23

Proceedings van het symposium wiskunde en de computer

onder redactie van
J. van Mill
G.Y. Nieuwland

Centrum voor Wiskunde en Informatica
Centre for Mathematics and Computer Science

ISBN 90 6196 374 5
NUGI-code: 811

Voorwoord

De Nederlandse Mathematisch Congressen zijn in 1964 begonnen vooral met de bedoeling, aan jonge Nederlandse wiskundigen voor hun eerste optreden buiten het eigen instituut, een vriendelijke omgeving te bieden. Sindsdien is veel gebeurd - met de wiskunde, de Nederlandse wiskundigen (jong of anderszins) en hun instituten - dat het belang van *deze* doelstelling heeft gerelativeerd. De congressen zijn gebleven, maar worden door de organisatoren steeds meer gebruikt als een podium om binnen de Nederlandse wiskundige kring zaken van algemeen belang aan de orde te stellen: nieuwe ontwikkelingen, *trends,* controversen zelfs.

In dit licht leek voor het vijfde lustrum *wiskunde en de computer* een heel geschikt thema. 1964 viel alweer tien jaar na de geboorte van de ARRA: de eerste Nederlandse computer gebouwd bij het toenmalige Mathematisch Centrum; een initiatief van wiskundigen in een nog (vrijwel) computeranalfabetisch Nederland. Nog vijfentwintig jaar later is automatisering in ons land een gegeven, maar blijft de vraag of de wiskundigen, na hun veelbelovend begin, op de juiste manier op deze ontwikkeling hebben ingespeeld. Zeker: jaargang na jaargang hebben onze afgestudeerden in het brede terrein van de automatisering hun werk gevonden, maar hun identiteit als wiskundige zijn ze daarbij kwijtgeraakt.

Hoe dit ook zij, de congrescommissie achtte een discussie van de *state of the art* terzake in het kader van dit lustrum op zijn minst op zijn plaats. Gekozen werd voor een symposium in geconcentreerde vorm: vier parallelle sessies van elk drie voordrachten, over zuivere wiskunde, toegepaste wiskunde, stochastiek en didactiek - steeds in interactie met de computer. Om het keuzeprobleem voor de congresbezoekers niet op de spits te drijven, werd besloten te proberen de volledige teksten vooraf ter beschikking te stellen: de redacteuren zijn grote dank verschuldigd allereerst aan de auteurs, maar daarna vooral aan het

Centrum voor Wiskunde en Informatica, die dit huzarenstukje in goede samenwerking mogelijk hebben gemaakt.

Helemaal probleemloos is het natuurlijk niet gegaan: één van de sprekers heeft zich op het laatste moment teruggetrokken, een andere wenste tenslotte geen tekst in te leveren. Dat dit boekje toch twaalf bijdragen telt, is dan ook te danken aan de welwillendheid van prof. dr. N. G. de Bruijn, die toestemming gaf zijn lustrumrede - over het congresthema - op te nemen, en van prof. dr. P. C. Baayen die zijn inleiding op de onder zijn leiding op het congres te voeren discussie ter beschikking heeft gesteld. De doelstellingen van plaatsbepaling, opinievorming en traceren van mogelijke controverse die de congrescommissie bij de keuze van het congresthema voor ogen stonden, zijn in het breed perspectief van hun bijdragen goed uit de verf gekomen.

Bij dit alles is het goed te bedenken, dat bij alle verandering één traditie van de Nederlandse Mathematische Congressen de afgelopen 25 jaar steeds is gehandhaafd, en ook voor de toekomst bewaard verdient te blijven: het is allereerst een *vriendelijke* omgeving.

J. van Mill
G. Y Nieuwland

Inhoud

Zuivere wiskunde en de computer

Toegepaste wiskunde en de computer

Stochastiek en de computer

Didactiek en de computer

Discussie

Wiskunde en de Computer

N.G. de Bruijn
Eikenlaan 2
5671 AB Nuenen

Bij dit jubileum, het 25ste congres van het Wiskundig Genootschap, vraagt men een rede over het thema Wiskunde en de Computer. Aangezien het congres veelvuldig gelegenheid biedt om op allerlei aspecten van het thema grondig in te gaan, zal ik daarbij geen wetenschappelijke verhandeling geven, maar wat luchtiger beschouwingen van algemene aard. Daarbij gaat het niet uitsluitend over de relatie tussen wiskunde en computers, maar ook over wiskunde, over wiskundigen, over informatica, en over wat wiskunde, wiskundigen, computers en informatica aan elkaar kunnen hebben.

Nederland heeft een oude traditie inzake het verbinden van wiskunde en computer. In een vroeg stadium, kort na de oprichting van het Mathematisch Centrum in 1946, werd het ontwikkelen van computergebruik en zelfs het bouwen van computers samen met wiskunde in één instituut ondergebracht. Grote dank zijn wij verschuldigd aan de visie van de oprichters van dat centrum, maar misschien nog groter aan de man die gedurende een lange reeks van jaren de spil was: dat was Adriaan van Wijngaarden. Vandaag lijkt het me de juiste gelegenheid er bij het Wiskundig Genootschap op aan te dringen de nagedachtenis aan die naam vast te leggen, bijvoorbeeld door hem te binden aan een periodiek uit te reiken prijs aan jongeren die de banden tussen wiskunde en computer weten te versterken.

Gedurende een reeks van jaren heeft Nederland in de voorste gelederen gestaan, en even heeft het erop geleken dat in 1951 Amsterdam gekozen zou worden als zetel voor een mondiaal rekencentrum, maar die kans is voorbijgegaan. Uiteindelijk was het de keus tussen Amsterdam en Rome; men koos Rome, en daar kwam het tot niets. In Amsterdam zou het waarschijnlijk wèl een succes geworden zijn.

Ook op dit ogenblik heeft Nederland een goede naam inzake de verbinding

van informatica met wiskunde, en dat is in de eerste plaats te danken aan de traditie afkomende van Van Wijngaarden.

Laat ons eens de vraag stellen wat de invloed is van de computer op het doen en laten van de wiskundige. De vraag naar wat de computer voor de wiskundige te betekenen heeft gaat over twee geheel verschillende dingen: rekenapparatuur en communicatieapparatuur, en die twee zijn bij de huidige technologie in één soort apparaten ondergebracht. Rekenapparatuur is al eeuwenoud, in de vorm van mechanische rekenmachines, en nomogramapparaten zoals rekenlinealen. Hoewel men het geen apparaten noemt, vallen ook tabellen zoals logaritmentafels in deze rubriek. De wiskundigen van de laatste eeuwen hebben zich zelden verwant gevoeld met al deze vormen van rekenhulp, en lieten het werk met genoegen over aan fysici, astronomen en landmeters. Alleen de numerieke specialisten voelden zich er verantwoordelijk voor, maar die kwamen dan ook vaak uit een niet-wiskundige discipline. Het moderne handrekenapparaat doet ongeveer alles wat deze klassieke hulpmiddelen ons boden. Het gemartel met logaritmentafels is bij het schoolonderwijs vervangen door het zoveel plezieriger knoppendrukken. Of het meer inzicht geeft hangt af van de leraar. Bij goed onderwijs aan goede leerlingen geeft het apparaatje duidelijk meer kansen om inzicht en ervaring op te doen. Maar in het algemeen is het onderwijskundig misschien wel neutraal. Als iets weinig moeite kost kan men het snel doen, als het veel moeite kost denkt men er misschien beter bij na.

Het programmeerbare handrekenapparaat is sinds het algemeen gangbaar worden van personal computers aan het verdwijnen, of aan het overgaan in draagbare versies van de personal computer. Laten we dus bij moderne rekenapparatuur de gedachten beperken tot apparaten van het type personal computer of work station, al dan niet aangesloten aan grotere systemen, en nagaan wat dat voor de wiskundige betekent. Dat het voor numerici een belangrijke zaak is, behoeft geen betoog. Het feit dat er bij zoveel toepassingen, overal in wetenschap en techniek, op zeer grote schaal numeriek computerwerk wordt verricht, heeft een klasse van wiskundige numerieke specialisten doen ontstaan die blijvend nuttig werk van hoge kwaliteit kunnen doen. Daaromheen groeit natuurlijk wiskundige theoretische ondersteuning, die in karakter niet afwijkt van wat men zuivere wiskunde noemt.

Maar hoe zit het met de wiskundigen die niet tot deze numerieke wereld behoren? Die hebben altijd geleerd het rekenen te vermijden, en in de plaats daarvan analytisch werk te doen. Of ze werkten in gebieden die helemaal niets met numerieke realisatie te maken hadden. Het zou weinig zin hebben bij de opleiding van zulke wiskundigen veel aan numeriek werk te doen. Laten we ervoor oppassen onze smaak overmatig te laten beïnvloeden door onze hulpmiddelen.

Maar, zal men zeggen, ook analytisch werk kan door een computer gebeuren, met steeds betere systemen van formule-manipulatie. Maar dat is nog de vraag. Het onaangename formulewerk uit de elementaire calculus speelt in de wiskundige analyse maar een heel beperkte rol. Als de analyticus er toevallig

mee te maken krijgt, is het juist bijzonder goed dat hij het helemaal zelf kan, want het doel is meestal om tot inzicht in generalisatie te komen. Alleen als de analyticus in de positie verkeert om uitvoerig formulewerk voor anderen te moeten doen, zal hij computerhulp inschakelen. Maar in dat geval kunnen en zullen die anderen die computerhulp zelf kunnen oproepen, evenals vroeger de landmeter mechanische en tabellarische rekenhulp kon hanteren zonder hulp van een wiskundige. Kortom, veel van wat computers kunnen rekenen met getallen of met formules ligt op routine-matige terreinen. En dat is juist niet wat wiskundigen willen. Zodra we buiten die routines komen, zijn computers nog niet zo goed. En ze zullen niet gauw veel beter worden, want we zijn zelf niet in staat zijn duidelijk te specificeren wat we eigenlijk willen.

Een geheel andere kwestie is het gebruik van computers voor wiskundig experimenteerwerk. Bij wiskundig onderzoek krijgt men vaak de ideeën aan de hand van enkele doorgerekende voorbeelden. Vaak kan dat zonder hulpmiddelen gebeuren, in het bijzonder wanneer de allereenvoudigste gevallen al een goede indicatie vormen. Soms vereist het bekijken van voorbeelden veel werk, en dan kan een computer een uitstekende hulp zijn. Riemann zou zijn vermoeden over de nulpunten van de zetafunctie niet hebben geuit wanneer hij niet met grote analytische vaardigheid in staat was geweest vast te stellen dat de stuk of wat eerste niet-reële nulpunten allemaal op de kritieke lijn lagen. Met een computer heeft men het veel gemakkelijker dan Riemann. Triviaal is het dan nog steeds niet, maar het erbij betrokken rekenwerk wordt uit handen genomen. En bovendien, wanneer men het eenmaal aan de gang heeft, is het een klein kunstje de kwestie veel verder door te rekenen dan in Riemanns vermogen lag.

Het voorbeeld van de zetafunctie is wat uitzonderlijk wegens zijn hardnekkigheid. In de meeste gevallen waarin vermoedens door bekijken van voorbeelden tot stand komen is de zaak veel sneller afgerond. Daar komt bij dat men van het werk dat nodig is om de voorbeelden te genereren vaak al genoeg inzichten verkrijgt om het bewijs rond te krijgen. Deze laatste opmerking kan ons aan het denken zetten. Wanneer men een computer inschakelt om voorbeelden te bewerken kan men leren door het nadenken over het programma. Dat kan een wezenlijke hulp zijn: alleen al het nadenken over het netjes opstellen van probleemspecificatie, voordat het eigenlijke programmeren begint, kan leiden tot inzichten die helpen bij het uiteindelijke bewijs. Maar van de executie van het programma leert men niets. De computer is meestal een zwijgende zwarte doos, die niet klaagt over de last die sommige onderdelen geven, maar alles met brute kracht overwint. Wanneer men zonder machine moet werken is men gedwongen onderweg allerlei trucjes te bedenken om het werk hanteerbaar te houden, en die trucjes kunnen ideeën geven voor het uiteindelijke bewijs. Dat mist men wanneer computers al het werk doen.

In vele soorten van wiskundig werk kan men deze gang van zaken waarnemen, zowel bij eindige, combinatorische of algebraïsche dingen, als bij continue. Steeds moet men erop bedacht zijn dat men minder leert dan bij handwerk. Vaak geldt: hoe krachtiger de pakketten, hoe minder men leert.

Maar soms leert men méér door het gebruik van machines, op een niveau waar men anders niet aan zou zijn toegekomen. Dat samenspel met een computer bij het ontdekken van samenhang is een bron van grote vreugde. Maar er komt ook een soort verslaving om de hoek kijken. Het kan nl. tot gevolg hebben dat men de smaak gaat beperken tot alleen die onderwerpen die lekker met een computer gaan, en de andere verwaarloost. Deze neiging kunnen we Procrustisatie noemen, omdat Procrustes het probleem dat zijn gasten soms te lang waren voor zijn bedden, oploste door alles af te hakken wat naar buiten stak.

Maar nu de computer als communicatiemiddel. Dat woord doet het eerst denken aan electronische postverzending, doch dat is maar een klein deel van de zaak. Belangrijker is manuscriptopbouw. In samenwerking met computer, afdrukker en fotokopieerapparaat is een auteur in staat om artikelen en rapporten aan het toetsenbord samen te stellen, en alle handschrift, knip- en plakwerk te vermijden. Voor wiskundig werk met veel lettertypen en formules zijn er op dit ogenblik al prachtige systemen in omloop, zoals TEX van D.E. Knuth. Weliswaar zijn die nog niet overdreven vriendelijk voor de gebruiker, in het bijzonder voor degene die er maar af en toe mee te maken krijgt, maar er zullen ongetwijfeld nog verbeteringen komen.

Een bezwaar is dat er door zulke fraaie systemen veel werk op de schouders komt van degenen die toch al moeten woekeren met de steeds schaarser wordende tijd die voor onderzoek ter beschikking staat.

De wiskundige kan voor het onderzoek ook gebruik maken van de computer als communicatiemiddel waarbij hij zelf de ontvanger is. Op dit ogenblik is hiervan nog niet zo veel te merken, maar er is alle kans op uitbreiding. Wat al werkt is dat men via computernetwerken in de Mathematical Reviews kan bladeren, maar het is te hopen dat deze manier van raadpleging kan worden uitgestrekt tot de publicaties en rapporten zelf, en wel in een goed bijgewerkte vorm, zodat men niet langer het risico loopt dat het bestudeerde werk inmiddels verouderd is. En vooral is er behoefte aan het snel raadplegen van encyclopedische werken. Als men bijvoorbeeld denkt aan het onder leiding van Erdélyi uitgevoerde Bateman-project, dat voor alle speciale functies een zo volledig mogelijk overzicht van formules en stellingen probeert te geven, en als men zich realiseert dat er altijd weer toevoegingen, en, wat erger is, correcties nodig zijn, dan voelt men de behoefte aan een gegevensbank waarin dat standaardwerk op betrouwbare wijze wordt bijgehouden en van annotaties voorzien. Iedereen die wat wetenswaardigs toe te voegen heeft, of wat weet te corrigeren, kan dat doen. Natuurlijk dient men ervoor te zorgen dat er door anonieme bijdragen geen schade of congestie ontstaat. Men moet er niet aan denken dat zo'n gegevensbank door virussen zou worden aangevallen.

Het is duidelijk dat het huidige publicatiewezen op de helling zou moeten. Het is de vraag of het wetenschappelijke tijdschrift zijn huidige vorm moet blijven houden. Een zeer groot deel van de kosten van het publicatiewezen wordt betrekkelijk geruisloos gedragen door auteurs, redactie en referees. Niet alleen de onkosten aan apparatuur maar ook de werktijd wordt voor een flink deel

betaald door de instituten waar deze mensen in dienst zijn. Het zou dus voor de hand liggen het gehele tijdschriftenwezen meer en meer in de publieke sfeer te brengen. De bottleneck van het publicatiewezen is de kwaliteitsbewaking: het refereeing-systeem. Men zou er over kunnen denken om de refereeing te concentreren in een klein aantal instituten voor elk onderwerp, waar refereeing een professionele zaak zou zijn. Dat zou het mogelijk maken om dicht bij elkaar liggend werk van verschillende auteurs tot een geheel te verenigen, zodat de lezer in de eerste plaats die overzichten raadpleegt, en daarna pas misschien de originele bijdragen. Maar dit is niet eenvoudig van de grond te krijgen in tijden van bezuiniging. Alleen een instelling als de American Mathematical Society kan zulke dingen gaan ondernemen. Wie een gigantische organisatie als de Mathematical Reviews kan beheren heeft al heel wat ervaring die voor het herzien van het tijdschriftenwezen nodig is.

Naast de referatentijdschriften is er nog een grote organisatie die nagenoeg geheel met computers wordt gedreven: de Citation Index. Ik heb de indruk dat die bij wiskundigen eigenlijk nog te weinig bekend is.

We hadden het erover wat wiskunde en wiskundigen aan computers hebben. Maar we kunnen de vraag ook omkeren: wat hebben de computers, de informatica en de informatici aan wiskunde.

De voorgeschiedenis van de hedendaagse computer ligt zo goed als geheel op het gebied van het numerieke rekenen. Vanuit die wereld ontsproten ook de hogere programmeertalen, die men in zekere zin als de ruggegraat van de informatica kan beschouwen. In later jaren werd de numerieke wiskunde als inspiratiebron voor de informatica minder belangrijk. Weliswaar is een groot gedeelte van de rekentijd van grote computers aan numerieke berekeningen gewijd, maar daaruit valt voor de informatica veel minder te leren dan in de begintijd. Wel zijn er in de informatica nog belangrijke delen van research die met benaderde reële getallen te doen hebben, zoals robotica en beeldherkenning, maar dat staat meestal niet dicht bij de numerieke wiskunde als wiskundige theorie, evenmin als alles wat rekent met gehele getallen onder de getaltheorie moet worden gerangschikt. De grote problemen van de informatica zijn van geheel andere aard. Ze hebben betrekking op het hanteerbaar maken van grote complexe structuren van allerlei soort. Die structuren zijn vaak zelf weer computersystemen, of componenten van computers. En op dit ogenblik staat de informatica voor de taak een zinvol gebruik te leren maken van de in aantocht zijnde gigantische supercomputers. Over een paar jaren zullen ze er ineens overal zijn, evenals de personal computer het in een slordige tien jaar tot een alledaags huiskamerapparaat heeft gebracht. Die supercomputers zullen de informatici heel wat problemen geven, en dat zijn dan meestal geen problemen uit de numerieke wiskunde, maar problemen over het beheersen van complexiteit. Beheersen van complexiteit betekent in de eerste plaats het aanbrengen van een wiskundige structuur, in de tweede plaats het daarin stellen van wiskundige problemen, in de derde plaats het oplossen van zulke problemen. En ook als de problemen nog niet opgelost zijn, of zelfs nog niet gesteld zijn, zullen er beslissingen moeten worden genomen. De toegepast

wiskundige werkt dan met de natte vinger, de zuivere geeft het eenvoudig op.

Men kan met een geleerd gezicht zeggen dat complexiteit geheel onder de combinatoriek valt, maar de combinatoriek als mathematische discipline heeft nog maar weinig aan algemeen gestructureerde hulpmiddelen om de zaken aan te pakken die men als combinatorische problemen formuleren kan. Wel heeft de wereld van de combinatoriek een redelijk taalgebruik en begrippenapparaat gekweekt dat voor de informatica bruikbaar is.

Als we de wiskundige disciplines zouden moeten noemen die voor de informatica nu en in de nabije toekomst van belang zijn, dan zijn dat de combinatoriek, de logica en algebra. Ik noem de algebra omdat die een goede ondergrond zowel voor combinatorische complexiteit als voor de logica kan bieden. Ook de waarschijnlijkheidsrekening en de statistiek zijn van belang, aangezien in onze samenleving het verbeteren van de gemiddelde prestatie van een systeem de eerste prioriteit heeft, ondanks een eventueel risico van zeer zelden voorkomende rampsituaties.

Een tak van wiskunde en logica die misschien wel eens een geweldige opbloei zou kunnen krijgen door de supercomputers is kunstmatige intelligentie. Of dat werkelijk tot iets zal leiden is op dit ogenblik niet goed te taxeren. Het voor ons duidelijkste onderdeel van kunstmatige intelligentie is theorem proving, waarbij men een computer laat zoeken naar een bewijs voor een door ons zelf opgegeven propositie. Hoewel het in speciale gevallen wel eens verbluffend werkt, verwacht ik er in het algemeen niet veel van.

Wat wèl haalbaar, profijtelijk en interessant zou kunnen zijn is het samenspel van mens en kunstmatige intelligentie, waarbij de mens de grote lijnen geeft en de computer details invult. Bovendien houdt de computer dan de algemene administratie bij.

Voor het bijhouden van de administratie heeft men de beschikking over perfecte justificatiesystemen. Ik gebruik de term justificatie en niet verificatie, omdat men bij verificatie denkt aan het controleren van een detail, van een formule of een enkel bewijs. Met justificatie bedoel ik iets dat veel meer omvat: het rechtvaardigen van een groot geheel, van gehele theorieën en van het samenspel tussen verschillende theorieën. Zonder kunstmatige intelligentie is justificatie een moeizaam bedrijf. Wie de teksten moet opstellen die aan een justificatiesysteem worden aangeboden moet geheel formeel maken wat de wiskundige luchthartig kan doen op grond van ervaring. Die ervaring van de wiskundige houdt meestal wel in dat hij in staat is alles tot in de kleinste details te verzorgen als dat zou moeten. Er is redelijke hoop dat in een verificatiesysteem kunstmatige intelligentie een flink deel kan doen van hetgene de wiskundige meent te mogen overslaan. Erg gewenst is ook dat het kunstmatig intelligente apparaat in staat is te gissen naar de bedoeling van niet geheel volledige mededelingen, en aan de gebruiker bevestiging van die gissingen kan vragen. Voor sommige onderdelen van de wiskunde is te verwachten dat een intelligent justificatiesysteem het werk van de wiskundige wel eens aanzienlijk zou kunnen verlichten. En hoe prettig zou het niet voor de referee zijn van een artikel als een auteur toelichtingen geeft die zijn

klaargemaakt voor automatische verificatie. Dan hoeft 'alleen nog maar' gekeken te worden of het werk nieuw en interessant is.

Op zichzelf beschouwd heeft de gedachte aan een justificatiesysteem niets met computers te maken. Als men afziet van de nog grotendeels in de toekomst liggende hulp van kunstmatige intelligentie moet immers al het werk toch uit mensenhand komen. Maar de menselijke geest is zwak, kan iets over het hoofd zien, en het feit dat een ketting niet sterker is dan de zwakste schakel maakt zeer lange kettingen erg kwetsbaar. Ook de geweldige berg van administratie maakt het gebruik van een computer gewenst. Maar belangrijker is misschien dat de gedachte aan een computer richting geeft aan onze formalisatie. Wat we niet aan een machine kunnen uitleggen is nog niet formeel genoeg.

Bij het bekijken van de relatie tussen wiskunde en informatica komen we tot een kernvraag waarvan de beantwoording organisatorische consequenties zou kunnen hebben. De vraag is: Is de informatica een onderdeel van de wiskunde of is het een zelfstandige discipline? Die vraag heeft verschillende aspecten die we de revue zullen laten passeren.

Eerst moet ik zeggen dat het hanteren van de naam Wiskunde en Informatica al de suggestie wekt dat het ene niet een stuk van het andere zou zijn. Het is zoiets als als Wedstrijdsport en Voetballen. Toch zal ik gemakshalve de termen Informatica en Wiskunde hanteren, en dan bedoelen de Informatica en de Overige Wiskunde. Maar juister lijkt mij om inplaats van Wiskunde en Informatica te spreken over Wiskundige Wetenschappen en Met Name Informatica.

Het antwoord op de vraag of een vak als onderdeel van een ander beschouwd moet worden is in hoge mate historisch bepaald. Voor de informatica is de historie wat te kort om als richtsnoer te dienen. Het kan geen kwaad een paar andere voorbeelden te bekijken.

Is de scheikunde een onderdeel van de natuurkunde? De scheikunde houdt zich bezig met de vraag uit welke atomen de moleculen van een stof zijn opgebouwd, en hoe men daarin verandering kan aanbrengen. Ook de natuurkunde houdt zich bezig met moleculen en atomen, en met nog veel meer. Het is heel lastig om het begrip natuurkunde dusdanig te omschrijven dat de scheikunde erbuiten valt, tenzij men definities geeft waarin de terminologie 'met uitzondering van' wordt gehanteerd. Maar er is een lange traditie om scheikunde en natuurkunde als verschillende vakken te beschouwen. Scheikundige technologie was al eeuwenlang zo goed als onafhankelijk van fysische technologie opgegroeid. Laatstgenoemde werd hoofdzakelijk werktuigbouw genoemd; men denke aan pompen en stoommachines. Ook de elektrotechniek, een 100% dochter van de fysica, heette geen fysische technologie. Aan het begin van deze eeuw leefde de elektrotechniek met werktuigbouwkunde in Delft in één afdeling. Niet zozeer omdat het allebei kinderen van de natuurkunde waren, maar omdat de belangrijkste successen van de elektrotechniek in de voorafgaande decennia hadden gelegen bij de stroomopwekking met behulp van zeer grote machines. Bij enig nadenken zal

iedereen toegeven dat scheikunde, werktuigbouwkunde en elektrotechniek onderdelen van de natuurkunde en de natuurkundige technologie zijn, maar niemand zal de consequentie willen trekken dat ze dan maar bij de natuurkunde moeten worden ondergebracht. Iedereen zal moeten toegeven dat geneeskunde en diergeneeskunde stukken van de toegepaste biologie zijn, maar ook dat is geen reden om ze hun afzonderlijke status te ontnemen.

Bij het beantwoorden van de vraag of een vak als deel van een ander dient te worden beschouwd moeten we niet alleen principes aan de orde stellen, maar vooral kijken naar dingen als het karakter, de leefsfeer, de traditie, de al dan niet gemeenschappelijke opleiding en vooropleiding.

Om een voorbeeld dichter bij huis te nemen kijken we eens naar de mathematische statistiek. Het is toegepaste waarschijnlijkheidsrekening, en dat laatste is een onbetwist deel van de wiskunde. Bijna overal wordt de statistiek dan ook als een deel van de wiskunde beschouwd. De neiging om de toepassing van een vak een zelfstandige positie te geven was bij de statistiek altijd minder sterk dan bij de andere voorbeelden die ik noemde. Dat komt o.a. door de leefsfeer: statistiek wordt evenals wiskunde hoofdzakelijk op papier achter een bureau bedreven. En het begin van de opleiding heeft meestal heel veel gemeen met de opleiding van de andere wiskundigen.

Het ligt voor de hand een vakgebied een discipline te noemen wanneer er in de universitaire opleiding een flinke portie basiskennis voor alle studenten in dat gebied gezamenlijk gedoceerd wordt. In die zin was wiskunde een halve eeuw geleden een discipline, maar gezien de huidige ontwikkelingen is het de vraag of het over 20 jaar nog zo zal zijn. Wat sinds mensenheugenis centraal lag was de voornamelijk 18de eeuwse calculus, maar het onderwijs daarin is danig aan het afkalven. De meeste zuivere wiskundigen zien er niet zoveel nut meer in, en sommigen zien de teloorgang zelfs met enige vreugde, zoals destijds de onderwijshervormers in het vervolgonderwijs met groot enthousiasme de griekse meetkunde opofferden aan het excerceren met slecht begrepen verzamelingsnotaties. In beide gevallen, de meetkunde en de calculus, gaat het om zeer belangrijke verworvenheden van onze cultuur, die gedachtenloos worden opgeofferd aan de wens om snel door te dringen tot het voor ogen staande gebied van superspecialisatie. Daar de numerieke wiskunde niet meer het centrale gebied van de informatica is, ziet het er naar uit dat ook voor informatici die calculus voor de bijl zal gaan.

In het grootste gedeelte van de 20ste eeuw was er van de calculus echter een stuk afgesplitst dat de geest van Cauchy, Riemann en Weierstrasz ademde. Het was de exacte analyse, met epsilons en delta's, waarin men het redeneren met moderne wiskundige strengheid kon leren. Het heeft er alle schijn van dat ook dit onderwerp op de tocht staat. De positie van de analyse als centraal stuk van de wiskunde is al flink ondergraven. In veel onderdelen van de wiskunde, zowel zuiver als toegepast, kan men met heel weinig analyse toe. De exacte analyse wordt dan als een lastig struikelblok ervaren, want moeilijk is het inderdaad. Onze samenleving durft niemand meer ergens toe te dwingen, en daarom zal het onderwijs in de exacte analyse als centraal punt van alle wiskundeopleiding misschien geen lang leven meer beschoren zijn.

Kortom, als we het begrip discipline binden aan het hebben van een flink stuk gemeenschappelijke opleiding, dan is ook de wiskunde geen discipline meer. En het is dan ook niet eerlijk om de vraag of statistiek en informatica delen van de wiskunde zijn, te koppelen aan normen waar de wiskunde zelf al niet meer aan voldoet.

Laten we eens even doen of wiskunde en informatica twee verschillende vakken zijn, om dan vervolgens wat treffende overeenkomsten en verschillen op te merken.

Het meeste van wat er in de informatica leeft gaat net als in de wiskunde met formules en symbolen. Talen met metatalen en interpretaties eromheen. Die interpretaties kan men zich misschien als materieel voorstellen, maar dat is bij veel andere stukken toegepaste wiskunde niet anders. Wat het formele deel betreft gelden dezelfde absolute normen als in de wiskunde. Alles wat gezegd wordt is als definitief bedoeld, als helemaal correct. Zoals de zaken er staan zijn ze er voor eeuwig. Er is geen speld meer tussen te krijgen. De mode, of de smaak, of de mogelijkheid om te generaliseren, kunnen maken dat men iets dat goed was vervangt door iets anders dat ook goed is, maar dat men interessanter, relevanter, bruikbaarder, helderder, mooier vindt. Ik beschrijf hier de ideale situatie. De dagelijkse werkelijkheid laat heel wat fouten zien, maar over het afkeurenswaardige daarvan is men eenstemmig. Dit zijn allemaal karaktertrekken die informatica met wiskunde gemeen heeft. In andere wetenschappen is dat anders. Elke uitspraak is daar nog voor verschillende interpretaties vatbaar. Absolute normen voor de geldigheid van uitspraken zijn er niet. Alleen waar men zich abstract gaat uitdrukken in een wiskundig formalisme, kan men in die andere wetenschappen aanspraak maken op universele geldigheid, tenminste voor het stukje wiskunde, niet voor de interpretatie.

Het is hier niet mijn bedoeling te zeggen dat om deze reden de wiskunde en informatica een hogere status hebben dan andere vakken, want men kan deze absoluutheid met evenveel recht de dood in de pot vinden. In elk geval is informatica in deze zin een wiskundige wetenschap, en eigenlijk veel meer dan sommige andere delen van de toegepaste wiskunde waar men door de nood gedwongen zich hier en daar in speculaties moet begeven.

Voordat we verder gaan moeten we proberen te zeggen wat we onder informatica verstaan. Het ontwerpen en produceren van componenten voor computers rekenen we er niet toe, dat zetten we bij de toegepaste fysica (in ruime zin, het zou ook elektrotechniek of fijnmechanica kunnen zijn). Het ontwerpen van een computer uit zulke onderdelen, ruw gezegd computerarchitectuur, kunnen we weer wèl tot de informatica rekenen. Maar zo'n onderscheid tussen componenten en architectuur is tijdgebonden; het wordt bij het voortschrijden van de technologie steeds vager. Dat betekent dan gewoon dat het begrip informatica zich weer verruimt. Verder is alle nadenken over het gebruik van computers tot de informatica te rekenen, en het ontwerpen van systemen die het anderen gemakkelijker maken om computers te gebruiken. Niet alle computergebruik is informatica, evenmin als het besturen

van een rijwiel werktuigbouwkunde is. Maar het nadenken over het gebruik is wel informatica. Overigens moet je oppassen met wat je over informatica zegt. Alles kan morgen alweer anders zijn.

Gezien de dienende taak van de informatici tegenover gebruikers ligt het voor de hand dat veel werk van informatici tijdgebonden is, en handelt over de mogelijkheden van bestaande systemen en bestaande apparatuur. Maar er is een kern die van de technologie onafhankelijk is, en die een meer blijvende waarde heeft. En het is juist die kern die van wiskundige aard is. Natuurlijk staan vele dingen in de informatica ver af van de centrale onderdelen van de wiskunde. Maar dat is in de wiskunde zèlf niet minder het geval.

