3 Lösung von Optimalsteuerungsproblemen mit direkten Verfahren

Zur Vorbereitung auf den ersten Teil der Übung (Besprechung und Fragen zur Übung) wird empfohlen, sich mit den im Folgenden vorgestellten theoretischen Grundlagen sowie dem betrachteten Beispiel vertraut zu machen. Darüber hinaus wird empfohlen, die Aufgaben 1. und 2. zu lösen und sich grundsätzliche Gedanken zur Umsetzung der vorgestellten Theorie in eine MATLAB-Implementierung zu machen.

Numerische Verfahren zur Lösung von Optimalsteuerungsproblemen der Form

$$\min_{\mathbf{u}(\cdot)} \quad J(\mathbf{u}(\cdot)) = \varphi(t_1, \mathbf{x}(t_1)) + \int_{t_0}^{t_1} l(t, \mathbf{x}(t), \mathbf{u}(t)) \, \mathrm{d}t \tag{3.1a}$$

$$\text{u.B.v.} \quad \dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}, \mathbf{u}), \qquad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{3.1b}$$

$$\boldsymbol{\psi}(t_1, \mathbf{x}(t_1)) = \mathbf{0} \tag{3.1c}$$

$$\mathbf{h}(t, \mathbf{x}(t), \mathbf{u}(t)) \leq \mathbf{0}, \qquad \forall t \in [t_0, t_1] \tag{3.1d}$$

werden häufig in *direkte* und *indirekte* Verfahren unterteilt. Indirekte Verfahren basieren auf der Lösung der notwendigen Optimalitätsbedingungen. Diese liegen typischerweise in Form eines Randwertproblems, das z. B. aus dem Minimumsprinzip von Pontryagin resultiert, vor.

Direkte Verfahren beruhen auf einer Diskretisierung von (3.1). Diese geschieht üblicherweise mit einer Parametrierung des Zustandes $\mathbf{x}(t)$ und/oder der Stellgrößen $\mathbf{u}(t)$ im Sinne einer Beschreibung der zeitlichen Verläufe mit endlich vielen Parametern. Dadurch wird schlussendlich das unendlich-dimensionale Optimalsteuerungsproblem (3.1) auf ein endlich-dimensionales statisches Optimierungsproblem reduziert, das mit Verfahren aus der (beschränkten) statischen Optimierung gelöst werden kann.

Bei den direkten Verfahren wird häufig zwischen *Teil-* und *Volldiskretisierung* unterschieden. Im Folgenden soll näher auf die Volldiskretisierung eingegangen werden. Dazu betrachte man die Diskretisierung des Zeitintervalls $[t_0, t_1]$ in Form des *Zeitgitters*

$$t_0 = t^1 < t^2 < \cdots < t^N = t_1 \tag{3.2}$$

mit N als Anzahl der Stützstellen. Basierend auf diesem Zeitgitter wird eine Parametrierung der Zustands- und Stellgrößen mit endlich vielen Parametern

$$\tilde{\mathbf{x}}^{\mathrm{T}} = \begin{bmatrix} \left(\mathbf{x}^1\right)^{\mathrm{T}} & \left(\mathbf{x}^2\right)^{\mathrm{T}} & \cdots & \left(\mathbf{x}^N\right)^{\mathrm{T}} \end{bmatrix} \qquad \tilde{\mathbf{u}}^{\mathrm{T}} = \begin{bmatrix} \left(\mathbf{u}^1\right)^{\mathrm{T}} & \left(\mathbf{u}^2\right)^{\mathrm{T}} & \cdots & \left(\mathbf{u}^N\right)^{\mathrm{T}} \end{bmatrix} \tag{3.3}$$

vorgenommen. Im Weiteren soll gelten $\mathbf{u}^i = \mathbf{u}(t^i)$, $\mathbf{x}^i = \mathbf{x}(t^i)$, $i = 1, 2, \ldots, N$, siehe Abbildung 3.1. Die Parameter (3.3) stellen die Freiheitsgrade dar und werden zu einem

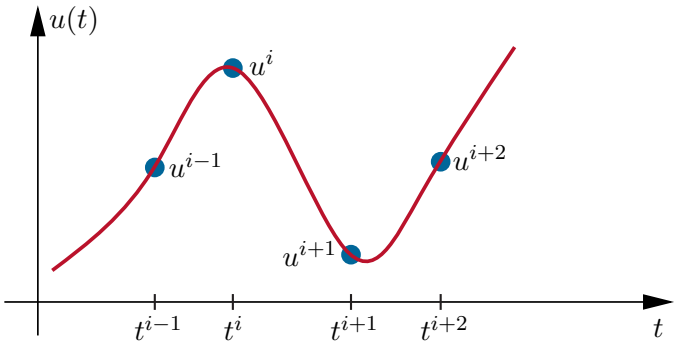

Abbildung 3.1: Veranschaulichung der Parametrierung der Stellgröße $u(t)$.

