Karlsruher Institut für Technologie

Elektrotechnisches Institut

# Kapitel 4 – Implementierung PI-Regler ("Wrap Up")



Vorlesung

**Regelung Leistungselektronischer Systeme**

Sommersemester 2022

Dr.-Ing. Andreas Liske

# Strecke

1. Stromrichter : Totzeitglied
2. Last :

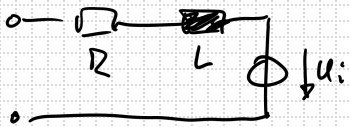
Drossel / Kondensator → I

mit $R_{par}$ → $PT_1$

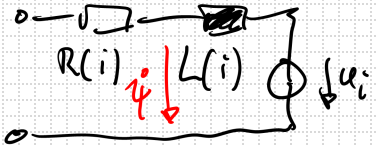
Maschinen → $PT_1$ mit Gegenspannung

RLS

linear




nichtlinear




→ "Nichtlineares Modell"

→ Kennfelder der Flussverkettung $\psi(i)$

EREA

Bei $\omega_0$ : I-Verhalten !

## PI-Regler


$$G_r(s) = V_p\left(1 + \frac{1}{sT_N}\right)$$

$$y = e \cdot V_p\left(1 + \frac{1}{sT_N}\right) = \underbrace{V_p \cdot e}_{P} + \underbrace{\frac{V_p}{T_N} \cdot e \cdot \frac{1}{s}}_{I}$$

Proportionalverstärkung: $V_p = V_p$

Integralverstärkung: $V_I = \frac{V_p}{T_N}$ → $V_I$ muß mit $V_p$ geändert werden, wenn $T_N$ konstant sein soll.

Reglerparameter: $V_p$, $T_N$

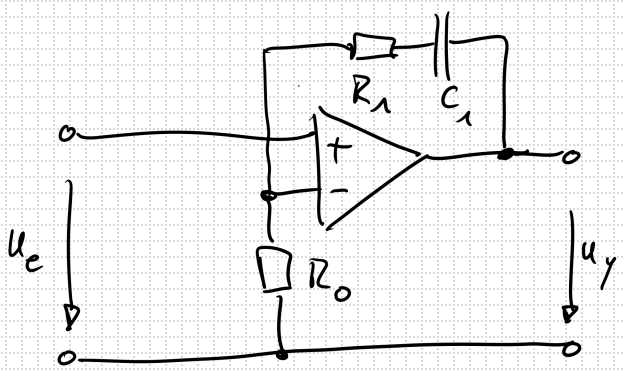
- Bei $\omega_D$: P-Verhalten ⇒ $\omega_D$ wird über $V_p$ eingestellt
- bei $\omega < \omega_D$: I-Verhalten ⇒ $\omega_N = \frac{1}{T_N} \overset{!}{\ll} \omega_D$

# Parametrierung PI-Regler

| | allg. Frequenz-KL-Verfahren (Bode) | Betragsoptimum | Symm. Optimum |
|---|---|---|---|
| Strecke | beliebig, $T_t$ gültig | $PT_1$ | $PT_1$ |
| Näherungen | — | $T_t \approx PT_1'\big\|_{T_1'=T_t}$<br>$PT_1 \cdot PT_1 = PT_2 \approx PT_1\big\|_{T_1=T_\sigma}$ | $T_t \approx PT_1\big\|_{T_1=T_t}$<br>$PT_1 \cdot PT_1 = PT_2 \approx PT_1\big\|_{T_1=T_\sigma}$ |
| $\omega_D$ | beliebig<br>$\Rightarrow \varphi_R \approx 55° \Rightarrow \omega_D = \frac{1}{2\ldots3T_t}$ | don't care | don't care |
| $V_P$ | $V_P = \frac{1}{\lvert F_S\rvert}\Big\|_{\omega_D}$ | $V_P = \frac{T_A}{2 \cdot \underbrace{T_\sigma}_{T_0} \cdot V_t \cdot V_A \cdot V_i}$ | $V_P = \frac{2\pi J}{T_M}$<br>$J$: Trägheit<br>$T_M$: Zeitkonst. des Mech. |
| $T_N$ | $T_N = \frac{1}{\omega_N}$ mit $\omega_N \leq \frac{1}{2}\omega_D$<br>$\Rightarrow T_N \geq 2 \cdot \frac{1}{\omega_D} = 4 \cdot T_t$ | $T_N = T_A$ | $T_N = 4 \cdot T_\sigma$ |

