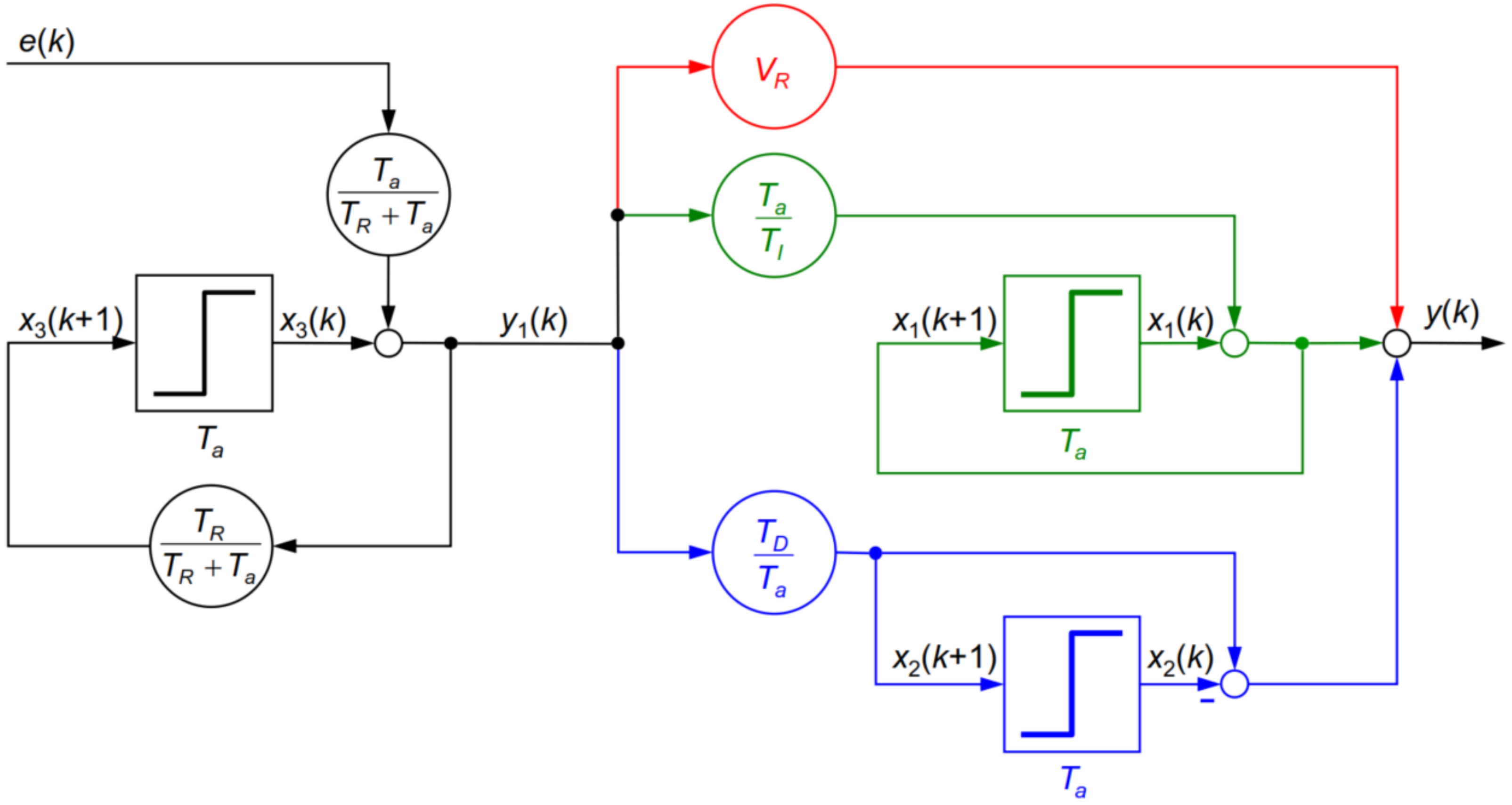
Regelungstechnik

für BEI4, BMEI4, BMF6

Prof. Dr. B. Wagner



Kapitel 9
Zeitdiskrete Regelung
Der Regler als Programmcode

Grundlagen der Regelungstechnik

Kap. 1: Was ist eine Regelung? Was ist ein Regelkreis?
Kap. 2: Was ist eine Übertragungsfunktion? Was kann ich aus ihr sehen?
Kap. 3: Was ist ein Frequenzgang? Wie zeichne ich ein Bode-Diagramm?
Kap. 4: Wie finde ich in der Praxis die Strecken-Übertragungsfunktion $G_S(s)$?
Kap. 5: Welche Standard-Regler gibt es?
Kap. 6: Wie ist das mit der Stabilität? Was wollen uns die Herren Nyquist + Evans (WOK) sagen?
Kap. 7: Wie stelle ich einen Regler sinnvoll ein?
Kap. 8: Welche Erweiterungen des einfachen („einschleifigen") Regelkreises gibt es?

$\Rightarrow$ Fertig mit der Theorie !

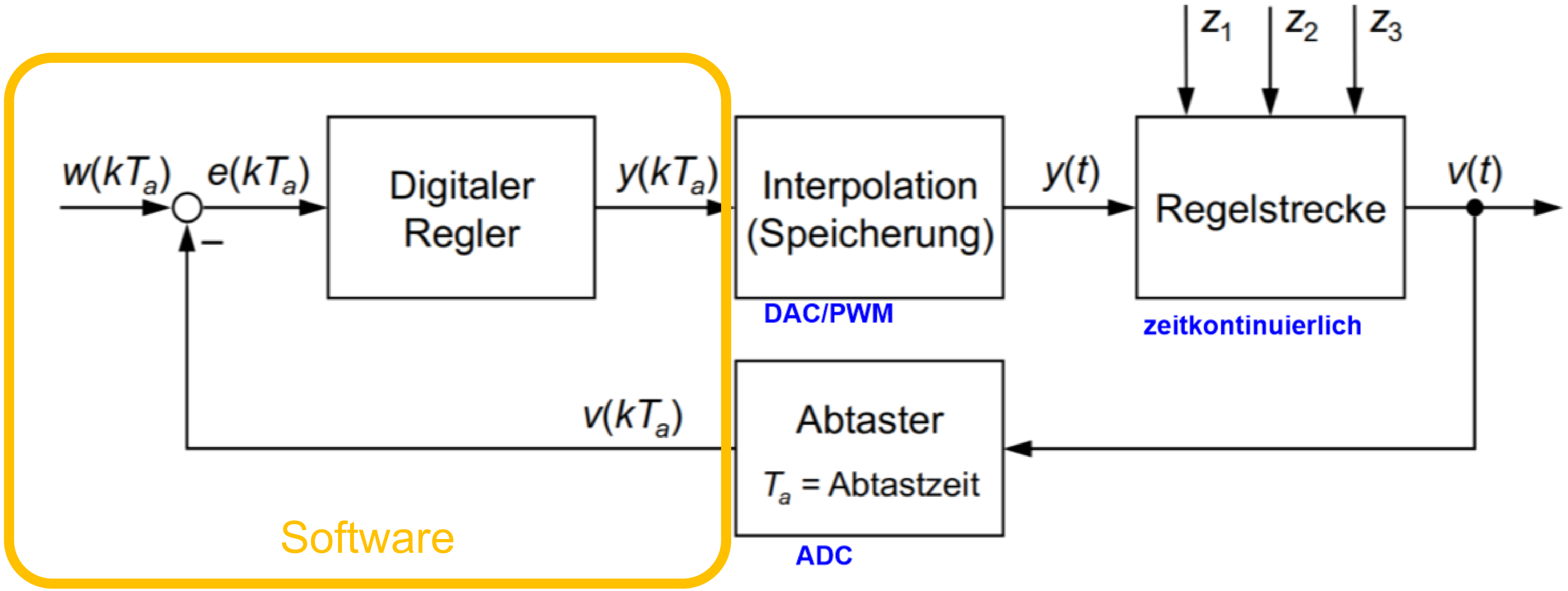
Nun noch: Wie realisiere ich eine Regelung? Wie programmiere ich einen Regler $G_R(s)$ in Software?