Vektor der Optimierungsvariablen

$$\mathbf{y} = \begin{bmatrix} \mathbf{x}^1 \\ \mathbf{u}^1 \\ \mathbf{x}^2 \\ \mathbf{u}^2 \\ \vdots \\ \mathbf{x}^N \\ \mathbf{u}^N \end{bmatrix} \tag{3.4}$$

zusammengefasst[1]. Damit kann das Optimalsteuerungsproblem (3.1) in ein statisches Optimierungsproblem überführt werden. Der Wert des Kostenfunktionals (3.1a) wird mit der *Trapezregel* in der Form

$$J_d(\mathbf{y}) = \varphi\left(t^N, \mathbf{x}^N\right) + \sum_{i=1}^{N-1} \frac{1}{2}\left(t^{i+1} - t^i\right)\left[l\left(t^i, \mathbf{x}^i, \mathbf{u}^i\right) + l\left(t^{i+1}, \mathbf{x}^{i+1}, \mathbf{u}^{i+1}\right)\right] \tag{3.5}$$

angenähert. Für die Ungleichungsbeschränkungen (3.1d) wird vereinfachend gefordert, dass sie lediglich an den Gitterpunkten erfüllt sind. Damit folgt

$$\mathbf{h}\left(t^i, \mathbf{x}^i, \mathbf{u}^i\right) \leq \mathbf{0}, \qquad i = 1, 2, \ldots, N \tag{3.6}$$

und für die Endbedingungen (3.1c) erhält man

$$\boldsymbol{\psi}\left(t^N, \mathbf{x}^N\right) = \mathbf{0}\ . \tag{3.7}$$

Eine Diskretisierung des Differenzialgleichungssystems (3.1b) mit der *impliziten Trapezregel* liefert die algebraischen Gleichungsbedingungen

$$\mathbf{x}^{i+1} - \mathbf{x}^i - \frac{t^{i+1} - t^i}{2}\left[\mathbf{f}\left(t^i, \mathbf{x}^i, \mathbf{u}^i\right) + \mathbf{f}\left(t^{i+1}, \mathbf{x}^{i+1}, \mathbf{u}^{i+1}\right)\right] = \mathbf{0}, \qquad i = 1, 2, \ldots, N-1 \tag{3.8a}$$

[1] Dies zeichnet die Volldiskretisierung aus. Bei der Teildiskretisierung werden lediglich die Parameter $\mathbf{u}^1, \mathbf{u}^2, \ldots, \mathbf{u}^N$ für die Stellgrößen in den Vektor der Optimierungsvariablen aufgenommen.

©, Institut für Automatisierungs- und Regelungstechnik, TU Wien

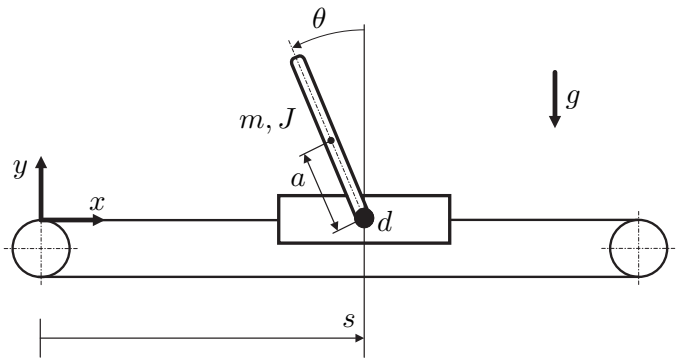

Abbildung 3.2: Wagen-Pendel-System.

