Signale & Systeme
Analyse eines Systems

Matrikelnr.: 9085107 & 1234567
Dozent: Apl. Prof. Dr. Lutz Gröll
Kurs: TINF22B1

21. November 2023

1 Vorgehen in Regelungstechnik

Zur Ermöglichung einer strukturierten Vorgehensweise bedarf es einer Orientierung. Dieser Arbeit liegt die in der Vorlesung behandelte Vorgehensweise in der Regelungstechnik zugrunde, welche nachfolgend erläutert wird.

1. Detaillierungsgrad für das System festlegen
2. Physikalisches Modell einschließlich der Störsignale erstellen
3. Eingangs-, Ausgangs-, Zustands- und Störgrößen des Systems festlegen
4. Physikalische Einheiten festlegen und ggf. Normierung in Prozent
5. Analyse des Modells
   (a) Ruhelagen, Anfangswerte, evtl. Linearisierung nichtlinearer Systeme
   (b) Systemeigenschaften ermitteln
6. Entwurf
   (a) Ziele, Arbeitspunkte und Trajektorien festlegen
   (b) Entwurf: Struktur, Bereich, Verfahren und Kriterien
7. Simulation und Rückinterpretation

2 Das System

2.1 Eigenschaften des Systems

Wir untersuchen ein System mit Nennergrad 1, einer Totzeit T_t und einem proportionalen Übertragungsverhalten. Daraus ergibt sich die Bezeichnung PT_1T_t.

2.2 Die Übertragungsfunktion

Formel 1 zeigt die Übertragungsfunktion die in diesem Skript analysiert wird. Formel 3 zeigt eine etwas andere Darstellung der Formel, sodass die Eigenschaften des Systems leichter abzulesen sind.

$$G(s) = \frac{Y(s)}{U(s)} \tag{1}$$

$$= \frac{3 * e^{-2s}}{(1+s)} \tag{2}$$

$$= \frac{3}{1+s} * e^{-2s} \tag{3}$$

2.3 Eingabe in MATLAB

Für die Eingabe der Übertragungsfunktion in MATLAB gibt es zwei Möglichkeiten.
In manueller Eingabe schreibt man `G(s) = 3 / (1 + s) * exp(-2 * s).`
In Computer-Algebra wird der Befehl `sys = tf([3], [1, 1],"IODelay", 2)` benötigt. Hierbei werden in den beiden Vektoren die Koeffizienten vor s^0 bzw. s^1 (im Nenner) angegeben. Bei komplizierteren Gleichungen hilft dabei der Befehl `expand(G(s)`, der einem die benötigten Koeffizienten liefert, was hier jedoch nicht nötig war. Darauf folgt der Befehl „IODelay“, mit dem die Totzeit $t = 2$ angegeben wird.
In beiden Fällen wird in MATLAB damit die Formel 3 dargestellt.

3 Darstellungsformen des Systems

3.1 Explizite Darstellung des Übertragungsoperators

3.2 Zustandsraumdarstellung

Die allgemeine Form der Zustandsraumdarstellung lautet:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

Mit dem MATLAB-Befehl `ss(sys)` (ss steht für State Space) berechnet MATLAB automatisch die Werte für unser System. Somit sieht die Zustandsraumdarstellung hier folgendermaßen aus:

$$\dot{x} = \begin{bmatrix}-1\end{bmatrix} x + \begin{bmatrix}2\end{bmatrix} u \tag{4a}$$
$$y = \begin{bmatrix}1.5\end{bmatrix} x + \begin{bmatrix}0\end{bmatrix} u \tag{4b}$$

Man kann auch die Zustandsraumdarstellung direkt eingeben, indem man man die Matrizen A, B, C, D belegt und den Befehl `sys = ss(A, B, C, D)` ausführt. Von hier aus kommt man mit `tf(sys)` wieder zur Übertragungsfunktion.

