DE ALGORITHMISCHE METHODE

door Prof. Dr. J. Ponstein

Tegenstelling met de formulewiskunde

Veel mensen die wiskunde in de praktijk toepassen zullen er weinig bezwaar tegen hebben dit vak *formule*wiskunde te noemen, omdat er een overstelpende hoeveelheid wiskundige resultaten is dat de vorm heeft van een formule. Wat is de inhoud van een bol met straal R? Antwoord: $4\pi R^3/3$. Wat is de waarde van een kapitaal na 12 jaar als de aanvankelijke waarde K was, en dit bedrag op de bank werd gezet tegen een rente van p procent per jaar, als ook telkens de rente van de rente werd gekweekt? Antwoord: $K(1+p/100)^{12}$. Tenslotte nog een derde voorbeeld, en daarop zullen we steeds terugkomen. Als men telkens de voorraad van een produkt moet aanvullen, als elke aanvulling onmiddellijk geschiedt nadat de wens ertoe wordt geuit en c kost, als de voorraadkosten om het produkt in opslag te houden per dag en per eenheid k zijn, en de vraag naar het produkt v eenheden per dag is, hoe groot moet de aanvulling dan telkens zijn en na hoeveel dagen is de voorraad uitgeput, indien men het *totaal* van aanvullingskosten en voorraadkosten per dag zo klein mogelijk wil houden? Antwoorden respectievelijk: $(2cv/k)^{1/2}$ en $(2c/kv)^{1/2}$.

De formulewiskunde heeft een paar aantrekkelijke voordelen. Vooreerst lost men er hele klassen van problemen in één klap mee op. Zo kan men in de gegeven voorbeelden R, resp. K en p, resp. c, v en k nog willekeurige waarden geven, en voor al deze waarden zijn de formules van toepassing. Bovendien hebben die formules een aangename taakverdeling tot gevolg: de theoreticus leidt ze af, de practicus past ze toe. De laatste hoeft zich niet in te laten met wiskundige bijzonderheden. Wel moet hij de onderstellingen van de eerste zorgvuldig nagaan.

Op een aantal gekompliceerde uitzonderingen na zijn formules ideaal, zolang ze geldig zijn. Maar reeds voor een eenvoudig probleem als het bovengenoemde derde voorbeeld is die geldigheid twijfelachtig. De gegeven antwoorden hebben namelijk betrekking op een *aantal* dagen en een *aantal* eenheden. Het is in de praktijk vaak heel redelijk om te eisen dat de uitkomsten dan ook *geheel*tallig zijn, en dat zijn ze doorgaans volgens de formules niet, ook al zijn c, k en v zelf geheeltallig. Met de eis van geheeltalligheid erbij zijn de formules in voorbeeld drie in de meeste gevallen fout, en moeten ze worden vervangen door het volgende antwoord.

A1. Bereken $(2c/kv)^{1/2}$.

A2. Als de uitkomst een geheel getal is, dan zijn de formules juist, als we bovendien nog aannemen dat v een geheel getal is, en dit zullen we nu steeds onderstellen.

A3. Als de uitkomst geen geheel getal is, bereken dan het grootste getal q dat kleiner is dan $(2c/kv)^{1/2}$. (Dus als $c = v = 1$; $k = 1/10$, dan is $(2c/kv)^{1/2} = 4{,}5$ en $q = 4$.)

A4. Als $q(q+1) \geqslant 2c/kv$, dan is het optimaal aantal dagen q, en is de optimale aanvulling qv.

A5. Als $q(q+1) \leqslant 2c/kv$, dan is het optimaal aantal dagen $q+1$, en is de optimale aanvulling $(q+1)v$.