mit der zusätzlichen Restriktion

$$\mathbf{x}^1 = \mathbf{x}_0 \ . \tag{3.8b}$$

Aus den bisherigen Ergebnissen folgt das zum Optimalsteuerungsproblem (3.1) zugehörige statische Optimierungsproblem zu

$$\min_{\mathbf{y}} \quad J_d(\mathbf{y}) \tag{3.9a}$$

$$\text{u.B.v.} \quad \mathbf{h}_d(\mathbf{y}) \leq \mathbf{0} \tag{3.9b}$$

$$\mathbf{g}_d(\mathbf{y}) = \mathbf{0} \tag{3.9c}$$

wobei $\mathbf{h}_d(\mathbf{y})$ aus (3.6) und $\mathbf{g}_d(\mathbf{y})$ aus (3.7) und (3.8) resultieren.

Die Berechnung einer optimalen Steuerung mittels der gezeigten Methode der Volldiskretisierung soll am Beispiel des *Wagen-Pendel-Systems* gemäß Abbildung 3.2 durchgeführt werden. Dieses System besteht aus einem in x-Richtung verschiebbaren Wagen, an dem frei drehbar ein Pendelstab befestigt ist. Die zugehörigen Systemgleichungen lauten

$$\dot{\mathbf{x}} = \frac{\mathrm{d}}{\mathrm{d}t}\begin{bmatrix}\theta\\ \omega\\ s\\ v\end{bmatrix} = \underbrace{\begin{bmatrix}\omega\\ \frac{mga\sin(\theta) - d\omega + ma\cos(\theta)u}{J+ma^2}\\ v\\ u\end{bmatrix}}_{\mathbf{f}(t,\mathbf{x},u)} \tag{3.10}$$

mit der Wagenposition s, dem Pendelwinkel θ, dem Massenträgheitsmoment $J = 0.0361\,\mathrm{kg\,m^2}$, dem Schwerpunktsabstand $a = 0.42\,\mathrm{m}$ 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 durch die Beschleunigung des Wagens $u = \ddot{s} = \dot{v}$ gegeben.

©, Institut für Automatisierungs- und Regelungstechnik, TU Wien

Das zugrundeliegende Optimalsteuerungsproblem besitze die Form

$$\min_{u(\cdot)} \quad J(u(\cdot)) = \varphi(t_1, \mathbf{x}(t_1)) + \int_{t_0}^{t_1} l(t, \mathbf{x}(t), u(t)) \, \mathrm{d}t \tag{3.11a}$$

$$\text{u.B.v.} \quad \dot{\mathbf{x}} = \mathbf{f}(t, \mathbf{x}, u), \qquad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{3.11b}$$

$$\boldsymbol{\psi}(t_1, \mathbf{x}(t_1)) = \mathbf{0} \tag{3.11c}$$

mit

$$\varphi(t_1, \mathbf{x}(t_1)) = \frac{1}{2}\mathbf{x}^{\mathrm{T}}(t_1)\mathbf{S}\mathbf{x}(t_1) \qquad \mathbf{S} \geq \mathbf{0} \tag{3.12a}$$

$$l(t, \mathbf{x}, u) = 1 + \frac{1}{2}\left(\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} + Ru^2\right), \quad \mathbf{Q} > \mathbf{0}, \quad R > 0 \, , \tag{3.12b}$$

den Systemgleichungen gemäß (3.10) und den Endbedingungen

$$\boldsymbol{\psi}(t_1, \mathbf{x}(t_1)) = \mathbf{M}\mathbf{x}(t_1) - \mathbf{b} \tag{3.12c}$$

mit $\mathbf{M} \in \mathbb{R}^{q \times 4}$ und $q \leq 4$. Die Matrizen der linearen Endbeschränkung (3.12c) und die Gewichtungsmatrizen in (3.12a), (3.12b) lauten

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{Q} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{S} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.13a}$$

$$\mathbf{b} = \mathbf{0} \qquad R = 0.1 \, . \tag{3.13b}$$

Bearbeiten Sie folgende Aufgaben:

1. Berechnen Sie die Jacobimatrizen $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(t, \mathbf{x}, u)$, $\frac{\partial \mathbf{f}}{\partial u}(t, \mathbf{x}, u)$, $\frac{\partial \varphi}{\partial \mathbf{x}}(t, \mathbf{x})$, $\frac{\partial l}{\partial \mathbf{x}}(t, \mathbf{x}, u)$ und $\frac{\partial l}{\partial u}(t, \mathbf{x}, u)$ der Funktionen $\mathbf{f}$, φ und l aus (3.10) und (3.12a), (3.12b). Ermitteln Sie weiters den Gradienten $\left(\frac{\partial J_d}{\partial \mathbf{y}}\right)^{\mathrm{T}}(\mathbf{y})$ der Kostenfunktion (3.5) in Abhängigkeit der Optimierungsvariablen $\mathbf{y}$ gemäß (3.4).

