



Prof. Dr. S. Zacher

# Model Discretizer von MATLAB

Hinweise zur Digitalisierung von analogen Simulink-Modellen

“Model Discretizer selectively replaces continuous Simulink® blocks with discrete equivalents. Discretization is a critical step in digital controller design and for hardware in-the-loop simulations.”

*MathWorks, Model Diskretizer* <http://de.mathworks.com/help/simulink/ug/model-discretizer.html>

## Abstract, Urheberrechts- und Haftungshinweis

Ab Versionen R2014 und R2015 ermöglicht MATLAB/Simulink die Digitalisierung von Simulink-Modellen mit den Menü-Befehlen

*Analysis* → *Control Design* → *Model Discretizer*

automatisch durchführen.

Ein Simulink-Modell wird damit automatisch von Laplace-Transformierten Übertragungsfunktionen  $G(s)$  in die z-Transformierte Übertragungsfunktionen  $G(z)$  umgewandelt. Dies erkennt man sofort durch den Schriftzug *zoh*, der auf allen digitalisierten Bausteinen nach der Ausführung des *Model Discretizer* erscheint.

Nachfolgend werden an einem Beispiel zwei Wege der Digitalisierung beschrieben: mit dem o.g. *Model Discretizer* und nach der z-Transformation.

---

Die vorliegende Publikation unterliegt der Urheberrecht. Alle Rechte sind bei Dr, S. Zacher vorbehalten. **All rights are by the author, Dr. S. Zacher, reserved.** Die Weiterentwicklung oder Nutzung der Publikation ohne Referenz auf Urheber ist nicht zugelassen. **No use of this publication without references on the author.**

Für die Anwendung der vorliegenden Publikation in der Industrie, im Laborbetrieb und in anderen praktischen Fällen sowie für eventuelle Schäden, die aus unvollständigen oder fehlerhaften Angaben über das dynamische Systeme ergeben können, übernimmt der Autor keine Haftung. **For the practical use of the results of this publication takes the author no responsibility.**

# INHALT

Einführung	Seite 4
1. Analoges Simulationsmodell des Regelkreises	Seite 5
2. Solver-Einstellungen für Code-Generierung	Seite 6
3. Digitalisierung:	
3.1 mit Model-Discretizer	Seite 7
3.2 nach z-Transformation	Seite 10
4. Anhang: z-Transformation einer P-T2 Strecke	Seite 14
5. Zusammenfassung	Seite 16

## Einführung

Die vorliegende Publikation ist meine Antwort auf die folgende Frage eines Studenten zum Wahlfach MBSE „Modellbasierte Softwareentwicklung“:

Sehr geehrter Herr Zacher,

Laut Ihrem Skript ist die z-Transformation notwendig, um einen zeitdiskreten Baustein zu erhalten, welchen ich anschließend in Form eines C-Programmes auf den Board *Arduino* laden kann.

So muss ich nun die Strecke bzw. den Regler entsprechend diskretisieren.

Die Übertragungsfunktionen lauten:                      Strecke:  $G_s(s) = 17,1667 / ((1+0,1s)(1+0,2s))$

Regler:  $GR(s) = (0,0146s+0,0728) / 0,2s$

Die besten Versuche sahen wie folgt aus:

Als erstes habe ich die Übertragungsfunktion in eine Differentialgleichung umgewandelt.

Da diese 2. Ableitungen enthielt, formte ich daraus mittels einfacher Integration eine Integro-Differentialgleichung, welche ich zeitdiskret als Differenzengleichung dargestellt habe.

