Stefan Leutenegger, Michael Benz, Daniel Ambühl, Simon Wieser, Jan Mathys, Dominic Seibold, Yanik Schwab

Regelung eines Lastkrans

Praktikum Mess- und Regelungstechnik
Anleitung zum Versuch

Institut für Dynamische Systeme und Regelungstechnik
Eidgenössische Technische Hochschule Zürich

19. Februar 2024

IDSC

Inhaltsverzeichnis

1 Einleitung **1**

2 Vorbereitung **1**
2.1 Versuchsaufbau 1
2.2 Modellierung 2

3 Versuchsdurchführung **9**
3.1 Kalibrierung 9
3.2 PI-Seillängenregelung 9
3.3 Getrennte SISO-Regelung von Katzenposition und Winkel 10
3.4 Zustandsregelung (MIMO-Regelung) 11

1 Einleitung

Das Kranmodell bietet als elektromechanisches System mit relativ einfachem Aufbau eine gute Möglichkeit, erste praktische Regelungserfahrung zu machen - die hoffentlich zu zahlreichen Aha-Erlebnissen führt.

Im Vorbereitungsteil wird ein Systemmodell gebildet und analysiert. Dieser Teil ist vor der Teilnahme am Messlabor zu lesen und es müssen die enthaltenen Vorbereitungsaufgaben bearbeitet werden. Kenntnis des Stoffes von Regelungstechnik II ist dafür Voraussetzung.

Der Versuch ist in vier Teile gegliedert: Zuerst werden Kalibrierungsparameter für den Kran bestimmt. In einem zweiten Schritt wird eine Seillängenregelung mittels PI-Regler ausgelegt. Nachdem die Regelung der Katzenposition und der Winkelregelung getrennt voneinander mittels SISO-Reglern angegangen wird, werden zum Schluss zwei Zustandsregler ausgelegt.

Es muss kein Laborbericht geschrieben werden. Natürlich empfehlen wir aber, die Resultate in sinnvoller Form zusammenzustellen und mit diesem Skript aufzubewahren.

In Abbildung 1 ist der Verladekran dargestellt, für welchen die Regler implementiert werden.

Abbildung 1: Verladekran

2 Vorbereitung

2.1 Versuchsaufbau

Verladekräne werden bei Dockarbeiten und in Anlagen grösserer Firmen verwendet. Im Messlabor ist eine einfache Versuchsanlage gleicher Struktur aufgebaut. Abbildung 2 zeigt den Versuchsaufbau schematisch. Zwei unabhängige stark untersetzte Gleichstrommotoren sorgen einerseits für eine vertikale Bewegung der Last (Hubmotor) und andererseits für die horizontale Verschiebung der Laufkatze[1] Bei einem normalen Arbeitsvorgang fährt die Katze aus einer bestimmten Ausgangsposition (aus welcher ein Greifer eine Last aufnimmt) in eine gegebene Endposition. Durch die

[1] Eine Laufkatze ist der Schlitten, mit dem sich die horizontale Position der Last verändern lässt.

Fahrbewegung der Katze wird die Last zu nur schwach gedämpften Pendelschwingungen angeregt. Um die Position der Katze und der Last feststellen zu können, ist ein Potentiometer am Getriebe des Hubmotors und ein weiteres an der Umlenkrolle des Seilzuges für die Katze angebracht. Der Auslenkungswinkel wird mit einem elektromechanischen Winkelsensor an der Katze gemessen.