2. Schreiben Sie die im Folgenden aufgelisteten Funktionen in MATLAB und beachten Sie dazu die nachstehenden Hinweise.

Für diese Übung stehen auf der Homepage des Instituts vorbereitete MATLAB-Dateien im zip-Archiv `uebung3.zip` zum Download bereit. Darin sind Vorlagen für alle zu implementierenden Funktionen enthalten sowie Funktionen zum Überprüfen Ihrer Ergebnisse.

Hinweis:

- Achten Sie auf eine allgemeine Implementierung, sodass Sie die Parameterwerte des Systems und des Optimalsteuerungsproblems leicht ändern können.
- Im Sinne einer einfachen und schnellen Implementierung können Sie Va-

©, Institut für Automatisierungs- und Regelungstechnik, TU Wien

riablen, die in mehreren Funktionen benötigt werden, mittels des Schlüsselworts `global` in jeder betreffenden Funktion als globale Variablen deklarieren und damit direkt darauf zugreifen. Alternativ können Sie diese Variablen auch z. B. in Form eines *Struct Arrays* nach den jeweils angegebenen Funktionsargumenten übergeben.

- Achten Sie in Ihrer Implementierung darauf, die Anzahl der Diskretisierungsstützstellen N variabel zu belassen. Berücksichtigen Sie weiters, dass das Zeitgitter (3.2) nicht zwingenderweise äquidistant sein muss, d. h. es darf $t^{i+1} - t^i \neq \text{konst.}$, $i = 1, 2, \ldots, N-1$, gelten. Um dies umzusetzen können Sie z. B. die Zeitpunkte $t^1, t^2, \ldots, t^N$ in einem Vektor speichern und entweder als globale Variable oder als Feld eines *Struct Arrays* den jeweiligen Funktionen (z. B. `costFct`) zur Verfügung stellen.

- `[f,dfdx,dfdu] = calcf(t,x,u)`: Berechnung der rechten Seite der Systemgleichungen gemäß (3.10) und der Jacobimatrizen $\frac{\partial \mathbf{f}}{\partial \mathbf{x}}$ und $\frac{\partial \mathbf{f}}{\partial u}$.
- `[phi,dphidx] = calcphi(t,x)`: Berechnung der Endkosten gemäß (3.12a) und der Jacobimatrix $\frac{\partial \varphi}{\partial \mathbf{x}}$.
- `[l,dldx,dldu] = calcl(t,x,u)`: Berechnung der Integralkosten gemäß (3.12b) und der Jacobimatrizen $\frac{\partial l}{\partial \mathbf{x}}$ und $\frac{\partial l}{\partial u}$.
- `[Jd,dJdy] = costFct(y)`: Berechnung des Wertes der Kostenfunktion (3.5) in Abhängigkeit der Optimierungsvariablen $\mathbf{y}$ gemäß (3.4). Zusätzlich soll auch der Gradient $\left(\frac{\partial J_d}{\partial \mathbf{y}}\right)^{\mathrm{T}}(\mathbf{y})$ in Form der Variable `dJdy` retourniert werden.

  Hinweis: Überprüfen Sie die Korrektheit Ihrer Implementierung durch Vergleich mit der zur Verfügung gestellten Funktion `costFctVer.p`. Diese Funktion verwendet intern ein äquidistantes Zeitgitter mit $t_0 = 0\,\mathrm{s}$, $t_1 = 4\,\mathrm{s}$ und $N = 50$ Diskretisierungspunkten. Sie akzeptiert ein Argument in Form der Optimierungsvariablen $\mathbf{y}$ gemäß (3.4). Vergleichen Sie die Rückgabewerte Ihrer Funktion mit denen von `costFctVer.p` für mehrere verschiedene Werte der Optimierungsvariablen $\mathbf{y}$.