Het antwoord is er aanzienlijk ingewikkelder op geworden. Soms is het niet eens eenduidig, en wel als $q(q+1) = 2c/kv$, want dan zijn er *twee* optimale oplossingen van het probleem (het probleem met $c = v = 1$ en $k = 1/10$ is er toevallig zo één). Het antwoord heeft de vorm gekregen van een reeks opdrachten in een bepaalde volgorde. Dit nu is karakteristiek voor het *algorithme*. Eerlijkheidshalve moeten we opmerken dat de formulering van enkele van de stappen A1-A5 te wensen over laat, omdat ze meer iets hebben van een bewering, dan van een opdracht. Om dit laatste gaat het: men moet precies weten wat men *doen* moet, ook al zal menigeen willen weten waarom de opdrachten zo en niet anders moeten worden uitgevoerd. Uit een precieze formulering zal ook moeten blijken wat er gedaan moet worden als $q\ (q+1) = 2c/kv$ in ons voorbeeld. We hebben dan de keus uit drie mogelijkheden:

a. Neem als aantal dagen q.
b. Neem als aantal dagen $q+1$.
c. Kies als aantal dagen q of $q + 1$.

Als het aantal dagen eenmaal vast ligt, dan geldt dat natuurlijk ook van de aanvulling, reden waarom we die verder buiten beschouwing laten. Keuzen a. en b. zijn eenvoudiger dan keus c., zodat men in de praktijk de laatste keus buiten beschouwing zal laten. Wij zullen hier echter juist met keus c. verder gaan teneinde een zekere graad van ingewikkeldheid te bereiken! We zullen dus kiezen tussen q of $q+1$, of beter we zullen laten kiezen tussen deze twee mogelijkheden. De vraag is dan „wie kiest er? ". Het zal vaak voorkomen dat we de uitwerking van een probleem overlaten aan een rekenautomaat, zodat we dit redeloze ding zullen moeten laten kiezen. Een rekenautomaat kan echter niet kiezen, zodat mogelijkheid c. dreigt te vervallen. Laten we daarom mogelijkheid c. vervangen door:

c' Neem als aantal dagen q of $q+ 1$ al naar de uitslag van een kansexperiment, waarbij de kans op beide uitkomsten 1/2 is.

Opnieuw komt er een kink in de kabel, omdat opdracht c' niet *uitvoerbaar* is, althans niet door een (digitale) rekenautomaat, want zo'n apparaat is voor de volle honderd procent deterministisch van aard, en kan dus strikt genomen geen kansexperimenten uitvoeren. Het blijkt echter dat het volgende algorithme zeer goed als een benadering kan dienen (over het waarom ervan zullen we ons niet uitlaten).

B1. Stel $x_0 = 1$.
B2. Bereken x_0 maal 455470314 (nu nog een triviale opdracht!).
B3. Deel de gevonden uitkomst door 2147483647, en bepaal van die deling de rest (het quotient doet er niet toe).
B4. Stel de rest gelijk aan x_1 en deel x_1 door 2147483647, en stel de uitkomst gelijk aan r (dus r is een echte breuk).
B5. Herhaal opdracht B2 t/m B4, maar vervang eerst overal x_1 door x_2 en daarna x_0 overal door x_1 (opdracht B1 blijft ongewijzigd).

Herhaal opdrachten B2 t/m B4 opnieuw, maar vervang eerst overal x_2 door x_3 en daarna x_1 door x_2. Enzovoort.
Bereken zo $x_1, x_2 \ldots, x_n, \ldots$.

In het bovenstaande zijn telkens tussen haakjes mededelingen gedaan die niet bij het algorithme horen. Veronderstel nu dat $x_1, \ldots, x_n$ zijn uitgerekend, maar dat de waarde van n niet bekend is, bijvoorbeeld doordat het algorithme B1-B5 voor allerlei doeleinden dienst heeft gedaan; en dat we worden geconfronteerd met de keus tussen q en q+1. Dan kunnen we opdracht c' als volgt als een reeks *uitvoerbare* opdrachten herschrijven:

C1. Bereken x_{n+1} en de daarbij horende waarde van r volgens de stappen B2 t/m B4.
C2. Als $r \leqslant 0{,}5$ neem dan als aantal dagen q.
C3. Als $r > 0{,}5$ neem dan als aantal dagen q+1.

