2 < \cdots < t^N = t_1$$

ausgewertet, d.h.

$$\boldsymbol{x}^i = \boldsymbol{x}(t^i), \qquad u^i = u(t^i), \qquad i = 1, ..., N.$$

Die diskretisierten Trajektorien sind also die Freiheitsgrade des Steuerungsproblems und werden in dem zu optimierenden Vektor

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{x}^1 \\ u^1 \\ \vdots \\ \boldsymbol{x}^N \\ u^N \end{pmatrix}$$

zusammengefasst. Der Wert des Kostenfunktionals kann z.B. mit der Trapezregel angenähert werden:

$$J_\mathrm{d}(\boldsymbol{y}) = \varphi\left(t^N, \boldsymbol{x}^N\right) + \sum_{i=1}^{N-1} \frac{1}{2}\left(t^{i+1} - t^i\right)\left[l\left(\boldsymbol{x}^i, u^i\right) + l\left(\boldsymbol{x}^{i+1}, u^{i+1}\right)\right].$$

Die Gleichungsnebenbedingungen $\boldsymbol{g}_\mathrm{d}(\boldsymbol{y})$ setzen sich aus der Anfangsbedingung (3), der Endbedingung (2) und der Systemgleichung (1) zusammen, wobei zur Berückisichtigung der

Systemgleichung, diese ebenfalls diskretisiert werden muss. Dazu bietet sich z.B. die implizite Trapezregel

$$\boldsymbol{x}^{i+1} - \boldsymbol{x}^i - \frac{t^{i+1} - t^i}{2}\left[\boldsymbol{f}(\boldsymbol{x}^i, u^i) + \boldsymbol{f}(\boldsymbol{x}^{i+1}, u^{i+1})\right] = 0, \qquad i = 1, ..., N-1$$

an. Das verbleibende Optimalsteuerungsproblem kann dann in der Form

$$\begin{aligned} \min_{\boldsymbol{y}\in\mathbb{R}^{5N}} \quad & J_\mathrm{d}(\boldsymbol{y}) \\ \text{u.B.v.} \quad & \boldsymbol{g}_\mathrm{d}(\boldsymbol{y}) = 0 \end{aligned} \tag{4}$$

angeschrieben werden.

Die folgenden Aufgaben sollen für $t_0 = 0\,\mathrm{s}$, $t_1 = 2.5\,\mathrm{s}$ und $N = 50$ bearbeitet werden.

a) Implementieren Sie die Funktionen J_d, $\boldsymbol{g}_\mathrm{d}$ und die Jacobimatrizen $\frac{\partial J_\mathrm{d}}{\partial \boldsymbol{y}}$, $\frac{\partial \boldsymbol{g}_\mathrm{d}}{\partial \boldsymbol{y}}$. Achten Sie darauf, dass die Parameter des Systems, des Optimalsteuerungsproblems sowie der Diskretisierung nachträglich noch angepasst werden können.

b) Wie viele Diskretisierungspunkte N muss man mindestens wählen damit das diskretisierte Problem (4) stets eine Lösung besitzt?

c) Finden Sie eine Lösung $\boldsymbol{y}^*$ des Problems (4) mittels **sequentieller quadratischer Programmierung** [Gra19, Abschnitt 4.5.1]. Verwenden Sie dafür die Funktion `minimize` aus dem SciPy-Modul `scipy.optimize` (mit Schlüsselwortargument: `method="SLSQP"`). Achten Sie darauf, aktuelle Programmbibliotheken zu verwenden, z.b. `scipy==1.4.0`, `numpy==1.15.2` und `sympy==1.5` (SymPy ist optional). Erhöhen Sie gegebenenfalls die Anzahl der maximal zulässigen Iterationen des Optimierungsalgorithmus.

d) Bestimmen Sie nun noch eine weitere Lösung so, dass die Eingangsgrößen- und Zustandsgrößenbeschränkungen

$$\begin{aligned} -12\,\mathrm{m\,s^{-2}} \le u(t) \le 12\,\mathrm{m\,s^{-2}} \qquad & \forall \quad t \in [t_0, t_1] \\ -0.8\,\mathrm{m} \le s(t) \le 0.8\,\mathrm{m} \qquad & \forall \quad t \in [t_0, t_1] \end{aligned}$$

nicht verletzt werden. Lesen Sie dafür die Dokumentation der Funktion `minimize`.

e) Schreiben Sie eine Funktion die das Tupel (t_0, t_1) und die Lösung $\boldsymbol{y}^*$ als Argumente erhält und das System (1) ausgehend von $\boldsymbol{x}_0$ mit dem optimalen Eingangsgrößenverlauf von $t = t_0$ bis $t = t_1$ simuliert. Zum Vergleich der Zeitverläufe der simulierten Zustandsgrößen mit den optimierten Zustandsgrößen aus $\boldsymbol{y}^*$ soll diese Funktion außerdem einen Plot erzeugen. Für die Simulation kann die Funktion `solve_ivp` aus dem SciPy-Modul `scipy.integrate` verwendet werden.

Literatur

[Gra19] K. Graichen. *Skript zur Vorlesung: Numerische Optimierung und modellprädiktive Regelung*. Lehrstuhl für Regelungstechnik, Friedrich-Alexander-Universität Erlangen-Nürnberg, Sommersemester 2019.

[Ste20] A. Steinböck. *Skript zur Vorlesung: Optimierung*. Institut für Automatisierungs- und Regelungstechnik, Technische Universität Wien, Wintersemester 2019/2020. URL: https://www.acin.tuwien.ac.at/file/teaching/master/optimierung/WS2019/Optimierung_VO.pdf.