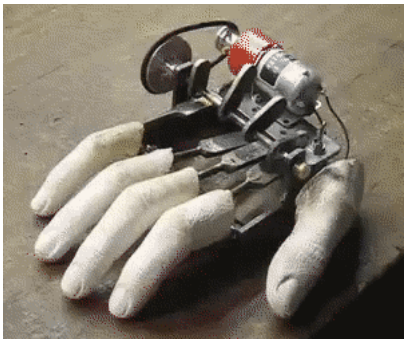
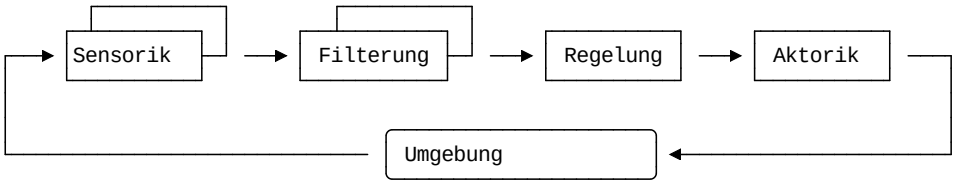
Regelungstechnik

Parameter	Kursinformationen
Veranstaltung:	`Softwareprojekt Robotik`
Semester	`Wintersemester 2021/22`
Hochschule:	`Technische Universität Freiberg`
Inhalte:	`Grundlegende Begriffe der Regelungstechnik`
Link auf GitHub:	https://github.com/TUBAF-Ifi-LiaScript/VL_Softwareentwicklung/blob/master/11_Regelungstechnik.md
Autoren	Sebastian Zug & Georg Jäger



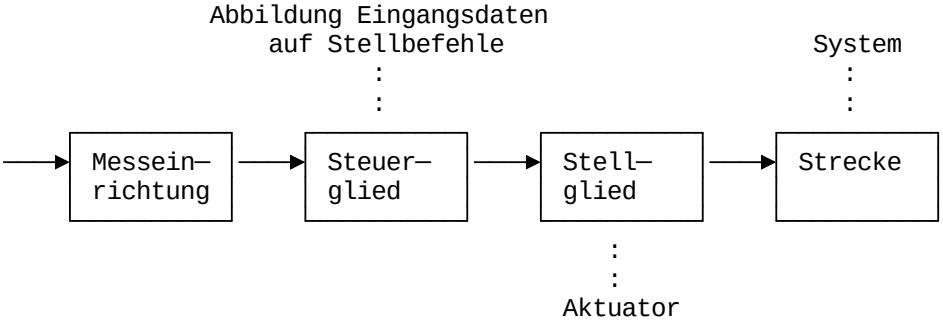
Wie weit waren wir gekommen?

... wir generieren ein "rohes" Distanzmessignal und haben es gefiltert.

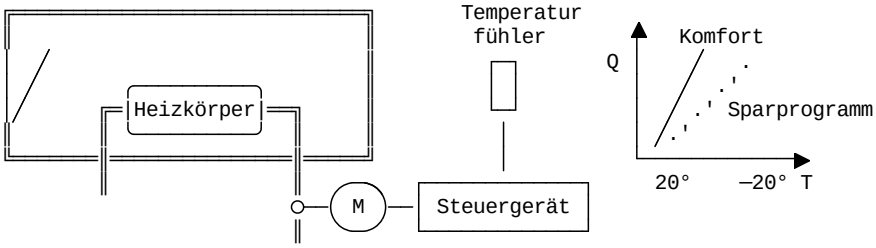


Begriffliche Einordnung

Definition „Steuerung“ nach DIN 19226: „Das Steuern, die Steuerung, ist ein Vorgang in einem System, bei dem eine oder mehrere Größen als Eingangsgrößen andere Größen als Ausgangsgrößen aufgrund der dem System eigentümlichen Gesetzmäßigkeiten beeinflussen. Kennzeichen für das Steuern ist der offene Wirkungsweg.“