De theorie leert dat de kans op de waarde q vrijwel exact gelijk is aan 1/2, en hetzelfde geldt voor de kans op de waarde q+1. Zoals gezegd is de gevolgde weg om de oplossing van het voorraadprobleem te vinden nodeloos ingewikkeld, maar zij heeft het voordeel dat duidelijk is geworden dat algorithmen niet automatisch uitvoerbaar zijn, en dat we het ene algorithme kunnen gebruiken als subalgorithme in een ander algorithme. Bovendien is de tegenstelling met de formulewiskunde thans maximaal geaccentueerd, want ziet de lezer kans om het optimaal aantal dagen berekend via C1-C3 in één formule aan te geven, zo mogelijk in afhankelijkheid van de waarde van n?

Een omschrijving van wat een algorithme is

Men kan zich heel eenvoudig afmaken van het geven van een definitie van algorithme door te stellen: een algorithme is dat wat zonder meer kan worden vertaald in een computerprogramma. Ofschoon deze definitie aanvaardbaar is, maakt ze enkele aspecten van het algorithme niet voldoende expliciet. Op de eerste plaats moet het algorithme met zijn resultaten voor de dag komen na een *eindig* aantal elementaire bewerkingen, zoals optellen, ongelijkheden testen, enzovoort. Het is natuurlijk heel gemakkelijk opdrachten te bedenken, die een oneindig aantal bewerkingen vergen. Bijvoorbeeld: tel de omgekeerden van de kwadraten van alle positieve gehele getallen bij elkaar op: $1+1/4+1/9+\ldots$ Deze opdracht is tot op zekere hoogte nog uitvoerbaar ook. Ten tweede moeten de opdrachten *konstruktief* van aard zijn. Deze eis overlapt enigszins die van de uitvoerbaarheid. Een algorithme mag bijvoorbeeld nooit de volgende clausule bevatten: „... als er in de decimalebreukontwikkeling van het getal π tien 9's onmiddellijk na elkaar voorkomen, doe dan ...", want niemand weet heden ten dage hoe het in dit verband met π gesteld is, en bovendien weet niemand hoe je in een eindig (!) aantal stappen zou kunnen vaststellen of die tien 9's in de ontwikkeling van π voorkomen of niet. Ten derde moeten de numerieke waarden van grootheden zoals c, k en v in het derde voorbeeld bekend zijn. Dit heeft het onplezierige gevolg dat we in het algemeen het gehele algorithme opnieuw door moeten als we deze

waarden wijzigen. Daar weegt echter het voordeel tegen op dat we met algorithmen problemen te lijf kunnen, die niet onder het beslag van formules kunnen worden gebracht.

Ook al is het algorithme dan een *eindig, welomschreven, uitvoerbaar* proces van *konstruktieve* aard, toch kunnen er zich allerlei verrassingen voordoen bij de uitvoering van een algorithme. Zo is het vaak vrijwel onmogelijk een goede schatting te geven van het aantal bewerkingen dat moet worden uitgevoerd, alvorens het resultaat bekend is. Dit opent de mogelijkheid van algorithmen die zeer veel tijd in beslag nemen, in vele gevallen te veel tijd, zodat ze halverwege moeten worden afgebroken, met dikwijls als triest gevolg dat de reeds bestede tijd voor niets werd besteed. Het is vanwege deze en dergelijke problemen dat het voor de meer ingewikkelde algorithmen een vereiste is dat de theoreticus, die het algorithme heeft bedacht, en de practicus, die er mee moet werken, veel meer samenwerken dan bij de traditionele formulewiskunde nodig is. Dit draait er meestal op uit dat de practicus zich zal moeten inlaten met een aantal technische details van de algorithmen die hij hanteert. Welbekende voorbeelden zijn de algorithmen voor het oplossen van lineaire programmeringsproblemen. Wil men volledig profijt trekken van bijvoorbeeld primaire en duale technieken en van parametrisch programmeren, dan zal men in staat moeten zijn zelfstandig variaties te bedenken, en dat vereist natuurlijk inzicht in deze technieken. Bij primaire technieken werkt men voortdurend met oplossingen die aan alle voorwaarden voldoen, behalve aan die van optimaliteit; bij duale technieken zorgt men er omgekeerd voor dat steeds aan een optimaliteitsvoorwaarde wordt voldaan, maar dit gaat ten koste van het nietgeldigzijn van andere voorwaarden. Bij beide technieken streeft men naar de situatie waar aan alle voorwaarden is voldaan. Parametrisch programmeren betekent de effecten vaststellen van het wijzigen van parameters. Daarbij spelen de zojuist genoemde technieken ook weer een rol. Overigens zij opgemerkt dat veel van deze variaties in de algorithmen zijn ingebouwd, waarvoor men soms moet betalen met een zekere gebondenheid aan de ingebouwde mogelijkheden en met een niet altijd strikt nodige omvang en gecompliceerdheid.