Ohne Matlab kann die Zustandsraumdarstellung in einfachen Fällen mithilfe des Substitutionstricks aus der Übertragungsfunktion erstellt werden. Für Systeme, bei denen auch $\dot{u}, \ddot{u}, \ldots$ in der Differentialgleichung vorkommt, ist das Ganze ein bisschen komplizierter, wird aber in Wikipedia beschrieben und kann entsprechend angewendet werden. Bei sprungfähigen Systemen sieht das noch etwas anders aus und es muss erst eine Polynomdivision gemacht werden, um D zu erhalten. Im Mehrgrößenfall ist es noch komplizierter.

Sind die Matrizen A, B, C, D bekannt, kommt man mit der folgenden Formel zur Übertragungsfunktion $G(s)$, wobei I die Einheitsmatrix darstellt.

$$G(s) = C(s * I - A)^{-1} * B + D$$

In unserem System erhält man damit aber die eigentliche Übertragungsfunktion, da die Totzeit T_t verloren geht. Im Zustandsraum stellt sich die Totzeit folgendermaßen dar:

$$\dot{x} = Ax + Bu(t - T_t)$$

–> HIER NOCHMAL GROELL NACHFRAGEN

Explizite Darstellung

Sind zusätzlich die Anfangswerte bekannt, so lässt sich $x(t)$ und $y(t)$ wie folgt berechnen:

$$x(t) = e^{A(t-t_0)} * x(0) * \int_0^t e^{A(t-t_0)} Bu(\tau)\, d\tau$$
$$y(t) = Ce^{A(t-t_0)} * x(0) + C \int_0^t e^{A(t-t_0)} Bu(\tau)\, d\tau + Du(t)$$

Hier ist bei der Eingabe in MATLAB wichtig zu beachten, dass für $e^{(\dots)}$ nicht `exp`, sondern `expm`, verwendet wird, da man hier die Matrizenmultiplikation benötigt.

3.2.1 Integralgleichung

3.2.2 Differentialgleichung

ACHTUNG: Beim Erstellen dieses Dokuments hatte ich zuerst die ANfangswerte vergessen! Das passiert mir in der Klausur nicht, da sonst ein Punkt weg ist !AUSRUFEZEICHEN!11!!1! Die Anfangswerte werden für die Simulation natürlich benötigt, weil die Funktion wissen muss, wo sie startet. Die Anfangswerte müssen mit den Anfangswerten der Ein- Ausgangsdarstellung korrespondieren. Das bedeutet $y(0) = CX(0^-) + D(0^-)$
$\dot{y}(0^-) = CAx(0^-) + CBu(0^-) + D\dot{u}(0^-)$

3.3 Eingangs-Ausgang Differentialgleichung

Dieses und das folgende Kapitel sind bei meinen Mitschrieben als 3a und 3b zusammengefasst

$$\begin{aligned} G(s) &= \frac{3}{1+s} * e^{-2s} \\ G_{PT_1}(s) = \frac{Y(s)}{U(s)} &= \frac{3}{1+s} \\ Y(s)(1+s) &= (3) * U(s) \\ Y(s) + Y(s) * s &= 3 * U(s) \\ \dot{y}(t) + y(t) &= 3 * u(t) \end{aligned}$$

In dieser Differentialgleichung taucht die Totzeit T_t nicht auf, da das Totzeitglied nicht mit einer gewöhnlichen Differentialgleichung beschrieben werden kann. Für ein Totzeitglied gilt: $y(t) = u(t-T_t)$. Setzt man unsere Totzeit $T_t = 2$ in die Ein-/Ausgangsdiffentialgleichung für das PT_1-Glied ein, würde folgen:

$$\begin{aligned} \dot{y}(t) + u(t-2) &= u(t) \\ \dot{y}(t) &= u(t) - u(t-2) \end{aligned}$$

Somit ist es für einige Betrachtungen einfacher, die Totzeit zu vernachlässigen.

3.4 Übertragungsfunktion

Entweder wir streichen das erste kapitel und schreiben hier den Kram hin, oder iwie doppelt, oder ne andere Lösung -> Theresa ist gefragt!