En wat verstaan we onder wiskunde? Dat is een oud vak waarvan we niet meer hoeven te zeggen wat het is. Wiskunde is datgene waarmee wiskundigen zichzelf bezig houden (en soms trachten anderen bezig te houden). Dit antwoord verschuift de vraag naar wat dan wel een wiskundige is. In elk geval is het begrip aan veranderingen onderhevig, zij het dan niet zo snel als het begrip informatica. Over wat de aard van de wiskunde is, is erg veel te zeggen. In dit verband wil ik even wijzen op een boek van R.E. Moritz met citaten, meest van onderhoudende aard, over wiskunde en wiskundigen. Het waren er 2160, en dat nog maar tot 1913. Sinds beginnende hoogleraren niet meer zoveel intreeredes houden raakt zulk materiaal ongelukkigerwijze een beetje in de vergetelheid.

Laat me proberen iets te zeggen over het karakter van de wiskunde, en in het bijzonder dat van de wiskunde van deze eeuw. Ik ben me ervan bewust hoe voorzichtig men daarbij moet blijven, want ook een geoefend historicus zou stekeblind kunnen zijn voor de grote veranderingen die zich onder zijn neus aan het voltrekken zijn. Men kan wel zeggen dat de stijl van de wiskunde van de 20ste eeuw ongeveer gestalte kreeg in het prille begin van de eeuw, in de tijd dat mensen als Hilbert vorm gaven aan het axiomatische en sterk algebraïserende denken dat de gehele eeuw kenmerkt. Ook begon deze stijl al in die tijd de toepassingen van de wiskunde te doordringen.

De kern van onze wiskunde is een taal, die zo krachtig en betrouwbaar is dat die als communicatiemiddel tussen wiskundigen universeel geaccepteerd wordt. Het is die taal die de wiskunde typeert, en niet het karakter van de onderwerpen die besproken worden. Weliswaar ligt niet helemaal vast wat precies tot die taal gerekend moet worden, hoeveel logica erin hoort, in hoeverre dingen als verzamelingen en gelijkheid tot de regels van de taal zelf behoren, maar dat geeft weinig misverstand. Naast dat formeel aandoende taalgebruik zijn er ook andere dingen, zoals methodiek en heuristiek, die grotendeels de vreugde van het wiskundig bedrijf vormen, maar de meeste wiskundigen zijn het erover eens dat dat hulpmiddelen zijn voor het bereiken van het uiteindelijke doel: het bestudeerde wiskundige omderwerp compleet beredeneerd en gedocumenteerd weer te geven. Soms duurt het lang voordat een vakgebied in het gareel van de wiskundige taal is gebracht, in het bijzonder wanneer het wel een formele structuur heeft maar geen structuur die past in het gangbare wiskundige taalgebouw. Overigens moet steeds de waarschuwing worden geuit dat

formalisering wel nuttig en nodig, maar niet het enige is. In zekere zin kan men de geformaliseerde vorm wel het kerkhof van de wiskunde noemen, elk onderdeel netjes in zijn eigen graf. Maar ook ervaart men dat een formalisering nieuwe inzichten kan geven, door over die formalisering vragen te stellen. Vele onderwerpen uit de wiskunde, zoals bijv. algebra, zouden zonder formalisering niet van de grond gekomen zijn.

Een merkwaardige tekortkoming van de tegenwoordig gangbare wiskundige taal is het ontbreken van het notatiesysteem van de lambdacalculus, dat eigenlijk onmisbaar is voor de wiskunde van na het jaar 1800. Men heeft zich altijd maar beholpen met omschrijvingen in natuurlijke taal, wanneer een expliciet bekende afbeelding door een symbool moest worden voorgesteld. En in de gevallen waar daar een soort afkortingen in telegramstijl voor werden ingevoerd werkten die maar één stap diep, in tegenstelling tot alle andere wiskundige notaties die in iedere gewenste combinatie konden worden samengesteld. Dat het in de 19de eeuw niet tot goede functienotatie kwam is verklaarbaar: het is een grote worsteling geweest om van het functiebegrip van Euler naar dat van de moderne wiskunde over te gaan. Bij Euler lag het functiebegrip in de metataal, en processen waarbij stukken metataal naar de taal zelf moeten worden overgeheveld liggen gevoelig. Maar voor het feit dat nu, tegen het einde van de 20ste eeuw, de lambdanotaties nog steeds geen algemene ingang hebben gevonden, kan ik maar één redelijke verklaring bedenken: het komt doordat Bourbaki het niet gedaan heeft.

Naast de wiskundige taal heeft men metataal, of liever metatalen, waarin men óver de wiskundige taal spreekt. En verder gaan veel van de wiskundige gesprekken en geschriften in een andere toonaard over wanneer men bezig is met interpretatie van wat men gezegd heeft. In toegepaste wiskunde kan dat gaan over werkelijk bestaande objecten, maar in de zuivere wiskunde is er niet zoiets. De wiskundige objecten bestaan eenvoudig niet, hoe lang en hoe eenstemmig we er ook over praten. Dat wil zeggen, ze bestaan in de zin waarin ook Micky Mouse, O.B. Bommel en Sherlock Holmes bestaan, en niet zoals Churchill bestaan heeft. Wat in de zuivere wiskunde interpretatie heet, moet niet gezien worden als het leggen van verbanden met een werkelijkheid, maar als het leggen van verbanden tussen verschillende formele of half-formele systemen. Als dat verbandleggen formeel gebeurt noemen we het zoiets als metatheorie of modeltheorie, als het wat meer intuïtief gebeurt noemen we het interpretatie.

In de 20ste eeuw heeft zich het al eerder aangevangen proces voltrokken waarbij allerlei geheel verschillende delen van de wiskunde in een enkel taalsysteem worden gevangen. Laat ons dat proces integratie noemen. Men denkt misschien hierbij aan Descartes die de meetkunde tot algebra maakte. Daarmee was echter niet het bestaande meetkundige systeem ondergebracht maar er was een nieuw systeem gemaakt met geheel andere methoden. Over de kwaliteit van die nieuwe methoden behoeven we niet te twisten, maar de stap van

Descartes is niet wat we met integratie bedoelen. Eigenlijk is het voor het eerst Hilbert die rond de eeuwwisseling een axiomatische opzet voor de meetkunde gaf, en daarmee was het wiskundige taal geworden, los van plaatjes kijken en los van algebraïsering die wezensvreemd was aan het systeem van Euclides.

Tegenwoordig denkt men de integratie onder de knie te hebben met behulp van de Zermelo-Fraenkel-verzamelingstheorie. Daar kunnen we wat vraagtekens bij zetten. De gedachte is dat alle wiskundige objecten wel op de een of andere manier als verzamelingen kunnen worden gecodeerd, en dat het redeneren over die objecten dan is vervangen door redeneren over verzamelingen.

Tegen dit gezichtspunt zijn verschillende dingen in te brengen. Een ervan is dat men alleen maar zegt dat men het kan doen, en het in werkelijkheid niet doet. De meeste wiskundigen redeneren dan ook zonder al hun objecten als verzamelingen te beschouwen, en gebruiken verzamelingsterminologie alleen wanneer ze hun objecten aan het verzamelen zijn. Een ander bezwaar is dat men zijn wiskunde koppelt aan een volledig willekeurig coderingssysteem. Hoe onbelangrijk dat systeem is ziet men bij een gesprek tussen twee wiskundigen die hun begrip van het reële getal op verschillende manieren als verzamelingen hebben gecodeerd. Aangezien ze hebben geleerd elke referentie aan hun codering te vermijden, merken ze geen ogenblik dat ze het over totaal verschilende dingen hebben. Ze zouden het oneens worden over de voor beiden zinvolle vraag of 3 een element is van -1. Deze gang van zaken is niet erg bevredigend. Het zich beroepen op het universum van Zermelo-Fraenkel is niet wiskundig verwerpelijk, maar het is hypocriet.

Hilbert werkte voor de meetkunde heel anders. Hij deed niet de minste moeite om de fundamentele begrippen als punt en lijn in het paradijs van Cantor onder te brengen, en weigerde om lijnen als verzamelingen van punten op te vatten. Hij begon zijn meetkunde met het invoeren van primitieve typen: 'We nemen aan dat er dingen zijn die we punten noemen en dat er dingen zijn die we lijnen noemen'.

Hilberts axiomatiek voor de meetkunde is overigens nooit populair geworden. Het kwam in een tijd waarin de griekse meetkunde alleen nog maar in het schoolonderwijs leefde, en daarvoor was die axiomatisering te moeilijk. Het was ook de vraag of de bestaande situatie, waarin de meetkunde hier en daar een beroep op de aanschouwing deed, ook niet wat positiefs had: het ademde eigenlijk de geest van het zuivere redeneren in een stukje van de fysica.

Maar ik wil het nu over een deel van de griekse meetkunde hebben dat Hilbert niet axiomatiseerde, nl. het construeren met passer en lineaal. Sinds onheugelijke tijden had men constructies beschreven en de correctheid ervan serieus bewezen, maar een formele behandeling was er niet. Het valt onmiddellijk op hoe sterk dat constructiebedrijf op informatica lijkt. Het is natuurlijk verstandig de parallellen te bekijken, te bedenken langs welke wegen men het wel en wee van die constructies in de wiskundige taal zou kunnen integreren, en wat de informatica ervan zou kunnen leren.

Laat me enkele parallellen noemen. De constructie van de middelloodlijn van een puntenpaar wordt de eerste keer uit de doeken gedaan, maar bij

toepassing in een samengestelde constructie wordt die middelloodlijnconstructie niet telkens opnieuw beschreven. Precies wat men nu een subroutine noemt. De if-then-else komt overal voor waar men in gevallen moet splitsen, bijv. afhankelijk van de ligging van een reeds geconstrueerd punt ten opzichte van een reeds geconstrueerde cirkel. De while-do, waarbij een constructie moet worden herhaald totdat er aan de een of andere conditie voldaan is, komt bijv. voor als men een lijnstuk net zo lang met zichzelf moet verlengen totdat men over een gegegeven rechte heen is. In het bijzonder hebben we ermee te maken bij constructies op een gegeven eindig blad papier, waarbij men in een aantal stappen de schaal moet verkleinen totdat de constructie op het papier komt, en dan weer terug naar het resultaat, waarvan we van te voren al wisten dat het wèl op het papier zou komen. Dan zijn er de gevallen van non-determinisme, waar tijdens de constructie ergens willekeurige punten gekozen worden waarvan de willekeur geen invloed heeft op het eindresultaat. Ook de rol van variabelen met verschillende status, locale, globale, enz. is als bij de informatica. Verder is er wat betreft probleemspecificatie, constructiebeschrijving en correctheidsbewijs geen verschil met het 2000 jaar jongere zusje, de informatica. En de fase van het programmaontwerp was wat we bij de meetkunde de 'analyse' noemden.

Wat de meetkunde 'vies' maakt, ook bij de axiomatisering van Hilbert, is het moeten maken van onaangename uitzonderingen. Meetkundigen sprongen daar vaak nonchalant mee om. Als ze het hadden over een willekeurig punt en een willekeurige lijn dan bedoelden ze meestal dat die lijn ook weer niet zo willekeurig moest zijn dat die door het willekeurige punt ging. De gedachte dat de uitzonderingen de maat nul hadden zat er op de een of andere manier ingebakken. Men had het gevoel dat eerlijke willekeur ook allerlei onaangenaamheden zou vermijden die pas later in het betoog zouden opdoemen.

Voor de meetkundige theorie is de last met die uitzonderingen nog wel te verdragen, maar als men constructies met passer en lineaal moet gaan bespreken hopen de moeilijkheden zich op. Het was dus geen aantrekkelijk onderwerp om te formaliseren. Als ik zeg dat het goed is om over de formalisering van de meetkundige constructies na te denken dan probeer ik niet aan te raden om het werkelijk te gaan doen, maar alleen om te overwegen hoe het zou kunnen. Van de manier waarop we met meetkundige intuïtie omgaan valt ongetwijfeld nog wel wat te leren voor andere situaties.

De wereld van de meetkundige constructies laat ook erg vroege voorbeelden zien van een soort van inbedding van een systeem in een ander. Ik noem hier Mascheroni en Monge. Mascheroni liet zien dat alles wat met passer en lineaal geconstrueerd kan worden ook geconstrueerd kan worden met passer alleen. Om dat te bewijzen was het voldoende om te laten zien dat bij de grondconstructies van Euclides de lineaal kon worden thuisgelaten. De 'while-do' was daarbij niet van de lucht. En Monge liet zien dat hij voor de ruimtelijke constructies een model kon maken in termen van vlakke constructies alleen. Deze beschrijvende meetkunde van Monge was trouwens wat we tegenwoordig een subroutinebibliotheek zouden noemen.

Bij die meetkundige constructies komen we ook principiële moeilijkheden tegen die de informatica nauwelijks kent, doordat de informatici zich doorgaans tot eindige structuren kunnnen beperken. Het zou anders zijn wanneer computers echt met reële getallen zouden rekenen inplaats van met stompjes ervan. Als tijdens een meetkundige constructie in gevallen moet worden gesplitst afhankelijk van de ligging van een geconstrueerd punt binnen, buiten of op een eerder geconstrueerde cirkel, mogen we het dan als observeerbaar aannemen in welk van die drie gevallen we verkeren? Er zijn verschillende spelregels voor mogelijk. Een formele behandeling van de meetkundige constructies komt in geen geval onder die observeerbaarheid uit.

Maar hoe kunnen we het bedrijf van meetkundige constructies inclusief specificaties en correctheidsbewijzen samen met de daarbij benodigde meetkundige theorie onderbrengen in een geïntegreerd systeem? We denken hier aan een justificatiesysteem, waarin alles met onberispelijke strengheid wordt vastgelegd en geen enkel beroep op de ervaring van de bereidwillige lezer wordt gedaan.

Het zou bij integratie bijzonder onhandig zijn om alle betrokken meetkundige objecten, geconstrueerde objecten en constructies te coderen als verzamelingen. Moeten we ons de door Gauss bedachte constructie van de regelmatige 17-hoek voorstellen als een punt in de verzamelingshemel van Zermelo-Fraenkel? Veel van het wezenlijke zou gaan zitten in het coderingssysteem, en dat is geen onderwerp van gesprek in de gebruikelijke behandeling van de 17-hoek.

Een geheel andere opzet van het beschrijven van wiskunde wordt verkregen door het gebruik van typentheorie. Ik vermeldde al het gebruik van het introduceren van primitieve typen door Hilbert. Hoe de gehele wiskunde hiermee kan worden vastgelegd is aangetoond in het justificatiesysteem Automath, waarin men eigenlijk uitsluitend in termen van typering spreekt. Alle dingen waarover we willen praten hebben typen waarover we kunnen praten. Om er een gevoel voor te krijgen denke men aan de rol van de substantieven in onze westerse talen. We beschouwen zinnen van de vorm 'Piet is een man', en zeggen dat daarin 'man' het type van 'Piet' is. De zin wordt een typering genoemd. Om het wat meer op wiskundige symboliek te laten lijken zullen we het zinsdeel 'is een' door een dubbele punt vervangen: 'Piet : man'. In onze natuurlijke taal is er niet zoiets als het eenduidige type van Piet, want Piet kan ook een schilder zijn of een moslim. Maar we stellen ons de in de wiskunde voorkomende situatie voor waarbij er een oertype is waarmee Piet oorspronkelijk ten tonele is gevoerd, hier bijv. 'persoon', en dat alle andere uitspraken met 'is een' daarvan afgeleid zijn met behulp van Piets speciale eigenschappen. We zullen verder uitsluitend aan typeren in termen van oertypen denken. In de wiskunde zijn de namen niet Piet of Jan, maar in het algemeen meer samengestelde expressies uit de lambdacalculus. Bij de typen gaan de gedachten allereerst uit naar expressies die uit een enkel woord of enkel symbool bestaan, zoals een symbool voor 'natuurlijk getal'. Maar wanneer we later ook bewijsklassesn als typen gaan toelaten zal blijken dat het gewenst is om voor de typen algemene lambda-expressies te nemen.

De lambda-expressies zijn zelf ook opgebouwd met getypeerde variabelen, en de typen van die variabelen worden bij de lambda's in de formules vermeld. De lambda-expressies zijn redelijk eenvoudige taalkundige objecten. Laten we eerst kijken naar 'totale' lambda-expressies, die geen vrije variabelen hebben. Het zijn binaire bomen waarin de binaire knopen voorzien zijn van een symbool T (wat staat voor 'typering') of een symbool A (voor 'applicatie'). De eindpunten in de boom dragen óf een symbool τ (wat staat voor 'supertype') óf geen symbool, maar in dat geval gaat er van dat eindpunt een pijl naar een T-knoop, en wel zo dat die T-knoop ligt op het pad van het eindpunt naar de wortel van de boom, met dien verstande dat de pijl bij die T-knoop van de rechterkant af komt. In de interpretatie naar gewone wiskundige formules wordt de T-knoop een getypeerde lambda, het eindpunt een instantie van de in die lambda ingevoerde gebonden variabele, en de sub-boom die links van de T-knoop uitgroeit wordt het type van de variabele.

Wanneer men van deze boom een sub-boom beschouwt, kunnen daarin eindpunten voorkomen waarvan de pijlen naar een T buiten de sub-boom wijzen; die spelen dan de rol van vrije variabelen. Ook de typeringsoperatie is eenvoudig te beschrijven in termen van de bomen. Als een boom in het uiterst rechts gelegen eindpunt een τ heeft dan zeggen we dat de boom de graad 0 heeft, en we kennen aan de boom geen type toe. In alle andere gevallen is er wel een type, en dat wordt verkregen door de uiterst rechts staande variabele te vervangen door het type ervan (en dat vindt men door de pijl te volgen naar de T en daarvan de linker sub-boom te nemen). We krijgen nu een verdere indeling in graden: bomen waarvan het type de graad 0 heeft geven we de graad 1, bomen waarvan het type de graad 1 heeft geven we de graad 2, enz.

Het Automath-systeem heeft laten zien hoe men met zulke lambda-bomen de wiskundige taal geheel in de vingers krijgt, en voor definitieve machinale verificatie klaar maakt. Het is niet alleen zo dat de in de gewone wiskundige taal voorkomende expressies als lambda-bomen kunnen worden beschouwd: ook een gehele wiskundige theorie in een dik boek is een lambda-boom. En dat is niet alleen een theoretische maar ook een praktische mogelijkheid: presentatie in Automath hoeft nooit wezenlijk langer te zijn dan geheel volledige presentatie met gangbare middelen.

Heel belangrijk is dat in deze typentheoretische presentatie objecten en bewijzen geheel analoog behandeld worden. In feite gaat dit terug op een idee van Heyting van vóór 1930. Die gedachte was dat het bewijs van een implicatie kan worden gezien als een afbeelding die aan elk bewijs van het linkerlid een bewijs van het rechterlid toevoegt. Bewijzen als argumenten en als waarden van afbeeldingen dus. En evenals bij gewone wiskundige objecten kan het voorkomen dat het domein van de afbeelding leeg is. Om voort te bouwen op die gedachte van Heyting moeten aan bewijzen namen gegeven worden, en in het algemeen kunnen dat weer samengestelde namen in de vorm van lambda-expressies zijn. Dit wijkt af van de gangbare stijl waarin men zich voorstelt dat alleen aan de in een stelling bewezen propositie een formele naam gegeven wordt, terwijl aan de stelling als geheel een informele naam wordt toebedacht. Met die informele naam wordt echter niet als wiskundig symbool

gemanipuleerd. Met de namen voor bewijzen doen we dat wèl. Als er in de propositie van een stelling een parameter voorkomt zal die in het algemeen ook in de bewijsexpressie staan. Substitueert men nu in de expressie voor dat bewijs een speciale waarde voor die parameter dan krijgt men een bewijs voor de propositie die ontstaat door diezelfde substitutie op de oorspronkelijke propositie los te laten. Aan deze eenvoudige opmerking ziet men al iets van het manipuleren met bewijsexpressies. Een andere eenvoudige bewerking krijgt men wanneer in de oorspronkelijke stelling een onderstelling stond, die voor het geval van de een of andere specialisatie echt bewezen kan worden. Dan is er sprake van een soort substitutie die in de wiskunde wel altijd werd gevoeld maar nooit werd genoteerd. Wát er gesubstitueerd moet worden is duidelijk: het is het bewijs dat we in het speciale geval voor de onderstelling hebben. Maar is er een variabele die door dit bewijs moet worden vervangen? Die is er als we de onderstelling van de oorspronkelijke stelling opvatten als de introductie van een bewijsvariabele. In plaats van te zeggen 'onderstel $a > b$' zeggen we 'zij x een bewijs voor $a > b$'.

Als we deze gedachten vervolgen komt ervan alles en nog wat los. Stellingen krijgen dezelfde syntactische vorm als de invoering van afkortende namen voor objecten (en de afkortende naam correspondeeert met het bewijs van de stelling). Een axioma krijgt dezelfde vorm als het scheppen van een primitief object, en dat object gaat dan gehanteerd worden alsof het een bewijs voor het axioma zou zijn. En het bewijs van een implicatie van een propositie A naar een propositie B krijgt inderdaad dezelfde syntactische vorm als een afbeelding van de klasse van alle bewijzen voor A naar de klasse van alle bewijzen voor B, precies wat Heyting bedoelde. Het type van die afbeelding is de klasse van alle bewijzen voor de implicatie. Alles wat we gewend zijn geweest in de gangbare presentatie van de wiskunde gaat even schuiven, maar direct daarna komen we in een stabiele situatie terecht waarin alles netjes een vaste plaats heeft. Dat is dan de typentheoretische opvatting over de wiskunde.

In dit typentheoretische systeem zijn logica en wiskunde innig verweven. Logische stellingen kunnen worden afgeleid, de resultaten kunnen gebruikt worden bij wiskundige bewijzen, en bij de afleiding van stellingen uit de logica kan van wiskundige verworvenheden gebruik worden gemaakt. Het lijkt duidelijk op hetgeen we voor meetkundige constructies en computerprogramma's willen. De wiskundige objecten, bewijzen, constructies en programma's worden dingen van eenzelfde niveau. Men kan ook nog verder gaan met categorieën van dingen die men hanteert. Men kan bijv. in eenzelfde boek bewijzen geven met klassieke logica, met intuïtionistische, met negatievrije, en dan kunnen soms de resultaten bij het ene soort legitiem gebruikt worden bij het andere. Wat het taalgebouw verenigt is dat voor elk van deze categorieën hetzelfde substitutiemechanisme en dezelfde lambdacalculus werkt. Substitutie en lambda-calculus werken ook onafhankelijk van de graden van de expressies. Die simpele lambda-bomen werpen dus rijke vruchten af.

In vele logische systemen wordt het proces van het afkorten van expressies alleen informeel beschouwd, alsof afkortingen metataal zijn. In Automath vormen ze een wezenlijk bestanddeel van de taal, en ze zijn dan ook direct aan te

wijzen als onderdelen van de lambda-boom die een compleet boek representeert. Hiermee is exponentiële explosie voorkomen, en het praktisch werken in Automath is daardoor bij iedere soort van wiskundig materiaal redelijk goed mogelijk geworden. Een duidelijk verschil met vele andere systemen is overigens ook dat Automath zich geheel beperkt tot wat men de prelogica kan noemen: de kunst van het manipuleren met logisch en wiskundig materiaal. Wat men gewoonlijk logica noemt, zoals de invoering en eigenschappen van ontkenning, conjunctie, enz., staat niet in de taaldefinitie. Het wordt aan de gebruiker overgelaten voor de logica zelf een vorm te kiezen in de de boeken die in de taal worden geschreven.

Ik heb de overtuiging dat in zulk een systeem van typentheorie en lambdacalculus de integratie van grote stukken formele wetenschap pas goed tot stand kan komen, en wel op een wijze die aangepast is aan onze gewone manier van spreken. Daarmee krijgen we de mogelijkheden om het begrip 'wiskunde' te verruimen. Wiskunde is alles wat met wiskundige taal behandeld kan worden, plus de metataal, de interpretaties, de methodiek en de heuristiek eromheen. En ik heb de overtuiging dat enig inzicht in dat taalgebouw werkelijk helpt om de aard van de wiskunde beter te doorgronden, ook in het onderwijs aan beginnelingen.

Eenmaal beschikkende over een systeem dat integratie mogelijk maakt, kunnen we erover nadenken hoe we de wereld van de meetkundige constructies, en de wereld van de computerprogramma's kunnen beschrijven. En die werelden kunnen van elkaar leren. Om daar even nader op in te gaan kijken we eens naar de verschillende wegen die kunnen worden ingeslagen bij het weergeven van meetkundige constructies in een formele taal. Een voor de hand liggende mogelijkheid is om zich voor te stellen dat we de meetkundige constructies uitvoeren in het meetkundige vlak zelf. Het construeren van een rechte is dan zoiets als het 'hebben' van die rechte. Het feit dat we de rechte in de meetkundige taal konden vastleggen betekent nog geen constructie. Het hebben van de rechte wordt gemotiveerd op zekere regels die gebruik maken van het feit dat we enkele andere punten en lijnen reeds hebben. Wat er nu in de taal wordt opgeschreven is een expressie voor een bewijs van het hebben. In de metataal kan men zeggen dat daarmee een constructie is weergegeven, in de taal zelf gebeurt dat niet.

Een tweede systeem beschouwt naast het abstracte meetkundige vlak zoiets als een fysisch vlak, een stuk papier. Maar we praten in feite natuurlijk alleen over een mathematisch model van dat stuk papier. De spelregels worden zo opgesteld dat alles wat we op dat papier kunnen benoemen kan worden geïnterpreteerd als te zijn geconstrueerd met passer en lineaal. De expressies waarmee we in de taal werken stellen nu geen bewijzen voor maar figuren op het papier. In de metataal kan men zeggen dat de opbouw van die expressies laat zien hoe de constructie is uitgevoerd.

In beide systemen is het mogelijk constructies te beschrijven en te bewijzen dat ze het beoogde doel bereiken. Toch wil een wiskundige méér. Hij wil bijvoorbeeld als stelling uitspreken dat er geen constructie met passer en

lineaal bestaat die een kubus levert die het dubbele volume heeft van een gegeven kubus. Bij dat voorbeeld ziet men misschien pas echt goed wat ik met een geïntegreerd systeem bedoel. Er komen algebra, meetkunde, logica en constructies bij te pas, en alles moet tot één sluitend betoog zijn verweven. Wanneer men uitspraken over de onmogelijkheid van de uitvoering van zekere constructieopdrachten wil doen, dan is het nodig om over constructies zèlf te praten. Dat wordt dan iets waarbij de constructies geen gewone wiskundige objecten zijn (dat zou het geval zijn als men ze met verzamelingen zou coderen), maar een derde soort dingen in de klasse waarin we al de categorieën 'object' en 'bewijs' hebben ondergebracht. En weer kan men profiteren van hetzelfde substitutiemechanisme. Technisch is het bespreken van constructies dan nog steeds niet eenvoudig; we kunnen niet zo maar over het begrip constructie praten, maar moeten daar steeds de soort van de gegevens en de soort van het resultaat bij vermelden.

In systemen die opgezet zijn om de semantiek van computerprogramma's te beschrijven kan men onmiddellijk parallellen ontdekken met wat ik hier over meetkundige constructies schetste.

Wanneer we wiskunde ruim opvatten als alles wat in zulke geïntegreerde systemen te beschrijven is, dan is men duidelijk op weg om informatica tot deel van de wiskunde te verklaren. Met natuurlijk de reserve dat er veel toegepaste informatica is, net zoals er allerlei vormen van toepassingen van andere delen van de wiskunde zijn.

Maar laten we eens proberen verschillen waar te nemen tussen wiskunde en informatica. Die zijn hoofdzakelijk van historische en sociale aard.

Wat we tegenwoordig informatica noemen heeft vorm gekregen in de periode van 1945 tot 1965. In die tijd waren er weinig wiskundigen bij betrokken. Het vak is ontwikkeld door fysici en ingenieurs, en we kunnen daar niet dankbaar genoeg voor zijn. Eigenlijk was de wiskunde het enige gebied in de beta-wereld dat geen belang bij computers had. Computers waren er voor numerieke berekeningen op zeer grote schaal, en wie eraan werkte kwam uit een sfeer waar men behoefte had aan de uitkomsten. Al heel vroeg werd in allerlei vakken de richting van onderzoek en ontwikkeling sterk beïnvloed door het feit dat er zo snel en zo veel gerekend kon worden. De wiskunde had geen problemen waar zulke behoeften sterk genoeg waren en genoeg aanzien genoten om de relatief hoge kosten van veel rekenwerk te motiveren. Bijgevolg bleven bijna overal de wiskundigen afzijdig van de snelle ontwikkelingen op computergebied. Een instituut als het Mathematisch Centrum in Amsterdam, waar wiskunde en informatica samen probeerden te leven, was een witte raaf.

De ontwikkeling van de informatica als wiskundige wetenschap begon toen grote onontwarbare programma's in machinecode de behoefte deden ontstaan aan uitdrukkingsmiddelen om de programmeurs onafhankelijk te maken van de speciale computer met speciale machinecode waarmee daar ter plaatse op dat ogenblik gewerkt moest worden. Dat gaf aanleiding tot de ontwikkeling

van hogere programmeertalen, en dat opende een nieuwe wereld. Eerst moesten er compilers gemaakt worden die voor speciale machines de programmeertaal in machinecode konden omzetten. Vervolgens kon er dank zij die talen beter over de heuristiek van het ontwikkelen van programma's worden nagedacht, en men kon gaan beginnen aan een wetenschappelijke studie van semantiek, wat neerkomt op het bewijzen dat de uitkomsten van de executie van een programma aan de van te voren gestelde specificatie voldoen.

Inmiddels was het aantal informatici al groot geworden, met veel onderlinge internationale contacten. Maar in de beginperiode was er weinig contact met wiskundigen, en zelfs ook weinig met de beoefenaren van de grondslagen der wiskunde. Daardoor is het te begrijpen dat er zich in die informaticawereld terminologieën en opvattingen instelden die verschilden van de in de wiskundige wereld gangbare. Zo was lange tijd het formele denken over programma's zeer sterk gebonden aan de syntax ervan, en er werd te gemakkelijk gedacht over de gelijkenis van de in de programmeertaal voorkomende functie-expressies met de overeenkomstige formules in de wiskundige taal. De wiskundige had nooit de plicht gehad zich met de syntax van zijn taal bezig te houden, en hier moest het ineens wel. Een andere kwestie bij programmeertalen was de rol van zekere projectieoperatoren die zo nauw gekoppeld waren aan de daarmee corresponderende wiskundige variabelen dat men ze ook maar variabelen ging noemen. In het onderwijs wekt dat heden ten dage nog verwarring. Het doet wat denken aan het gebruik van het woord variabele in de thermodynamica, waar men bijv. de temperatuur T een variabele noemt, omdat de temperatuur af en toe verandert in de tijd. In de wiskunde heeft het begrip variabele een geheel andere betekenis. Er is geen enkele reden om ervan af te zien het wiskundige taalgebruik ook bij wiskundige beschouwingen in informatica voort te zetten. We kunnen ons afvragen of die misverstanden over variabelen niet toe te schrijven zijn aan het grote aantal fysici onder de werkers van het eerste uur in de informatica.

Vaak zijn dingen die heel goed binnen de bestaande wiskundige formalismen passen door informatici opgedist als iets heel afzonderlijks. De telegramstijl van BNF voor het genereren van grammatica's komt doorgaans uit de hemel vallen. Dat hangt dan nog samen met het gebruik van de term recursieve definitie voor dingen die helemaal geen definities zijn. De term recursieve specificatie zou aanvaardbaarder geweest zijn.

Een jaar of wat geleden sprak ik met een informaticus die zei dat informatica niet alleen anders was dan wiskunde, maar ook moeilijker en diepzinniger, omdat de informatica met de tijd te maken had en de wiskunde alleen statisch was. In mijn verbouwereerdheid kon ik alleen mompelen dat ik dacht dat Newton eens rekening had gehouden met een tijdvariabele bij berekeningen over het wel en wee van het zonnestelsel, en zelfs naar de tijd had gedifferentieerd, maar pas later bedacht ik wat de werkelijke achtergrond van zijn opmerking was. Zijn methodologie schreef voor dat er over tijd niet mocht worden gesproken. En er werd natuurlijk wel in het achterhoofd over gedacht, zodat het spreekverbod de intuïtie geweld aan deed.

Terugkomende op mijn opmerkingen over de sterke concentratie op syntax

merk ik op dat de wiskundigen zich bij de meetkundige constructies nooit hadden bekommerd om het coderen van hun constructieopdrachten. Nu we tegenwoordig zo leuk op schermpjes kunnen tekenen komen er taaltjes als LOGO die de computer duidelijk moeten maken welke constructies we bedoelen, met alle beperkingen van de eindige resolutie en eindige grootte van het scherm. Het zou belachelijk zijn het programmaschrijven in LOGO als het wezen van het meetkundige construeren te willen zien. Toch is dat eigenlijk wat er in de begintijd van het programmeren gebeurde: de programmatekst was het programma, en voor de beschrijving van de algoritme zoals een wiskundige die zou geven was geen taalapparaat beschikbaar.