$\Rightarrow$ Entwurfsstrategien für zeitdiskrete Software-Regler

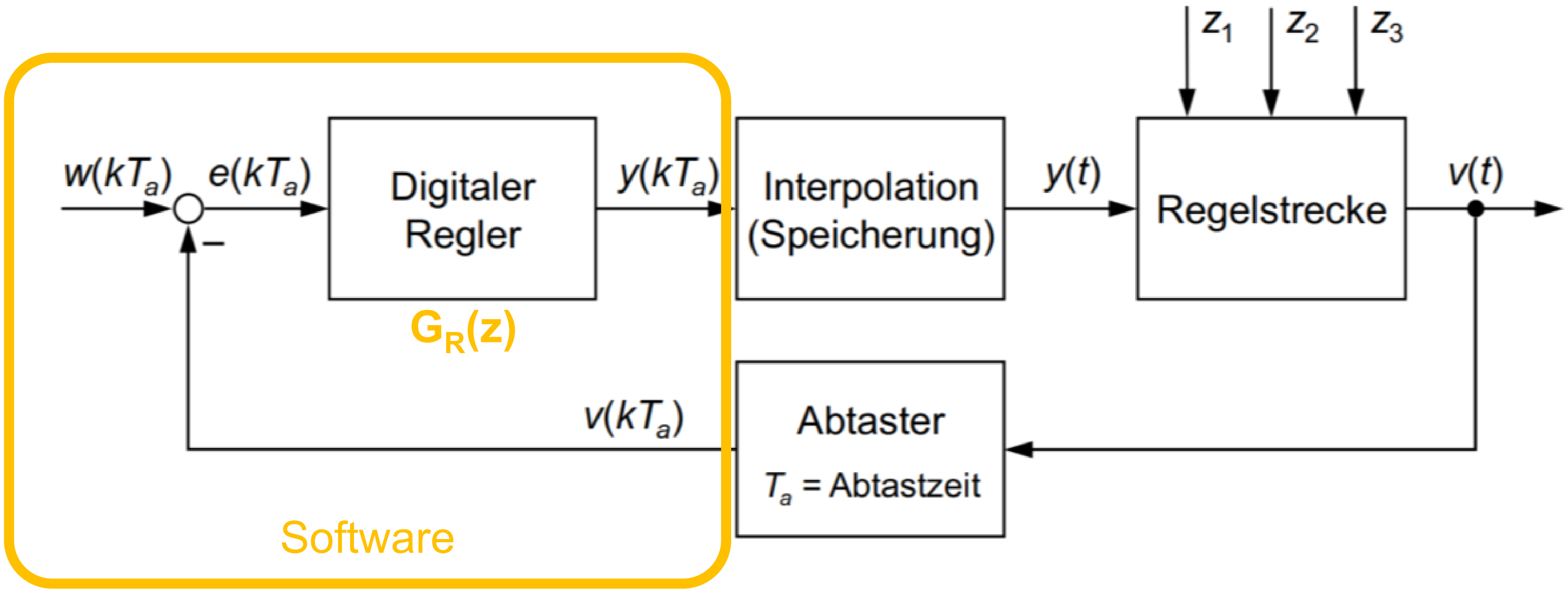
$\Rightarrow$ Diskretisierungsmethoden

$\Rightarrow$ Umsetzung einer (Regler-)Übertragungsfunktion in einer Programmiersprache

Blockschaltbild einer digitalen Regelung



Blockschaltbild einer digitalen Regelung



Zeitdiskrete Systeme

- Statt Integratoren (1/s) treten Verzögerungsglieder auf (1/z)
- „1/z“ steht für eine Verzögerung um einen Abtasttakt
- Stabilität: alle Pole eines stabilen Abtastsystems liegen im Einheitskreis

Entwurf einer zeitdiskreten Regelung $G_R(z)$

Ausgangspunkt: zeitkonti. Strecke $G_S(s)$

"quasikontinuierliche zeitdiskrete Regelung"

Zeitkonti. Reglerentwurf $G_R(s)$

SO
BO
Amplituden-/Phasenreserve
Chien, Hrones, Reswick

Regler-diskretisierung $G_R(z)$

T_ab = Abtastzeit
--> "klein"

Reglerrealisierung

"echt zeitdiskrete Regelung"

Strecken-diskretisierung $G_S(z)$

Zeitdiskreter Reglerentwurf $G_R(z)$

T_ab = Abtastzeit
--> "groß"

Programmierung

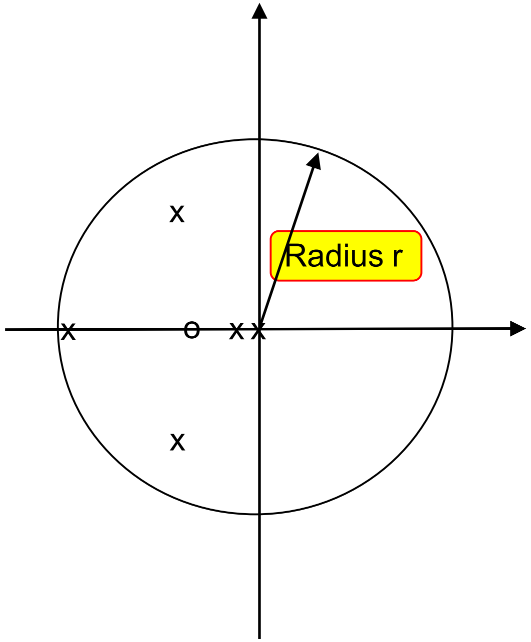
Quasikontinuierliche Regelung

Zeitdiskrete Regelung soll sich „auf Strichstärke“ wie zeitdiskrete Regelung verhalten

Dafür muss sich die Abtastzeit **Ta** an der Dynamik von Strecke und Regler orientieren.