3.5 Gewichtsfunktion g(t) (Impulsantwort)

Dieses und das folgende Kapitel sind bei meinen Mitschrieben als 4a und 4b zusammengefasst

In Abbildung 3.5 ist die Reaktion des Systems auf technischen Eingangssprung zu sehen.
In MATLAB lässt sich das mit dem Befehl `impulse(sys)` simulieren, wodurch man die numerische Lösung erhält. In MATLAB lässt sich das mit dem Befehl `step(sys)` simulieren, wodurch man die numerische Lösung erhält. Die Übergangsfunktionfunktion $h(t)$ lässt sich aber auch analytisch berechnen und man erhält die explizite Lösung. Hierbei hilft einem der Befehl `ilaplace()`, der die Laplace-Transformierte zurückliefert:

```
syms g(t)
g(t) = ilaplace(H(s))
t = [0:0.1:8]
plot(t, g(t))
```

Beide Befehle führen zum folgenden Graphen:

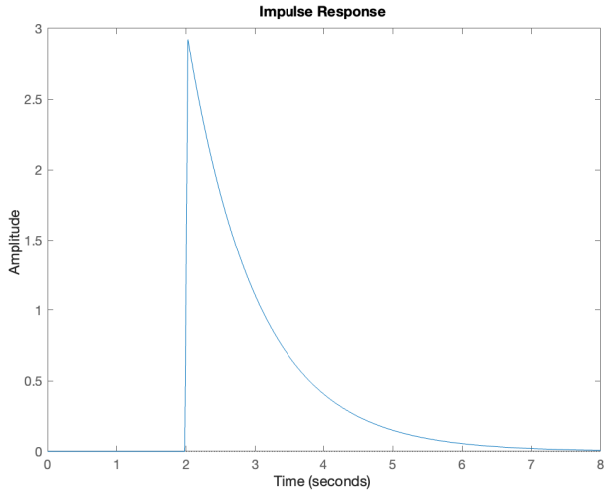

Abbildung 1: Impulsantwort

3.6 Übergangsfunktion h(t) (Sprungantwort)

In Abbildung 3.6 ist die Reaktion des Systems auf den Heaviside-Funktion zu sehen. Diese ist wie folgt definiert:

$$\sigma(t) = \begin{cases} 0 & \text{für t} < 0 \\ 1 & \text{für t} > 0 \end{cases}$$

In MATLAB lässt sich das mit dem Befehl `step(sys)` simulieren, wodurch man die numerische Lösung erhält. Die Übergangsfunktionfunktion $h(t)$ lässt sich aber auch analytisch berechnen und man erhält die explizite Lösung. Hierbei hilft einem der Befehl `ilaplace()`, der die Laplace-Transformierte zurückliefert:

```
H(s) = G(s) / s
syms h(t)
h(t) = ilaplace(H(s))
t = [0:0.1:8]
plot(t, h(t))
```

Beide Befehle führen zum folgenden Graphen:

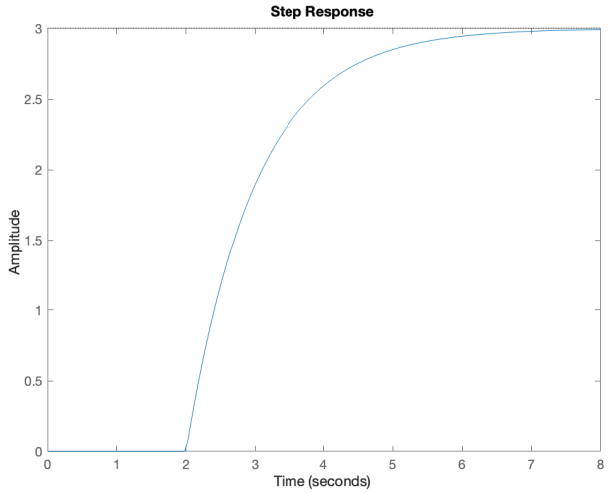

Abbildung 2: Sprungantwort

3.7 Frequenzgang ($G(j * \omega)$)

Dieses und die folgenden Kapitel sind in meinen Mitschrieben als 5a, 5b und 5c zusammengefasst.

Zum Frequenzgang $G(j * \omega)$ gelangt man, indem man in $G(s)$ das s durch $j * \omega$ ersetzt.

Das bedeutet, das für jede Frequzenz ω eine komplexe Zahl j mit Real- und Imaginärteil oder mit Betrag und Phase darstellt.

3.8 Ortskurve(Nyquist-Plot)

Im Nyquist-Plot wird der über ω parametrisierte Frequenzgang als Kurver in der komplexen Ebene mit Real- und Imaginärteil dargestellt. MATLAB bietet hier den Befehl `nyquist(sys)` an, welche zum folgenden Plot führt. Dabei zeigt MATLAB die Kurve von $-\infty$ bis $+\infty$ an, was in unserem Beispiel jedoch nicht nötig ist.

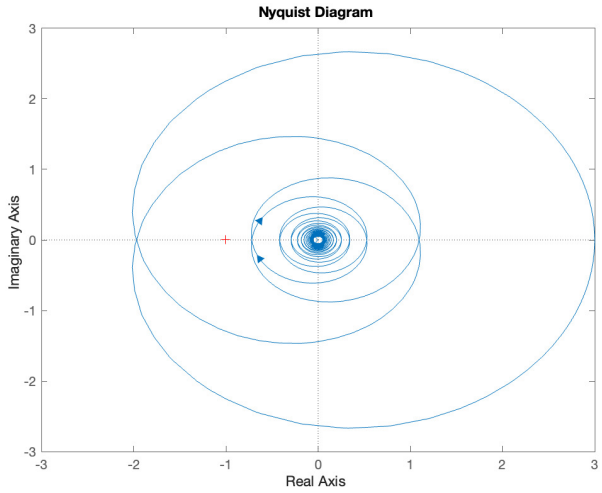

Abbildung 3: Nyquist-Plot

Darüber hinaus ist es in MATLAB möglich, mit dem Cursor entlang der Kurve zu gehen, um sich einzelne Werte genauer anzuschauen, wie in Abbildung 3.8 dargestellt.

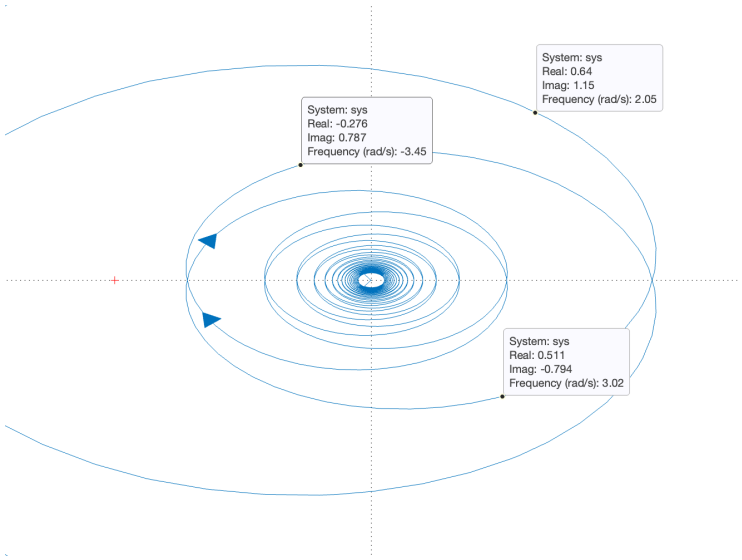

Abbildung 4: Screenshot: Informationen im Nyquist-Plot

Systemtheoretisch entspricht der Frequenzgang einem Schnitt der komplexen

Funktion $G(s)$ entlang der imaginären Achse: $G(s)\Big|_{s=j*\omega}$

3.9 Bode-Diagramm

Im Bode-Diagramm sind der Betrag des Frequenzgangs über ω in Dezibel(dB) und die Phase des Frequenzgangs über ω in Grad(°) dargestellt. Damit die Kurven schön aussehen, verwendet MATLAB eine doppelt logarithmische Darstellung. Der dazugehörige MATLAB-Befehl ist `bode(sys)`. Streng genommen wird in MATLAB im oberen Bild die Amplitudenverstärkung angezeigt, die ein Sinus-Signal stationär erfährt (bei Eigenwert = 1), währenddessen im unteren Bild die Phasenverstärkung zu sehen ist.

