DCS01

Digital Control Systems

Collegedictaat

Versie 4.0b

E.H.W. van de Logt
E.M. van den Bergh
J.W. Peltenburg
J.Z.M. Broeders

Versiehistorie

Datum	Versie	Omschrijving	Auteur
25-05-2016	4.0b[1]	Fout in vergelijking (2.6) en vergelijking (4.2) verbeterd	VersD
17-05-2016	4.0a	Fout in vergelijking (3.9) verbeterd	BroJZ
15-06-2015	4.0	Omgezet naar LaTeX, diverse fouten verbeterd, hoofdstukken 4 en 5 herschreven, bibliografie toegevoegd.	BroJZ
13-05-2013	3.0	Nieuwe hoofdversie van collegedictaat. Grote delen overgenomen van dictaat E.H.W. van de Logt en E.M. van den Bergh. • Nieuwe lay-out. • Enkele voorbeelden uitgebreider gemaakt. • Plaatjes toegevoegd integratiemethoden. • Opgaven aangepast en toegevoegd. Voor voorgaande versiehistorie zie archief (2011-2012).	PelJH

Collegedictaat Digital Control Systems van Hogeschool Rotterdam is in licentie gegeven volgens een Creative Commons Naamsvermelding-NietCommercieel-GelijkDelen 3.0 Nederland-licentie.

[1] Toelichting versiecodering *A.Bc*: A = grote aanpassing, B = kleine aanpassing, c = taal- of wiskundige correcties.

Inhoudsopgave

1

Introductie

Dit collegedictaat is onderdeel van het vak DCS01(Digital Control Systems) en is bestemd voor derdejaars studenten van de opleiding Elektrotechniek. In deze module wordt voortgebouwd op de theorie van de basisvakken Regeltechniek (REG01 en REG02) en op de theorie van Digital Signal Processing (DSP01). Waar we bij REG01 en REG02 nog de nadruk gelegd hebben op de theorie zelf, wordt in deze module de nadruk gelegd op de praktijk: de student is na afloop van deze module in staat om regelsystemen te ontwerpen, te simuleren en te bouwen, en om deze optimaal in te kunnen stellen. Voor het praktijkgedeelte van deze module wordt gebruik gemaakt van MATLAB / Simulink, waarmee de student zelf modellen maakt van een dynamisch systeem. Met behulp van deze modellen kan het regeltechnisch gedrag gesimuleerd en bestudeerd worden. Met name het optimaal kunnen instellen van een regelaar is een belangrijke competentie voor een HBO-ingenieur. In de beroepspraktijk krijgt deze vaak te maken met regelaars, die tegenwoordig overwegend digitaal zijn.

Dit collegedictaat bevat aanvullende notities, behorende bij het college Digital Control Systems (DCS01). De meeste van deze notities worden ook behandeld in het college en zijn samengevat in de daarbij behorende slides. Bij elkaar vormen ze het complete lesmateriaal voor deze module.

Dit document bevat de volgende hoofdstukken:

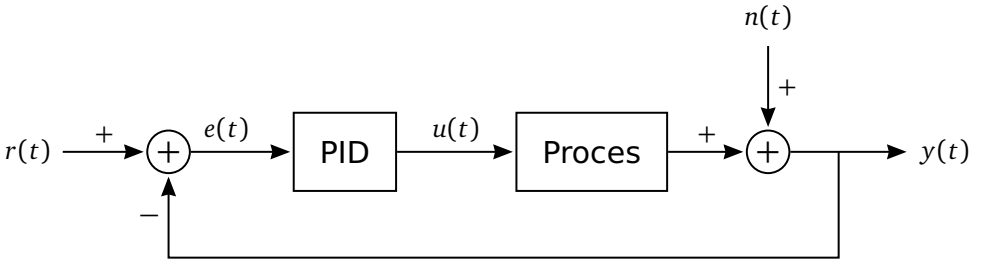
- Hoofdstuk 2: digitale PID-regelaars. Hier worden diverse afleidingen gegeven van discrete PID-regelaars, die afgeleid zijn van de ideale (tekstboek versie) tijdcontinue PID-regelaar.
- Hoofdstuk 3: verbetering aan digitale PID-regelaars. Dit hoofdstuk bevat een aantal verbeteringen die toegevoegd kunnen worden aan de discrete PID-regelaars, zoals die in hoofdstuk 2 afgeleid zijn.
- Hoofdstuk 4: Vinden van de optimale parameters. Hier worden diverse methoden en algoritmen beschreven om de optimale PID-parameters te vinden.
- Hoofdstuk 5: Systeemidentificatie. Dit hoofdstuk bevat een beschrijving van een adaptieve PID-regelaar, die zelf de optimale parameters kan bepalen m.b.v. het least-squares-algoritme.

2

Digitale PID-regelaars

Een PID-regelaar probeert de uitgangswaarde van een te regelen proces (in de Engelse literatuur kom je het woord 'plant' hiervoor regelmatig tegen) op een vooraf ingestelde waarde te houden. Deze waarde wordt ook wel de referentiewaarde of het setpoint (SP) genoemd. De regelaar doet dit door de afwijking van de gewenste waarde te bepalen. Dit signaal noemt men vaak het 'error'-signaal. Deze errorwaarde is gedefinieerd als het verschil tussen het setpoint en de uitgangswaarde (of 'output') van het te regelen proces (bijv. een temperatuurwaarde of een waterhoogte). Deze uitgangswaarde wordt ook vaak de procesvariabele genoemd. In de praktijk bevat de uitgangswaarde ook wat ruis. In sommige systemen wordt hier rekening mee gehouden, maar in theoretische besprekingen van de basisprincipes van regelaars wordt dit vaak weggelaten. Hoe groter de errorwaarde, des te meer zal de regelaar bij moeten sturen. De meest gebruikte regelaar in de industrie is ontegenzeggelijk de PID-regelaar. Hierbij staat de afkorting PID voor Proportioneel, Integrerend en Differentiërend. Met behulp van een drietal parameters (hierover later meer) kan de complete PID-regelaar ingesteld worden.

We kunnen een typisch PID-geregeld proces weergeven met het wiskundige model van figuur 2.1.



Figuur 2.1: Algemeen model van een PID-geregeld systeem.

In dit model is:

t	tijdvariabele;
$r(t)$	referentiewaarde oftewel setpoint;
$e(t)$	afwijking of error t.o.v. de gewenste output;
$u(t)$	output van de regelaar oftewel input van het proces;
$n(t)$	een ruissignaal oftewel noise;
$y(t)$	de uitgangswaarde, output oftewel procesvariabele.

Alle waarden die je in dit overzicht ziet zijn als functie van de tijd, t, gegeven. Bij DCS01 gaan we er uiteindelijk vanuit dat alle waarden kunnen veranderen in de tijd. Ook gaan we later bekijken wat er gebeurt als zelfs het proces zelf verandert in de tijd.

De algemene vergelijking voor een PID-regelaar in het tijdcontinue domein wordt gegeven door:

$$u(t) = K_c \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau)\, d\tau + T_d \frac{de(t)}{dt} \right) \tag{2.1}$$

Hierbij is:

K_c	Proportionele versterkingsfactor;
T_i	Tijdconstante integrerende versterking [s];
T_d	Tijdconstante differentiërende versterking [s].

Door middel van een Laplace-transformatie kan deze tijdcontinue functie naar het s-domein getransformeerd worden:

$$U(s) = K_c\left(1 + \frac{1}{T_i \cdot s} + T_d \cdot s\right)E(s) \tag{2.2}$$

Hierbij is s de Laplace-operator en is gelijk aan $\alpha + j\omega$. Uitgaande van vergelijking (2.2) kunnen we de overdrachtsfunctie bepalen van deze PID-regelaar in het Laplace-domein ($H_r(s)$):

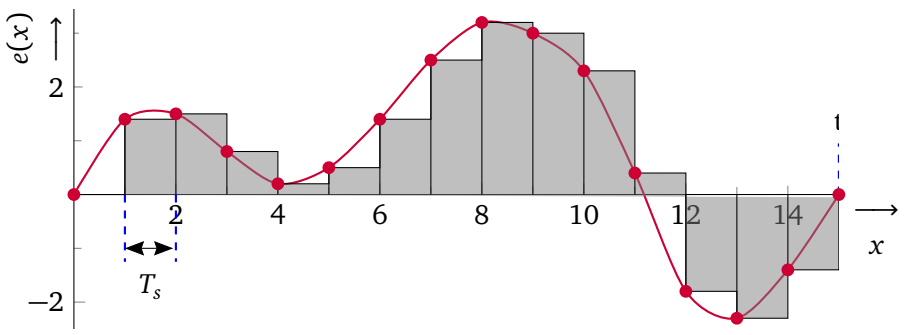
$$H_r(s) = \frac{U(s)}{E(s)} = K_c\left(1 + \frac{1}{T_i \cdot s} + T_d \cdot s\right) \tag{2.3}$$

Digitale regelaars werken in het tijddiscrete domein in plaats van het tijdcontinue domein, omdat ze werken op samples van signalen en niet met het directe signaal zelf.

Om een tijddiscrete versie te krijgen van deze tijdcontinue PID-regelaar, moeten we de integrerende en differentiërende componenten van vergelijking (2.1) tijd-discreet maken. Hiervoor maken we eerst even een uitstapje naar twee deelparagrafen: eentje over numerieke integratie en eentje over numerieke differentiatie.

2.1 Numerieke integratie

Uit de integraalrekening weten we dat het bepalen van een integraal overeenkomt met het bepalen van het oppervlak onder de functie waarover we integreren. Het oppervlak onder de curve kunnen we benaderen door hier rechthoeken van te maken en het oppervlak van al deze rechthoeken bij elkaar op te tellen. In figuur 2.2 is dit concept weergegeven.



Figuur 2.2: Forward Rectangular Method (FRM).

Als we nu iedere T_s seconde een sample nemen van het signaal, dan kan het oppervlak onder een functie $e(x)$ als volgt benaderd worden:

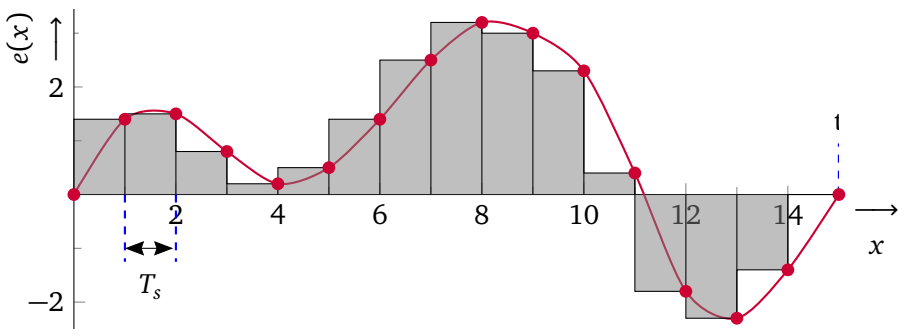
$$\int_0^t e(x)\,dx \approx T_s \cdot e[0] + T_s \cdot e[1] + \cdots + T_s \cdot e[k-1] = T_s \sum_{i=1}^{k} e[i-1]$$

$$\text{met } k = \left\lfloor \frac{t}{T_s} + \frac{1}{2} \right\rfloor \quad (2.4)$$

$k = \left\lfloor \frac{t}{T_s} + \frac{1}{2} \right\rfloor$ betekent dat bij deze formule k de naar beneden afgeronde waarde van $\frac{t}{T_s} + \frac{1}{2}$ is. Dat is hetzelfde als $\frac{t}{T_s}$ afronden naar een geheel getal.

De aanpak voor het benaderen van de integraal $e(x)$ van vergelijking (2.4) wordt ook wel de Forward Rectangular Method genoemd, of de 'ondersom' nemen van de functie.

Een alternatieve manier om het oppervlak te bepalen is m.b.v. de 'bovensom'. Dit wordt ook wel de Backward Rectangular Method (BRM) genoemd. Dit is weergegeven in figuur 2.3.



Figuur 2.3: Backward Rectangular Method (BRM).

Het oppervlak onder een functie $e(x)$ wordt bij deze methode als volgt benaderd:

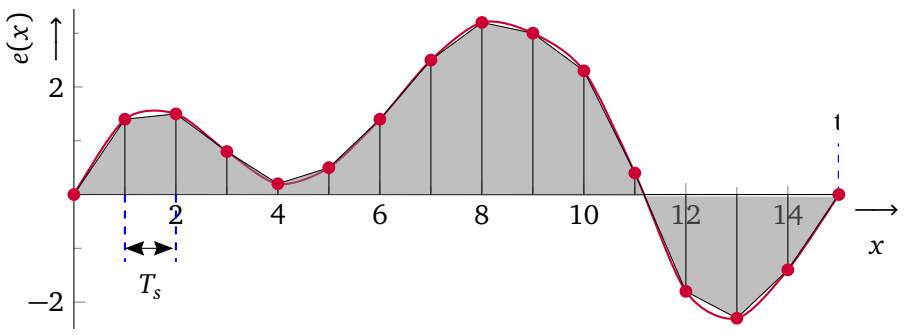
$$\int_0^t e(x)\,dx \approx T_s \cdot e[1] + T_s \cdot e[2] + \cdots + T_s \cdot e[k] = T_s \sum_{i=1}^{k} e[i] \text{ met } k = \left\lfloor \frac{t}{T_s} + \frac{1}{2} \right\rfloor \tag{2.5}$$

Naast deze twee methoden is er nog een derde methode die vaak gebruikt wordt. Dat is de 'trapeziummethode' (TRAP). Hierin wordt het oppervlak onder een stapje van T_s niet benaderd met een rechthoekje maar met een rechthoekig trapezium. De oppervlakte van zo'n trapezium is gelijk aan het gemiddelde van 2 opeenvolgende functiewaarden maal de sampletijd. Dit is meestal nauwkeuriger dan het gebruik van de FRM en BRM. Dit is goed te zien in figuur 2.4.

Het oppervlak onder een functie $e(x)$ wordt bij deze methode als volgt benaderd:

$$\int_0^t e(x)\,dx \approx T_s \cdot \frac{e[1]+e[0]}{2} + T_s \cdot \frac{e[2]+e[1]}{2} + \cdots + T_s \cdot \frac{e[k]+e[k-1]}{2} \tag{2.6}$$

$$= \frac{T_s}{2} \sum_{i=1}^{k} e[i] + e[i-1] \text{ met } k = \left\lfloor \frac{t}{T_s} + \frac{1}{2} \right\rfloor$$



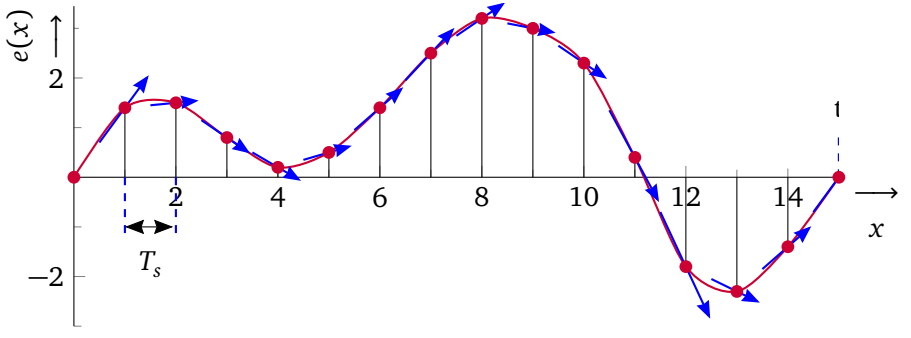
Figuur 2.4: Trapeziummethode (TRAP).

2.2 Numerieke Differentiatie

Bij numerieke differentiatie vindt iets soortgelijks plaats. Uit de differentiaalrekening weten we dat het differentiëren overeenkomt met het bepalen van de raaklijn in een punt. Als benadering nemen we het verschil van twee opeenvolgende waarden van e en delen dat door het verschil de bijbehorende waarden van x. In een formule:

$$e'(x) = \frac{de}{dx} \approx \frac{e[k] - e[k-1]}{T_s} \quad \text{met } k = \left\lfloor \frac{t}{T_s} + \frac{1}{2} \right\rfloor \tag{2.7}$$

Deze zogenoemde tweepuntsbenadering is zichtbaar in figuur 2.5.



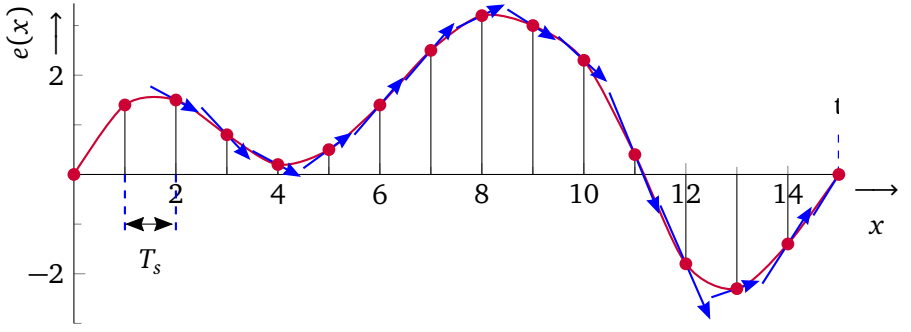
Figuur 2.5: Numeriek Differentiëren m.b.v. de tweepuntsbenadering.

In plaats van de relatief eenvoudige tweepuntsbenadering voor het differentiëren (zie vergelijking (2.7)), kan ook gebruik gemaakt worden van een driepunts- of een vierpuntsbenadering voor het differentiëren. In dat geval gaat vergelijking (2.7) over in:

Driepuntsbenadering:

$$e'(x) = \frac{de}{dx} \approx \frac{3 \cdot e[k] - 4 \cdot e[k-1] + e[k-2]}{2 \cdot T_s} \quad \text{met } k = \left\lfloor \frac{t}{T_s} + \frac{1}{2} \right\rfloor \tag{2.8}$$

Deze driepuntsbenadering is zichtbaar in figuur 2.6.

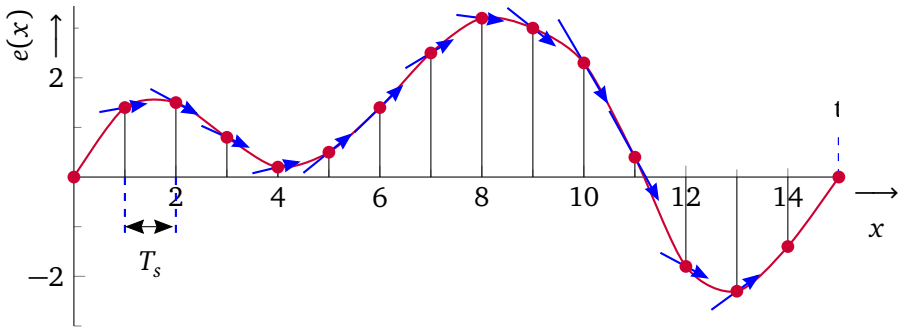


Figuur 2.6: Numeriek Differentiëren m.b.v. de driepuntsbenadering.

Vierpuntsbenadering:

$$e'(x) = \frac{de}{dx} \approx \frac{e[k+2] + 3 \cdot e[k+1] - 3 \cdot e[k] - e[k-1]}{6 \cdot T_s} \quad \text{met } k = \left\lfloor \frac{t}{T_s} + \frac{1}{2} \right\rfloor \tag{2.9}$$

Bij deze vierpuntsbenadering wordt gebruik gemaakt van samples uit de toekomst. Deze vierpuntsbenadering is zichtbaar in figuur 2.7.



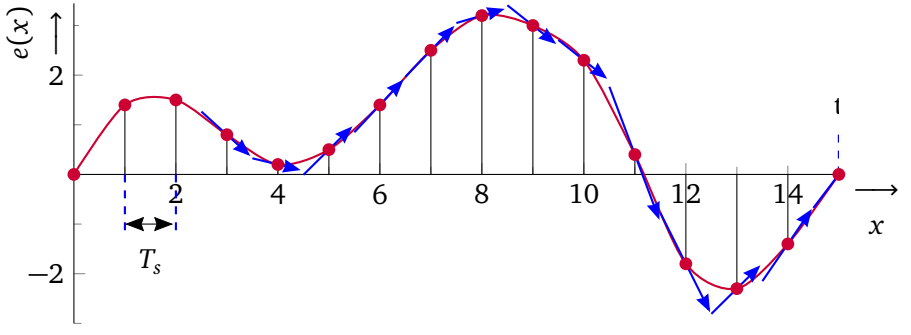
Figuur 2.7: Numeriek Differentiëren m.b.v. de vierpuntsbenadering.

Alternatieve vierpuntsbenadering:

$$e'(x) = \frac{de}{dx} \approx \frac{11 \cdot e[k] - 18 \cdot e[k-1] + 9 \cdot e[k-2] - 2 \cdot e[k-3]}{6 \cdot T_s}$$

$$\text{met } k = \left\lfloor \frac{t}{T_s} + \frac{1}{2} \right\rfloor \quad (2.10)$$

Deze vierpuntsbenadering maakt geen gebruik van samples uit de toekomst en is zichtbaar in figuur 2.8.



Figuur 2.8: Numeriek Differentiëren m.b.v. de alternatieve vierpuntsbenadering.

2.3 Afleiding van een tijddiscrete PID-regelaar

Met behulp van de bovenstaande vergelijkingen voor numeriek integreren en differentiëren kunnen we de formule van een tijddiscrete PID-regelaar afleiden.

Als voorbeeld maken we gebruik van de Forward Rectangular Method (FRM) (vergelijking (2.4)) om de integrator om te zetten en numerieke differentiatie met de tweepuntsbenadering (vergelijking (2.7)) om de differentiator om te zetten.

Nogmaals gegeven de vergelijking van de uitgang van de regelaar vergelijking (2.1):

$$u(t) = K_c\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)\,d\tau + T_d\frac{de(t)}{dt}\right) \tag{2.1}$$

Als we dit samplen met sampletijd T_s en we passen de bovengenoemde numerieke methoden toe dan krijgen we:

$$u[k] = K_c\left(e[k] + \frac{T_s}{T_i}\sum_{i=1}^{k} e[i-1] + \frac{T_d}{T_s}\big(e[k]-e[k-1]\big)\right)$$

$$\text{met } k = \left\lfloor \frac{t}{T_s} + \frac{1}{2} \right\rfloor \tag{2.11}$$

Waarbij k de tijdvariabele van het tijddiscrete systeem is, oftewel het samplenummer. De bovenstaande vergelijking (2.11) wordt ook wel het *positiealgoritme* genoemd en kent in de praktijk een aantal nadelen.

Opdracht 1: Andere integratiemethodieken

Leid de uitgang van een tijddiscrete regelaar ook af met behulp van:

a) de BRM;

b) de TRAP-methode.

2.4 Het snelheidsalgoritme

Een groot nadeel van vergelijking (2.11), de tijddiscrete variant van de uitgang van de regelaar, is te vinden in het gedeelte van de integrator. In de praktijk willen we deze vergelijking graag gebruiken, maar dat betekent dat we alle voorgaande waarden van het errorsignaal $e[k]$ moeten opslaan. Dit komt omdat de formule nu zo geschreven is, dat het integrerende deel de som van alle voorgaande waarden van $e[k]$ bepaalt met behulp van een somreeks. De somreeks loopt over alle voorgaande waarden van $e[k]$.

We kunnen een praktijkvoorbeeld beschouwen. Stel je voor dat we een regeling hebben waarbij met een sample-rate van $T_s = 0,01$ s, het toerental van een motor uitgelezen wordt met een nauwkeurigheid van 8 bits. Een ritje van Rotterdam naar Eindhoven duurt ongeveer 1.5 uur = 5400 seconden. Aan het eind van de rit hebben we dus 5400 / 0.01 = 540000 samples van een byte genomen die een geheugenruimte in beslag nemen van 540000 / 1024 $\approx$ 527 kilobyte. Dat lijkt niet veel, maar is voor een simpele en goedkope digitale regeling een hoop data! We hebben in de praktijk voor zulke hoeveelheden extra geheugenchips nodig om de data op te slaan. Dit maakt het systeem duurder.

Verder moet nu ook op het eind een somreeks bepaald worden over 540000 samples. Een ATmega328 van Atmel draaiend op 16 MHz (die zich bijv. ook op het 'Arduino' platform bevindt) zou er in theorie aan het eind van ons autoritje al minstens 0,034 seconden over doen om de integraal uit te rekenen. Dit is al langer dan de sample-rate van de regeling zelf; er is dus al lang geen tijd meer over om de samples in te lezen en de uitgang op tijd te bepalen. Om dit op te lossen zouden we een snellere en dus duurdere processor moeten kopen!

Een derde nadeel (buiten dit voorbeeld om) is dat een verandering in de proceswaarden K_c of T_i direct leidt tot een verandering in de waarde van de integrerende actie in zijn totaliteit. Dit kan bijvoorbeeld resulteren in overflows of andere der-

gelijke problemen. In de praktijk is dit vaak niet acceptabel omdat dit instabiliteit kan veroorzaken.

Om deze problemen tegen te gaan, kunnen we gebruik maken van een recursieve variant van vergelijking (2.11), welke we het *snelheidsalgoritme* noemen. De afleiding van dit algoritme gaat als volgt.

In plaats van elke keer de nieuwe uitgang van de regelaar, $u[k]$, te bepalen, kunnen we ook de voorgaande waarde van $u[k]$ nemen, d.w.z. $u[k-1]$, en het verschil met de gewenste $u[k]$, genaamd $\Delta u[k]$, er bij optellen zodat:

$$u[k] = u[k-1] + \Delta u[k] \tag{2.12}$$

We kunnen nu afleiden dat $\Delta u[k]$ gegeven wordt door:

$$\Delta u[k] = u[k] - u[k-1] \tag{2.13}$$

We nemen nu de $u[k]$ die afgeleid is met de FRM in vergelijking (2.11), nogmaals hieronder getoond:

$$u[k] = K_c\left(e[k] + \frac{T_s}{T_i}\sum_{i=1}^{k} e[i-1] + \frac{T_d}{T_s}\big(e[k]-e[k-1]\big)\right)$$

$$\text{met } k = \left\lfloor \frac{t}{T_s} + \frac{1}{2} \right\rfloor \tag{2.11}$$

Door in deze formule voor k telkens $k-1$ in te vullen vinden we $u[k-1]$

$$u[k-1] = K_c\left(e[k-1] + \frac{T_s}{T_i}\sum_{i=1}^{k-1} e[i-1] + \frac{T_d}{T_s}\big(e[k-1]-e[k-2]\big)\right) \tag{2.14}$$

Nu kunnen we met behulp van vergelijking (2.13) $\Delta u[k]$ bepalen:

$$\Delta u[k] = K_c\left(e[k]-e[k-1]+\frac{T_s}{T_i}e[k-1]+\frac{T_d}{T_s}\big(e[k]-2e[k-1]+e[k-2]\big)\right) \tag{2.15}$$

Nu dat $\Delta u[k]$ bekend is kunnen we dit weer invullen in vergelijking (2.12). De uitgang van de regelaar wordt nu gegeven door:

$$u[k] = u[k-1]+K_c\Bigg(e[k]-e[k-1]+\frac{T_s}{T_i}e[k-1]+ \frac{T_d}{T_s}\big(e[k]-2e[k-1]+e[k-2]\big)\Bigg) \tag{2.16}$$

We zien hier een formule met een recursief gedeelte, namelijk $u[k-1]$. Dit heeft als voordeel dat we nu in de integrator geen somreeks met alle voorgaande samples meer terugvinden. De waarde van de integrator zit nu in $u[k-1]$ verwerkt. We hoeven dus in de praktijk geen groot duur geheugen en geen snellere en duurdere processors meer toe te passen!

Het snelheidsalgoritme kent ook een algemene, kortere notatie, namelijk:

$$u[k] = u[k-1]+q_0e[k]+q_1e[k-1]+q_2e[k-2] \tag{2.17}$$

met:

$$q_0 = K_c\left(1+\frac{T_d}{T_s}\right) \qquad q_1 = K_c\left(\frac{T_s}{T_i}-1-\frac{2T_d}{T_s}\right) \qquad q_2 = \frac{K_cT_d}{T_s} \tag{2.18}$$

Aangezien alle q's (voorlopig) constanten zijn, lijkt vergelijking (2.17) veel op de (reeds bekende) algemene notatie voor een IIR-filter. De PID-regelaar in snelheids-

algoritme (oftewel de recursieve vorm) is dan ook een specifieke implementatie van een IIR-filter!

Er zijn nog een aantal andere aan de praktijk gerelateerde aspecten. Voor al deze implementaties geldt dat de sampletijd T_s klein moet zijn in verhouding tot de tijdconstante van het te regelen proces.

Verder is een belangrijke eigenschap van PID-regelaars dat ze relatief ongevoelig zouden moeten zijn voor wijzigingen in de PID-parameters. De hierboven beschreven regelaars zijn dat nog niet voldoende. Op deze zaken komen we later in dit document nog terug.

Opdracht 2: Het snelheidsalgoritme.

a) Reken de afleiding van $\Delta u[k]$ na en controleer dat vergelijking (2.15) klopt.

b) Pas het snelheidsalgoritme ook toe op de formule voor $u[k]$ die je bij opdracht 1a met behulp van de BRM hebt afgeleid.

c) Pas het snelheidsalgoritme ook toe op de formule voor $u[k]$ die je bij opdracht 1b met behulp van de TRAP-methode hebt afgeleid.

d) Waarom is het snelheidsalgoritme handiger om in software te implementeren dan het positiealgoritme?

2.5 Directe transformatiemethoden

Een andere methode om van de tijdcontinue variant naar een tijddiscrete variant van de PID-regelaar over te gaan is door de Laplace-getransformeerde van de PID-regelaar direct om te zetten naar het z-domein. Hiervoor kunnen we drie methoden gebruiken.

De meest gebruikte methode is de Bilineaire Transformatie (BLT). In de literatuur, maar ook in MATLAB, wordt deze transformatie ook wel 'Tustin' transformatie

Tabel 2.1: Antwoorden van opdracht 2.

Coëfficiënten	FRM	BRM	TRAP
q_0	$K_c\left(1+\frac{T_d}{T_s}\right)$	$K_c\left(1+\frac{T_s}{T_i}+\frac{T_d}{T_s}\right)$	$K_c\left(1+\frac{T_s}{2T_i}+\frac{T_d}{T_s}\right)$
q_1	$K_c\left(\frac{T_s}{T_i}-1-\frac{2T_d}{T_s}\right)$	$-K_c\left(1+\frac{2T_d}{T_s}\right)$	$K_c\left(\frac{T_s}{2T_i}-1-\frac{2T_d}{T_s}\right)$
q_2	$\frac{K_cT_d}{Ts}$	$\frac{K_cT_d}{Ts}$	$\frac{K_cT_d}{Ts}$

genoemd. Twee andere methoden zijn de 'Backward Euler'- en 'Forward Euler'-methode.

De BLT heeft de volgende vorm:

$$s \approx \frac{2}{T_s}\cdot\frac{z-1}{z+1} = \frac{2}{T_s}\cdot\frac{1-z^{-1}}{1+z^{-1}} \tag{2.19}$$

De Backward Euler gebruikt de volgende substitutie:

$$s \approx \frac{z-1}{z\cdot T_s} \tag{2.20}$$

De Forward Euler gaat als volgt:

$$s \approx \frac{z-1}{T_s} \tag{2.21}$$

Als we bijvoorbeeld terugkijken naar vergelijking (2.3):

$$H_r(s) = \frac{U(s)}{E(s)} = K_c\left(1+\frac{1}{T_i\cdot s}+T_d\cdot s\right) \tag{2.3}$$

Dan kunnen we nu s substitueren met een van de bovenstaande vormen. Als voorbeeld gebruiken we de Backward-Euler-methode. We kunnen nu de overdracht bepalen van de regelaar in het z-domein door elke s in vergelijking (2.3) te vervangen door $\frac{z-1}{z \cdot T_s}$:

$$H_r(z) = K_c \left(1 + \frac{1}{T_i \cdot \frac{z-1}{z \cdot T_s}} + T_d \cdot \frac{z-1}{z \cdot T_s} \right) \tag{2.22}$$

Omdat we uiteindelijk de formule in de vorm $U(z) = \ldots$ willen hebben, proberen we alles binnen een enkele breuk te krijgen, aangezien $H(z) = \frac{U(z)}{E(z)}$.

Dubbele breuk wegwerken en constanten naar voren halen:

$$H_r(z) = K_c \left(1 + \frac{T_s}{T_i} \cdot \frac{z}{z-1} + \frac{T_d}{T_s} \cdot \frac{z-1}{z} \right) \tag{2.23}$$

Omwerken naar gemeenschappelijke noemer:

$$H_r(z) = K_c \left(1 + \frac{T_s}{T_i} \cdot \frac{1}{1-z^{-1}} + \frac{T_d}{T_s} \cdot \left(1 - z^{-1}\right) \right) \tag{2.24}$$

$$= K_c \left(1 + \frac{\frac{T_s}{T_i}}{1-z^{-1}} + \frac{\frac{T_d}{T_s} \cdot \left(1 - z^{-1}\right)}{1} \right) \tag{2.25}$$

$$= K_c \left(\frac{1-z^{-1}}{1-z^{-1}} + \frac{\frac{T_s}{T_i}}{1-z^{-1}} + \frac{\frac{T_d}{T_s} \cdot \left(1 - z^{-1}\right)^2}{1-z^{-1}} \right) \tag{2.26}$$

$$= K_c \left(\frac{1 - z^{-1} + \frac{T_s}{T_i} + \frac{T_d}{T_s} \cdot \left(1 - 2z^{-1} + z^{-2}\right)}{1 - z^{-1}} \right) \tag{2.27}$$

Dus:

$$H_r(z) = \frac{U(z)}{E(z)} = K_c \left(\frac{1 - z^{-1} + \frac{T_s}{T_i} + \frac{T_d}{T_s} \cdot \left(1 - 2z^{-1} + z^{-1}\right)}{1 - z^{-1}} \right) \tag{2.28}$$

Als we kruislings vermenigvuldigen vinden we:

$$U(z) - U(z) \cdot z^{-1} = K_c \Big(E(z) - E(z) \cdot z^{-1} + \frac{T_s}{T_i} \cdot E(z) + \frac{T_d}{T_s} \cdot \left(E(z) - 2E(z) \cdot z^{-1} + E(z) \cdot z^{-2}\right) \Big) \tag{2.29}$$

Terugtransformeren van het z-domein naar het tijddomein geeft:

$$u[k] - u[k-1] = K_c \Big(e[k] - e[k-1] + \frac{T_s}{T_i} \cdot e[k] + \frac{T_d}{T_s} \cdot \left(e[k] - 2e[k-1] + e[k-2]\right) \Big) \tag{2.30}$$

Deze vergelijking kan ook geschreven worden als:

$$u[k] = u[k-1] + q_0 e[k] + q_1 e[k-1] + q_2 e[k-2] \tag{2.31}$$

met:

$$q_0 = K_c\left(1 + \frac{T_s}{T_i} + \frac{T_d}{T_s}\right) \qquad q_1 = -K_c\left(1 + \frac{2T_d}{T_s}\right) \qquad q_2 = \frac{K_c T_d}{T_s} \tag{2.32}$$

Wat valt je op aan vergelijking (2.31) en vergelijking (2.32) in vergelijking met vergelijking (2.17) en de bij opdracht 2 gevonden waarden voor q_0, q_1 en q_2?

Opdracht 3: Forward Euler en BLT.

Probeer de bovenstaande afleiding ook te maken met:

a) de Forward-Euler-methode;

b) de Bilineaire-Transformatie-methode.

Ga daarbij uit van vergelijking (2.3).

2.6 Samenvatting

In dit hoofdstuk hebben we gezien hoe we van de tijdcontinue variant van een PID-regelaar kunnen overgaan naar een tijddiscrete variant van een PID-regelaar. Met behulp van numerieke methoden kunnen we het positiealgoritme van de PID-regelaar afleiden. Helaas kent deze vorm een aantal implementatiebeperkingen. Met behulp van het snelheidsalgoritme kunnen we een efficiëntere variant afleiden van de PID-regelaar. Verder kunnen we ook gebruik maken van directe transformatiemethodieken, zoals de Forward Euler, Backward Euler en de Bilineaire Transformatie. Deze methodieken komen van pas in het volgende hoofdstuk, als we verbeteringen aan gaan brengen aan de regelaars.

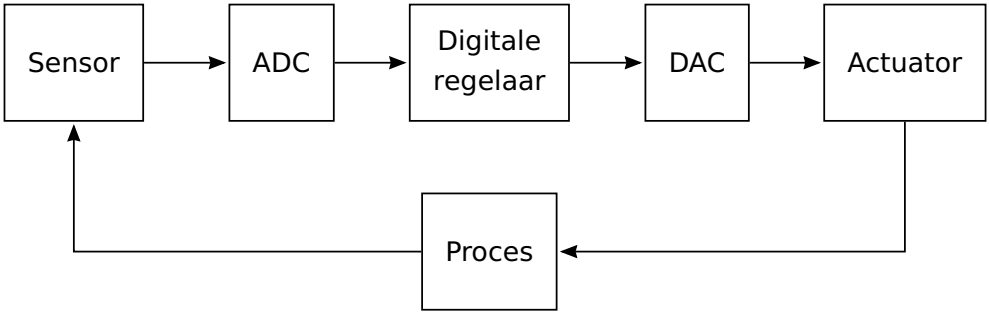
3

Verbeteringen aan digitale PID-regelaars

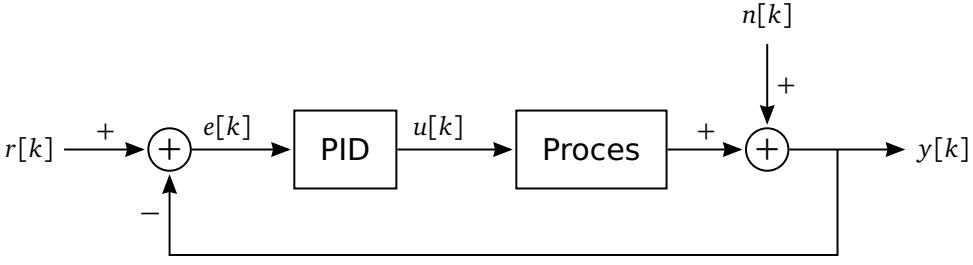
Vergelijking (2.1) beschrijft de algemene vergelijking voor een PID-regelaar in het tijdcontinue domein. In het tijddiscrete domein komt dit overeen met het doorrekenen van de bijbehorende differentievergelijking. Bij de praktische implementatie van zo'n regelaar lopen we tegen problemen aan, die bij een analoge regelaar niet altijd op zullen treden.

Een model van een typisch digitaal geregeld systeem is gegeven in figuur 3.1.

Een bepaalde eigenschap die we willen regelen van het systeem wordt gemeten d.m.v. een sensor. Deze waarde wordt door een ADC gesampled. Een actuator zorgt voor de nieuwe input van het systeem. De actuator wordt aangestuurd vanuit een DAC, die de proceswaarde vanuit de digitale regelaar ontvangt. Voor de regelaar gezien is dus dankzij de DAC en de ADC de hele omgeving tijddiscreet geworden, en kunnen we het in figuur 2.1 getoonde model tekenen zoals in figuur 3.2 gegeven is.



Figuur 3.1: Algemeen model van een PID-geregeld systeem.



Figuur 3.2: Model van een tijddiscreet PID-geregeld systeem

Bij een analoge regelaar zijn de meeste terugkoppellussen op een elektrische manier begrenst in bandbreedte. Hierdoor is het niet mogelijk om zeer grote en snelle veranderingen in bijv. het foutsignaal te krijgen, waardoor de regeling instabiel gedrag kan vertonen.

Het berekenen van de differentievergelijking gebeurt volgens vergelijking (2.31). Grote verandering in bijvoorbeeld een variabele of een parameter kunnen leiden tot grote wijzigingen in de output van de PID-regelaar. Wat verder ook nog meespeelt, is dat de ingelezen waarde $y[k]$ vaak beïnvloed is door ruis $n[k]$.

Het negatieve effect van grote en snelle wijzigingen in het foutsignaal kan onderdrukt worden door het errorsignaal $e[k]$ een specifieke behandeling te geven, alvorens het door de PID-regelaar gebruikt gaat worden ('preprocessing'). Een andere mogelijkheid is het aanpassen van de differentievergelijking van de PID-

regelaar, zodat een deel van deze preprocessing direct in het algoritme van de PID-regelaar gebeurt. Deze mogelijkheden worden in dit hoofdstuk verder uitgewerkt.

Aan het eind van dit hoofdstuk komen nog enkele praktische aspecten aan bod.

3.1 Filteren van de differentiërende actie

Vaak zijn er verstoringen in de gemeten waarden van de procesoutput $y[k]$. Dit zijn vaak relatief hoge frequentiecomponenten. Indien een D-actie gebruikt wordt in de PID-regelaar, dan reageert deze op veranderingen in het errorsignaal. Daarmee wordt deze actie dan ook gevoelig voor deze hoge frequentiecomponenten. Dit kan weer leiden tot een te grote uitsturing aan de uitgang van de PID-regelaar. Dit kan soms een ongewenst effect geven.

De D-actie kan begrensd worden, om een te grote uitsturing te voorkomen. Meer gebruikelijk is het toevoegen van een eenvoudig 1^e- of 2^e-orde laagdoorlaatfilter, zodat de versterking lager wordt voor hogere frequenties.

We gaan uit van de in vergelijking (2.3) gegeven Laplace-getransformeerde van de overdracht van een PID-regelaar:

$$H_r(s) = \frac{U(s)}{E(s)} = K_c\left(1 + \frac{1}{T_i \cdot s} + T_d \cdot s\right) \tag{2.3}$$

We kennen de overdracht in het Laplace-domein van een laagdoorlaatfilter (**l**ow-**p**ass):

$$H_{lp}(s) = \frac{1}{1 + \gamma \cdot s} \tag{3.1}$$

Als we dit toevoegen aan de D-actie van vergelijking (2.3) resulteert dit in:

$$H_{rl}(s) = \frac{U(s)}{E(s)} = K_c\left(1 + \frac{1}{T_i \cdot s} + T_d \cdot \frac{s}{1 + \gamma \cdot s}\right) \tag{3.2}$$

Hierbij is γ een kleine tijdconstante die in de praktijk meestal ingesteld wordt op ongeveer 10% van de T_d-waarde (bij de meeste industriële PID-regelaars varieert γ tussen 6,25% en 12,5% van de waarde van T_d [1]).

Als we vergelijking (3.2) willen implementeren in een digitale regelaar dan moet deze vergelijking getransformeerd worden naar het z-domein. Als voorbeeld hebben we hier gebruik gemaakt van de directe transformatie via de Backward-Euler-methode, zie vergelijking (2.20).

Uiteindelijk komen we bij deze afleiding uit op de volgende overdracht (een volledige afleiding voor Backward Euler wordt in het college besproken):

$$H_r(z) = K_c \left(\frac{\gamma + T_s + \frac{\gamma T_s}{T_i} + \frac{T_s^2}{T_i} + T_d - z^{-1}\left(2\gamma + T_s + \frac{\gamma T_s}{T_i} + 2T_d\right) + z^{-2}(T_d + \gamma)}{\gamma + T_s - z^{-1}(T_s + 2\gamma) + \gamma z^{-2}} \right) \tag{3.3}$$

Door dezelfde stappen toe te passen als in de voorgaande voorbeelden kunnen we deze overdracht terugtransformeren naar het tijddiscrete domein. Let er op dat in de bovenstaande vergelijking onder de deelstreep nu factoren bij de verschillende macht-termen van z staan. Als we dit omwerken naar de algemene vorm krijgen we ook coëfficiënten in het recursieve deel. Deze noemen we p.

De uitgang van de regelaar wordt nu gegeven door:

$$u[k] = p_1 \cdot u[k-1] + p_2 \cdot u[k-2] + q_0 \cdot e[k] + q_1 \cdot e[k-1] + q_2 \cdot e[k-2] \tag{3.4}$$

Met de volgende coëfficiënten:

$$p_1 = \frac{T_s + 2\gamma}{T_s + \gamma} \qquad p_2 = \frac{-\gamma}{T_s + \gamma}$$

$$q_0 = K_c\left(1+\frac{T_s}{T_i}+\frac{T_d}{T_s+2\gamma}\right) \qquad q_1 = \frac{-K_c}{T_s+\gamma}\left(T_s+\gamma\left(2+\frac{T_s}{T_i}\right)+2T_d\right)$$

$$q_2 = K_c\frac{T_d+\gamma}{T_s+\gamma} \tag{3.5}$$

Als we in plaats van de Backward-Euler-methode de bilineaire directe transformatiemethode toepassen (zie vergelijking (2.19)), dan krijgen we de volgende coëfficiënten:

$$p_1 = \frac{4\gamma}{T_s+2\gamma} \qquad p_2 = \frac{T_s-2\gamma}{T_s+2\gamma}$$

$$q_0 = K_c\left(1+\frac{T_s}{2T_i}+\frac{2T_d}{T_s+2\gamma}\right) \qquad q_1 = K_c\left(\frac{\frac{T_s^2}{T_i}-4\gamma-4T_d}{T_s+2\gamma}\right)$$

$$q_2 = K_c\frac{2\gamma-T_s+\frac{T_s^2}{2T_i}-\frac{\gamma T_s}{T_i}+2T_d}{2\gamma+T_s} \tag{3.6}$$

Opdracht 4: LPF in de D-actie

Transformeer nu vergelijking (3.2) zelf naar het tijddiscrete domein met behulp van de directe transformatie volgens

a) de Backward-Euler-methode. Als het goed is levert dit vergelijking (3.4) op met de coëfficiënten uit vergelijking (3.5);

b) de Backward-Euler-methode;

c) de bilineaire methode. Als het goed is levert dit vergelijking (3.4) op met de coëfficiënten uit vergelijking (3.6).

3.2 PID-regelaar type A, B en C

In paragraaf 2.5 hebben we het snelheidsalgoritme van een PID-regelaar met behulp van Backward Euler afgeleid. De differentievergelijking werd toen:

$$u[k]-u[k-1]=K_c\Bigg(e[k]-e[k-1]+\frac{T_s}{T_i}\cdot e[k]+ \frac{T_d}{T_s}\cdot\big(e[k]-2e[k-1]+e[k-2]\big)\Bigg) \quad (2.30)$$

Hierbij is $e[k]$ het errorsignaal. Dit is gelijk aan het setpoint (de referentiewaarde) minus de gemeten proceswaarde $y[k]$. Het probleem bij deze implementatie van een PID-regelaar is dat, als de referentiewaarde op een andere waarde ingesteld wordt, dit direct invloed heeft op de grootte van de proportionele actie, maar ook op de grootte van de differentiërende actie. Hele snelle veranderingen in het setpoint kunnen dus voor ongewenste effecten in de P- en D-actie zorgen! Een type regelaar waarbij dit probleem niet verholpen is (zoals in vergelijking (2.30)) wordt ook wel een 'type A regelaar' genoemd.

Deze type A regelaar kan minder gevoelig gemaakt worden voor setpoint veranderingen door het setpoint te verwijderen uit de differentiërende actie. Dit kunnen we als volgt beredeneren; omdat we alleen willen regelen met veranderingen op de lange termijn, kunnen we aannemen dat r[k] (het setpoint) over een korte periode hetzelfde blijft.

Dit betekent dat we kunnen aannemen dat: $r[k]\approx r[k-1]\approx r[k-2]\approx\cdots$.

Aangezien $e[k]=r[k]-y[k]$ kunnen we een substitutie maken in de D-actie. Nu gaat vergelijking (2.30) over in:

$$u[k]-u[k-1]=K_c\left(e[k]-e[k-1]+\frac{T_s}{T_i}e[k]\right)+$$

$$K_c\left(\frac{T_d}{T_s}\Big(r[k]-y[k]-2\big(r[k-1]-y[k-1]\big)+\big(r[k-2]-y[k-2]\big)\Big)\right) \tag{3.7}$$

Observeer nogmaals dat $r[k] \approx r[k-1] \approx r[k-2] \approx \cdots$. We kunnen nu een aantal $r[k]$'s in de D-actie tegen elkaar wegstrepen. Dit resulteert in het wegvallen van het setpoint uit de D-actie. Deze vorm wordt ook wel een 'Type B regelaar' genoemd:

$$u[k]-u[k-1]=K_c\Bigg(e[k]-e[k-1]+\frac{T_s}{T_i}e[k]+ \frac{T_d}{T_s}\big(-y[k]+2y[k-1]-y[k-2]\big)\Bigg) \tag{3.8}$$

We kunnen ditzelfde principe ook toepassen op de proportionele actie. In dat geval gaat de bovenstaande differentievergelijking over in:

$$u[k]-u[k-1]=K_c\Bigg(-y[k]+y[k-1]+\frac{T_s}{T_i}e[k]+ \frac{T_d}{T_s}\big(-y[k]+2y[k-1]-y[k-2]\big)\Bigg) \tag{3.9}$$

Een regelaar waarbij het setpoint zowel uit de P- als de D-actie is weggehaald wordt een 'Type C regelaar' genoemd.

3.3 Integrator wind-up

Een ander aandachtspunt bij het implementeren van digitale regelaars is het verschijnsel van de zogenoemde integrator wind-up. Gegeven vergelijking (2.11),

die een discrete PID-regelaar voorstelt, die met behulp van FRM is bepaald:

$$u[k] = K_c\left(e[k] + \frac{T_s}{T_i}\sum_{i=1}^{k} e[i-1] + \frac{T_d}{T_s}\big(e[k]-e[k-1]\big)\right)$$

$$\text{met } k = \left\lfloor \frac{t}{T_s} + \frac{1}{2} \right\rfloor \quad (2.11)$$

Stel je de situatie voor dat het errorsignaal (om wat voor reden dan ook[2]) een flinke tijd positief is. Dit errorsignaal wordt geïntegreerd in de tijd. In het geval van vergelijking (2.11) betekent dit dat de som bepaald wordt van alle voorgaande errorwaarden. De integrerende actie kan in dit geval behoorlijk groot worden.

Het beste is om dit te illustreren met een voorbeeld: stel dat de errorwaarde $e[i]$ gelijk is aan 0,05 voor de laatste 50 waarden. De totale somwaarde van de integrerende actie bedraagt in dit geval 50 * 0,05 = 2,5; oftewel 250 % (uitgaande van het feit dat we daadwerkelijk over de laatste 50 intervallen sommeren om de integraal te bepalen).

Stel nu dat het errorsignaal $e[i]$ vervolgens net iets kleiner wordt dan 0, bijvoorbeeld $-0,02$. Dan duurt het nog minstens 75 samples voordat het integrerende deel van het outputsignaal weer terug bij 100% is! Dit is natuurlijk een onacceptabel resultaat, dat kan leiden tot een langdurig overschot in het te regelen proces.

Dit fenomeen staat bekend als integrator wind-up. De oplossing hiervoor is om de totale waarde van de integrerende actie te beperken op de maximale waarde van het outputsignaal. In het voorbeeld heeft het dus zin om de integrerende actie te beperken tot 100%.

[2] In de praktijk gebeurt dit vaak door zogenoemde actuatorverzadiging. Het proces is dan niet in staat om de output van de regelaar te volgen omdat de actuator al maximaal aangestuurd wordt.

3.4 Beperkte precisie

Binnen een digitaal systeem werken we niet alleen met discrete stappen in de tijd, maar ook met discrete stappen in de waarden van het setpoint, de procesvariabele, enz., omdat de waarden gerepresenteerd worden door een eindig aantal bits.

In de praktijk wordt vaak gerekend met zogenoemde fixed-point getallen omdat fixed-point berekeningen sneller zijn dan floating-point berekeningen en ook minder energie kosten. Bij het gebruik van fixed-point getallen moeten we goed opletten dat we de berekeningen met voldoende precisie uitvoeren.

Een voorbeeld: stel dat we gebruik maken van een 12-bit AD-converter. Iedere stap in het $e[k]$-signaal is nu $\frac{1}{2^{12}} = \frac{1}{4096}$ groot.

Stel nu dat we de volgende vergelijking willen uitrekenen: $\Delta u_i[k] = \frac{K_c \cdot T_s}{T_i} e[k]$, waarbij gegeven is dat $K_c \cdot T_s$ gelijk aan 1 is en T_i gelijk aan 3550 is.

Stel nu dat we dit naïef programmeren met behulp van 16-bits integer getallen als $\Delta u_i[k] = \frac{e[k]}{3550}$. Het errorsignaal zal dan minimaal 87% van de full-scale-waarde (4095 voor een 12-bit ADC) moeten bedragen om voor $\Delta u_i[k]$ een waarde groter dan 0 berekend te krijgen.

Een minder naïeve implementatie zal gebruik maken van 16-bits fixed-point getallen met 12 bits voor en 4 bits na de decimale punt. We moeten dan $e[k]$ eerst vermenigvuldigen met 16 (4 plaatsen naar links schuiven). Er is geen gevaar voor overflow omdat het $e[k]$-signaal afkomstig is van een 12-bit ADC. De berekening wordt dan uitgevoerd als $\Delta u_i[k] = \left(\frac{e[k]*16}{3550}\right)$. Het errorsignaal zal dan slechts minimaal 5,5% van de full-scale-waarde (4095 voor een 12-bit ADC) moeten bedragen om een waarde groter dan 0 berekend te krijgen. Als dit nog steeds niet acceptabel is zal met 32-bits fixed-point getallen gerekend moeten worden waarbij 20 bits voor en 12 bits na de decimale punt gebruikt worden. In dit geval kunnen we de berekening uitvoeren als: $\Delta u_i[k] = \left(\frac{e[k]*4096}{3550}\right)$ en wordt de berekening met de hoogst mogelijke precisie uitgevoerd.

Bij een incrementele regelaar (snelheidsalgoritme) wordt alleen het verschil uitgerekend. Hierdoor kan de nauwkeurigheid wat groter worden dan bij een absolute regelaar (positie algoritme).

Het werken met fixed-point berekeningen is soms onontkoombaar, bijv. omdat de snelheid of het energiegebruik zeer belangrijk is. Maar met het krachtiger worden van microcontrollers verdient het aanbeveling om te kijken of het mogelijk is om toch met floating-point berekeningen te werken. Het vereenvoudigt het implementeren van de regelaar in software aanzienlijk.

3.5 Industriële regelaars

De ontwikkeling van elektronica en in het bijzonder de computertechnologie (microcontrollers) heeft er in geresulteerd dat vrijwel alle commercieel verkrijgbare regelaars digitaal zijn en niet langer analoge elementen bevatten, zoals dat jaren geleden nog wel gebruikelijk was.

Praktische regelaars bestaan uit een of meerdere PID-regelaars. PID-regelaars zijn vaak beschikbaar als functieblokken in PLC's en regelsystemen. De output van zo'n regelaar is meestal analoog (analoge output, AO) of digitaal (digitale output, DO). Meestal wordt de digitale output gebruikt om een apparaat aan of uit te schakelen. Ook zijn er regelaars met 2 digitale outputs: hierbij wordt de ene output gebruikt om een apparaat aan of uit te schakelen en de andere output wordt gebruikt om de richting aan te geven (omhoog/omlaag, links/rechts, verwarmen/koelen etcetera).

Het analoge outputsignaal genereert meestal een spanningswaarde tussen 0 V en een maximale spanning. Het digitale outputsignaal is meestal een in pulsbreedte gemoduleerd signaal van een bepaalde frequentie. De waarde van de regelaar bepaalt de duty-cycle van het outputsignaal (bijv. als de output van de regelaar 20% is, dan is het signaal 20% van de periodetijd hoog en 80% van de periodetijd laag).

Naast deze standaard outputs is er meestal ook een communicatiepoort aanwezig. Dit is vaak een RS232/485 seriële interface.

Belangrijk bij iedere PID-regelaar zijn de instellingen. Naast de gebruikelijke waarden (K_c, T_i, T_d, T_s en γ voor het D-filter) kunnen vaak ook de boven- en ondergrens van het outputsignaal van de regelaar ingesteld worden.

Daarnaast kan de polariteit van de regelaar vaak ingesteld worden. Hiermee wordt in feite het teken van de versterking ingesteld (+ of -). Hiervoor worden de termen direct-acting (ook wel non-inverting of heating-loop genoemd) en reverse-acting (ook wel inverting of cooling-loop genoemd) gebruikt.

Bij een direct-acting controller is sprake van een positieve versterking. Dit houdt in dat wanneer de output van de PID-regelaar verhoogd wordt, de procesvariabele ook groter zal worden. Bij een verwarmingssysteem zal een grotere waarde van de output van de regelaar meestal leiden tot een hogere temperatuur.

Bij een reverse-acting controller is sprake van een negatieve versterking. Denk hier bijvoorbeeld aan een regelaar voor een koelkast. Indien de output van de PID-regelaar verhoogd wordt, zal de procesvariabele (de koelkast temperatuur) juist lager worden.

Andere veel voorkomende functies/instellingen van de huidige generatie PID-regelaars zijn:

- Kunnen schakelen tussen handmatige bediening en automatisch regelen.
- De mogelijkheid hebben om de regelaar te programmeren, waarbij het setpoint in de tijd geprogrammeerd kan worden (bijv. om een bepaalde cyclus af te lopen).
- De mogelijkheid om meerdere regelaars achter elkaar te zetten (cascade-loop).
- De mogelijkheid om de regelaar zelf de optimale PID-parameters te laten bepalen (auto-tuning). Meer hierover in het volgende hoofdstuk.

4

Het vinden van de PID-parameters

Nu we de tijddiscrete regelaars hebben afgeleid en ook enkele interessante eigenschappen en verbeteringen aan de regelaars hebben besproken, rest de vraag: “Hoe stellen we de regelaar eigenlijk in?”. In dit hoofdstuk wordt hier op verschillende manieren (voor simpel te regelen systemen) antwoord op gegeven.

Bij het vinden van de optimale set parameters voor een digitale PID-regelaar denken we al snel aan moderne digitale signaalbewerking. Toch is de basis hiervoor reeds begin jaren ’40 gelegd, in een tijd dat er nog geen digitale regelaars waren. De regelaars in die tijd waren mechanische en pneumatische apparaten.

Het was in 1942 dat de heren Ziegler en Nichols hun wereldberoemde artikel publiceerden: “settings for automatic controllers” [9]. De inhoud van dit artikel wordt nog steeds gebruikt. Sterker nog: in de komende paragrafen zullen we de essentie van hun verhaal behandelen.

4.1 Stap-responsie van een eerste-orde-proces met tijdvertraging

Veel industriële processen, die geregeld dienen te worden, kunnen gemodelleerd worden als een eerste-orde-systeem met tijdvertraging (FOPTD = First Order Process with Time Delay). De overdrachtsfunctie van zo'n eerste-orde-systeem wordt dan voorgesteld door:

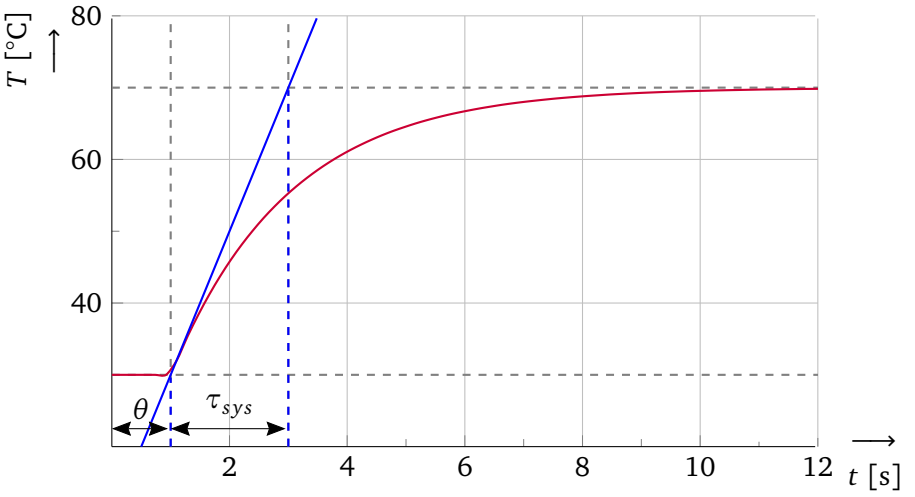
$$G(s) = \frac{Y(s)}{U(s)} = \frac{K_{sys}e^{-\theta s}}{\tau_{sys}s + 1} \tag{4.1}$$

Hierin is:

- $G(s)$: de overdrachtsfunctie van het te regelen proces.
- $Y(s)$: de output van het te regelen proces, dit wordt ook wel de procesvariabele PV genoemd. De eenheid hangt af van het te regelen proces. Stel bijvoorbeeld dat er een temperatuur geregeld moet worden dan is de eenheid bijvoorbeeld °C.
- $U(s)$: de output van de PID-regelaar en de input voor het te regelen proces. Deze waarde wordt vaak uitgedrukt in een percentage van het bereik van het outputsignaal (0 tot 100%).
- Dode tijd θ: de tijdvertraging van het te regelen proces in seconden.
- K_{sys}: de versterking van het te regelen proces. De eenheid hangt af van het te regelen proces. Stel bijv. dat $Y(s)$ een temperatuur voorstelt, dan is de eenheid van K_{sys} gelijk aan °C/%.
- τ_{sys}: de tijdconstante van het te regelen proces in seconden.

De stapresponsie van een FOPTD met $K_{sys} = 1$ °C/%, $\tau_{sys} = 2$ s en $\theta = 1$ s op een stap van 30 naar 70% is gegeven in figuur 4.1.

De dode tijd θ is eenvoudig uit de stapresponsie af te lezen. Je ziet dat in figuur 4.1 geldt $\theta = 1$ s. Als we de raaklijn tekenen in het eindpunt van de dode tijd (de



Figuur 4.1: De stapresponsie van een FOPTD.

blauwe lijn in figuur 4.1) dan kunnen we ook τ_{sys} aflezen. Je ziet dat in figuur 4.1 geldt $\tau_{sys} = 2$ s.

Bedenk dat vergelijking (4.1) een model is voor een proces dat we met een PID-regelaar willen regelen. Geen enkel praktisch proces zal zich exact zo gedragen als het model. In de volgende paragraaf zullen we de methode bespreken die bedacht is door Ziegler en Nichols waarmee de instellingen voor de PID-regelaar gevonden kunnen worden voor een proces waarvan de response *lijkt* op de response van een FOPTD.

4.2 Open-loop-methode van Ziegler-Nichols

Stel dat we een proces dat zich *ongeveer* gedraagt als een FOPTD willen regelen met een PID-regelaar. We kunnen de instellingen van de PID-regelaar volgens de open-loop-methode[3] van Ziegler en Nichols als volgt vinden[4]:

1. Zorg ervoor dat de regellus (zie figuur 2.1) doorbroken wordt. Dit kan in de praktijk gedaan worden door de PID-regelaar op handbediening te zetten.
2. Regel het proces handmatig zodanig dat de uitgang een waarde heeft die in de praktijk vaak als setpoint wordt gebruikt en wacht tot de uitgangswaarde constant blijft.
3. Breng nu handmatig een stap op de output van de regelaar (dat is de input van het proces) aan en registreer de stapresponsie. Het kiezen van een geschikte stapgrootte is een kwestie van ervaring. Het verdient in principe de voorkeur om de stapgrootte zo groot mogelijk te nemen, dat is namelijk het meest nauwkeurig. Aan de andere kant willen we het proces dat we regelen in sommige gevallen ook niet te veel verstoren alleen maar om de regelaar in te kunnen stellen. We nemen als voorbeeld een temperatuurregeling (deze gedragen zich in de praktijk vrijwel altijd als een FOPTD) met een met K_{sys} van 1 °C/% die we handmatig hebben ingesteld op 30% wat dus overeenkomt met 30 °C. Vervolgens brengen we een stap aan van 30% naar 70%. De stapgroote in % noemen we Δp. In dit voorbeeld geldt dus $\Delta p = 70\% - 30\% = 40\%$. De stapresponsie is gegeven in figuur 4.2. Je ziet dat deze stapresponsie niet helemaal gelijk is aan de stapresponsie van een FOPTD (zie figuur 4.1) maar dat het er voldoende op lijkt om de FOPTD als model te gebruiken.
4. Uit de stapresponsie moeten we nu 2 waarden aflezen:
    - De genormaliseerde stijging van de stapresponsie. In dit document wordt deze grootheid aangeduid met a^*. Om deze te kunnen vinden

[3] De Engelse benaming is 'open-loop method'. Dit kan in het Nederlands vertaald worden als 'openkringmethode'. Omdat de Engelse termen 'open-loop' in de regeltechniek, ook in het Nederlands, vaak voorkomt wordt in dit dictaat de term open-loop-methode gebruikt.

moeten we eerst de stijging van de stapresponsie, aangeduid met a vinden. Deze kan gevonden worden door de helling te bepalen van de meest steile raaklijn aan de stapresponsie die we kunnen vinden[4]. In de tijd van Ziegler en Nichols was dit een kwestie van schatten, maar tegenwoordig kunnen we de stapresponse inlezen in een computer en de maximale waarde van de afgeleide opzoeken[5]. We zouden dit kunnen doen m.b.v. van MATLAB, maar moderne digitale regelaars kunnen dit ook zelf. De waarde van a^* is gelijk aan $a/\Delta p$.

- De dode tijd θ: hiervoor nemen we de tijd tussen de start van de stap en het tijdstip waarop de meest steile raaklijn aan de stapresponsie de beginwaarde van de output snijdt.

Opdracht 5: Bepalen van a^* en θ

Bepaal zelf de waarde van a^* en θ uit de in figuur 4.2 gegeven stapresponsie.

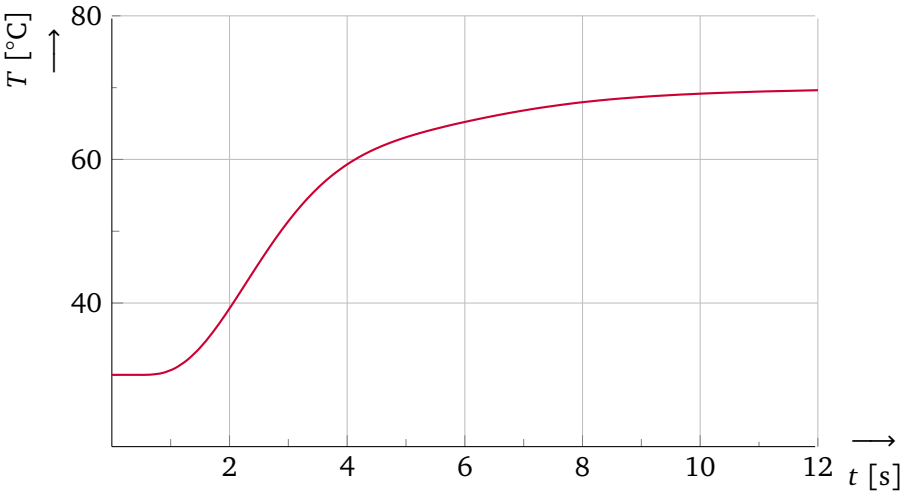
5. Op basis van deze waarden (a^* en θ) waren Ziegler en Nichols destijds al in staat om een PID- of PI-regelaar zodanig in te stellen, dat deze een goed regelgedrag liet zien. De PID-regelparameters en PI-regelparameters voor deze open-loop-methode van Ziegler-Nichols staan in tabel 4.1 vermeld.

Opdracht 6: Bepalen van de parameters van de PID-regelaar

Bepaal met behulp van de bij opdracht 5 bepaalde waarden van a^* en θ de parameters voor de PID-regelaar van dit proces volgens de open-loop-methode van Ziegler-Nichols.

[4] We zoeken dus de helling van de raaklijn in het buigpunt van de stapresponsie.

[5] We kunnen het tijdstip waarop de afgeleide maximaal is eenvoudig vinden door de tweede afgeleide van de stapresponsie gelijk te stellen aan nul.



Figuur 4.2: De stapresponsie van het te regelen proces.

Tabel 4.1: PID-parameters volgens de open-loop-methode van Ziegler-Nichols.

Methode / Parameters	K_c [%/?]	T_i [s]	T_d [s]
Ziegler-Nichols open-loop (PID)	$K_c = \frac{1,2}{\theta \cdot a^*}$	$T_i = 2,0 \cdot \theta$	$T_d = 0,5 \cdot \theta$
Ziegler-Nichols open-loop (PI)	$K_c = \frac{0,9}{\theta \cdot a^*}$	$T_i = 0,5 \cdot \theta$	–

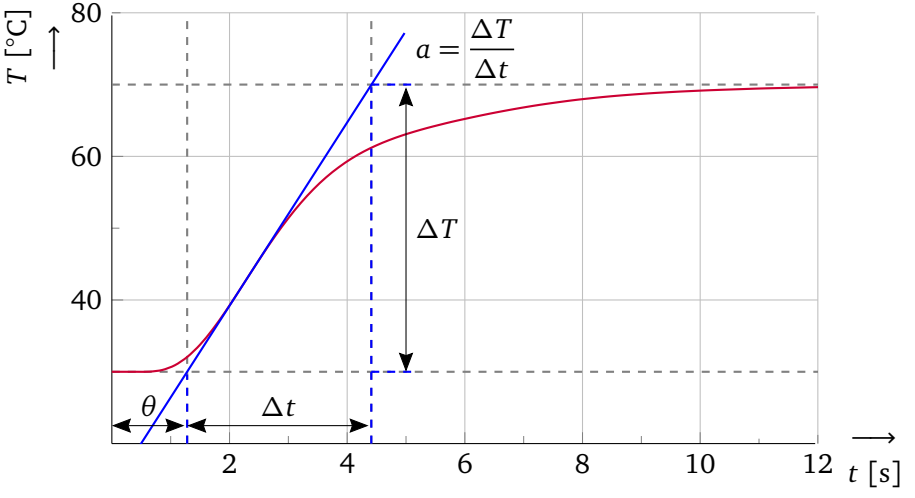
Het voordeel van deze methode is de eenvoud. Door een stap op het systeem te zetten, kan eigenlijk al heel snel een redelijke instelling bepaald worden voor de PID-regelaar. Van dit principe maken veel auto-tuning functies in commercieel verkrijgbare regelaars dan ook dankbaar gebruik. Merk op dat het bij deze methode *niet* nodig is om de stapresponsie helemaal tot de eindwaarde te laten lopen. Zodra we het buigpunt gezien hebben kunnen we stoppen.

De Ziegler-Nichols-methode is erg eenvoudig om te gebruiken maar levert in de praktijk meestal niet het gewenste gedrag op [5, blz. 32-4]. De regelaar geeft in de

meeste gevallen te weinig demping en de regelaar is te gevoelig voor variaties in de procesparameters. Om deze reden zijn er vele alternatieve methoden bedacht waarvan we er enkele zullen bespreken in paragraaf 4.4.

4.3 Antwoorden van opdrachten 5 en 6

In figuur 4.3 zie je hoe de waarden van a en θ gevonden kunnen worden met behulp van de raaklijn door het buigpunt van de stapresponsie.



Figuur 4.3: De stapresponsie van het te regelen proces met de raaklijn door het buigpunt.

Uit figuur 4.3 lezen we af $\theta \approx 1,3$ s, $\Delta t \approx 4,4-1,3 = 3,1$ s en $\Delta T = 70-30 = 40$ °C. We kunnen nu a berekenen: $a = \frac{\Delta T}{\Delta t} = 40/3,1 = 12,9$ °C/s.

De waarde van a^* kunnen we nu eenvoudig berekenen: $a^* = a/\Delta p = 12,9/40 = 0,32$ °C/(s·%). Met behulp van tabel 4.1 kunnen we nu de PID-parameters een-

voudig berekenen:

$$K_c = \frac{1,2}{\theta \cdot a^*} = \frac{1,2}{1,3 \cdot 0,32} = 2,88\ \%/°C \tag{4.2}$$

$$T_i = 2,0 \cdot \theta = 2,0 \cdot 1,3 = 2,6\ \text{s} \tag{4.3}$$

$$T_d = 0,5 \cdot \theta = 0,5 \cdot 1,3 = 0,65\ \text{s} \tag{4.4}$$

4.4 Andere open-loop-methoden

Bij de in paragraaf 4.2 besproken methode schatten we twee procesparameters waarmee we de PID-parameters bepalen. Betere instellingen voor de PID-parameters kunnen gevonden worden door drie in plaats van twee procesparameters te bepalen. We schatten niet alleen de versterking en de dode tijd maar ook de tijdconstante van het systeem. De eersten die dit bedachten waren Cohen en Coon [3] in 1953.

We bepalen weer de stapresponsie van de open-loop, net zoals in de vorige paragraaf. We laten de responsie nu echter wel lopen totdat de eindwaarde bereikt is. Dit kan overigens erg lang duren, zeker indien het te regelen proces beschikt over een grote tijdconstante. Voor deze meting is een nauwkeurig eindwaarde van groot belang (in figuur 4.2 is dat 70 °C). Hoe nauwkeuriger deze gemeten wordt, des te beter. Op basis hiervan wordt een ΔT bepaald ($\Delta T = T_{start} - T_{eind}$).

De tijdconstante van een systeem (τ_{sys}) is bij een zuiver eerste-orde-systeem het tijdstip waarop de stapresponsie gestegen is met 63,2% van ΔT[6]. Bij een systeem met dode tijd θ geldt dat het tijdstip waarop de stapresponsie gestegen is met

[6] De stapresponsie van een zuiver eerste-orde-systeem in procenten van de verandering van het uitgangssignaal is gelijk aan $(1 - e^{\frac{-t}{\tau_{sys}}}) \cdot 100\%$. Als we $t = \tau_{sys}$ invullen geeft dit ongeveer 63,2% en als we $t = \frac{1}{3}\tau_{sys}$ invullen geeft dit ongeveer 28,3%.

63,2% van ΔT gelijk is aan $\theta + \tau_{sys}$. Daarnaast wordt bij een zuiver eerste-orde-systeem een derde van de tijdconstante bereikt op het moment dat de stapresponsie gestegen is met 28,3% van ΔT. Bij een systeem met dode tijd θ geldt dat het tijdstip waarop de stapresponsie gestegen is met 28,3% van ΔT gelijk is aan $\theta + \frac{1}{3}\tau_{sys}$.

Als dus beide bovengenoemde tijdstippen (laten we ze respectievelijk t_1 en t_2 noemen) bepaald worden, dan hebben we een stelsel van twee vergelijken met twee onbekenden:

$$\begin{cases} t_1 = \theta + \frac{1}{3}\tau_{sys} \\ t_2 = \theta + \tau_{sys} \end{cases} \tag{4.5}$$

We kunnen de waarden van τ_{sys} en θ bepalen door dit stelsel op te lossen:

$$\begin{cases} \tau_{sys} = \frac{3}{2}(t_2 - t_1) \\ \theta = t_2 - \tau_{sys} \end{cases} \tag{4.6}$$

Om de versterking van het te regelen proces (K_{sys}) te bepalen, wordt de output van het proces gedeeld door de inputwaarde (= de output van de PID-regelaar).

Met behulp van K_{sys}, τ_{sys} en θ kunnen we een aantal nieuwe methoden bekijken. Want naast Ziegler en Nichols zijn vele anderen bezig geweest met het vinden van optimale PID-regelparameters. Een korte selectie van de meest gebruikte methoden staat in tabel 4.2. De IEEE database bevat maar liefst 762 papers met zowel 'PID' als 'tuning' in de titel.

Hierbij wordt Cohen-Coon vaak voor trage processen (lange dode tijd) gebruikt. De minimum-ITAE [6] zorgt ervoor dat de integraal van de absolute error vermenigvuldigd met de tijd geminimaliseerd wordt (ITAE staat voor: Integral of Time-weighted Absolute Error). Deze methode levert een nauwkeurige regelaar op. Beide methoden zijn echter, net zoals de Ziegler-Nichols-methode, gevoelig voor veranderingen in de procesparameters [5]. De AMIGO (Approximate M-constraint Integral Gain Optimization) -methode [2] is de meest recente van de hier gegeven

Tabel 4.2: PID-parameters volgens verschillende open-loop-methoden.

Methode / Parameters	K_c [%/?]	T_i [s]	T_d [s]
Ziegler-Nichols open-loop (PID)	$K_c = \frac{1,2 \cdot \tau_{sys}}{K_{sys} \cdot \theta}$	$T_i = 2,0 \cdot \theta$	$T_d = 0,5 \cdot \theta$
Ziegler-Nichols open-loop (PI)	$K_c = \frac{0,9 \cdot \tau_{sys}}{K_{sys} \cdot \theta}$	$T_i = 3,33 \cdot \theta$	–
Cohen-Coon (PID)	$K_c = \frac{\tau_{sys}}{K_{sys} \cdot \theta} \cdot \left(\frac{\theta}{4 \cdot \tau_{sys}} + \frac{4}{3} \right)$	$T_i = \theta \cdot \frac{32 \cdot \tau_{sys} + 6 \cdot \theta}{13 \cdot \tau_{sys} + 8 \cdot \theta}$	$T_d = \theta \cdot \frac{4 \cdot \tau_{sys}}{2 \cdot \theta + 11 \cdot \tau_{sys}}$
Cohen-Coon (PI)	$K_c = \frac{\tau_{sys}}{K_{sys} \cdot \theta} \cdot \left(\frac{\theta}{12 \cdot \tau_{sys}} + \frac{9}{10} \right)$	$T_i = \theta \cdot \frac{30 \cdot \tau_{sys} + 3 \cdot \theta}{9 \cdot \tau_{sys} + 20 \cdot \theta}$	–
minimum-ITAE (PID)	$K_c = \frac{1,357}{K_{sys}} \cdot \left(\frac{\theta}{\tau_{sys}} \right)^{-0,947}$	$T_i = \frac{\tau_{sys}}{0,842} \cdot \left(\frac{\theta}{\tau_{sys}} \right)^{0,738}$	$T_d = 0,381 \cdot \tau_{sys} \cdot \left(\frac{\theta}{\tau_{sys}} \right)^{0,995}$
minimum-ITAE (PI)	$K_c = \frac{0,859}{K_{sys}} \cdot \left(\frac{\theta}{\tau_{sys}} \right)^{-0,977}$	$T_i = \frac{\tau_{sys}}{0,674} \cdot \left(\frac{\theta}{\tau_{sys}} \right)^{0,680}$	–
AMIGO (PID)	$K_c = \frac{1}{K_{sys}} \cdot \left(0,2 + 0,45 \cdot \frac{\tau_{sys}}{\theta} \right)$	$T_i = \frac{0,4 \cdot \theta + 0,8 \cdot \tau_{sys}}{\theta + 0,1 \cdot \tau_{sys}} \cdot \theta$	$T_d = \frac{0,5 \cdot \theta \cdot \tau_{sys}}{0,3 \cdot \theta + \tau_{sys}}$
AMIGO (PID)	$K_c = \frac{0,35 \cdot \tau_{sys}}{K_{sys} \cdot \theta}$	$T_i = 13,4 \cdot \theta$	–

methoden en is ontwikkeld met het doel om robuste instellingen te vinden waarbij de PID-regelaar minder gevoelig is voor variaties in de procesparameters.

Het zal duidelijk zijn dat de open-loop-methoden alleen gebruikt kunnen worden als het proces zelfregelend is. Als het systeem van zichzelf instabiel is moet een andere methode worden gebruikt om de PID-parameters in te stellen, bijvoorbeeld een van de closed-loop-methoden die in paragraaf 4.6 worden behandeld.

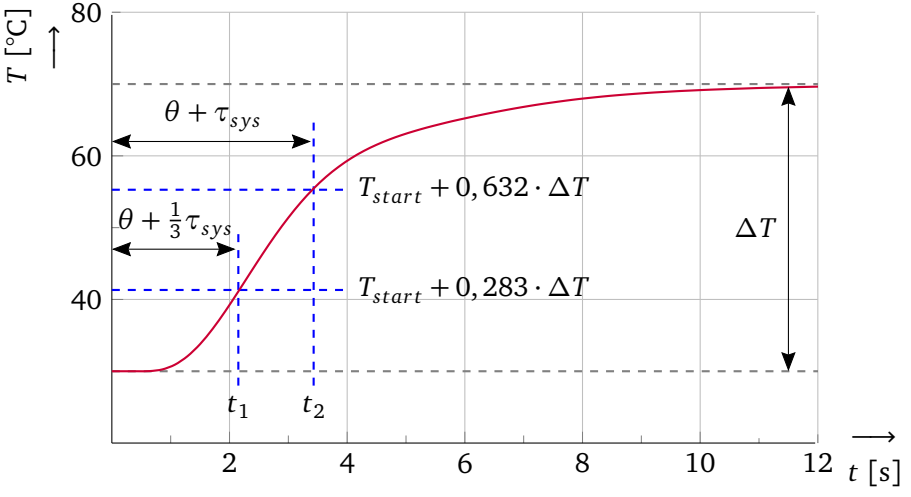
Opdracht 7: Bepalen van de parameters van de PID-regelaar volgens de minimum-ITAE-methode

a) Bepaal zelf de waarde van K_{sys}, τ_{sys} en θ uit de in figuur 4.2 gegeven stapresponsie met behulp van de in deze paragraaf beschreven methode.

b) Bepaal met behulp van deze waarden de parameters voor de PID-regelaar van dit proces volgens de minimum-ITAE-methode.

4.5 Antwoorden van opdracht 7

Uit figuur 4.4 lezen we af: $t_1 \approx 2,2$ s en $t_2 \approx 3,4$ s. Met behulp van vergelijking (4.6) bepalen we τ_{sys} en θ: $\tau_{sys} = \frac{3}{2} \cdot (3,4 - 2,2) = 1,8$ s en $\theta = 3,4 - 1,8 = 1,6$ s. De waarde van K_{sys} kan als volgt berekend worden $K_{sys} = \frac{\Delta T}{\Delta p} = \frac{40}{40} = 1$ °C/%.



Figuur 4.4: De stapresponsie van het te regelen proces. Op tijdstip t_1 heeft de output de waarde $T_{start} + 0,283 \cdot \Delta T$ bereikt en op tijdstip t_2 heeft de output de waarde $T_{start} + 0,632 \cdot \Delta T$ bereikt.

Met behulp van de bepaalde waarden voor K_{sys}, τ_{sys} en θ kunnen we met behulp van tabel 4.2 de parameterwaarden van de PID-regelaar vinden volgens de minimum-ITAE-methode:

$$K_c = \frac{1,357}{K_{sys}} \cdot \left(\frac{\theta}{\tau_{sys}}\right)^{-0,947} = \frac{1,357}{1} \cdot \left(\frac{1,6}{1,8}\right)^{-0,947} = 1,52\ \%/°C \tag{4.7}$$

$$T_i = \frac{\tau_{sys}}{0,842} \cdot \left(\frac{\theta}{\tau_{sys}}\right)^{0,738} = \frac{1,8}{0,842} \cdot \left(\frac{1,6}{1,8}\right)^{0,738} = 1,96\ \text{s} \tag{4.8}$$

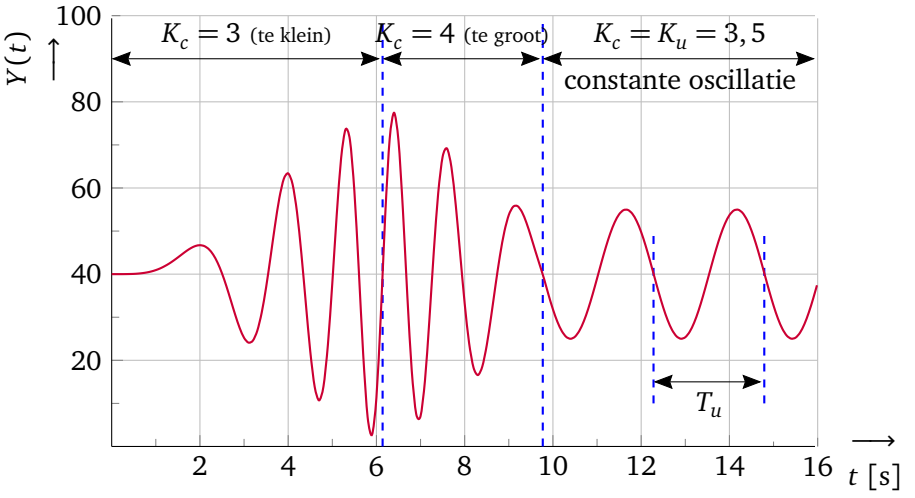
$$T_d = 0,381 \cdot \tau_{sys} \cdot \left(\frac{\theta}{\tau_{sys}}\right)^{0,995} = 0,381 \cdot 1,8 \cdot \left(\frac{1.6}{1,8}\right)^{0,995} = 0,61\ \text{s} \tag{4.9}$$

4.6 Closed loop response

Een andere methode om de PID-parameters te bepalen is de zogenoemde 'closed-loop'-responsie. Bij deze methode is het de bedoeling om de 'lus te sluiten' en de regelaar te laten regelen. De T_i-waarde wordt zo groot mogelijk gemaakt en de T_d-waarde zo klein mogelijk, waardoor ze effectief uitgeschakeld zijn en er uitsluitend met de K_c-waarde gewerkt wordt. De K_c-waarde van de regelaar wordt vervolgens zodanig opgevoerd, dat het systeem begint te oscilleren. Deze methode wordt daarom ook wel de "oscillatie methode" genoemd. In hoofdstuk 5 wordt deze methode nog verder gecombineerd/uitgebreid met systeemidentificatie.

De K_c-waarde waarbij het totale systeem begint te oscilleren, wordt K_u genoemd. De periodetijd van de oscillaties, T_u, is ook van belang bij deze methode. Uit figuur 4.5 kun je aflezen dat, in dit specifieke geval, de K_c-waarde waarbij het totale systeem begint te oscilleren, 3,5 is en dat de periodetijd T_u ongeveer 2,5 seconden is.

Een direct nadeel van deze methode is dat het in sommige installaties ongewenst is om het systeem te laten oscilleren, het kan zelfs gevaarlijk zijn. Wees je dus altijd bewust van het systeem waarmee je aan het werk bent!



Figuur 4.5: De responsie van het te regelen proces in closed loop bij verschillende waarden van K_c.

Ook hier zijn weer een aantal vuistregels te geven, hoe een regelaar ingesteld kan worden. Een korte selectie van de meest gebruikte methoden staat in tabel 4.3 [2, 7, 8, 9].

Voor de closed-loop AMIGO-methoden geldt $\kappa = \left(\frac{1}{K_{sys} \cdot K_u}\right)$.

De methode van Takahashi is hier met name van belang. Het geeft een optimale instelling voor de in hoofdstuk 3 afgeleide type C PID-regelaar, zie vergelijking (3.9). Een voordeel van deze methode is dat ook direct de sampletijd T_s hierin meegenomen is.

4.7 Industriële regelaars

In [5, blz. 32-19] worden 17 industriële controllers opgesomd die automatisch de PID-parameters kunnen bepalen. Tien daarvan gebruiken een open-loop-methode. Een andere methode die veel gebruikt wordt is de zogenoemde 'relay autotuner'.

Tabel 4.3: PID-parameters volgens verschillende closed-loop-methoden.

Methode / Parameters	K_c [%/?]	T_i [s]	T_d [s]
Ziegler-Nichols closed loop (PID)	$K_c = \frac{K_u}{1,7}$	$T_i = \frac{T_u}{2}$	$T_d = \frac{T_u}{8}$
Tyreus-Luyben	$K_c = \frac{K_u}{2,2}$	$T_i = 2,2 \cdot T_u$	$T_d = \frac{T_u}{6,3}$
Takahashi digitale regelaar	$K_c = 0,6 \cdot K_u \cdot \left(1 - \frac{T_s}{T_u}\right)$	$T_i = \frac{T_u - T_s}{2}$	$T_d = \frac{{T_u}^2}{8 \cdot T_u - T_s}$
AMIGO PID	$K_c = \left(0,3 - 0,1 \cdot \kappa^4\right) \cdot K_u$	$T_i = \frac{0,6}{1 + 2 \cdot \kappa} \cdot T_u$	$T_d = \frac{0,15 \cdot (1 - \kappa)}{1 - 0,95 \cdot \kappa} \cdot T_u$
AMIGO PI	$K_c = 0,16 \cdot K_u$	$T_i = \frac{1}{1 + 4,5 \cdot \kappa} \cdot T_u$	–

Bij deze methode wordt de PID-regelaar vervangen door een relay met hysteresis. Voor veel processen zorgt dit ervoor dat het outputsignaal gaat oscilleren met periodetijd T_u. De waarde van K_u kan door een amplitudemeting worden bepaald. Als de waarde van T_u en K_u bekend zijn, dan kunnen de PID-parameters met behulp van de formules behorende bij één van de closed-loop-methoden (zie tabel 4.3) bepaald worden.

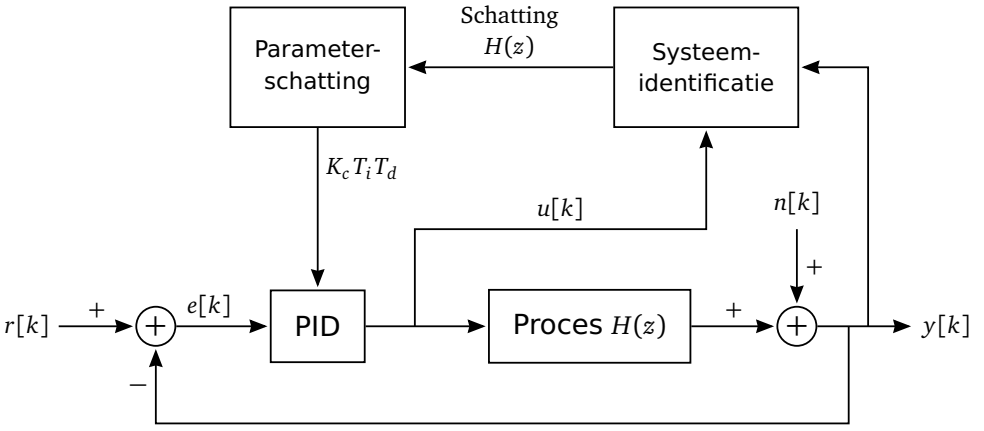
5

Systeemidentificatie

De in hoofdstuk 4 besproken instelmethodieken voor de parameters worden vaak verwerkt in zelfinstellende regelaars. Echter hebben we gezien dat dit alleen goed werkt voor systemen die zich ongeveer als FOPTD systemen gedragen, of die kunnen oscilleren. Soms is het helemaal niet gewenst dat een systeem oscilleert, of bevat de poolbaan van een systeem geen complexe component zodat het systeem niet *kan* oscilleren. Verder gingen we er vanuit dat na het instellen van de regelaar het niet nodig is de parameters bij te stellen. We impliceren hiermee dat het systeem door de tijd dus niet verandert, maar in de werkelijkheid kan dit wel degelijk het geval zijn.

5.1 Doel en achtergrond van systeemidentificatie

We gaan in dit hoofdstuk zien hoe we de eigenschappen van een onbekend systeem (de overdracht) kunnen schatten, om vanuit deze schatting de parameters in te stellen. Dit soort schattingen vergen veel rekenkracht. Daarom is het binnen de regeltechniek pas recentelijk mogelijk om regelaars zelf systeemidentificatie uit te laten voeren en zichzelf bij te laten stellen *tijdens* het regelen. Dit is schematisch weergegeven in figuur 5.1.



Figuur 5.1: Context van systeemidentificatie

We zien in figuur 5.1 het (onbekende) te regelen proces, de PID-regelaar, een systeemidentificatieblok en een parameterschattingsblok.

Systeemidentificatie werkt door te kijken naar de input van het proces $u(k)$ en de output van het proces $y(k)$. Aan de hand van deze inputs en outputs kan de overdracht van het systeem geschat worden. Als we de overdracht kennen, kunnen we deze gebruiken om met o.a. de methodieken van hoofdstuk 4 de optimale parameters van de PID-regelaar te vinden. Dit kan een continu bijsturend (adaptief) proces zijn (self-tuning) of een eenmalige actie (auto-tuning).

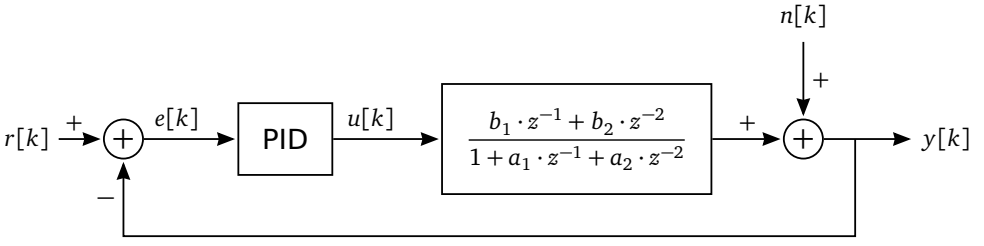
Hiermee komen we dan ook direct op het feitelijke doel van systeemidentificatie uit:

- Geef een schatting voor de parameters van het onbekende proces.
- Gebruik deze schatting om de parameters van de PID-regelaar optimaal in te stellen.

Met het bepalen van de parameters van een onbekend systeem komen we uit op een gebied binnen de signaalbewerking waarbij we veel met matrices en vectoren gaan rekenen.

5.2 Het procesmodel

Bij de bespreking van systeemidentificatie en in het bijzonder de Least Squares Method (die we verderop in dit hoofdstuk zullen uitwerken), maken we een schatting van de overdracht van het te regelen proces. We beperken ons tot een model van het proces waarbij de overdracht er uitziet als in figuur 5.2.



Figuur 5.2: Model voor systeemidentificatie

Als we de ruis buiten beschouwing laten ($n[k] = 0$) dan geldt in figuur 5.2:

$$\frac{y[z]}{u[z]} = \frac{b_1 \cdot z^{-1} + b_2 \cdot z^{-2}}{1 + a_1 \cdot z^{-1} + a_2 \cdot z^{-2}} \tag{5.1}$$

Kruislings vermenigvuldigen en y[z] isoleren geeft:

$$y[z] = -a_1 \cdot y[z] \cdot z^{-1} - a_2 \cdot y[z] \cdot z^{-2} + b_1 \cdot u[z] \cdot z^{-1} + b_2 \cdot u[z] \cdot z^{-2} \tag{5.2}$$

Als we dit terugtransformeren naar het tijddomein, dan vinden we de differentievergelijking van het proces ($y[k]$ als functie van $y[k-n]$ en $u[k-n]$):

$$y[k] = -a_1 \cdot y[k-1] - a_2 \cdot y[k-2] + b_1 \cdot u[k-1] + b_2 \cdot u[k-2] \tag{5.3}$$

Bij de systeemidentificatie maken we *geen* gebruik van het huidige sample $u[k]$. We schatten namelijk de nieuwe waarde voor $y[k]$ met behulp van de voorgaande waarden van y en u. We kennen dus bij het maken van de schatting de huidige

input $u[k]$ nog niet. Als we namelijk op tijdstip k zijn aangekomen dan weten we ook $y[k]$ en dan hoeven we deze dus niet meer te schatten.

We kunnen dit ook in vectornotatie opschrijven. We definiëren een vector $\mathbf{c}$, die de coëfficiënten a en b bevat:

$$\mathbf{c} = \begin{bmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \end{bmatrix} \tag{5.4}$$

Vanaf nu schrijven we bij het definiëren van een vector de vector als volgt op:

$$\mathbf{c}^{\mathrm{T}} = \begin{bmatrix} a_1 & a_2 & b_1 & b_2 \end{bmatrix} \tag{5.5}$$

De T staat voor 'geTransponeerd'. De vector is nu gedraaid weergegeven, dit doen we voor de leesbaarheid (het scheelt een hoop ruimte op de pagina).

We definiëren vervolgens een tijdafhankelijke vector $\mathbf{f}[k]$ die de voorgaande inputs $u[k]$ en outputs $y[k]$ bevat:

$$\mathbf{f}[k]^{\mathrm{T}} = \begin{bmatrix} -y[k-1] & -y[k-2] & u[k-1] & u[k-2] \end{bmatrix} \tag{5.6}$$

De vector $\mathbf{f}[k]^{\mathrm{T}}$ wordt ook wel de *datavector* genoemd.

Nu kunnen we de differentievergelijking een stuk gemakkelijker opschrijven als het inwendig product van deze twee vectoren:

$$y[k] = \mathbf{c}^{\mathrm{T}} \cdot \mathbf{f}[k] \tag{5.7}$$

Want:

$$y[k] = \mathbf{c}^{\mathrm{T}} \cdot \mathbf{f}[k] = \begin{bmatrix} a_1 & a_2 & b_1 & b_2 \end{bmatrix} \begin{bmatrix} -y[k-1] \\ -y[k-2] \\ u[k-1] \\ u[k-2] \end{bmatrix}$$

$$= -a_1 \cdot y[k-1] - a_2 \cdot y[k-2] + b_1 \cdot u[k-1] + b_2 \cdot u[k-2] \tag{5.8}$$

Stel dat we op *meerdere* tijdstippen willen kijken wat $y[k]$ is, bijvoorbeeld de afgelopen zes tijdstippen. We kunnen dan in plaats van een vector $\mathbf{f}[k]$ ook een matrix $\mathbf{F}$ bouwen als volgt:

$$\mathbf{F} = \begin{bmatrix} \mathbf{f}[k-5]^{\mathrm{T}} \\ \mathbf{f}[k-4]^{\mathrm{T}} \\ \mathbf{f}[k-3]^{\mathrm{T}} \\ \mathbf{f}[k-2]^{\mathrm{T}} \\ \mathbf{f}[k-1]^{\mathrm{T}} \\ \mathbf{f}[k]^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} -y[k-6] & -y[k-7] & u[k-6] & u[k-7] \\ -y[k-5] & -y[k-6] & u[k-5] & u[k-6] \\ -y[k-4] & -y[k-5] & u[k-4] & u[k-5] \\ -y[k-3] & -y[k-4] & u[k-3] & u[k-4] \\ -y[k-2] & -y[k-3] & u[k-2] & u[k-3] \\ -y[k-1] & -y[k-2] & u[k-1] & u[k-2] \end{bmatrix} \tag{5.9}$$

Als we de matrix $\mathbf{F}$ vermenigvuldigen met de vector $\mathbf{c}$ dan krijgen we de uitgang van het proces op in dit geval zes tijdstippen. Dit geven we aan met de vector $\mathbf{y}[k]$,

helemaal uitgeschreven:

$$\mathbf{y}[k] = \begin{bmatrix} y[k-5] \\ y[k-4] \\ y[k-3] \\ y[k-2] \\ y[k-1] \\ y[k] \end{bmatrix} = \begin{bmatrix} -y[k-6] & -y[k-7] & u[k-6] & u[k-7] \\ -y[k-5] & -y[k-6] & u[k-5] & u[k-6] \\ -y[k-4] & -y[k-5] & u[k-4] & u[k-5] \\ -y[k-3] & -y[k-4] & u[k-3] & u[k-4] \\ -y[k-2] & -y[k-3] & u[k-2] & u[k-3] \\ -y[k-1] & -y[k-2] & u[k-1] & u[k-2] \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ b_1 \\ b_2 \end{bmatrix} \tag{5.10}$$

Omdat we dit niet telkens helemaal op willen schrijven gebruiken we gewoon:

$$\mathbf{y}[k] = \mathbf{Fc} \tag{5.11}$$

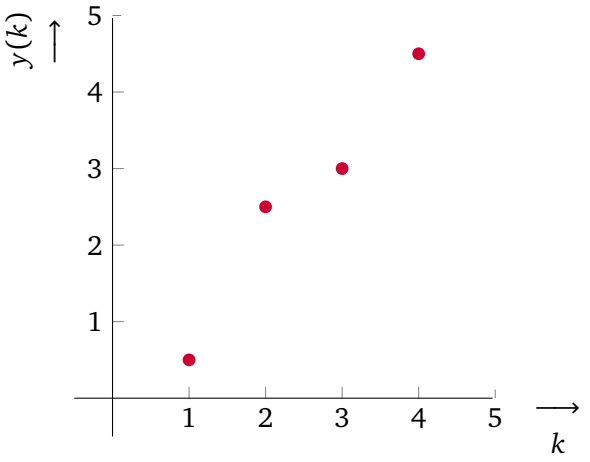
5.3 Least Squares Method

Voordat we overgaan tot het toepassen van de Least Squares Method met de hierboven opgestelde matrices en vectoren, beginnen we met een eenvoudiger voorbeeld.

Stel dat we een set met samples hebben $y[1] = 0,5; y[2] = 2,5; y[3] = 3$ en $y[4] = 4,5$. Deze datapunten zijn weergegeven in figuur 5.3

Omdat we vermoeden dat het hier een linear proces betreft willen we, zo goed mogelijk, een rechte lijn door deze datapunten trekken. In dit voorbeeld nemen we aan dat we zeker weten dat $y[0] = 0$ dus we kunnen de lijn modelleren als de functie:

$$\widehat{y}[k] = \widehat{a}k \tag{5.12}$$



Figuur 5.3: De datapunten.

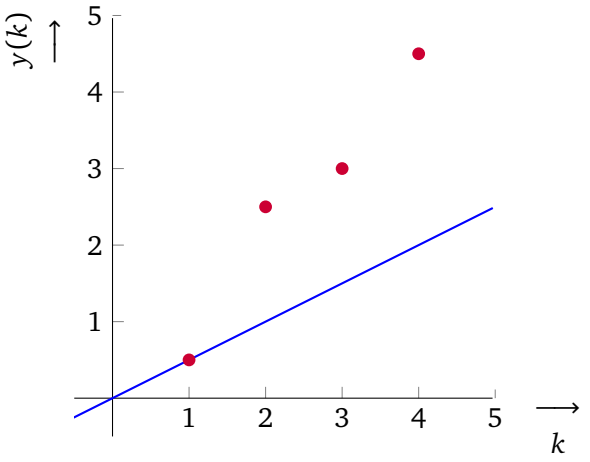
Met het dakje op de y ($\widehat{y}$) geven we aan dat dit een schatting is voor de werkelijke, onbekende functie y.

De constante $\widehat{a}$ (de helling van de lijn) is nu een onbekende constante, welke we willen schatten om het model compleet te maken. Als we dit op een naïeve manier doen, kijken we naar een willekeurig punt, bijvoorbeeld voor $k = 1$ geldt $y[1] = 0,5$. We kunnen dit nu invullen in de vergelijking (5.12) en heel eenvoudig $\widehat{a}$ oplossen:

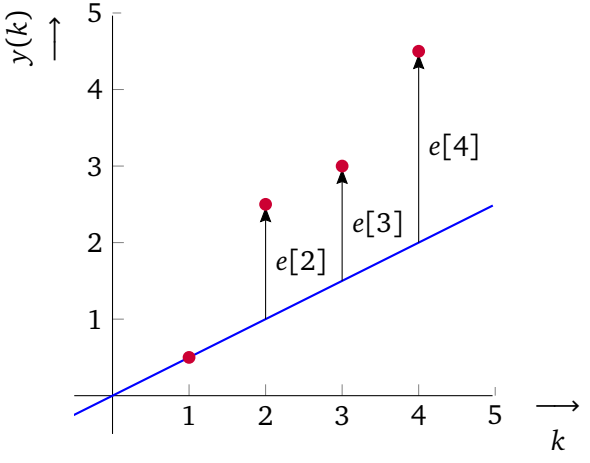
$$\begin{aligned} 0,5 &= \widehat{a} \cdot 1 \\ \widehat{a} &= 0,5 \end{aligned} \tag{5.13}$$

Deze lijn is weergegeven in figuur 5.4.

We kunnen nu bekijken wat de afwijking, of de error, is van de schatting (de blauwe lijn) ten opzichte van de werkelijkheid (de datapunten). In figuur 5.5 zijn de errors van de schattingen weergegeven met behulp van een zwarte pijl.



Figuur 5.4: Met 1 datapunt geschatte lijn.



Figuur 5.5: De errors van de schatting.

De werkelijke waarde van het datapunt wordt gegeven door de schatting op dat punt plus de error op dat punt, oftewel:

$$y[k] = \hat{y}[k] + e[k] \tag{5.14}$$

Dus de error wordt gegeven door het verschil van de werkelijke waarde met de waarde van de schatting:

$$e[k] = y[k] - \widehat{y}[k] \tag{5.15}$$

Bij het maken van een schatting willen we de errors zo klein mogelijk te maken (*minimaliseren*).

De essentie van de kleinste kwadraatmethode is nu als volgt. We nemen het *kwadraat* van elke fout en tellen deze kwadratische fouten bij elkaar op. Door het kwadraat te nemen worden ten eerste alle fouten positief, zodat fouten elkaar niet kunnen opheffen als we de fouten bij elkaar optellen. Ten tweede tellen zo grotere fouten zwaarder mee dan kleinere fouten.

De *totale kwadratische fout* in onze bovenstaande schatting, vergeleken met alle datapunten die we hebben, noemen we E. Deze wordt gegeven door:

$$E = \sum_k e[k]^2 = \sum_k \big(y[k] - \widehat{y}[k]\big)^2 \tag{5.16}$$

We laten in deze somreeks k alle samples die we hebben doorlopen.

In ons voorbeeld met $\widehat{y}[k] = \widehat{a}k$, $\widehat{a} = 0,5$ en $k = 1 \ldots 4$ geldt:

$$E = \sum_{k=1}^{4} \big(y[k] - 0,5k\big)^2 \tag{5.17}$$

Door alle datapunten uit ons voorbeeld in te vullen kunnen we E als volgt berekenen:

$$\begin{aligned} E &= \big(y[1]-0,5\cdot 1\big)^2+\big(y[2]-0,5\cdot 2\big)^2+\big(y[3]-0,5\cdot 3\big)^2+\big(y[4]-0,5\cdot 4\big)^2 \\ &= (0,5-0,5\cdot 1)^2+(2,5-0,5\cdot 2)^2+(3-0,5\cdot 3)^2+(4,5-0,5\cdot 4)^2 \\ &= 0^2+1,5^2+1,5^2+2,5^2 \\ &= 0+2,25+2,25+6,25 \\ &= 10,75 \end{aligned} \tag{5.18}$$

Stel dat we E zien als een functie van de schatting $\widehat{a}$. We kunnen nu een $\widehat{a}$ vinden waarvoor de E minimaal is. We doen dit door de *afgeleide* te nemen van E naar $\widehat{a}$ en deze afgeleide gelijk te stellen aan 0. We bepalen zo het *minimum* van de E; precies wat we willen, dit maakt immers de schatting het best!

Wiskundig noteren we dit als:

$$\frac{dE}{d\widehat{a}} = \frac{d\Big(\sum_k \big(y[k]-\widehat{y}[k]\big)^2\Big)}{d\widehat{a}} = 0 \tag{5.19}$$

In ons voorbeeld geeft dit:

$$\frac{dE}{d\widehat{a}} = \frac{d\Big(\sum_{k=1}^{4} \big(y[k]-\widehat{a}\cdot k\big)^2\Big)}{d\widehat{a}} = 0 \tag{5.20}$$

Als we het kwadraat uitschrijven vinden we:

$$\frac{dE}{d\widehat{a}} = \frac{d\left(\sum_{k=1}^{4}\left(y[k]^2 - 2\cdot y[k]\cdot\widehat{a}\cdot k + \widehat{a}^2\cdot k^2\right)\right)}{d\widehat{a}} = 0 \tag{5.21}$$

We kunnen de som nu term voor term differentiëren naar $\widehat{a}$:

$$\frac{dE}{d\widehat{a}} = \sum_{k=1}^{4} -2\cdot y[k]\cdot k + 2\cdot\widehat{a}\cdot k^2 = 0 \tag{5.22}$$

Als we de waarden uit ons voorbeeld invullen krijgen we:

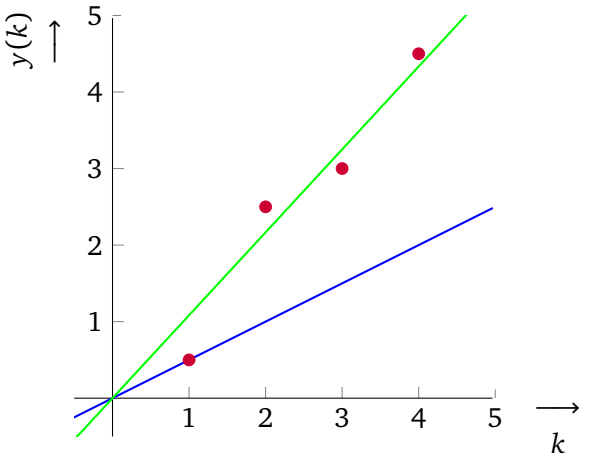
$$\begin{aligned}
\frac{dE}{d\widehat{a}} &= -2\cdot y[1]\cdot 1 + 2\cdot\widehat{a}\cdot 1^2 + -2\cdot y[2]\cdot 2 + 2\cdot\widehat{a}\cdot 2^2 + \\
&\qquad -2\cdot y[3]\cdot 3 + 2\cdot\widehat{a}\cdot 3^2 + -2\cdot y[4]\cdot 4 + 2\cdot\widehat{a}\cdot 4^2 = 0 \\
&= -2\cdot 0{,}5\cdot 1 + 2\cdot\widehat{a}\cdot 1^2 + -2\cdot 2{,}5\cdot 2 + 2\cdot\widehat{a}\cdot 2^2 + \\
&\qquad -2\cdot 3\cdot 3 + 2\cdot\widehat{a}\cdot 3^2 + -2\cdot 4{,}5\cdot 4 + 2\cdot\widehat{a}\cdot 4^2 = 0 \\
&= -1 + 2\cdot\widehat{a} + -10 + 8\cdot\widehat{a} + -18 + 18\cdot\widehat{a} + -36 + 32\cdot\widehat{a} = 0 \\
&= -65 + 60\cdot\widehat{a} = 0
\end{aligned} \tag{5.23}$$

Nu vinden we voor $\widehat{a}$:

$$\widehat{a} = \frac{65}{60} = 1{,}083333 \tag{5.24}$$

In figuur 5.6 is deze nieuwe schatting voor $\widehat{a}$ weergegeven.

We zien hier dat de nieuwe schatting (groene lijn) een stuk beter is dan de oude schatting (blauwe lijn). De totale error van de oude schatting was $E = 10{,}75$ voor



Figuur 5.6: Met 1 datapunt geschatte lijn (blauw) en met LMS geschatte lijn (groen).

$\widehat{a} = 0,5$. Met de nieuwe schatting $\widehat{a} = 1,083333$, kunnen we op dezelfde manier E uitrekenen. Dit geeft een totale kwadratische error van slechts $E = 0,541667$.

De Least Squares Method geeft een *minimale kwadratische fout*.

5.4 Least Squares Method met vectoren en matrices

In de vorige paragraaf hebben we de Least Squares Method gebruikt en telkens alles helemaal uitgeschreven. We kunnen dit ook korter opschrijven met vectoren. Stel dat we de volgende vectoren definiëren, behorend bij het voorbeeld in paragraaf 5.3.

De k-waarden van de samples in een vector:

$$\mathbf{k}^{\mathrm{T}} = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix} \tag{5.25}$$

De y-waarden van de samples in een vector:

$$\mathbf{y}^{\mathrm{T}} = \begin{bmatrix} y[1] & y[2] & y[3] & y[4] \end{bmatrix} \tag{5.26}$$

De schatting:

$$\widehat{\mathbf{y}} = \widehat{a}\mathbf{k} \tag{5.27}$$

De errors per punt als vector wordt nu gegeven door:

$$\mathbf{e} = \mathbf{y} - \widehat{\mathbf{y}} \tag{5.28}$$

De *totale kwadratische fout*, E, wordt gegeven door:

$$E = \mathbf{e}^{\mathrm{T}}\mathbf{e} \tag{5.29}$$

Opdracht 8: Totale fout met vectoren

Ga voor jezelf na dat vergelijking (5.29) gelijk is aan vergelijking (5.16).

Als we nu de afgeleide van de totale kwadratische fout willen bepalen, schrijven we de kwadratische fout eerst wat verder uit:

$$E = \mathbf{e}^{\mathrm{T}}\mathbf{e} = (\mathbf{y} - \widehat{\mathbf{y}})^{\mathrm{T}}(\mathbf{y} - \widehat{\mathbf{y}}) = (\mathbf{y} - \widehat{a}\mathbf{k})^{\mathrm{T}}(\mathbf{y} - \widehat{a}\mathbf{k}) = \left(\mathbf{y}^{\mathrm{T}} - (\widehat{a}\mathbf{k})^{\mathrm{T}}\right)\left(\mathbf{y} - \widehat{a}\mathbf{k}\right) \tag{5.30}$$

Uitvermenigvuldigen geeft:

$$E = \mathbf{y}^{\mathrm{T}}\mathbf{y} - \mathbf{y}^{\mathrm{T}}\widehat{a}\mathbf{k} - (\widehat{a}\mathbf{k})^{\mathrm{T}}\mathbf{y} + (\widehat{a}\mathbf{k})^{\mathrm{T}}\widehat{a}\mathbf{k} \tag{5.31}$$

Omdat geldt $\mathbf{y}^{\mathrm{T}}\widehat{a}\mathbf{k} = (\widehat{a}\mathbf{k})^{\mathrm{T}}\mathbf{y}$ kunnen we dit vereenvoudigen tot:

$$E = \mathbf{y}^{\mathrm{T}}\mathbf{y} - 2(\widehat{a}\mathbf{k})^{\mathrm{T}}\mathbf{y} + (\widehat{a}\mathbf{k})^{\mathrm{T}}\widehat{a}\mathbf{k} \tag{5.32}$$

We kunnen dit nu weer differentiëren naar $\hat{a}$ en daarna gelijkstellen aan 0 om de minimale waarde van E te vinden:

$$\frac{dE}{d\hat{a}} = \frac{d\left(\mathbf{y}^{\mathrm{T}}\mathbf{y} - 2(\hat{a}\mathbf{k})^{\mathrm{T}}\mathbf{y} + (\hat{a}\mathbf{k})^{\mathrm{T}}\hat{a}\mathbf{k}\right)}{d\hat{a}} \tag{5.33}$$

We kunnen nu term voor term differentiëren:

$$\frac{d\left(\mathbf{y}^{\mathrm{T}}\mathbf{y}\right)}{d\hat{a}} = 0 \qquad \frac{d\left(-2(\hat{a}\mathbf{k})^{\mathrm{T}}\mathbf{y}\right)}{d\hat{a}} = -2\mathbf{k}^{\mathrm{T}}\mathbf{y} \qquad \frac{d\left((\hat{a}\mathbf{k})^{\mathrm{T}}\hat{a}\mathbf{k}\right)}{d\hat{a}} = \mathbf{k}^{\mathrm{T}}\hat{a}\mathbf{k} + (\hat{a}\mathbf{k})^{\mathrm{T}}\mathbf{k} \tag{5.34}$$

Bij het differentiëren van de derde term hebben we de productregel toegepast.

Dus de totale afgeleide is:

$$\frac{dE}{d\hat{a}} = -2\mathbf{k}^{\mathrm{T}}\mathbf{y} + \mathbf{k}^{\mathrm{T}}\hat{a}\mathbf{k} + (\hat{a}\mathbf{k})^{\mathrm{T}}\mathbf{k} \tag{5.35}$$

Omdat geldt $(\hat{a}\mathbf{k})^{\mathrm{T}}\mathbf{k} = \mathbf{k}^{\mathrm{T}}\hat{a}\mathbf{k}$ kunnen we dit vereenvoudigen tot:

$$\frac{dE}{d\hat{a}} = -2\mathbf{k}^{\mathrm{T}}\mathbf{y} + 2\mathbf{k}^{\mathrm{T}}\hat{a}\mathbf{k} \tag{5.36}$$

Als we dit gelijkstellen aan nul, dan vinden we de formule voor de schatting $\hat{a}$ met vectoren waarbij E minimaal is:

$$-2\mathbf{k}^{\mathrm{T}}\mathbf{y} + 2\mathbf{k}^{\mathrm{T}}\hat{a}\mathbf{k} = 0 \tag{5.37}$$

$$\hat{a} = \frac{\mathbf{k}^{\mathrm{T}}\mathbf{y}}{\mathbf{k}^{\mathrm{T}}\mathbf{k}} \tag{5.38}$$

Opdracht 9: Bereken $\widehat{a}$ met vectoren

Ga voor jezelf na dat het invullen van $y = \begin{bmatrix} 0,5 & 2,5 & 3 & 4,5 \end{bmatrix}^{\mathrm{T}}$ en $k = \begin{bmatrix} 1 & 2 & 3 & 4 \end{bmatrix}^{\mathrm{T}}$ in vergelijking (5.37) hetzelfde resultaat geeft als vergelijking (5.24).

De Least Squares Method kunnen we ook toepassen als we een ingewikkelder model willen gebruiken voor de schatting van het systeem, met meer coëfficiënten (bijvoorbeeld als we denken dat een goed model voor het systeem is: $\widehat{y} = \widehat{a_1} \cdot k^2 + \widehat{a_2} \cdot k + \widehat{a_3}$).

We kunnen dit doen door van de te schatten coëfficiënten ook een vector $\widehat{\mathbf{a}}$ te maken, bijvoorbeeld $\widehat{\mathbf{a}}^{\mathrm{T}} = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}$. We kunnen nu gelijk overgaan tot de Least Squares Method binnen de context van DCS01, omdat dit een identiek probleem betreft.

5.5 De LSM voor het schatten van de overdracht van een onbekend systeem

We hebben in paragraaf 5.2 gezien dat we de uitgang van het systeem op meerdere tijdstippen kunnen opschrijven als:

$$\mathbf{y}[k] = \mathbf{Fc} \tag{5.11}$$

Waarbij $\mathbf{F}$ de matrix met samples van voorgaande outputs en inputs is, en $\mathbf{c}$ de vector met de coëfficiënten van de overdracht. We schrijven in het vervolg, voor het gemak, gewoon $\mathbf{y}$ in plaats van $\mathbf{y}[k]$).

We kunnen nu ook een schatting voor $\mathbf{y}$ maken. Deze noemen we $\widehat{\mathbf{y}}$ en wordt gegeven door:

$$\widehat{\mathbf{y}} = \mathbf{F}\widehat{\mathbf{c}} \tag{5.39}$$

Dit kun je lezen als: de schatting voor een aantal uitgangen van het systeem krijg je door de matrix met samples van de voorgaande outputs en inputs te vermenigvuldigen met een vector met geschatte coëfficiënten.

De errors per punt als vector wordt, zoals we al eerder hebben gezien, gegeven door:

$$\mathbf{e} = \mathbf{y} - \widehat{\mathbf{y}} \tag{5.28}$$

Als we in deze vergelijking, vergelijking (5.39) invullen, vinden we:

$$\mathbf{e} = \mathbf{y} - \mathbf{F}\widehat{\mathbf{c}} \tag{5.40}$$

De *totale kwadratische fout* wordt, zoals we al gezien hebben, gegeven door:

$$E = \mathbf{e}^{\mathrm{T}}\mathbf{e} \tag{5.29}$$

Als we in deze vergelijking, vergelijking (5.40) invullen, vinden we:

$$E = (\mathbf{y} - \mathbf{F}\widehat{\mathbf{c}})^{\mathrm{T}}(\mathbf{y} - \mathbf{F}\widehat{\mathbf{c}}) \tag{5.41}$$

Transpose binnen de haakjes halen geeft:

$$E = \left(\mathbf{y}^{\mathrm{T}} - (\mathbf{F}\widehat{\mathbf{c}})^{\mathrm{T}}\right)\left(\mathbf{y} - \mathbf{F}\widehat{\mathbf{c}}\right) \tag{5.42}$$

Uitvermenigvuldigen geeft:

$$E = \mathbf{y}^{\mathrm{T}}\mathbf{y} - \mathbf{y}^{\mathrm{T}}\mathbf{F}\widehat{\mathbf{c}} - (\mathbf{F}\widehat{\mathbf{c}})^{\mathrm{T}}\mathbf{y} + (\mathbf{F}\widehat{\mathbf{c}})^{\mathrm{T}}\mathbf{F}\widehat{\mathbf{c}} \tag{5.43}$$

Omdat geldt $\mathbf{y}^{\mathrm{T}}\mathbf{F}\widehat{\mathbf{c}} = (\mathbf{F}\widehat{\mathbf{c}})^{\mathrm{T}}\mathbf{y}$ kunnen we dit vereenvoudigen tot:

$$E = \mathbf{y}^{\mathrm{T}}\mathbf{y} - 2(\mathbf{F}\widehat{\mathbf{c}})^{\mathrm{T}}\mathbf{y} + (\mathbf{F}\widehat{\mathbf{c}})^{\mathrm{T}}\mathbf{F}\widehat{\mathbf{c}} \tag{5.44}$$

We kunnen dit nu weer differentiëren naar $\widehat{\mathbf{c}}$ en gelijkstellen aan 0 om de minimale waarde van E te vinden. Omdat we differentiëren naar een vector moeten we de partiële afgeleide bepalen:

$$\frac{\partial E}{\partial \widehat{\mathbf{c}}} = \frac{\partial \left(\mathbf{y}^{\mathrm{T}}\mathbf{y} - 2(\mathbf{F}\widehat{\mathbf{c}})^{\mathrm{T}}\mathbf{y} + (\mathbf{F}\widehat{\mathbf{c}})^{\mathrm{T}}\mathbf{F}\widehat{\mathbf{c}}\right)}{\partial \widehat{\mathbf{c}}} \tag{5.45}$$

We kunnen nu term voor term partieel differentiëren:

$$\frac{\partial \left(\mathbf{y}^{\mathrm{T}}\mathbf{y}\right)}{\partial \widehat{\mathbf{c}}} = 0 \qquad \frac{\partial \left(-2(\mathbf{F}\widehat{\mathbf{c}})^{\mathrm{T}}\mathbf{y}\right)}{\partial \widehat{\mathbf{c}}} = -2\mathbf{F}^{\mathrm{T}}\mathbf{y} \qquad \frac{\partial \left((\mathbf{F}\widehat{\mathbf{c}})^{\mathrm{T}}\mathbf{F}\widehat{\mathbf{c}}\right)}{\partial \widehat{\mathbf{c}}} = \mathbf{F}^{\mathrm{T}}\mathbf{F}\widehat{\mathbf{c}} + (\mathbf{F}\widehat{\mathbf{c}})^{\mathrm{T}}\mathbf{F} \tag{5.46}$$

Bij het differentiëren van de derde term hebben we de productregel toegepast.

Dus de totale partiële afgeleide is:

$$\frac{\partial E}{\partial \widehat{\mathbf{c}}} = -2\mathbf{F}^{\mathrm{T}}\mathbf{y} + \mathbf{F}^{\mathrm{T}}\mathbf{F}\widehat{\mathbf{c}} + (\mathbf{F}\widehat{\mathbf{c}})^{\mathrm{T}}\mathbf{F} \tag{5.47}$$

Omdat geldt $(\mathbf{F}\widehat{\mathbf{c}})^{\mathrm{T}}\mathbf{F} = \mathbf{F}^{\mathrm{T}}\mathbf{F}\widehat{\mathbf{c}}$ kunnen we dit vereenvoudigen tot:

$$\frac{\partial E}{\partial \widehat{\mathbf{c}}} = -2\mathbf{F}^{\mathrm{T}}\mathbf{y} + 2\mathbf{F}^{\mathrm{T}}\mathbf{F}\widehat{\mathbf{c}} \tag{5.48}$$

Als we dit gelijkstellen aan nul, dan vinden we de formule voor de schatting $\widehat{\mathbf{c}}$ waarbij E minimaal is:

$$-2\mathbf{F}^{\mathrm{T}}\mathbf{y}+2\mathbf{F}^{\mathrm{T}}\mathbf{F}\widehat{\mathbf{c}}=0 \tag{5.49}$$

$$\mathbf{F}^{\mathrm{T}}\mathbf{F}\widehat{\mathbf{c}}=\mathbf{F}^{\mathrm{T}}\mathbf{y} \tag{5.50}$$

$$(\mathbf{F}^{\mathrm{T}}\mathbf{F})^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{F}\widehat{\mathbf{c}}=(\mathbf{F}^{\mathrm{T}}\mathbf{F})^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{y} \tag{5.51}$$

$$\widehat{\mathbf{c}}=(\mathbf{F}^{\mathrm{T}}\mathbf{F})^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{y} \tag{5.52}$$

Vergelijking (5.52) geeft dus de *geschatte coëfficiënten* voor het proces dat gegeven is in figuur 5.2 waarvoor de *totale kwadratische fout* het kleinst is

Opdracht 10: Foute afleiding

Waarom is het *niet* mogelijk om vergelijking (5.50) als volgt te vereenvoudigen:

$$\mathbf{F}^{\mathrm{T}}\mathbf{F}\widehat{\mathbf{c}}=\mathbf{F}^{\mathrm{T}}\mathbf{y} \tag{5.50}$$

$$(\mathbf{F}^{\mathrm{T}})^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{F}\widehat{\mathbf{c}}=(\mathbf{F}^{\mathrm{T}})^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{y} \tag{5.53}$$

$$\mathbf{F}\widehat{\mathbf{c}}=\mathbf{y} \tag{5.54}$$

$$\mathbf{F}^{-1}\mathbf{F}\widehat{\mathbf{c}}=\mathbf{F}^{-1}\mathbf{y} \tag{5.55}$$

$$\widehat{\mathbf{c}}=\mathbf{F}^{-1}\mathbf{y} \tag{5.56}$$

Tip: bedenk dat niet elke matrix geïnverteerd kan worden.

5.6 Voorbeeld van LSM in MATLAB

Stel dat we een bepaald systeem hebben waarvan we de overdracht willen schatten, bijvoorbeeld:

```
>> Hz = tf([0 1 0.5 -1],[1 1 0.2 0.3],1)

Hz =
       z^2 + 0.5 z - 1
  -----------------------
  z^3 + z^2 + 0.2 z + 0.3
```

Stel dat we dit proces een willekeurige (random) input geven van 10 samples:

```
>> N = 10

N =
     10

>> u = rand(N,1)

u =
     0.1576
     0.9706
     0.9572
     0.4854
     0.8003
     0.1419
     0.4218
     0.9157
     0.7922
     0.9595
```

We kunnen nu meten wat de uitgang van het systeem is (`lsim` haalt een bepaalde input door een systeem en berekent de output):

```
>> y = lsim(Hz,u)

y =
```

```
         0
    0.1576
    0.8918
    0.3615
   -0.5938
    0.3398
   -0.2728
    0.0754
    0.8619
    0.0331
```

We gaan nu met de besproken methode proberen de overdracht van het systeem te schatten: we doen alsof we de oorspronkelijke overdracht niet meer weten (zoals in het echt).

Stel dat we aannemen dat we een systeem met twee coëfficiënten in de teller en noemer hebben:

$$H(z) = \frac{\widehat{b}_1 z^{-1} + \widehat{b}_2 z^{-2}}{1 + \widehat{a}_1 z^{-1} + \widehat{a}_2 z^{-2}} \tag{5.57}$$

Onze schatting voor de coëfficiënten is dan:

$$\widehat{\mathbf{c}}^{\mathrm{T}} = \begin{bmatrix} \widehat{a}_1 & \widehat{a}_2 & \widehat{b}_1 & \widehat{b}_2 \end{bmatrix} \tag{5.58}$$

Deze kunnen berekend worden met behulp van vergelijking (5.52).

$$\widehat{\mathbf{c}} = (\mathbf{F}^{\mathrm{T}}\mathbf{F})^{-1}\mathbf{F}^{\mathrm{T}}\mathbf{y} \tag{5.52}$$

We kunnen de benodigde matrix $\mathbf{F}$, zie vergelijking (5.9) als volgt bouwen in MATLAB:

```
>> F = [ -y(2:N-1) -y(1:N-2) u(2:N-1) u(1:N-2) ]

F =
   -0.1576         0    0.9706    0.1576
```

```
   -0.8918   -0.1576    0.9572    0.9706
   -0.3615   -0.8918    0.4854    0.9572
    0.5938   -0.3615    0.8003    0.4854
   -0.3398    0.5938    0.1419    0.8003
    0.2728   -0.3398    0.4218    0.1419
   -0.0754    0.2728    0.9157    0.4218
   -0.8619   -0.0754    0.7922    0.9157
```

We kunnen nu simpelweg vergelijking (5.52) invoeren in MATLAB om de geschatte coëfficiënten te vinden:

```
>> c = ((F'*F)^-1)*(F'*y(3:N))

c =
   -0.0526
    0.4258
    1.1084
   -0.8163
```

Als je dit nadoet, vind je misschien iets andere waarden, omdat we een willekeurige input hebben gebruikt.

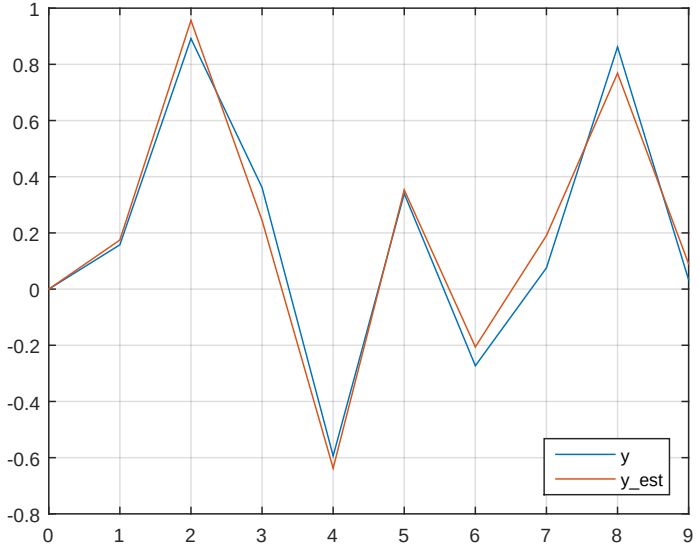
We kunnen nu van de schatting (estimation) een systeem maken:

```
Hz_est = tf([c(3) c(4)],[1 c(1) c(2)],1)

Hz_est =
       1.108 z - 0.8163
  ------------------------
  z^2 - 0.05262 z + 0.4258
```

Hiermee kunnen we berekenen en plotten wat de geschatte uitvoer telkens is geweest. In figuur 5.7 is de werkelijke waarde van $y[k]$ en de geschatte waarde $\widehat{y}[k]$ weergegeven voor $k = 0..9$.

```
>> y_est = lsim(Hz_est,u)

y_est =
         0
    0.1747
```

```
    0.9563
    0.2446
   -0.6377
    0.3531
   -0.2058
    0.1905
    0.7684
    0.0899

>> k=0:9

k =
     0     1     2     3     4     5     6     7     8     9

>> plot(k,y,k,y_est)
>> legend('y', 'y\_est', 'Location', 'southeast')
>> grid on
```



Figuur 5.7: De werkelijke waarde van de output $y[k]$ en de geschatte waarde van de output $\widehat{y}[k]$ voor $k = 0..9$.

We zien hier dat de geschatte overdracht al een aardig goede schatting voor de output geeft. Dit is de benadering waarbij het totaal van het kwadraat van de errors minimaal is (gegarandeerd door de LSM) dat een systeem met twee coëfficiënten in de teller en in de noemer kan genereren aan de hand van de gegeven inputs.

Echter als we gaan kijken naar de coëfficiënten als we het systeem modelleren met drie coëfficiënten in de teller en in de noemer, dan bouwen we **F** als volgt:

```
>> F = [ -y(3:N-1) -y(2:N-2) -y(1:N-3) u(3:N-1) u(2:N-2) ↩
 ↪ u(1:N-3)]

F =
   -0.8918   -0.1576         0    0.9572    0.9706    0.1576
   -0.3615   -0.8918   -0.1576    0.4854    0.9572    0.9706
    0.5938   -0.3615   -0.8918    0.8003    0.4854    0.9572
   -0.3398    0.5938   -0.3615    0.1419    0.8003    0.4854
    0.2728   -0.3398    0.5938    0.4218    0.1419    0.8003
   -0.0754    0.2728   -0.3398    0.9157    0.4218    0.1419
   -0.8619   -0.0754    0.2728    0.7922    0.9157    0.4218
```

Als we nu de geschatte coëfficiënten berekenen vinden we:

```
>> c = ((F'*F)^-1)*(F'*y(4:N))

c =
    1.0000
    0.2000
    0.3000
    1.0000
    0.5000
   -1.0000
```

Dan zien we dat deze exact overeen komen met die van het echte systeem! De schatting is dus 100% correct als we precies hetzelfde aantal coëfficiënten en voldoende meetwaarden hebben!

5.7 Voorbeeld met ruis

In de fysieke wereld hebben we altijd en overal last van ruis in onze metingen. Ruis kan afkomstig zijn van verschillende zaken; van slechte bekabeling met overspraak tot kwantisatieruis van een ADC tot zelfs de kosmische achtergrondstraling van de oerknal.

In ons bovenstaande voorbeeld kunnen we eenvoudig ruis simuleren in MATLAB.

```
>> ynoise = y + rand(10,1).*0.4 - 0.2

ynoise =
    0.0623
   -0.0281
    1.0314
    0.5351
   -0.5223
    0.4429
   -0.1756
    0.0323
    0.9241
   -0.0984
```

We tellen zo aardig wat ruis (tussen -0,2 en 0,2) op bij onze samples. Laten we de schatting nogmaals maken voor drie coëfficiënten in de teller en noemer. Eerst stellen we **F** weer opnieuw op, maar nu met de samples die ruis bevatten.

```
>> F = [-ynoise(3:N-1) -ynoise(2:N-2) -ynoise(1:N-3) ↩
 ↪ u(3:N-1) u(2:N-2) u(1:N-3)]

F =
   -1.0314    0.0281   -0.0623    0.9572    0.9706    0.1576
   -0.5351   -1.0314    0.0281    0.4854    0.9572    0.9706
    0.5223   -0.5351   -1.0314    0.8003    0.4854    0.9572
   -0.4429    0.5223   -0.5351    0.1419    0.8003    0.4854
```

```
    0.1756   -0.4429    0.5223    0.4218    0.1419    0.8003
   -0.0323    0.1756   -0.4429    0.9157    0.4218    0.1419
   -0.9241   -0.0323    0.1756    0.7922    0.9157    0.4218
```

Als we nu de geschatte coëfficiënten berekenen vinden we:

```
>> c = ((F'*F)^-1)*(F'* ynoise(4:N))

c =
    1.0853
    0.2073
    0.2232
    0.9524
    0.7627
   -0.9488
```

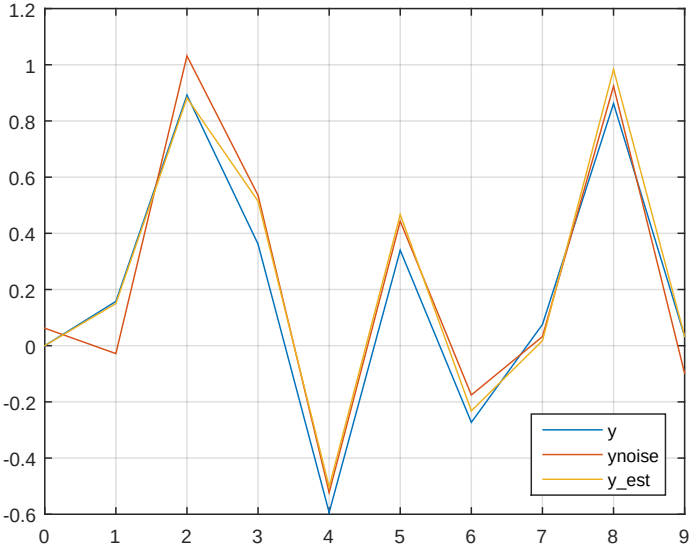
We zien nu dat de coëfficiënten beginnen af te wijken van de werkelijke waarden. Dit is allemaal het gevolg van de ruis, die de schatting onnauwkeuriger maakt.

In figuur 5.7 is de werkelijke waarde van y, de waarde van ynoise en de geschatte waarde y_est weergegeven voor $k = 0..9$.

```
>> Hz_est = tf([c(4) c(5) c(6)],[1 c(1) c(2) c(3)],1)

Hz_est =
     0.9524 z^2 + 0.7627 z - 0.9488
  -----------------------------------
  z^3 + 1.085 z^2 + 0.2073 z + 0.2232

>> y_est = lsim(Hz_est,u)

y_est =
         0
    0.1501
    0.8817
    0.5144
   -0.5031
```

```
    0.4668
   -0.2321
    0.0180
    0.9836
    0.0334
```

```
>> plot(k,y,k,ynoise,k,y_est)
>> legend('y', 'ynoise', 'y\_est', 'Location', 'southeast')
>> grid on
```



Figuur 5.8: De werkelijke waarde van y, de waarde van ynoise en de geschatte waarde y_est weergegeven voor $k = 0..9$.

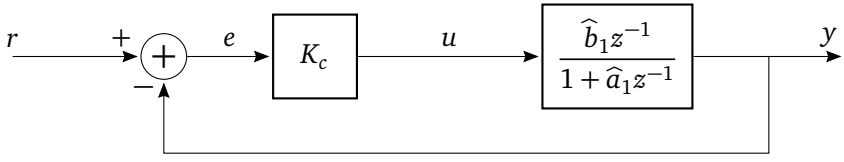
We zien hier dat de output voor het systeem met de geschatte coëfficiënten iets afwijkt van de werkelijkheid, maar de schatting is zelfs bij zo'n grote hoeveelheid ruis nog best goed.

5.8 Oscillatiemethode bij een 1e orde systeem

In de voorgaande paragrafen hebben we gezien hoe het mogelijk is om, uitgaande van gemeten waarden van y en u, direct een schatting te geven voor de coëfficiënten a en b (in de vector c) van het te regelen proces. Deze geschatte parameters gaan we nu gebruiken bij de oscillatiemethode (closed loop response), zoals beschreven in paragraaf 4.6.

Een systeem dat aanhoudend oscilleert, heeft een of meerdere polen op de eenheidscirkel liggen. De overdrachtsfunctie van het teruggekoppelde systeem in figuur 5.9 is

$$G_t(z) = \frac{K_c \frac{\widehat{b}_1 z^{-1}}{1+\widehat{a}_1 z^{-1}}}{1 + K_c \frac{\widehat{b}_1 z^{-1}}{1+\widehat{a}_1 z^{-1}}} = \frac{K_c \widehat{b}_1 z^{-1}}{1 + \widehat{a}_1 z^{-1} + K_c \widehat{b}_1 z^{-1}} = \frac{K_c \widehat{b}_1}{z + \widehat{a}_1 + K_c \widehat{b}_1} \tag{5.59}$$



Figuur 5.9: Teruggekoppeld 1e orde systeem met proportionele regelaar K_c.

Het gesloten-lus systeem heeft duidelijk één pool. Omdat een complexe pool nooit alleen voor kan komen (dan zou zijn complex toegevoegde immers een tweede pool moeten zijn), blijven er voor deze pool twee mogelijkheden over indien er sprake moet zijn van een aanhoudende oscillatie: $z = 1$ of $z = -1$. Bij $z = 1$ is de bijbehorende frequentie 0 Hz, waardoor deze afvalt en $z = -1$ als enige optie overblijft. Door dit gegeven in vergelijking (5.59) te gebruiken volgt (let op

vervanging $K_u = K_c$)

$$\begin{aligned} z + \widehat{a}_1 + K_u \widehat{b}_1 &= z + 1 \\ \widehat{a}_1 + K_u \widehat{b}_1 &= 1 \\ K_u &= \frac{1 - \widehat{a}_1}{\widehat{b}_1} \end{aligned} \tag{5.60}$$

Omdat bij $z = -1$ geldt dat de bijbehorende frequentie f_u de helft is van de sampling frequentie f_s geldt voor de oscillatieperiode $T_u = 2T_s$.

Nu deze gegevens bekend zijn kunnen de formules voor de optimale instelling van een Takahashi regelaar de optimale instellingen bepaald worden (zie tabel 4.3):

$$K_c = 0.6K_u\left(1 - \frac{T_s}{T_u}\right) = 0.3\frac{1 - \widehat{a}_1}{\widehat{b}_1} \tag{5.61}$$

$$Ti = \frac{T_u - T_s}{2} = \frac{T_s}{2} \tag{5.62}$$

$$T_d = \frac{T_u^2}{8(T_u - T_s)} = \frac{T_s}{2} \tag{5.63}$$

Samenvattend worden bij systeem-identificatie (“self-tuning”) iedere keer opnieuw de volgende stappen doorlopen:

1. Schat coëfficiënten a en b van het te regelen proces.
2. Bepaal K_u en T_u.
3. Bepaal de optimale parameters (K_c, T_i en T_d) voor de Takahashi regelaar m.b.v. tabel 4.3.

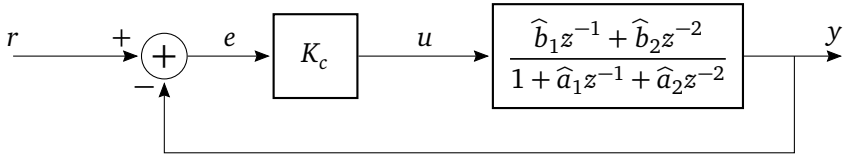
   In de volgende paragraaf wordt deze methode nogmaals toegepast voor een 2^e orde systeem.

5.9 Oscillatiemethode bij een 2e orde systeem

Ook bij het teruggekoppelde 2e orde systeem in figuur 5.10 wordt eerst de overdrachtsfunctie van het totale systeem bepaald:

$$G_t(z) = \frac{K_c \frac{\widehat{b}_1 z^{-1} + \widehat{b}_2 z^{-2}}{1 + \widehat{a}_1 z^{-1} + \widehat{a}_2 z^{-2}}}{1 + K_c \frac{\widehat{b}_1 z^{-1} + \widehat{b}_2 z^{-2}}{1 + \widehat{a}_1 z^{-1} + \widehat{a}_2 z^{-2}}} = \frac{K_c(\widehat{b}_1 z^{-1} + \widehat{b}_2 z^{-2})}{1 + \widehat{a}_1 z^{-1} + \widehat{a}_2 z^{-2} + K_c(\widehat{b}_1 z^{-1} + \widehat{b}_2 z^{-2})} \tag{5.64}$$

$$= \frac{K_c(\widehat{b}_1 z + \widehat{b}_2)}{z^2 + \widehat{a}_1 z + \widehat{a}_2 + K_c(\widehat{b}_1 z + \widehat{b}_2)} = \frac{K_c \widehat{b}_1 z + K_c \widehat{b}_2}{z^2 + \left(\widehat{a}_1 + K_c \widehat{b}_1\right) z + \widehat{a}_2 + K_c \widehat{b}_2}$$



Figuur 5.10: Teruggekoppeld 2e orde systeem met proportionele regelaar K_c.

Ook hier geldt weer dat voor een aanhoudend oscillerend systeem de polen op de eenheidscirkel moeten liggen. Omdat er nu sprake is van twee polen zijn er meerdere mogelijkheden omdat er nu ook complex geconjugeerde paren gevormd kunnen worden. De opties worden hieronder per stuk doorgenomen.

Situatie 1: 2 complex geconjugeerde polen

Een pool wordt in het complexe vlak beschreven door $z_p = \alpha + j\beta$ en zijn complex geconjugeerde door $\bar{z}_p = \alpha - j\beta$. De noemer van de overdrachtsfunctie zou derhalve de vorm

$$(z - z_p)(z - \bar{z}_p) = (z - \alpha - j\beta)(z - \alpha + j\beta) = (z - \alpha)^2 + \beta^2 = z^2 - 2\alpha z + \alpha^2 + \beta^2 \tag{5.65}$$

hebben. Eerder werd verondersteld dat beide polen op de eenheidscirkel liggen, dus er geldt $\alpha^2+\beta^2=1$ en bovenstaande vergelijking vereenvoudigt tot

$$(z-z_p)(z-\bar{z}_p)=z^2-2\alpha z+1 \tag{5.66}$$

Als dit gelijk wordt gesteld aan de noemer in vergelijking (5.64) vinden we (voor K_c wordt wederom K_u ingevuld vanwege de veronderstelde oscillatie):

$$\begin{cases} \widehat{a}_1+K_u\widehat{b}_1=-2\alpha \\ \widehat{a}_2+K_u\widehat{b}_2=1 \end{cases} \Rightarrow \begin{cases} \widehat{a}_1+\frac{1-\widehat{a}_2}{\widehat{b}_2}\cdot\widehat{b}_1=-2\alpha \\ K_u=\frac{1-\widehat{a}_2}{\widehat{b}_2} \end{cases} \Rightarrow \begin{cases} \alpha=\frac{\widehat{a}_2\widehat{b}_1-\widehat{a}_1\widehat{b}_2-\widehat{b}_1}{2\widehat{b}_2} \\ K_u=\frac{1-\widehat{a}_2}{\widehat{b}_2} \end{cases} \tag{5.67}$$

Omdat α het reële deel van het complexe polenpaar is, geldt $\cos(\Omega)=\alpha$, met Ω de genormeerde frequentie van de pool, m.a.w.

$$\Omega=\arccos(\alpha) \qquad \text{met } \Omega=2\pi\frac{f_u}{f_s}=2\pi\frac{T_s}{T_u} \tag{5.68}$$

en derhalve

$$2\pi\frac{T_s}{T_u}=\arccos(\alpha)\Leftrightarrow T_u=\frac{2\pi T_s}{\arccos(\alpha)} \tag{5.69}$$

Hiermee zijn wederom alle benodigde gegevens bekend en kunnen de formules voor de optimale instelling van een Takahashi regelaar uit tabel 4.3 gebruikt worden.

Situatie 2: 2 reële polen op dezelfde locatie

Als het systeem aanhoudend dient te oscilleren, kan de (dubbele) pool zich slechts bevinden in $z_p=-1$. De noemer van de overdrachtsfunctie zou er dan uitzien als

$$(z-z_p)^2=(z+1)^2=z^2+2z+1 \tag{5.70}$$

Deze vergelijking is niets anders dan vergelijking (5.66) met $\alpha = -1$. Derhalve geldt (ook) hier dat

$$K_u = \frac{1-\widehat{a}_2}{\widehat{b}_2} \qquad T_u = \frac{2\pi T_s}{\arccos(-1)} = 2T_s$$

Situatie 3: 2 reële polen op verschillende locaties

Om het systeem aanhoudend te laten oscilleren dient een van de twee reële polen op $z = -1$ te liggen. De tweede pool z_{p_2} kan vrij gekozen worden, maar dient op de reële as en binnen de eenheidscirkel te liggen. De noemer van de overdrachtsfunctie kan daarom als volgt gefactoriseerd worden:

$$(z+1)(z-z_{p_2}) \tag{5.71}$$

Er geldt derhalve dat de noemer in vergelijking (5.64) deelbaar moet zijn door $(z+1)$ en dat de rest van deze deling exact 0 is. Anders was $z = -1$ geen pool van het systeem.

Omwille van de leesbaarheid herschrijven we eerst de noemer in vergelijking (5.64):

$$z^2 + \underbrace{\left(\widehat{a}_1 + K_c\widehat{b}_1\right)}_{c_1} z + \underbrace{\widehat{a}_2 + K_c\widehat{b}_2}_{c_0} = z^2 + c_1 z + c_0 \tag{5.72}$$

Met behulp van een staartdeling kan $z^2 + c_1 z + c_0$ gedeeld worden door $(z+1)$ en er volgt

$$\frac{z^2 + c_1 z + c_0}{z+1} = z + c_1 - 1 + \frac{1 - c_1 + c_0}{z+1} \tag{5.73}$$

Hieruit volgt dat de restterm slechts 0 is indien $1 - c_1 + c_0 = 0$, ofwel

$$1 - \widehat{a}_1 - K_u \widehat{b}_1 + \widehat{a}_2 + K_u \widehat{b}_2 = 0$$

$$K_u = \frac{\widehat{a}_1 - \widehat{a}_2 - 1}{\widehat{b}_2 - \widehat{b}_1} \tag{5.74}$$

Ook hier geldt weer $T_u = 2T_s$ door de pool in $z = -1$.

Bibliografie

[1] Kiam Heong Ang, G. Chong en Yun Li. “PID control system analysis, design, and technology”. In: *Control Systems Technology, IEEE Transactions on* 13.4 (jul 2005), p. 559–576. ISSN: 1063-6536 (geciteerd op p. 27).

[2] K.J. Åström en T. Hägglund. “Revisiting the Ziegler–Nichols step response method for PID control”. In: *Journal of Process Control* 14.6 (2004), p. 635–650. ISSN: 0959-1524 (geciteerd op pp. 43, 47).

[3] G.H. Cohen en G.A. Coon. “Theoretical consideration of retarded control”. In: *Trans. Asme* 75.1 (1953), p. 827–834 (geciteerd op p. 42).

[4] Finn Haugen. “Ziegler-Nichols’ Open-Loop Method”. In: (2010). URL: http://home.hit.no/~hansha/documents/control/theory/zn_open_loop_method.pdf (geciteerd op p. 38).

[5] William S. Levine, red. *The Control Handbook*. Second. CRC Press, 2011 (geciteerd op pp. 40, 43, 47).

[6] A.M. Lopez e.a. “Tuning Controllers with Error-Integral Criteria”. In: *Instrumentation Technology* (1976), p. 57–62 (geciteerd op p. 43).

[7] R.H.C. Takahashi, P.L.D. Peres en P.A.V. Ferreira. “Multiobjective H_2/H_∞ guaranteed cost PID design”. In: *Control Systems, IEEE* 17.5 (1997), p. 37–47 (geciteerd op p. 47).

[8] Bjorn D. Tyreus en William L. Luyben. “Tuning PI controllers for integrator/-dead time processes”. In: *Industrial & Engineering Chemistry Research* 31.11 (1992), p. 2625–2628 (geciteerd op p. 47).

[9] J. G. Ziegler en N. B. Nichols. “Optimum Settings for Automatic Controllers”. In: *Transactions of ASME* 64 (1942), p. 759–768 (geciteerd op pp. 35, 47).