# Implementierung PI-Regler

analog: OP-Amps

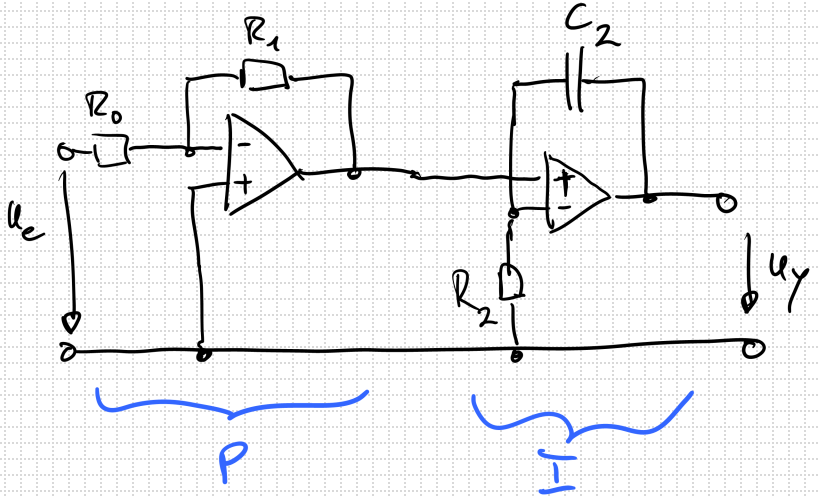



nichtinvertierender PI-Regler:

$$F_R(s) = V_P\left(1 + \frac{1}{sT_N}\right)$$

$$V_P = 1 + \frac{R_1}{R_0}$$

$$T_N = (R_0 + R_1)\cdot C_1$$



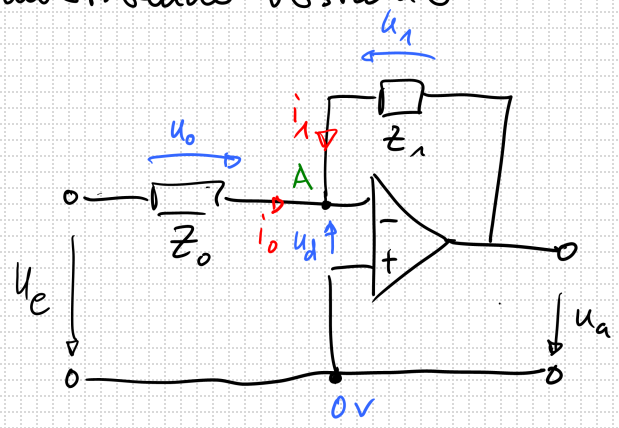

invertierender PI-Regler mit unabhängig einstellbaren Parametern

$$V_P = -\frac{R_1}{R_0}$$

$$T_N = R_2 C_2$$

Herleitung:

a) invertierender Verstärker




1) $u_d = 0V \Rightarrow \varphi_A = 0V$

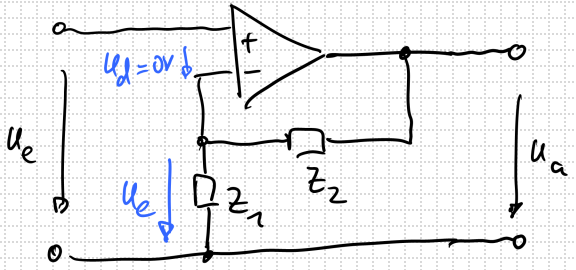
$\Rightarrow u_0 = u_e$

$u_1 = u_a$

2) $i_0 + i_1 = 0$

$$\frac{u_e}{z_0} + \frac{u_a}{z_1} = 0 \Rightarrow \boxed{\frac{u_a}{u_e} = -\frac{z_1}{z_0}}$$