Dabei galt

$$\int y(t) dt = T_a * y[k-1] + T_a/2 * \Delta y[k]$$

$$\text{für } \Delta y[k] = y[k] - y[k-1]$$

Mit den Vorschriften

$$x[k] \rightarrow X(z)$$

$$x[k-1] \rightarrow 1/z * X(z)$$

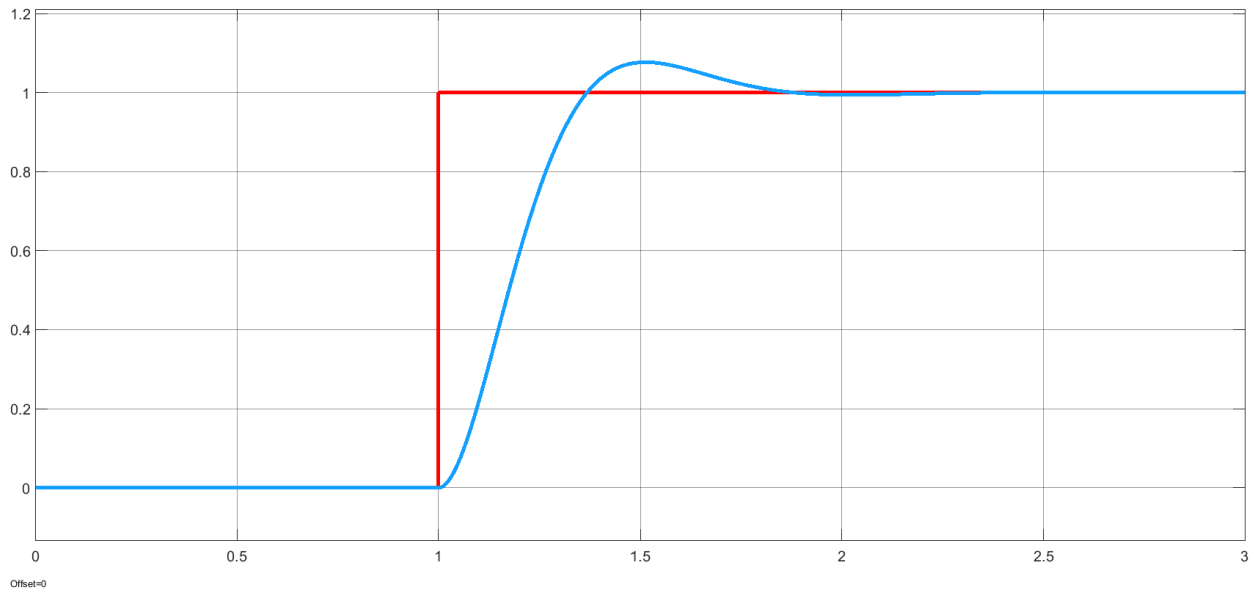
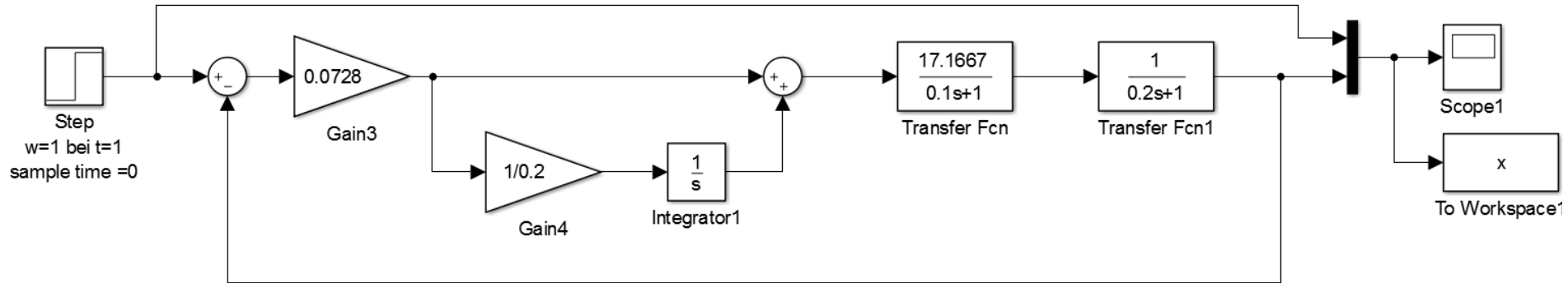
erhielt ich eine Übertragungsfunktion im z-Bereich wie folgt:

$$G_s(z) = (17,1667 * T_a^2 * z + 17,1667 * T_a^2) / ((0,04 + 0,6 * T_a + T_a^2) * z + (T_a^2 - 0,04))$$

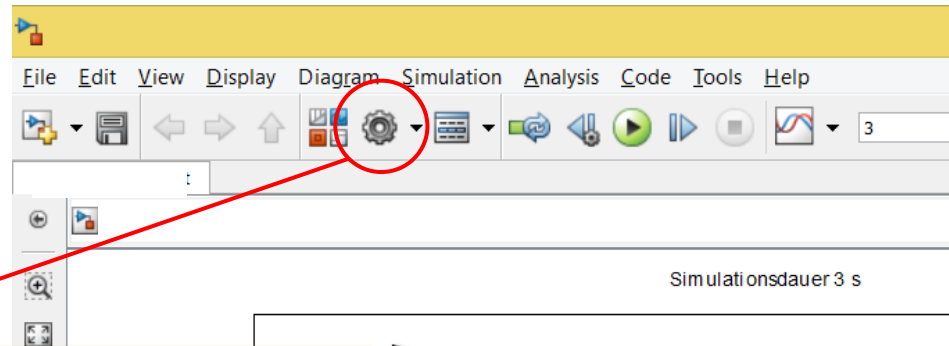
Vergleichende Simulation mit den originalen Blöcken erwies allerdings, dass keine Übereinstimmung zu sehen war.