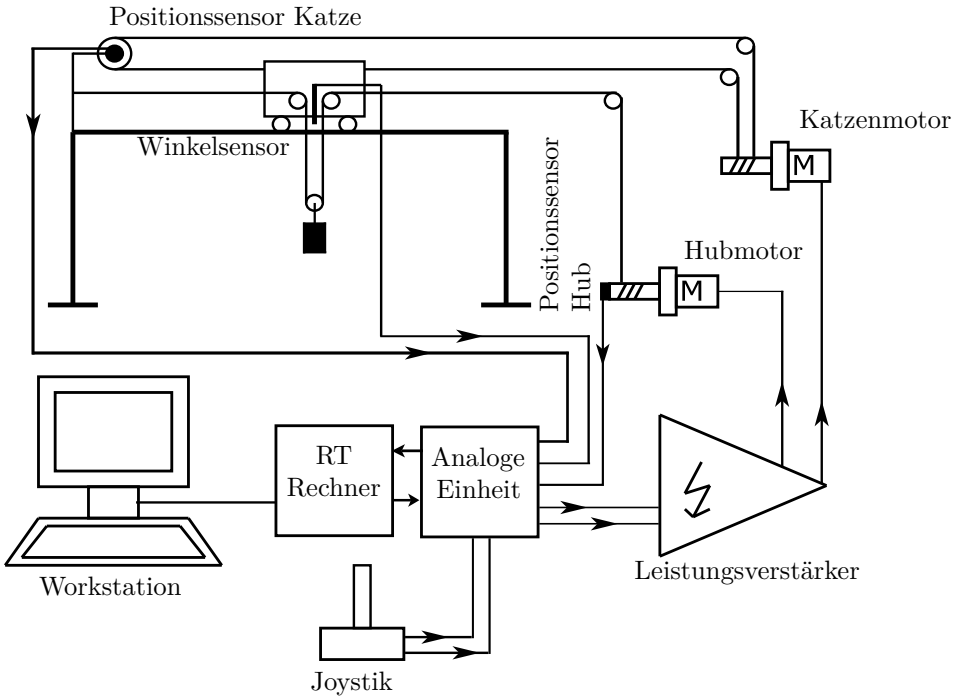


Abbildung 2: Versuchsaufbau

Die Peripheriekarte des Computers ist eine Echtzeit-Rechenkarte (Real-Time), ausgerüstet mit D/A- sowie A/D-Wandler und mit eigenem Prozessor. Auf dieser Karte läuft der Regler, der Messdaten verarbeitet und Stellgrössen berechnet bzw. setzt: hier sind dies die Ankerspannungen der Motoren. Diese Karte der Firma `dSPACE` wird auch als DSP-Karte bezeichnet (Digital Signal Processing). Die Steuersignale aus dem D/A-Wandler werden über den Leistungsverstärker an die Motoren weitergegeben.

Für dieses System gilt es nun einen Regler zu finden, der die Last unter möglichst kleinen Schwingungen innert kurzer Zeit vom Startpunkt ans Ziel bringt.

Hausaufgabe 1: Identifizieren Sie Stell- sowie Messgrössen.

2.2 Modellierung

2.2.1 Bewegungsdifferentialgleichungen

Nicht nur um Aufwand bei der Modellbildung einzusparen, sondern auch um ein aussagekräftiges Modell zu erhalten, soll es nur die relevanten Teile der Systemdynamik enthalten. Daher werden folgende Vereinfachungen gemacht:

1. Die Last ist nur ein Massepunkt.

2. Das Seil ist starr und masselos.
3. Der Luftwiderstand ist sehr schwach und dämpft die Pendelschwingungen praktisch nicht. Er wird deshalb vernachlässigt.
4. Die Motoren sind stark: Als Eingangsgrössen müssen deshalb nicht die Motormomente genommen werden, sondern direkt die Geschwindigkeiten. Diese hängen über einen konstanten Faktor mit der Ankerspannung zusammen.

Könnte die letzte Vereinfachung nicht gemacht werden, müsste an der Katze der Impulssatz aufgestellt werden und der Impulssatz der Last würde erheblich komplizierter.

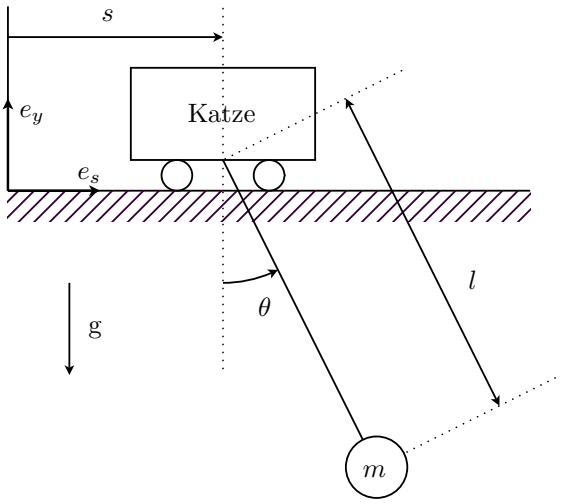


Abbildung 3: Koordinatensystem

Die Position der Last ist im Inertialsystem:

$$\vec{r}_{OL} = \begin{bmatrix} s \\ 0 \end{bmatrix} + \begin{bmatrix} l\sin\theta \\ -l\cos\theta \end{bmatrix} \tag{1}$$

Nun kann der Impulssatz angewendet werden:

$$m_L \cdot \ddot{\vec{r}}_{OL} = \sum \vec{F}_a \tag{2}$$

Also:

$$m_L \begin{bmatrix} \ddot{s} + \ddot{l}\sin\theta + 2\dot{\theta}\dot{l}\cos\theta + \ddot{\theta}l\cos\theta - \dot{\theta}^2 l\sin\theta \\ -\ddot{l}\cos\theta + 2\dot{\theta}\dot{l}\sin\theta + \ddot{\theta}l\sin\theta + \dot{\theta}^2 l\cos\theta \end{bmatrix} = \begin{bmatrix} -S\sin\theta \\ S\cos\theta - m_L g \end{bmatrix} \tag{3}$$

Nach der Elimination der Seilkraft S ergibt sich:

$$\ddot{\theta} = \frac{1}{l} \cdot \left(-g\sin\theta - 2\dot{\theta}\dot{l} - \ddot{s}\cos\theta\right) \tag{4}$$

Hausaufgabe 2: Welche alternativen Möglichkeiten gibt es, um die Differentialgleichung der Strecke zu bestimmen?