Faustformel als Anhaltspunkt:

man erhält quasikontinuierliches Verhalten der Regelung
für $\boxed{\mathbf{Ta \leq 0.1 / r}}$ mit r = Radius eines Kreises in der s-Ebene um den Nullpunkt, der alle Pole und Nullstellen von Regler $G_R(s)$ und Strecke $G_S(s)$ einschließt



Die „Faustformel“ wird in der Praxis häufig verletzt
$\Rightarrow$ Stabilitäts-/ Dynamik-Kontrolle (Rechnung / Simulation) durchführen!!!

Tipp für die Praxis:
Lassen Sie die Abtast- und Rechenzeit als zusätzliche Totzeit in den Reglerentwurf eingehen (=> in Summe kleiner Zeitkonstanten T_σ)

Beispiel 1

Tempomat-Regelstrecke aus dem Praktikum mit Kompensationsregler aus dem 3. Versuch:

$$G_S(s) = \frac{1{,}71}{1 + 13s + 22s^2}; \qquad G_R(s) = \frac{1 + 13s + 22s^2}{4{,}38s + 4{,}12s^2}$$

$\Rightarrow$ Bestimme alle Pole und Nullstellen der Strecke und des Reglers

$\Rightarrow$ Trage alle Pole und Nullstellen in komplexe Ebene ein

$\Rightarrow$ Bestimme Radius

$\Rightarrow$ Alternativ: rechnerische Lösung = Bestimme alle Beträge aller Pole und Nullstellen und suche maximalen Betrag

$\Rightarrow$ Bestimme Ta nach der Faustformel!

Beispiel 1

Tempomat-Regelstrecke aus dem Praktikum mit Kompensationsregler aus dem 3. Versuch:

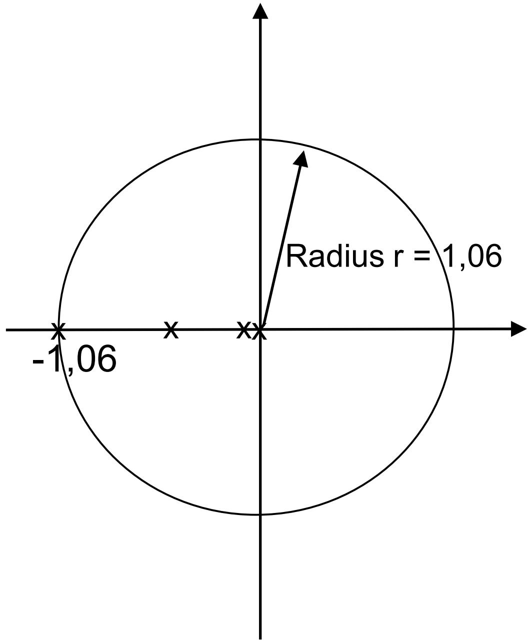
$$G_S(s) = \frac{1{,}71}{1 + 13s + 22s^2}; \qquad G_R(s) = \frac{1 + 13s + 22s^2}{4{,}38s + 4{,}12s^2}$$

$\Rightarrow$ Bestimme alle Pole und Nullstellen der Strecke und des Reglers 0; -0,5; -0,091; -1,06

$\Rightarrow$ Trage alle Pole und Nullstellen in komplexe Ebene ein

$\Rightarrow$ Bestimme Radius

r = 1,06

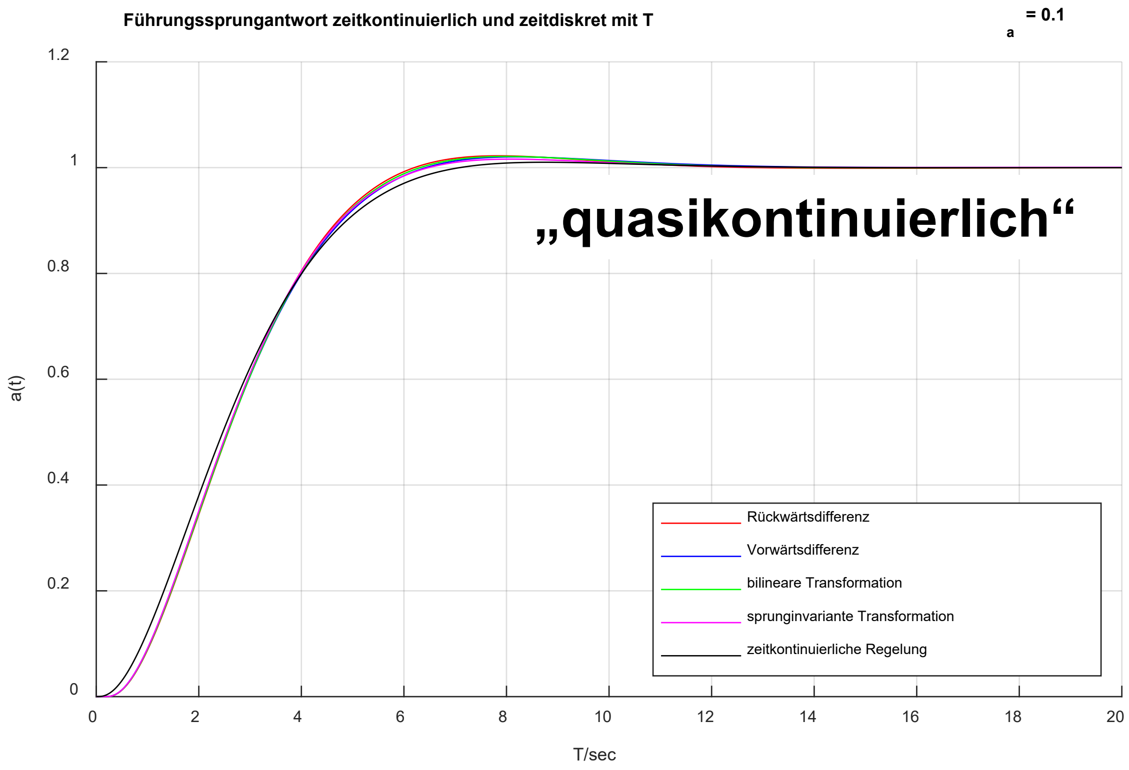


$\Rightarrow$ Alternativ: rechnerische Lösung = Bestimme alle Beträge aller Pole und Nullstellen und suche maximalen Betrag

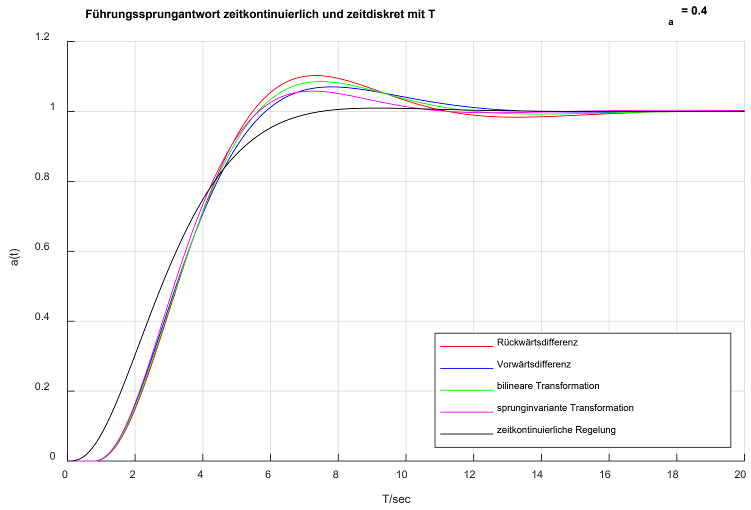
$\Rightarrow$ Bestimme Ta nach der Faustformel!