Over de vorm van het algorithme

Hij die een algorithme formuleert moet er van uitgaan dat noch de programmeur die het in een computerprogramma vertaalt, noch de gebruiker, kennis, laat staan inzicht, *moeten* hebben in de details van het algorithme. Een algorithme kan derhalve in hoge mate onbegrijpelijk zijn voor programmeur of gebruiker, als we hier met het woord „onbegrijpelijk" doelen op het waarom van het algorithme. Voor de meeste lezers zal dit vermoedelijk het geval zijn ten aanzien van het algorithme B1-B5. Des te belangrijker is het daarom dat een algorithme verstaanbaar is, dat men het kan lezen en dat men dan precies weet wat men moet doen. Het is als het ware een zaak van bevelen uitvoeren, ook al ziet men de zin van die bevelen niet in. De verstaanbaarheid van algorithmen (en ook van andere zaken) wordt bevorderd door het gebruik van zoveel mogelijk geformaliseerde taal. Hoe meer formali-

satie, des te minder kans op interpretatiefouten. In het uiterste geval past men een volledige formalisatie toe. Dan is het verschil tussen de gehanteerde taal en een zogenaamd probleem-georienteerde *computer*taal (zoals Algol, Fortran, Cobol, enz.) niet groot meer, en is het niet meer dan verstandig zich te bedienen van zo'n reeds bestaande taal.

Laten we terugkeren tot het algorithme B1-B5 en zien wat het gevolg is van een zekere formalisatie. Daarbij letten we in het bijzonder op stap B5, waar in verband met een precieze formulering heel wat aan mankeert, al was het alleen maar om het gebruik van het woord „enzovoort", want hoe moet je dit woord vertalen? We voeren twee nieuwe symbolen in, het eerste komt in vrijwel elk geformaliseerd algorithme voor (of zou er dienst kunnen doen), het tweede komt toevallig voor omdat we te maken krijgen met gehele getallen. In plaats van te schrijven „stel de waarde van p gelijk aan . . .", schrijven we „p: = . . .", waarin „: =" als één symbool moet worden opgevat. B1 kunnen we zodoende vervangen door $x_0 : = 1$. Het tweede symbool heeft betrekking op het bepalen van een rest. Met [p] geven het grootste gehele getal aan dat niet groter is dan p. Dan is na te cijferen dat de rest die wordt verkregen als we a delen door b, gelijk is aan a-[a/b]b. Laat nu het nietnegatieve gehele getal n gegeven zijn (dus bijvoorbeeld n = 0 of n =15, enzovoort). Dan kunnen we B1-B5 als volgt formaliseren:

D1. $x_0 : = 1$.
D2. i: = 0.
D3. *als* i = n *stop, zo niet ga naar* D4.
D4. $x_{i+1} : = 455470314x_i - [455470314x_i/2147483647]2147483647$.
D5. r: = $x_{i+1}/2147483647$ en gebruik r in kansexperiment.
D6. i: = i+1.
D7. *ga naar* D3.

Hoe staat het nu met de uitvoerbaarheid van dit algorithme? Stappen D1 en D2 geven geen problemen in een computer, want men kan met behulp van een computer waarden geven aan variabelen. In stap D3 moeten twee waarden worden vergeleken, en moet eventueel worden gestopt. Beide operaties zijn in de computer ingebouwd. In stap D4 staat rechts van het : = symbool een uitdrukking waarvan de waarde moet worden bepaald. Dat geeft ook geen problemen, omdat de waarde van x_i bekend is! In D5 komt de clausule voor „en gebruik r in kansexperiment". Deze clausule is als zodanig natuurlijk volkomen onaanvaardbaar, maar is hier gebruikt als een *afkorting* van een subalgorithme, waarin het gebruik van r in een kansexperiment nauwkeurig wordt omschreven. Dit subalgorithme zou er als volgt uit kunnen zien, waarbij we verwijzen naar C1-C3 en met d het optimaal aantal dagen aangeven:

E1. d: = q.
E2. *als* r > 0,5 d: = q+1, *zo niet ga naar* E3.
E3. . . .

In D6 wordt de waarde van i met één verhoogd! D7 tenslotte bevat de opdracht terug te keren naar stap D3, en ook op dergelijke *sprong*opdrachten is de computer berekend. Terugspringen naar een reeds eerder uitgevoerde

opdracht betekent een herhaling van de uitvoering van een aantal opdrachten, mogelijk met inbegrip van de sprongopdracht zelf. Kennelijk dreigt een eeuwig durende rondgang. Maar van deze dreiging behoeven we ons niets aan te trekken, omdat de waarde van i bij elke rondgang één hoger wordt en dus vroeg of laat gelijk zal worden aan de waarde van n, op welk moment het programma stopt via stap D3. Het algorithme is dus ook eindig. Verificatie van de andere aan algorithmen gestelde eisen laten we aan de lezer over.

Al bestaat het algorithme uit zeven stappen, de duur van het algorithme is geen konstante waarde, maar hangt af van n. Gemakkelijk is vast te stellen dat het aantal stappen dat wordt uitgevoerd gelijk is aan 5n+3. In verzwakte vorm krijgen we hetzelfde voordeel dat we bij het gebruik van formules hebben, waarbij we immers allerlei parameters onbepaald mochten laten: de waarde van n moet weliswaar bekend zijn voordat we starten met de *uitvoering* van het algorithme, maar het algorithme *zelf* is goed voor alle (nietnegatieve gehele) waarden van n. Bij het gebruik van de computer is het slechts nodig de waarde van n zoals dat in het computerjargon heet *„in te lezen"*.

Een enkel woord over de noodzaak van algorithmen

Zelfs in een vak als fysica, waar de formulewiskunde een uiterst belangrijke rol speelt, zijn de algorithmen reeds lang in gebruik, en wel voor het bepalen van de numerieke waarde van allerlei variabelen. Want reeds een simpele formule als $K(1+p/100)^{12}$ levert niet onmiddellijk de numerieke waarde van het antwoord, als bijvoorbeeld K = 2500 en p = 4,75. Numerieke wiskunde is nodig voor de uiteindelijke bepaling van de waarde van de uitkomsten, en deze tak van de wiskunde maakt een intens gebruik van algorithmen. Zal men in de fysica veelal toch van formules uitgaan, de *economie* en aanverwante vakken leveren talrijke problemen, waarvoor het antwoord niet in formulevorm bekend is en vermoedelijk ook niet te geven is. Twee bijzonderheden zijn hiervoor op de eerste plaats verantwoordelijk. Aan de ene kant het feit dat vaak iets moet worden *geoptimaliseerd* (zie het derde voorbeeld), en aan de andere kant het feit dat allerlei samenhangen niet met vergelijkingen kunnen worden beschreven, maar met behulp van *on*gelijkheden moeten worden vastgelegd, zoals dat bijvoorbeeld bij lineaire programmeringsproblemen meestal het geval is.