2.2.2 Zustandsraumdarstellung

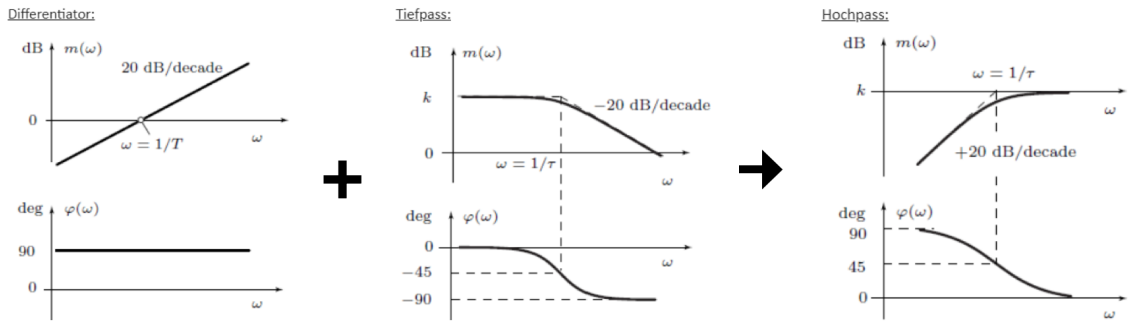
Die Eingangsgrössen sind:

$$\begin{aligned} u_1 &= \dot{s} \\ u_2 &= \dot{l} \end{aligned} \tag{5}$$

Und die Zustandsvariablen:

$$\begin{aligned} x_1 &= s \\ x_2 &= \dot{s} \\ x_3 &= \theta \\ x_4 &= \dot{\theta} = \omega \\ x_5 &= l \end{aligned} \tag{6}$$

In der Bewegungsgleichung tritt $\ddot{s} = \dot{x}_2$ auf. Rein mathematisch müsste also der Eingang u_1 abgeleitet werden um $\dot{x}_2$ zu erhalten. Da ein reiner Differentiator nicht umgesetzt werden kann (Signal würde unendlich verstärkt bei hohen Frequenzen), filtern wir zusätzlich mit einem Tiefpassfilter. Ein Differentiator und ein Tiefpassfilter zusammen ergeben wiederum einen Hochpassfilter wie in folgenden Abbildung veranschaulicht wird:



Statt

$$\ddot{s} = \dot{u}_1 \tag{7}$$

wird approximiert:

$$\ddot{s} = \frac{u_1 - \dot{s}}{\Delta t} = \omega_g \left(u_1 - \dot{s}\right) \tag{8}$$

Bzw. mit den oben definierten Eingangs- und Zustandsgrössen:

$$\dot{x}_2 = \omega_g \left(-x_2 + u_1\right) \tag{9}$$

ω_g stellt hierbei die Eckfrequenz des Hochpasses dar und wird auf 100 bis 1000 rad/s gesetzt. Gleichung (4) wird mit (9) zu

$$\dot{x}_4 = \frac{1}{x_5} \cdot \left(-g \sin x_3 - 2 x_4 u_2 - \omega_g \left(-x_2 + u_1\right) \cos x_3\right) \tag{10}$$

Der komplette Satz Zustandsgleichungen $\dot{x} = f(x, u)$ lautet:

$$\begin{aligned} \dot{x}_1 &= x_2 \\ \dot{x}_2 &= \omega_g \left(-x_2 + u_1\right) \\ \dot{x}_3 &= x_4 \\ \dot{x}_4 &= \tfrac{1}{x_5} \cdot \left(-g \sin x_3 - 2 x_4 u_2 - \omega_g \left(-x_2 + u_1\right) \cos x_3\right) \\ \dot{x}_5 &= u_2 \end{aligned} \tag{11}$$

Hausaufgabe 3: Zeichnen Sie die Bodeplots von u_1 nach x_1 und von u_1 nach $\dot{x}_2$

2.2.3 Linearisierung

Das erstellte Modell soll nun linearisiert werden. Es kann hier auf eine Normierung verzichtet werden, da sich die numerischen Werte der Zustandsgrössen in der gleichen Grössenordnung bewegen.

Zunächst muss ein sinnvoller Betriebspunkt festgelegt werden, um das System (11) nachher linearisieren zu können gemäss:

$$\delta\dot{x} = \left.\frac{\partial f\,(x,u)}{\partial x}\right|_{\substack{x=x_0\\u=u_0}} \cdot \delta x + \left.\frac{\partial f\,(x,u)}{\partial u}\right|_{\substack{x=x_0\\u=u_0}} \cdot \delta u \tag{12}$$

Als Betriebspunkt wird die Gleichgewichtslage des Pendels gewählt, also gilt $x_{2,0} = 0$, $x_{3,0} = 0$ und $x_{4,0} = 0$ Die Position s der Katze hat anschaulicher Weise keinen Einfluss auf das Systemverhalten, darum muss kein $x_{1,0}$ festgelegt werden. Die Länge des Pendels x_5 hat jedoch einen grossen Einfluss auf das Verhalten, insbesondere ist die Eigenfrequenz des Pendels abhängig von der Seillänge. Wir setzen $x_{5,0} = l_0 = 0.5\ m$.