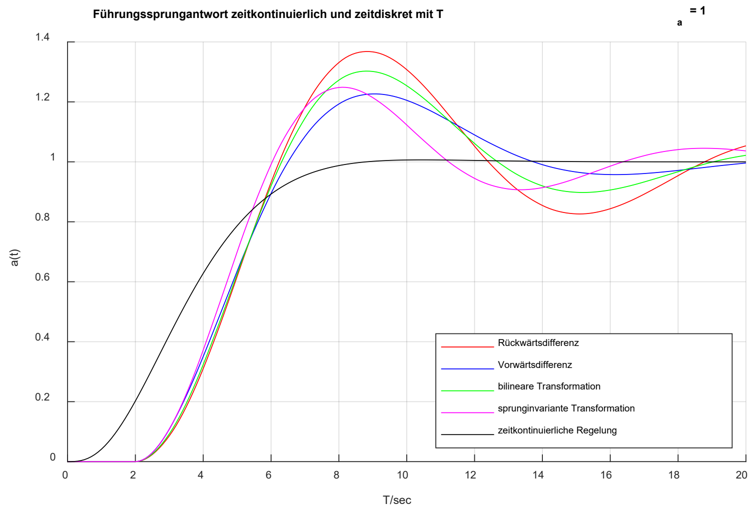
$T_a \le 0{,}1/1{,}06 = 0{,}094$

Beispiel 1: Regelkreissimulation mit unterschiedlichen T_a



Verhalten bei Verletzen der Faustformel:





Bei diesen Simulationen wurden die Reglerkoeffizienten nicht an die Wahl der Abtastzeit angepasst.

Diskretisierungsmethoden

Vorwärtsdifferenz = „explizite Euler-Diskretisierung“

$\Rightarrow$ die einfachste und in der Praxis die beliebteste Methode

Rückwärtsdifferenz = „implizite Euler-Diskretisierung“

$\Rightarrow$ etwas unempfindlicher gegen größere T_a

Bilineare Transformation

$\Rightarrow$ Standard-Methode für den Entwurf von Digitalfiltern
$\Rightarrow$ bietet in der Praxis meist im geschlossenen Regelkreis keine großen Vorteile

Sprunginvariante Transformation

$\Rightarrow$ Numerisch aufwendige Rechnung
$\Rightarrow$ Tipp: Funktion „c2d“ in Matlab (c2d = continuous to discrete mit Methode „tustin“ als Option)
$\Rightarrow$ Methode der Wahl für den „echt zeitdiskreten Entwurf“ einer Abtastregelung (Streckendiskretisierung => AUT5)

Bei quasikontinuierlicher Regelung mit **$Ta \leq 0.1/r$** ist in der Regel die Diskretisierungsmethode nebensächlich!

Vorwärtsdifferenz (1)

Bsp: PT1

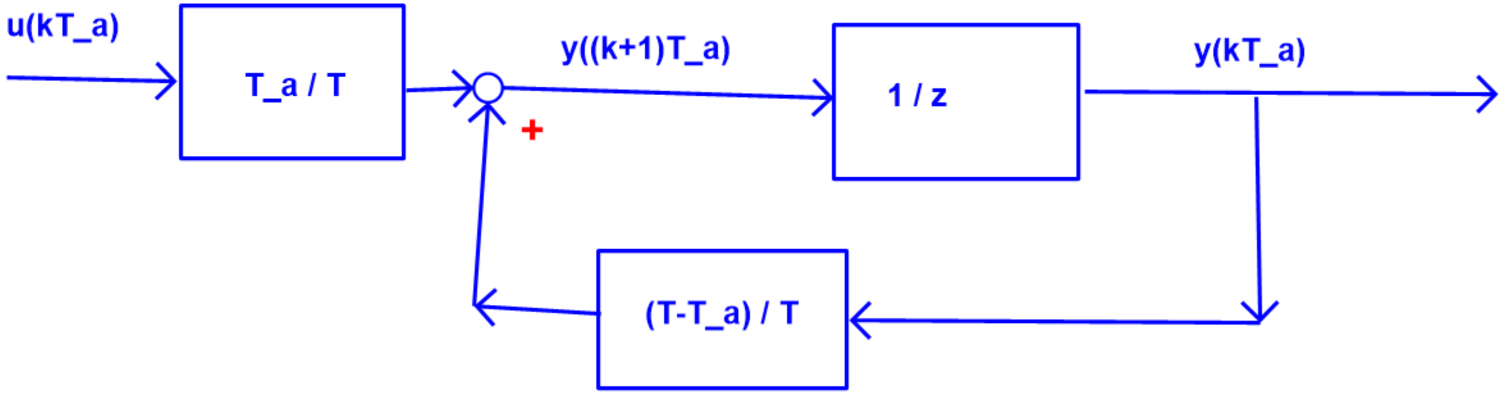
$$G(s) = \frac{1}{1+sT}$$

$$\rightarrow \dot{y}(t) = \frac{1}{T} \cdot \left(-y(t) + u(t)\right)$$

$$\text{Näherung: } \frac{y((k+1) \cdot T_a) - y(kT_a)}{T_a} = \frac{1}{T} \cdot \left(-y(kT_a)\right) + \frac{1}{T} \cdot u(kT_a)$$

$$\rightarrow y\left((k+1) \cdot T_a\right) = y(kT_a) \cdot \left(1 - \frac{T_a}{T}\right) + \frac{T_a}{T} \cdot u(kT_a)$$

„neu“ = „alt“ + „alt“



Vorwärtsdifferenz (2)

z-Übertragungsfunktion

$$G(z) = \frac{\frac{T_a}{T} \cdot \frac{1}{z}}{1 - \frac{T - T_a}{T \cdot z}} = \frac{\frac{T_a}{T}}{z - \frac{T - T_a}{T}}$$

alternative Methode: Aus $G(s)$ folgt $G(z)$ mit Ersetzen: $s = \frac{z-1}{T_a}$

Rückwärtsdifferenz (1)

$$\dot{y}(t) = \frac{1}{T}\cdot\left(u(t) - y(t)\right)$$

$$\frac{y(k\cdot T_a) - y((k-1)\cdot T_a)}{T_a} = \frac{1}{T}\cdot u(k\cdot T_a) - \frac{1}{T}\, y(k\cdot T_a)$$