# 1. Analoges Simulationsmodell eines Regelkreises

Simulationsdauer 4 s



## 2. Solver-Einstellungen für Code-Generierung



Simulationsdauer 3 s

Configuration Parameters

- Category
- List
- Select:
- Solver**
- Data Import/Export
- Optimization
- Diagnostics
- Hardware Implementation
- Model Referencing
- Simulation Target
- Code Generation

Simulation time  
Start time: 0.0 Stop time: 3

Solver options  
Type: Variable-step Solver: auto (Automatic solver selection)

Additional options

- Select:
- Solver
- Data Import/Export
- Optimization
- Diagnostics
- Hardware Implementation
- Model Referencing
- Simulation Target
- Code Generation

Simulation time  
Start time: 0.0 Stop time: 3

Solver options  
Type: Fixed-step Solver: auto (Automatic solver selection)

Additional options

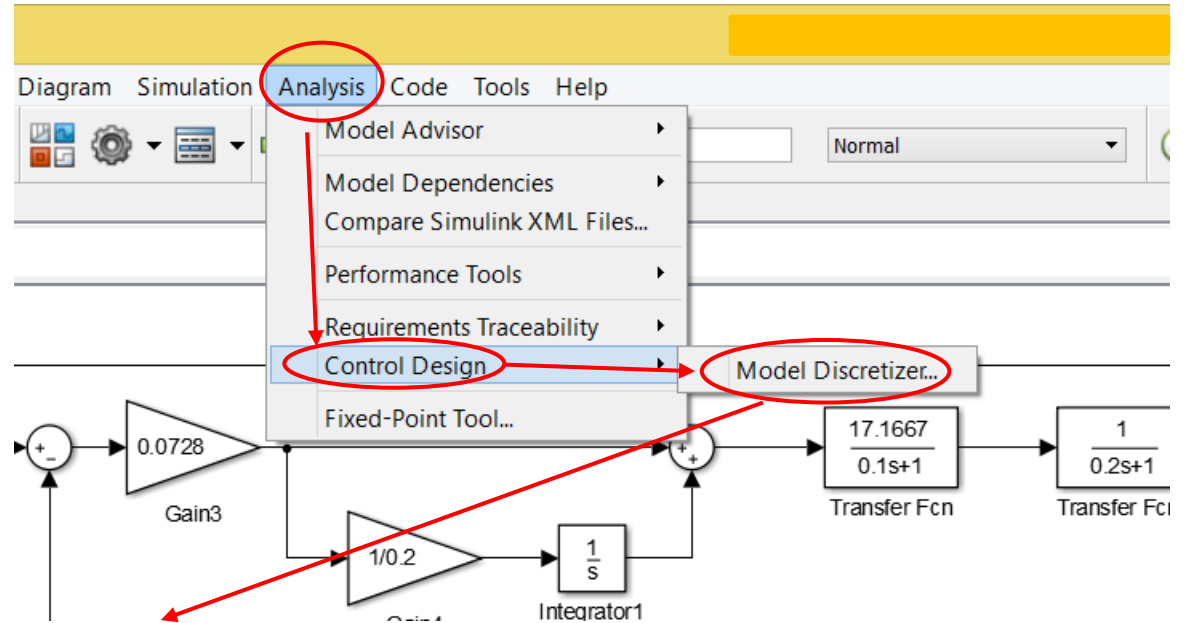
Fixed-step size (fundamental sample time): 0.001