Wir erhalten das folgende linearisierte System in der Form $\dot{x} = Ax + Bu$:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \\ \dot{x}_5 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & -\omega_g & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & \frac{\omega_g}{l_0} & -\frac{g}{l_0} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} + \begin{bmatrix} 1 & 0 \\ \omega_g & 0 \\ 0 & 0 \\ -\frac{\omega_g}{l_0} & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{13}$$

Gemessen werden die Grössen $x_1 = s$, $x_3 = \theta$ und $x_5 = l$. Diese sind also die Ausgangsgrössen des Systems. Zusätzlich interessiert uns aber noch die Position der Last (s_l, y_l). Die allgemeine Gleichung $y = Cx + Du$ wird dann (ebenfalls linearisiert) zu:

$$\begin{bmatrix} s_l \\ y_l \\ s \\ \theta \\ l \end{bmatrix} = \begin{bmatrix} 1 & 0 & l_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} \tag{14}$$

Hausaufgabe 4: Vervollständigen Sie das detaillierte Signalflussbild mithilfe von (13) und (14). Was finden Sie aussergewöhnlich daran? Was ist der Grund dafür?

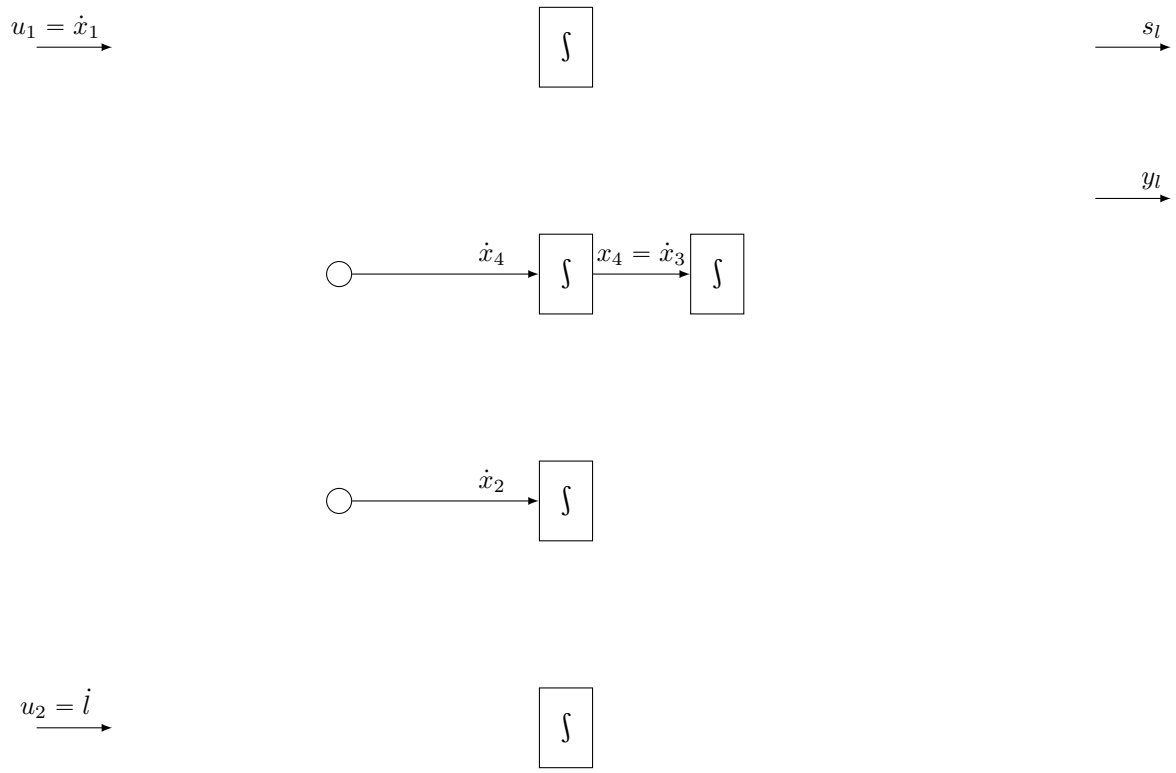


Abbildung 4: Signalflussbild

2.2.4 Übertragungsfunktionen

Weil das (um den Gleichgewichtszustand linearisierte) System entkoppelt ist, können die Systeme $\Sigma_s : u_1 \to s$ und $\Sigma_\theta : u_1 \to \theta$ getrennt von $\Sigma_l : u_2 \to l$ als SISO-Systeme betrachtet werden.

Aus den Systemmatrizen lassen sich die entsprechenden Übertragungsfunktionen ableiten:

$$\Sigma_s(s) = \frac{S}{U_1} = \frac{1}{s} \tag{15}$$

$$\Sigma_\theta(s) = \frac{\Theta}{U_1} = \frac{-s}{l_0 \left(s^2 + \frac{g}{l_0}\right) \left(\frac{s}{\omega_g} + 1\right)} \tag{16}$$

$$\Sigma_l(s) = \frac{L}{U_2} = \frac{1}{s} \tag{17}$$

Man beachte das negative Übertragungsverhalten von $\Sigma_\theta(s)$. Dieses ist nur eine Folge der Einführung der Variablen und kann anschaulich erklärt werden: beginnt man die Katze in positive s-Richtung (nach rechts) auszulenken, so wird der Winkel zunächst negativ (Massenträgheit: das Pendel schlägt nach links aus).

Hausaufgabe 5: Betrachten Sie die beiden Bode-Plots und ordnen Sie ihnen $\Sigma_s(j\omega)$ und $\Sigma_\theta(j\omega)$ zu!