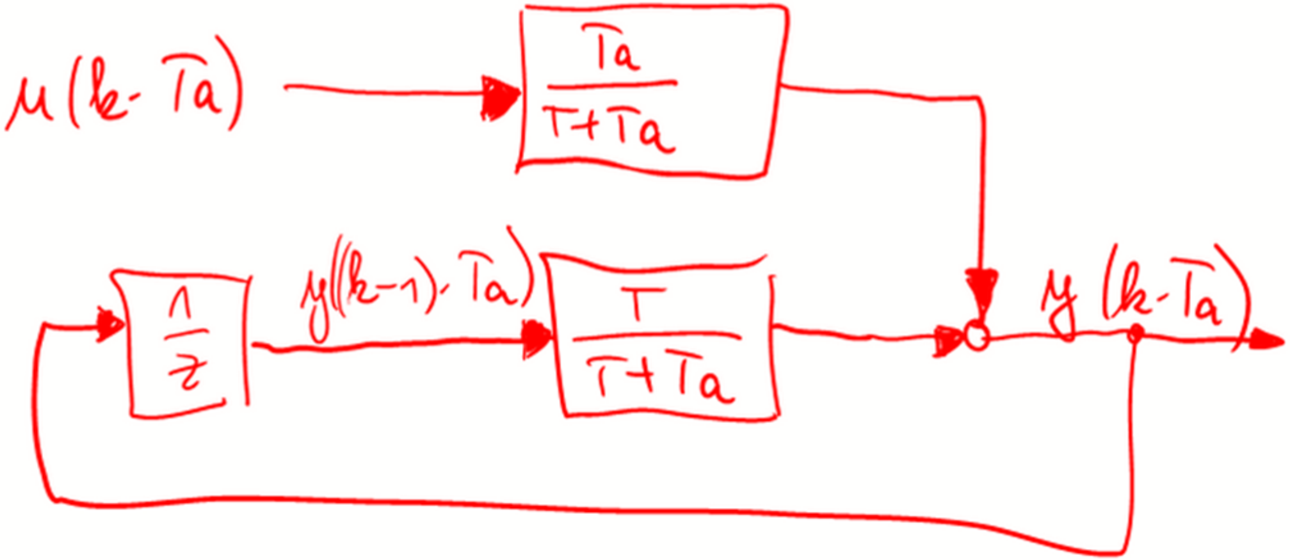
--> implizite Gleichung!

$$y(k\cdot T_a) - y((k-1)\cdot T_a) = \frac{T_a}{T}\cdot u(k\cdot T_a) - \frac{T_a}{T}\cdot y(k\cdot T_a)$$

$$y(k\cdot T_a)\left(1 - \frac{T_a}{T}\right) = \frac{T_a}{T}\cdot u(k\cdot T_a) + \frac{T_a}{T} y((k-1)\cdot T_a)$$

Rückwärtsdifferenz (2)

$$y(k \cdot T_a) = \frac{T_a}{T+T_a} \cdot u(k \cdot T_a) + \frac{T}{T+T_a} \cdot y((k-1) \cdot T_a)$$



$$G(z) = \frac{\frac{T_a}{T+T_a}}{1 - \frac{T}{z \cdot (T+T_a)}}$$

$$= \frac{z \cdot \frac{T_a}{T+T_a}}{z - \frac{T}{T+T_a}}$$

Aus $G(s)$ folgt $G(z)$ mit Ersetzen: $s = \frac{1-z^{-1}}{T_a}$

Die bilineare Transformation

→ Taylorreihe: $s \approx \frac{2(z-1)}{T_a(z+1)}$

Die bilineare Transformation…
⇒ … ist DIE Transformation für Digitalfilter (z. B. Frequenzweiche…)
⇒ … hat in der Praxis bei quasikontinuierlicher Regelung keine nennenswerten Vorteile
⇒ … wird deswegen in der Praxis kaum bis gar nicht eingesetzt zur Diskretisierung von Reglern!

Fazit:
⇒ Meist wird die Vorwärtsdifferenz, manchmal auch die Rückwärtsdifferenz verwendet.

Umrechnung im Frequenzbereich: ersetze „s“ durch …

Vorwärtsdifferenz	Rückwärtsdifferenz	Bilineare Transformation
$s \Leftrightarrow \frac{z-1}{T_a}$	$s \Leftrightarrow \frac{z-1}{z\,T_a}$	$s \Leftrightarrow \frac{(z-1)2}{(z+1)\,T_a}$

Beispiel: $G(s) = \frac{6s+2}{2s+1}$ **Abtastzeit Ta = 0,25**

$$G(z) = \frac{6 \cdot \frac{z-1}{0{,}25} + 2}{2 \cdot \frac{z-1}{0{,}25} + 1} = \frac{24z - 22}{8z - 7}$$

$$G_{vor}(z) = \frac{24z - 22}{8z - 7}$$

$$G_{rueck}(z) = \frac{26z - 24}{9z - 8}$$

$$G_{bil}(z) = \frac{50z - 46}{17z - 15}$$

keine Näherung wie bei Methoden davor!

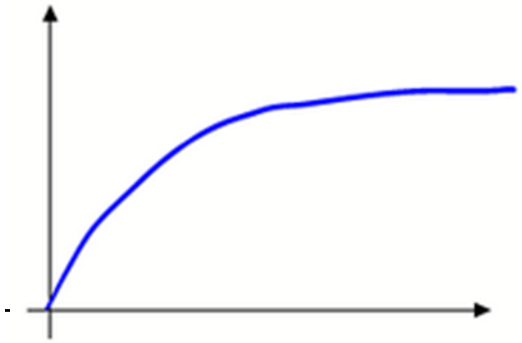
Idee: Sprungantwort im Zeitdiskreten stimmt an den Abtastwerten überein mit der Sprungantwort des zeitkontinuierlichen Systems.

Vorgehen: $A(s) \rightarrow a(t) \rightarrow (t = kT_a) \rightarrow A(z) \rightarrow G(z)$

(bei $\sigma(t)$ nur $t = k$!!)

