# Digitale regelsystemen

## 1 Inleiding

### 1.1 Bij dit hoofdstuk

Bij de analyse en het ontwerp van regelsystemen is het gebruik van computers nagenoeg onontbeerlijk. Met simulatieprogramma's kan het systeemgedrag worden onderzocht en kan een mathematisch model worden bepaald. Dan is het nog maar een kleine stap om de invloed van de regelacties op het gedrag van het geregelde systeem – letterlijk en figuurlijk – in beeld te brengen.

Een stap verder is echter om met behulp van een computer een proces te regelen. De toepassing van computers voor dit doel is zeer uitdagend. Eén computer kan een geheel proces regelen en organiseren. Dit betekent dat – in theorie – een redelijk goedkoop instrument een heel complex proces in de hand kan houden. In de praktijk valt een en ander overigens nog wel tegen. In verband met de vereiste veiligheid, bedienbaarheid en controleerbaarheid worden zulke computergeregelde systemen veelal nog heel complex.

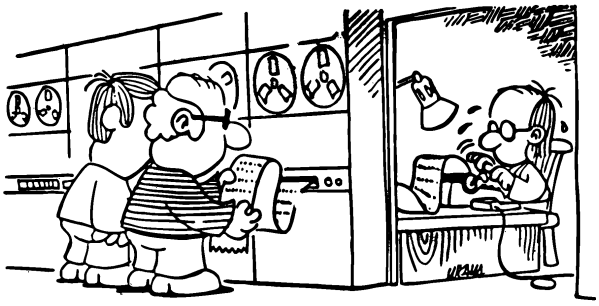
Aspecten als datacommunicatie, bekabeling, presentatie van procesgegevens ten behoeve van de operator, rapportagefaciliteiten enzovoort spelen een belangrijke rol bij de 'value for money' beoordeling. Veel fabrikanten leveren echter de apparatuur (hardware) en de programmatuur (software) voor dergelijke regelsystemen.

De interface tussen mens en machine (zie fig. 1a) en alle facetten van de automatisering, waaronder rapportage (zie fig. 1b) bieden vele uitdagingen aan de regeltechnicus als procesautomatiseerder.

### 1.2 Introductie

In vrijwel elk modern regelsysteem wordt de rol van regelaar uitgevoerd door een computer (microprocessor). Men spreekt dan van een *digitaal regelsysteem*. In de meeste gevallen is de invloed van het digitale karakter van de processor op het systeemgedrag dermate gering dat deze systemen als analoog kunnen worden beschreven, zodat de theorie uit dit boek zonder beperking kan worden toegepast. Voor die gevallen waarin er wel sprake is van

a. ‘optimale’ interactie tussen mens en machine

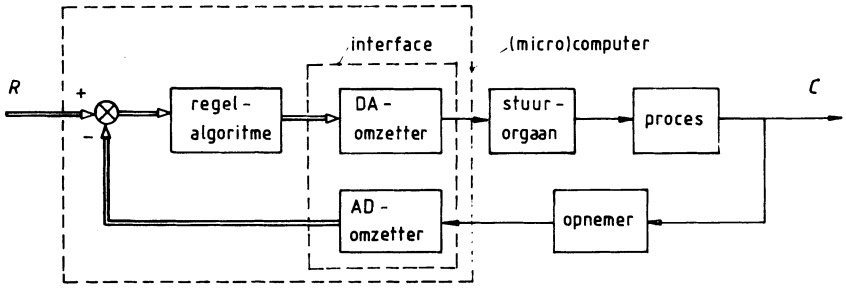


b. ‘volautomatische’ rapportage

**Figuur 1** Mens-machine-interactie en rapportage

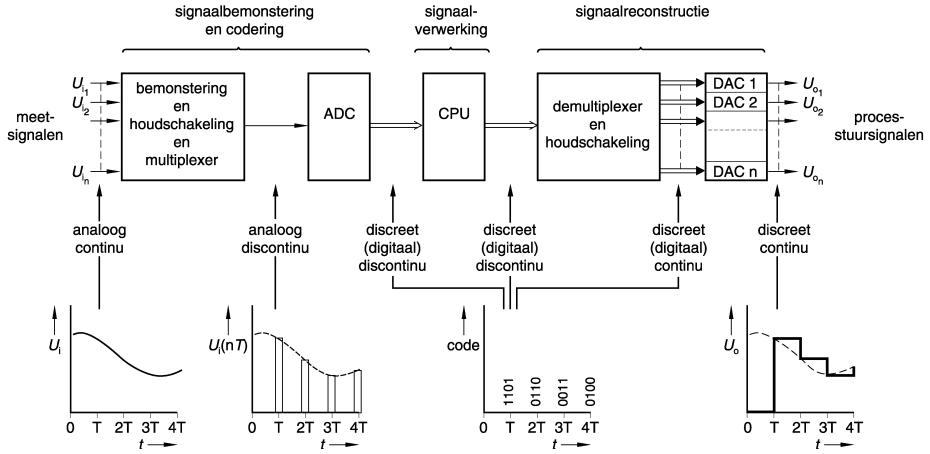
invloed van het digitale karakter en ook om enig inzicht te geven in dergelijke systemen, wordt hier een samenvatting gegeven van de theorie van digitale regelsystemen.
De belangrijkste consequentie van de toepassing van een computer in een regelkring is het feit dat er *signaalbemonstering* moet plaatsvinden. Het sequentiële karakter van de computer maakt dit noodzakelijk. De continue signalen afkomstig van het te regelen proces, de meetsignalen, worden eerst discreet in de tijd gemaakt door het nemen van *signaalmonsters*. Deze worden vervolgens gekwantificeerd door de grootte van deze monsters uit te drukken in een getalcode. Dit proces van het omzetten van een analoog signaal in een digitaal signaal wordt uitgevoerd door de *AD-omzetter* in de *interface* (zie fig. 2).
De bewerking van het verschil tussen gewenste waarde en gemeten waarde – het foutsig-

naal – geschiedt door het regelalgoritme, waarna het resultaat als stuursignaal weer aan het proces wordt aangeboden. Hier moet een omgekeerde signaalbewerking plaatsvinden, om weer een continu signaal af te kunnen geven aan het proces. Deze zogenaamde signaalreconstructie wordt uitgevoerd door de *DA-omzetter* in de interface. Soms is er een directe (digitale) sturing mogelijk, bijvoorbeeld als het stuurorgaan een stappenmotor is.




**Figuur 2** Directe digitale regeling

Het schema in figuur 2 geeft de eenvoudigste vorm van een digitaal regelsysteem weer, de directe digitale regeling (Engels: DDC, Direct Digital Control).
In de praktijk worden vaak meerdere processen of procesdelen door één computer geregeld. In dat geval selecteert een multiplexer eerst het meetsignaal van de desbetreffende regellus, de nieuwe stuurwaarde wordt bepaald en door demultiplexing toegevoerd aan de ingang van het desbetreffende (deel)proces. Daarna gebeurt hetzelfde met het volgende meetsignaal van de volgende regellus enzovoort.
Het is duidelijk dat in zo'n regellus verschillende signaalvormen optreden, waardoor het systeemgedrag kan afwijken van analoge regelkringen. Het deel van de regellus waarin deze afwijkende signalen optreden is weergegeven in figuur 3.
De multiplexers, die de aansluiting van meerdere processen op één computer verzorgen, hebben in principe geen invloed op de signaalvormen, op voorwaarde dat ze voldoende snel zijn.
Duidelijk is dat het analoge ingangssignaal *zowel in tijd als in amplitude* wordt gediscretiseerd. De *discretisatie in de tijd* ontstaat doordat op bepaalde tijdstippen de signaalwaarde wordt bepaald en in de vorm van een amplitude van een puls ('sample') wordt vastgehouden. De discretisatie in de amplitude ontstaat doordat, na de 'sample- en hold-functie', de amplitudewaarde wordt omgezet in een binair gecodeerd getal met eindige woordlengte, door een analoog/digitaal-convertor (ADC).
Het vergelijken van het meetsignaal met de gewenste waarde (setpoint) geschiedt eveneens in de computer: via bijvoorbeeld een keyboard of zogenoemd operatorpanel wordt het referentiesignaal ingevoerd.




**Figuur 3** De signaalvormen in een digitale regellus

In de volgende paragrafen worden achtereenvolgens de belangrijkste onderwerpen uit de digitale regeltheorie besproken, te weten signaalbemonstering, signaalreconstructie en digitale regelalgoritmen (waaronder PID).

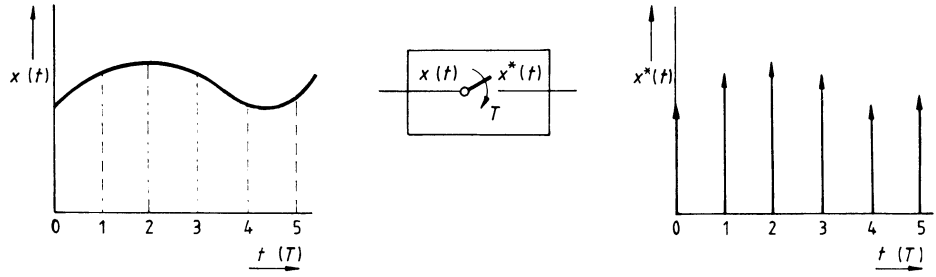
## 2 Signaalbemonstering en reconstructie

Zoals in de inleiding reeds is vermeld, is de belangrijkste consequentie van een computersturing het feit dat de binnenkomende signalen bemonsterd moeten worden. Soms wordt een ingangssignaal direct in digitale vorm aangeboden, bijvoorbeeld bij een codeschijf voor het meten van een hoekverdraaiing.

In figuur 4 is het bemonsteren op equidistante tijdstippen van een continu signaal geschetst. Symbolisch wordt de bemonstering aangegeven door middel van een schakelaar, die elke $T$ seconden (de bemonsteringsperiode) even sluit en de signaalwaarde van $x(t)$ aan de computer doorgeeft. Het bemonsterde signaal wordt aangeduid met $x^*(t)$. Mathematisch – dus in theorie – is het bemonsterde signaal te schrijven als een oneindige reeks pulsen, met gelijke onderlinge afstand $T$ volgens:

$$x^*(t) = \sum_{n=0}^{\infty} x(nT) \cdot \delta(t - nT) \qquad (1)$$

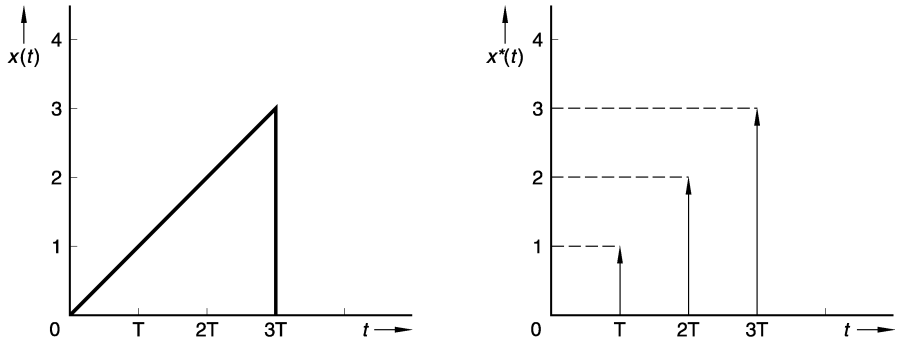
De waarde van de $\delta$-functie is hier gedefinieerd als zijnde 1 op de tijdstippen $t = nT$, met $n = 0 \rightarrow \infty$.




**Figuur 4** Signaalbemonstering

● **Voorbeeld 1**

Het analoge te bemonsteren signaal heeft een vorm zoals in figuur 5.




**Figuur 5** Een bemonsterd signaal aangegeven met een impulsreeks

Er geldt dan:

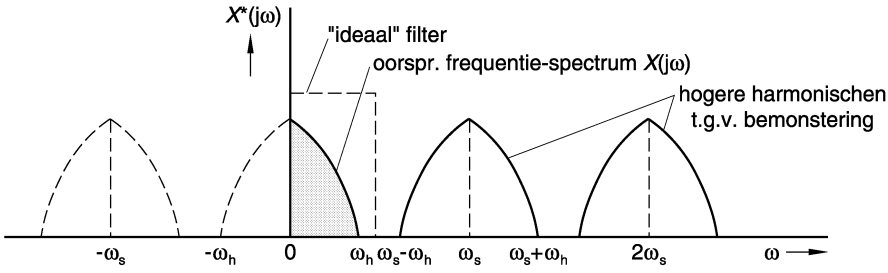
$$x^*(t) = \delta(t - T) + 2\delta(t - 2T) + 3\delta(t - 3T) \qquad (2)$$

. . . . . .

De uitdrukking volgens (1) is voor berekeningen minder aantrekkelijk. Dit is dan ook de reden dat, in geval discretisering in de tijd optreedt, men zijn toevlucht neemt tot een speciale beschrijvingswijze, namelijk de *z-transformatie*. In het kader van deze beknopte verhandeling wordt hierop niet verder ingegaan.
Een frequentieanalyse van het bemonsterde signaal levert via een Fourier-transformatie van de $\delta$-reeks in formule (1) na enig rekenwerk:

$$X^*(j\omega) = \frac{1}{T} \sum_{n=-\infty}^{\infty} X(j\omega + jn\omega_s) \tag{3}$$

waarin $\omega_s = \dfrac{2\pi}{T}$ de bemonsterings(hoek)frequentie voorstelt.
Blijkbaar vertoont het frequentiespectrum van $X^*(j\omega)$ een periodiek karakter. In figuur 6 is dit aangegeven, uitgaande van een bekend spectrum $X(j\omega)$. Gemakshalve is de invloed van de factor $1/T$ in deze weergave genegeerd.



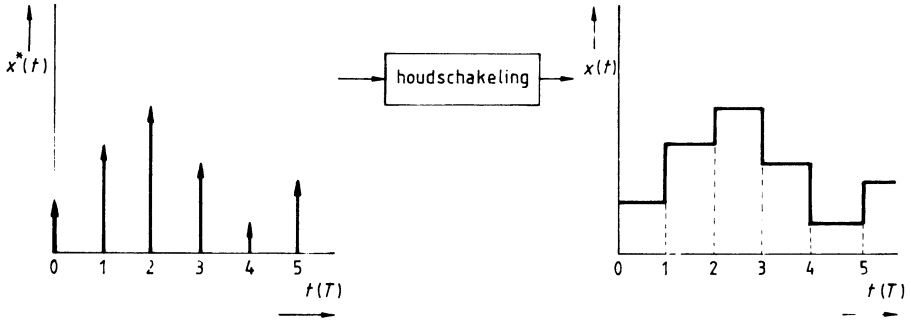

**Figuur 6** Frequentiespectrum

De hoogst voorkomende signaalfrequentie in $X(j\omega)$ is gegeven door $\omega_h$. Het blijkt dat er rond $\omega = 0$, $\omega = \pm\omega_s$, $\omega = \pm 2\omega_s$ enzovoort frequentiebanden ontstaan met breedte $2\ \omega_h$. Er ontstaat door het bemonsteren een signaal met zeer veel en ook hoge frequentiecomponenten. In figuur 6 is ook te zien dat het oorspronkelijke signaal weer terug te krijgen is door de toepassing van een ideaal filter met een laagdoorlaat karakter. Deze correcte signaalreconstructie is niet meer mogelijk indien $\omega_s < 2\omega_h$, omdat de frequentieband van $0 \rightarrow \omega_h$ en die van $\omega_s \rightarrow \omega_h$ elkaar dan overlappen.
Blijkbaar is het voor het correct kunnen reconstrueren van een bemonsterd signaal nodig dat in elk geval geldt: $\omega_s > 2\omega_h$. Deze stelling staat bekend als het *bemonsteringstheorema van Shannon*.
Het overlappen van de frequentiebanden wordt *aliasing* genoemd en dient te worden vermeden. Indien er onverhoopt toch aliasing optreedt, is het oorspronkelijke signaal niet meer juist te reconstrueren.
Aliasing kan op twee manieren worden vermeden, namelijk door:
– verhoging van de bemonsteringsfrequentie tot voldaan wordt aan $\omega_s > 2\ \omega_h$;
– het opnemen van een zogenoemd anti-aliasing filter, waarmee het meetsignaal in frequentie wordt verlaagd tot $\omega_h < \frac{1}{2}\ \omega_s$.

Voordat het stuursignaal wordt toegevoerd aan het proces moet het signaal eerst weer continu worden gemaakt. In tegenstelling tot wat zojuist gesuggereerd is, gebeurt dat meestal niet met behulp van een ideaal filter. De in de praktijk vrijwel uitsluitend voorkomende vorm van signaalreconstructie is gegeven door de betrekking:

$$x(t) = x(nT) \quad \text{voor: } nT \leq t < (nT + T) \tag{4}$$

Dit betekent dat het gereconstrueerde signaal tussen de bemonsteringstijdstippen constant en gelijk aan de laatst berekende signaalwaarde wordt gehouden. In figuur 7 is dit aangegeven.




**Figuur 7** Signaalreconstructie door middel van een nulde-orde houdschakeling

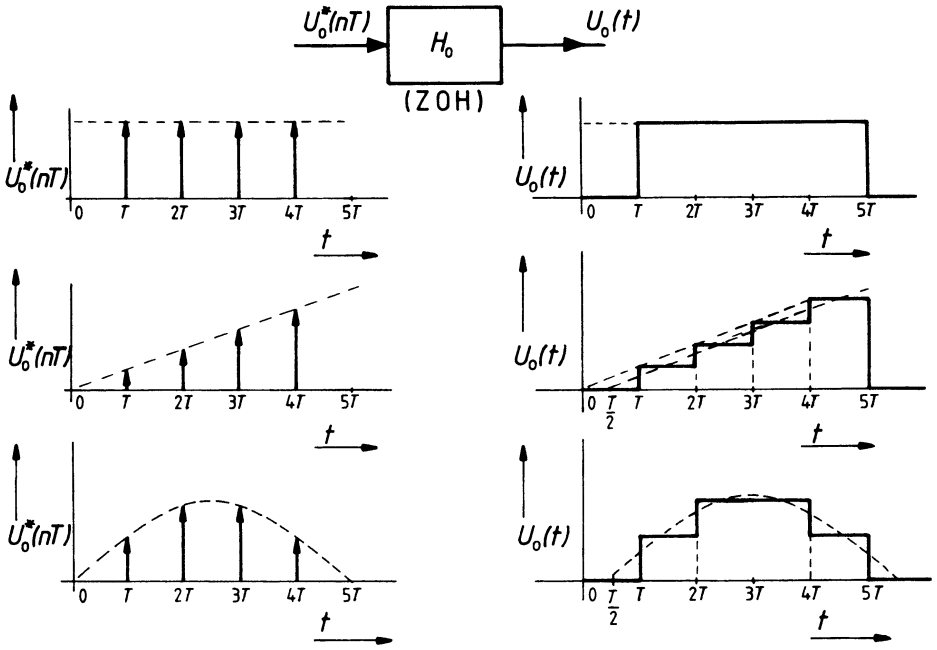
Een dergelijke eenvoudige vorm van signaalreconstructie wordt gerealiseerd door een zogenoemde *nulde-orde houdschakeling* ('zero order hold', ZOH). Zo'n *ZOH-functie* is in een computer eenvoudig te realiseren door het stuursignaal (getal) op een geheugenplaats (buffer) te zetten en te laten volgen door een DA-converter. Zolang er nog geen nieuw stuursignaal berekend is, blijft de DA-converter-uitgangsspanning constant tot de inhoud van de geheugenplaats wordt overschreven door een nieuwe stuurwaarde. Uit het gereconstrueerde signaal van figuur 7 is te zien dat deze vorm van reconstructie verre van 'ideaal', maar aanvaardbaar, is.

Om de invloed van bemonstering en reconstructie (met een ZOH) op het regelgedrag te beschrijven zijn in figuur 8 eerst enkele voorbeelden gegeven. Met een stippellijn is de 'gemiddelde waarde' van het gereconstrueerde signaal aangegeven. Dit middelen kan in de praktijk worden uitgevoerd door een eenvoudig (niet-ideaal) laagdoorlaatfilter.

Zo'n filter blokkeert de hoge frequenties in het signaal. Deze hoge frequenties komen vooral tot uiting in de steile flanken van de signalen.

Hieruit blijkt dat de belangrijkste invloed van deze extra stappen (t.o.v. een analoge regeling) bestaat uit een verschuiving in de tijd. De rafelige vorm van het signaal zal goeddeels weer worden gladgestreken door de filterende werking van het proces zelf (laagdoorlaatfilter). Uit figuur 8 blijkt dat de vertraging ongeveer een halve bemonsteringsperiode bedraagt. Dit blijkt ook uit een wiskundige analyse (die we hier verder achterwege laten). Voorwaarde hierbij is wel dat de bemonstering voldoende snel geschiedt, dus: $\omega_h \ll \omega_s$.

In de praktijk wordt, ook in verband met de stabiliteit van de regelkring, de bemonstering een factor 5 à 10 sneller genomen dan de snelste verandering in het meetsignaal, zodat dus moet gelden:




**Figuur 8** Enkele voorbeelden van signaalreconstructie door een nulde-orde houdschakeling

$$\omega_s = (5 \text{ à } 10) \cdot \omega_h \tag{5}$$

Meestal is het frequentiespectrum van het ingangssignaal onbekend. In dat geval kan worden gerekend met: $\omega_h \approx \frac{2\pi}{\tau_{dom}}$, waarin $\tau_{dom}$ de dominante tijdconstante van het (closed-loop)systeem voorstelt. (N.B. Het gaat hier om grootte-ordes, niet om exacte waarden!)

- **Conclusie**

1. De duur van de bemonsteringsperiode $T$ wordt 5 à 10 keer kleiner gekozen dan de dominante tijdconstante van het als eerste-ordeproces benaderde, geregelde systeem.
2. De totale invloed van het bemonsteren, de signaalbewerking en het reconstrueren in de computer kan voor $\omega_h \ll \omega_s$ worden benaderd door een extra looptijd ter grootte van $\frac{1}{2}T$ seconden.

Bij het ontwerp van een digitale regelkring kan dus, na het kiezen van de bemonsteringsperiode $T$, voor de invloed van de computer een extra looptijd van $\frac{1}{2}T$ seconde worden toegevoegd aan de proceseigenschappen. Vervolgens kan bijvoorbeeld met behulp van een poolbaan-constructie een digitaal te realiseren regelactie worden bepaald.

# 3 Realisatie van digitale regelaars

## 3.1 Beschrijving door een differentievergelijking

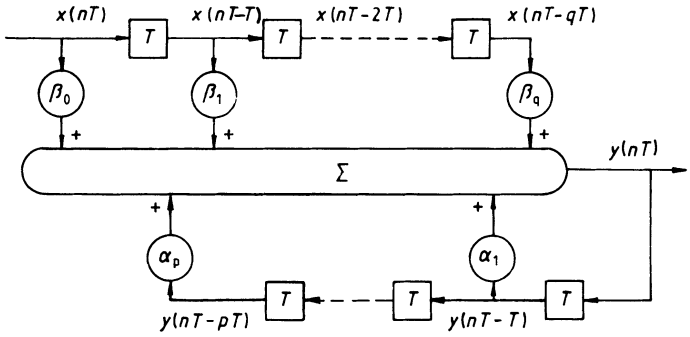
In tegenstelling tot de mathematische beschrijving van systemen met continue signalen (nl. met differentiaalvergelijkingen) worden systemen met discrete signalen beschreven door middel van differentievergelijkingen. Een signaalbewerking (bijv. een regelalgoritme) in de CPU van de computer kan ook in de vorm van een differentievergelijking worden weergegeven. Er wordt immers gewerkt met signaalwaarden op de bemonsteringstijdstippen.
Indien het ingangssignaal (waarop het algoritme wordt toegepast) wordt voorgesteld door $x(nT)$, dan wordt het uitgangssignaal (resultaat van de bewerking) $y(nT)$ gegeven door de volgende differentievergelijking:

$$y(nT) = \sum_{k=1}^{p} \alpha_k \, y(nT - kT) \; + \; \sum_{k=0}^{q} \beta_k \, x(nT - kT) \qquad (6)$$

Het huidige uitgangssignaal $y(nT)$ bestaat dus uit een (gewogen) sommatie van een aantal ($p$) voorafgaande uitgangswaarden en een aantal ($q + 1$) ingangswaarden, inclusief het huidige ingangssignaal.
Op de betekenis van de differentievergelijking volgens (6) wat de realisatie van een bepaalde regeloverdracht betreft, wordt hier verder niet ingegaan; wel zullen enkele veelvuldig voorkomende signaalbewerkingen, zoals PID, worden besproken.

## 3.2 Realisatieschema's

Een signaalbewerking, voorgesteld door de differentievergelijking volgens (6) kan in een zogenoemd *realisatieschema* worden weergegeven. De meest voor de hand liggende realisatie van vergelijking (6) is in figuur 9 aangegeven. Men noemt dit het realisatieschema voor *directe programmering.*




**Figuur 9** Realisatieschema volgens de directe programmering

Op andere programmeringsvormen met hun specifieke voor- en nadelen wordt hier niet verder ingegaan.
Vanuit het *realisatieschema* kan gemakkelijk het bijbehorende computerprogramma (applicatiesoftware) worden geschreven of worden geconfigureerd uit softwareblokken.

## 3.3 Regelalgoritmen

De meest toegepaste methode voor het realiseren van een digitaal regelalgoritme is die waarbij men uitgaat van de overbrengingsfunctie van een continue regelactie. Indien de bemonsteringsperiode *voldoende klein* is, kan de hoofdbewerking in de differentiaalvergelijking $\mathrm{d}f/\mathrm{d}t$ worden benaderd door het differentiequotiënt (de zogenoemde achterwaartse differentie):

$$\frac{\mathrm{d}f}{\mathrm{d}t} \approx \frac{f(nT) - f(nT - T)}{T} \tag{7}$$

In principe kan met deze benadering elke differentiaalvergelijking in een differentievergelijking worden omgevormd die het verband tussen ingangs- en uitgangssignaal beschrijft.

- **Voorbeeld 2**

Een analoge PI-regelaar wordt gegeven door de volgende overbrengingsfunctie:

$$\frac{Y(s)}{X(s)} = G(s) = K_r(1 + \frac{1}{s\tau_i}) \tag{8}$$

Er geldt dan:

$$\tau_i s Y(s) = K_r X(s) + K_r \tau_i s X(s) \tag{9}$$

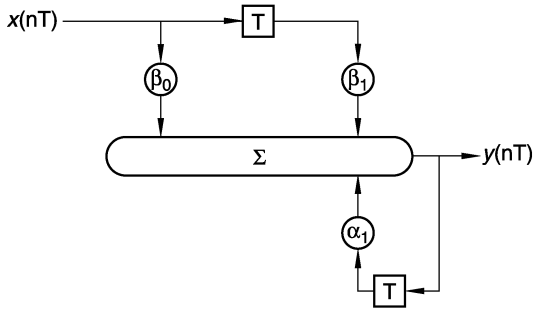
Bedenk hierbij dat geldt: $s \longleftrightarrow \dfrac{\mathrm{d}}{\mathrm{d}t}$, dan volgt hieruit met behulp van (10.7):

$$\tau_i \frac{y(nT) - y(nT - T)}{T} = K_r x(nT) + K_r \tau_i \frac{x(nT) - x(nT - T)}{T} \tag{10}$$

Ofwel:

$$y(nT) = y(nT - T) + K_r(1 + \frac{T}{\tau_i})x(nT) - K_r x(nT - T) \tag{11}$$

Het realisatieschema voor de directe programmering ziet er nu uit zoals in figuur 10.
Hierin geldt voor de factoren $\alpha_k$ en $\beta_k$: $\alpha_1 = 1, \beta_0 = K_r(1 + \dfrac{T}{\tau_i})$ en $\beta_1 = -K_r$.
. . . . . .




**Figuur 10** Realisatieschema voor PI-regelaar

Zo kan ook de realisatie van een digitale PID-regelaar worden gevonden. Voor een analoge PID-regelaar in parallelle vorm geldt de overbrengingsfunctie:

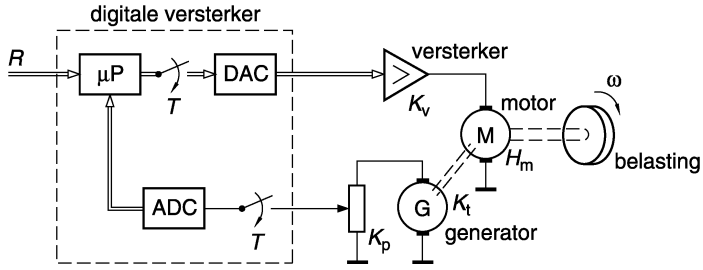
$$\frac{Y(s)}{X(s)} = G(s) = K_r(1 + \frac{1}{s\tau_i} + s\tau_d) \tag{12}$$

Dit levert voor de digitale regelaar de volgende differentievergelijking op (ga dit na!):

$$y(nT) = y(nT - T) + K_r(1 + \frac{T}{\tau_i} + \frac{\tau_d}{T})x(nT) -$$
$$-K_r(1 + \frac{2\tau_d}{T})x(nT - T) + \frac{K_r\tau_d}{T}x(nT - 2T) \tag{13}$$

## 4 Voorbeeld van een directe digitale regeling

Als voorbeeld van een directe digitale regeling geven we hier een snelheids-servosysteem (zie fig. 11).




**Figuur 11** Digitale regeling van een snelheids-servosysteem

Van het systeem zijn de volgende gegevens bekend:
– motor: overbrengingsfunctie

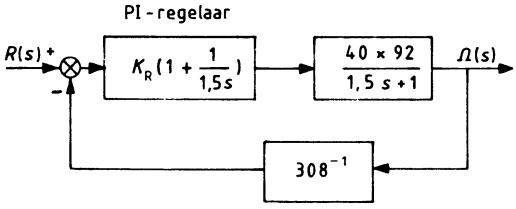
$$H_m(s) = \frac{92}{1,5s + 1} \text{ rad/Vs} \tag{14}$$

- servo-versterker: $K_v = 40$;
- tacho-generator: $K_t = 0,025$ Vs/rad;
- terugkoppelfactor (potentiometer): $K_p = 0,13$.

Van het systeem wordt geëist dat de statische fout (verschil tussen gewenste en gemeten toerental) op een stapvormige setpointverandering gelijk aan nul is, terwijl de tijdconstante van het geregelde systeem circa 0,5 seconde moet bedragen. Indien de regeling met behulp van een analoge regelaar zou worden uitgevoerd, blijkt een PI-regelaar een goede keus, met als overbrengingsfunctie:

$$G(s) = K_r(1 + \frac{1}{s\tau_i}), \qquad \text{met: } \tau_i = 1,5 \text{ seconde} \tag{15}$$

Het blokschema van het geregelde systeem ziet er dan uit als figuur 12.




**Figuur 12** Blokschema van de continue regeling van het snelheids-servosysteem

Voor de overdracht van het geregelde systeem geldt dan:

$$\frac{\Omega(s)}{R(s)} = \frac{308}{s\tau + 1}, \qquad \text{met: } \tau = \frac{0,125}{K_r} \tag{16}$$

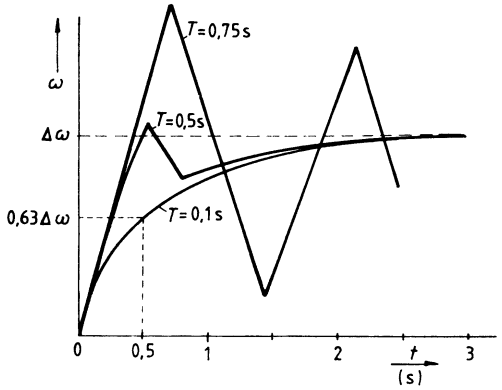
Indien de tijdconstante van het geregelde systeem 0,5 seconde moet bedragen, moet $K_r$ dus gelijk aan 0,25 zijn.
De realisatie van de digitale PI-regelaar is dezelfde als die in voorbeeld 2 behandeld is.
Een geschikte keuze van de bemonsteringsperiode is volgens formule (5). De hoogst voorkomende frequentie $\omega_h$ associëren we hier met de zogenaamde kantelfrequentie van dit systeem:

$$T \approx \frac{1}{5} \cdot 0,5 = 0,1 \text{ seconde} \tag{17}$$

Door deze keuze van de bemonsteringsperiode wordt het acceptabel om de invloed van de digitale regelaar, benaderd door een extra looptijd ter grootte van een halve bemonsteringsperiode, dus 0,05 seconde, te verwaarlozen.
De differentievergelijking voor de regelaar luidt dan:

$$y(nT) = y(nT - T) + 0,267\, x(nT) - 0,250\, x(nT - T) \qquad (18)$$

In figuur 13 zijn enkele stapresponsies van het geregelde systeem weergegeven voor verschillende waarden van de bemonsteringsperiode, namelijk $T = 0,1$ seconde, $T = 0,5$ seconde en $T = 0,75$ seconde. Voor elke waarde is het regelalgoritme hiervoor aangepast.




**Figuur 13** Stapresponsies voor verschillende waarden van de bemonsteringsperiode

Duidelijk is de verslechtering in het dynamisch gedrag waar te nemen indien de bemonsteringsfrequentie afneemt. De benaderingsformule volgens formule (5) wordt immers bij grotere waarden van $T$ steeds minder bruikbaar.

## 5 Samenvatting

Digitale regelsystemen bieden veel voordelen. Het regelprobleem kan meer geïntegreerd worden met het automatiseringsprobleem.
De beschrijving van digitale signalen is in het tijddomein noch in het s-domein gemakkelijk. Er is een speciale behandelingsmethode voor deze problemen, in het zogenoemde $z$-domein, waarmee bijvoorbeeld de stabiliteitsvraag goed kan worden beantwoord.
Digitale regelalgoritmen kunnen het best worden afgeleid van de gebruikelijke analoge regelacties. De keuze van de juiste bemonsteringsperiodeduur is zeer belangrijk. Om aliasing te vermijden moeten de meetsignalen meestal eerst worden gefilterd, alvorens ze in een digitaal signaal om te zetten.

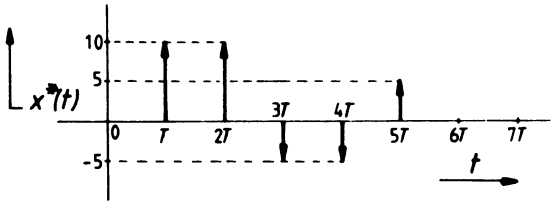
De benadering van de digitaliseringsaspecten door een extra looptijd(je) ter grootte van een halve bemonsteringsperiode biedt de mogelijkheid om het digitale regelsysteem te beschouwen als zijnde analoog.

## 6 Opgaven

1. Hoe luidt het bemonsteringstheorema van Shannon?

2. Een signaalbewerking in de CPU, met ingangssignaal $u_a(nT)$ en uitgangssignaal $u_o(nT)$, wordt beschreven door: $4u_a(0) + 2u_a(-T) = 7u_o(0) + 8u_o(-T)$.
   Er geldt: $u_a(0) = 0$, $u_a(-T) = -1$, $u_a(T) = 1$, $u_o(-T) = 2$.
   Teken het realisatieschema van deze bewerking en bereken: $u_o(0)$ en $u_o(T)$.

3. Gegeven worden de bemonsteringswaarden volgens figuur 14.
   Teken het uitgangssignaal, indien deze bemonsteringswaarden worden toegevoerd aan een nulde-orde houdschakeling.




**Figuur 14** Bemonsteringswaarden

4. Welke benadering voor de gevolgen van signaalbemonstering en signaalreconstructie met behulp van een nulde-orde houdschakeling wordt in de praktijk veel toegepast?

5. Hoe komen we in de praktijk vaak tot een juiste keuze van de bemonsteringsfrequentie $\omega_s$?

6. Bepaal het realisatieschema van het digitale regelalgoritme overeenkomend met de tamme continue PD-regelactie volgens de overbrengingsfunctie: $\frac{Y(s)}{E(s)} = K_R \frac{1+s\tau_d}{1+s\frac{\tau_d}{10}}$.