Tasking and sample time options

Periodic sample time constraint: Unconstrained

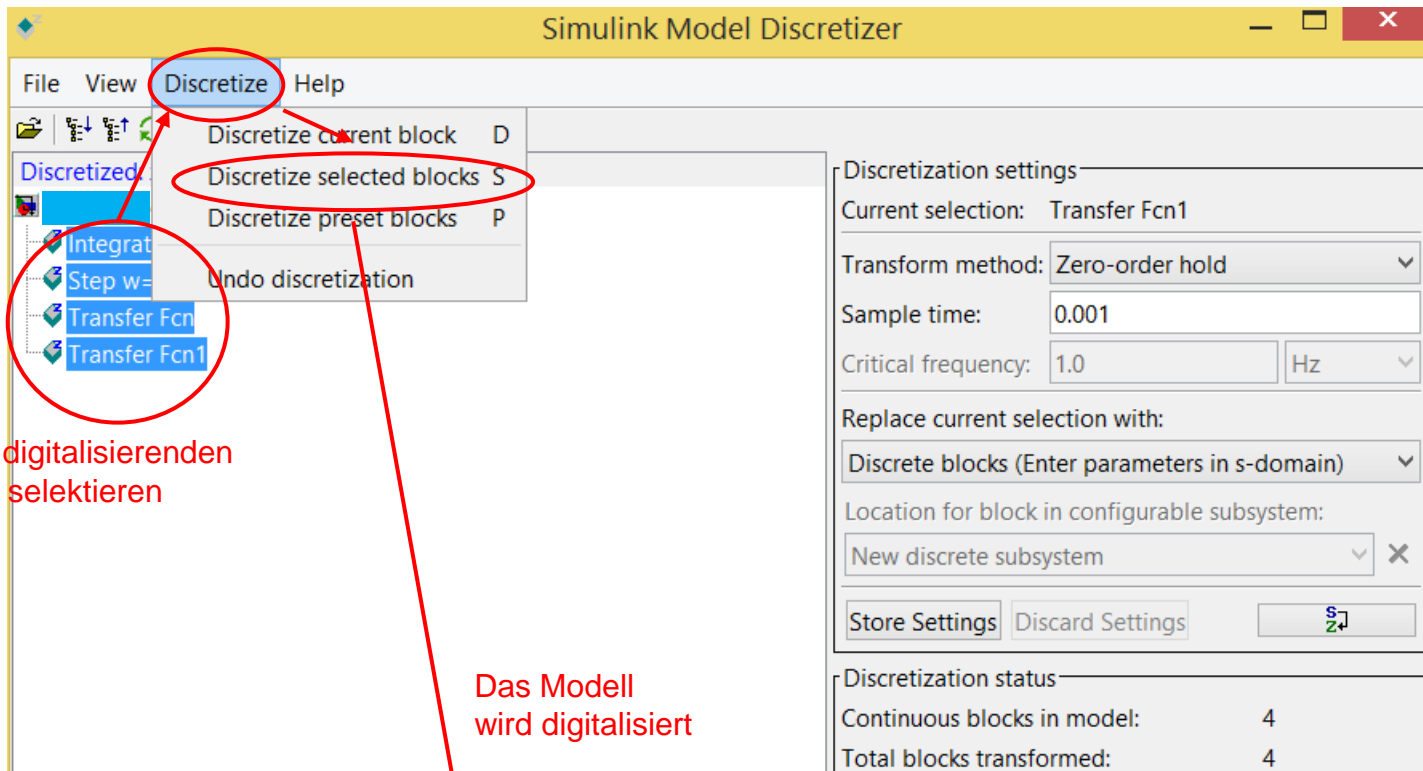
Tasking mode for periodic sample times: Auto

### 3.1 Digitalisierung mit *Model Discretizer*



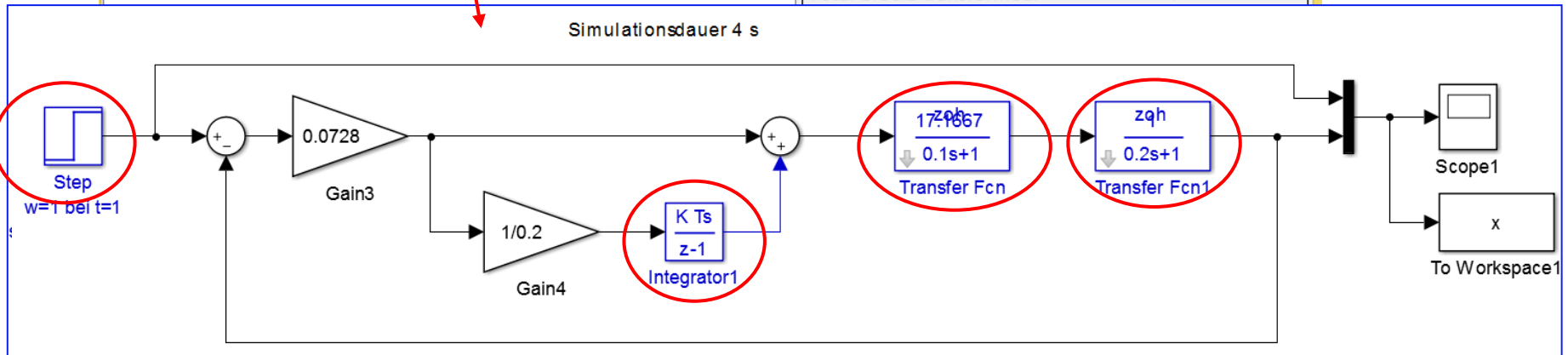
The 'Simulink Model Discretizer' dialog box is shown. The 'Discretization settings' section includes: 'Current selection: Hoeft\_Studienarbeit', 'Transform method: Zero-order hold', 'Sample time: 0.001', and 'Critical frequency: 1.0 Hz'. The 'Sample time' field is circled in red.