Herleitung am Beispiel eines PT$_1$-Systems $G(s) = \frac{1}{1+sT}$

$\rightarrow G(z) = \frac{z-1}{z} \cdot A(z)$



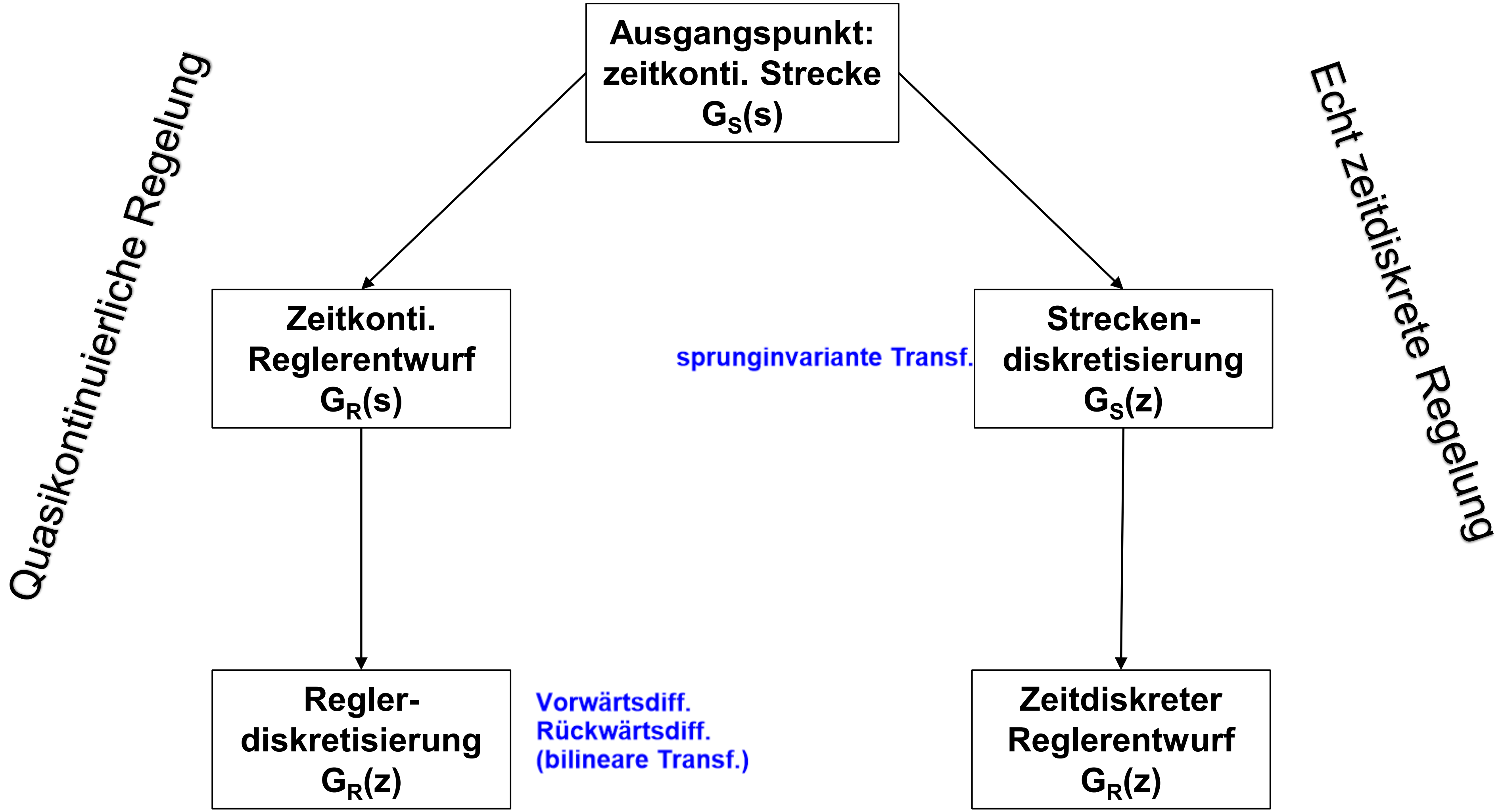
zeitkontinuierlich: $a(t) = 1 - e^{-t/T}$

Abtastung: $a(k\cdot T) = 1 - e^{-k\cdot T_a/T}$

z-Transformation: $A(z) = \frac{z}{z-1} - \frac{z}{z-c}$ mit $c = e^{-T_a/T}$

Hauptnenner $A(z) = \frac{z\cdot(1-c)}{(z-1)\cdot(z-c)} = \underbrace{\text{Sprung}(z)}_{=\frac{z}{z-1}} \cdot G(z) \Rightarrow \boxed{G(z) = \frac{1-c}{z-c}}$

Entwurf einer zeitdiskreten Regelung $G_R(z)$



Der Integrator im Kontinuierlichen und im Zeitdiskreten

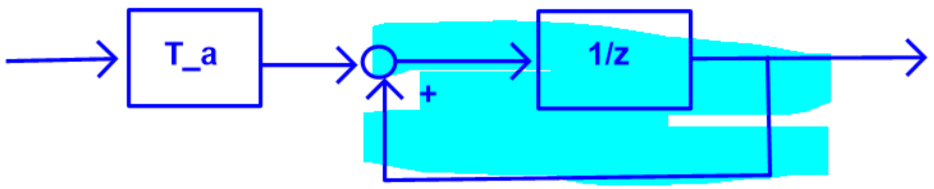
$$G(s) = \frac{1}{s}$$

Merkmal eines integrierenden Systems im Zeitdiskreten:

- immer Rückführung mit Faktor +1

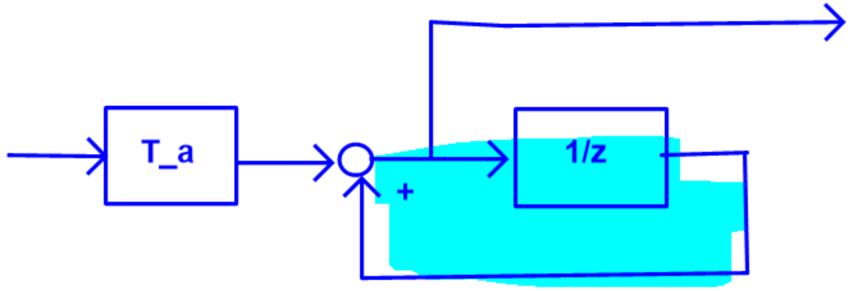
Vorwärtsdifferenz:

T_a / (z-1)



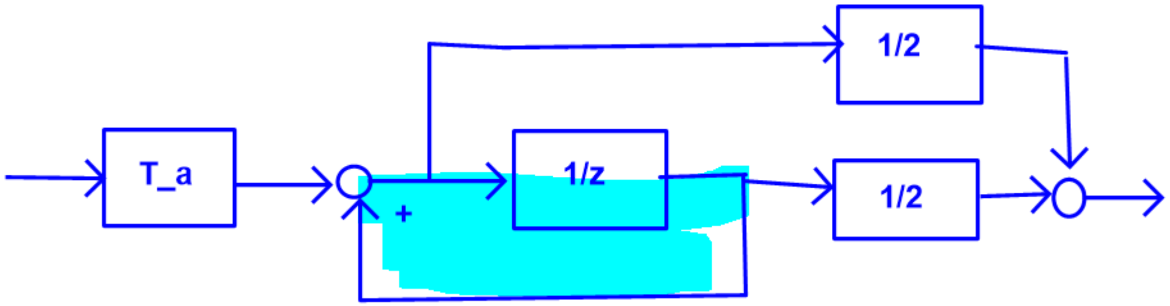
Rückwärtsdifferenz:

(z*T_a) / (z-1)



Bilineare Transformation:

$$\frac{(z+1)\cdot T_a}{(z-1)\cdot 2}$$



Der Integrator im Kontinuierlichen und im Zeitdiskreten

Technische Hochschule Nürnberg

??? Merkmal eines Systems mit I-Anteil…

… im Zeitkontinuierlichen:

--> Pol bei s = 0

… im Zeitdiskreten:

--> Pol bei z = +1
--> Summe der Koeffizienten im Nenner = 0 ?

$$G(s) = \frac{2s - 1}{s(2s+3)} \qquad G(s) = \frac{17s + 2}{s^3 + 2s^2 + 3s} \qquad G(z) = \frac{17z + 15}{z^3 - 2z^2}$$

$$G(\overset{z}{\cancel{s}}) = \frac{z^3 + 1{,}7z^2 - 0{,}18z - 0{,}72}{z^4 - 3{,}7\underset{z}{\cancel{s}}^3 + 5{,}12\underset{z}{\cancel{s}}^2 - 3{,}14\underset{z}{\cancel{s}} + 0{,}72}$$

--> kein I-Anteil da
Summe der Koeffizienten
nicht 0 bzw. bei
Einsetzen von z = 1 wird
Nenner nicht 0

--> I-Anteil

Der Differenzierer im Zeitkontinuierlichen und im Zeitdiskreten

$$G(s) = s$$

--> nicht realisierbar

Vorwärtsdifferenz:

G(z) = (z-1) / T_a

--> nicht kausal --> nicht realisierbar!

Rückwärtsdifferenz:

G(z) = (z-1) / z*T_a

--> realisierbar, da Zählergrad = Nennergrad

in der Praxis trotzdem DT1, um Rauschen zu unterdrücken

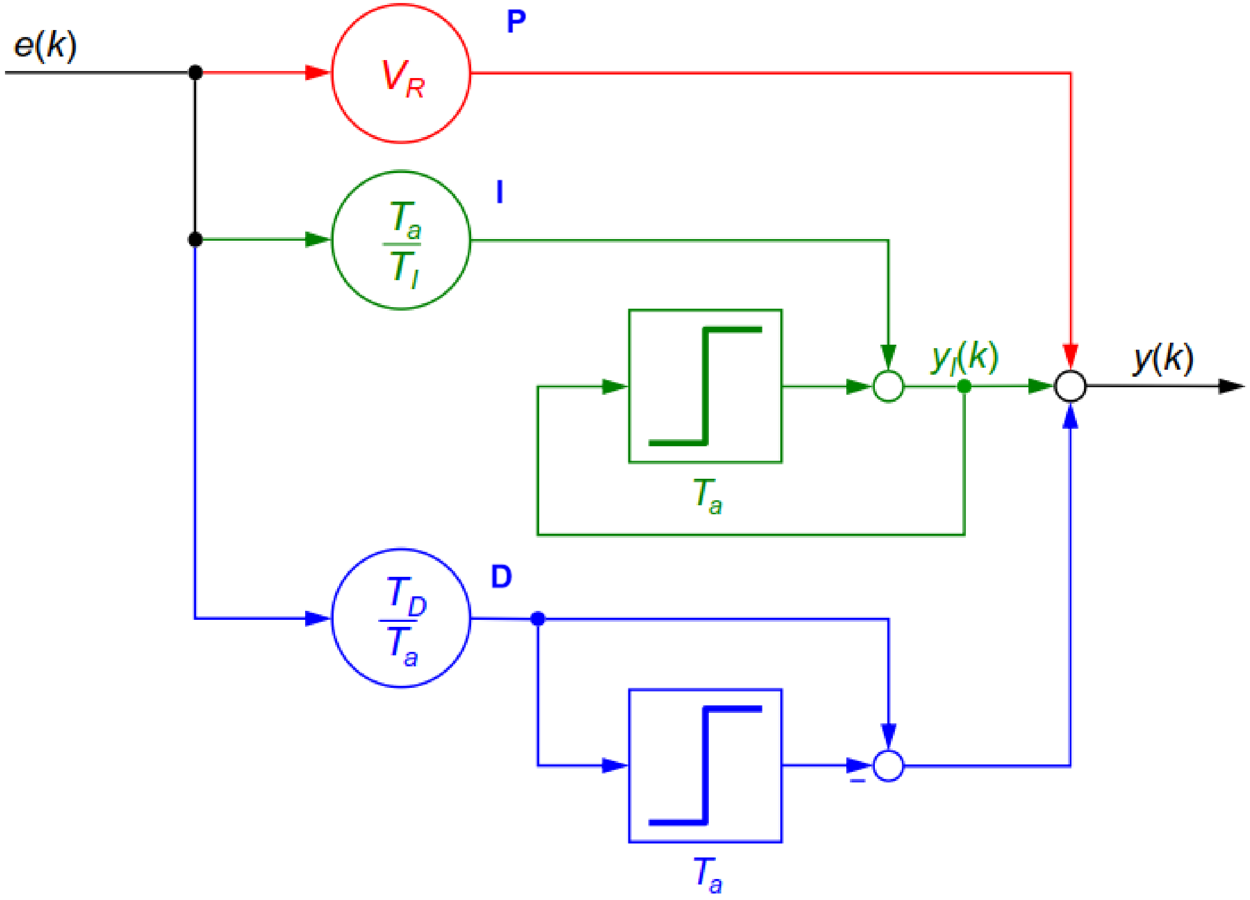
Bilineare Transformation:

$$\frac{(z-1)\cdot 2}{(z+1)\cdot T_a}$$

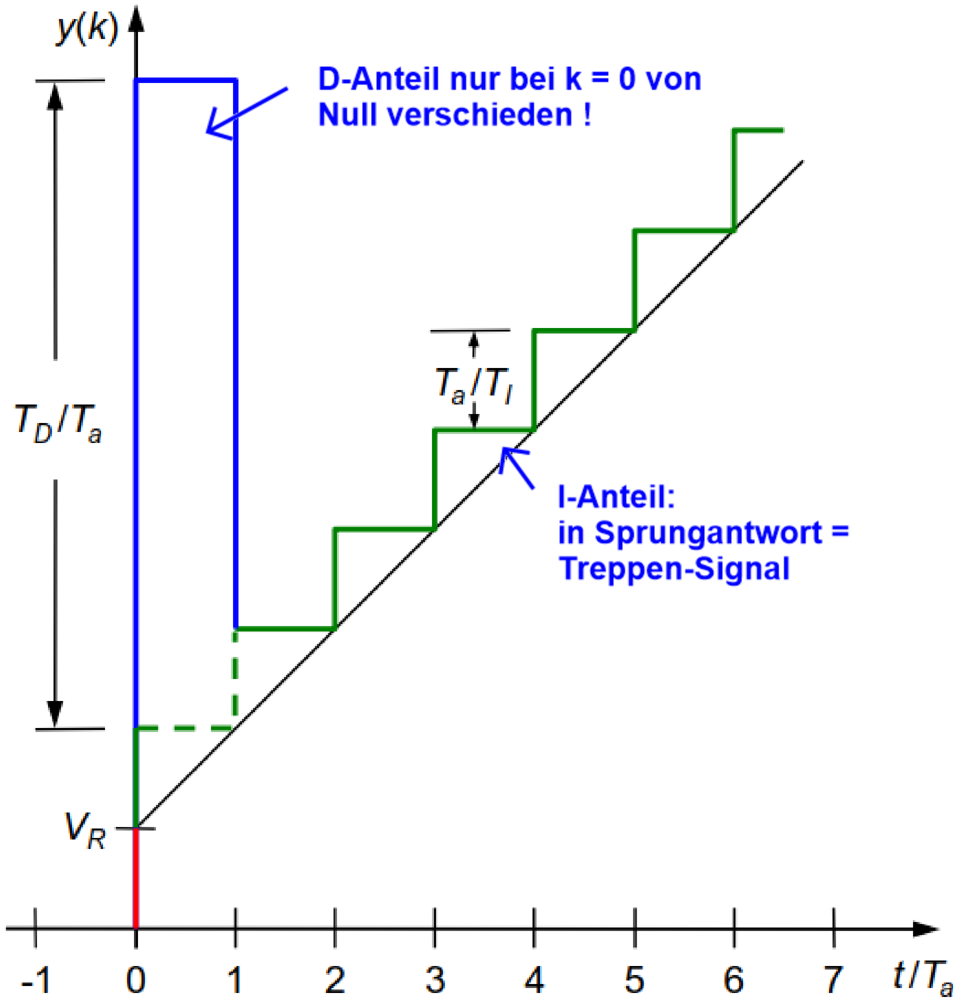
--> realisierbar, aber idR nicht verwendet