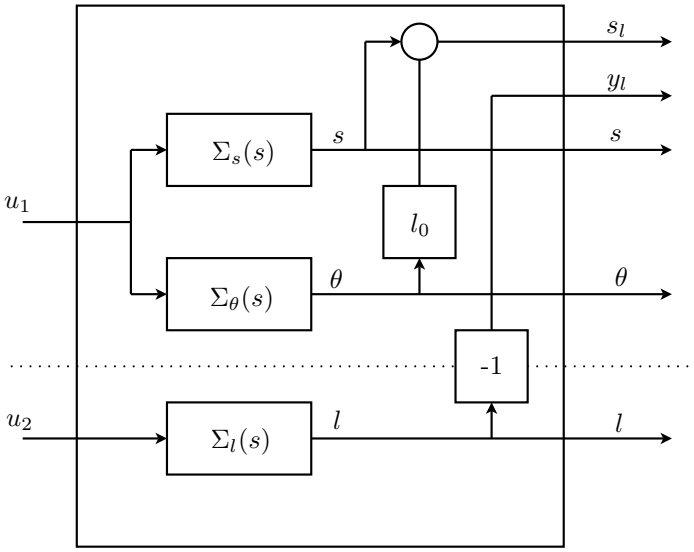


Abbildung 5: Unterteiltes System

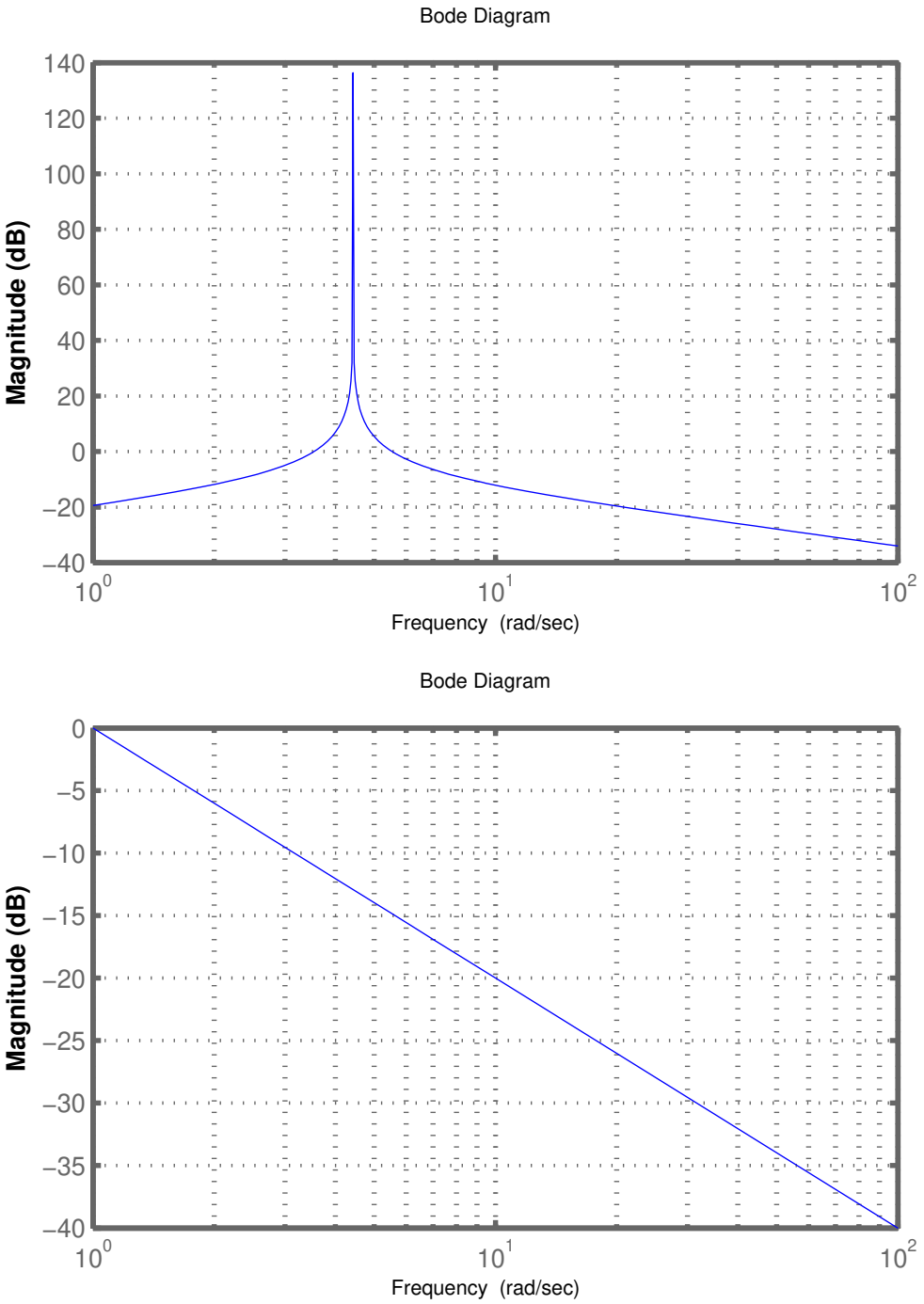


Abbildung 6: Übertragungsfunktionen $\Sigma_s(j\omega)$ und $\Sigma_\theta(j\omega)$

3 Versuchsdurchführung

Gehen Sie nun wie folgt vor um sich mit dem Versuch vertraut zu machen:

1. Gehen Sie als erstes den ganzen Aufbau durch und versuchen Sie die Elektromotoren und die Sensoren zu identifizieren. Besprechen Sie es miteinander und mit den Hilfsassistent*innen.
2. Machen Sie sich mit dem Versuchsaufbau vertraut: Am Joystick muss der Schalter auf „manuell“ gestellt sein, damit die Regelstrecke von Hand gesteuert werden kann. Versuchen Sie die Position der Katze möglichst schnell zu verändern, ohne dass die Last zu stark schwingt.
3. Beim Computerstart öffnet sich Matlab automatisch und es wird die Matlab/Simulink-Umgebung initialisiert. Neben Matlab wird eine Software benötigt, die eine Verbindung zum Controller des Krans herstellt. In diesem Fall verwenden wir die `dSpace`-Umgebung. Mit dieser lassen sich Messwerte anzeigen und Stellwerte verändern, während ein Regler aktiv steuert.

3.1 Kalibrierung

Um die zuverlässige Funktion des Systems zu gewährleisten, muss es jeweils kalibriert werden. Die Sensoren geben eine Spannung aus, welche in eine Länge oder in einen Winkel umgerechnet werden müssen. Dazu wird jeweils ein „gain“ multipliziert und ein „offset“ addiert. Öffnen Sie die `dSpace`-Umgebung (Rechts neben dem Matlab-Symbol in der Taskbar) und darin die Aufgabe `Exercise00`. Klicken Sie „go online“ damit die Messwerte angezeigt werden. Bestimmen Sie die Kalibrierungsparameter und notieren sich diese. Verwenden Sie als Nullpunkt der s-Achse den Nullpunkt des vormontierten Massstabs. Geben Sie danach die gefunden Werte im Matlab file `setSimulinkParameters.m` ein. Damit werden die Kalibrierungswerte für die folgenden Aufgaben übernommen.

3.2 PI-Seillängenregelung

In dieser Aufgabe arbeiten Sie zunächst im Simulink. Gehen Sie also nun zum Simulink-File `kran_ueb1`.

1. PI-Regler bauen:
   - Vervollständigen Sie das Signalflussbild zu einem PI-Regler mit folgender Struktur:

$$C_l(s) = k_p \cdot \left[1 + \frac{1}{s \cdot T_i}\right] \tag{18}$$

   - Die Blöcke, welche Sie brauchen, sind bereits im Simulink-File drin. Welchen Einfluss haben die Parameter k_p und T_i auf das Regelsystem? **Beginnen Sie mit $k_p = 1$.** Bestimmen Sie die Durchtrittsfrequenz ω_c (bei ausgeschaltetem I-Teil) von $L(j\omega)$ und wählen Sie die Eckfrequenz $1/T_i$ des I-Teils **mindestens eine Dekade kleiner.**
2. Geschwindigkeitsgrenze und Anti Reset Windup
   - Überlegen Sie sich nun eine sinnvolle vertikale Maximalgeschwindigkeit (testen Sie diese!). Fügen Sie dem Regler einen Saturator-Block hinzu und limitieren Sie damit die Geschwindigkeit entsprechend. Welche Probleme entstehen bei limitierter Geschwindigkeit, wenn der I-Teil des Reglers aktiv ist?
   - Um die Probleme zu umgehen, muss der I-Teil ausgeschaltet werden, sobald Sättigung auftritt. Ist dies nicht der Fall, wird der I-Teil den Fehler weiter aufintegrieren. Fügen Sie vor den I-Teil den gegebenen „Switch“ ein: sein Steuersignal ist $u_{nachSaturator} - u_{vorSaturator}$. Wenn das Steuersignal 0 ist (also keine Sättigung stattfindet) soll der I-Teil wie bisher operieren, sonst soll ihm der konstante Wert 0 übergeben werden, damit er nicht weiter „volläuft“. Um die genaue Funktionsweise des Switch zu verstehen:

Doppelklick auf das Icon. Fügen Sie die ganze Struktur schliesslich aus `kran_ueb1` hinzu (Anti Reset Windup in Struktur einbauen).

3. Regler testen:

   - Der Regler soll nun getestet werden. Drücken Sie Ctrl+B „Build“, um aus dem Simulink Modell C-Code zu generieren, der dann automatisch kompiliert, gelinkt und auf den DSP geladen wird.
   - Die `dSPACE`-Umgebung ermöglicht online Änderungen von Reglerparameter und Referenzwerte, ohne das Matlab/Simulink Skript erneut zu compilieren. Öffnen Sie das Programm `dSPACE` auf dem Desktop und laden Sie das File `LoadingCrane Exercise01`. Die Fehlermeldung, die auftritt, soll zweimal mit „YES“ akzeptiert werden. Sobald Sie „Go Online“ drücken, läuft der Regler auf dem Echtzeitsystem und kann am Joystick eingeschaltet werden. Der Schalter kann und soll auch als Not-Aus verwendet werden.
   - Nun können online die Sollhöhe und die Reglerparameter geändert werden. Verändern Sie die Reglerparameter in beide Richtungen und beobachten Sie die Auswirkungen mittels Sollwertsprüngen von verschiedenen (aber nicht allzu grossen) Höhen. Verwenden Sie hierzu den Slider und beachten Sie die Beschränkungen, die Ihnen das Kranmodell vorgibt! Testen Sie nun das Schwingverhalten auch für grössere Sollwertsprünge. Der Parameter k_p kann nun erhöht werden, ohne dass zu hohe Geschwindigkeiten auftreten werden.

3.3 Getrennte SISO-Regelung von Katzenposition und Winkel

Die verbleibenden Grössen s und θ sollen getrennt geregelt werden.

Das Problem bei diesem Ansatz ist, dass die Systeme $\Sigma_s(s)$ und $\Sigma_\theta(s)$ den gleichen Input u_2 haben. Die Regler werden einander also stören. Da sie aber in verschiedenen Frequenzbändern operieren können, wird die gegenseitige Beeinflussung stark beschränkt.

1. Öffnen Sie nun das Simulink File `kran_ueb2` und machen Sie sich mit der Reglerstruktur vertraut.
2. Bauen Sie die PI-Seillängenregelung aus der vorherigen Übung ein. Hierfür kopieren Sie die Struktur und fügen Sie bei der „PI-Länge“ wieder ein. Achtung: Der Feedback-Loop ist bereits in der Reglerstruktur implementiert.
3. Öffnen Sie auch das Skript `LoadingCrane Exercise02` in der dSPACE Umgebung und identifizieren Sie die Blöcke, die die Parameter vom Simulink Modell `kran_ueb2` beeinflussen.
4. Zunächst soll nur der Regelkreis des Winkels geschlossen werden: θ soll mittels P-Regler $C_\theta(s) = k_p$ auf 0 stabilisiert werden.

   - Warum wird der Regelkreis des Winkels und nicht derjenige der Position s zuerst geschlossen?
   - Überlegen Sie sich, warum kein I-Teil gebraucht wird.
   - Nun soll ein Wert für k_p gefunden werden. Für Frequenzen deutlich unterhalb ω_g kann approximiert werden:

$$\Sigma_\theta(s) = \frac{\Theta}{U_1} \approx \frac{-s}{l_0\left(s^2 + \frac{g}{l_0}\right)} \tag{19}$$

   Die Übertragungsfunktion des geschlossenen θ-Regelkreises (ohne gleichzeitiger s-Regelung) lautet nun:

$$T_\theta(s) = \frac{k_p^\theta \cdot \Sigma_\theta(s)}{1 + k_p^\theta \cdot \Sigma_\theta(s)} = \frac{-\frac{k_p^\theta}{l_0}s}{s^2 - \frac{k_p^\theta}{l_0}s + \frac{g}{l_0}} \tag{20}$$

Wobei das Nennerpolynom in der (quadratischen) Form $s^2 + 2\delta\omega_0 s + \omega_0^2$ ist.

Identifizieren sie die Eigenfrequenz ω_0 sowie die Dämpfung δ von T_θ (s).

Bestimmen sie k_p^θ so, dass sich $\delta = \frac{\sqrt{2}}{2}$ ergibt, was einem vernünftiges Einschwingverhalten entspricht. Dass k_p^θ dabei negativ wird, mag ungewohnt sein, ist aber eine direkte Folge des negativen Übertragungsverhaltens von $\Sigma_\theta(s)$. Editieren Sie k_p^θ entsprechend im Simulink-File.

Lassen Sie den Winkelregler laufen: stossen Sie das Pendel von Hand an und beobachten Sie, wie der Regler arbeitet.

5. Der Katzenpositions-Regelkreis wird ebenfalls mittels P-Regler $C_s(s) = k_p^s$ geschlossen. Bei der Bestimmung von k_p^s muss beachtet werden, dass die Durchtrittsfrequenz des resultierenden Loopgains $L_s(j\omega_c^s)$ genügend kleiner als ω_0 ist (Eigenfrequenz des geschlossenen θ-Regelkreises). So regt die s-Regelung das Pendel nicht zum Schwingen an.

   Dass $\omega_c^s \ll \omega_c^\theta$ erfüllt sein muss, damit sich die Regler nicht zu stark konkurrenzieren, ist hier die schwächere Bedingung und mit $\omega_c^s < \omega_0$ automatisch erfüllt.

   - Wählen Sie k_p^s so, dass ω_c^s höchstens halb so gross wie ω_0 wird. Geben Sie den gefundenen Wert im Simulinkmodell in den entsprechenden Block ein.
   - Nehmen Sie den Regler im Betrieb („Build“ (Ctrl+B) im Simulink und „Go Online“ in der dSPACE Umgebung) und testen Sie ihn. Was wird passieren, wenn Sie k_p^s erhöhen, bis gilt: $\omega_c^s > \omega_0$? Probieren Sie es aus!