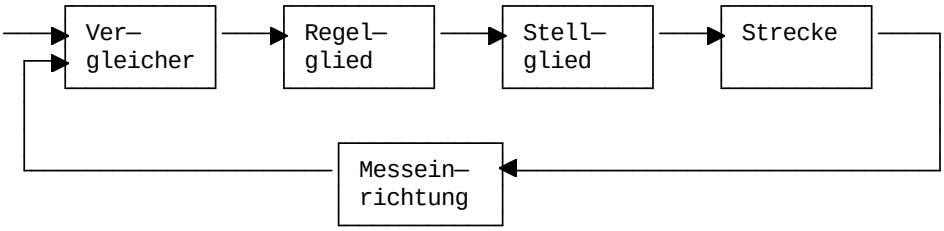


Anwendungsbeispiel: Heizungsanlage mit zentralem Außentemperaturfühler

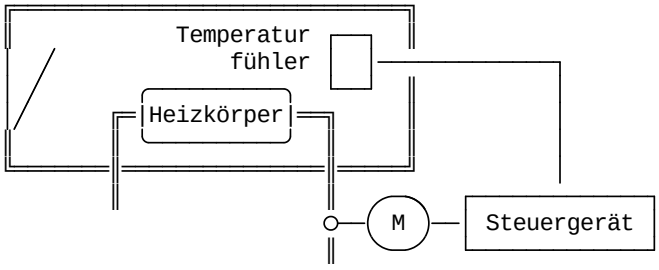


Eine Störung des Systemmodells durch das Öffnen der Tür kann nicht ausgeglichen werden.

Definition „Regelung“ nach DIN 19226 : „Das Regeln, die Regelung, ist ein Vorgang, bei dem eine Größe, die zu regelnde Größe (Regelgröße), fortlaufend erfasst, mit einer anderen Größe, der Führungsgröße, verglichen und abhängig vom Ergebnis dieses Vergleichs im Sinne einer Angleichung an die Führungsgröße beeinflusst wird. Kennzeichen für das Regeln ist der geschlossene Wirkungskreislauf, bei dem die Regelgröße im Wirkungsweg des Regelkreises fortlaufend sich selbst beeinflusst.“



Begriffe im Regelkreis	**Bedeutung**
Regelungsgröße x	„Ziel“ der Regelung, auch „Istwert“ In der Verfahrenstechnik x zumeist ein physikalischer (z. B. Temperatur, Druck, Durchfluss) oder chemischer Zustand (z. B. pH-Wert, Härte usw.)
Führungsgröße w	Zumeist tritt w in Form einer mechanischen oder elektrischen Größe (Kraft, Druck, Strom, Spannung etc.) auf und wird im geschlossenen Regelkreis mit der Regelgröße x verglichen.
Rückführgröße r	Die aus der Messung der Regelgröße hervorgegangene Größe, die zum Reglereingang auf das Vergleichsglied zurückgeführt wird.
Regeldifferenz e	Die Eingangsgröße e des Regelgliedes ist die vom Vergleichsglied errechnete Differenz aus Führungsgröße und Regelgröße.
Stellgröße y	Die Stellgröße wird vom Regler bzw. bei Verwendung eines Stellers vom Steller generiert.
Störgröße z	



Dabei unterscheiden wir zwei grundlegende Regelungskonzepte:

Merkmal	Folgeregler	Festwertregler
	Führungsgröße variabel	Führungsgröße w auf einen konstanter Wert eingestellt
Performance	Eine schnell veränderliche Führungsgröße erfordert einen Regel-kreis mit gutem Führungs-verhalten.	Festwertregler haben die Aufgabe Störungen auszuregeln und sind dementsprechend auf ein gutes Störverhalten auszulegen.
Beispiel	Abstandsbasierte Geschwindigkeitsregelung eines Fahrzeuges	Inverses Pendel

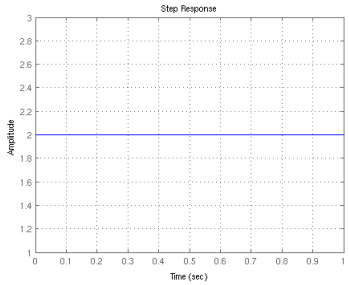
Allgemeines Vorgehen bei der Reglerauslegung

In beiden Fällen muss man sich bewusst sein, dass die Strecke nicht unmittelbar auf die Reglerinputs "anspringt" sondern auf die Änderung der Führungsgröße mit einem eigenen Verzögerungsverhalten regiert. Entsprechend der Abbildung der Stellgröße y auf die Regelgröße x unterscheiden wir dabei folgende Basistypen, wobei jeweils die Sprungantwort gezeigt wird.

Die Diagramme in diesem Bereich gehen auf den Autor "Chris828" zurück und sind unter https://de.wikipedia.org zu finden.