### 3.1 Digitalisierung mit Model Discretizer



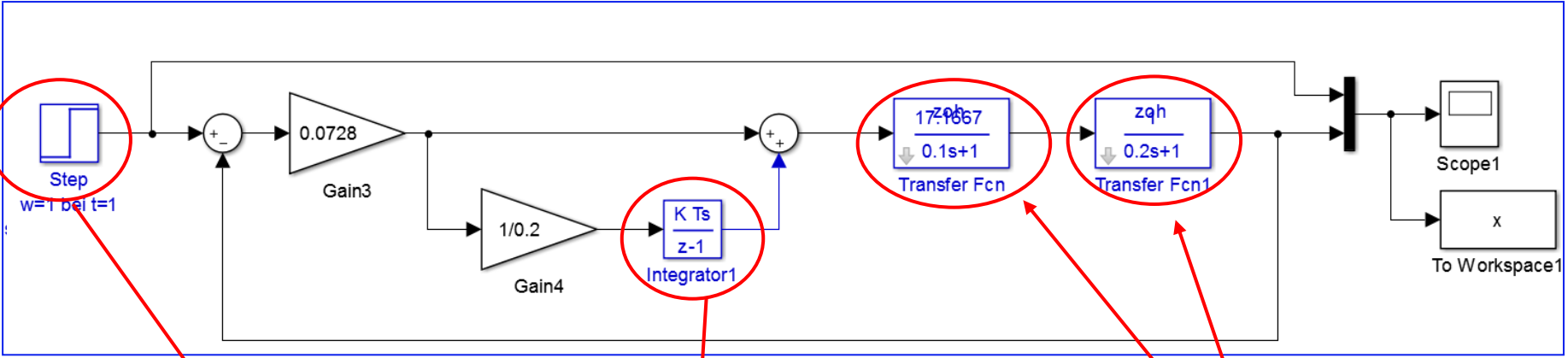
Die zu digitalisierenden Blöcke selektieren

Das Modell wird digitalisiert





### 3.1 Digitalisierung mit Model Discretizer



Source Block Parameters: Step w=1 bei t=1 sampl...

Step

Output a step.

Parameters

Step time:

Initial value:

Final value:

Sample time:

Interpret vector parameters as 1-D

Enable zero-crossing detection

OK Cancel Help Apply

Die Blöcke und Parameter werden automatisch umgestellt

Function Block Parameters: Integrator1

Discrete Integrator

Discrete-time integration or accumulation of the input signal.

Main Signal Attributes State Attributes

Integrator method: Integration: Forward Euler

Gain value: 1.0

External reset: none

Initial condition source: internal

Initial condition: 0

Initial condition setting: Output

Sample time (-1 for inherited): 0.001

Limit output

Upper saturation limit: inf

Lower saturation limit: -inf

Show saturation port

OK Cancel Help Apply

Die Parameter der Strecke werden automatisch nach zoh-Methode umgerechnet

## 3.2 Digitalisierung nach z-Transformation

### Beispiel: PI-Regler mit P-T2-Strecke

Gegeben ist die P-T2-Strecke mit  $K_{PS} = 0,5$   $T_1 = 1,5$   $T_2 = 0,3$

$$G_S(s) = \frac{K_{PS}}{(1 + sT_1)(1 + sT_2)}$$

Die Strecke wird mit einem PI-Regler nach dem Betragsoptimum geregelt:

$$G_R(s) = \frac{K_{PR}(1 + sT_n)}{sT_n}$$

Die Übertragungsfunktion des aufgeschnittenen Regelkreises:

$$G_0(s) = G_R(s)G_S(s) = \frac{K_{PR}(1 + sT_n)}{sT_n} \cdot \frac{K_{PS}}{(1 + sT_1)(1 + sT_2)}$$

Nach der Kompensation mit  $T_n = T_{\text{größte}} = T_1 = 1,5$

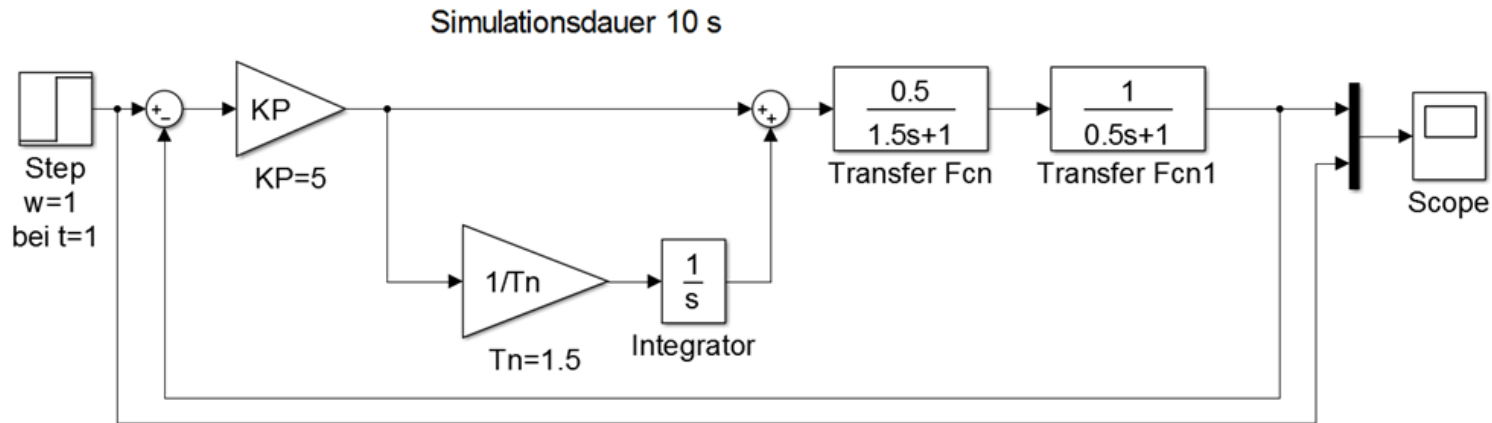
ergibt sich:

$$G_0(s) = \frac{K_{PR}K_{PS}}{sT_n(1 + sT_2)}$$

Daraus folgt nach dem Betragsoptimum, Grundtyp „A“:

$$K_{PR} = \frac{T_n}{2K_{PS}T_2} \quad K_{PR} = \frac{1,5}{2 \cdot 0,5 \cdot 0,3} = 5$$

### 3.2 Digitalisierung nach z-Transformation



Da das SIL nur für digitale Regelkreise funktioniert, wird zuerst die Strecke für Abtastzeit  $T = 0,1$  s digitalisiert (siehe Buch Zacher, „Regelungstechnik-Aufgaben“, 3. Auflage, 2012, Aufgabe 4.2.11, Seite 112):

$$G_S(s) = \frac{K_{PS}}{(1+sT_1)(1+sT_2)} \implies G_{HS}(z) = 0,187 \frac{0,0275z + 0,0157}{(z - 0,7189)(z - 0,9352)}$$

Auch die z-Übertragungsfunktion des PI-Reglers mit gegebenen Kennwerten

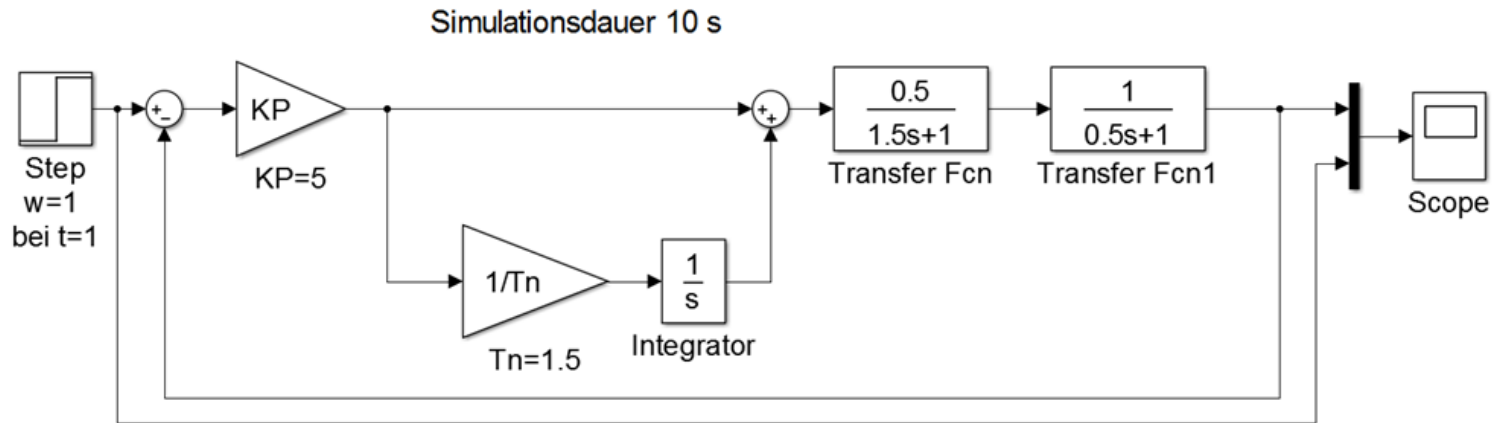
$$K_{PR} = 5$$

$$T_n = 1,5 \text{ s}$$

wird aus der Tabelle Tabelle 1.3.4, Seite 12, des Buches S. Zacher „Regelungstechnik-Aufgaben“, 3. Auflage, 2012, übernommen:

$$G_R(s) = \frac{K_{PR}(1+sT_n)}{sT_n} \implies G_R(z) = \frac{c_1z - c_2}{z-1} = K_{PR} \frac{z - 0,93}{z-1}$$

### 3.2 Digitalisierung nach z-Transformation



Da das SIL nur für digitale Regelkreise funktioniert, wird zuerst die Strecke für Abtastzeit  $T = 0,1$  s digitalisiert (siehe Buch Zacher, „Regelungstechnik-Aufgaben“, 3. Auflage, 2012, Aufgabe 4.2.11, Seite 112):

$$G_S(s) = \frac{K_{PS}}{(1+sT_1)(1+sT_2)} \implies G_{HS}(z) = 0,187 \frac{0,0275z + 0,0157}{(z - 0,7189)(z - 0,9352)}$$

Auch die z-Übertragungsfunktion des PI-Reglers mit gegebenen Kennwerten

$$K_{PR} = 5$$

$$T_n = 1,5 \text{ s}$$

wird aus der Tabelle Tabelle 1.3.4, Seite 12, des Buches S. Zacher „Regelungstechnik-Aufgaben“, 3. Auflage, 2012, übernommen:

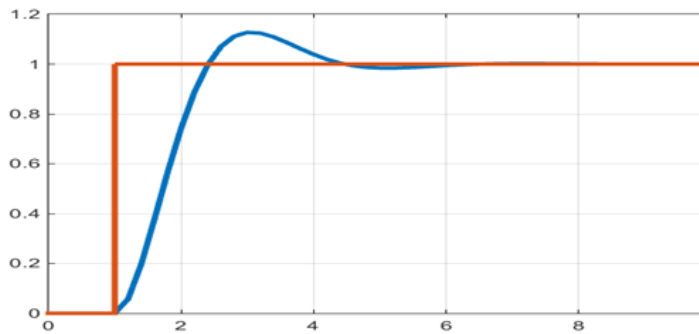
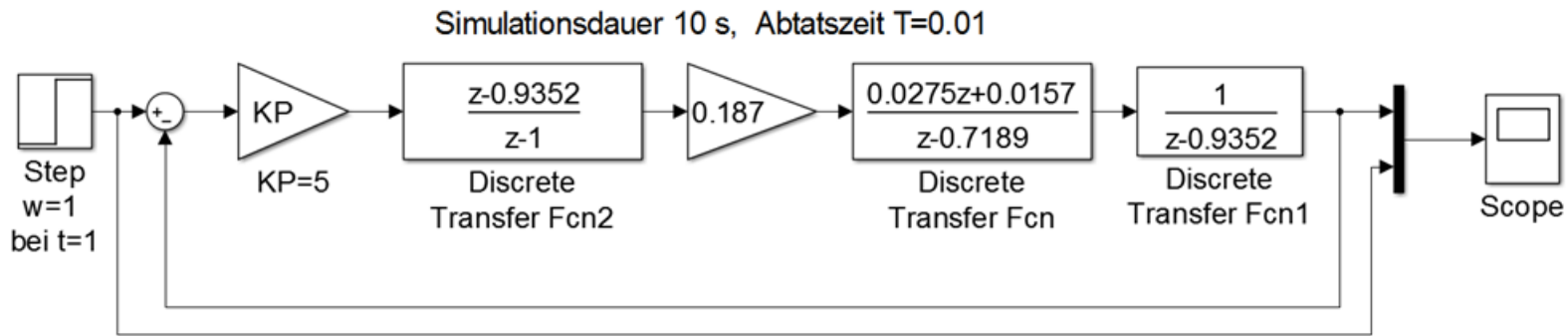
$$G_R(s) = \frac{K_{PR}(1+sT_n)}{sT_n} \implies G_R(z) = \frac{c_1z - c_2}{z-1} = K_{PR} \frac{z - 0,93}{z-1}$$

## 3.2 Digitalisierung nach z-Transformation

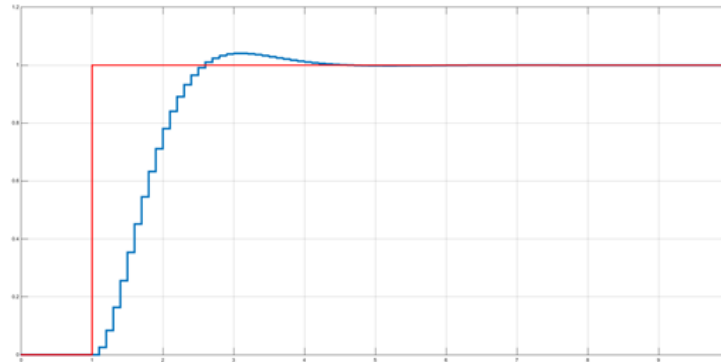
mit

$$c_1 = K_{PR} = 5$$

$$c_2 = K_{PR} \left( 1 - \frac{T}{T_n} \right) = K_{PR} \left( 1 - \frac{0,1}{1,5} \right) = 0,93K_{PR}$$