b) nicht invertierender Verstärker



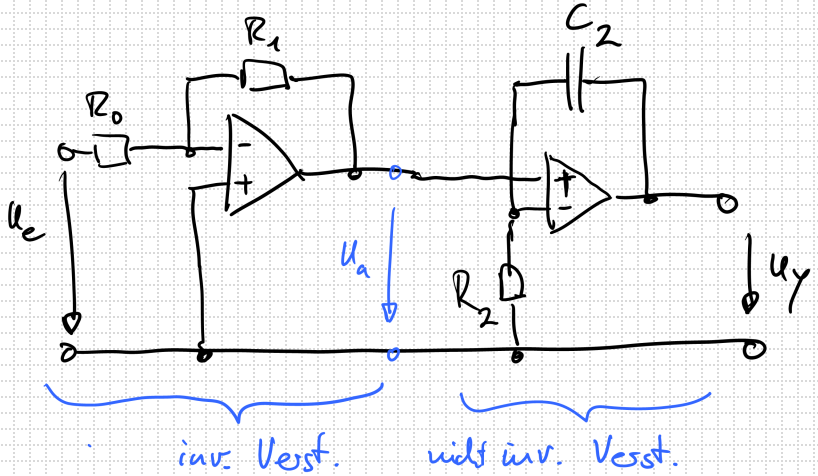

Sp.-Teiler:

$$u_e = \frac{z_1}{z_1 + z_2} \cdot u_a$$

$$\boxed{\frac{u_a}{u_e} = \frac{z_1 + z_2}{z_1} = 1 + \frac{z_2}{z_1}}$$

$\rightarrow$ mit $z_1 = R_2$ und $z_2 = \frac{1}{j\omega C_2}$: $\frac{u_a}{u_e} = 1 + \frac{1}{j\omega R_2 C_2}$

c) Serienschaltung ergibt:

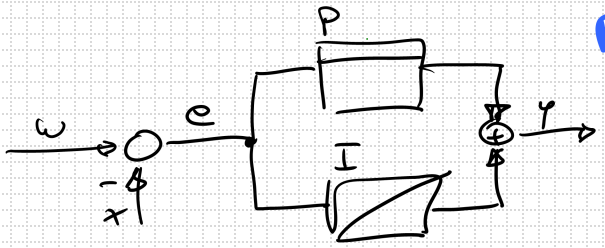



$$\frac{u_y}{u_e} = \frac{u_a}{u_e} \cdot \frac{u_y}{u_a} = -\frac{R_1}{R_0} \cdot \left(1 + \frac{1}{j\omega R_2 C_2}\right) \quad \Rightarrow \text{PI-Regler}$$