3.4 Zustandsregelung (MIMO-Regelung)

Als Alternative zu Aufgabe 2 soll nun eine Zustandsregelung für das verbleibende Subsystem ausgelegt werden. Da die Zustände $x_2 = \dot{s}$ und $x_4 = \dot{\theta}$ nicht gemessen werden, ergeben sich zwei Möglichkeiten:

1. In einen ersten Versuch werden Katzen- $\dot{s}$ und Winkelgeschwindigkeit $\dot{\theta}$ von den Messgrössen s und θ direkt abgeleitet und danach kann sehr anschaulich ein normaler Zustandsregler ausgelegt werden. Physikalisch sind $x_2 = \dot{s}$ und $x_4 = \dot{\theta}$ die Ableitungen von $x_1 = s$ und $x_3 = \theta$. Um das Rauschen, mit dem die Messdaten beaufschlagt sind, zu unterdrücken, werden aber keine reinen Differentiatoren eingesetzt: ähnlich wie bei der Modellierung schon mal angetroffen werden sie um Tiefpässe erweitert und wir erhalten:

$$\begin{aligned} X_{2,rek} &= \frac{s}{\tau_{r,v}s+1} \cdot X_1 \\ X_{4,rek} &= \frac{s}{\tau_{r,\theta}s+1} \cdot X_3 \end{aligned} \tag{21}$$

   In `kran_ueb3a` ist diese Rekonstruktion der Zustandsvariablen bereits implementiert und die Eckfrequenzen $\frac{1}{\tau_r}$ sind auf ca. 1000 rad/s eingestellt.

   Noch nicht bestimmt ist der Gewichtungsvektor K der Zustandsrückführung: er soll so gewählt werden, dass er das Gütekriterium

$$J = \int_0^\infty \left(x^T Q x + u^T R u \right) dt \tag{22}$$

   minimiert wird.

   - Die Lösung des Problems (Riccatti-Gleichung) wird vom Matlab-Skript `LQ` übernommen. Öffnen Sie es und verändern Sie die Werte der Matrizen, das heisst gewichten Sie den Regler anders:

     Q soll nur positive Diagonal-Elemente enthalten: diese bestimmen, wie wichtig eine kleine Abweichung der betreffenden Zustandsgrösse vom Sollwert ist.

$R = \rho \cdot I$: Die Höhe des (positiven) Parameters ρ bestimmt, ob ein kleiner Reglerausgang u gewünscht ist. Je tiefer Sie also ρ wählen, desto aggressiver wird Ihr Regler!

Führen Sie das Skript aus (`F5`). Nun befindet sich der Vektor K im Matlab-Workspace. Betrachten Sie die verschiedenen Plots im Matlab, insbesondere die Sprungantworten! Wiederholen Sie diesen Schritt, bis Sie zufrieden sind. Bevor Sie jetzt zum Simulink wechseln, muss das Matlab-File geladen werden!

- Wechseln Sie nun ins Simulink und kopieren Sie wieder die PI-Regelung der Seillänge.
- Testen Sie den Zustandsregler indem Sie in der dSpace Umgebung das Skript `LoadingCrane Exercise03a` aktivieren. Wiederholen Sie das Prozedere bis Sie mit dem Regler zufrieden sind: Änderung der Matrizen im Matlab –> Plots checken –> RUN –> wechseln ins Simulink –> Regler testen. Vergessen Sie nicht, jedes Mal erneut das Skript auszuführen sowie den Schritt „Build“ vorzunehmen. Schliessen Sie auch die Sollwert-Trajektorien an, um gleichzeitige Sollwertsprünge für l und s zu testen!

2. Nun wird ein Zustandsbeobachter eingeführt, der nichts anderes als unser Systemmodell darstellt, dem dann die Zustände entnommen werden können. Es wird gemäss LQG/LTR-Iteration vorgegangen. Kern-Idee des LTR (Loop Transfer Recovery) ist, dass der Output der realen Strecke ständig mit demjenigen des Beobachters verglichen werden muss: der Fehler wird dann dem Beobachter über die Beobachterverstärkungs-Matrix zurückgeführt und korrigiert. Diese Korrektur entspricht einem dualen Zustandsregelungsproblem des Beobachters, das ebenfalls über die Minimierung eines Gütekriteriums gelöst wird.

   - Öffnen Sie das Matlab-Skript `LQGLTR`. Die Parameter zur Bestimmung der Beobachterverstärkungs-Matrix sind bereits gewählt und sollen so belassen werden. Genau gleich wie in der vorherigen Aufgabe müssen Sie jetzt nur noch die Matrizen Q und R wählen. Als Startwerte können Sie dieselben Werte wie vorhin einsetzen. Führen Sie das Skript aus (`F5`), und betrachten Sie Plots. Wiederholen Sie diese Schritte, bis Sie mit den Sprungantworten zufrieden sind.
   - Öffnen sie das Simulink-Modell `kran_ueb3b`, versuchen Sie, die Struktur zu verstehen und kopieren Sie die PI-Seillängenregelung in den entsprechenden Block.

     Testen Sie den Regler (indem Sie in der dSpace Umgebung das Skript `LoadingCrane Exercise03b`) aktivieren und passen Sie die Gewichtungsmatrizen an, bis Sie mit der Performance zufrieden sind.