Afgezien van enkele eenvoudige gevallen die men gewoonlijk in leerboeken aantreft is de ontwikkeling van een programma tot nu toe bijna overal een zaak van intuïtie. Heel vaak is er op het moment dat men begint te programmeren een chaos van verlangens, die, zo die al worden geformuleerd voordat men aan het programma begint, tot een onoverzichtelijke specificatie aanleiding geven. Maar ook als we aannemen dat we het over een programmeerprobleem hebben waarbij de specificatie eenvoudig en doorzichtig is, dan kan het programma nog moeilijk zijn. Dat is de normale situatie bij de meetkundige constructies. Uit de meetkunde hebben we de ervaring dat er bij een eenvoudig gestelde specificatie geen enkele verwachting van een simpele oplossing behoeft te worden gekoesterd; het hele geheim zit in de probleemanalyse. Het bewijs van de correctheid van de constructie is in de meetkunde daarentegen doorgaans heel direct. Men volgt eenvoudig de constructiestappen en het staat er. Meestal, maar het hoeft natuurlijk niet. Stelt U zich maar voor dat we een punt moeten construeren volgens zekere specificaties. Met een diepzinnige meetkundige beschouwing die niets met passers en linealen te maken heeft laten we zien dat het gezochte punt het zwaartepunt is van een driehoek die al bekend is. De constructie is nu heel eenvoudig, maar onthult niets van het bewijs uit die diepzinnige beschouwing. In de informatica kunnen we zulke voorbeelden ook aangeven, maar meestal staan we voor het feit dat het bedenken van het programma een kwestie is van een goede greep en het bewijs daaruit min of meer routinematig volgt. Tenminste, als men zich een goede programmeermethodologie heeft aangemeten, en het is geen slechte methodologie om bij de ontwikkeling van het programma al rekening te houden met de latere plicht tot een bewijs. En als ik het woord routinematig gebruik wil ik niet vergeten dat het erg lastig kan zijn om tot een goede routine te komen. Heeft men die eenmaal, dan kan men bergen verzetten.

Informatici, en zelfs de allereenvoudigste programmeurs werken geregeld met hun intuïtie bij het verbinden van verschillende formalismen, zoals het verbinden van de taal waarin de specificatie staat met de taal van het programma en de taal waarmee men over het programma spreekt. Willen we dat intuïtieve hard maken, en beveiligen tegen onverwachte denkfoutjes, dan hebben we een geïntegreerd systeem nodig waarin de verschillende formalismen door een enkel superformalisme zijn overkoepeld.

Overigens ligt de relatie tusen het formele en het intuïtieve in de informatica niet wezenlijk anders dan in de wiskunde. Binnen de wiskunde is de rol van de intuïtie niet in alle deelgebieden dezelfde, en niet bij alle beoefenaren dezelfde. Belangrijk hierbij is in hoeverre een gebied geïntegreerd is. Maar zelfs in gebieden die geheel geformaliseerd zijn wordt er niet altijd gewerkt in termen van die formalisatie.

Het was niet alleen in de meetkunde dat de wiskundigen ervaring met algoritmen hadden. Ook de getallentheorie kende er vele. De algoritmen voor optelling en vermenigvuldiging waren te kinderachtig om er theorie aan te wijden, maar die voor de worteltrekking rook al wat naar numerieke wiskunde. Er werd pas semantiek bedreven bij de beroemde algoritme van Euclides voor de grootste gemene deler van twee gehele getallen. In 1935 moest ik bij het vak Rekenkunde voor de akte Wiskunde K1 het bewijs van de goede afloop van die algoritme leveren. Groot was mijn verbazing toen ik ik 1970 op een internationaal congres meemaakte dat een gerenommeerd Amerikaans informaticus in een hoofdvoordracht dat bewijs uitvoerig uiteenzette. Dit schetst de grote afstand die de wereld van de informatici tot die van de wiskundigen had. En het vergelijken van computerprogramma's met de aloude meetkundige constructies ben ik nog nooit in een boek over informatica tegengekomen.

Natuurlijk zijn er bij de computerprogrammering kwesties die bij de rekenkundige en meetkundige algoritmen niet voorkomen. Bij computerprogrammeren wordt men veel vaker dan bij de meetkundige constructies geconfronteerd met de noodzaak geheugengebruik en rekentijd binnen de perken te houden; bij de getaltheoretische constructies behoefde wel op rekentijd maar niet op geheugenruimte gekeken te worden.

Als we wiskunde en informatica in groter verband bekijken zien we dat het totaal van het in omloop zijnde wiskundig materiaal tamelijk netjes gestructureerd is, en dat zulks voor de informatica veel minder geldt. De wiskunde lijkt op een keurig park, met bloementuin, speelweide en diverse groentenveldjes (waarvan sommige knollentuinen zijn), de informatica heeft meer weg van een weelderig oerwoud met uitbundige doch weinig georganiseerde flora en fauna. Historisch is dat heel begrijpelijk. Toen de wiskunde haar huidige vorm kreeg waren er enkele honderden wiskundige onderzoekers op aarde, waarvan er maar enkele tientallen echt meetelden. Men kende en men begreep elkaar. Daardoor is verklaarbaar dat er een duidelijke basis van wiskundige kennis was waarover iedereen beschikte. Dat basispakket was rijk en sterk genoeg om in de loop van de tijd veranderingen te assimileren. En veel veranderingen waren er niet, misschien maar één grote doorbraak per generatie. De principes van de heldere en onwrikbare wiskundige taal werden zo algemeen erkend dat ze niet behoefden te worden bestudeerd, men leerde het vanzelf wel. En lang nadat de stijl was vastgelegd bleef de wiskunde bijeen. Men hoeft geen groot voorstander van Bourbaki te zijn om te erkennen dat ook die weer een bewonderenswaardige unificatie tot stand heeft gebracht (hoewel jammer genoeg zonder lambdacalculus).

De informatica daarentegen groeide op toen de wetenschappelijke wereld al een enorme schaalvergroting had ondergaan. Daar kwam nog bovenop dat de informatica te maken had met een computermarkt waar vele en vele miljarden in omgingen, en waar zelfs op basis van de afvallende kruimpjes nog wetenschappelijk kon worden gewerkt. Voor een vak dat zijn vorm nog niet gevonden heeft is die situatie wel plezierig, maar niet erg geschikt om die vorm alsnog te vinden. Het aantal onderzoekers dat iets wezenlijks had bij te dragen liep in de duizenden, in centra overal op de wereld. Allerwegen werd steun van overheden ondervonden, overal werden opleidingen op touw gezet, maar zonder sterke gemeenschappelijke basis. Natuurlijk zal er op den duur wel een soort stabilisatie ontstaan, maar dat kan nog even duren.

De fysica is een lappendeken van naast elkaar staande wiskundige modellen; binnen elke lap zijn de relaties sterk, maar onderling hangen de lapjes maar met een paar draadjes aan elkaar. Integratie ontbreekt, en de fysici, althans die van na Einstein, voelen dat ook niet als een gemis. Geen wonder dat bij zoveel inbreng van fysici er geen behoefte werd gevoeld aan integratie in de informatica.

Het feit dat informatica nog een jonge wetenschap is ziet men o.a. aan een overgrote ijver om steeds weer nieuwe formalismen te introduceren en daarvan wonderen te verwachten. Tekenend is de neiging om voor allerlei speciale situaties te proberen met nieuwe logica's uit de bus te komen. Wat men dan nodig heeft is helemaal geen nieuwe logica maar een nieuw algebraïsch apparaat dat aan de probleemstelling is aangepast. Het is bijzonder verwarrend om te proberen dat in de logica te trekken. Het doet me denken aan fysici die zeggen dat de quantenmechanica een andere logica nodig heeft dan de klassieke.

We richten ons vervolgens op de vraag inhoeverre de computer de wiskunde zal veranderen. Dat computers worden ingeschakeld waar dat maar mogelijk is, ligt voor de hand, en daar zal ik het nu niet over hebben. Men hoeft hiervoor ook niet in de toekomst te kijken. En van iedereen die wiskunde toepast is de werksfeer er geweldig door beïnvloed. Maar een wiskundige doet de dingen niet alleen omdat ze gevraagd worden. Er is vooral de neiging om de eigen smaak te volgen, hoe die ook bewust of onbewust bestuurd wordt door de mode. En, wat misschien nog belangrijker is, een wiskundige, en dat geldt voor iedere wetenschapper, kiest zijn onderwerpen zo dat hij er kans door maakt op een succesvolle carrière. Laten we dat ook maar tot de smaak rekenen.

De computer zal zonder enige twijfel invloed hebben op de smaak van de wiskundige, zowel in positieve als negatieve zin. Gebieden waar de computer alles doet, waar de mens de computer niet kan helpen en er niet mee kan concurreren, zullen geen enkele wiskundige blijvend boeien. Maar alle onderwerpen waarin de computer een substantiële en de mens een niet-triviale inbreng heeft, zullen op wiskundigen een grote aantrekkingskracht uitoefenen. Velen zullen het meest worden aangetrokken door gevallen waarin de computer helpt materiaal te verzamelen dat leidt tot ontdekkingen die de wiskundige vervolgens bewijst zonder computerhulp, behalve dan wellicht hulp van administratieve aard.

Het feit dat er plotseling allerwegen belangstelling ontstaat voor de vraag wat nu precies een wiskundig bewijs is, schijnt ook een effect van de computer te zijn. Het komt niet van de kant van degenen die computers gebruiken om menselijke bewijzen op correctheid te controleren, of die computers bewijzen laten vinden, al dan niet in samenwerking met de mens. Bij deze lieden moet men niet aankomen met twijfel over wat onder een bewijs moet worden verstaan. Een bewijs is een sluitende argumentatie in een van te voren precies vastgelegde taal. Voor de regels van die taal kan men natuurlijk nog verschillende keuzen maken, evenals over het totaal van de gebruikte axioma's. Iedereen weet dat het van zulke dingen afhangt wat een bewijs is. Voor deze mensen is dat alles wat er te zeggen is, en het is geheel vrij van emoties.

De plotseling weer opgekomen vraag naar wat een bewijs is, is vaak ingegeven door iets heel anders, dat ook met computers te maken heeft: de oplossing, althans de vermeende oplossing, van het vierkleurenprobleem.

Het vierkleurenprobleem was opgediend als probleem voor het maken van atlassen waarbij elk land egaal, en belendende landen verschillend moesten zijn gekleurd. Nadat was vastgesteld dat dit met vier kleuren kon in alle gevallen die ooit in een atlas zouden kunnen voorkomen, was de behoefte der atlasmakers natuurlijk bevredigd, maar de wiskundige wereld ging voort met niet geslaagde pogingen om het algemeen te bewijzen en wél geslaagde om te laten zien dat vorige vermeende bewijzen fout waren. Het ging door totdat men inzag dat het bedriegelijk lastig was, en dat tegelijk de lastigheid het enige interessante vormde. Het was niet zoals het vermoeden van Riemann dat allerlei interessante consequenties had; het vierkleurenvermoeden had er helemaal geen. Omstreeks 1950 kon iemand die alle energie op het vierkleurenprobleem wilde gooien misschien wel aanspraak maken op vriendelijke verzorging in een liefderijk tehuis, maar in geen geval op wetenschapsfondsen voor ondersteuning van zijn project. Maar in 1976 wisten Appel en Haken het met behulp van een enorme hoeveelheid computerhulp te brengen tot een sluitend bewijs. Net als in de meeste menselijke bewijzen zaten er hier en daar een paar haken en ogen die wel weer in orde gebracht werden. Of de laatste fouten er nu uitgehaald zijn is, naar ik meen, niet bekend. Is dit nu een bewijs van het vierkleurenprobleem?

Het gaat natuurlijk over twee geheel verschillende dingen. Het eerste is dat de bij het proces betrokken mensen iets over het hoofd gezien kunnen hebben. Een klein programmeerfoutje bijvoorbeeld. Dat is dan heel vervelend, maar geen aanleiding tot het bijstellen van ons begrip van wat nu een bewijs is. Het tweede is de vraag: als nu al zulke fouten eruit zijn, ongeacht hoe men dat constateert, is het dàn een bewijs? Natuurlijk moet men eisen dat er bewezen wordt dat de semantiek van het gebruikte computerprogramma met de specificaties overeenstemt; pas als dat tenslotte gebeurd zou zijn is de zaak voor serieuze discussie vatbaar. Maar de akeligheid is dat het bewijs zich dan nog uitstrekt over twee verschillende stukken van een lappendeken die met intuïtieve draden aan elkaar gezet zijn. Moet men die intuïtieve draden

erkennen? De doorgewinterde formalisten zullen eisen dat al het betrokken materiaal is opgenomen in een geïntegreerd systeem. Maar hebben we dan een bewijs? Dat wel, maar niet van het vierkleurenprobleem. Aannemende dat we een computer gebruiken die zich gedraagt zoals in keiharde specificaties is vastgelegd, is er een bewijs verkregen van de volgende implicatie: als die computer uiteindelijk het vierkleurenvermoeden bevestigt, dan is het vierkleurenvermoeden correct. De moeilijkheid zit niet in het feit dat we een fysische computer hebben: als we die vervangen door een mathematisch model zitten we nog met eenzelfde kwestie. In plaats van een bewijs in de wiskundige taal hebben we een bewijs in een metataal gemaakt.

Overigens is de stof die het bewijs van het vierkleurenprobleem deed opwaaien meer het gevolg van het feit dat het werk niet door iedereen vertrouwd werd dan van kritiek op het meewerken van een computer. Dat laatste was al heel vaak geaccepteerd, al in de tijd van de mechanische rekenmachines. Stelt U zich voor dat iemand zonder computer bewezen heeft dat elk even getal groter dan 10000000 de som is van twee priemgetallen, en dat een computer een lijst heeft gemaakt met splitsingen van alle even getallen tussen 2 en 10000000. Men zal dan vrij algemeen zeggen dat daarmee bewezen is dat elk even getal groter dan 2 een som van twee priemgetallen is. Het verschil met het geval van het vierkleurenprobleem is dat zo goed als iedereen in staat is met zijn eigen computer en eigen programmeersysteem de zaak nog eens over te doen. De kwestie van de lappendeken blijft bestaan. Bij het geval van die priemgetallen ziet U ook hoe men de lappendeken kan vermijden. Het is niet zo moeilijk om een computer niet alleen op te dragen de controle te verrichten, in het bijzonder van het feit dat de componenten werkelijk altijd priemgetallen zijn, maar die computer aan de hand van die berekeningen ook een bewijs te laten schrijven in termen van de gewone predikatencalculus, zonder enig beroep op rekenresultaten, van de stelling dat ieder even getal tussen 2 en 10000000 een som van twee priemgetallen is. Of het door de computer gemaakte bewijs nu correct is of niet, is niet bewezen (wel erg waarschijnlijk), maar het ligt in elk geval vast welk bewijs bedoeld wordt, en dat het een 'gewoon' bewijs is. En de mensen die zelf een programma hebben dat een computer wiskundige bewijzen laat controleren kunnen direct aan de gang gaan.

Ook als een waanzinnig lang bewijs computervrij in gewone wiskundige stijl gepresenteerd is, zullen er mensen zijn die betwijfelen of dat wel een bewijs genoemd kan worden. Een bevredigende definitie van een acceptabel bewijs zullen zij niet kunnen geven, want zoiets als begrijpelijkheid is niet additief. Erg belangrijk is ook of een bewijs modulair is. Een lang bewijs kan al acceptabeler zijn wanneer steeksgroepsgewijs modulen kunnen worden gekozen die snel kunnen worden gecontroleerd. De controleur kan er dan ook beter de tijd voor nemen, stukken delegeren, enz.

De historisch ontstane kloof tussen wiskunde en informatica wordt vergroot door het betreurenswaardige feit dat er wiskundigen zijn die zich wel eens denigrerend over informatica hebben uitgelaten. Toegegeven, wat binnen hun gezichtsveld komt is wiskundig niet altijd even diepzinnig. Maar de pedanten

onder de wiskundigen moeten bedenken dat niet alles binnen hun gezichtsveld komt. Pedanterie is een houding die het meest voorkomt bij mensen die maar binnen één gezichtshoek kunnen kijken, en werkt sterk polariserend tegenover andersdenkenden. Pedanterie is bijzonder geschikt om het gevoel van bijeenhoren in de eigen groep te versterken, maar heeft naar buiten weinig overtuigingskracht, behalve de overtuigingskracht die iedereen tegenover een ondeskundige buitenwereld heeft wanneer hij zich zeker van zijn zaak toont, ook als hij dat niet is.

Natuurlijk is ook in de wereld van de informatica niet alles ideaal. Door de explosieve groei van het computerwezen zijn groepen zonder lange traditie wel wat erg snel tot grote bloei gekomen. Snelle groei gevolgd door afremming geeft een sterke blijvende positie aan degenen die er het eerste waren, en weinig kansen voor degenen die daarna komen, al zijn die soms ook duidelijk beter.

Hoe voeden we onze wiskundigen op in de nieuwe wereld waarin de computer zo'n grote invloed en aantrekkingskracht heeft? Bij de opleiding tot verschillende soorten van toegepaste wiskunde ligt dat betrekkelijk duidelijk, maar bij de opleiding van zuivere wiskundigen is de relatie tot computers een probleem. Te proberen ze verre van computers te houden is natuurlijk heel dom. Met de kop in het zand komt men niet ver. Proberen overal waar dat maar kan de relatie tussen wiskunde en computers te leggen zou weer leiden tot procrustisatie, waarbij alles zou worden afgesneden wat geen relatie met computers heeft. En daarbij zouden misschien niet alleen de naar buiten stekende voeten maar vooral de naar buiten stekende hoofden het slachtoffer worden. Ook zou het fout zijn de wiskundeopleiding speciaal met abstracte delen van de theoretische informatica te infiltreren. Wat de aanstaande wiskundigen vooral moeten leren is om computers te gebruiken waar het nodig is en er verder zoveel mogelijk van af te blijven. En vooral ook de docenten dienen deze instelling te hebben.

Dat wiskunde en informatica aan de universiteiten bij elkaar blijven lijkt mij van levensbelang voor beide. Maar om bij elkaar te kunnen blijven is er een gemeenschappelijke basis in de opleiding nodig.

De calculus staat jammer genoeg op de tocht als onderwerp dat studenten in wiskunde en informatica gezamenlijk leren, en dat onderwerp zou men trouwens geen gemeenschappelijke basis kunnen noemen. De schoolmeetkunde waarbij men voorheen leerde wat bewijzen waren is nagenoeg verdwenen, en er valt niet aan te denken die achterstand met universitair onderwijs in te halen. De exacte analyse, waarin de wiskundigen gedurende het grootste deel van deze eeuw de taal van de wiskunde leerden, zij het dan door osmose, wordt niet relevant geacht voor de informatica, en bij de ingedikte opleidingen zullen ook niet alle richtingen in de wiskunde er voor blijven voelen.

Natuurlijk is algebra een goede kandidaat voor een stukje gezamenlijke opleiding, en ook combinatoriek. Maar geen van beide zijn in de beginfase geschikt om er de taal van de wiskunde mee te leren. Algebra niet omdat de studenten nog zo moeten wennen aan de trant van het denken in structuren,

en combinatoriek niet omdat daar eerst de betrekking tussen intuïtie en formulering nog moet worden geleerd.

Wil men komende generaties nog de taal van de wiskunde bijbrengen, met het hele bedrijf van redeneerregels, definities, axioma's, stellingen en bewijzen, en wil men dat dan voor wiskundigen en informatici gezamenlijk doen, dan kan men dat het beste expliciet doceren, liever dan met behulp van osmose. Daar zijn al ervaringen mee opgedaan. Op basis van natuurlijke deductie is het introduceren en elimineren van kwantoren geheel parallel aan operaties uit de propositielogica, en na één semester kunnen de studenten, fris van school komend, de taal van de wiskunde redelijk onder de knie hebben, zelfs als ze op school nog nooit een bewijs hadden gezien. Dan kunnen ze ook alles weten wat iedereen weten moet over vrije en gebonden variabelen, over wat existentie betekent, lambda's en zo meer. Alleen al om elementaire logica te leren is die methode veel doeltreffender dan de pogingen om de logica op te bouwen door te werken met nullen, enen en waarheidstafels, zoals men dat in vele boeken en cursussen nog aantreft. De taal van de wiskunde kan men daar niet uit leren, want die is al zo'n beetje vooröndersteld. De gehele prelogica blijft in de mist. Sommige van die boeken maken van logica iets dat het meest doet denken aan de trigonometrie van lang geleden, waar men vele formules moest leren, om die bij het herleiden van andere formules te kunnen toepassen. Ik heb nooit begrepen hoe mensen hebben kunnen denken dat ze met dit soort van behandeling een inleiding in de grondslag van het redeneren kunnen geven. De enige rechtvaardiging die men er doorgaans voor geeft is dat het zo vaak op deze manier gebeurt.

Docenten in de wiskunde hebben tegen het expliciet doceren van de taal van de wiskunde natuurlijk weer het bezwaar dat Bourbaki het niet deed. Maar Bourbaki kon zich nog op de methode der osmose beroepen. In de huidige situatie is osmose als onderwijsmethode niet meer werkzaam. Ik zie geen andere weg dan het expliciet onderwijzen van de taal van de wiskunde.

Het is misschien ten overvloede dat ik nog eens wil zeggen dat men nooit moet proberen zich tot onderwijs in het formele te beperken. Dat geldt zowel voor wiskunde als voor informatica. Het intuïtieve en het heuristische denken leert men alleen goed aan de hand van het spelen met voorbeelden. In ons hedendaagse gecomprimeerde onderwijs moet er worden opgeschoten, zo gauw het maar kan naar de specialistische toppen toe. Studenten krijgen zelden de kans om zelf iets belangrijks te herontdekken, want ze krijgen de oplossing van een probleem voorgeschoteld nog voordat ze het probleem echt begrepen hebben. En ze krijgen algemene theorieën opgedist zonder enig idee welke gedachten tot die theorie aanleiding hebben gegeven. Het enige wat men in deze richting voor groepen studenten gezamenlijk zou kunnen doen is een soort oefeningen in heuristiek en probleemoplossen. Ook daar is de behoefte bij wiskundigen ternauwernood verschillend van die bij informatici. Het zal niet gemakkelijk zijn daarvoor geschikte vormen te vinden, en misschien nog moeilijker om er tijd voor te vinden, maar het lijkt me belangrijk genoeg om er aandacht aan te besteden.

Tenslotte wil ik nog eens duidelijk zeggen dat naar mijn overtuiging informatica een wiskundige wetenschap is, behalve dan de stukken van de informatica die tot de technische fysica behoren, en dingen als medische informatica, bestuurlijke informatica, die ook volgens vele informatici helemaal geen informatica zijn, al houden ze zich ook bezig met het gebruik van programma's voor databanken en zo meer. De wiskunde kent een soortgelijke relatie tussen het hoofdgebied en toepassingsgebieden. Als de toepassingen voor de wiskundige routinematig zijn en voor hem geen nieuwe gezichtspunten opleveren, zal hij ze met genoegen aan anderen overlaten.

Maar de rest van de informatica is wiskunde. Eigenlijk zelfs meer zuivere dan toegepaste wiskunde. De toegepaste wiskunde is bezig met mathematische modellen voor niet-wiskundige situaties. Bij de informatica is echter de eerste stap stap van de fysische apparatuur naar een wiskundig model doorgaans triviaal, en die stap is dan het enige toegepast wiskundige deel. Het eigenlijke probleem voor de informaticus is om over dat meestal zeer onhandelbare wiskundige model wiskundige uitspraken te doen. De interpretatie daarvan in de fysische apparatuur is weer triviaal.

Toch heeft dat zuiver wiskundige bedrijf van de informaticus hier en daar het karakter van een ingenieurswetenschap. De problemen zijn nl. vaak zo gecompliceerd dat aan volledige formulering en volledige oplossing niet te denken valt. En heel vaak moeten er toch beslissingen worden genomen. Dan staat de informaticus in dezelfde situatie als de toegepast wiskundige en de ingenieur. De keuzen die dan gemaakt worden berusten op ervaringen, intuïtie, natte-vinger-werk of hoe men het maar noemen wil. Maar het blijft een wiskundige situatie waar men mee te maken heeft. En een paar jaar later kan het probleem misschien wél voor redeneringen toegankelijk zijn, als het niet inmiddels, zoals zoveel praktijk-problemen, van de urgentielijst is afgevoerd.

Het feit dat informatica een wiskundige wetenschap is moeten we proberen duidelijk te houden door een belangrijk stuk onderwijs gezamenlijk aan studenten in de wiskunde en in de informatica te geven. De dingen die daar in de eerste plaats in horen zijn expliciet onderwijs in de taal en de methoden van de wiskunde, in het bijzonder in heuristische methoden en in de kunst van het probleemoplossen. Naast een dergelijk centraal pakket voor alle studenten in alle wiskundige wetenschappen zullen er natuurlijk veel stukjes wiskunde en informatica zijn die sommige studenten in de informatica gemeenschappelijk leren met sommige studenten in de zuivere en toegepaste wiskunde.

Maar ik bedoel natuurlijk allerminst dat de huidige wiskundigen de baas moeten gaan spelen over de huidige opleidingen in de informatica. En men moet vooral niet proberen de studenten in de informatica meer van de huidige wiskunde in te prenten dan ooit nuttig zou kunnen zijn in hun informatica.

Zuivere wiskunde en de computer

Computers en (l-adische) Cohomologie

Jaap Top
Mathematisch Instituut
Rijksuniversiteit Utrecht

1. Inleiding

Bij het gebruik van computers in de algebra of de algebraïsche meetkunde wordt meestal in eerste instantie aan 'computer algebra' gedacht. Vele software pakketten op het gebied van manipuleren met algebraïsche formules, idealen in veeltermringen of ringen van differentiaaloperatoren, het berekenen van Hilbert-polynomen en dergelijke hebben allang hun nut bewezen. Niet alleen binnen vakgebieden als homologische of commutatieve algebra, algebraïsche meetkunde en getaltheorie, maar ook zeker in de theorie van differentiaalvergelijkingen en in de natuurkunde. Het is dan ook geen wonder dat computer algebra niet meer uitsluitend als hulpmiddel bij andere vakken bedreven wordt, maar dat het daarnaast steeds meer een eigen leven gaat leiden.

In deze voordracht gaan we evenwel niet in op het fenomeen 'computer algebra', maar op een heel andere manier waarop de computer gebruikt wordt. We hopen duidelijk te kunnen maken dat de computer ons in sommige gevallen in staat stelt, interessante vragen en voorbeelden, waar tot nu toe alleen theoretische of zich tot zeer speciale gevallen beperkende uitspraken over te doen waren, bevredigend te beantwoorden. Dit geldt zeker voor sommige problemen uit de complexe meetkunde. Het basisidee is, deze met behulp van meestal vrij abstracte theorie te herleiden tot een probleem over een eindig lichaam. Daar doet de computer voor ons het rekenwerk en de theorie vertaalt de uitkomsten tot een uitspraak over de complexe getallen.

We zullen dit toelichten aan de hand van twee voorbeelden. Het eerste komt uit de theorie van modulaire vormen. Het tweede betreft het uitrekenen van invarianten van 3-dimensionale complexe variëteiten.

2. Spitsvormen voor $\Gamma_0(N)$

2.1. Inleiding

De complexe vectorruimte $S_2(N)$ die we hieronder definiëren, speelt een belangrijke rol in onder andere de representatietheorie van $GL_2(\mathbf{Q})$, in de theorie van elliptische krommen en in de getaltheorie, bijvoorbeeld bij asymptotiek van klassen aantallen van imaginair kwadratische lichamen. Voor sommige toepassingen is het noodzakelijk heel concrete berekeningen in zo'n ruimte te doen die ver buiten hetgene wat nog gewoon met de hand te doen is, liggen. Op een voorbeeld van zo'n berekening en op hoe die dan met de computer wel binnen het bereik komt, zullen we kort ingaan.

2.2. Definities

Per definitie is $S_2(N)$ de ruimte van spitsvormen van gewicht 2 voor de groep $\Gamma_0(N)$, oftewel de eindig-dimensionale ruimte van holomorfe functies

$$f: \mathbf{H} \to \mathbf{C}$$

(met $\mathbf{H}$ het bovenhalfvlak in de complexe getallen), die voldoen aan

$$f\left[\frac{az+b}{cz+d}\right] = (cz+d)^2 f(z)$$

voor alle matrices $\begin{bmatrix} a & b \\ c & d \end{bmatrix} \in \Gamma_0(N)$, i.e. matrices in $SL_2(\mathbf{Z})$ waarvoor $c \equiv 0 \bmod N$, waarbij f 'moet verdwijnen op de spitsen'. Dit houdt in dat voor $\begin{bmatrix} a & b \\ c & d \end{bmatrix} \in SL_2(\mathbf{Z})$, de functie

$$g(z) := (cz+d)^{-2} f\left[\frac{az+b}{cz+d}\right],$$

die voldoet aan $g(z+N)=g(z)$, een Fourierontwikkeling

$$g(z) = \sum a_n e^{2\pi i n z/N}$$

moet hebben, met $a_n=0$ voor alle $n \leqslant 0$. Door op de vereniging $\mathbf{H}^* := \mathbf{H} \cup \mathbf{P}^1(\mathbf{Q})$ een nogal vreemde topologie te zetten, wordt

$$X := X_0(N) := \Gamma_0(N) \backslash \mathbf{H}^*$$

een Riemann-oppervlak. Dan is er een isomorfie

$$S_2(N) \cong H^0(X, \Omega^1_{X/C})$$

(die laatste ruimte is de ruimte van globale holomorfe 1-vormen op X), gegeven door $f \mapsto 2\pi i f(z)dz$. Hieruit volgt dat de dimensie van $S_2(N)$ gelijk is aan het geslacht van X (en dat is ongeveer $N/12$).

Op $S_2(N)$ werken zgn. Hecke-operatoren T_p (voor p priem, p geen deler van N), gedefinieerd door

$$T_p(\sum a_n e^{2\pi inz}) = \sum\{a_{pn} + pa_{n/p}\}e^{2\pi inz}.$$

Deze T_p's commuteren onderling. Men is geïnteresseerd in gemeenschappelijke eigenfuncties van deze operatoren; met name in die, waarvoor alle T_p's eigenwaarden in $\mathbf{Q}$ hebben.

2.3. Klassieke methoden

Voor heel kleine N is met behulp van een Selberg-spoorformule een basis voor $S_2(N)$ te bepalen. Mestre zegt dat deze aanpak voor het geval N priem en niet groter dan 5000 redelijk werkt.

Een andere aanpak is, de homologie van X te gebruiken. De ruimte $H^1(X, \mathbf{R})$ van reëel analytische 1-vormen op X is duaal met de homologiegroep $H_1(X, \mathbf{R})$. In de homologie definiëert men H_1^+ als het stuk waar de afbeelding geïnduceerd door $z \mapsto -\bar{z}$ triviaal werkt. Het idee is, de Hecke-operatoren in plaats van op

$$H^0(X, \Omega_X^1)'' \subset'' H^1(X, C),$$

te laten werken op $H_1^+(\mathbf{Z})$. Hierbij spelen zgn. modulaire symbolen een grote rol. De methode is ontwikkeld door Birch en Manin; computer programma's hiervoor zijn geschreven door Tingley ($\approx$ 1970; werkt voor $N \leqslant 300$) en Cremona (1988-89; N tot $\approx$ 2000).

2.4. De grafenmethode

De meetkunde van X maakt het mogelijk dit probleem op een heel andere manier te benaderen. Men kan $X_0(N)$ definiëren als schema over $\mathbf{Z}$. Over $\mathbf{F}_p$, voor een priem p die N deelt, ziet $X_0(N)$ eruit als een reducibele kromme. (Denk aan resultaten van Kronecker: hij had al een (singulier) model van $X_0(N)$ als vlakke kromme, geparametriseerd door paren $(j(z), j(Nz))$, (met j de j-invariant) en hij kende het verband

$$(j(z) - j(pz)^p)(j(z)^p - j(pz)) \equiv 0 \bmod p,$$

voor p priem.) Het bestuderen van deze reducibele krommen, mogelijk gemaakt door werk van Deligne & Rapoport, en Katz & Mazur, heeft geleid tot de zogenaamde 'grafenmethode'.

VOORBEELD. ($N = p$ priem; Mestre en Oesterlé 1984; het idee onafhankelijk ook geopperd door Van Geemen en uitgewerkt door Edixhoven.) Neem een $l \neq p$ priem en maak een graaf. De punten van de graaf zijn de (j-invarianten van) de elliptische krommen E over $\mathbf{F}_{p^2}$ met de eigenschap dat E over de algebraïsche afsluiting van $\mathbf{F}_p$ geen punt van orde p bevat. Verbindingen tussen punten zijn de afbeeldingen van graad l tussen de bijbehorende krommen. (De graaf is ongericht als we de verbinding gegeven door een afbeelding identificeren met die gegeven door de zgn. duale afbeelding.) Dan wordt de Hecke-operator T_l in essentie gegeven door de incidentiematrix van deze graaf. De zo verkregen matrices komen allang in de literatuur voor onder de naam 'Brandt-matrices'. Een belangrijke eigenschap van het hier geschetste algoritme

is dat alle berekeningen (in elk geval voor kleine l) gedaan kunnen worden over het lichaam met p^2 elementen. Op deze manier zijn voor alle priemgetallen tot 14000 alle eigenfuncties met rationale eigenwaarden bepaald.

In 1988 heeft Edixhoven een generalisatie gegeven tot het geval $N=p^2$. Het blijkt dat uit dezelfde graaf als hierboven met wat meer werk ook in die gevallen 'de' matrices van de T_l's gehaald kunnen worden. Dit is geïmplementeerd op een 1-megabyte Atari homecomputer; het werkt bijzonder snel voor p^2 tot ≈ 15000000.

Ook uit 1988 stamt een generalisatie van Birch naar het geval dat N een produkt is van allemaal verschillende priemgetallen.

3. Invarianten van bepaalde complexe drievouden

3.1. Inleiding

Zowel vanuit de fysica als vanuit de algebraïsche meetkunde is de laatste tijd veel aandacht besteed aan complex 3-dimensionale variëteiten. Bijvoorbeeld aan Calabi-Yau manifolds, die zowel in superstring theorie als in Mori's theorie van classificatie van drievouden een rol spelen. Er zijn o.a. artikelen van Schoen (1985), Hirzebruch (1986), Werner (1987), Green, Hubsch en Lütken (1988) en Wilson (1989) waarin voor specifieke voorbeelden invarianten van zulke manifolds worden berekend. Interessante voorbeelden zijn desingularisaties van hyperoppervlakken van graad 5 in $\mathbf{P}^4$ met alleen gewone dubbelpunten als singulariteiten. Heel andere voorbeelden, nl. desingularisaties van bepaalde doorsnedes van 4 kwadrieken in $\mathbf{P}^7$, komen voor in de theorie van zgn. Siegel modulaire drievouden.

3.2. De invarianten

Gegeven een glad, projectief drievoud V, wil men de invarianten $h^{p,q}:=$ de dimensie van de vectorruimte van harmonische differentiaalvormen op V, die in locale coördinaten te schrijven zijn als

$$f\cdot dz_{i_1}\wedge\cdots\wedge dz_{i_p}\wedge d\overline{z_{j_1}}\wedge\cdots\wedge d\overline{z_{j_q}}.$$

Er gelden de volgende standaard eigenschappen:

- $\Sigma_{p+q=n}h^{p,q}=h^n:=\dim H^n(V,C)$;
- $h^{p,q}=h^{q,p}$ en $h^{p,q}=h^{3-p,3-q}$;
- $h^{p,q}=0$ als p of q groter dan 3 is;
- $h^{0,0}=h^{3,3}=1$ (N.B. V is glad!)

Een prachtige manier om deze getallen uit te rekenen in sommige gevallen is gebaseerd op Deligne's resultaten over de Weil-vermoedens, en de Lefschetz-formule in l-adische cohomologie. Dit is heel recent gebruikt in preprints van Van Geemen & Nygaard en Van Geemen & Werner. We lichten deze techniek toe aan de hand van een voorbeeld.

3.3. Een voorbeeld
Zij V de variëteit verkregen door het drievoud in $\mathbf{P}^7$

$$V_{sing}: \begin{cases} y_0^2 & = x_0^2+x_1^2+x_2^2+x_3^2 \\ y_1^2 & = x_0^2-x_1^2+x_2^2-x_3^2 \\ y_2^2 & = x_0^2+x_1^2-x_2^2-x_3^2 \\ y_3^2 & = x_0^2-x_1^2-x_2^2+x_3^2 \end{cases}$$

op te blazen in z'n 96 dubbelpunten. Standaardtechnieken leveren (gebruik makend van het feit dat V_{sing} limiet is van gladde volledige doorsnijdingen) dat

$$h^1=h^5=h^{2,0}=h^{0,2}=0.$$

Verder is vrij gemakkelijk de Euler-karakteristiek $e(V)$ uit te rekenen:

$$e(V)=\sum_{i=0}^{6}(-1)^i h^i(V) = 256.$$

We concluderen

$$2h^2-h^3=254.$$

Door het residu te nemen van een differentiaalvorm op $\mathbf{P}^7$ met polen langs de 4 kwadrieken die V_{sing} definiëren is een holomorfe 3-vorm op V te construeren; dus

$$h^{3,0}=h^{0,3}\geqslant 1.$$

Zover komen we met alleen maar theorie; om verder te komen gaan we werken in karakteristiek p, en daar zal de computer ons helpen.

Voor $p\neq 2$ een priemgetal definieert V ook een gladde projectieve 3-dimensionale variëteit over $\mathbf{F}_p$. Er geldt dat het aantal punten van V over $\mathbf{F}_p$ gelijk is aan het aantal punten op V_{sing}, minus het aantal dubbelpunten dat rationaal over $\mathbf{F}_p$ is, plus wat we aan rationale punten na het opblazen voor zo'n singulariteit ervoor in de plaats krijgen. Bij het opblazen wordt een singulariteit vervangen door een kwadriek in $\mathbf{P}^3$, om daarop punten te tellen moet nagegaan worden of de beide lijnenscharen erop over $\mathbf{F}_p$ gedefinieerd zijn. Narekenen levert dat als $p\equiv 1 \bmod 16$, dan zijn zowel alle dubbelpunten als de lijnenscharen gedefinieerd over $\mathbf{F}_p$. De conclusie is

$$\text{als } p\equiv 1 \bmod 16 \text{ dan } \sharp V(\mathbf{F}_p)=\sharp V_{sing}(\mathbf{F}_p)-96+96(p+1)^2.$$

De computer levert dan

$$\sharp V(\mathbf{F}_{17}) = 13024-96+96\cdot 18^2=44032.$$

De rest van het verhaal is weer puur theorie. Rationale punten over $\mathbf{F}_p$ zijn precies de vaste punten van het Frobenius-morfisme

$$F: V\to V,$$

en dat aantal is ook te bepalen met een Lefschetz vaste punten stelling:

$$\#V(\mathbf{F}_p) = \sum_{i=0}^{6}(-1)^i \text{spoor}\{F;\ H^i(V_{\bar{\mathbf{F}}_p}, \mathbf{Q}_l)\}.$$

Uit Deligne's werk weten we dat de eigenwaarden van F op H^i algebraïsch zijn, en (in $\mathbf{C}$) absolute waarde $\leqslant p^{i/2}$ hebben. Verder geldt

$$\text{spoor}\{F;\ H^{3+i}\} = p^i \text{spoor}\{F;\ H^{3-i}\}$$

voor $i=0,1,2,3$. (Om van H^{3-i} naar H^{3+i} te komen moet herhaald cup-product met de klasse van een hypervlaksnede genomen worden; dit geeft elke keer een factor p.) We krijgen dus (voor $p=17$)

$$1+p^3+(1+p)\text{spoor}\{F;\ H^2\}-\text{spoor}\{F;\ H^3\} = 44032.$$

Uit vergelijkingsstellingen volgt dat

$$h^i = \dim H^i(V_{\bar{\mathbf{F}}_p}, \mathbf{Q}_l).$$

In ons geval is het spoor op de H^2 uit te rekenen omdat hier H^2 opgespannen wordt door klassen van divisoren gedefinieerd over $\mathbf{F}_p$; het is bekend dat Frobenius werkt als vermenigvuldiging met p op zulke klassen. Dus

$$\text{spoor}\{F;\ H^2\} = p \cdot h^2.$$

Tenslotte schatten we:

$$|\text{spoor}\{F;\ H^3\}| \leqslant h^3 \cdot p^{3/2}.$$

De conclusie is (gebruik dat $h^3 = 2h^2 - 254$)

$$|1+17^3+18\cdot 17\cdot h^2-44032| = |\text{spoor}\{F;\ H^3\}| \leqslant (2h^2-254)\cdot 17^{3/2}.$$

Er volgt dat

$$h^2 = 128 = h^{1,1}$$

en

$$h^3=2, \text{ dus } h^{3,0}=h^{0,3}=0 \text{ en } h^{2,1}=h^{1,2}=0.$$

Door over nog meer eindige lichamen punten te tellen, is het zelfs mogelijk de Galois-representatie die we hier op de H^3 hebben vrij expliciet in termen van een modulaire vorm te geven.

LITERATUUR

1. J.-F. MESTRE (1986). La Méthode des Graphes. Exemples et Applications. *Proc. Int. Conf. on Class Numbers and Fund. Units of Alg. Number Fields,* Katata.
2. S.J. EDIXHOVEN (1989). *Stable Models of Modular Curves and Applications,* Proefschrift, R.U. Utrecht.
3. BERT VAN GEEMEN, NIELS O. NYGAARD (1988). *L-functions of some Siegel Modular 3-folds,* Preprint, R.U. Utrecht.
4. BERT VAN GEEMEN, JÜRGEN WERNER (1989). *Nodal Quintics in* $\mathbf{P}^4$, Preprint, R.U. Utrecht.

Formele Methoden in Kennisrepresentatie

J.-J.Ch. Meyer
Vrije Universiteit, Amsterdam
Katholieke Universiteit, Nijmegen

In dit artikel wordt een beknopt overzicht gegeven van het belang van een formele (behandeling van de) representatie van kennis in de informatica. Hierbij passeren diverse bekende en minder bekende formalismen de revue. In het bijzonder wordt stilgestaan bij het gebruik van epistemische (kennis-) logica en het zeer uitdagende probleemgebied van niet-monotone redeneermethoden.

Als iemand kennis meent te bezitten, kent hij nog niet op de juiste wijze.
I Korintiërs 8,2

0. Inleiding

Kennisrepresentatie, het representeren van kennis met behulp van meer of minder formele technieken, speelt een belangrijke rol in een uitgestrekt gebied van de informatica (cf. Ringland & Duce [19], Thayse [23], Genesereth & Nilsson [5], Hart [8]). In feite is dit al het geval sinds het pionierswerk van het vak, zoals de term 'informatica' al suggereert: informatica is de wetenschap die zich met informatieverwerking (met name met behulp van computers) bezighoudt. Teneinde informatie te verwerken, dient deze eerst gerepresenteerd (of gecodeerd) te worden. In ruime zin is elke wijze van codering van informatie een vorm van kennisrepresentatie.

Brachman & Levesque [1] definiëren kennisrepresentatie als de representatie van beschrijvingen of afbeeldingen die, wat bepaalde saillante punten betreft, corresponderen met de wereld of toestand van de wereld. Meer in het bijzonder wordt met het probleem van de kennisrepresentatie bedoelt de vraag, hoe we menselijke kennis kunnen representeren/weergeven in termen van datastructuren die een machine kan verwerken (Negoita [16, p. 70]).

Deze problematiek doet zich met name voor in de volgende deelgebieden van de informatica:

a. gegevens- en kennisbanken;
b. computer-ondersteund ontwerp en fabriceren (CAD/CAM);
c. kunstmatige intelligentie, in het bijzonder theorem proving, planning en expertsystemen.

Ad a.

De noodzaak van het representeren van kennis is van 'oudsher' een belang bij het bouwen van gegevens- en kennisbanken. Om gegevens efficiënt te kunnen opslaan en manipuleren, is een goede keuze van de datastructuren essentieel. Deze noodzaak wordt nog groter in het geval de kennisbank ook zelf geacht

wordt deze manipulatie te verrichten teneinde gebruikersvriendelijk op vragen (queries) van de gebruiker te reageren.

Ad b.
Het ontwerpen en fabriceren met behulp van computers vereist dat men ontwerpen kan opslaan en manipuleren. De computer moet als het ware beschikken over een model van (een deel van) de wereld om simulaties te kunnen verrichten. Ook dit vereist kennisrepresentatie.

Ad c.
Bij het 'plannen' van acties teneinde bepaalde doelen te bereiken is het nodig dat men zowel de doelen als het gedrag van acties formeel kan representeren, zodat er geredeneerd kan worden hoe de doelen (het beste) kunnen worden bereikt. Bij het bouwen van expertsystemen, systemen die worden geacht te reageren/antwoorden zoals een menselijke deskundige op een specifiek terrein, is het eveneens noodzakelijk dat de kennis van de deskundige kan worden gerepresenteerd en 'in de computer gestopt'. Tevens moeten middelen voor handen zijn om met deze kennis om te gaan en te redeneren zoals de menselijke expert.

Deze problemen zijn verre van triviaal. Vaak worden er *ad hoc*-oplossingen gezocht en gevonden om (zo snel mogelijk) een werkend systeem te krijgen. Recentelijk echter is men tot het besef gekomen dat het op de langere termijn verstandig is, om te proberen de onderliggende principes te ontdekken en toekomstige systemen hier op te baseren.

Deze ontdekkingstocht voert langs allerlei bekende, maar ook onbekende formalismen, technieken, modellen en theorieën. We zullen enkele hiervan in het kort bespreken.

1. Methoden en theorieën

De klassieke propositie- en predikatenlogica zijn oorspronkelijk bedoeld om redeneringen in te voeren op formele grondslag. Dit maakt deze logica's ook geschikt om kennis mee te representeren (cf. Genesereth & Nilsson [5], Dougherty & Giardina [3], Ramsay [17]). Speciaal een bepaalde klasse van predikatenformules (de zogenaamde definiete Horn clauses) zijn erg populair voor het representeren van kennis (PROLOG). Zo is het bijvoorbeeld Sergot et al. [20] gelukt om een belangrijk deel van de Britse 'Nationality Act' in PROLOG uit te drukken.

De rol van de logica in de kennisrepresentatie is omstreden in de zin dat er bij lange na geen consensus is bereikt over haar nut op dit terrein (Israel [12], Ringland & Duce [19]). Vooral in de kunstmatige intelligentie wordt de beperking gevoeld van (klassieke) logica: een belangrijk bezwaar is dat het bij kennisrepresentatie vaak gaat over '*ordinary discourse*' en '*common-sense reasoning*', zaken die niet (direct) door (klassieke) logica kunnen worden gerepresenteerd. In het dagelijks leven drukt men zich slechts zelden werkelijk formeel uit en zijn onze redeneringen lang niet altijd deductief-logisch. Vaak is er sprake

van redeneerstappen met een bepaalde mate van onzekerheid, ook omdat de ons ter beschikking staande informatie niet altijd volledig/betrouwbaar is. Bovendien drukken we ons normaliter uit in een natuurlijke taal die qua semantiek veel ingewikkelder is dan een formele taal zoals die van de propositie- en predikatenlogica.

Daarom heeft men uitgezien naar andere formalismen en deze heeft men hetzij speciaal ontwikkeld hetzij gevonden in onder andere de logica's die in de *filosofie* waren ontwikkeld voor een beter begrip van specifieke noties, zoals kennis, noodzakelijkheid, tijd, modaliteit (in taalwetenschappen), en morele verplichting. Informatici hebben zich tot nu toe voornamelijk ontfermd over de temporele (tijds-) logica, kennislogica en de modale of intensionele logica meer in het algemeen (Turner [25], Genesereth & Nilsson [5], Ramsay [17]). Ook komt men een enkele keer een *meerwaardige* logica tegen, bijvoorbeeld een 3-waardige met *'(nog) onbekend'* als derde waarheidswaarde (cf. Thayse [23], Turner [25]), en een ∞-waardige logica, waarmee *graden van waarheid* kunnen worden uitgedrukt (Fuzzy logic, Zadeh [27]).

Voorbeelden van speciaal in de informatica (cq. kunstmatige intelligentie) ontwikkelde formalismen voor kennisrepresentatie zijn:

1. *Semantische netwerken.* Dit zijn (gerichte) grafen, waarin de knopen *concepten* representeren (bijvoorbeeld: BOEK, ROOD) en de (gerichte) bogen (of pijlen) *relaties* (bijvoorbeeld IS_GEKLEURD, IS_EEN). Met de IS_EEN-relatie is het mogelijk hiërarchieën van (sub)concepten te representeren. In feite zijn semantische netwerken te beschouwen als een grafische variant van de predicatenlogica, waarbij termen vervangen zijn door knopen en de relatie door pijlen (of meer precies gelabelde gerichte bogen).

2. *Conceptuele grafen.* Deze zijn zeer verwant aan de semantische netwerken, maar hebben een vast formaat van conceptrepresentatie: ze bestaan uit zogenaamde object-attribuut-waarde (O-A-V)-structuren, die van objecten de mogelijke waarden van de diverse attributen van een object geven. Bijvoorbeeld: (SCHRIJFT, JAN, BOEK_4).

3. *Frames.* Deze zijn te beschouwen als *object-georiënteerde* versies van conceptuele grafen. De kennis wordt gerepresenteerd relatief tot objecten, zodat het algemene O-A-V-formaat wordt vervangen door **Object** (attribuut$_j$, waarde$_j$), een 'frame' behorende bij object **Object**. De paren (attribuut$_j$, waarde$_j$) bij een object worden *'slots'* genoemd. Frames worden ook vaak georganiseerd in overervingshiërarchieën, zodanig dat niet bij elk meer gespecialiseerd object alle eigenschappen ('slots') van een meer algemeen object hoeven te worden opgeslagen. Soms worden ook 'default'-overervingen toegelaten: bepaalde eigenschappen van een meer algemeen object worden verondersteld overgeërfd te worden door een meer gespecialiseerd object, tenzij dat object een 'slot' heeft, dat anders aangeeft. Bijvoorbeeld: VOGEL (BEKWAAMHEID, VLIEGEN (default)), terwijl 'subobject' STRUISVOGEL (BEKWAAMHEID, NIET_VLIEGEN). Het redeneren met defaults (verstekwaarden) is

problematisch (cf. Etherington [4], Touretzky [24]) en in feite een bijzondere vorm van 'niet-monotoon' redeneren, hetgeen we dadelijk nog zullen tegenkomen.

We zullen ons nu meer in detail bezighouden met een formalisme dat aan de filosofische logica is ontleend, de *kennislogica*.

2. KENNISLOGICA

Kennislogica (ook wel *epistemische* logica genoemd) is een logica speciaal ontworpen voor het analyseren van de formele eigenschappen van kennis (en geloof, zie bijvoorbeeld Van der Hoek & Meyer [10]). Oorspronkelijk ontwikkeld in de filosofie (door onder andere Jaakko Hintikka [9]) is kennislogica de laatste jaren populair aan het worden onder informatici van divers pluimage. Kennislogica wordt in de informatica gebruikt voor de analyse van kennis in gedistribueerde systemen (protocol-verificatie en dergelijke), in gegevens- en kennisbanken, in automatische ontwerpmethoden (CAD), en in andere kunstmatig intelligente (computer)systemen, zoals expertsystemen (cf. Halpern [6], Vardi [26], Moore [15], Genesereth & Nilsson [5]).

De taal van kennislogica is die van klassieke logica uitgebreid met een (modale) kennisoperator. We beschouwen hier het propositionele geval.

We gaan uit van *primitieve proposities* (of *atomen*) $p,p_1,p_2,\ldots,q,q_1,q_2\ldots$ in een gegeven verzameling **P**. Andere proposities worden hiermee opgebouwd door middel van compositie met de propositionele connectieven $\vee,\wedge,\neg,\rightarrow,\leftrightarrow$ en de speciale unaire kennisoperator K. $K\phi$ wordt gelezen als 'ϕ is bekend' of 'men weet ϕ'. (In de literatuur komt ook een geïndiceerde versie van kennislogica voor: $K_i\phi$ betekent dan 'kenner i weet ϕ'. Wij zullen dit hier niet doen.)

Kennis (of liever de kennisoperator K) voldoet aan de volgende eigenschappen:

(i) $(K\phi\wedge K(\phi\rightarrow\psi))\rightarrow K\psi$ (kennis is afgesloten onder logische gevolgtrekking.)

(ii) $K\phi\rightarrow\phi$ (kennis is waar.)

(iii) $K\phi\rightarrow KK\phi$
(kennis van een assertie is zelf bekend; dit staat bekend onder de naam *positieve introspectie*.)

(iv) $\neg K\phi\rightarrow K\neg K\phi$
(*on*bekendheid van een assertie is zelf wel bekend; dit heet *negatieve introspectie*.)

Eigenschappen (i) - (iii) zijn tamelijk oncontroversieel. Eigenschap (iv) is echter filosofisch nogal aanvechtbaar: onkunde kan 'onbewust' en dus onbekend zijn. Om diverse technische redenen wordt in informaticacontext (iv) meestal toch aangenomen.

Als we de eigenschappen (i) - (iv) als axioma's van kennis nemen en daarover de axioma's van de propositielogica, modus ponens en de afleidingsregel

$$(\textit{generalisatie})\quad \frac{\phi}{K\phi}$$

toevoegen, verkrijgen we een axiomasysteem voor kennis dat in de literatuur bekend staat onder de naam S5. De betekenis van de generalisatieregel is als

volgt: als ϕ een theorema is, dan is ook $K\phi$ een theorema. Dit moet ten zeerste worden onderscheiden van een assertie $\phi \rightarrow K\phi$, hetgeen zou betekenen dat alles wat waar is ook bekend is (wat samen met eigenschap (ii) kennis met waarheid zou doen samenvallen).

We hebben nog niet gesproken over een formeel-logische *interpretatie* van formules met een kennisoperator. Dit wordt gedaan met behulp van zogenaamde *Kripke-structuren*, genoemd naar de filosoof/logicus Saul Kripke, die veel werk (zie bijvoorbeeld Kripke [13]) heeft verricht op het gebied van de (semantiek van) modale logica (cf. Chellas [2], Hughes & Cresswell [11]), waarvan kennislogica een versie/variant is.

Een Kripke-structuur (of model) M is een tupel (S, π, R) waarbij S een niet-lege verzameling (toestanden) is, π een functie $S \rightarrow (\mathbb{P} \rightarrow \{\text{true,false}\})$ (de waarheidstoekenning aan primitieve proposities per toestand) en R een binaire relatie $\subseteq S \times S$ (de 'bereikbaarheidsrelatie'). Gegeven een model $M = (S, \pi, R)$ en een toestand $s \in S$, geeft de relatie R de in s als mogelijk beschouwde werelden: in s wordt de verzameling toestanden $\{t \in S \mid sRt\}$ als mogelijk (alternatief) beschouwd op grond van de kennis die in s aanwezig is. Een Kripke-model is als zodanig beschouwd dus een manier van *kennisrepresentatie*.

Epistemische formules worden als volgt op een Kripke-structuur $M = (S, \pi, R)$ geïnterpreteerd: we beschouwen eerst een waarheidsrelatie $(M,s) \models \phi$, die aangeeft dat de formule ϕ waar is in de 'wereld' (M,s):

1. $(M,s) \models p \Leftrightarrow \pi(s)(p) = \text{true}\ (p \in \mathbb{P})$.
2. $(M,s) \models \neg\phi \Leftrightarrow (M,s) \not\models \phi$.
3. $(M,s) \models \phi \wedge \psi \Leftrightarrow (M,s) \models \phi$ en $(M,s) \models \psi$ en analoog voor de andere propositionele connectieven.
4. $(M,s) \models K\phi \Leftrightarrow$ voor alle t met sRt geldt $(M,t) \models \phi$.

De laatste clausule behoeft enige toelichting: in wereld (M,s) wordt ϕ gekend, als ϕ waar is in alle werelden die in s voor mogelijk worden gehouden. Hoewel er twijfel is over de aard van de 'werkelijke' wereld (er zijn immers mogelijk meerdere alternatieven), is er geen twijfel over de waarheid van ϕ: ϕ geldt in alle alternatieven.

Een formule ϕ heet geldig in een model $M = (S, \pi, R)$, notatie $M \models \phi$, als ϕ geldt in elke wereld (M,s) in M, dat wil zeggen als $(M,s) \models \phi$ voor elke $s \in S$. De formules die geldig zijn in een model M wordt aangeduid als $K(M)$, de *theorie* van M, of de *kennis* in M. Een formule ϕ heet geldig met betrekking tot een klasse $\mathcal{C}$ van modellen, notatie $\mathcal{C} \models \phi$ als ϕ geldig is in elk model M in de klasse $\mathcal{C}$, dat wil zeggen $M \models \phi$ voor elke $M \in \mathcal{C}$.

Men kan nu bewijzen dat de formules die afleidbaar zijn m.b.v. het systeem S5 precies die formules zijn die geldig zijn met betrekking tot de klasse $\mathfrak{S}$, de klasse van Kripke-structuren, waarin de relatie R een *equivalentierelatie* is. Deze 'mooie' correspondentie is trouwens één van de redenen waarom het systeem S5 zo populair is.

3. CASE-STUDY: HET KARAKTERISEREN VAN KENNISTOESTANDEN MET KENNISNISLOGICA

In de informatica is het begrip *'kennistoestand'* (of *'toestand van kennis'*) belangrijk. Het speelt bijvoorbeeld een rol bij de analyse van *communicatie* tussen processoren in een computernetwerk en bij de analyse van de kennis die is opgeslagen in een gegevens- of kennisbank.

Met behulp van kennislogica en in het bijzonder het systeem S5, kunnen we een kennistoestand karakteriseren als een verzameling van formules die aan bepaalde 'mooie' voorwaarden voldoet (Halpern & Moses [7]). We zeiden al dat een Kripke-model in feite een vorm van kennisrepresentatie is! Het bevat in 'gecodeerde' vorm de kennis van een systeem. Daarom identificeren we een kennistoestand met de theorie van een Kripke-model, en wel een S5-model. Een dergelijke verzameling Σ wordt een *stabiele verzameling* genoemd, en heeft de volgende eigenschappen:

a. Σ bevat alle (instanties van) *propositionele tautologieën*
b. Σ is afgesloten onder *logische consequenties*
c. $\phi \in \Sigma \Leftrightarrow K\phi \in \Sigma$
d. $\phi \notin \Sigma \Leftrightarrow \neg K\phi \in \Sigma$

In c. en d. herkennen we weer de eigenschappen van positieve respectievelijk negatieve introspectie: ϕ wordt gekend d.e.s.d.a. $K\phi$ gekend wordt en ϕ wordt *niet* gekend d.e.s.d.a. $\neg K\phi$ gekend wordt.

Stabiele verzamelingen karakteriseren aldus kennistoestanden van kenners (processoren, kennisbanken,...) die perfect (propositioneel) logisch redeneren en positieve en negatieve introspectie kunnen plegen.

Men kan over stabiele verzamelingen epistemische formules een interessant feit bewijzen:

STELLING. *Een stabiele verzameling wordt uniek bepaald door de puur propositionele formules (dat wil zeggen: formules zonder K-operator) die deze bevat.*

Deze stelling zegt dus, dat wanneer we de 'pure proposities' (de 'echte feiten') van een stabiele verzameling weten, we deze in haar geheel kunnen construeren. Met andere woorden, de 'echte' (niet-epistemische) gegevens bepalen een kennistoestand. Dit is natuurlijk niet tegenintuïtief: men zou inderdaad verwachten dat als men over al deze gegevens beschikt, men ook kan bepalen wat de kennis van een kenner is.

Niet alleen de 'pure proposities' bepalen een kennistoestand. Men kan soms ook epistemische formules gebruiken die een toestand van kennis (cq. een stabiele verzameling) bepalen. Echter, niet elke epistemische formule is hier voor geschikt. (Men kan bijvoorbeeld aantonen dat de formule $Kp \vee K\neg p$ niet eerlijk is.) Formules die op unieke wijze een kennistoestand representeren heten 'eerlijk', en spelen een belangrijke rol bij de studie / het begrijpen van systemen met *onvolledige informatie*. We zullen dadelijk zien dat er ook relaties zijn met niet-klassieke redeneermethoden op basis van onvolledige kennis,

waarin bepaalde onzekere (dat wil zeggen logisch niet-verantwoorde) stappen kunnen worden gemaakt.

4. CASE-STUDY: NIET-MONOTOON REDENEREN

Een tweede case-study die we zullen bekijken is die van de *niet-monotone redeneermethoden*. Dit zijn niet-klassieke en vrij problematische methoden van redeneren, die men met name in de *kunstmatige intelligentie* tegenkomt (Touretzky [24], Etherington [4], Reiter [18], Ramsay [17], Thayse [23], Smets et al. [22], Shoham [21]). Deze methoden hebben gemeen dat men *voorlopige* (en logisch onzekere) conclusies trekt, die bij nader inzien moeten worden teruggenomen (doordat nieuwe feiten deze tegenspreken).

Gezien als formele redeneermethoden met afleidbaarheidsrelatie $\vdash$, missen deze de eigenschap van monotonie: als $\Phi_1 \vdash \Phi_2$, hoeft nog niet noodzakelijkerwijs te gelden dat $\Phi_1, \Psi \vdash \Phi_2$.
Versterking van premissen hoeft nog niet tot behoud van (eerder) afgeleide resultaten te leiden.

Er zijn hoofdzakelijk twee (verwante) bronnen van dit soort redeneermethoden:
(1) In het dagelijks leven komen deze onvoorzichtige methoden van redeneren veelvuldig voor, bijvoorbeeld bij probabilistisch, plausibel, hypothetisch en 'counterfactual' redeneren, maar ook bij redeneren 'bij verstek': als er geen aanwijzing tot het tegenovergestelde is, wordt van een bewering (voorlopig) maar uitgegaan, omdat deze normaliter 'altijd' geldt. Dit dagelijks gebruik van niet-monotone redeneermethoden dringt in de kunstmatige intelligentie binnen via het deelgebied dat zich met 'common-sense reasoning' bezighoudt ten behoeve van het ontwerp van (kunstmatig) intelligente systemen.
(2) Soms wordt opzettelijk een niet-monotone redeneervorm gebruikt, uit redenen van efficiënte opslag van gegevens. Vaak wil men alleen *positieve* feiten representeren in een kennissysteem, omdat het aantal *negatieve* feiten dat van de positieve verre overtreft. (Denk aan een spoorboekje: we geven hierin liever niet aan, welke treinen er NIET rijden op willekeurige tijdstippen op willekeurige trajecten.)

Een consequentie van deze handelwijze is, dat het kennissysteem tot een *negatief* feit moet concluderen bij gebrek aan een positief feit. Dit introduceert nonmonotonie. Bijvoorbeeld als een kennisbank uit zijn gegevens G ϕ niet kan afleiden (dat wil zeggen $G \nvdash \phi$), dan besluit deze tot $\neg\phi$ (dat wil zeggen $G \vdash \neg\phi$). Als we nu ϕ aan de gegevens G toevoegen, hebben we natuurlijk $G \cup \{\phi\} \vdash \phi$ en $G \cup \{\phi\} \nvdash \neg\phi$.

We kunnen het verschijnsel nonmonotonie ook al zien optreden bij de karakterisering van kennistoestanden, zoals in de vorige paragraaf.

We hebben gezien dat bepaalde formules kennistoestanden karakteriseren. We kunnen nu een afleidbaarheidsrelatie $\vdash$ invoeren, als volgt:

$$\phi \vdash \psi \Leftrightarrow \text{ behoort tot de kennistoestand } \Sigma_\phi \text{ bepaald door } \phi.$$

(Merk op dat dit alleen welgedefinieerd is als ϕ een eerlijke formule is, dat wil zeggen een kennistoestand uniek bepaalt.)

De aldus gedefinieerde afleidingsrelatie $\vdash$ is niet monotoon. Dit kunnen we gemakkelijk als volgt inzien: Kies $\phi = p \in \mathbf{P}$. Dit is een pure propositie, en dus een eerlijke formule. Omdat $p \in \Sigma_\phi$, geldt ook $Kp \in \Sigma_\phi$ (Σ_ϕ is stabiel). Dit wil dus zeggen dat geldt $\phi \vdash Kp$. Dit is niet verrassend. Beschouw vervolgens $q \in \mathbf{P}$ zodanig dat $q \neq p$. Deze q zit niet in de kennis(toestand) bepaald door ϕ. Dit wil zeggen, dat $\neg Kq \in \Sigma_\phi$ en dus $\phi \vdash \neg Kq$. Echter, als we nu de (eerlijke) formule $\phi' = p \wedge q$ beschouwen, geldt dat zowel p als q in $\Sigma_{\phi'}$ zitten (namelijk $p \wedge q \in \Sigma_{\phi'}$ en $\Sigma_{\phi'}$ is propositioneel afgesloten), dus geldt ook dat Kp en Kq in $\Sigma_{\phi'}$ zijn. Dus $\phi' \vdash Kp \wedge Kq$, en dus $\phi' \nvdash \neg Kq$. Met andere woorden, de versterking van de premisse van ϕ tot ϕ' heeft tot gevolg dat $\neg Kq$ niet meer kan worden afgeleid.

Zoals al gezegd, zijn er nog vele andere niet-monotone redeneermethoden. Een belangrijk onderwerp van onderzoek is op dit moment het formeel behandelen van deze methoden en ze zo mogelijk onder één noemer brengen. Is er zoiets als een niet-monotone logica? (Of kan men het verschijnsel af met klassieke cq. monotone middelen: zie voor een poging Meyer & Van der Hoek [14].) Tot op heden zijn er zo'n tiental formalismen bekend. Men is begonnen deze te relateren, maar men kan nog niet zeggen dat er een echte consensus is bereikt over de aard van niet-monotonie (ondanks aardige pogingen, zoals de theorie van Shoham [21]). Dit is een uitdaging voor de toekomst: zijn we in staat om niet-monotone redeneermethoden, die indruisen tegen de klassieke opvatting van logica, te begrijpen en vruchtbaar te gebruiken?

DANKBETUIGING. Hierbij wil ik mijn vrouw Coby bedanken voor haar uitstekende hulp bij de tekstverwerking.

REFERENTIES

1. R.J. BRACHMAN & H.J. LEVESQUE (1985). *Readings in Knowledge Representation*, Morgan Kaufmann, Los Altos, California.
2. B.F. CHELLAS (1980). *Modal Logic: An Introduction*, Cambridge University Press, Cambridge/London.
3. E.R. DOUGHERTY & CH.R. GIARDINA (1988). *Mathematical Methods for Artificial Intelligence and Autonomous Systems*, Prentice Hall, Englewood Cliffs, New Jersey.
4. D.W. ETHERINGTON (1988). *Reasoning with Incomplete Information*, Pitman/Morgan Kaufmann, London/Los Altos.
5. M.R. GENESERETH & N.J. NILSSON (1987). *Logical Foundations of Artificial Intelligence*, Morgan Kaufmann, Los Altos.
6. J.Y. HALPERN (ed.) (1986). Theoretical aspects of reasoning about knowledge. *Proc. of the 1st Conference*, Monterey, Morgan Kaufmann, Los Altos.
7. J.Y. HALPERN & Y.O. MOSES (1984). Towards a theory of knowledge and ignorance. *Proc. Workshop on Non-Monotonic Reasoning*, AAAI.

8. A. HART (1986). *Knowledge Acquisition for Expert Systems*, Kogan Page, London.
9. J. HINTIKKA (1962). *Knowledge and Belief*, Cornell University Press.
10. W. VAN DER HOEK & J.-J.CH. MEYER (1988). *Possible Logics for Belief*, Techn. rapport IR-170, Vrije Universiteit, Amsterdam.
11. G.E. HUGHES & M.J. CRESSWELL (1977). *An Introduction to Modal Logic*, Methuen & Co. Ltd., London.
12. D. ISRAEL (1983). The role of logic in representation. *IEEE Computer 16(10)*, 37-42.
13. S. KRIPKE (1963). Semantic analysis of modal logic. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik* 9, 67-96.
14. J.-J.CH. MEYER & W. VAN DER HOEK (1988). *Non-Monotonic Reasoning by Monotonic Means*, Tech. rapport IR-171, Vrije Universiteit, Amsterdam.
15. R.C. MOORE (1985). A formal theory of knowledge and action. J.R. HOBBS, R.C. MOORE (eds.). *Formal Theories of the Commonsense World*, Ablex Publ. Comp., Norwood, New Jersey, 319-358.
16. C.V. NEGOITA (1985). *Expert Systems and Fuzzy Systems*, Benjamin/Cummings, Menlo Park, California.
17. A. RAMSAY (1988). *Formal Methods in Artificial Intelligence*, Cambridge University Press, Cambridge.
18. R. REITER (1987). Nonmonotonic reasoning. *Annual Reviews of Computer Science 2*, 147-187.
19. G.A. RINGLAND & D.A. DUCE (eds.) (1988). *Approaches to Knowledge Representation, An Introduction*, Wiley, New York.
20. M.J. SERGOT et al. (1986). The British nationality act as a logic program. *Comm. ACM 29(5)*, 370-386.
21. Y. SHOHAM (1988). *Reasoning about Change*, MIT Press, Cambridge, Massuchusetts.
22. PH. SMETS et al. (1988). *Non-standard Logics for Automated Reasoning*, Academic Press.
23. A. THAYSE (ed.) (1988). *From Standard Logic to Logic Programming* (Introducing a Logic Based Approach to Artificial Intelligence), John Wiley & Sons, Chichester/New York.
24. D.S. TOURETZKY (1986). *The Mathematics of Inheritance Systems*, Pitman/Morgan Kaufmann, London/Los Altos.
25. R. TURNER (1984). *Logics for Artificial Intelligence*, Ellis Horwood/Wiley, Chichester/New York.
26. M.Y. VARDI (ed.) (1988). *Proc. on the 2nd Conf. on Theoretical Aspects of Reasoning about Knowledge*, Pacific Grove, Morgan Kaufmann, Los Altos.
27. L.A. ZADEH (1975). Fuzzy logic and approximate reasoning. *Synthese 30*, 407-428.

Computeralgebra: Wetenschap en Techniek

Arjeh M. Cohen
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands
Rijksuniversiteit Utrecht
Budapestlaan 6, Utrecht

1. Inleiding

Er bestaan computer-systemen waarop niet alleen al het 'gewone rekenwerk' (differentiëren, integreren, rekenen met veeltermen, oplossen van vergelijkingen') mogelijk is, maar waarop ook met zelf te definiëren abstracte algebraïsche structuren (zoals ringen, algebra's, e.d.) gerekend kan worden. Ik verwacht dat deze systemen vrij wijd verbreid zullen worden. Het vak dat zich met de ontwikkeling van dergelijke systemen, het gebruik ervan, en de achterliggende wiskundige theorie bezighoudt wordt wel computeralgebra genoemd.

Er komen ook veel grafische faciliteiten: de weergave van grafieken, oppervlakken en polytopen, geconstrueerd aan de hand van door de gebruiker zelf gedefiniëerde functies en vergelijkingen, belicht, gedraaid en vervormd naar wens, zullen binnenkort tot de standaard hulpmiddelen van bijna elke wiskundige behoren. Misschien dat deze grafische aspecten beter als computermeetkunde dan als computeralgebra te boek kunnen staan; in ieder geval zal ik er in deze voordracht geen aandacht aan besteden.

Op zich is het niet verwonderlijk dat bepaalde aspecten van de algebra zich goed voor 'computerisatie' lenen. Algebra is immers de studie van operaties op verzamelingen (verg. [11]); en als het mogelijk is de elementen van de onderliggende verzameling exact te presenteren en de operaties door middel van algorithmen te beschrijven, dan is de weg vrij om verdere structuur-bepaling (denk aan deelverzamelingen gesloten onder bepaalde operaties, met bepaalde eigenschappen, enz.) met (computer)algorithmen te lijf te gaan.

Computeralgebra houdt zich o.a. bezig met de volgende vragen: Gegeven een structuur (bijv. een algebra, ring, halfgroep, groep, monoïde,...) en een probleem aangaande die structuur (bijv. 'Heeft de structuur een eenheid?'), bepaal òf er een methode is om dat probleem op te lossen; en zoja, geef een methode. Als het duidelijk is dat er een methode is, bekijk dan of er betere methoden

zijn; en geef aan hoe efficiënt ze is, met hoeveel geheugen-beslag de methode werkt, enz.; maar ook: kun je in de praktijk met zo'n methode komen?

Deze drie vragen doorlopen het hele scala van zuivere wetenschap tot techniek. Elke vraag verwijst naar een weliswaar boeiend maar voor deze voordracht veel te omvangrijk terrein. Daarom belicht ik de verschillende aspecten met behulp van enkele voorbeelden, voornamelijk gekozen uit eigen praktijk.

Om een indruk te geven hoe een computeralgebra-pakket werkt, geef ik eerst een voorbeeld hoe het gebruikt kan worden (enkele andere voorbeelden heb ik gegeven in [3]). Het betreft hier wiskunde op het niveau van eerste- of tweedejaars studenten.

2. Gebruik van een computeralgebra-pakket - een voorbeeld

In verband met enkele ideeën over voorstellingen van Liegroepen, wilde ik wat berekeningen uitvoeren van de volgende aard. Gegeven een polynomiale functie $F(a,b)$ in de variabelen a en b, schrijf op een rationale functie $f(x,y)$ in de variabelen x en y zodanig dat de reeksontwikkeling van f rond $(x,y)=(0,0)$ geeft

$$f(x,y)=\sum_{a,b\geqslant 0} F(a,b)x^a y^b.$$

Voor F had ik met name in gedachten de veeltermen

$$A2(a,b) = \tfrac{1}{2}(a+1)(b+1)(a+b+2),$$

$$B2(a,b) = \tfrac{1}{3!}(a+1)(b+1)(a+b+2)(2a+b+3),$$

en

$$G2(a,b) = \tfrac{1}{5!}(a+1)(b+1)(a+b+2)(a+2b+3)(a+3b+4)(2a+3b+5).$$

(De in Liegroepen geïnteresseerde lezer zal deze uitdrukkingen wellicht van blz. 140 in [8] herkennen.) Overwegende dat

$$\frac{c!}{(1-x)^{c+1}} = \sum_{a\geqslant 0}(a+1)\cdots(a+c)x^a$$

kwam ik tot de volgende Pascal-achtige functie, waarin de gelijkheid met herhaling wordt toegepast om F af te breken tot kleinere gevallen. Deze functie is geprogrammeerd voor gebruik in MAPLE†

† MAPLE is de naam van een in Waterloo ontwikkeld computeralgebra-pakket.

```
readlib(coeftayl):
mkrat := proc(F)
local adeg,bdeg,corr,i,antw,r,d,tt;
tt := expand(F);
adeg := degree(tt,a):
if tt=0 then antw := 0
   else
   d := expand(coeftayl(tt,a=0,adeg));
   bdeg := degree(d,b):
   d := coeftayl(d,b=0,bdeg);
   corr := 1;
   for i to adeg do corr := corr*(a+i) od;
   for i to bdeg do corr := corr*(b+i) od;
   corr := corr*d;
   tt := tt-corr;
   r:=((adeg!)*(bdeg!)*d/(((1-y)(bdeg+1))*((1-x)(adeg+1))));
   antw := mkrat(tt)+r
fi;
antw
end:
```

In de eerste regel van deze tekst wordt de bibliotheek-functie `coeftayl` ingelezen, die, na de aanroep `coeftayl(tt,a=0,adeg)`, de coefficient van a^{adeg} in de Taylorontwikkeling van *tt* naar *a* rond 0 levert. Dan wordt de functie `mkrat` gedefiniëerd, met als argument een veelterm F in a en b. De definitie is recursief: *antw* roept `mkrat` weer aan: voor de veelterm *corr* met dezelfde hoogste graads term $da^{adeg}b^{bdeg}$ als F (in de met behulp van `expand` volledig als som van termen uitgeschreven vorm) wordt de rationale functie r bepaald zo dat $mkrat(F)=mkrat(F-corr)+r$.

De gewenste resultaten worden nu als volgt verkregen:

```
    |\^/|       McMaple V4.0 --- January 1987
._|\|   |/|_.   Licensed by Univ of Waterloo
 \  MAPLE  /    Dept ZW, CWI, Amsterdam
 <_______>     Do not redistribute
    |          For on-line help, type  help( );

>

read m: # de functie mkrat wordt ingelezen van bestand m#
```

```
>
```

$$A2:=(a+1)*(b+1)*(a+b+2)/2:$$

```
            # bovenstaand voorbeeld A2 wordt ingevoerd #
>
```

$$B2:=(a+1)*(b+1)*(a+b+2)*(2*a+b+3)/6:$$

```
>
```

$$G2:=(a+1)*(b+1)*(a+b+2)*(a+2*b+3)*(a+3*b+4)*(2*a+3*b+5)/120:$$

```
>
  mkrat(A2);
```

$$-\frac{1}{(1-y)^2(1-x)^2}+\frac{1}{(1-y)^3(1-x)^2}+\frac{1}{(1-y)^3(1-x)^2}$$

```
>
a2 := normal(''); #brengt voorgaand resultaat in normaalvorm#
```

$$a2 := -\frac{-1+yx}{(-1+y)^3(-1+x)^3}$$

```
>
b2 := normal(mkrat(B2));
#hetzelfde als voor A2, maar nu in een samengestelde opdracht#
```

$$b2 := \frac{1+x-4yx+y^2x+y^2x^2}{(-1+y)^4(-1+x)^4}$$

```
>
  g2 := normal(mkrat(G2));  #idem voor G2 #
g2 :=
```

$$(1+x+8y+8y^2x-6y^4x+15y^3x^2-26y^3x^3+8y^3x^4+y^4x^3 +y^4x^4-26yx-41y^2x+78y^2x^2+8y^2+y^3+15yx^2-6yx^3 +yx^4-41y^2x^3) /((-1+y)^6(-1+x)^6)$$

Het resultaat kan eenvoudig gecontroleerd worden voor een aantal waarden van a en b. Daartoe is een functie als deze zeer geschikt:

```
poll := proc(f,a,b )
coeftayl(coeftayl(f,x=0 ,a) ,y=0,b)
end:
```

Een enkel voorbeeld.

```
poll(g2,1,1);
                                        64

>
subs(a=1,b=1,G2)  ;
                                        64
```

(met **for** lussen werkend kunnen moeiteloos veel meer waarden gecontroleerd worden).

Wat ik hier duidelijk wilde maken is dat, door op betrekkelijk natuurlijke wijze te formuleren wat je berekenen wilt (namelijk via de functie-definitie van `mkrat`), je vrijwel moeiteloos aan resultaten komt die anders alleen door 'dieper inzicht' (en/of ad hoc argumenten) of door veel standaard rekenwerk te verkrijgen zouden zijn.

3. Onbeslisbaarheid

Zoals gezegd, één van de vraagstellingen die onder computeralgebra vallen, is: Gegeven een structuur (bijv. een algebra, ring, halfgroep, groep, monoïde,...) en een probleem aangaande die structuur (bijv. 'Heeft de structuur een eenheid?'), is er een methode om dat probleem op te lossen? Aan dit onderwerp is al veel aandacht besteed voordat het woord computeralgebra in de mond genomen werd. Het heeft te maken met de vraag wat een algorithme is en hoe we een probleem formuleren. Zie bijv. de Russische encyclopedie [11] voor een korte inleiding in deze materie. De theorie van algorithmen die uitgaat van de these van Church, geeft een natuurlijke klasse $\mathfrak{A}$ aan van specifieke algorithmen (deze klasse valt onder andere te definiëren met behulp van recursieve functies of van Turing machine's). De begrippen (on)oplosbaar en (on)beslisbaar voor een collectie problemen $\mathfrak{B}$ wordt vervolgens gehanteerd als het criterium of er al dan niet een algorithme in $\mathfrak{A}$ te vinden is dat, indien toegepast op een probleem uit $\mathfrak{B}$, het (goede) antwoord geeft. Voor de collectie problemen 'Gegeven een Turing machine en een invoer, stopt de machine met die invoer (na een eindig aantal stappen)?', geparametriseerd door een Turing machine en een tekst die in de machine ingevoerd kan worden, is een klassiek voorbeeld van een onoplosbare collectie. (De collectie kan zelfs kleiner gemaakt worden met behoud van onoplosbaarheid: er is een Turing machine zodanig dat het probleem geparametriseerd door invoer x 'Stopt de machine als x ingevoerd wordt?' een onoplosbare collectie vormt. De existentie van zo'n Turing machine kan door een diagonalisatie-argument aangetoond worden.)

Maar ook in de algebra zijn er onoplosbare problemen. (In het dagelijkse leven wordt vaak van een probleem gesproken waar een hele collectie van problemen bedoeld wordt.) Een fundamenteel voorbeeld is het woord-probleem

in de groepentheorie. Een in de groepentheorie veel gebruikte manier om een groep te presenteren is die van de voortbrengers en relaties. Laat G een groep zijn. Een presentatie van G is per definitie een stel voortbrengers X van G tesamen met een stel relaties R van de vorm $x_1^{\epsilon_1}\cdots x_m^{\epsilon_m}=y_1^{\delta_1}\cdots y_n^{\delta_n}$ met $x_i,y_j\in X$ en $\epsilon_{i,\delta_j}=\pm 1$ voor elke i,j, zodanig dat G isomorf is met het quotient van de vrije groep F voortgebracht door X (gezien als verzameling letters) naar de kleinste normaaldeler N voortgebracht door R (opgevat als de verzameling 'woorden' $x_1^{\epsilon_1}\cdots x_m^{\epsilon_m}y_1^{-\delta_1}\cdots y_n^{-\delta_n}$ voor de relatie als boven uit R). Een element van G kan dus op een computer weergegeven worden door door een woord over het alfabet $X\sqcup X^{-1}$. (Hier is de conventie $X^{-1}=\{x^{-1}\in X|x\in X\}$ gehanteerd.) Maar deze representatie is niet eenduidig. Zelfs als $N=\{1\}$ (dus $G\cong F$ een vrije groep), dan vertegenwoordigen 1 en xx^{-1} het zelfde element. Het woord-probleem, geparametriseerd door twee woorden u en v over het alfabet $X\sqcup X^{-1}$, stelt de natuurlijke vraag: zijn de woorden u en v opgevat als elementen van G gelijk (precieser: geldt $\overline{u}N=\overline{v}N$ waar $\overline{u}$ het beeld van u in F is)?

Gaan we nog even terug naar de vrije groep F zelf, dan zien we dat de problemen hier niet al groot zijn: na wegpoetsen van alle voorkomens van xx^{-1} (voor $x\in X\sqcup X^{-1}$ komen we op een *gereduceerd woord.* Elk element van F correspondeert met een uniek gereduceerd woord. Het bovenbeschreven poetswerk van u en v tot gereduceerde woorden is dus een algorithme om de vraag $\overline{u}=\overline{v}$ op te lossen. Het woordprobleem is dus oplosbaar voor de vrije groep.

Volgens de stelling van Boone-Novikov zijn er groepen waarvoor het woord-probleem in de groepentheorie onoplosbaar is. Hieronder staat een weliswaar gecompliceerd, maar aanzienlijk eenvoudiger voorbeeld dan in de oorspronkelijke bewijzen te vinden is. Het is opgesteld door Collins [5] die een methode van Borisov gebruikte om deze groep uit een halfgroep met onoplosbaar woord-probleem te maken. (Op pag. 146 van [11] staat een voorbeeld van een eenvoudige halfgroep-presentatie waarvoor het woord-probleem onoplosbaar is.) Met $[x,y]$ (voor $x,y\in G$) geven we het element $xyx^{-1}y^{-1}$ aan; dit element heet de commutator van x en y.

STELLING (Boone-Novikov; cf. [5]). *Er is geen algorithme dat het woord-probleem oplost voor de groep met als presentatie*

voortbrengers : a,b,c,d,e,p,q,r,t,k;

en relaties :

$$[a,p]=[b,p]=[c,p]=[d,p]=[e,p]=p^9,$$

$$[q^{-1},a^{-1}]=[q^{-1},b^{-1}]=[q^{-1},c^{-1}]=[q^{-1},d^{-1}]=[q^{-1},e^{-1}]=q^9,$$

$$[a,r]=[b,r]=[c,r]=[d,r]=[e,r]=1,$$

$$pacqr=rpcaq,p^2adq^2r=rp^2daq^2,p^3bcq^3r=rp^3cbq^3,p^4bdq^4r=rp^4dbq^4,$$

$$p^5ceq^5r=rp^5caq^5,p^6deq^6r=rp^6edbq^6,p^7cdcq^7r=rp^7cdceq^7,p^8ca^3q^8r=rp^8a^3q^8,$$

$$p^9da^3q^9r=rp^9a^3q^9,ka^{-3}ta^3=a^{-3}ta^3k,$$

$$[t,p]=[t,q]=[k,p]=[k,q]=1.$$

Het bewijs van deze stelling berust op de existentie van een zeer eenvoudig gepresenteerde halfgroep waarvan het woord-probleem onoplosbaar is. Het al oudere bewijs van onoplosbaarheid van het woord-probleem in de halfgroep voert terug tot het stop-probleem voor Turing machine's.

4. TODD-COXETER

De presentatie van Collins' groep geeft in zekere zin aan dat er geen algorithme te vinden dat elke groep gegeven door (eindig veel) voortbrengers en relaties kan analyseren. Dat neemt niet weg dat er algorithmen zijn die in veel gevallen een goed resultaat leveren. Een voorbeeld daarvan is het Todd-Coxeter algorithme dat poogt om van een groep G en een ondergroep H de permutatie-voorstelling $H \setminus G$ te bepalen. We brengen in herinnering dat deze permutatie-voorstelling een morfisme van G naar de groep van permutaties van de verzameling $H \setminus G = \{Hg | g \in G\}$ is gegeven door het voorschrift $x \mapsto (Hg \mapsto Hgx)$. De elementen gH van de verzameling permutandi heten de nevenklassen van G naar H.

Laat gegeven zijn een groep G voorgesteld met voortbrengers X en relaties R. Zij verder $w_1, \ldots, w_t$ een stel woorden over het alfabet $X \sqcup X^{-1}$ en laat H de ondergroep van G zijn die voortgebracht is door de elementen corresponderend met $w_1, \ldots, w_t$. De Cayleygraaf van G met betrekking tot de gegeven presentatie en de woorden $w_1, \ldots, w_t$ is de graaf op de puntverzameling $H \setminus G$ waarvan de kanten gericht zijn en 'gekleurd' met X: er loopt een kant gekleurd $x \in X$ van punt Hg naar punt Hg' precies dan als $Hgx = Hg'$.

Het Todd-Coxeter algorithme probeert de Cayleygraaf te maken, uitgaande van het punt H, dat de naam 1 krijgt. Zie [13] voor een grondige behandeling van de methode. Kort beschreven, voor elk getekend punt en elke voortbrenger $x \in X$ worden de in- en uit-gaande kanten met kleur x getekend. Als het niet meteen duidelijk is of/waar onder de getekende punten het begin-, respectievelijk, eind-punt van deze kant terecht komt, dan wordt een nieuw punt gecreëerd. Bij tijd en wijle wordt de creatie van nieuwe punten afgewisseld door het samentrekken van coïncidenties. Er zijn drie soorten coïncidenties, waarvoor zorg gedragen moet worden:

(i) Als i een punt is waaruit $x \in X$ zowel naar j als naar k wijst en $j < k$, dan trekken we k met j samen (immers: x kan i maar naar één punt sturen want het moet een permutatie van $H \setminus G$ worden; dit punt wordt met ix aangeduid). Hetzelfde geldt voor twee kanten van dezelfde kleur x die in i uitkomen (dat correspondeert met ix^{-1}).

(ii) Als $x_1 \cdots x_r \in R$ en i is een punt, dan zal $ix_1 \cdots x_r$ met i samenvallen.

(iii) $w_i = x_1 \cdots x_r$, dan is $1x_1 \cdots x_r = 1$ (want $w_i \in H$ en $1 = H$ heeft tot gevolg dat $1w_i = H = 1$).

Als er geen coïncidenties en geen nieuwe punten meer zijn, dan is de Cayleygraaf van G met betrekking tot X, R en de woorden $w_1, \ldots, w_t$ tot stand gekomen.

STELLING (Mendelssohn, cf. [13]). Als de index $|H \setminus G|$ van H in G eindig is, dan termineert het Todd-Coxeter algorithme.

VOORBEELD. De groep G met presentatie: voortbrengers $X=\{x,y,z\}$ en relaties $R=\{x^2=y^2=z^2=1,(xy)^3=(xz)^2=(yz)^4=1\}$. We nemen als punt van de te construeren Cayleygraaf de ondergroep H voortgebracht door y en z. Laten we voor de duidelijkheid de kanten met zelfde begin- als eind-punt weg, dan resulteert de Cayleygraaf

$$\circ \overset{x}{\leftrightarrow} \circ \overset{y}{\leftrightarrow} \circ \overset{z}{\leftrightarrow} \circ \overset{y}{\leftrightarrow} \circ \overset{x}{\leftrightarrow} \circ$$

waarin H een eindpunt is.

VOORBEELD. Collins' groep van paragraaf 3 geeft over de ondergroep $\langle p,q,r,t,k\rangle$ index 1; dat wil zeggen dat a,b,c,d,e uitgedrukt kunnen worden in p,q,r,t,k.

VOORBEELD. De Weylgroep van type E_8 Zij W de Weylgroep van type E_8. Dan heeft W een presentatie op voortbrengers $r_1,\ldots,r_8$ die gegeven worden door het bekende Dynkin diagram van type E_8 (bij de nummering der punten houden we Bourbaki's volgorde aan). We zullen vaak i schrijven in plaats van r_i ($1\leqslant i\leqslant 8$). De relaties zijn $(ij)^3=1$ als $\{i,j\}$ een binding is in het diagram en $(ij)^2=1$ anders (dus ook als $i=j$). We selecteren een ondergroep D die isomorf is met de Weylgroep van type D_8. De ondergroep gegeven door

$$D=\langle 2,3,4,5,6,7,8,w^{-1}8w\rangle$$

waar $w=76543241563527614534132456785342134567 8$, voldoet. (De voortbrengers van D zijn hier de spiegelingen met respectievelijke wortels $\alpha_2,\alpha_3,\ldots,\alpha_8,h\alpha_0$ (= de langste wortel).) Het zogenaamde Coxeter element $t=14683257$ heeft banen ter lengte 15 op $D\backslash W$. Schrijf $c=134354365768$ en $d=134254316542347$. Het uitvoeren van Todd-Coxeter met CAYLEY† en het nagaan van de actie van de elementen t, c en d op de resulterende Cayleygraaf leert dat

$$Y=\{1,c,c^2,c^3,c^4,c^5,d,d^3,d^4\}\,\{t^i|0\leqslant i\leqslant 14\}$$

een stelsel nevenklassenrepresentanten is van D in W, d.w.z.

$$W=\coprod_{y\in Y} Dy.$$

5. KNUTH-BENDIX

Zie [9] voor een uitgebreidere inleiding in de Knuth-Bendix theorie dan ik hier zal geven. Laat **M** de (niet-commutatieve) vrije monoïde op eindig veel letters zijn. De elementen van **M** stellen op een computer weer te geven objecten voor. In het algemeen is het doel **M** te gebruiken om de elementen te

† CAYLEY is een in Sydney ontwikkeld computeralgebra-pakket, toegespitst op berekeningen aan groepen

representeren van een algebraïsche structuur A die op een of andere wijze aan $\mathbf{M}$ geliëerd is. Bij A valt te denken aan een groep die een quotient van $\mathbf{M}$ naar een stel relaties is zoals we in de vorige paragraaf zagen, maar ook aan een veeltermring, zoals we in de volgende paragraaf zullen zien.

In deze paragraaf zal ik voor A het quotient G naar een stel relaties R nemen. G is dus niet noodzakelijk een groep. Maar door verstandige keuze van R (om inversen e.d. te forceren) is het mogelijk deze eigenschap af te dwingen.

Om elementen van $\mathbf{M}$ te kunnen vergelijkingen gaan we uit van een lineaire orde $<$ op $\mathbf{M}$ die de relatie '|' (d.w.z. 'is een deler van') verfijnt. Voor alle $m,m',m'',n \in \mathbf{M}$ moet gelden

(i) als $m \neq 1$ dan $1<m$;

(ii) als $m'<m''$ dan $mm'n<mm''n$.

Een lineaire orde met deze eigenschappen heet een *reductie-orde* (in sommige litteratuur ook wel *admissibel*). De totale-graad-orde (eerst naar totale graad, dan naar lexicografische orde) is een voorbeeld. Een belangrijke eigenschap van de orde is welgefundeerdheid: dat wil zeggen dat elke dalende rij afbreekt (na eindig veel stappen). Het woordprobleem is nu in termen van de orde te formuleren. Een elementaire transformatie met behulp van R is een herschrijving van de vorm $umv \Rightarrow um'v$ met $m=m'$ een relatie uit R. Hier moeten we dus een linker en rechterlid van de relaties in R onderscheiden. Doen we dat dan spreken we van een *herschrijf-systeem*, waarin $m \Rightarrow m'$ een herschrijfregel is als $m>m'$. Twee woorden $w,w' \in \mathbf{M}$ geven hetzelfde element van het quotient G - verkregen door uitdelen naar de relaties R - weer als w' uit w verkregen kan worden door een rij van successieve transformaties van de vorm $umv \Rightarrow um'v$ met $m \Rightarrow m'$ een herschrijfregel (uit R) of een inverse herschrijfregel, d.w.z. $(m' \Rightarrow m) \in R$. Als je met twee woorden geconfronteerd wordt waarvan je gelijkheid moet testen, dan is het à priori niet duidelijk hoever je w eerst 'op moet blazen' (d.w.z. met inverse herschrijfregels moet behandelen) voor je met herschrijfregels bij w' kunt komen. We zeggen dat w *reduceert* (of: *te herschrijven is*) tot w', notatie $w \Rightarrow^* w'$ als er een serie herschrijfregels is dat, na successief op w losgelaten te zijn, w' oplevert.

Elke deelverzameling van $\mathbf{M}$ heeft een uniek kleinste element t.o.v. $<$. Dus als we een algorithme hebben om van een woord het kleinste woord te vinden dat hetzelfde element van G weergeeft, dan hebben we ook een algorithme dat het woord-probleem oplost. Het Knuth-Bendix algorithme, toegepast op een stel relaties R, probeert om een nieuw stel relaties R' met herschrijf-regels te maken die hetzelfde quotient G opleveren, maar met de additionele eigenschap dat als u,v,w woorden in $\mathbf{M}$ zijn zodanig dat u herschrijft tot zowel v als w, er een element x is waartoe zowel v als w te herschrijven zijn. Deze eigenschap heet wel *confluentie*. Het zal duidelijk zijn dat als er eenmaal een confluent stel herschrijfregels gevonden is, elk woord w tot een uniek minimaal woord w_0 te herschrijven is, en dat w_0 het unieke minimale woord is dat het bij w horende element van G weergeeft. Kortom, als het Knuth-Bendix algorithme slaagt in het vinden van een confluent systeem, dan is het woord-probleem oplosbaar door herschrijven.

VOORBEELD. Het volgende systeem van 10 herschrijfregels voor het eerste voorbeeld uit §4 is confluent m.b.t. de totale graad-reductie-orde met $x>y>z$.

$$x^2 \Rightarrow 1;\ y^2 \Rightarrow 1;\ z^2 \Rightarrow 1;$$

$$x^{-1} \Rightarrow x;\ y^{-1} \Rightarrow y;\ z^{-1} \Rightarrow z;$$

$$yxy \Rightarrow xyx;\ zx \Rightarrow xz;\ zyzy \Rightarrow yzyz;$$

$$zyxzyx \Rightarrow yzyxzy.$$

VOORBEELD. Voor Coxetergroepen bestaat een fraai stel herschrijfregels dat niet confluent is maar toch (betrekkelijk) snel tot goede resultaten leidt. Als $(m_{ij})_{1\leqslant i,j\leqslant n}$ een $n\times n$ matrix is met $m_{ii}=1$ en $m_{ij}=m_{ji}$ voor alle i,j, dan is het *Coxetersysteem* over $(m_{ij})_{1\leqslant i,j\leqslant n}$ de groep-presentatie met voortbrengers $r_1,\ldots,r_n$ en relaties $(r_ir_j)^{m_{ij}}=1$ voor alle $i,j\in\{1,\ldots,n\}$. (Het eerste voorbeeld van §4 is een Coxetersysteem.) De bijbehorende groep heet een *Coxetergroep*. Laat M de vrije monïde de over $r_1,\ldots,r_n$ zijn en beschouw de herschrijfregels

$$r_ir_i \Rightarrow 1 \quad (1\leqslant i\leqslant n),$$

$$r_ir_jr_i\cdots \Rightarrow r_jr_ir_j\cdots \quad (1\leqslant i,j\leqslant n) \qquad \text{beide zijden ter lengte } m_{ij}.$$

Tits heeft bewezen dat twee woorden hetzelfde element van W representeren dan en slechts dan als de een uit de ander te verkrijgen is door een serie van deze herschrijfregels. In het bijzonder kan het woord-probleem $w=w'$ in W? opgelost worden door eerst beide woorden te reduceren tot minimale lengte (dat is de lengte waarbij geen herschrijfregel van de tweede soort leidt tot een woord met herhaald voorkomen van een letter r_i), en vervolgens met de de tweede soort regels alle mogelijke alternatieven voor w op te schrijven en na te gaan of w' daar onder voor komt.

In plaats van het Knuth-Bendix algorithme in zijn algemeenheid te bespreken, wil ik een speciaal geval/variant voor veeltermringen behandelen waar het algorithme altijd slaagt (termineert) met als gevolg dat veel problemen in die ringen oplosbaar zijn.

6. GRÖBNER BASES

Laat R een lichaam zijn waarin gerekend kan worden (de vraag welke lichamen hiervoor in aanmerking komen slaan we over, maar u kunt de rationale getallen in gedachten nemen) en schrijf $T=R[X_1,\ldots,X_n]$ voor de veeltermring in de variabelen $X_1,\ldots,X_n$. De ring T speelt voor ringen een rol vergelijkbaar met de rol die F speelde in paragraaf 4, en ten dele de rol van M in paragraaf 5, namelijk daar waar het de weergave van de objecten betreft. In het bijzonder denken we ons in dat de elementen van T weer te geven zijn op een computer, en stellen we ons ten doel de structuur van quotientringen te onderzoeken. Daartoe geven we, uitgaande van een eindig stel veeltermen B, met (B) het ideaal in T voortgebracht door B aan; de elementen f en f' van T

representeren hetzelfde element van de quotientring $S=T/(B)$ als $f+(B)=f'+(B)$. Het probleem dat we allereerst te lijf gaan is het woordprobleem voor S: Gegeven B en twee veeltermen f en f', is er een algorithme dat beslist of $f=f' \bmod(B)$, dus of $f=f'$ in S geldt?

Een *monoom* in T is een element van T van de vorm $X_1^{a_1} \cdots X_n^{a_n}$, vaak afgekort tot X^a, waarin a staat voor $(a_1, \ldots, a_n) \in \mathbb{N}^n$. Met $\mathbf{M}$ geven we de monoïde van alle monomen aan. Deze monoïde zal grotendeels dezelfde rol spelen als de monoïde $\mathbf{M}$ uit de vorige paragraaf, namelijk daar waar het de contrôle op de reductie van onze objecten betreft.

De monoom $m=X^a \in \mathbf{M}$ kan met a in $\mathbb{N}^n$ geïdentificeerd worden. Een element $f \in T$ kan uniek geschreven worden als $f=\Sigma_{m \in \mathbf{M}} f_m m$. De summand $f_m m$ heet een *term* van f. We kiezen weer een reductie-orde $<$ op $\mathbf{M}$ (dus met de eigenschappen (i) en (ii) van boven). De lexicografische orde is nu een voorbeeld. De hoogste monoom van $f \in T$, geven we aan met Lm_f, dus

$$Lm_f := \max \mathbf{M}_f, \quad \text{waar} \quad \mathbf{M}_f := \{m \in \mathbf{M} | f_m \neq 0\}$$

en de coëfficiënt van dit monoom met Lc_f.

Voor $f \in T$ zoeken we een kanonieke representant van $f+(B) \in T/(B)$. Deze representant moet een zo klein mogelijk hoogste monoom hebben. Als, voor $f, g \in T$ er $m \in \mathbf{M}_f$ en $b \in B$ bestaan met

$$Lm_b | m \quad \text{en} \quad g=f-\frac{f_m}{Lc_b}\frac{m}{Lm_b}b$$

(zodat $g_m=0$), dan schrijven we $f \Rightarrow_B g$ of kortweg $f \Rightarrow g$. Merk op dat g dan kleiner is dan f in de zin dat $M_f > M_g$ (m.b.t. een natuurlijke door $>$ geïnduceerde orde) en dat $g+(B)=f+(B)$. Laat $g, h \in T$. We zeggen dat g *reduceert tot* of *herschrijfbaar is tot* h modulo B, notatie $g \Rightarrow^* h$, als er een (mogelijk triviale) rij reducties is die begint met g en eindigt met h. Verder heet g een *normaalvorm* m.b.t. B en $<$ als er geen $h \in T$ is met $g \Rightarrow_B h$, en heet $h \in T$ een *normaalvorm van* $g \in T$ als h in normaalvorm is en $g \Rightarrow^* h$. Nu zijn we toe aan het begrip *Gröbner basis*. B heet een *Gröbner basis* met betrekking tot $<$ als elk element van T een unieke normaalvorm heeft met betrekking tot B.

VOORBEELD. (Normaalvorm is niet noodzakelijk uniek). Neem $T=\mathbb{C}[X, Y]$ en $B=\{X^2Y-1, XY^2-1\}$. Dan zijn X en Y beide normaalvormen van X^2Y^2 m.b.t. de lexicografische orde. Het element $X-Y=Y(X^2Y-1)-X(XY^2-1) \in (B)$ is in normaal-vorm maar niet herschrijfbaar tot 0.

Op grond van het feit dat $>$ geïnduceerd op de collectie deelverzamelingen van $\mathbf{M}$ ook wel-gefundeerd is, is het eenvoudig om een algorithme te construeren dat f in normaalvorm brengt m.b.t. B: Zoek $b \in B$ en $m \in \mathbf{M}_f$ zó dat $Lm_b | m$; als b, m gevonden zijn, neem dan $f :=$ Normaalvorm $(B, f-f_m m(Lm_b Lc_b)^{-1} b)$ (recursie !); zo b en m niet bestaan, is f in normaalvorm.

STELLING (Karakterisatie van Gröbner bases, cf. [2]). *Zij $<$ een reductie-orde op de verzameling $\mathbf{M}$ van monomen in T. De volgende uitspraken aangaande een eindige deelverzameling B van T zijn equivalent.*

(i) *B is een Gröbner basis;*
(ii) *voor alle* $f,g,h \in T$ *met* $h \Rightarrow_B f$ *en* $h \Rightarrow_B g$, *bestaat er een* $k \in T$ *met* $f \Rightarrow_B^* k$ *en* $g \Rightarrow_B^* k$;
(iii) *elk element van* (B) *reduceert tot* 0;
(iv) 0 *is de unieke normaalvorm van elk element in B;*
(v) $\{Lm_f | f \in (B)\} = \bigcup_{b \in B, t \in \mathbf{M}} \{tLm_b\}$.

De karakterisaties van deze stelling zijn niet erg effectief; daarmee bedoel ik dat ze niet laten zien hoe je - in een eindig aantal stappen - kunt nagaan of een gegeven deelverzameling B van T een Gröbner basis is. Om dat doel te bereiken hebben we de volgende constructie nodig. Het *S-polynoom van twee veeltermen* $f,g \in T$, notatie $S(f,g)$, is

$$S(f,g) = \begin{cases} 0 & \text{als } f=0 \text{ of } g=0 \\ kgv(Lm_f, Lm_g)(Lc_g Lm_f^{-1} f - Lc_f Lm_g^{-1} g) & \text{anders} \end{cases}$$

STELLING. (Effectieve karakterisatie van Gröbner bases). *Laat* $<$ *een reductie-orde op de verzameling monomen in de veeltermring T zijn. Dan zijn equivalent:*
(i) *B is een Gröbner basis;*
(ii) *als* $b,b' \in B$, *dan is* 0 *een normaalvorm van* $S(b,b')$.

Een Gröbner basis van een (eindig voortgebracht) ideaal I van de veeltermring $T = R[X_1, \ldots, X_n]$ stelt ons nu in staat om elementen van T/I eenduidig weer te geven door elementen van T. Maar bijvoorbeeld ook om te beslissen of twee idealen samenvallen. Maar er is nog geen bijectieve correspondentie tussen idealen en Gröbner bases.

VOORBEELD. (Meerdere Gröbner bases m.b.t. tot dezelfde orde, van het zelfde ideaal). Neem $T = R[X,Y]$, $T = R[X,Y]$, en $B^{(i)} = \{X - Y^{1+3i},\ Y^3 - 1\}$. Dan is $I = (B^{(i)})$ onafhankelijk van de keuze van $i \in \mathbb{N}$. Elke $B^{(i)}$ is een Gröbner basis van I m.b.t. de lexicografische orde op $\mathbf{M}$ (waarbij $X > Y$):

$$\begin{aligned} S(X - Y^{1+3i}, Y^3 - 1) &= -(X - Y^{1+3(i+1)}) \Rightarrow_{B^{(i)}} (X - Y^{1+3(i+1)}) - (X - Y^{1+3i}) \\ &= Y^{1+3i}(1 - Y^3) \Rightarrow_{B^{(i)}}^* 0. \end{aligned}$$

Dit euvel is ook te verhelpen: laat I een ideaal van T zijn. We zeggen dat B een *gereduceerde Gröbner basis van I* is als het een Gröbner basis is en als elke $b \in B$ hoogste-monoom-ëfficiënt $Lc_b = 1$ heeft en in normaalvorm is m.b.t. $B \setminus \{b\}$.

STELLING (Gereduceerde Gröbner Bases, cf. [2]). *Laat I een ideaal in T zijn, en* $<$ *een reductie-orde op* $\mathbf{M}$. *Dan heeft I een unieke gereduceerde Gröbner basis B. Deze basis is een minimale Gröbner basis van I (in de zin dat geen deelverzameling van B ook een Gröbner basis van I is). Bovendien geldt* $|\{Lm_b | b \in B\}| = |B|$.

De bovenstaande stellingen leveren ook een algorithme om de Gröbner basis te vinden.

ALGORITHME. *GBasis*(B): Gegeven B lever een Gröbner basis van (B) af

$P:=\{\{b,b'\}|b,b'\in B\}$;
while $P\neq\varnothing$
do choose $\{b,b'\}\in P$;
$P:=P\setminus\{\{b,b'\}\}$;
$c:=Normaalvorm(B,S(b,b'))$;
if $c\neq 0$
then $P:=P\cup\{\{b,c\}|b\in B\}$;
$B:=B\cup\{c\}$;
fi
od;
return B

De toepassingen van de Gröbner basistheorie zijn legio. Wij vermelden nog slechts

STELLING (A. van Essen [7]). *Laat* $f:R^n\to R^n$ *een polynomiale afbeelding zijn gegeven door* $f(X)=(f_1(X),\ldots,f_n(X))$, *met* $f_i(X)\in T$ *voor elke* i. *Kies een reductie-orde op* $R[X,Y]$, *waar* $Y=(Y_1,\ldots,Y_n)$, *met* $Y_1<Y_2<\cdots<Y_n<X_1<X_2<\cdots<X_n$. *Schrijf* $B=\{Y_i-f_i(X)|1\leqslant i\leqslant n\}$. *Dan is* f *inverteerbaar dan en slechts dan als er* $g_{i(Y)}$ $(1\leqslant i\leqslant n)\in R[X,Y]$ *bestaan zodanig dat* $\{X_i-g_{i(Y)}|1\leqslant i\leqslant n\}$ *de gereduceerde Gröbner basis is van* (B). *Als* f *bovendien inverteerbaar is, dan wordt haar inverse gegeven door* $Y\mapsto(g_1(Y),\ldots,g_n(Y))$.

De Gröbner-basisstelling beweert weliswaar dat een bepaalde klasse van problemen oplosbaaar is, maar dat wil dan nog niet zeggen dat, voor veeltermen van enige omvang, de berekeningen ook op de ons ter beschikking staande computers uitvoerbaar zijn.

7. COMPLEXITEIT

Tot nu toe heb ik aandacht geschonken aan problemen waar het à priori niet duidelijk is dat er een oplossing is. Maar er zijn ook problemen waar niemand er zelfs over denkt deze vraag te stellen, zoals de vermenigvuldiging van twee gehele getallen. Daar spelen anderssoortige problemen zoals: hoe snel kun je twee getallen vermenigvuldigen. We komen dan op het terrein van de complexiteit; zie [1] en [10] voor boeiende introducties. Enkele resultaten van oorspronkelijk theoretische aard zijn al tot standaard-technieken voor programmatuur-pakketten verwerkt: de snelle (probabilistische) primaliteitstesten en factorisaties voor getallen, de Discrete Fourier Transformatie voor het snel vermenigvuldigen van veeltermen, de zogenaamde L(ower)U(pper-diagonal)P(ermutation matrix)-decompositie voor het snel vermenigvuldigen en inverteren van matrices; zie bijv. [4] voor een inleiding.

De complexiteit is naar mijn smaak juist zo boeiend omdat oorspronkelijk en constructief werk nodig is om algorithmen te verzinnen die sneller gaan dan bestaande.

De complexiteit van het Gröbner basis algorithme is berucht groot. Voor problemen met betrekking tot veeltermen van iets meer speciale aard zijn vele andere methoden (zoals varianten op de klassieke resultanten) bekend en vooral ook denkbaar, die nog maar weinig op hun complexiteit onderzocht zijn.

8. Techniek

Afgezien van de theoretische vragen die computeralgebra opwerpt, behoort ook een zekere vorm van techniek tot de computeralgebra: het bouwen van een pakket om goed bevonden algorithmen daadwerkelijk te realiseren en voor groter publiek ter beschikking te stellen. Het afgelopen jaar zijn er in Nederland tenminste twee van dergelijke pakketten geconstrueerd. Het ene pakket is FORM, een pakket van J. Vermaseren (van het FOM-instituut NIKHEFH) dat zich specialiseert in het snel uitvoeren van grote veelterm-bewerkingen en voor de hogere energie fysica relevantie formule - berekeningen. Het pakket FORM kenmerkt zich door de optimalisatie van tijd en geheugen tegenover grotere standaard-pakketten.

Ten tweede is daar een gespecialiseerd pakket, Lie geheten, dat het laatste jaar op het CWI ontwikkeld is ten behoeve van Lie- en Weyl-groep-theoretische berekeningen. Enkelvoudige Liegroep-voorstellingen kunnen geparametriseerd worden door vectoren met niet-negatieve heeltallige coëfficiënten, de zogenaamde dominante gewichten. De bij gewicht $[a,b]$ horende voorstelling van de Liegroep van 3×3 matrices met determinant 1 heeft dimensie $A2(a,b)$, waar $A2$ gedefiniëerd is als in §2. Dit volgt uit Weyl's dimensie-formule. De dimensie van zo'n voorstelling wordt in het pakket Lie uitgerekend met de oorspronkelijke vorm van Weyl's formule, die geldig is voor alle simpele Liegroepen. Het is op grond van deze formule in te zien dat de functie

$$x,y \mapsto \sum_{a,b\geqslant 0} A2(a,b)x^a y^b$$

rationaal in x en y is. De vraag die deze observatie opwerpt is nu: hoeveel specifieke waarden voor een gegeven simpele groep moet je doorrekenen om vast te stellen wat deze rationale functie is. Voor dit voorbeeld is de vraag niet al te moeilijk, maar als we de vraag in analoge situaties voor tensorproducten en restricties tot Lie-ondergroepen (in plaats van dimensies van voorstellingen) herhalen, is het antwoord (mij) onbekend). Zo kan het bouwen aan de praktische kant weer theoretische vragen oproepen. Voor meer interacties tussen wetenschap en technische beschouwingen, zie [6], het enige (mij bekende) boek dat de titel computeralgebra draagt.

Nog een voorbeeld. De standaard-methoden om het spoor van diagonaliseerbare elementen in Liegroep-voorstellingen te bepalen, vereisen de constructie van banen van vectoren onder de Weylgroep. De Weylgroep van de simpele Liegroep van $n\times n$-matrices met determinant 1 is de groep van alle

permutaties van n letters. Hiervan gebruik makend was het mogelijk de banen van deze Weylgroepen bijzonder snel en met gering geheugenbeslag af te lopen. Een soortgelijk idee kan voor Weylgroepen van alle klassieke typen A_n, B_n, C_n, D_n toegepast worden. Sterker nog, ook voor een Liegroep als die van type E_8 kon in LIE, door gebruikmaking van de nevenklassen van de ondergroep D van de Weylgroep W van type E_8 als gegeven in §4 en het aflopen van D-banen op de efficiënte manier (D is een Weylgroep van type D_8), een aanzienlijke verbetering bereikt worden. Techniek...

LITERATUUR

1. A. BORODIN, I. MONRO (1975). *The Computational Complexity of Algebraic and Numeric Problems,* Elsevier, Amsterdam.
2. B. BUCHBERGER (1985). *Gröbner Bases: An Algorithmic Method in Polynomial Ideal Theory,* Multidimensional System Theory, N.K. BOSE (ed.). Reidel, Dordrecht, 184-232.
3. ARJEH M. COHEN (1988). *Mathematical Formula Manipulation from a User's Point of View,* CWI Quarterly, 1(3), 53-63.
4. ARJEH M. COHEN, BERT RUITENBURG (1989). *Algebra and Computer Algorithms,* collegedictaat in voorbereiding.
5. D. COLLINS (1989). *Unsolvable Decision Problems,* een brief, 1-8.
6. J.H. DAVENPORT, Y. SIRET, E. TOURNIER (1988). *Computer Algebra, Systems and Algorithms for Algebraic Computation,* Acad. Press, London.
7. A. VAN ESSEN (1986). A criterion to deduce if a polynomial map is invertible and to compute its inverse, rapport Kath. Univ. Nijmegen, 1-6.
8. J.E. HUMPHREYS (1972). Introduction to Lie algebras and representation theory, Springer, New York.
9. J.W. KLOP, A. MIDDELDORP (1988). An introduction to Knuth-Bendix Completion. CWI Quarterly 1(3), 31-52.
10. D.E. KNUTH (1973). *The Art of Computer Programming Vol 3,* Addison-Wesley, Reading MA.
11. M. HAZEWINKEL (1988). *Encyclopaedia of Mathematics, Vol 1 A-B, An updated and annotated translation of the Soviet 'Mathematical Encyclopaedia',* Reidel, Dordrecht.
12. C.C. SIMS (1989). *Computation with Finitely Presented,* Groups preliminary draft of a book
13. M. SUZUKI (1982). Group Theory I, Springer Verlag, Berlin.

Ik bedank Bart de Smit voor zijn correcties van deze tekst tijdens de samenstelling ervan.

31 januari 1989

Toegepaste wiskunde en de computer

Toegepaste Wiskunde en Computer

A. van der Sluis
Mathematisch Instituut
Postbus 80010, 3508 TA Utrecht
The Netherlands

INLEIDING

In 1950 verscheen in het in 1946 opgerichte tijdschrift Mathematical Tables and other Aids to Computation de volgende mededeling van de redactie (MTAC 4 (1950), p. 39).

> Notes on Modern Numerical Analysis—I
>
> EDITORIAL NOTE: There is a general feeling that, once the problems of construction and maintenance of automatic digital computing machines are solved, the remaining problems will be relatively simple. This may be the case if attention is confined to standard classical problems; however, if an attempt is made to use these machines fully, one is likely to encounter formidable mathematical difficulties. It is expected that these difficulties will be discussed in the current mathematical journals; but there are also smaller, more technical problems which may cause trouble. It is believed that a discussion of these smaller problems will prove beneficial in avoiding a great many difficulties which are expected to arise when the machines are in actual operation; and we should like to urge interested persons to submit technical notes of this nature for future publication in the Automatic Computing Machinery Section of *MTAC*. These notes could be by-products of or preliminaries to more constructive investigations. It would be a great advantage, for expository purposes, if the authors, even at the expense of a choice of an extravagant example, could exhibit the troubles under discussion on a manual scale.

De tak van wiskunde die zich met de aangegeven problematiek ging bezig

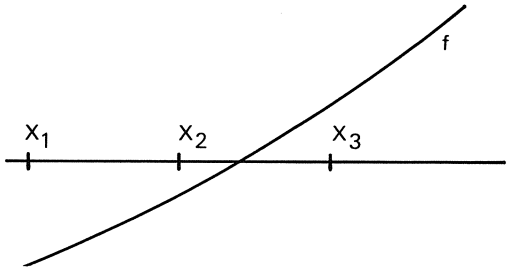
houden is de numerieke wiskunde, en niet alleen in het genoemde tijdschrift (dat overigens naderhand Mathematics of Computation is gaan heten), maar ook in een flink aantal nieuw opgerichte tijdschriften zijn inmiddels vele artikelen van de bedoelde soort verschenen.

Het is dan ook niet verwonderlijk dat men, wanneer men in de toegepaste wiskunde getalmatige uitkomsten voor problemen wil hebben, al gauw tegen de numerieke wiskunde aan loopt. Aangezien Uw spreker numericus is zal dit stuk aandacht geven aan enkele aspecten van de numerieke wiskunde. Aangezien deze beschouwing op een 'algemeen publiek' gericht moest zijn, zal ik, gelet op bovenstaande *editorial note,* beginnen met enkele voorbeelden van situaties waarbij men in het geheel niet verdacht is op problemen. Waar deze *editorial note* onmiddellijk gevolgd werd door een artikel getiteld: Solution of differential equations by recurrence relations (door J. Todd), en differentiaalvergelijkingen van oudsher een belangrijk onderwerp vormen binnen de toegepaste wiskunde, leek het mij aardig het grootste deel van mijn betoog aan dit onderwerp te wijden.

Het is overigens zelfs op dit beperkte gebied onmogelijk naar volledigheid te streven. Ik zal trachten U enig gevoel te geven van wat er bij deze processen gebeurt. Centraal in deze beschouwingen zal het begrip *stabiliteit* staan, omdat stabiliteitsproblemen de niet-numericus vaak voor de meest onverwachtse problemen stelt.

1. INSTABILITEIT a,b,c-FORMULE

Een der eerste ervaringen op mijn pad als numericus betrof een probleem van een fysicus, die van een functie f in een aantal punten de waarde kende (zie Fig. 1) en het nulpunt wenste te bepalen. Het functieverloop was behoorlijk lineair, zodat lineaire interpolatie op de punten x_2,x_3 al een behoorlijk goede benadering zou geven, maar hij wou nog iets meer, en legde een kwadratische functie door 3 punten, en loste de nu ontstane vierkantsvergelijking op met de a,b,c-formule. Een der wortels lag ver weg (noemen we hem λ_1), maar de andere (λ_2) lag niet tussen x_2 en x_3. Hoe kon dat?



FIGUUR 1

Bekijken we de a,b,c-formule

$$\frac{-b\pm\sqrt{b^2-4ac}}{2a} \tag{1.1}$$

voor het geval $|\lambda_1|>>|\lambda_2|$. Wegens $\frac{b}{a}=-(\lambda_1+\lambda_2)$ geldt $|\frac{b}{2a}|>>|\lambda_2|$, zodat $\sqrt{b^2-4ac}\approx|b|$. Een relatieve fout ϵ in $\sqrt{b^2-4ac}$ geeft dan een fout $\approx|\frac{\epsilon b}{2a}|$ in de wortels, en dus een relatieve fout $\approx|\frac{\epsilon b}{2a\lambda_2}|\approx|\frac{\epsilon}{2}\frac{\lambda_1}{\lambda_2}|$ in λ_2. Dit neemt kennelijk onbegrensd toe naarmate de functie meer lineair wordt, en het probleem dus eigenlijk eenvoudiger!

Dit is een klassiek (in meerdere betekenissen van het woord: de *a,b,c*-formule is al 5000 jaar bekend) voorbeeld van een instabiliteitsprobleem: de uitkomst van een rekenproces wordt verknoeid door fouten gemaakt tijdens het rekenproces die op het moment dat ze gemaakt werden als klein konden worden aangemerkt.

De *a,b,c*-formule is dus *instabiel* voor de absoluut kleinste wortel wanneer de wortels zeer verschillende grootte orde hebben. Hij is niet instabiel voor de absoluut grootste wortel: hierin ontstaat een relatieve fout $\approx|\frac{\epsilon b}{2a\lambda_1}|\approx\frac{\epsilon}{2}$. Dit levert ons meteen de uitweg: $\frac{c}{a\lambda_1}$ geeft dan ook een waarde voor λ_2 met een kleine relatieve fout.

Zo komt men dus (na 5000 jaar) op de volgende variant van de *a,b,c*-formule voor het stabiel berekenen van de absoluut grootste wortel λ_1 en de absoluut kleinste wortel λ_2:

$$\lambda_1 = \frac{-b-\mathrm{sgn}(b)\sqrt{b^2-4ac}}{2a} \tag{1.2}$$

$$\lambda_2 = \frac{2c}{-b-\mathrm{sgn}(b)\sqrt{b^2-4ac}} \tag{1.3}$$

Men ziet dat (1.3) ook nog goed is voor $a=0$, dus voor het puur lineaire geval.

De moraal van dit verhaal is dat wiskundig equivalente processen numeriek allesbehalve equivalent hoeven te zijn, en dat men zelfs in zeer alledaagse situaties op zijn hoede moet zijn voor de stabiliteitsval.

2. Instabiliteit bij recursies

Een andere alledaagse situatie waarbij men heel snel tegen stabiliteitsproblemen aan loopt wordt gevormd door recursies.

Als voorbeeld bekijken we

$$a_n = \int_0^1 t^n e^{-t}\,dt \tag{2.1}$$

(een incomplete Γ-functie), te berekenen voor $n=0,1,\ldots,10$. Dit is eigenlijk een opgave voor het eerstejaars analyse practicum:

$$a_0=1-e^{-1},\quad a_n=na_{n-1}-e^{-1},\quad n\geqslant 1. \tag{2.2}$$

Neemt men echter $e^{-1}=0.3679$ (correct afgerond), dan vindt men, exact

rekenend, $a_7 = -0.23$, hetgeen duidelijk fout is omdat $a_n > 0$ voor alle n. De oorzaak van dit foute antwoord is dat een fout in a_0 na n recursieslagen met $n!$ vermenigvuldigd in a_n doorwerkt (in de eerste, tweede, derde ... recursieslag wordt overigens opnieuw de foute waarde van e^{-1} gebruikt, waardoor het effect nog versterkt wordt).

Opnieuw dus een voorbeeld van een kleine fout tijdens het proces (nl. die in e^{-1}) die het uiteindelijke antwoord verknoeit. Het is duidelijk dat de fout opbouw zodanig is dat men ook bij gebruik van een veel nauwkeuriger waarde van e^{-1} toch niet tot veel hogere waarden van n komt.

Hoe moet het dan wel? We merken op dat de recursie teruglopend luidt:

$$a_{n-1} = \frac{1}{n}(a_n + e^{-1}), \tag{2.3}$$

zodat een eenmaal gemaakte fout nu drastisch gedempt wordt. Nemen we nu bijv. $\tilde{a}_{20} = 0$, dan geldt $0 < a_{20} - \tilde{a}_{20} < \int_0^1 t^{20} dt = \frac{1}{21}$. Gaat men nu, startend met $\tilde{a}_{20} = 0$, de recursie (2.3) in dan krijgt men $\tilde{a}_n$ waarvoor $0 < a_n - \tilde{a}_n < \frac{1}{21 \cdot 20 \cdot \cdots \cdot (n+1)}$. Wegens $a_n > \frac{e^{-1}}{n+1}$ geven de $\tilde{a}_n$ dus al heel spoedig relatief zeer nauwkeurige benaderingen voor de a_n (natuurlijk moet men bij een preciese analyse ook nog de fout in e^{-1} en de afrondfouten onderweg meenemen; wij gaan daaraan nu voorbij).

Recursies die in voorwaartse richting instabiel zijn zijn niet noodzakelijk in terugwaartse richting stabiel. Bijvoorbeeld de recursie

$$a_n - 3\tfrac{1}{2}a_{n-1} + 3\tfrac{1}{2}a_{n-2} - 1 = 0 \tag{2.4}$$

heeft als oplossingen de lineaire combinaties van de basisoplossingen $(b_n), (c_n), (d_n)$: $b_n = 2^n$, $c_n = 1$, $d_n = 2^{-n}$. Omdat de recursie homogeen is (d.w.z. rechterlid 0) geschiedt foutvoorplanting via dezelfde recursie. Voor de oplossing (b_n) groeit het effect van fouten dus evensnel aan als de oplossing zelf, zodat de recursie stabiel is voor deze oplossing. Voor de oplossing (d_n) is dit niet het geval omdat deze oplossing snel daalt terwijl fouten nog steeds snel stijgend worden voortgeplant. Om precies dezelfde reden is de terugwaartse recursie juist instabiel voor (b_n) en stabiel voor (d_n). Voor de oplossing (c_n) is de recursie echter instabiel in beide richtingen.

Dit soort overwegingen heeft geleid tot methoden om de recursie en de oplossingsruimte te splitsen in componenten waarvoor de voorwaartse recursie stabiel is en in componenten waarvoor de terugwaartse recursie stabiel is. Een en ander is o.a. van belang voor het numeriek oplossen van differentiaalvergelijkingen met randwaardeproblemen. Voor een uitvoerige behandeling zie [2].

3. Gewone differentiaalvergelijkingen, het begin (Adams)

Er kunnen nog veel meer voorbeelden genoemd worden van problemen die, mathematisch gesproken, in eindige termen oplosbaar zijn, maar waarvoor deze mathematische oplossingswijze, recht toe recht aan in een computer gestopt,

geen goede resultaten oplevert of veel te inefficient is (men kan hierbij o.a. denken aan stelsels lineaire vergelijkingen).

Richten wij ons nu echter op een klasse van problemen die i.h.a. niet in eindige termen zijn op te lossen omdat ze principieel door een limietproces gedefineerd zijn. We houden ons verder bezig met (stelsels) gewone differentiaalvergelijkingen

$$u'(t) = f(t,u(t)), \quad u(t_0) = u_0, \tag{3.1}$$

waarin u en f vectorwaardige functies zijn. Aangezien de theorie hiervoor echter net zo loopt als voor scalaire functies zullen we ons hoofdzakelijk tot het scalaire geval beperken.

Het verhaal vindt zijn oorsprong in onderzoekingen van F. Bashforth betreffende de vorm van druppels die op een horizontaal vlak rusten dat zij niet bevochtigen. In een aanvraag van 1855 aan de Royal Society om £50 rekengeld (!) bericht hij (zie [3, p.1]) dat J.C. Adams

> 'furnished me with a perfectly satisfactory method of calculating by quadratures the theoretical forms of drops of fluid from the differential equation of Laplace'.

De methode van Adams bestaat hieruit dat men in de rij punten (t_n): $t_n = t_0 + nh$, $h>0$ willekeurig, een rij benaderende oplossingswaarden (u_n) definieert d.m.v. de relatie

$$u_n = u_{n-1} + \int_{t_{n-1}}^{t_n} \phi_n(t)dt \tag{3.2}$$

waarin ϕ_n het interpolatie polynoom is zodat $\phi_n(t_{n-i}) = f(t_{n-i},u_{n-i})$, $i = 1, \ldots, k$ voor zekere vaste k. Dit komt neer op een formule

$$u_n = u_{n-1} + h\sum_{1}^{k} b_i f(t_{n-i},u_{n-i}) \tag{3.3}$$

met b_i onafhankelijk van h en f. Adams spreekt hier echter helemaal niet over, maar drukt de integraal in (3.2) uit in eindige differenties van de genoemde f-waarden.

Hij kijkt ook eens naar de betrekking

$$u_n = u_{n-1} + \int_{t_{n-1}}^{t_n} \bar{\phi}_n(t)dt \tag{3.4}$$

met $\bar{\phi}_n$ het interpolatiepolynoom zodat $\bar{\phi}_n(t_{n-i}) = f(t_{n-i},u_{n-i})$, $i = 0, \ldots, k-1$, hetgeen neerkomt op een formule

$$u_n = u_{n-1} + h\sum_{0}^{k-1} \beta_i f(t_{n-i},u_{n-i}) \tag{3.5}$$

met β_i onafhankelijk van h en f. Ook dit komt bij Adams (uiteraard) niet voor

maar wel constateert hij dat bij het uitdrukken van de integraal in (3.4) in eindige differenties de coefficienten hiervan veel sneller afnemen dan in (3.2) (wat niet verwonderlijk is omdat het interpolatiepolynoom in (3.2) geïntegreerd wordt buiten het steunpunteninterval en in (3.4) juist daarbinnen), en dat (3.4) dus een betere nauwkeurigheid geeft.

Hij gebruikt daarom (3.2) helemaal niet, maar bepaalt door interpolatie (of eigenlijk extrapolatie) uit $u_{n-1}, \ldots, u_{n-k}$ een benaderingswaarde $\tilde{u}_n$ voor u_n, en doet, daarmee startende één Newton-Raphson slag op de (i.h.a. niet-lineaire) vergelijking in u_n die (3.4) (en dus (3.5)) bij gegeven $u_{n-1}, \ldots, u_{n-k}$ voorstelt. De aldus verkregen benaderingswaarde voor u_n wordt nu geaccepteerd als de waarde van u_n waarmee verder gerekend wordt bij het bepalen van u_{n+1}, etc.