Der PID-Regler im Zeitdiskreten
⇒ realisierbar mit der Rückwärtsdifferenz!

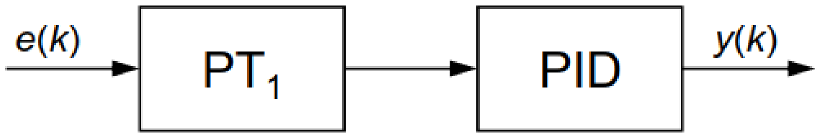
Strukturbild:



Sprungantwort:

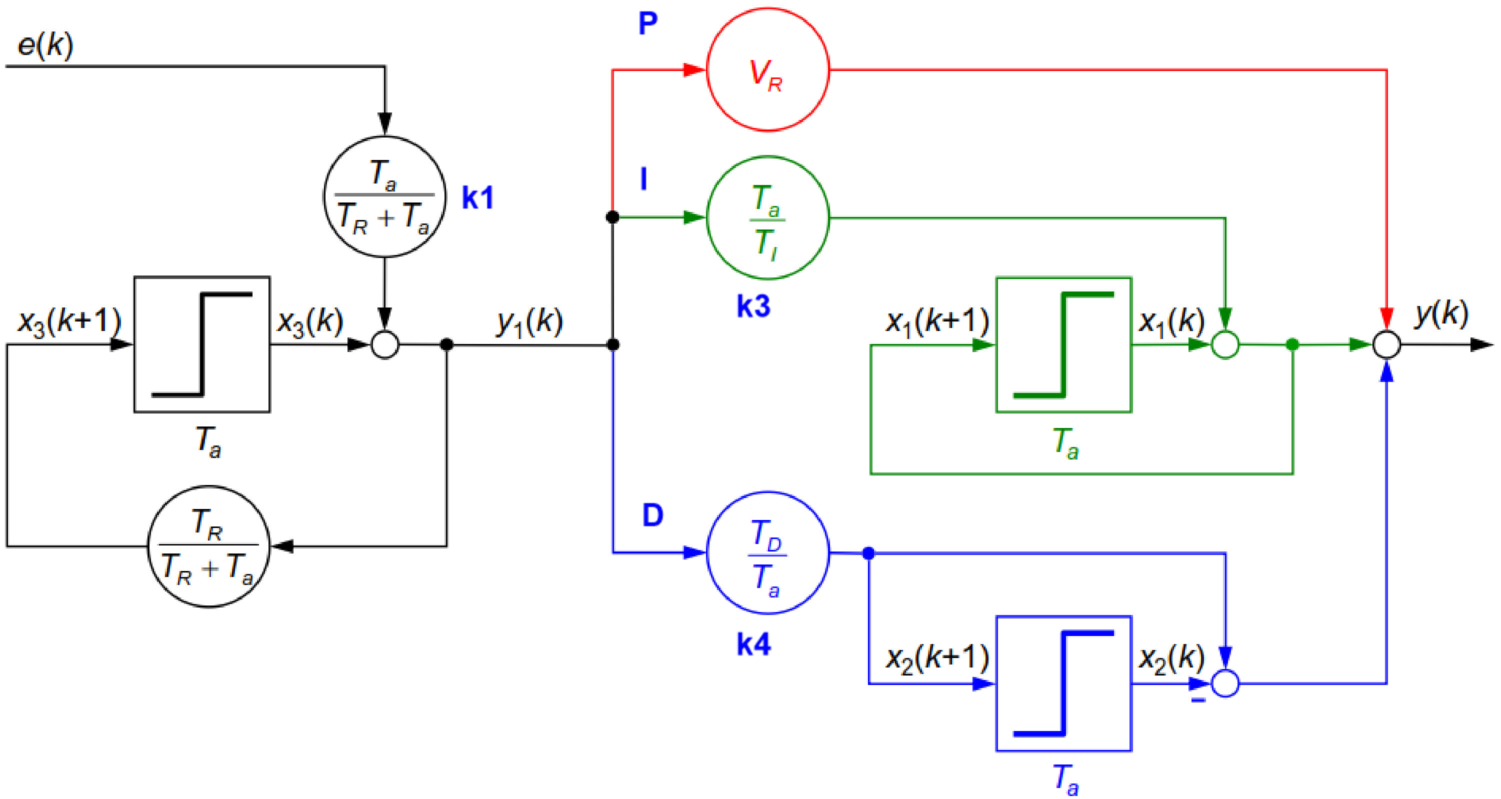


Der zeitdiskrete PIDT$_1$-Regler als Programm



$$G_R(s) = V_R \cdot \frac{1+\frac{1}{sT_N}+sT_V}{1+sT_R} = \frac{1}{1+sT_R}\cdot V_R\left(1+\frac{1}{sT_N}+sT_V\right) = \underbrace{\frac{1}{1+sT_R}}_{PT_1\text{ - Teil}} \cdot \underbrace{\left(V_R+\frac{1}{sT_I}+sT_D\right)}_{PID\text{ - Teil}}$$

PT1:




```
// Initialisierung von Variablen und
Parametern
void init(void){
Ta = ...;
TR = ...;
TI = ...;
TD = ...;
VR = ...;

k1 = Ta/(TR+Ta);
k2 = ...; k3=...; k4 = ...;

x1k = 0;
x2k = 0;
x3k = 0;
}

void TimerInterrupt(void) {

// Einlesen der Regelabweichung
ek = Read_Ek();

// Bestimme Ausgangssignal
y1k = k1* ek + x3k;
yk = (VR + k3 + k4)*y1k + x1k - x2k;
StellsignalAusgabe(yk);

// Aktualisieren der Reglerzustände
x1neu = x1k + k3*y1k;
x2neu = k4*y1k;
x3neu = y1k*k2;

// Umspeichern
x1k = x1neu;
x2k = x2neu;
x3k = x3neu;
}
```