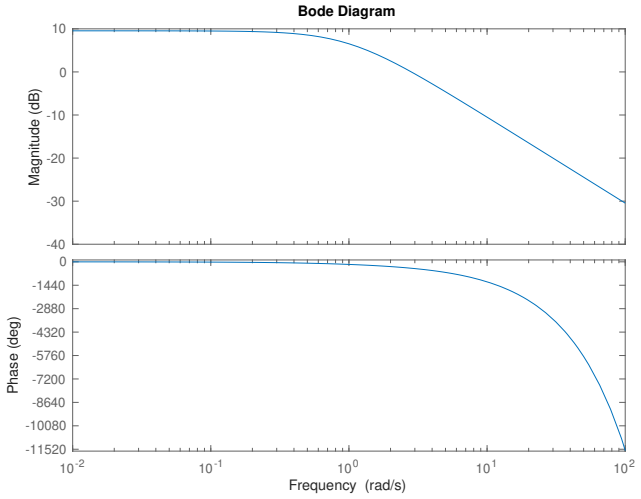

Abbildung 5: Bode-Plot

3.10 Statische Kennlinie

Da es sich bei unserem System um ein System mit proportionalen Übertragungsverhalten handelt, entspricht die statische Verstärkung $K = G(0)$ bzw. $K = \lim_{t\to\infty} h(t)$. Somit ergibt sich bei unserem System eine Verstärkung von 3, weshalb die statische Kennlinie nur eine konstante Funktion ist. Mithilfe des MATLAB-Befehls `fplot(3)` lässt sich das graphisch darstellen.

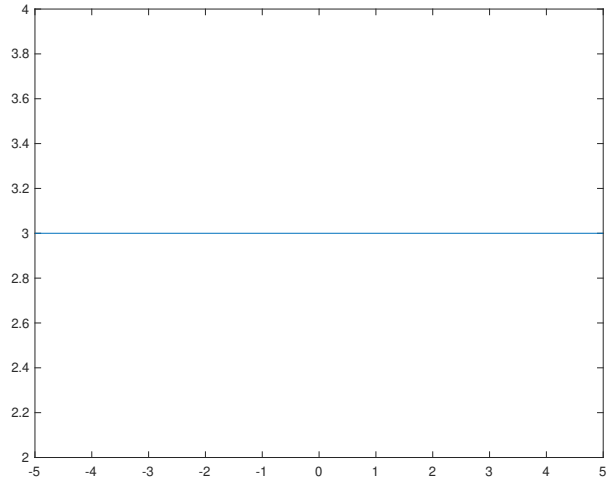

Abbildung 6: Statische Kennlinie

3.11 Pol-Nulstellen-Plot

Abbildung 3.11 zeigt den Pol-Nulstellen-Plot zu unserem System. Der passende MATLAB-Befehl hierfür lautet `pzplot(sys)`. MATLAB markiert mit **o** die Nullstellen und mit **x** die Polstellen des Systems. In unserem Beispiel hat das System aber keine Nulstellen, da der Zählergrad 0 ist, weshalb in der Abbildung kein **o** zu sehen ist. Da der Nenner unseres Systems aber den Grad 1 mit $1 + s$ ist, hat unserer System eine reale Polstelle bei s = -1, was in der Abbildung mit den **x** gekennzeichnet ist.

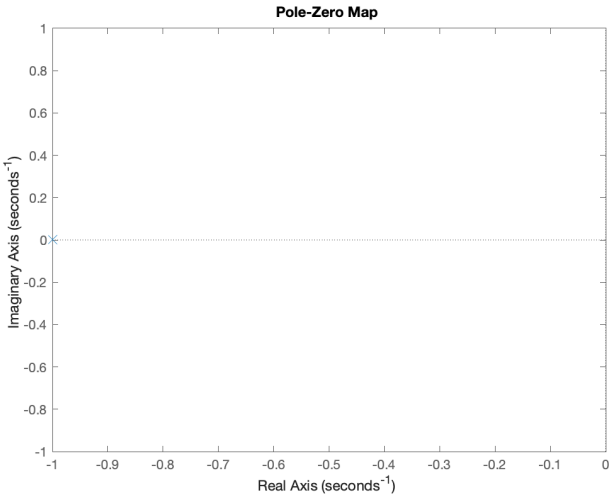

Abbildung 7: Bode-Plot