$V_P = -\frac{R_1}{R_0}$ $\qquad$ $T_N = R_2 C_2$

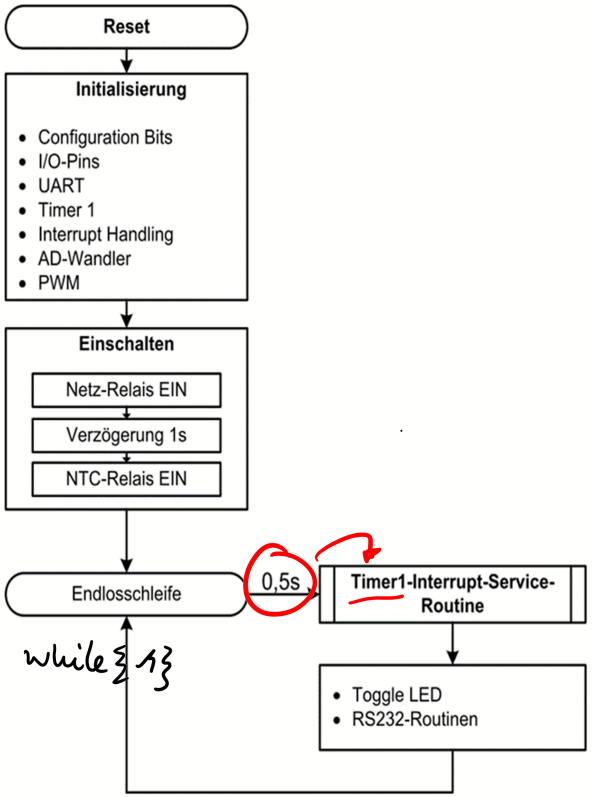
- digitale Implementierung

$$y = e \cdot V_p\left(1 + \frac{1}{sT_N}\right) = \underbrace{V_p \cdot e}_{P} + \underbrace{\frac{V_p}{T_N}}_{V_I} \cdot e \cdot \underbrace{\frac{1}{s}}_{I}$$




→ Echtzeitregelung bedeutet interruptgesteuerte Programme

⇒ Regelung läuft innerhalb einer ISR (Interrupt Service Routine) ab.



Interrupt des ADC, sobald der neue Messwert vorliegt
→ wird üblicherweise mit fPWM gemessen

while {} → ADC - ISR / Regelungs - ISR
↳ kann auch vom Modulator getriggert sein

Codebeispiele in C

Sollwert, aktueller Messwert

```
e = w - x;          // Regeldifferenz berechnen
y = Vp · e + VI · esum;   // PI - Algorithmus
esum = e + esum;    // Fehlersummenbildung (Integral)
```

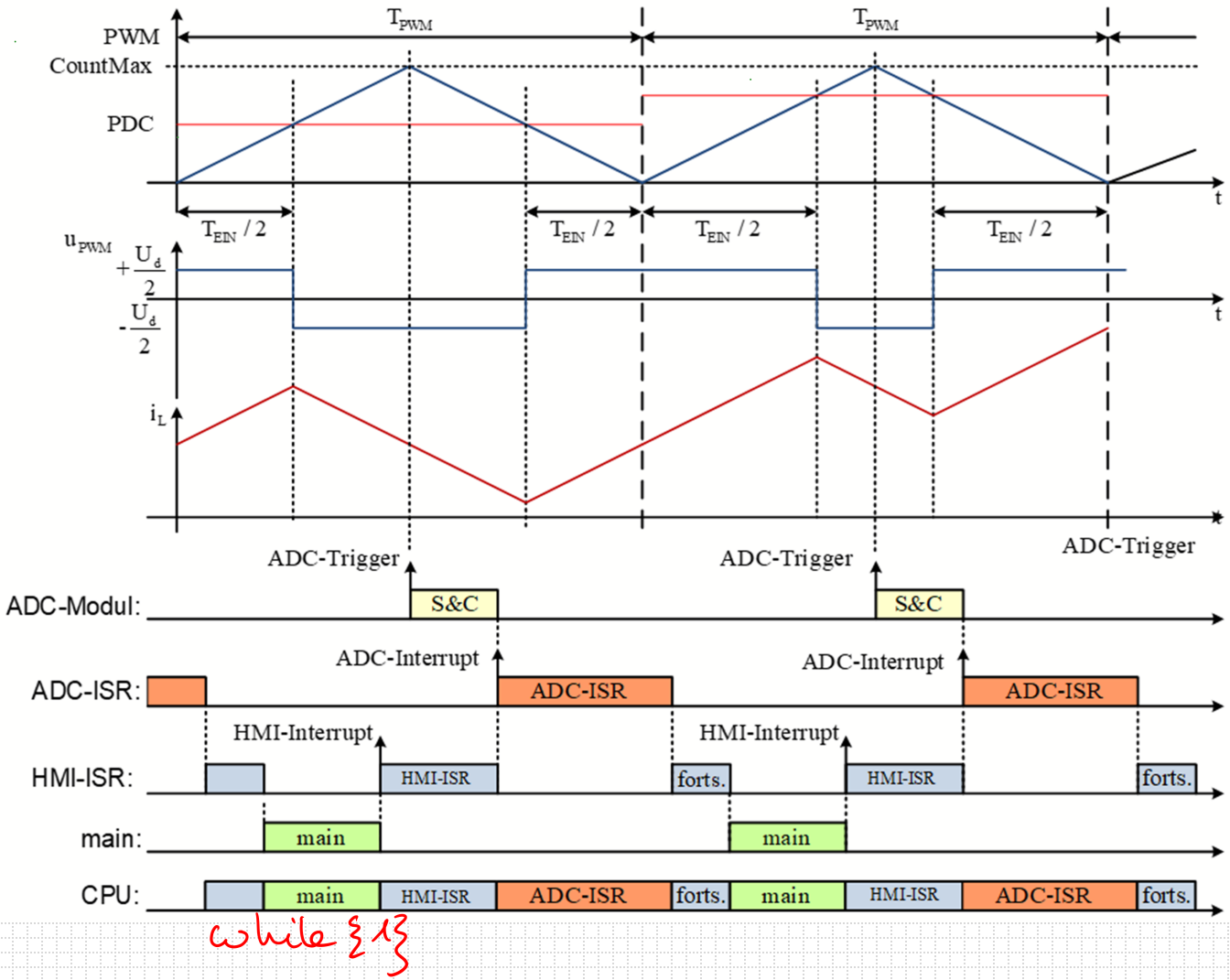
Fehlersumme im Takt k:

$$e_{sum,k} = e_{sum,k-1} + e_k$$

in C: `esum += e;`

# Timing – zeitliche Ablaufsteuerung

z.B. 20µs



# Bitshift – schnelle Division

- Division dauert rel. lange → in Echtzeitsignalverarbeitung vermeiden!
- Statt Division (oder Floating-Point-Multiplikation)
  1) Multiplikation mit Kehrwert
  2) Bitshift nutzen
     - Verschieben der Bits um $n$ Stellen — in C
       - nach rechts $\hat{=}$ Division durch $2^n$ — "$>> n$"
       - nach links $\hat{=}$ Multiplikation mit $2^n$ — "$<< n$"
     - Bsp:

       $1\ 1\ 1\ 0 \;\hat{=}\; 14 \quad | \gg 1$

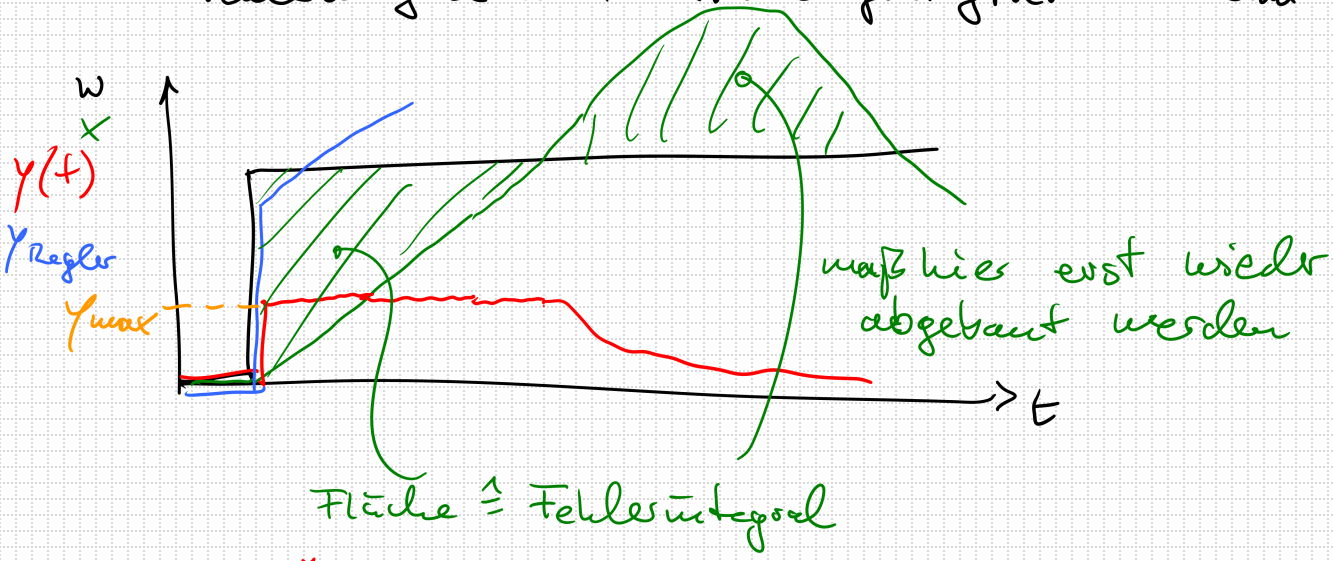
       $0\ 1\ 1\ 1 \;\hat{=}\; 7$
     - Zeitaufwand: 1 Takt!