Folgende Aufgaben sollen für den zweiten Teil der Übung (Übungseinheit, Besprechung der Ergebnisse) gelöst werden:

3. Das beschränkte statische Optimierungsproblem (3.9) soll in MATLAB implementiert und mittels der Funktion `fmincon` gelöst werden.

   a) Für `fmincon` wird dazu neben `costFct` eine Funktion `[hd,gd] = constrFct(y)` benötigt, die die Werte der Funktionen $\mathbf{h}_d(\mathbf{y})$ bzw. $\mathbf{g}_d(\mathbf{y})$ in Form der Variablen `hd` und `gd` berechnet.

   Hinweis: Überprüfen Sie die Korrektheit Ihrer Implementierung durch Vergleich mit der zur Verfügung gestellten Funktion `constrFctVer.p`. Diese Funktion akzeptiert als Argument nur die Optimierungsvariable

> `y` gemäß (3.4) mit der selben Diskretisierung wie `costFctVer.p`. Die Systemparameter sind wie oben angegeben angenommen. Vergleichen Sie die Rückgabewerte Ihrer Funktion mit denen von `constrFctVer.p` für mehrere verschiedene Werte der Optimierungsvariablen `y`.

b) Lösen Sie das Optimierungsproblem (3.9) mit `fmincon`. Verwenden Sie dafür $t_0 = 0\,\mathrm{s}$, $t_1 = 4\,\mathrm{s}$ und ein äquidistantes Zeitgitter mit $N = 50$ Punkten. Die Anfangsbedingung lautet $\mathbf{x}_0 = \begin{bmatrix}\pi & 0 & 0 & 0\end{bmatrix}^{\mathrm{T}}$.

> **Hinweis:** Verwenden Sie in `fmincon` den Algorithmus `interior-point` und erhöhen Sie die maximale Anzahl der Funktionsauswertungen mit der Option `MaxFunctionEvaluations`.

c) Stellen Sie ihre Lösung mithilfe der vorhandenen Funktion `plotsol(y,t_1)` mit der Lösung des Optimierungsproblems `y` $\in \mathbb{R}^{5N}$ und der Endzeit `t_1`.

4. Experimentieren Sie weiter mit `fmincon` um eine Aufschwingtrajektorie zu finden, welche am realen Versuchsaufbau realisiert werden kann.

a) Berücksichtigen Sie dabei durch zusätzliche Box-Beschränkungen, dass die Wagenposition $s(t)$ und die Beschleunigung des Wagens $u(t)$ physikalisch gemäß

$$\begin{aligned} -0.7\,\mathrm{m} \leq s(t) \leq 0.7\,\mathrm{m}, &\quad \forall t \in [t_0, t_1] \\ -12\,\frac{\mathrm{m}}{\mathrm{s}^2} \leq u(t) \leq 12\,\frac{\mathrm{m}}{\mathrm{s}^2}, &\quad \forall t \in [t_0, t_1] \end{aligned}$$

beschränkt sind. Passen Sie gegebenenfalls die Anfangs- und Endbedingungen, Endzeit sowie die Gewichtungsmatrizen des Optimierungsproblems an um diese Bedingungen einhalten zu können.

b) Speichern Sie die gefundene Trajektorie mithilfe des MATLAB-Befehls `save` in eine Datei `trajectory.mat`. Speichern Sie dabei die Lösung in folgendem Format

$$\begin{aligned} \mathtt{t} &= \begin{bmatrix} t^1 & \dots & t^N \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^N \\ \mathtt{u} &= \begin{bmatrix} u(t^1) & u(t^2) & \dots & u(t^N) \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^N \\ \mathtt{x} &= \begin{bmatrix} \mathbf{x}(t^1) & \mathbf{x}(t^2) & \dots & \mathbf{x}(t^N) \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{N\times 4}\,. \end{aligned}$$

Damit können die Trajektorien am realen Versuchsaufbau geladen und in Echtzeit aufgeschalten werden.

c) Überprüfen Sie mithilfe der Funktion `check_trajectory(filename)`, ob Ihre Trajektorie im richtigen Datenformat exportiert wurde und die physikalischen Beschränkungen des Versuchsaufbaus einhält.

d) Laden Sie Ihre Trajektorie im TUWEL-Kurs der Übung hoch.

©, Institut für Automatisierungs- und Regelungstechnik, TU Wien