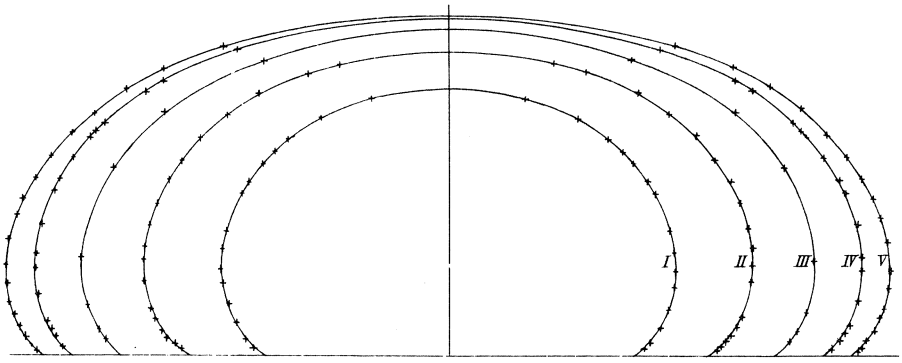
Ten aanzien van de keuze van k en h merkt hij op (p. 18):

> 'It will usually be found expedient to choose h so small as to render it unnecessary to proceed beyond fourth order differences.'

(hij neemt dus in feite $k=4$), en verderop (p. 29):

> 'We may, of course, change the value of h whenever the more or less rapid rate of diminution of the successive differences shews that it is expedient to increase or diminish the interval.'

Van enig kwantitatief criterium, laat staan van enige theorievorming, was echter geen sprake. Wel kreeg hij fraaie resultaten, zoals Fig. 2 toont (hierin stellen de getrokken krommen de berekende vormen voor van kwikdruppels die op een vlak rusten en de kruisjes gemeten punten).



FIGUUR 2

4. THEORIEVORMING

Theorievorming kwam pas op gang in de vijftiger jaren, toen Dahlquist de algemenere klasse van *lineaire multistepmethoden*

$$u_n+\alpha_1 u_{n-1}+\cdots+\alpha_k u_{n-k} = h[\beta_0 f(t_n,u_n)+\cdots+\beta_k f(t_{n-k},u_{n-k})] \qquad (4.1)$$

ging onderzoeken. Het aantrekkelijke van deze methoden is op het eerste gezicht dat men bij gelijke k veel meer parameters ter beschikking heeft (nl. k $\alpha's$ en $k+1$ $\beta's$, terwijl bij Adams de $\alpha's$ vast liggen). Hierdoor zou men dus een twee maal zo hoge orde van nauwkeurigheid kunnen afdwingen, waarbij opgemerkt zij dat de orde van een multistep gedefinieerd is als het kleinste getal p zodat, als men in (4.1) u_i vervangt door $u(t_i)$, het verschil van linker- en rechterlid te schrijven is als

$$\gamma u^{(p+1)}(t_n)h^{p+1}+O(h^{p+2}), \tag{4.2}$$

zulks voor alle differentiaalvergelijkingen met voldoend gladde oplossingen, waarbij $\gamma\neq 0$, γ onafhankelijk van h en f, en de O-term uniform voor t_n in een compactum.

Dahlquist bewees de beroemde *convergentiestelling* (zie [5]; ook [6] of [14]):

STELLING 1. Noodzakelijk en voldoende opdat voor elke continue en u-Lipschitz continue f en voor elke startprocedure waarvoor $u_0(h),\ldots,u_{k-1}(h)\to u_0$ als $h\to 0$ de oplossing van (4.1) voor $h\to 0$ bestaat en uniform naar u convergeert op elk compactum $[t_0,b]$ waarop u bestaat is dat

(i) de vergelijking

$$x^k+\alpha_1 x^{k-1}+\cdots+\alpha_k = 0 \tag{4.3}$$

uitsluitend wortels binnen of op de complexe eenheidscirkel heeft en de wortels op de eenheidscirkel enkelvoudig zijn;

(ii) de multistep minstens orde 1 heeft, dwz. dat

$$\begin{cases} 1+\alpha_1+\cdots+\alpha_k = 0 \\ k+(k-1)\alpha_1+\cdots+\alpha_{k-1} = \Sigma\beta_i. \end{cases} \tag{4.4}$$

□

In deze stelling is een eis als (ii) niet zo verrassend, want deze orde uitspraak zegt hoe goed de recursie de differentiaalvergelijking benadert. Eis (i) is wellicht verrassender. Om in te zien wat deze betekent merken we op dat men met (4.1) per recursiestap in feite een term van de orde van (4.2) verwaarloost (zelfs precies een term (4.2) als $\beta_0=0$), en de vraag is hoe de recursie zo'n fout voortplant. Bezien we hiertoe eens de multistep

$$u_n+4u_{n-1}-5u_{n-2} = h[4f(t_{n-1},u_{n-1})+2f(t_{n-2},u_{n-2})] \tag{4.5}$$

(die van de orde 3 is) en de eenvoudige differentiaalvergelijking

$$u'(t) = \cos(t). \tag{4.6}$$

De recursie luidt nu

$$u_n+4u_{n-1}-5u_{n-2} = h[4\cos(t_{n-1})+2\cos(t_{n-2})] \tag{4.7}$$

zodat fouten volgens oplossingen van de homogene recursie $u_n+4u_{n-1}-5u_{n-2}=0$ worden voortgeplant. Aangezien het bij deze homogene recursie behorende karakteristieke polynoom $\lambda^2+4\lambda-5$) de nulpunten -5 en

1 heeft, heeft de recursie de basisoplossingen $((-5)^n)$ en (1), zodat het effect van een eenmaal gemaakte fout (en dat geldt ook voor afrondfouten) al na een luttel aantal slagen fantastisch wordt opgeblazen, onafhankelijk van h. Eis (i) voorkomt dat het homogene deel van de recursie aangroeiende oplossingen heeft, en dus dat het effect van eenmaal gemaakte fouten al te zeer opblaast. Deze eis is derhalve een *stabiliteitseis.*

Dus kort samengevat: fouten moeten niet alleen klein zijn als ze gemaakt worden (hiervoor zorgt eis (ii)), maar ook redelijk voortgeplant worden (hiervoor zorgt eis (i)). Het is nu ook duidelijk dat als aan de eisen van de stelling niet voldaan is, het al voor heel eenvoudige differentiaalvergelijkingen daadwerkelijk mis gaat.

Tegelijkertijd met zijn convergentiestelling bewees Dahlquist zijn eerste *barrièrestelling*, waarmee hij de hoop de bodem inslaat dat met geschikte keuze van $\alpha's$ en $\beta's$ bij gelijke k een twee maal zo hoge orde kan worden verkregen als met Adams:

STELLING 2. Stelling 1, voorwaarde (i), impliceert

$$p \leqslant k+2 \quad \text{als} \quad k \quad \text{even is}$$
$$p \leqslant k+1 \quad \text{als} \quad k \quad \text{oneven is}$$
$$p \leqslant k \quad \text{als} \quad \beta_0 \leqslant 0 \qquad \square$$

De orde van een convergente multistep is dus nooit hoger dan $k+2$. Niettemin kan men van de beschikbare parameter ruimte gebruik maken om andere wenselijke eigenschappen te krijgen.

De theorie betreffende orde en convergentie van multisteps en varianten daarvan is inmiddels zeer uitgebreid (zie bijv. [17] en [19]). Een curieus resultaat is nog dat Dahlquist's barrièrestelling ontdoken kan worden door cyclisch gebruik van een aantal geschikt gekozen multisteps (zie [19, § 4.3]).

5. GROOTTE EN STRUCTUUR VAN DE BENADERINGSFOUT

De vraag is natuurlijk of je in praktijksituaties iets over de grootte van de benaderingsfout kunt zeggen. Daarvoor is het nuttig eerst iets theoretisch over de grootte en de structuur van deze fout te zeggen. Het aardige van deze foutentheorie is dat deze zich al heel goed aan heel eenvoudige gevallen laat demonstreren.

Als we noteren $\bar{u}_n = u(t_n)$, en in verband hiermee verder de oplossing van (3.1) ook maar met $\bar{u}$ aangeven, geldt de identiteit

$$\bar{u}_n - \bar{u}_{n-1} = hf(t_{n-1}, \bar{u}_{n-1}) + \frac{h^2}{2}\bar{u}''(\xi_n) \tag{5.1}$$

waaruit de methode van Euler volgt (die dus van orde 1 is):

$$u_n - u_{n-1} = hf(t_{n-1}, u_{n-1}). \tag{5.2}$$

Beschouwen we nu de differentiaalvergelijking

$$u'(t) = \lambda(t)u(t) + g(t), \quad u(t_0) = u_0 \tag{5.3}$$

dan geldt met (5.1) en (5.2):

$$\bar{u}_n - \bar{u}_{n-1} = h[\lambda_{n-1}u_{n-1} + g_{n-1}] + \tau_n, \quad \tau_n = \frac{h^2}{2}\bar{u}''(\xi_n). \tag{5.4}$$

$$u_n - u_{n-1} = h[\lambda_{n-1}u_{n-1} + g_{n-1}] \tag{5.5}$$

zodat we dus per stap van (5.5) a.h.w. een fout τ_n maken. Het is verleidelijk te denken dat de totale fout de som van al die τ_n's, is, maar dat is niet zo. Definiëren we de *globale fout* $e_n = \bar{u}_n - u_n$ dan krijgen we

$$e_n - e_{n-1} = h\lambda_{n-1}e_{n-1} + \tau_n \tag{5.6}$$

zodat

$$e_n = \tau_n + (1 + h\lambda_{n-1})e_{n-1} = \cdots = \tag{5.7}$$

$$= \sum_{p=1}^{n} \prod_{j=p}^{n-1} (1 + h\lambda_{n-j})\tau_p \tag{a}$$

$$= \sum_{p=1}^{n} [\exp(\int_{t_p}^{t_n} \lambda(s)ds) + O(h)]\tau_p \tag{b}$$

$$= \frac{h}{2}\int_{t_0}^{t_n} \exp(\int_{\xi}^{t_n} \lambda(s)ds)\bar{u}''(\xi)d\xi + O(h^2). \tag{c}$$

CONCLUSIE:

- de globale fout e_n is een gewogen som van de lokale fouten τ_i.
- de lokale fouten worden versterkt voortgeplant als $\lambda > 0$, verzwakt als $-2 < h\lambda < 0$ (zie (5.7(a)).
- de lokale fouten worden op $O(h)$ na voortgeplant volgens oplossingen van de homogene differentiaalvergelijking (zie de term met exp in (5.7(b)) en dat is een optimaal gedrag.
- de fout in een vast punt t is te schrijven als

$$e(t) = h\phi(t) + O(h^2) \tag{5.8}$$

(zie (5.7(c)) met

$$\phi(t) = \frac{1}{2}\int_{t_0}^{t} \exp(\int_{\xi}^{t} \lambda(s)ds)\bar{u}''(\xi)d\xi. \tag{5.9}$$

Deze ϕ is dus in feite de oplossing van

$$\phi'(t) = \lambda(t)\phi(t) + \frac{1}{2}\bar{u}''(t), \quad \phi(t_0) = 0. \tag{5.10}$$

De fouten hangen dus op een gladde wijze van de plaats af.

Een en ander geldt op overeenkomstige wijze voor alle multisteps die aan de condities (i) en (ii) van stelling 1 voldoen en voor differentiaalvergelijkingen (ook niet lineaire) met voldoend gladde oplossingen. Voor multisteps van orde p krijgt men zo

$$e(t) = h^p\phi(t)+O(h^{p+1}) \tag{5.11}$$

met

$$\phi'(t) = \frac{\partial f}{\partial u}|_{(t,\bar{u}(t))}\cdot\phi(t)+\gamma\bar{u}^{(p+1)}(t), \quad \phi(t_0)=0 \tag{5.12}$$

met γ als in (4.2) (zie [14], maar eigenlijk is het ook al bij Dahlquist te vinden).

6. Standaard software; foutschatting in de praktijk

Uiteraard zullen deze theoretische beschouwingen moeten uitmonden in algemeen bruikbare programmatuur, want computer gebruikers wensen voor standaard problemen (en zo kan men gewone differentiaalvergelijkingen toch wel aanduiden) standaard programmatuur die hen in staat stelt hun problemen efficiënt en met een gegeven tolerantie op te lossen.

Grote programmabibliotheken zoals NAG (Numerical Algorithms Group) en IMSL (International Mathematical Subroutine Library) bevatten programma's (zelfs meerdere) om differentiaalvergelijkingen op te lossen en we bezien eens hoe die werken.

In de eerste plaats schatten ze de lokale fout (τ_n, in ons Euler geval). Wegens $\bar{u}'(t_n)=f(t_n,\bar{u}(t_n))$ zou men kunnen hopen (alweer in ons Euler geval) $\tau_n=\frac{1}{2}h^2\bar{u}''(\xi_n)$ te kunnen schatten m.b.v.

$$\bar{u}''(t_n) \approx \frac{1}{h}[f(t_n,u_n)-f(t_{n-1},u_{n-1})] \tag{6.1}$$

en bijv. in [18, p. 27] gebeurt dat ook gewoon. Maar dat mag natuurlijk alleen maar als de u_i voldoend goede benaderingen zijn van $\bar{u}_i$, en dat is, dankzij (5.8), ook zo:

$$\begin{aligned}\bar{u}''(\xi_n) &= \frac{1}{h}[f(t_n,\bar{u}_n)-f(t_{n-1},\bar{u}_{n-1})]+O(h)= \\ &= \frac{1}{h}[f(t_n,u_n)-f(t_{n-1},u_{n-1})]+\frac{\partial f}{\partial u}\phi(t_n)-\frac{\partial f}{\partial u}\phi(t_{n-1})+O(h) \\ &= \frac{1}{h}[f(t_n,u_n)-f(t_{n-1},u_{n-1})]+O(h).\end{aligned} \tag{6.2}$$

Hierin speelt dus niet alleen de grootte van de globale fouten e_n een rol, maar ook hun 'gladheid', en een en ander geldt alleen maar als de functie $\bar{u}$ glad genoeg is. Ook dit geldt algemener m.b.v. (5.10).

Aangezien voor voldoend gladde functies de O-term in (5.11) nog weer te schrijven is als $h^{p+1}\psi(t)+O(h^{p+2})$ kan men ook d.m.v. numerieke differentiatie op de u_i of de u_i en de $f(t_i,u_i)$ de $\bar{u}^{(p+1)}$ in τ_n schatten. In de praktijk doet men dit m.b.v. combinaties van de u_i en $f(t_i,u_i)$ die men toch al heeft (de zgn. predictor - corrector techniek, zie bijv. [19, Thm. 4.5.11]).

De fout per stap schatten gaat dus wel, alhoewel het bitse commentaar van Byrne en Hindmarsh dat we in §8 citeren tenminste ten dele ook hier van toepassing is. Maar te schatten hoe die lokale fouten worden voortgeplant is een heel andere zaak: daarvoor moet men in feite de homogene vergelijking (of

in het niet lineaire geval de variatievergelijking) oplossen, en denkt U nu ook even aan stelsels. In standaard software doet men dat niet. Daarmee komt dan tevens het bepalen van een toelaatbare τ_n, en daarmee van een toelaatbare stapgrootte, in de lucht te hangen. Men behelpt zich door de gebruiker een grootheid TOL (tolerantie) op te laten geven die de stapgrootte bestuurt.

Zo lezen we bij het IMSL programma DGEAR, uitgave 1982:

> 'The error which is controlled by way of the parameter TOL is an estimate of the local truncation error, that is, the error committed on taking a single step with the method, starting with data regarded as exact. This is to be distinguished from the global truncation error, which is the error in any given computed value of $y(x)$ as a result of the local truncation errors from all steps taken to obtain $y(x)$. The latter error accumulates in a non-trivial way from the local errors, and is neither estimated nor controlled by the routine. Since it is usually the global error that a user wants to have under control, some experimentation may be necessary to get the right value of TOL to achieve the users needs. If the problem is mathematically stable, and the method used is appropriately stable, then the global error at a given x should vary smoothly with TOL in a monotone increasing manner.'

Dus: men streeft er naar de fout per stap onder TOL te houden. Als (wat gelukkig vaak voorkomt) de differentiaalvergelijking stabiel is (wat betekent dat de homogene of de variatievergelijking alleen dalende oplossingen heeft), dan is de totale fout hoogstens de som van alle lokale fouten; evenwel, wat de som van die fouten is of hoeveel van die fouten er worden gemaakt, dat wordt de gebruiker niet gewaar. Hoe komt deze er dan achter hoe nauwkeurig de oplossing is? Daarvoor dient waarschijnlijk de laatste volzin in het citaat: draai het programma eerst eens met een zekere TOL en ook eens met een $10\times$ zo kleine TOL; dan zal de fout in de met de kleinste TOL verkregen oplossing wel klein zijn t.o.v. het verschil der beide oplossingen. Bij het corresponderende NAG programma wordt dit zelfs expliciet geadviseerd, en in feite ook bij de 1987 editie van het IMSL programma, waarin ook een andere behandeling van TOL:

> 'An attempt is made to control the norm of the local error such that the global error is proportional to TOL. More than one run with different values of TOL can be used to estimate the global error. Generally it should not be greater than 0.001.'

De eerste volzin wekt de indruk dat TOL nu gebruikt wordt als een soort fout per lengteeenheid, wat mij op zich een goede zaak lijkt. Mocht dit waar zijn, dan zou in het stabiele geval de globale fout in het punt t dus omstreeks $(t-t_0)\cdot\text{TOL}$ zijn. Mededelingen hiervoor ontbreken echter. Het advies omtrent de bovengrens 0.001 voor TOL is curieus: dit lijkt te duiden op een

vooronderstelling van de programmeur over de grootte van de oplossing van de differentiaalvergelijking, dan wel op een relatief gebruik van TOL. Ik vind dat de gebruiker recht heeft op meer informatie.

Hoe dan ook, bij het rekenen aan reactoren of andere problemen met instabiele differentiaalvergelijkingen blijft het oppassen.

Tenslotte nog iets over de efficiency die de programmatuur moet nastreven. Dit wordt gerealiseerd door in het programma een hele familie multisteps op te nemen, zeg 1 tot 6 staps multisteps, en daaruit telkens die multistep te kiezen waarvoor bij de gegeven tolerantie eis de stapgrootte maximaal is. Dit orde veranderings mechanisme lost tevens het startprobleem voor multisteps op (d.w.z. het probleem dat een k-staps multistep k startwaarden nodig heeft terwijl de differentiaalvergelijking er maar één geeft): het programma begint gewoon met een 1-staps methode, en laat de orde dan stap voor stap toenemen.

Men moet overigens wel bedenken dat bij het veranderen van orde of stapgrootte de gladheidseigenschap van de globale fout, die voor de locale foutschatting nodig was (zie (6.2)), verloren kan gaan, evenals de stabiliteit. In [11] en [12] vindt men stellingen die ongeveer zeggen dat wanneer men bij een k staps Adams methode na stapgrootteverandering of ordeverandering met 1 gedurende k stappen geen stapgrootte- of ordeverandering toepast, convergentie en stabiliteit behouden blijven (men zie de betreffende artikelen voor de preciese uitspraken). Over de invloed op de lokale foutschatting vindt men niet veel (zie ook [4]).

7. Singuliere storingsproblemen/stijve differentiaalvergelijkingen

Bij het gebruik van een computer gaat het er, zoals we reeds eerder opmerkten, niet alleen om dat men een benaderende oplossing met een redelijke nauwkeurigheid krijgt, maar ook dat dit redelijk efficiënt gebeurt. Er zijn klassen differentiaalvergelijkingen waarbij dit laatste erg uit de hand kan lopen. Een zo'n klasse is die van singuliere storingsproblemen, waarbij het weer een stabiliteitskwestie is die dit veroorzaakt.

Bekijken we eens de modelvergelijking

$$u'(t) = \mu(\sin(t)-u(t))+\cos(t) \tag{7.1}$$

met als algemene oplossing $u(t)=\sin(t)+Ce^{-\mu t}$. Als $\mu>>1$ noemt men het probleem singulier gestoord. Alle oplossingen bewegen zich dan razendsnel naar de oplossing $u(t)=\sin(t)$. De oplossingsverzameling is dus erg inflexibel, vandaar dat men dergelijke differentiaalvergelijkingen ook wel *stijf* noemt.

Zij $u(0)=0$ (dus $\bar{u}(t)=\sin(t)$) en passen we Euler's methode (5.2) toe. Dan krijgen we (zie (5.7))

$$e_n = \tau_n+(1-h\mu)\tau_{n-1}+(1-h\mu)^2\tau_{n-2}+\cdots \tag{7.2}$$

en $\tau_n=\frac{h^2}{2}\bar{u}''(\xi_n)=-\frac{h^2}{2}\sin(\xi_n)$. Klaarbijkelijk moet nu $|1-h\mu|\leqslant 1$ als men niet wil dat de fouten onbegrensd opgeblazen worden, oftewel $h\leqslant\frac{2}{\mu}$. De

stabiliteitskwestie pakt hier dus heel onaangenaam uit: de lokale fouten τ_n hangen niet van μ af, en zijn bijv. voor $h=0.01$ behoorlijk klein, niettemin wordt men bij grote μ gedwongen tot onaanvaardbaar kleine h.

De vraag is of er methoden zijn die, hoe groot $h\mu$ ook is, fouten stabiel voortplanten. Men kan gemakkelijk inzien dat dit voor multisteps met $\beta_0=0$ niet kan. Men komt dus sowieso bij multisteps met $\beta_0\neq 0$ terecht, hetgeen eigenlijk niet zo prettig is omdat (4.1) bij gegeven $u_{n-1},\ldots,u_{n-k}$ een i.h.a. niet-lineaire vergelijking in u_n voorstelt (en denkt U weer aan stelsels). Maar dan is de wens ook te realiseren. Nemen we bijv. *Euler achterwaarts*:

$$\bar{u}_n-\bar{u}_{n-1} = hf(t_n,\bar{u}_n)+\tau_n,\quad \tau_n = -\frac{h^2}{2}\bar{u}''(\xi_n) \tag{7.3}$$

$$u_n-u_{n-1} = hf(t_n,u_n). \tag{7.4}$$

We krijgen nu voor onze differentiaalvergelijking (7.1)

$$e_n-e_{n-1} = -h\mu e_n+\tau_n \tag{7.5}$$

zodat

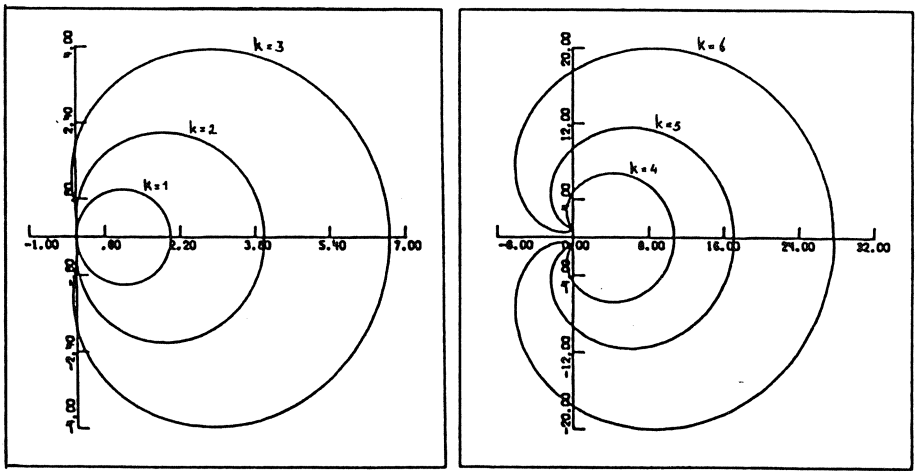
$$e_n = \frac{1}{1+h\mu}\tau_n+\left[\frac{1}{1+h\mu}\right]^2\tau_{n-1}+\cdots \tag{7.6}$$

Fouten worden nu dus zeer verzwakt voortgeplant, en we constateren met erkentelijkheid dat de fout kleiner is naarmate μ groter wordt. Niettemin moet worden vastgesteld dat deze methode maar lage orde heeft. Zijn er ook methoden van hogere orde?

Bekijken we algemener stelsels $u'(t)=f(t,u(t))$ en stel dat we willen dat voor alle eigenwaarden μ van de Jacobi matrix D_uf in het negatieve complexe halfvlak stabiele foutvoortplanting optreedt ongeacht de grootte van $h\mu$, dan is er weer een barrièrestelling (weer van Dahlquist, zie [7], ook [13, p. 218]) die zegt dat dit alleen maar kan met een hoogstens 2-de orde methode. Wanneer we onze eisen wat minder hoog stellen kunnen we echter verder komen. Gear ([8]) heeft multisteps van de gedaante

$$u_n+\alpha_1u_{n-1}+\cdots+\alpha_ku_{n-k} = h\beta_0f(t_n,u_n) \tag{7.7}$$

geconstrueerd van de orde k voor $k\leqslant 6$, waarbij stabiele foutvoortplanting plaats vindt als $h\mu$ links van de betreffende contour in Fig. 3 ligt. Bij deze multisteps is het linkerlid blijkbaar op een factor $h\beta_0$ na het numerieke differentiatie resultaat gebaseerd op Langrange interpolatie op u, vandaar dat men ze als BDF (backward differentiation formula) methoden aanduidt. Curieus is dat BDF formules voor $k\geqslant 7$ niet meer aan de algemene stabiliteitseis (i) in stelling 1 voldoen (zie bijv. [13, p. 135]). In de praktijk vindt men orde 6 echter zeer acceptabel.



FIGUUR 3

8. FOUTSCHATTING BIJ STIJVE DIFFERENTIAALVERGELIJKINGEN

Schatting van de lokale fout τ_n doet men in de praktijk op dezelfde wijze als in §6 beschreven, dus bijv. voor Euler achterwaarts (zie (7.4)) weer met (6.1) (of equivalent). Men moet zich echter realiseren dat hier een probleem zit. De in (6.2) gegeven argumentatie gaat nl. terug op (5.11) (en ook voor het stijve geval beroept men zich bijv. in [4] hier ook inderdaad op), en het gebruikelijke bewijs voor (5.11) levert daarin een O-term met een 'verborgen constante' van de gedaante $\exp(L(t-t_0))$, waarin L de Lipschitz constante van f is, zodat voor (7.1) geldt $L=|\mu|$ (zie bijv. [17, Theorem 9.1], waarbij het venijn zit in de aanroep van Theorem 8.13). Het lijkt dus in het stijve geval erg onwaarschijnlijk dat de O-term in (5.11) verwaarloosbaar is voor de h-waarden die men wenst te gebruiken, hetgeen de bedoelde foutschatting dus op losse schroeven zet.

Dat de zaak desalniettemin niet hopeloos is zullen we nu demonstreren voor de klasse vergelijkingen (waartoe ook (7.1) behoort)

$$u'(t) = -\mu(u(t)-g(t)), \quad u(t_0)=u_0^* \tag{8.1}$$

en Euler achterwaarts en uiteraard $h\mu \gg 1$. Weer met $e_n=\bar{u}_n-u_n$ geldt

$$\begin{aligned}\bar{u}''(\xi_n) &= \frac{1}{h}[f(t_n,\bar{u}_n)-f(t_{n-1},\bar{u}_{n-1})]+\theta_n\frac{h}{2}\bar{u}'''(\eta_n), \quad |\theta_n|\leqslant 1, \\ &= \frac{1}{h}[f(t_n,u_n)-f(t_{n-1},u_{n-1})]-\frac{\mu}{h}(e_n-e_{n-1})+\theta_n\frac{h}{2}\bar{u}'''(\eta_n).\end{aligned} \tag{8.2}$$

Zij nu ϕ de gladde oplossing van (8.1) en ψ: $\psi(t)=e^{-\mu t}$ de 'inschakelfunctie'. Dan geldt wegens $h\mu \gg 1$ al na één of enkele h-stappen in hoge mate van precisie $\bar{u}^{(i)} \approx \phi^{(i)}$, $i=0,1,2,3$. We zullen nu laten zien dat $\frac{\mu}{h}(e_n-e_{n-1})$ omstreeks $\frac{h}{2}\phi'''(t_n)$ $(\approx \frac{h}{2}\bar{u}'''(t_n))$ is. Dat betekent dan dat als je $\bar{u}''(\xi_n)$ mag

schatten m.b.v. $\frac{1}{h}[\bar{u}'(t_n)-\bar{u}'(t_{n-1})]$ (d.w.z. dat $|\frac{h}{2}\bar{u}'''(\eta_n)|\ll|\bar{u}''(\xi_n)|$), het ook wel mag met $\frac{1}{h}[f(t_n,u_n)-f(t_{n-1},u_{n-1})]$, hetgeen een optimaal resultaat is.

Wegens (7.6) geldt

$$\frac{\mu}{h}(e_n-e_{n-1}) = \tag{8.3}$$

$$= \frac{\mu}{h}\left[\frac{\tau_n-\tau_{n-1}}{1+h\mu}+\frac{\tau_{n-1}-\tau_{n-2}}{(1+h\mu)^2}+\cdots+\frac{\tau_2-\tau_1}{(1+h\mu)^{n-1}}+\frac{\tau_1}{(1+h\mu)^n}\right].$$

Passen we dit toe op de ϕ-componenten van de τ_i dan levert dit ongeveer op

$$\frac{\mu}{h}\frac{h^2}{2}\left[\frac{\phi''(t_n)-\phi''(t_{n-1})}{1+h\mu}+\frac{\phi''(t_{n-1})-\phi''(t_{n-2})}{(1+h\mu)^2}+\cdots\right] \tag{8.4}$$

$$= \frac{\mu}{h}\frac{h^3}{2}\left[\frac{\phi'''(v_n)}{1+h\mu}+\frac{\phi'''(v_{n-1})}{(1+h\mu)^2}+\cdots\right] \approx \frac{h}{2}\phi'''(v_n).$$

Voor de ψ-componenten van de τ_i krijgen we, na omschikking van de termen

$$\frac{\mu}{h}\left[\frac{\tilde{\tau}_n}{1+h\mu}-\frac{\tilde{\tau}_{n-1}\mu h}{(1+h\mu)^2}-\cdots-\frac{\tilde{\tau}_1\mu h}{(1+h\mu)^n}\right] \tag{8.5}$$

met $\tilde{\tau}_n=(1+h\mu)\psi_n-\psi_{n-1}$, zodat in (8.5) de laatste term overweegt, hetgeen resulteert in $\frac{\mu^2}{(h\mu)^n}$, en dit is reeds voor lage waarden van n verdwijnend klein. Hiermee zijn impliciet voorwaarden aangegeven waaronder het beweerde geldt.

Voor hogere orde BDF formules kan men iets soortgelijks aantonen, en dat kan ook nog voor stelsels mits deze niet al te snel veranderlijk zijn. We gaan hier niet verder op in, evenmin als op de vraag of men niet beter $\frac{1}{h^2}(u_n-2u_{n-1}+u_{n-2})$ kan gebruiken dan $\frac{1}{h}[f(t_n,u_n)-f(t_{n-1},u_{n-1})]$. De literatuur laat ons op dit gebied nogal in de steek. In 1975 schreven Byrne en Hindmarsh (zie [4]):

> 'Many of the assumptions used to derive these error estimates are in practice unjustified and unjustifiable. The past values do not usually satisfy (2.36) (dat is onze relatie (5.11)), the neglected remainder terms are not generally negligible and the smooth differentiability of u is often not present. However, the resulting formulas easily allow for the selection algorithm for step size and order. That algorithm, to be described shortly, works well in practice and has a basis that is at least heuristically valid and is in part rigorously valid'.

De situatie is inmiddels nog niet erg verbeterd. Er schijnt een tweede deel van [17] op stapel te staan dat geheel over stijve differentiaalvergelijkingen zou

gaan. Misschien dat daar alles in staat. Artikelen als [16], [1] en [20] met asymptotische ontwikkelingen voor de benaderingsfout naar h en ϵ wekken in elk geval verwachtingen.