Sprungantwort des analogen Kreises



Sprungantwort des digitalisierten Kreises

# 4. Anhang: z-Transformation einer P-T2 Strecke

nach dem Buch S, Zacher, „Regelungstechnik-Aufgaben“,  
3. Auflage, 2012, Aufgabe 4.2.11, Seite 112

$$G_S(s) = \frac{K_{PS}}{(1+sT_1)(1+sT_2)}$$

$$K_{PS} = 0,5$$

$$T_1 = 1,5 \text{ s}$$

$$T_2 = 0,5 \text{ s}$$

$$\text{Abtastzeit } T = 0,1 \text{ s}$$

$$G_{HS}(z) = \frac{z-1}{z} \cdot Z \left[ \frac{G_S(s)}{s} \right] = \frac{z-1}{z} \cdot Z \left[ \frac{0,5}{s(1+sT_1)(1+sT_2)} \right]$$

$$G_{HS}(z) = \frac{0,5}{T_1 T_2} \frac{z-1}{z} \cdot Z \left[ \frac{ab(a-b)}{s(s+a)(s+b)} \right] = 1,11 \frac{z-1}{z} \cdot Z \left[ \frac{ab(a-b)}{s(s+a)(s+b)} \right]$$

$$Z \left[ \frac{ab(a-b)}{s(s+a)(s+b)} \right] = \frac{N_z(z)z}{(z-1)(z-a_1)(z-b_1)}$$

$$a_1 = e^{-aT} = e^{-\frac{T}{T_2}} = e^{-0,33} = 0,7189$$

$$b_1 = e^{-bT} = e^{-\frac{T}{T_1}} = e^{-0,067} = 0,9352$$



#### 4. Anhang: z-Transformation einer P-T2 Strecke

$$N_z(z) = (a - b - ae^{-bT} + be^{-aT})z + [(a - b)e^{-(a+b)T} - ae^{-aT} + be^{-bT}]$$

$$N_z(z) = 0,0274 \cdot z + 0,0157$$

$$Z \left[ \frac{ab(a-b)}{s(s+a)(s+b)} \right] = \frac{N_z(z)z}{(z-1)(z-a_1)(z-b_1)}$$

$$G_{HS}(z) = 1,11 \frac{z-1}{z} \cdot Z \left[ \frac{ab(a-b)}{s(s+a)(s+b)} \right]$$

$$G_{HS}(z) = \frac{1,11}{5,94} \frac{z-1}{z} \cdot Z \left[ \frac{5,94}{s(s+a)(s+b)} \right]$$

$$G_{HS}(z) = 0,187 \frac{z-1}{z} \cdot \frac{(0,0275z + 0,0157)z}{(z-1)(z-0,7189)(z-0,9352)}$$

$$G_{HS}(z) = 0,187 \frac{(0,0275z + 0,0157)}{(z-0,7189)(z-0,9352)}$$

## 5. Zusammenfassung

Die vorliegende Publikation ist in erster Linie für Teilnehmerinnen und Teilnehmer des Unterrichts „MBSE: Modellbasierte Softwareentwicklung“ geeignet.

Die Simulation mit MATLAB/Simulink ist sowohl für analoge, als auch für digitale Regelkreise möglich. Dagegen erfolgt die automatische C-Code Generierung mit *MATLAB Coder*, *Simulink Coder* und *PLC Coder* nur für digitale Kreise. Ab Versionen R2014 und R2015 ermöglicht MATLAB/Simulink auch die Digitalisierung von analogen Simulink-Modellen mit den Menü-Befehlen

*Analysis* → *Control Design* → *Model Discretizer*

automatisch durchführen. Jedoch zuerst soll die Simulation im *Solver* von *Variable-Step* in *Fixed-Step* umgestellt werden.

In der vorliegende Publikation wurde Schritt für Schritt beschrieben, wie man an *Solver* und *Model Discretizer* gelangt und wie die Umstellungen gemacht werden.

Ein Simulink-Modell wird damit automatisch von Laplace-Transformierten Übertragungsfunktionen  $G(s)$  in die z-transformierte Übertragungsfunktionen  $G(z)$  umgewandelt. Dies erkennt man sofort durch den Schriftzug *zoh*, der auf allen digitalisierten Bausteinen nach der Ausführung des *Model Discretizer* erscheint.

Obwohl dieser Weg für die Code-Generierung zu empfehlen ist, wurde auch die Digitalisierung mittels z-Transformation beschrieben und an einem Beispiel verdeutlicht.