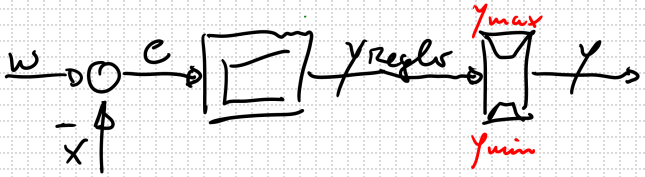
Bsp: $V_p = 4{,}6 \approx 4{,}625 = \frac{37}{8} = \frac{37}{2^3} \;\hat{=}\; \cdot 37 \gg 3$

$V_p \cdot e \;\hat{=}\; (37 \cdot e) \gg 3$ Dauer: 2 Takte

# Wind-up-Effekt

- Reale Stellglieder sind begrenzt ⇒ Stellgrößenbegrenzung
- Wird vom Regler eine Stellgröße gefordert, die nicht innerhalb der Grenzen liegt
  - Regler kann nicht arbeiten
  - Großsignalbereich
  - Fehlerintegral wird weiter aufintegriert → "Wind-up"-Effekt

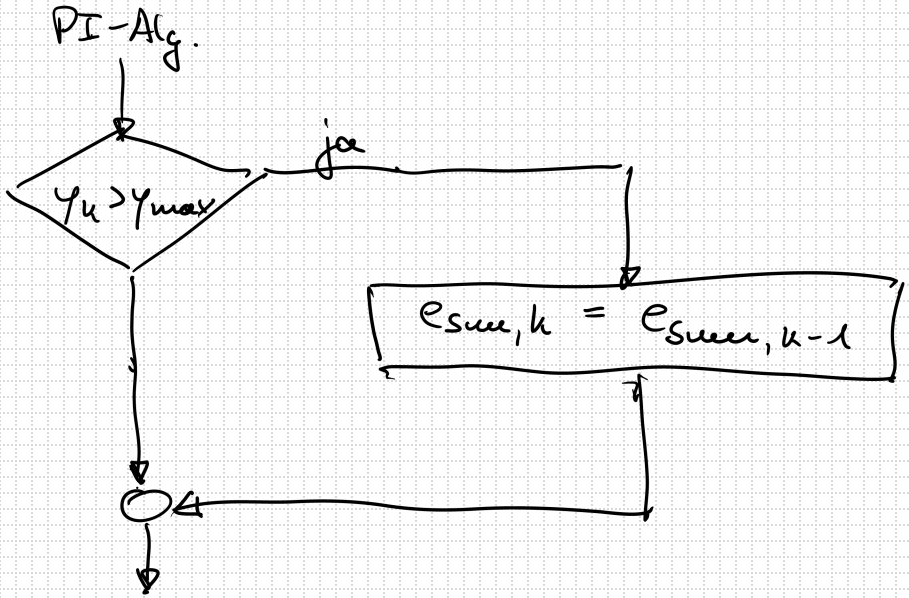


## Anti-Wind-up-Maßnahmen

a) I-Anteil anhalten ("Clamping")

Nach der Berechnung der neuen Stellgröße $y_k$ in Takt $k$:



## Anti-Wind-up-Maßnahmen

b) I-Anteil zurückrechnen ("Back Calculation")

in Abh. von $y_k - y_{max}$

Nach der Berechnung der neuen Stellgröße $y_k$ im Takt $k$: