14 Digitale netwerken

14.1 Inleiding

De belangrijkste toepassing van een digitale computer in een regelkring is de realisatie van een *digitale bewerking* van signalen in een *directe digitale regeling* (DDC). Men noemt een dergelijke realisatie in de computer wel een *digitaal netwerk*; de functies van het digitale netwerk zijn dan:

- Realisatie van de regelacties (PID) of een compensatienetwerk.
- Verwerking en filtering van meetsignalen.

In dit hoofdstuk zullen we vooral bezien hoe de digitale netwerken worden gerealiseerd; meestal wordt de werking aangegeven in een z.g. *realisatieschema*.
Indien een digitale bewerking met behulp van een digitale computer plaatsvindt, moeten een paar aspecten in overweging worden genomen. In de hedendaagse computertechnieken zijn het *aantal geheugenplaatsen* en de *nauwkeurigheid* nauwelijks een probleem. Wel belangrijk is de *rekensnelheid*. Deze is afhankelijk van het type computer en de programmering, hetgeen straks zal blijken.
Realisatieschema's geven aan hoe de berekeningen in de z.g. Central Processing Unit (CPU) van de computer verlopen. Vanuit een realisatieschema kan een programma worden geschreven.
Daar de realisatie van de digitale bewerking – afhankelijk van het instructiepakket van de computer – op verschillende manieren kan plaatsvinden, wordt deze slechts schematisch aangegeven. Op de programmering zelf wordt hier niet ingegaan.
De hoofdbewerkingen van de digitale realisatie zijn:

a Optellen en aftrekken van getallen.
b Het gedurende een aantal bemonsteringsperioden vasthouden van een aantal waarden.
c Vermenigvuldigen met een constante factor.

Deze bewerkingen zijn in een digitale computer zeer gemakkelijk te realiseren.
Het uitgangspunt is de te realiseren z-getransformeerde overbrengingsfunctie $D(z)$:

$$D(z) = \frac{Y(z)}{X(z)} = \frac{b_m z^m + \ldots + b_1 z + b_0}{a_n z^n + \ldots + a_1 z + a_0} \tag{14.1}$$

waarbij $Y(z)$ = uitgangssignaal van het digitale netwerk en $X(z)$ = ingangssignaal van het digitale netwerk.
Na uitdelen van deze uitdrukking volgt:

$$D(z) = \frac{Y(z)}{X(z)} = c_i z^{m-n} + c_{i-1} z^{m-n-1} + \ldots \text{ met } c_i = \frac{b_m}{a_n} \text{ enz.} \tag{14.2}$$

Indien we bedenken dat de term z^{-k} overeenkomt met een *tijdverschuiving* (vertraging!) over k bemonsteringsperioden, houdt dit in dat in ieder geval $m \leq n$ is, omdat anders een positieve macht in z voorkomt, hetgeen op een *voorspelling* zou duiden. Een belangrijke eis die dus aan de te realiseren overbrengingsfunctie $D(z)$ gesteld moet worden is dat de hoogste graad van z in de noemer minstens gelijk is aan de hoogste graad van z in de teller. Indien aan deze eis is voldaan, kan $D(z)$ worden gerealiseerd door de betrekking tussen de in- en uitgang als *differentievergelijking* te schrijven en vervolgens uit deze betrekking de uitgang op de bemonsteringstijdstippen, $y(nT)$, als functie van de overige variabelen te schrijven. Aan de hand van een voorbeeld wordt dit toegelicht.

Voorbeeld 14.1
Gegeven:
De overbrengingsfunctie:

$$D(z) = \frac{Y(z)}{X(z)} = \frac{az}{z+b} \tag{14.3}$$

We kunnen $D(z)$ schrijven als:

$$D(z) = \frac{a}{1+bz^{-1}}$$

of ook:
$$Y(z) + b\,z^{-1}\,Y(z) = a\,X(z) \tag{14.4}$$

Terugtransformatie naar het tijddomein levert de differentievergelijking:

$$y(nT) + by(nT-T) = ax(nT)$$

of
$$y(nT) = ax(nT) - by(nT-T) \tag{14.5}$$

De differentievergelijking volgens (14.5) geeft aan dat de uitgang $y(nT)$ gelijk is aan 'a keer' de bemonsterde ingangswaarde $x(nT)$ verminderd met 'b keer' de *voorafgaande* uitgangswaarde $y(nT-T)$.
In fig. 14.1 is het realisatieschema van uitdrukking (14.5) aangegeven. De vermenigvuldigingsfactoren a en b zijn in rondjes aangegeven, de tijdvertraging van T seconden (een geheugenplaats!) in een rechthoekje.

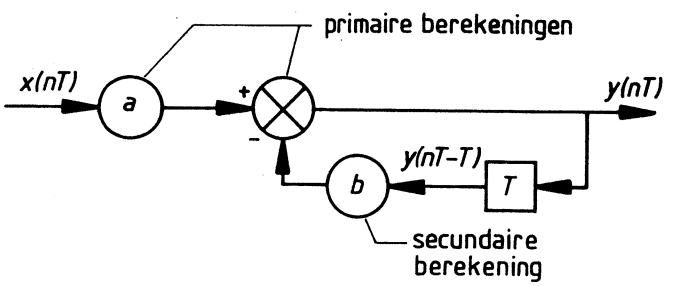


Fig. 14.1 Realisatieschema van $D(z) = \frac{az}{z+b}$

Hoewel het er bij een dergelijk eenvoudig realisatieschema als in fig. 14.1 weinig op aankomt hoe dit is geprogrammeerd, is het toch belangrijk op te merken, dat de berekening door de computer een extra *wachttijd* (looptijd) veroorzaakt. Voor meer ingewikkelde berekeningen zal hiermee terdege rekening moeten worden gehouden. Ook de haalbaarheid van de eisen ten aanzien van de 'sampling-time' T hangt hiervan af. Om de wachttijd, veroorzaakt door de computerbewerking, zo klein mogelijk te maken, dient onderscheid te worden gemaakt in z.g. *primaire* en *secundaire* berekeningen. Deze berekeningen zijn voor het voorbeeld 14.1 in fig. 14.1 aangegeven.
Volgens differentievergelijking (14.5) is de 'bovenste weg' in dit realisatieschema 'direct' nodig. De nieuwe term $by(nT-T)$ hoeft pas op het volgende bemonsteringstijdstip beschikbaar te zijn, zodat bij deze bewerking 'minder haast' nodig is. Deze laatste bewerking kan dus *tussen* de bemonsteringstijdstippen geschieden!
De realisatie volgens fig. 14.1 is een voorbeeld van de z.g. *directe programmering*. In de volgende paragraaf zullen nog enkele andere programmeringsvormen worden besproken. De te kiezen programmeringsvorm hangt af van de toepassing (vereiste snelheid) en het type (micro)computer. Zo kan b.v. het aantal beschikbare registers in de CPU bepalend zijn voor de keuze van de programmeringsvorm.

14.2 Enkele programmeringsvormen

Al naar gelang de realisatie van een digitaal netwerk met overbrengingsfunctie $D(z)$ plaatsvindt, onderscheiden we:
- directe programmering,
- serieprogrammering,
- parallelprogrammering,
- canonieke programmering.

We zullen achtereenvolgens deze programmeringsvormen bespreken en in een voorbeeld uitwerken.

De directe programmering
Uitgangspunt bij de directe programmering is de uitdrukking voor het uitgangssignaal op de bemonsteringstijdstippen volgens:

$$y(nT) = \sum_{k=0}^{m} a_k\, x\,(nT - kT) - \sum_{k=1}^{n} b_k\, y\,(nT - kT) \qquad (14.6)$$

Met behulp van deze uitdrukking is de momentele waarde van het uitgangssignaal te berekenen uit een aantal bemonsteringswaarden van het ingangssignaal en een aantal bemonsteringswaarden van het uitgangssignaal.

Voorbeeld 14.2
We beschouwen het digitale netwerk met overbrengingsfunctie:

$$D(z) = \frac{Y(z)}{X(z)} = \frac{1 - 0{,}25\, z^{-1}}{(1 - z^{-1})(1 - 0{,}5\, z^{-1})} \qquad (14.7)$$

Door uitvermenigvuldiging van de teller en daarna kruiselings vermenigvuldigen ontstaat de uitdrukking:

$$Y(z) - 1{,}5\,z^{-1}\,Y(z) + 0{,}5\,z^{-2}\,Y(z) = X(z) - 0{,}25\,z^{-1}\,X(z) \qquad (14.8)$$

en na terugtransformatie:

$$y(nT) = 1{,}5\,y(nT-T) - 0{,}5\,y(nT-2T) + \\ + x(nT) - 0{,}25\,x(nT-T) \qquad (14.9)$$

Het realisatieschema wordt hiermee als aangegeven in fig. 14.2.

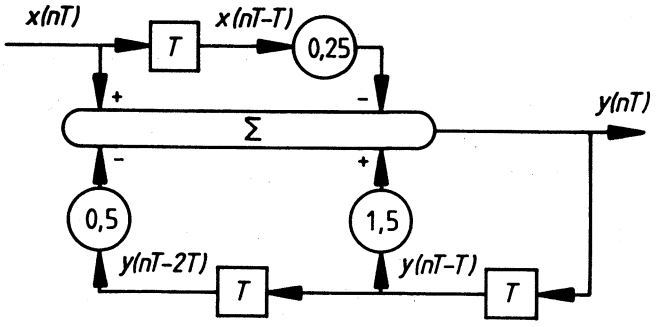


Fig. 14.2 Realisatieschema van $D(z)$ volgens de directe programmering

De serieprogrammering
Het uitgangspunt bij de serieprogrammering is het splitsen van de te realiseren overbrengingsfunctie $D(z)$ in factoren $D_i(z)$ en $G_j(z)$, waarbij:

$$D_i(z) = \frac{1}{d_i z^{-1} + 1} \quad \text{en} \quad G_j(z) = c_j z^{-1} + 1 \qquad \begin{matrix} i = 1, \ldots, n \\ j = 1, \ldots, m \end{matrix} \qquad (14.10)$$

Daarmee wordt $D(z)$ geschreven in de vorm:

$$D(z) = \delta\, G_1(z) \cdot G_2(z) \ldots G_m(z) \cdot D_1(z) \cdot D_2(z) \ldots D_n(z) \qquad (14.11)$$

met δ = constante.

De overbrengingsfuncties $D_i(z)$ en $G_j(z)$ worden afzonderlijk gerealiseerd en vervolgens in *serie* gezet.

Voorbeeld 14.3
Uitgaande van de overbrengingsfunctie $D(z)$ in voorbeeld 14.2 geldt:

$$D(z) = \frac{1 - 0{,}25\,z^{-1}}{(1 - z^{-1})(1 - 0{,}5\,z^{-1})}$$

$$= (1 - 0{,}25\,z^{-1}) \cdot \frac{1}{1 - z^{-1}} \cdot \frac{1}{1 - 0{,}5\,z^{-1}}$$

In fig. 14.3 wordt deze realisatie gegeven.

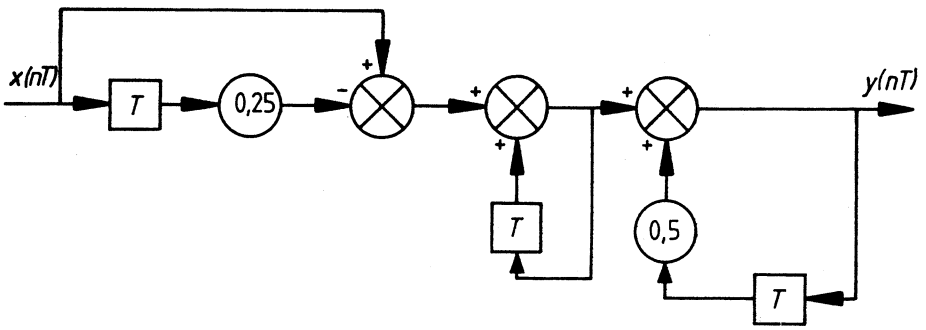


Fig. 14.3 Realisatieschema van $D(z)$ volgens de serieprogrammering

Parallelprogrammering
Bij de parallelprogrammering wordt de te realiseren overbrengingsfunctie $D(z)$ gesplitst in de volgende vorm:

$$D(z) = \sum_{i=1}^{n} D_i(z) \tag{14.12}$$

waarin $D_i(z)$ dezelfde vorm heeft als die bij de serieprogrammering; de overbrengingsfuncties $D_i(z)$ worden nu parallel geschakeld.

Voorbeeld 14.4
De overbrengingsfunctie $D(z)$ is weer dezelfde als in de voorbeelden 14.2 en 14.3 en wordt nu door breuksplitsing geschreven in de vorm:

$$D(z) = \frac{1{,}5}{1 - z^{-1}} - \frac{0{,}5}{1 - 0{,}5\, z^{-1}}$$

De realisatie wordt aangegeven in fig. 14.4.

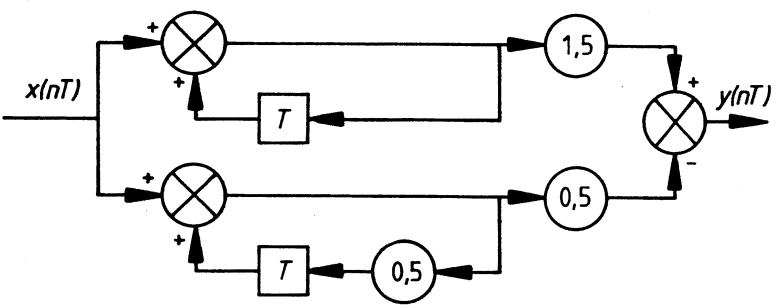


Fig. 14.4 Realisatieschema van $D(z)$ volgens de parallelprogrammering

Canonieke programmering

We gaan uit van de algemene schrijfwijze van de overbrengingsfunctie $D(z)$:

$$D(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + \ldots + b_m z^{-m}}{a_0 + a_1 z^{-1} + \ldots + a_n z^{-n}} \tag{14.13}$$

Hieruit volgt de differentievergelijking:

$$\begin{aligned} &a_0 y(kT) + a_1 y(kT-T) + \ldots + a_n y(kT-nT) = \\ &= b_0 x(kT) + b_1 x(kT-T) + \ldots + b_m x(kT-mT) \end{aligned} \tag{14.14}$$

of ook:

$$\begin{aligned} y(kT) = \frac{b_0}{a_0} x(kT) + \left\{ \frac{b_1}{a_0} x(kT-T) - \frac{a_1}{a_0} y(kT-T) \right\} + \ldots \\ + \left\{ \frac{b_2}{a_0} x(kT-2T) - \frac{a_2}{a_0} y(kT-2T) \right\} + \ldots \end{aligned} \tag{14.15}$$

De algemene realisatie van (14.15) is gegeven in fig. 14.5.

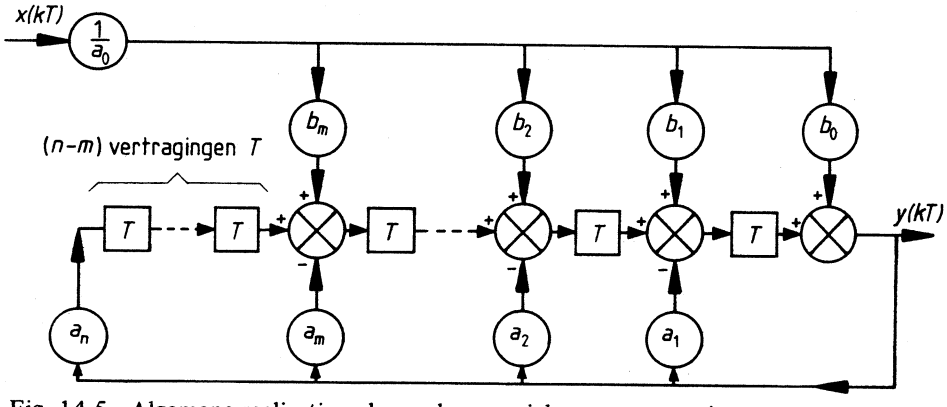


Fig. 14.5 Algemene realisatie volgens de canonieke programmering

Voorbeeld 14.5

Uitgaande van de eerder gerealiseerde overbrengingsfunctie $D(z)$ vinden we nu:

$$\begin{aligned} y(nT) = x(nT) + \left\{ -0{,}25\, x(nT-T) + 1{,}5\, y(nT-T) \right\} + \\ - 0{,}5\, y(nT-2T) \end{aligned}$$

Het realisatieschema wordt nu als aangegeven in fig. 14.6.

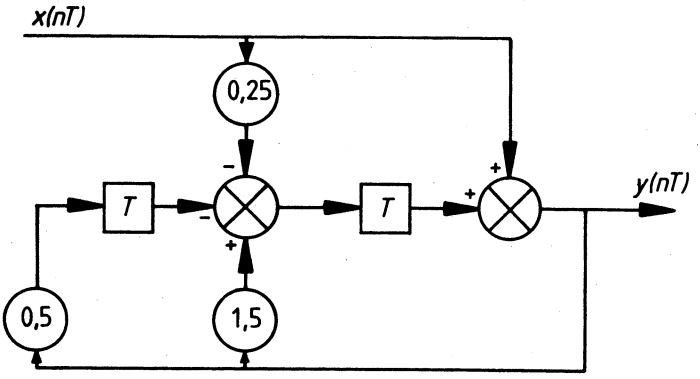


Fig. 14.6 Realisatieschema van $D(z)$ volgens de canonieke programmering

14.3 Regelalgoritmen

In het geval van directe digitale regeling (DDC) moeten de regelacties in de procescomputer worden gerealiseerd. De benodigde regelalgoritmen kunnen worden afgeleid van de realisaties bij continue regelaars. In principe blijft een *proportionele regelactie* een vermenigvuldiging van het foutsignaal met een constante factor (proportionele versterking). Een *integrerende regelactie* ontstaat in principe uit het bepalen van de *som* van een aantal bemonsteringswaarden van het foutsignaal. Een differentiërende regelactie kan in principe worden gerealiseerd door het bepalen van *verschillen* in opeénvolgende bemonsteringswaarden van het foutsignaal.
Behalve de regelacties in het regelalgoritme wordt in de procescomputer ook het aftrekpunt gerealiseerd voor de vergelijking van het 'setpoint' en de gemeten waarde.
In de praktijk worden twee groepen regelalgoritmen onderscheiden, en wel *positie-regelalgoritmen* en *snelheids-regelalgoritmen*.
Bij een *positie-regelalgoritme* is het uitgangssignaal van dit algoritme *een directe maat voor de stand van het corrigerend orgaan* b.v. een klep. Indien de procescomputer uitvalt gaat in dit geval het *corrigerend orgaan* naar de nulstand.
Bij een *snelheids-regelalgoritme* vormt het uitgangssignaal van het regelalgoritme *een maat voor de verstelling van het corrigerend orgaan* ten opzichte van de vorige stand. Indien in dit geval de procescomputer uitvalt, dan blijft het corrigerend orgaan in de voorgaande positie staan. Omdat het regelalgoritme alleen een verstelsignaal afgeeft dient het corrigerend orgaan zelf dusdanig te zijn uitgevoerd dat de voorgaande stand vastgehouden wordt. Is het corrigerend orgaan een spanningsgestuurd element dan kan b.v. een stappenmotor met aangekoppelde potentiometer dienst doen als ontvanger van het stuursignaal van de procescomputer. Het door het regelalgoritme afgegeven verstelsignaal moet dan worden omgezet in een aantal pulsen waarmee de stappenmotor zoveel stappen maakt dat de potentiometer de juiste stuurspanning voor het regelorgaan afgeeft. Indien het corrigerend orgaan een klep is dan zou deze voor het realiseren van een snelheids-algoritme eveneens door een stappenmotor kunnen worden aangedreven.
In de praktijk blijkt het snelheids-algoritme het meest bruikbaar. De voordelen ten opzichte van het positie-algoritme zijn:

1 Bij *uitvallen* (storing, onderhoud) van de digitale regelaar blijft de laatste informatie over het uitgangssignaal bewaard, omdat slechts veranderingen in het stuursignaal wor-

den doorgegeven. De laatste positie van het corrigerend orgaan blijft dan dus gehandhaafd.

2 Het *overschakelen* naar een analoge regelaar (b.v. als 'back-up system') of handbediening geschiedt zonder plotselinge veranderingen (nominale regelaaruitgang gelijk aan nul!), waardoor eventuele beschadigingen aan het corrigerend orgaan en/of verzadigingen in het proces zouden kunnen optreden (Eng. 'bumpless transfer').

3 Het stuurorgaan (b.v. stappenmotor) zorgt zelf voor de *houdwerking*, zodat deze rechtstreeks vanuit de digitale computer kan worden gestuurd (geen DA-omzetter en houdschakeling nodig!)

In fig. 14.7a en b zijn de twee verschillende configuraties aangegeven.

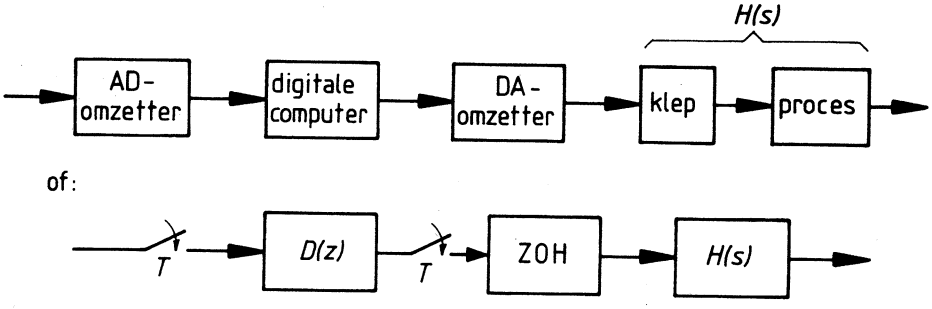


Fig. 14.7a Configuratie en blokschema positie-regelalgoritme

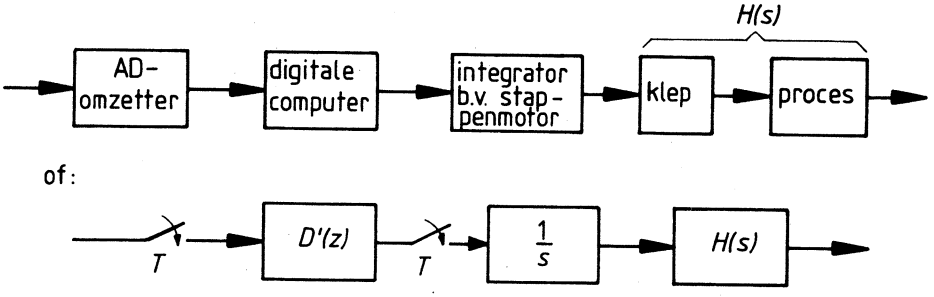


Fig. 14.7b Configuratie en blokschema snelheids-regelalgoritme

Alvorens de bovengenoemde regelalgoritmen nader uit te werken tot PID-algoritmen zullen we eerst de afzonderlijke regelacties beschouwen. Steeds wordt hierbij het ingangssignaal door e en het uitgangssignaal door y voorgesteld. De afleiding van de discrete regelactie vindt steeds plaats door discretisatie van de overeenkomstige analoge regelactie. De toegepaste discretisatie is slechts geoorloofd indien het bemonsteringsinterval T voldoende klein is.

Afzonderlijke regelacties volgens het positie-algoritme

Bij een *proportionele actie* (versterking) moet gelden dat de uitgang van de regelaar recht evenredig is met de ingang volgens:

$$y(t) = k_p e(t), \text{ dus: } y(nT) = k_p e(nT) \tag{14.16}$$

waarin k_p = proportionaliteitsfactor.

In het z-domein levert dit:

$$Y(z) = k_p E(z) \quad \text{of} \quad \frac{Y(z)}{E(z)} = k_p \tag{14.17}$$

Bij een *integrerende actie* wordt het foutsignaal geïntegreerd; dit betekent *optelling* in de discrete regelactie:

$$y(t) = \frac{1}{\tau_i} \int_0^t e(t)dt, \text{ dus } y(nT) = \frac{T}{\tau_i} \sum_{k=0} e(kT) = k_i \sum_{k=0}^{n} e(kT) \tag{14.18}$$

waarin $k_i = \dfrac{T}{\tau_i}$ = integratiefactor.

Ook geldt:

$$y(nT - T) = k_i \sum_{k=0}^{n-1} e(kT) \tag{14.19}$$

Aftrekken van (14.18) en (14.19) levert:

$$y(nT) - y(nT - T) = k_i e(nT) \tag{14.20}$$

In het z-domein levert dit:

$$Y(z) - z^{-1} Y(z) = k_i E(z) \text{ of}$$

$$\frac{Y(z)}{E(z)} = \frac{k_i}{1 - z^{-1}} \tag{14.21}$$

Bij een *differentiërende actie* wordt het foutsignaal gedifferentieerd en in het discrete geval wordt de uitgang van de regelaar evenredig met het *verschil* tussen de regelaaringang op $t = nT$ en op $t = nT - T$ bepaald, dus:

$$y(t) = \tau_d \frac{de(t)}{dt} \text{ en: } y(nT) = \tau_d \frac{(e(nT) - e(nT - T))}{T} =$$

$$= k_d (e(nT) - e(nT - T)) \tag{14.22}$$

waarin $k_d = \dfrac{\tau_d}{T}$ = differentiatieconstante.

In het z-domein levert dit:

$$Y(z) = k_d (E(z) - z^{-1} E(z)) \quad \text{of} \quad \frac{Y(z)}{E(z)} = k_d (1 - z^{-1}) \tag{14.23}$$

Afzonderlijke regelacties volgens het snelheids-regelalgoritme

In plaats van het totale regelsignaal voor de positie van het corrigerend orgaan moet nu de verandering ten opzichte van de voorgaande positie als uitgangssignaal worden gegenereerd.

De formules voor het snelheids-algoritme zijn op eenvoudige wijze uit die voor het positie-algoritme af te leiden. De regelaaruitgang wordt nu met $y'(nT)$ en de overbrengingsfunctie van de regelaar met $D'(z)$ aangeduid.

De *P-actie* van het positie-algoritme wordt:

$$y(nT) = k_p e(nT) \tag{14.24}$$

en ook $$y(nT-T) = k_p e(nT-T) \tag{14.25}$$

zodat $$y'(nT) = y(nT) - y(nT-T) = k_p(e(nT) - e(nT-T)) \tag{14.26}$$

In het z-domein levert dit:

$$Y'(z) = k_p\,(E(z) - z^{-1}E(z)) \quad \text{of: } D'(z) = \frac{Y'(z)}{E(z)} = k_p(1 - z^{-1}) \tag{14.27}$$

De I-actie wordt gevonden uit formule (14.20):

$$y'(nT) = y(nT) - y(nT-T) = k_i e(nT) \tag{14.28}$$

In het z-domein levert dit:

$$Y'(z) = k_i E(z) \quad \text{of: } D'(z) = \frac{Y'(z)}{E(z)} = k_i \tag{14.29}$$

De D-actie wordt:

$$y(nT) = k_d(e(nT) - e(nT-T))$$

en $$y(nT-T) = k_d(e(nT-T) - e(nT-2T))$$

zodat $$y'(nT) = y(nT) - y(nT-T) = k_d(e(nT) - 2e(nT-T) + e(nT-2T)) \tag{14.30}$$

In het z-domein levert dit:

$$Y'(z) = k_d(E(z) - 2z^{-1}E(z) + z^{-2}E(z))$$

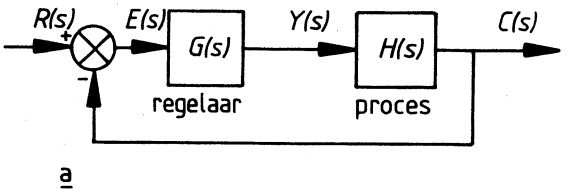
of: $$D'(z) = \frac{Y'(z)}{E(z)} = k_d(1 - 2z^{-1} + z^{-2}) \tag{14.31}$$

In tabel 14.1 zijn de regelacties nogmaals naast elkaar gezet.

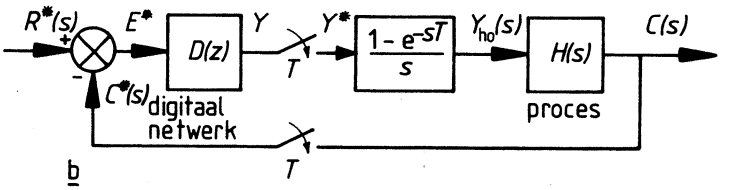
$k_i = \frac{T}{\tau_i}$ $k_d = \frac{\tau_d}{T}$ regelactie	E(z) D(z) Y(z) positie-algoritme	E(z) D'(z) Y'(z) snelheids-algoritme
P: $y(nT) = k_p e(nT)$	k_p	$k_p(1 - z^{-1})$
I: $y(nT) = y(nT-T) + + k_i e(nT)$	$\frac{k_i}{1 - z^{-1}}$	k_i
D: $y\,n(T) = k_d(e(nT) - e(nT - T))$	$k_d(1 - z^{-1})$	$k_d(1 - 2z^{-1} + z^{-2})$

Tabel 14.1 Regelacties voor positie- en snelheids-regelalgoritmen

Uit tabel 14.1 blijkt hoe we een discreet regelalgoritme kunnen realiseren, uitgaande van gewenste proportionele-, integrerende- en differentiërende acties.
Vergelijking van de analoge en de digitale regelconfigurantie, zie fig. 14.8, leert echter dat feitelijk ook het effect van de houdschakeling bij het ontwerp van de regelactie dient te worden betrokken.



a



b

Fig. 14.8 Vergelijking tussen analoge en digitale procesregeling

Indien de bemonsteringsfrequentie voldoende hoog kan worden gekozen, zodat de invloed van de (nulde orde) houdschakeling kan worden verwaarloosd, dan kunnen de regelacties dus worden ontworpen aan de hand van de resultaten in tabel 14.1.
De waarden van de integratietijd en van de differentiatietijd worden daarbij door vermenigvuldiging respectievelijk deling door T omgevormd tot de factoren k_i en k_d. De integratie- en differentiatietijden hebben daarbij waarden als ware het proces analoog geregeld.

Als te verwachten is dat door het bemonsteren dusdanige invloeden ontstaan, dat daarmee bij het ontwerp van de regelacties rekening moet worden gehouden, dan kan als volgt worden gehandeld:

1 De bemonsteringseffecten worden als een extra looptijd ter grootte van $\frac{1}{2}T$ meegenomen in het ontwerp, door verwerking in het bodediagram of in de gehanteerde instelregels; zie hiervoor hoofdstuk 15. De dan gevonden waarden van τ_i en τ_d worden daarna omgevormd tot k_i respectievelijk k_d, en er wordt verder gewerkt met tabel 14.1.
2 Er volgt een ontwerp van de regelacties in het z-domein, waarbij men in feite niet gebonden is aan een conventionele regelactie, maar een (realiseerbare) combinatie van nulpunt(en) en polen kan worden gekozen.

Opmerkingen:

1 Van de hiervoor gehanteerde benaderingen voor regelacties kan in de praktijk handig gebruik gemaakt worden voor het realiseren van een digitale signaalbewerking. De continue bewerking $\frac{df}{dt}$ wordt daarbij vervangen door de benadering:

$$\frac{f(nT) - f(nT - T)}{T} \tag{14.32}$$

Voorbeeld 14.6
Te realiseren de bewerking:

$$H(s) = \frac{s + 1}{2s + 1} = \frac{Y(s)}{X(s)} \tag{14.33}$$

Er geldt: $sX(s) + X(s) = 2s\,Y(s) + Y(s)$
De bijbehorende D.V. luidt: $\frac{dx(t)}{dt} + x(t) = 2\frac{dy(t)}{dt} + y(t)$
Discretisatie levert:

$$\frac{x(nT) - x(nT - T)}{T} + x(nT) = 2\,\frac{y(nT) - y(nT - T)}{T} + y(nT) \tag{14.34}$$

Met $y\,(nT)$ als gewenst uitgangssignaal vinden we:

$$y(nT) = \frac{1}{T + 2}\,\{2y(nT - T) + (T + 1)x(nT) - x(nT - T)\} \tag{14.35}$$

Na z-transformatie volgt:

$$D(z) = \frac{Y(z)}{X(z)} = \frac{1 + T - z^{-1}}{2 + T - 2z^{-1}} \tag{14.36}$$

2 Indien de digitaal te realiseren functie gegeven is in de vorm van een sommatie van afgeleiden van in- en uitgangssignalen, kan nog sneller tot de z-vorm worden gekomen door de substitutie:

$$s = \frac{1 - z^{-1}}{T} = \frac{z - 1}{zT}$$

Dit volgt uit toepassing van de z-transformatie op uitdrukking (14.32).

Voorbeeld 14.7
Te realiseren de bewerking als gewenst in uitdrukking (14.33).
Dan geldt ook:

$$D(z) = \frac{\frac{z-1}{zT} + 1}{2\frac{z-1}{zT} + 1} = \frac{z - 1 + zT}{2z - 2 + zT} = \frac{1 + T - z^{-1}}{2 + T - 2z^{-1}}$$

Dit is (uiteraard) hetzelfde resultaat als in (14.36).

14.4 Voorbeelden van regelalgoritmen

Met behulp van de in de vorige paragraaf afgeleide betrekkingen die zijn samengevat in Tabel 14.1 zullen uit de overbrengingsfuncties van enkele continue regelaars de digitale regelalgoritmen worden afgeleid.

Voorbeeld 14.8
We gaan uit van het geval dat de continue regelaar een PI-regelaar is en een overbrengingsfunctie heeft als volgt:

$$G(s) = K\left(1 + \frac{1}{sT_i}\right) = K + \frac{K}{sT_i} \tag{14.37}$$

Voor de discrete realisatie als *positie-regelalgoritme* geldt volgens de tabellen:

$$D(z) = k_p + \frac{k_p \cdot k_i}{1 - z^{-1}}, \text{ met } k_p = K \text{ en } k_i = \frac{T}{T_i} \tag{14.38}$$

Hieruit volgt $$D(z) = \frac{a - bz^{-1}}{1 - z^{-1}} \tag{14.39}$$

met $a = k_p\,(1 + k_i)$

en $b = k_p$

Indien het *snelheids-algoritme* wordt toegepast, geldt:

$$\begin{aligned} D'(z) &= k_p\,(1 - z^{-1}) + k_p k_i \\ &= k_p\,(1 + k_i) - k_p z^{-1} \\ &= a - bz^{-1} \end{aligned} \tag{14.40}$$

waarin a en b dezelfde waarden hebben als in (14.39).

Voorbeeld 14.9
We gaan uit van de PID-cascaderegelaar met de overbrengingsfunctie:

$$\begin{aligned} G(s) &= K\left(1 + \frac{1}{s\tau_i}\right)(s\tau_d + 1) \\ &= K\left(1 + \frac{\tau_d}{\tau_i} + s\tau_d + \frac{1}{s\tau_i}\right) \end{aligned} \tag{14.41}$$

De discrete realisatie volgens het positie-algoritme hiervan levert met behulp van de tabel:

$$D(z) = k_p \left[k'_p + k_d\,(1 - z^{-1}) + \frac{k_i}{1 - z^{-1}} \right]$$

waarin $k_p = K$

$$k'_p = 1 + \frac{\tau_d}{\tau_i}$$

$$k_d = \frac{\tau_d}{T}$$

en $k_i = \dfrac{T}{\tau_i}$

Hieruit volgt: $k_d k_i = \tau_d / \tau_i$ en $k'_p = 1 + k_d k_i$ (14.42)

De uitdrukking voor $D(z)$ wordt dan:

$$\begin{aligned} D(z) = \frac{Y(z)}{E(z)} &= k_p \left[\frac{(1 + k_d k_i)(1 - z^{-1}) + k_i + k_d (1 - z^{-1})^2}{1 - z^{-1}} \right] \\ &= \frac{k_p(1 + k_d k_i + k_i + k_d) - k_p(2k_d + 1 + k_i k_d)z^{-1} + k_p k_d z^{-2}}{1 - z^{-1}} \end{aligned} \tag{14.43}$$

14.5 Realisatie van regelalgoritmen

Ten aanzien van de praktische toepassing van digitale regelalgoritmen dienen enige voorzorgsmaatregelen te worden getroffen om ongewenste verschijnselen bij voorbaat uit te sluiten. Deze verschijnselen zijn o.a.:

- het *wegdriften* van het proces bij toepassing van een *snelheids-regelalgoritme* indien opéénvolgende meetwaarden weinig verschillen, b.v. als het uitgangssignaal van een proces nagenoeg constant is of als de bemonsteringsfrequentie relatief hoog is. Indien in een dergelijk geval wordt uitgegaan van een P- of PD-regelalgoritme kan het verstelsignaal steeds zo klein zijn dat hierop door het proces niet wordt gereageerd, b.v. omdat de resolutie van de toegepaste DAC te klein is of omdat het verstelorgaan een bepaalde drempel heeft.
  Een remedie tegen dat verschijnsel is de toevoeging van een I-actie aan het regelalgoritme. Zoals de lezer zelf kan nagaan wordt hierdoor wegdriften voorkomen.
- *digitale oscillatie*. Dit kan optreden ten gevolge van de eindige resolutie van de toegepaste ADC en DAC. In een gesloten regellus ontstaat hierdoor een slingering waardoor afwisselend het minstwaardige bit 0 en 1 wordt. Een ADC en DAC met een hogere resolutie verkleint uiteraard de amplitude van de slingering. Ook kunnen convertoren met een niet-lineaire omzetting worden toegepast. Men doet dit, omdat de omzetting van signalen met een kleine amplitude een relatief grotere fout oplevert dan bij signalen met een grote amplitude. De eigenlijke AD-omzetter wordt dan voorafgegaan door een *signaalcompressor*. Na (uniforme) kwantisering in de AD-omzetter zelf, wordt het geconverteerde signaal weer *geëxpandeerd*.
- zogenaamde *output-wind-up*. Dit kan ontstaan in digitale regelaars doordat b.v. ten gevolge van langdurige integratie van een positief of negatief foutsignaal er een 'overflow'-situatie optreedt in de computer. Daardoor kan het uitgangssignaal naar nul terugvallen. De remedie is, om evenals dit gebeurt in analoge regelaars een signaalbegrenzing toe te passen in de computer. Bovendien zal de 'integrator' veelal 'ontladen' (eng. 'desaturation') moeten worden daar het anders te lang kan duren voordat deze weer normale uitgangswaarden produceert, hetgeen zou kunnen leiden tot grote 'overshoot' in de procesuitgang.
- het effect van *afrondingsfouten* bij de berekeningen volgens het regelalgoritme kan worden gereduceerd door bij deze berekening meer bits mee te nemen dan in het uitgangssignaal tot uitdrukking wordt gebracht. Een te kleine bemonsteringsperiode leidt al snel tot dergelijke fouten (zie bijvoorbeeld het geringe verschil tussen de coëfficiënten a en b in het PI-algoritme volgens (14.39) indien $T \ll \tau_i$).

Verder dient bij de praktische realisatie van de regelaar nog rekening te worden gehouden met een aantal factoren die de hanteerbaarheid van de regeling vergroten, namelijk:

- Er moet een voorziening zijn om de regellus om te schakelen van automatische regeling (AUT) op handinstelling (MAN) en omgekeerd. Zo'n omschakeling dient bovendien zoveel mogelijk 'stootloos' te geschieden (eng. 'bumpless transfer').
- Er dient gemakkelijk gekozen te kunnen worden tussen een positie- en een snelheidsregelalgoritme, afhankelijk van de toegepaste actuator (stuurorgaan), die met het regelalgoritme wordt aangestuurd.
- De differentiërende regelactie dient bij voorkeur geen invloed te hebben, indien het foutsignaal verandert tengevolge van een verstelling van de gewenste waarde (setpoint verandering). Men bereikt dit door de differentiërende actie in de terugkoppellus

op te nemen. De dynamica van het regelsysteem (rondgaande overdracht) verandert hierdoor niet. Evenzo kan dit voor de proportionele regelactie geschieden (eng. 'avoiding proportional and derivative kick').

- Teneinde de 'desaturation' van de integrerende actie sneller te laten verlopen (zie 'output-wind-up'), kan men telkens een *herberekening* in deze bijdrage toepassen, ingeval de maximale (MAX) of minimale (MIN) waarde van het stuursignaal wordt overschreden. Evenzo kan men dit doen tijdens handinstelling, zodat de gewenste 'bumpless transfer' plaatsvindt.
- Uit praktisch oogpunt past men veelal de parallelconfiguratie van de PID-regelaar toe. De sommatie van de proportionele (PROP), integrerende (INT) en differentiërende (DER) regelactie vormt dan het stuursignaal voor de actuator. Men kan dan afzonderlijk de P-, I- en D-actie instellen. De acties beïnvloeden elkaar niet. Merk op dat het vanuit het ontwerpstandpunt vaak handiger is de cascade-configuratie te kiezen, omdat deze beter aansluit bij de polen en nulpunten beschouwingen. Beide regelaars zijn echter in elkaar 'om te rekenen', zie ook § 15.3.

In fig. 14.9 is een praktische realisatie van een PID-algoritme aangegeven, waarin de meeste hiervoor genoemde aspecten zijn verwerkt.

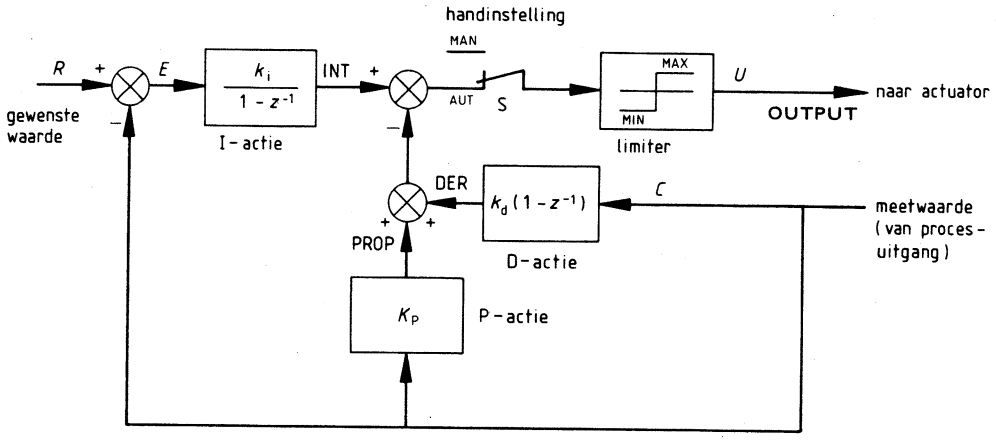


Fig. 14.9 Praktische realisatie van een PID-algoritme

In fig. 14.10 is een stroomschema weergegeven, waarin de genoemde herberekeningen zijn opgenomen.
Vanuit dit stroomschema kan gemakkelijk in de gewenste applicatie-software de benodigde programmatuur worden geschreven.

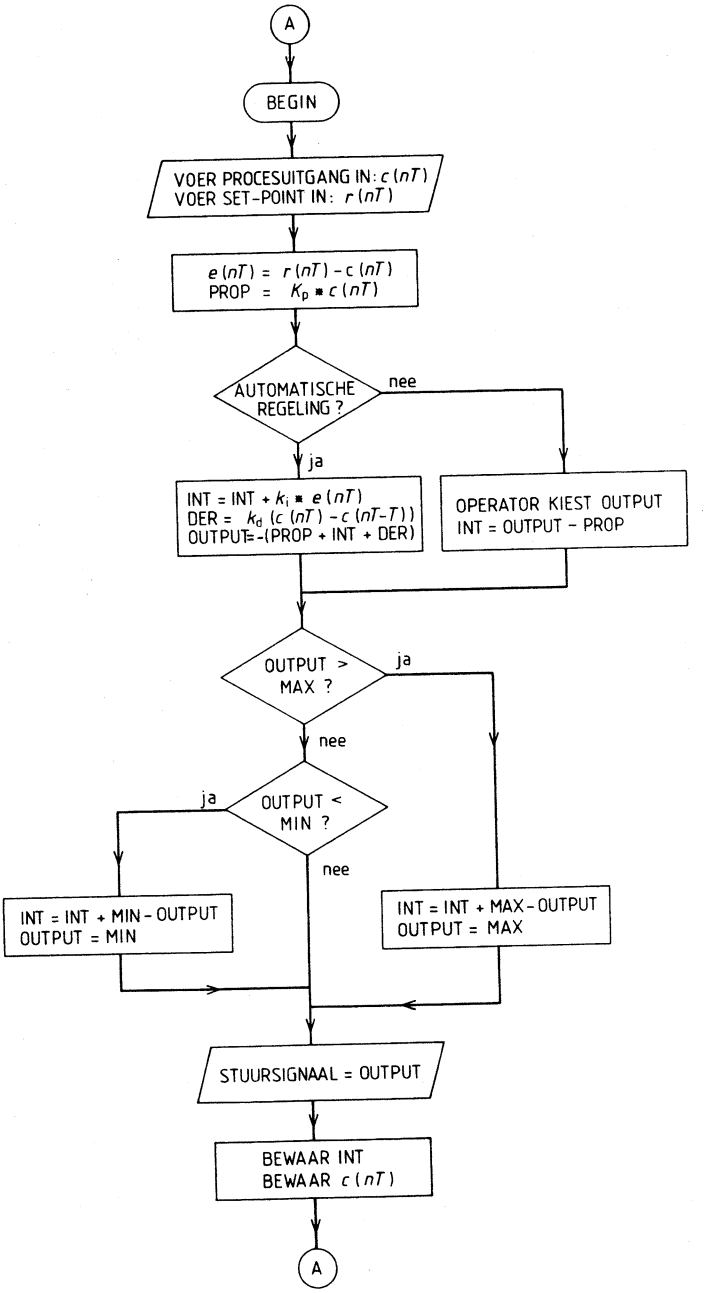


Fig. 14.10 Stroomschema praktische PID-regelaar

14.6 Opgaven

1 *a* Realiseer volgens de directe programmering de overbrengingsfunctie:

$$H(z) = \frac{z^2 + z - 2}{z^2 + 3} \tag{14.44}$$

b Idem voor:

$$H(z) = \frac{1}{z^3} \tag{14.45}$$

c Idem voor:

$$H(z) = \frac{z^2 - z + 2}{z - 1} \tag{14.46}$$

2 Gegeven:
Het realisatieschema van een digitaal netwerk volgens fig. 14.11.

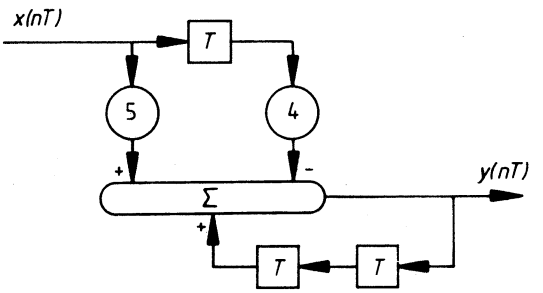


Fig. 14.11 Realisatie van een digitaal netwerk van opgave 2

Gevraagd:
Bepaal de overbrengingsfunctie $\frac{Y(z)}{X(z)}$.

3 Gegeven:
Het volgende realisatieschema van een digitaal netwerk volgens fig. 14.12.

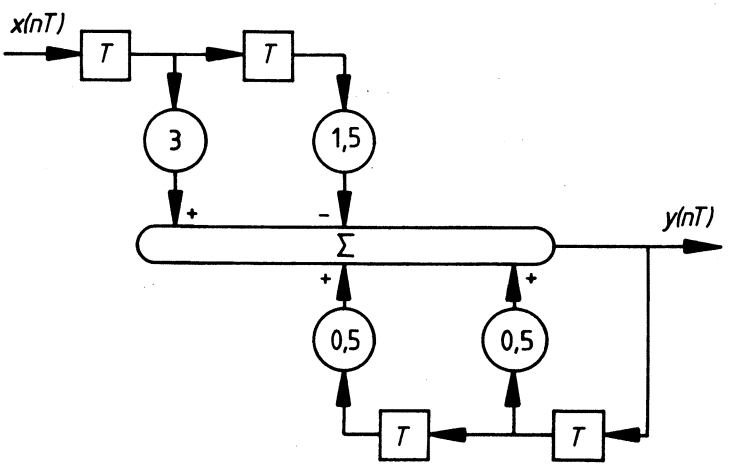


Fig. 14.12 Realisatie van een digitaal netwerk van opgave 3

Gevraagd:

a Bepaal de overbrengingsfunctie $\frac{Y(z)}{X(z)}$.

b Maak het realisatieschema volgens de canonieke programmering.

4 Gegeven:
De schakeling van een continue regelactie volgens fig. 14.13.

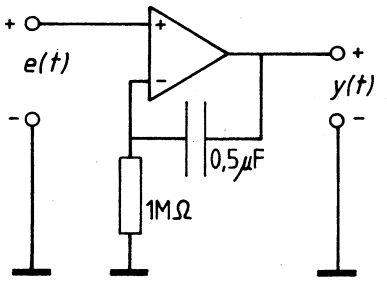


Fig. 14.13 Schema van een continue regelactie

De operationele versterker wordt ideaal verondersteld.
Bepaal de overbrengingsfunctie $D(z)$ van de digitale regelactie met dezelfde functie als de gegeven schakeling voor zowel positie- als snelheids-regelalgoritme. Kies als bemonsteringsperiode $T = 0{,}25$ seconde.

5 Bepaal $D'(z) = \frac{Y(z)}{E(z)}$ voor het digitale snelheids-regelalgoritme overeenkomend met de continue PID-regelaar volgens:

$$y(t) = k_p \left\{ e(t) + \frac{1}{\tau_i} \int_0^t e(t)\,\mathrm{d}t + \tau_d \frac{\mathrm{d}e(t)}{\mathrm{d}t} \right\} \tag{14.47}$$

6 Bepaal $D(z) = \frac{Y(z)}{E(z)}$ voor het digitale positie-regelalgoritme overeenkomend met de continue PD-regelaar volgens de overbrengingsverhouding:

$$\frac{Y(\mathrm{j}\omega)}{E(\mathrm{j}\omega)} = k_p \left\{ \frac{1 + \mathrm{j}\omega\tau_d}{1 + \mathrm{j}\omega \frac{\tau_d}{10}} \right\} \tag{14.48}$$

Maak hierbij gebruik van de benaderingsformule volgens (14.32).