Proportionale Strecke

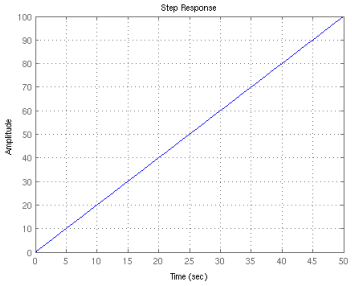
$$x(t) = K_p \cdot y(t) \text{ mit } K_p = 2$$



Beispiele: Verstärker, Spannungsteiler, Sensoren mit vernachlässigbarem Verzögerungsverhalten

Integral wirkende Strecke

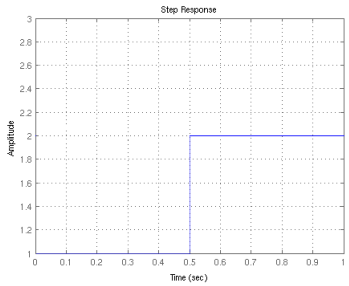
$$\dot{y}(t) = K_i \cdot u(t) \text{ mit } K_i = 2$$



Beispiele: Wassertank, Kondensator, Geschwindigkeit als Integration

Verzögerungsglied 0.Ordnung (Totzeitglied)

$$y(t) = u(t - T_t) \text{ mit } T_t = 0.5s$$



Beispiele: Rechenzeit für die Sensordatenverarbeitung, Material auf einem Förderband,

Verzögerungsglied 1.Ordnung (PT1-Glied)

$$y(t) = K(1 - e^{-\frac{t}{T}}) \text{ mit } K = 2, T = 1s$$

Beispiele: Gleichstrommotor

Verzögerungsglied 2.Ordnung (PT2-Glied)

$a_2\ddot{y}(t) + a_1\dot{y}(t) + a_0 y(t) = b_0 u(t)$ mit $K = 2, T = 1, D = 0.2, 1, 5$ mit D als Dämpfung

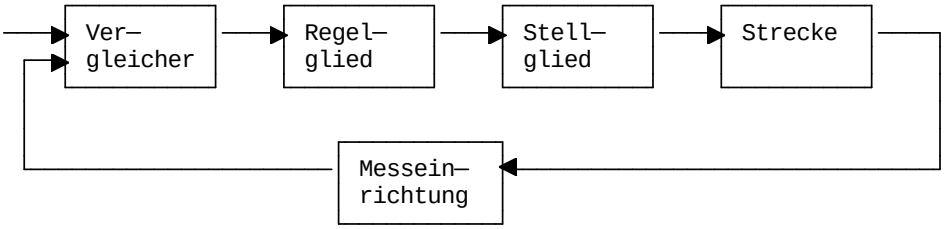
Beispiele: Zwei hintereinander geschaltete PT1-Glieder, Gleichstrommotor (Spannung → Drehzahl) mit berücksichtigter Induktivität

Wie kann eine Strecke anhand des Modells weiter charakterisiert werden?

Bode-Diagramm einer PT1 Strecke

Regler

Grundsätzlich lassen sich die Konzepte für die Regelung in zwei Kategorien einteilen. Legen wir über einen einfachen Schwellwert lediglich einen Maximalwert fest, zu dem y aktiviert oder eben deaktiviert wird, so spricht man von einem Zweipunkt-Regler. Ein solche Realisierung kann aber nur diskrete Zielgrößen formulieren. Im Unterschied dazu übergeben kontinuierlich wirkende Regler eine kontinuierliche Stellgröße an die Strecke.



Diskrete Regler

Für diskrete Regler existiert nur ein Spektrum an Zuständen, das als Basis der Regelung dient. Anhand eines Schwellwertes wird eine Heizungssteuerung an- oder ausgeschaltet.

[1]

[1] *Wikimedia Grafik des Autors "Heinrich Kümmeke"*

Kontinuierliche Regler

Die wichtigsten klassischen Regler sollen im Folgenden kurz vorgestellt werden. Dazu zählen Proportional-, Integral- und Differenzialregler.

Güteforderungen können u. a.die Stabilität, die Störkompensation und Sollwertfolge, sowie die Robustheit der Regler betreffen.

Proportionalregler

In ihrer einfachsten Form spricht man von einem verzögerungsfreien P-Regler, dabei verändert sich der Wert der Stellgröße $u(t)$ proportional zur Regelabweichung $e(t)$:

$$u(t) = K_P \cdot e(t)$$

Der statische Faktor K_P gibt die Stärke an, mit der der P-Regler auf die Regelabweichung reagiert. Bei dem hier vorgestellten Regler handelt es sich des Weiteren um einen Regler mit Proportionalglied 0ter Ordnung (PT0). Die Ordnung eines Gliedes gibt dessen Verzögerung an.

Integralregler

Beim integralwirkenden Regler (I-Regler) wird die Stellgröße $u(t)$ durch Integration der Regeldifferenz $e(t)$ gebildet. Die Stellgröße strebt dabei nur einem konstanten Wert zu, wenn die Regeldifferenz mit fortschreitender Zeit t gegen null geht.

$$u(t) = \frac{1}{T_N} \int_0^t e(\tau) d\tau$$

Integralwirkende Regel sind im Vergleich zu anderen Reglern zwar langsamer, haben jedoch den Vorteil, dass sie eine Abweichung von Soll- zu Regelgröße vollständig eliminieren können.

Differenzialregler

Bei differenzialwirkenden Regelungsgliedern (D-Regler) bestimmt die Änderung der Regelabweichung die der Stellgröße. Ein verzögerungsfreies D-Glied ist wie folgt definiert:

$$u(t) = T_D \frac{de(t)}{dt}$$

Achtung: In der Literatur wird bisweilen die Aussage getroffen, dass 90 Prozent der Einsatzfälle mit einem PID Regler handhabbar sind. Dies mag sicher pauschalisiert und wenig belegbar sein, deckt sich aber mit den Erfahrungen vieler Anwender.

Schwierig zu handhaben sind dabei insbesondere:

- Nicht-lineare Prozesse
- Totzeitglieder
- Fehlendes reales Leistungsverhalten der Aktorik

Implmentierung

Die größte Herausforderung liegt in der Bestimmung der entsprechenden magischen Konstanten, die ein rasche Annäherung an die Sollgröße bei gleichzeitig minimalem Überschwingen gewährleistet.

"Effects of varying PID parameters (Kp,Ki,Kd) on the step response of a system." Wikimedia Grafik des Autors "Physicsch" Link

Beispiel 1

Ein PID-Regler kann einfach durch Addition der einzelnen Regelglieder gebildet werden und die unterschiedlichen Faktoren (K_P, K_I, K_D) sind die Stellschräubchen an denen der Regler eingestellt werden kann. Mit einem Wert gleich 0 kann das jeweilige Regelglied auch ausgeschaltet werden.

$$u(t) = \underbrace{K_P \cdot e(t)}_{\text{Proportionalteil}} + \overbrace{K_I \cdot \sum_{t=0}^{t} e(t) + u(0)}^{\text{Integralteil}} + \underbrace{K_D \cdot (e(t) - e(t-1))}_{\text{Differenzialteil}}$$

Anhand eines Anwendungsbeispiels soll nunmehr die Wirkung genauer untersucht werden. Nehmen wir an, dass wir die Anfahrbewegung eines Roboters in ihrem Zeitverhalten kennen.

system_simulation.py

```python
# https://www.csestack.org/control-systems-simulation-python-example/
import math
import matplotlib.pyplot as plt
import numpy as np

steps = 2000

class SecondOrderSystem:
  def __init__(self, d1, d2):
    if d1 > 0:
      e1 = -1.0 / d1
      x1 = math.exp(e1)
    else: x1 = 0
    if d2 > 0:
      e2 = -1.0 / d2
      x2 = math.exp(e2)
    else: x2 = 0
    a = 1.0 - x1    # b = x1
    c = 1.0 - x2    # d = x2
    self.ac = a * c
    self.bpd = x1 + x2
    self.bd = x1 * x2
    self.init_system()

  def init_system(self):
    self.Yppr = 0
    self.Ypr = 0

  def __call__(self, X):
    Y = self.ac * X + self.bpd * self.Ypr - self.bd * self.Yppr
    self.Yppr = self.Ypr
    self.Ypr = Y
    return Y

fig, ax = plt.subplots()

Plant = SecondOrderSystem(250, 100)
t = np.arange(0, steps)
y = np.arange(0, steps, dtype = float)
for speed in [10, 100, 150, 200]:
  for index, value in enumerate(t):
      y[index]=Plant(speed)
  ax.plot(t, y, label = f"Voltage level '{speed}'")
  Plant.init_system()

plt.xlabel("Time")
plt.ylabel("Speed")
plt.legend()
plt.grid()
plt.show()
plot(fig) # <- this is required to plot the fig also on the LiaScript
  canvas
```

Kombinieren wir nun das Ganze mit einem Regler, so lässt sich das Verhalten nach unterschiedlichen Parametern optimieren.

control_loop.py

```python
# https://www.csestack.org/control-systems-simulation-python-example/
import math
import matplotlib.pyplot as plt
import numpy as np

steps = 1000

class PIDControlBlock:
  def __init__(self, Kp, Ki, Kd):
    self.Kp = Kp
    self.Ki = Ki
    self.Kd = Kd
    self.Epr = 0
    self.Eppr = 0
    self.Epppr = 0
    self.Sum = 0

  def __call__(self, E):
    self.Sum += 0.5 * self.Ki * (E + self.Epr)      # where T ~1
    U = self.Kp * E + self.Sum + 0.1667 * self.Kd * (E - self.Epppr + 3.0 *
      (self.Epr - self.Eppr))
    self.Epppr = self.Eppr
    self.Eppr = self.Epr
    self.Epr = E
    return U

class ClosedLoopSystem:
  def __init__(self, controller, plant) :
    self.P = plant
    self.C = controller
    self.Ypr = 0

  def __call__(self, X):
    E = X - self.Ypr
    U = self.C(E)
    Y = self.P(U)
    self.Ypr = Y
    return Y

Plant = SecondOrderSystem(250, 100)
#Pid = PIDControlBlock(5, 0.0143, 356.25)
Pid = PIDControlBlock(2, 0, 0)
Ctrl = ClosedLoopSystem(Pid, Plant)
t = np.arange(0, 1001)
setpoints = np.zeros(len(t))
setpoints[np.where(t > 50)]= 100
y = np.arange(0, 1001, dtype = float)
for index, value in enumerate(setpoints):
    y[index]=Ctrl(value)
fig, ax = plt.subplots()
ax.plot(t, setpoints, label = "Setpoint")
ax.plot(t, y, label = "Controled speed level")
plt.xlabel("Time")
plt.ylabel("Speed")
plt.legend()
plt.grid()
plt.show()

plot(fig) # <- this is required to plot the fig also on the LiaScript
  canvas
```

Beispiel 2

Das folgende Beispiel ist dem Buch von Peter Corke, "Robotics, Vision & Control", Springer 2017 entnommen. Der Code im Repository - `move_to_pose.py` - basiert auf einer Implementierung von AtsushiSakai - Link. Der Code wurde so angepasst, dass mehrere Sätze von Reglerparamtern parallel evaluiert werden können.

Wir wollen einen Roboter von einer Startposition zu einer Zielposition fahren lassen. Das Ganze allein mit einem PID als reaktives Verhalten umgesetzt werden.

Ausgangspunkt ist dabei das kinematische Modell eines differentiell getriebenen Roboters:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} cos\theta & 0 \\ sin\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v \\ w \end{pmatrix}$$

Der Fehler zwischen der Start und der Zielposition lässt sich beschreiben mit:

$$\rho = \sqrt{\Delta_x^2 + \Delta_y^2}$$
$$\alpha = tan^{-1}\frac{\Delta_y}{\Delta_x} - \theta$$
$$\beta = -\theta - \alpha$$

Diese Darstellung kann auch als Polarkoordinatenrepresentation der Problems interpretiert werden. Lediglich die Richtungsinformation des Roboters im Ziel β kommt als zusätzlicher Term hinzu.

Die Bewegungsgleichung in Polarkoordinaten ergibt sich damit zu

$$\begin{pmatrix} \dot{\rho} \\ \dot{\alpha} \\ \dot{\beta} \end{pmatrix} = \begin{pmatrix} -cos\alpha & 0 \\ \frac{sin\alpha}{\rho} & -1 \\ -\frac{sin\alpha}{\rho} & 0 \end{pmatrix} \begin{pmatrix} v \\ w \end{pmatrix}$$

für $-\frac{1}{2}\pi <= \alpha < \frac{1}{2}\pi$.

Da ρ, α und β den Fehler gegenüber der Zielgröße darstellen können wir diese mit einer proportionalen Verstärkung auf die Bewegungsvorgaben abbilden.

$$v = k_\rho \rho$$
$$w = k_\alpha \alpha + k_\beta \beta$$

Aufgabe: Erklären Sie den Einfluss der drei Proportionalen Regelerparamter bei der Bewegung des Roboters.

Trajektorie	k_ρ	k_α	k_β
rot	5	3	3
blau	5	15	3

Welchen Einfluss haben dabei die Beschränkungen des Modells in Bezug auf die maximalen Geschwindigkeiten `MAX_LINEAR_SPEED` und `MAX_ANGULAR_SPEED`?

Implementierung unter ROS

Für ROS gibt es ein Paket, welches einen PID Controller beispielhaft implementiert. Die entsprechende Dokumentation findet sich unter http://wiki.ros.org/pid

Evaluieren Sie das Systemverhalten für verschiedene Reglerkonfigurationen! Nutzen Sie dafür die Möglichkeit auf die Parameter des Reglers über `dynamic_reconfigure` zurückzugreifen.

Aufgaben

- Evaluieren Sie das Paket ROS control, dass einen PID Controller implementiert. Implementieren Sie mit Ihrem Roboter eine Geradeausfahrt auf schwierigem Gelände.