LITERATUUR

1. W. AUZINGER, R. FRANK, F. MACSEK (1986). *Asymptotic Error Expansions for Stiff Equations, Part I: The Strongly Stiff Case,* Institut für angewandte und numerische Mathematik, Technische Universität Wien, Report Nr. 67/86.
2. U.M. ASCHER, R.M.M. MATTHEY, R.D. RUSSELL (1988). *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations,* Prentice Hall series in computational mathematics, Englewood Cliffs, Prentice Hall.
3. F. BASHFORTH (1883). *An Attempt to test the Theories of Capillary Action by Comparing the Theoretical and Measured Forms of Drops of Fluid. With an Explanation of the Method of Integration Employed in Constructing the Tables which give the Theoretical Forms of such Drops,* by J.C. ADAMS. Cambridge, at the university press.
4. G.D. BYRNE, A.C. HINDMARSH (1975). A polyalgorithm for the numerical solution of ordinary differential equations. *ACM Trans. on Math. Software 1,* 71-96.
5. G. DAHLQUIST (1956). Convergence and stability in the numerical integration of ordinary differential equations. *Math. Scand. 4,* 33-53.
6. G. DAHLQUIST (1959). Stability and Error Bounds in the Numerical Integration of Ordinary Differential Equations, Trans. of the Royal Inst. of Techn. (Kungl. Tekniska Högskolans Handlinger), Stockholm nr. 130.
7. G. DAHLQUIST (1963). A special stability problem for linear multistep methods. *BIT 3,* 27-43.
8. C.W. GEAR (1971). *Numerical Initial Value Problems in Ordinary Differential Equations,* Englewood Cliffs, Prentice Hall.
9. C.W. GEAR (1971). The automatic integration of ordinary differential equations. *Comm. ACM 14,* 176-179.
10. C.W. GEAR (1971). Algorithm 407: DIFSUB for solution of ordinary differential equations. *Comm. ACM 14,* 185-190.
11. C.W. GEAR, K.W. TU (1974). The effect of variable mesh size on the stability of multistep methods. *SIAM J. Numer. Anal. 11,* 1025-1043.
12. C.W. GEAR, D.S. WATANABE (1974). Stability and convergence of variable order multistep methods. *SIAM J. Numer Anal. 11,* 1044-1058.
13. R.D. GRIGORIEFF (1977). *Numerik gewöhnlicher Differentialgleichungen. Band 2: Mehrschrittverfahren. Teubner Studienbücher Mathematik.* Stuttgart, B.G. Teubner.
14. P. HENRICI (1962). *Discrete Variable Methods in Ordinary Differential Equations,* New York etc., John Wiley.
15. P. HENRICI (1963). *Error Propagation for Difference Methods,* SIAM series in applied mathematics, New York etc., John Wiley.

16. E. HAIRER, C. LUBICH (1988). Extrapolation at stiff differential equations. *Numer. Math. 52,* 377-400.
17. E. HAIRER, S.P. NØRSETT, G. WANNER (1987). *Solving Ordinary Differential Equations,* Vol. I: *Nonstiff Problems,* Springer series in computational mathematics vol. 8. Berlin etc, Springer-Verlag.
18. W.E. MILNE (1953). *Numerical Solution of Differential Equations,* New York etc., John Wiley.
19. H.J. STETTER (1973). *Analysis of Discretization Methods for Ordinary Differential Equations,* Berlin etc., Springer-Verlag.
20. M. VAN VELDHUIZEN (1984). Asymptotic expansions of the global error for the implicit midpoint rule (stiff case). *Computing 33,* 185-192.

Computeralgebra en Numerieke Wiskunde: Samen op Weg?

J.A. van Hulzen
Universiteit Twente, Faculteit Informatica
Postbus 217, 7500 AE Enschede

In kort bestek wordt geprobeerd enig inzicht te geven in de mogelijkheden van het gebruik van een computeralgebra-systeem voor de constructie van programma's voor het numeriek oplossen van een probleem. Hierbij is zoveel als mogelijk vermeden technische details te geven. Programmageneratie en -optimalisatie krijgen vooral aandacht.

1. Inleiding

De betaalbaarheid en de mogelijkheden van personal computers en werkstations zijn van grote betekenis voor het werk van een toegepast wiskundige. Geavanceerde mathematische programmatuur komt hierdoor binnen handbereik. Hierbij kan met name gedacht worden aan numerieke, grafische of computeralgebraïsche faciliteiten, die zowel afzonderlijk als geïntegreerd bruikbaar zijn. Maar kan dat al?

Wij gaan nader in op enige van de vele aspecten van een gecombineerd gebruik van programmatuur voor numerieke berekeningen en voor computeralgebra. Wij bespreken welke rol computeralgebra kan spelen bij het genereren van programma's voor het numeriek oplossen van een probleem. Aannemende dat we de beschikking hebben over b.v. de NAG-bibliotheek [37], kan de vraag gesteld worden hoe een computeralgebra-systeem te gebruiken voor de constructie van de probleemspecificatie en oplossingsstrategie, volgens de door de bibliotheek gestelde richtlijnen. Men denke hierbij b.v. aan de generatie van Jacobianen of Hessianen. Een aanvullende vraag is of executie van de aldus geconstrueerde programma's tot geloofwaardige oplossingen leidt, of anders gezegd, is foutanalyse automatiseerbaar? In samenhang hiermee rijst bovendien de vraag aan welke computerarchitecturen hierbij gedacht moet of mag worden.

Alvorens hier nader op in te gaan lijken een paar inleidende opmerkingen over computeralgebra-systemen op zijn plaats. De door Kahrimanian [30] en Nolan [39] in 1953 gepubliceerde differentiatie-programma's worden wel als de eerste pogingen erkent om een computer te gebruiken voor het uitvoeren van formeel wiskundige handelingen. Tegenwoordig kennen we een rijke diversiteit van computeralgebra-systemen [27]. Sommigen hiervan worden veelvuldig en routinematig gebruikt bij het oplossen van niet-triviale problemen [1,7,55].

Heel algemeen gesteld, zou men in navolging van Loos [35] onder computeralgebra dat deel van de informatica kunnen verstaan, dat zich bezig houdt met ontwerp, analyse, implementatie en gebruik van algebraïsche algoritmen. Goede inleidingen in de computeralgebra zijn van de hand van Pavelle, Rothstein en Fitch [40] en van Yun en Stoutemyer [55]; recente overzichten zijn [5,8,12].

De mogelijkheden van bekende systemen voor algemeen gebruik, zoals MACSYMA [41], MAPLE [9], muMATH [54], REDUCE [13] of SCRATCHPAD [29], hangen nauw samen met de vroege successen van wat nu computeralgebra heet bij integratie, hemelmechanica, relativiteitstheorie en hoge energie fysica, b.v. Deze toepassingen waren van invloed op de klasse van mathematische uitdrukkingen, die in deze systemen behandelbaar zijn en die aan zekere transformaties, als differentiëren of integreren, onderworpen kunnen worden. We richten onze aandacht dus op klassieke-algebra-systemen

Polynoom- en rationale-functie-algebra werd en wordt hierbij als basisvoorziening beschouwd. Vaak wordt hierbij intern een recursieve representatie gehanteerd. Dit betekent dat een polynoom in meerdere variabelen opgevat wordt als univariabel polynoom, waarvan de coëfficiënten polynomen zijn in de overige variabelen. Hierdoor ontstaat intern een boomrepresentatie, waarbij variabelen, exponenten en coëfficiënten als bladeren aangemerkt kunnen worden. Dit betekent dat de coëfficiënt- en exponent-arithmetiek apart geprogrammeerd kunnen worden. Bij een interne representatie waarbij volledige expansie van de te behandelen polynomen gebruikt wordt, kan na vaststelling van een variabele- en een graadordening een kanonieke representatie verkregen worden. Hiermee wordt bedoeld dat de gebruiker van zo'n systeem de vrijheid heeft een polynoom naar eigen smaak in te typen, maar dat het systeem er intern steeds dezelfde codering voor gebruikt. Deze codering staat feitelijk los van de coëfficiënt-arithmetiek, welke via generieke faciliteiten geregeld kan worden. Deze codering laat op een vergelijkbare manier toe de variabelen van een andere invulling te voorzien. B.v. door op deze manier de elementaire transcendente funkties in het spel te betrekken. Hiermee worden tevens enige problemen ingevoerd. Hoe is bij twee niet-polynomiale expressies vast te stellen of het verschijningsvormen van één en hetzelfde mathematische object zijn? Dit z.g. vereenvoudigingsprobleem is niet triviaal; in het algemeen zelfs onbeslisbaar [6]. Expansie van polynomiale vormen is bovendien niet altijd aan te raden, omdat het in veel gevallen eerder inzichtversluierend dan inzichtbevorderend werkt. En is het laatste niet feitelijk de bedoeling? Dit betekent dat b.v. de mogelijkheid moet bestaan om een willekeurige uitdrukking als variabele in een polynoom aan te merken. Als dit toegestaan is, kan intern een willekeurig geneste uitdrukking opgeslagen en behandeld worden. We kunnen dan zeker niet meer van een kanonieke representatie spreken. Waar wel naar gestreefd wordt, is de opslag van een z.g. normale representatie. Dit betekent dat expressies, die equivalent met 0 zijn als 0 herkend kunnen worden.

Het moge intuïtief duidelijk zijn dat de kwaliteit van de interne algebra - in samenhang met de interne representatie - bepaalt hoe aantrekkelijk zo'n systeem in de praktijk is. Oudere systemen kennen vaak één interne

representatie, die t.g.v. latere ontwikkelingen steeds verder gemodificeerd is, b.v. door uitbreiding van de toegestane coëfficiënt-domeinen. Tegenwoordig worden wel vele min of meer probleemafhankelijke interne representaties toegelaten, waarbij de efficiëntie verhoogd wordt via adekwate, probleemgebied gebonden bibliotheek-voorzieningen.

We hebben geprobeerd in het kort een beeld te schetsen van het type uitdrukkingen, die men tegen kan komen bij het gebruik van een computeralgebra-systeem. De conclusie mag zijn: feitelijk alles. Het hangt van de interne algebra af hoe efficiënt of effectief een systeem is. Verder is de wijze van presenteren van de bewerkingen, die op expressies uitgevoerd zijn - de uitvoer dus - van groot belang. De verschijningsvorm van het gepresenteerde resultaat is in het algemeen slechts een van de vele mogelijkheden. Een natuurlijke en daarom logische tendens is om bij het ontwerp van de algoritmen, die de interne algebra besturen, te proberen de structuur die in de aangeboden expressies aanwezig is zo lang als mogelijk te handhaven. Hermodellering kan dan via een aantal aan de gebruiker aan te bieden faciliteiten gebeuren. Een van deze mogelijkheden zou kunnen zijn de uitvoer aan te bieden in syntactisch correct FORTRAN.

We dienen te bedenken dat de gebruikerstaal, die veelal een vriendelijk en eenvoudig karakter heeft, in het algemeen niet hetzelfde is als de voor de constructie van het systeem gebruikte programmeertaal. Daar veel computeralgebra-systemen interactief bruikbaar zijn, vindt commando-interpretatie plaats. Dit betekent dat de ingetypte opdracht omgezet wordt naar een interne codering, die leidt tot een evaluatie-opdracht op lager niveau, waarvan een onderdeel is de verzorging van de uitvoer, de systeemreactie dus. Systeem-uitbreidingen kunnen daarom i.h.a. op verschillende niveaus tot stand gebracht worden. Als we de uitvoerpresentatie b.v. willen modificeren zal dat i.h.a. het best kunnen gebeuren door op het systeem-programmeringsniveau in te grijpen.

Laten we nu terugkeren naar ons eigenlijke voornemen. We behandelen eerst de z.g. symbolic-numeric interface. Daarna werken we enige aspecten daarvan nader uit, zoals programma-generatie en code-optimalisatie.

2. De symbolic-numeric interface

Aspecten van de symbolic-numeric interface zijn vrij gedetailleerd besproken door Brown en Hearn [4] en in aanvulling daarop door Ng [38]. De klaarblijkelijke noodzaak voor zo'n tussenschakeling suggereert, dat er communicatieproblemen zijn m.b.t. de uitwisseling van informatie tussen algebra-systemen en voorzieningen, die meer speciaal ontworpen zijn voor het numeriek oplossen van problemen. Brown en Hearn onderscheiden twee typen problemen: numerieke evaluatie van symbolische, d.w.z. via een algebra-systeem verkregen, resultaten en hybride technieken, d.w.z. mengvormen van symbolische en numerieke technieken. Dit betekent dat een onderdeel daarvan ook numerieke evaluatie van symbolische resultaten kan zijn.

Voor numerieke evaluatie, dat wil in het algemeen zeggen gebruik van niet-exacte arithmetiek, kan gekozen worden voor interpretatieve evaluatie, d.w.z.

voor het gebruik van het algebra-systeem zelf, of voor de generatie van arithmetiek in een andere programmeertaal, leidend tot uitvoer in die taal. Beide alternatieven hebben voor- en nadelen.

Interpretatie kan aantrekkelijk zijn als het om 'one shot' toepassingen gaat, en aannemende dat willekeurige lengte floating point arithmetiek in de computeralgebra-context bruikbaar is. Een voorbeeld hiervan is b.v. het door Sasaki voor REDUCE gemaakte pakket [46]. Hij stelde experimenteel vast dat zijn faciliteiten, als hij portabiliteits-eisen in aanmerking neemt, ongeveer half zo snel zijn als overeenkomende FORTRAN-faciliteiten [31]. Vergelijkbare conclusies zijn door Steele [48] en Pitman [42] getrokken m.b.t. MACSYMA. Pitman creëerde zelfs de mogelijkheid FORTRAN-subroutines om te zetten in LISP-functies, waardoor het mogelijk werd de IMSL-bibliotheek in een MACSYMA-context te gebruiken. Als nadeel kan natuurlijk aangemerkt worden, dat een foutanalyse als begeleidend verschijnsel nodig blijft. Als men slechts differentiatie als formeel wiskundige handeling nodig heeft, bestaan alternatieven voor het gebruik van een computeralgebra-systeem. Programmapakketten als PASCAL-SC bieden de mogelijkheid dit type bewerking in de eigen context te doen, om vervolgens de berekeningen m.b.v. intervalarithmetiek op betrouwbare wijze uit te voeren [11,32,43,44,45]. Een nadeel van deze benadering is, dat de in een algebra-systeem aanwezige vereenvoudigings-algoritmen niet toegepast kunnen worden, waardoor de voor het berekenen van (partiële) afgeleiden gegenereerde code inefficiënt kan worden. Dit hoeft natuurlijk voor problemen met een beperkte omvang niet bezwaarlijk te zijn.

Het alternatief - generatie van arithmetische toekenningsopdrachten in een andere programmeertaal - is evenmin volmaakt. Veel computeralgebrasystemen bieden de mogelijkheid de verkregen resultaten in b.v. FORTRAN-notatie uit te voeren. Maar als een gebruiker besluit dit te doen, is zijn kennelijke intentie deze uitvoer in een volledig programma in te passen. Dat betekent dat hij op enigerlei wijze de samenhang tussen de arithmetische uitdrukkingen, die feitelijk de probleemstructurering weergeven, zelf via tekst-uitvoeropdrachten of later m.b.v. een tekstverwerkingsprogramma moet verzorgen. Dit vraagt dus om meer geavanceerde mogelijkheden. We gaan hierop in de volgende paragraaf in.

De omvang van de om te zetten uitdrukkingen kan een aanvullend probleem vormen. Toepassingen en toepassingsstrategieën, zoals b.v. beschreven door Cook [10], van den Heuvel, van Hulzen en Goldman [20], Smit en van Hulzen [47], Steinberg en Roache [49] en Wang [50] maken duidelijk dat het gebruik van computeralgebra met een zekere zorg gebeuren moet. Uit dit soort toepassingen blijkt duidelijk de zin van Hearn's waarschuwing [18,19], dat we op een effectieve manier moeten leren omgaan met de structuur, die inherent in een probleem aanwezig is. Een zinnige illustratie hiervan is b.v. Wang's behandeling van in de probleembeschrijving aanwezige symmetrie [50]. Als eerder gesuggereerd, en ook door Hearn gesteld [18,19], is een verkregen symbolisch resultaat slechts een van de vele mogelijke verschijningsvormen van een gegeven functie. Via veelal heuristische optimalisatie-technieken kan zo'n

beschrijving drastisch veranderd worden. De hier bedoelde verandering betekent een poging de door de uitdrukking weergegeven arithmetiek te minimaliseren door herhaaldelijk optredende identieke delen te vervangen door een enkel symbool, waarbij dit symbool de rol van aan de betreffende deeluitdrukking toegekende naam gaat spelen. Deze vorm van uitvoerbeïnvloeding behoeft dus een ingrijpen in de standaard systeem-strategie voor het afhandelen van een gebruikerscommando. We gaan hier nader op in in paragraaf 4.

Als we eenmaal in staat zijn efficiënte code te genereren voor het uitvoeren van numerieke berekeningen zou het bovendien erg aantrekkelijk zijn als we ook in staat zouden zijn garanties met betrekking tot de betrouwbaarheid en/of stabiliteit van de aldus vervaardigde code te verkrijgen. Gezien het bestaan van pakketten voor het uitvoeren van meervoudige lengte floating point arithmetiek, zou het aantrekkelijk kunnen zijn om, voordat de berekeningen werkelijk uitgevoerd worden, via het gebruik van aan een algebra-systeem toegevoegde programmatuur voor deze berekeningen te kunnen vaststellen met welke precisie ze uitgevoerd moeten worden om te voorkomen, dat de door de floating point arithmetiek veroorzaakte foutgeneratie leidt tot een overwoekering van de onvermijdelijke propagatie van invoer-onnauwkeurigheden. Experimenten op dit gebied zijn nog beperkt van omvang [21,36].

De bovengenoemde aspecten van het genereren van code voor numerieke verwerking hoeven niet noodzakelijk gekoppeld gedacht te worden aan de traditionele sequentiële afhandeling van een programma. Experimenten m.b.t. generatie van programmatuur voor vector- en parallel-architecturen zijn b.v. beschreven door Berman [2], Goldman en van Hulzen [17] en door Wang en Sharma [52]. Dit aspect zal ongetwijfeld de komende jaren steeds grotere aandacht krijgen.

3. Programmageneratie

Het enige gereedschap dat tot voor enige jaren via systemen als REDUCE en MACSYMA aangeboden werd was een systeem-switch, die het toeliet uitvoer in FORTRAN-notatie te produceren. Het gegeven dat FORTRAN-uitvoer veelal aan een maximaal aantal regels gebonden is kon daarom, naast de al eerder aangeduide problemen m.b.t. programma-structurering, tot frustraties leiden

MACTRAN [53] en VAXTRAN [34] waren de eerste twee pakketten, die als reactie hierop ontworpen werden om de gebruiker behulpzaam te zijn bij dit soort werkzaamheden. MACTRAN is als uitbreiding van MACSYMA te gebruiken en laat toe volledige FORTRAN-subroutines te construeren, gebruikmakend van door de gebruiker zelf klaargemaakte z.g. template-files. Zo'n file bevat de structuur van het te maken programma(deel), afgewisseld met later te activeren stukken met MACSYMA-commando's. Zo'n file wordt in een MACSYMA-sessie afgehandeld door eenvoudigweg de passieve tekstdelen, de structuur van het programma definiërend, te copiëren naar de uitvoer-file en de in de actieve delen opgenomen MACSYMA-commando's zo uit te voeren, dat de resultaten ook naar dezelfde uitvoer-file gezonden worden.

Het resultaat van de uitvoering van de actieve delen behoort natuurlijk tot betekenisvolle arithmetische toekenningsopdrachten te leiden. VAXTRAN, in Franz LISP gecodeerd en bedoeld als uitbreiding van de mogelijkheden van MACSYMA, lijkt veel op MACTRAN. Maar in aanvulling op MACTRAN bevat het een meer algemene interface tussen MACSYMA en programmatuur voor numerieke berekeningen. Alhoewel het via VAXTRAN mogelijk is gegenereerde FORTRAN-code te compileren en vervolgens direct uitvoerbaar en aanroepbaar te maken vanuit een MACSYMA-omgeving, is de generatie van de arithmetische uitdrukkingen uitsluitend gebaseerd op het gebruik van de FORTRAN-switch. Dit kan dus in de praktijk het gebruik van VAXTRAN beperken tot problemen van beperkte omvang.

GENTRAN is van meer recente datum. Het is oorspronkelijk eveneens in Franz LISP geprogrammeerd als MACSYMA-instrument. De aanvankelijke bedoeling was dit programma te gebruiken voor de generatie m.b.v. MACSYMA van RATFOR-code, die in combinatie met een bestaand FORTRAN-pakket voor eindige-element-analyse bruikbaar zou moeten zijn [16]. Een tweede versie van GENTRAN is in RLISP geschreven om in een REDUCE-omgeving gebruikt te kunnen worden [15]. De mogelijkheden werden uitgebreid. Deze versie maakt het mogelijk de interne prefixvormen van REDUCE om te zetten naar C, RATFOR of FORTRAN 77. Dit pakket maakt nu deel uit van REDUCE 3.3. Er is intussen ook een nieuwe MACSYMA-versie van gemaakt, terwijl aan verdere uitbreidingen van de REDUCE-versie gewerkt wordt, b.v. aan de mogelijkheid omzettingen naar PASCAL-SC te verkrijgen. Niet onvermeld mag hier blijven, dat GENCRAY in ontwikkeling is, een vergelijkbaar pakket, dat speciaal gericht wordt op vervaardiging van programma's in CFT77 (Cray ForTran-77) [51].

Wat kan GENTRAN meer dan b.v. MACTRAN? Het template-file mechanisme is aanwezig. Een opvallend verschil is echter dat de uitvoer-expressies m.b.v. door de gebruiker verschafte richtlijnen gesegmenteerd kunnen worden. Een ander opmerkelijk verschil is dat volledige syntactisch correcte REDUCE- of MACSYMA-commando's omgezet kunnen worden naar de beoogde doeltaal. Voorbeelden van zulke commando's zijn b.v. declaraties, FOR-loops, REPEAT..UNTIL en WHILE-constructies en functie-definities. Zonder speciale voorzieningen zal een door het speciale woord GENTRAN voorafgegaan commando letterlijk vertaald worden in een syntactisch correcte en equivalente opdracht in de doeltaal. GENTRAN biedt via enige speciale toekenningsopdrachten echter de mogelijkheid delen van zo'n te vertalen opdracht eerst te evalueren. Deze mengmogelijkheid laat toe efficiënte doeltaal-code te vervaardigen. Voorbeelden hiervan zijn te vinden bij Gates [14] en van Hulzen [25,26].

4. Code-optimalisatie

De bestaande computeralgebra-systemen laten eenvoudig de generatie toe van wiskundige uitdrukkingen, die qua omvang ons bevattingsvermogen feitelijk te boven gaan. De wens om efficiënte programma's te kunnen maken is daarom ook direct gekoppeld aan het eerder genoemde vereenvoudigingsprobleem.

Hierbij speelt ook de vraag mee, zeker op de achtergrond, of het eigenlijk wel mogelijk is om in algemene termen over vereenvoudiging te spreken. Het betekent dus dat mens-machine interactie mogelijk moet zijn bij het modelleren van wiskundige uitdrukkingen. Dit is zeker zo als de eerder genoemde strategie, die gebaseerd is op structuurhandhaving, niet bevredigend werkt. Dan toe te passen technieken zijn nog vaak heuristisch van aard [18,19]. De eerste aanzet is afkomstig van Brenner en Cook [10] en ontwikkeld in een MACSYMA omgeving. Vergelijkbare faciliteiten zijn daarna ook voor REDUCE ontwikkeld [22]. De strategie is gebaseerd op de vooronderstelling dat bij de beschrijving van een probleem de fysische betekenis van de variabelen een gewogen rol kan spelen. Hergroepering, eventueel ook in delen van een expressie, door deze als polynoom in enige interessante variabelen op te vatten, blijkt soms verrassende resultaten op te leveren. Bij deze locale hergroepering kan gebruik gemaakt worden van zowel expansie als factorisatie. Soms helpt het ook als locaal factoren opgespoord kunnen worden, die slechts enige van alle termen van een (deel)som gemeen hebben.

Bij deze pogingen de beschrijving van de functie, die door een lange expressie gegeven is, compacter te maken wordt van de expressie als één te transformeren object uitgegaan. De achterliggende gedachte is daarbij dat een kortere representatie in het algemeen meer inzicht biedt dan een langere beschrijvingswijze. Deze strategie laat daarom herhaald optredende, identieke delen in een expressie ongemoeid. Zeker voor een latere numerieke verwerking van zulke uitdrukkingen ware het echter aan te bevelen deze herhaald optredende patronen op te sporen en door een nieuwe variabele te vervangen. Hierbij natuurlijk aannemende, dat aan deze nieuwe variabele tijdig een waarde toegekend wordt, die berekend wordt volgens de in de vervangen deelexpressie gedefinieerde rekenregels. Zowel REDUCE (STRUCTR) als MACSYMA (OPTIMIZE) kent zo'n voorziening. Deze faciliteiten zijn gebaseerd op de vooronderstelling, dat de structuur van de te behandelen uitdrukking niet mag veranderen. Dit betekent b.v. dat de mogelijke geldigheid van de commutatieve eigenschap niet gebruikt wordt. In expressiedelen als $a + b + c$ en $a + c$ wordt $a + c$ daarom ook niet aangemerkt als gemeenschappelijke deelexpressie (voortaan gde te noemen). Een voordeel van zo'n strategie is, dat snel - immers makkelijk herkenbaar - opvallende herhalingen opgespoord kunnen worden. Daarnaast is het ook eenvoudig mogelijk zulke herstructureringen in b.v. FORTRAN om te zetten. Dit leidt tot een goedkope en redelijk effectieve manier van optimalisering. Iets dergelijks kan gesteld worden voor het gebruik van Horner-regels bij de produktie van uitvoer-expressies. Deze technieken hebben echter allen het nadeel dat pogingen te komen tot beknopter schrijfwijzen van uitdrukkingen zich beperken tot op iedere uitdrukking apart toe te passen slimmigheden.

Bij de wens programma-generatie via een computeralgebra-systeem mogelijk te maken speelt natuurlijk een rol, dat het veel aantrekkelijker is in ieder geval alle blokken te activeren code, die vaak zullen corresponderen met straightline code, in hun geheel efficiënter te schrijven. Nog aansprekender zou het zijn als volledige programma's aan zo'n opknapbeurt onderworpen kunnen worden.

Dit suggereert dat aan compiler-optimalisatie verwante technieken mogelijk bruikbaar zijn, bij pogingen efficiënte omzetting naar numerieke code te verkrijgen. Hierbij moet echter bedacht worden dat het type programma, waar het hier om gaat, niet als doorsnede van de door een compiler om te zetten programma's aangemerkt kan worden. Knuth's conclusies na een empirische studie over de opbouw van FORTRAN-programma's [33], dat de in- en uitvoer activiteiten vaak kostbaarder zijn dan de in een programma opgenomen arithmetiek zijn hier zeker niet van toepassing. We richten onze aandacht nu op de vraag wat op dit gebied nu al mogelijk is.

De tot nu toe ontwikkelde programmatuur [23,24,28] gaat uit van enige premissen. Als numerieke code ontwikkeld moet worden ligt het voor de hand dat er sprake is van een omvangrijk, veelal praktisch probleem. De er aan ten grondslag liggende modelvorming is gebaseerd op wiskundige regels, waarbinnen commutativiteit en associativiteit gelden en de bekende distributieve eigenschappen toepasbaar zijn. De transformatie-activiteiten zijn gericht op verzamelingen toekennings-opdrachten, waarin het linkerlid als herkenningssymbool fungeert van de in het rechterlid beschreven arithmetische operaties, die na uitvoering in een aan de linkerlid-variabele toe te kennen waarde resulteren. Deze waarde is natuurlijk afhankelijk van de aan de in het rechterlid voorkomende variabelen toe te kennen invoerwaarden. De optimalisatie van deze beschrijvingen zal bestaan uit het minimaliseren van de door de rechterleden gedefinieerde arithmetiek. De optimalisatie-strategie draagt een heuristisch karakter, zodat invulling van de begrippen optimaal en minimaal steeds afhankelijk is van de gekozen werkwijze. Minimalisatie van arithmetiek is gekoppeld aan de veronderstelling dat een programma sequentieel afgehandeld wordt. Dit betekent dat zulke technieken bijstelling behoeven zodra generatie van programma's, die op b.v. vectormachines verwerkt moeten worden, ter sprake komt.

Hoe kunnen we in algemene termen aangeven wat de opbouw van de te behandelen rechterleden kan zijn? Als we, zoals in LISP b.v. gebruikelijk is, uitgaan van een prefix-representatie, bestaat iedere uitdrukking uit een paar (operator . lijst van operanden) of uit een losse constante of variabele. De operanden worden gedacht op dezelfde manier geconstrueerd te worden. We onderscheiden nu drie mogelijkheden voor de operator: PLUS, MAAL of IETS_ANDERS. De laatste mogelijkheid omvat naast QUOTIENT b.v. ook functie-symbolen als SIN, COS, e.d. Als we de constanten gehele getallen laten zijn, zal bij uitsluiting van de IETS_ANDERS mogelijkheid, een rechterlid een multivariabel polynoom over $\mathbb{Z}$ zijn. Zo'n polynoom kan worden opgevat als een som (product) van eenvoudige of samengestelde termen (factoren). Een eenvoudige term is een constante, een variabele of het produkt van een constante en een variabele. Als dus een polynoom een som is $(3a + 2b + 3b^2c(3a + 2b)(c + d)^2)$, heeft het een (eventueel leeg) eenvoudig deel $(3a + 2b)$, gevormd door een lineaire combinatie van variabelen, eventueel uitgebreid met een constante, en uit een (eventueel leeg) samengesteld deel, bestaande uit termen, die produkten zijn, die geen constante veelvouden van variabelen zijn $(3b^2c(3a + 2b)(c + d)^2)$. Deze produkten worden

eenvoudig genoemd als ze de structuur van een monoom hebben. Bevatten ze sommen als factoren, dan worden ze samengesteld genoemd. Een polynoom als produkt $(3b^2c(3a + 2b)(c + d)^2)$ opgevat bestaat dus uit een eventueel leeg eenvoudig deel, een monoom $(3b^2c)$, en een samengesteld deel, bestaande uit produkten van sommen $((3a + 2b)(c + d)^2)$. Deze omschrijving is iets te eenvoudig. Immers een som kan ook nog tot een zekere macht verheven zijn $((c + d)^2)$. Alhoewel de werkelijkheid dus nog iets omslachtiger is volstaan we nu met dit beeld. Het suggereert dat ieder polynoom uiteengelegd kan worden in een verzameling lineaire uitdrukkingen in de variabelen en in een andere verzameling, die bestaat uit monomen.

Uitgaande van de geldigheid van de commutatieve wet kunnen we een matrix-representatie ontwerpen voor een of meerdere polynomen. We associëren hiertoe een som-matrix met de lineaire delen en een produkt-matrix met de monomen. Als de kolommen van deze matrices met de in de verzameling polynomen voorkomende variabelen geassocieerd worden en de rijen resp. met de lineaire delen en de monomen, kunnen de coëfficiënten in de lineaire uitdrukkingen en de exponenten in de monomen gebruikt worden als niet-nul elementen van deze matrices. We dienen aanvullende informatie aan de rijen te koppelen, om ervoor te zorgen dat de polynomen weer reconstrueerbaar zijn. Bovendien moeten de bij monomen horende constante coëfficiënten en de eventueel bij sommen horende exponenten ook op zinnige manier opgeslagen worden (resp. voor b.v. $3b^2c$ en $(c + d)^2$). De aldus te creëren matrices met geheeltallige elementen kunnen gebruikt worden om op te sporen waar zich gde's bevinden. Informeel gezegd is een snelle zoekstrategie nodig om zo groot mogelijke deelmatrices te vinden waarvan de rang 1 is.

Hoe nu te handelen met de operatoren, die IETS_ANDERS zijn? Zoals hierboven gesuggereerd, kan een variabele, die bij de beschrijving van een polynoom gebruikt wordt, ook een algemenere betekenis toegekend worden als plaatsbekleder van IETS_ANDERS. Dit is precies wat gebeurt als verzamelingen rechterleden in de bovengeschetste matrices opgeslagen worden. De delen, die bestuurd worden door operatoren, die tot de IETS_ANDERS categorie behoren worden vervangen door het systeem gegenereerde namen. Deze namen worden dan nieuwe variabelen in de polynoom-schema's. Er wordt natuurlijk intern een tabel bijgehouden waarin terug te vinden is welke naam bij welk deel hoort. We moeten bovendien bedenken, dat operatoren, die tot de IETS_ANDERS verzameling behoren, in het algemeen niet-commutatief zijn, terwijl hun operanden ook weer willekeurige uitdrukkingen mogen zijn. Dit betekent dat er ook een operand-administratie is, die toelaat de daarvoor in aanmerking komende operanden direct bij de gde-zoekactiviteiten te betrekken.

Tenslotte merken we op dat de beperking tot geheeltallige coëfficiënten niet wezenlijk is. De benodigde arithmetiek kan een generiek karakter krijgen als andere coëfficiënt-domeinen toegelaten worden.

Wat zullen we onder een gde verstaan? Of, iets anders gezegd, wat is de minimale omvang van zo'n deelexpressie? De achterliggende gedachte, dat het om het verminderen van arithmetische operaties gaat, moet effectueerbaar zijn.

Ten minste één optelling of vermenigvuldiging moet dus uitgespaard worden, zodat $2a + 3c$ als gde van $e_1 = 2a + 4b + 3c$ en $e_2 = 4a + 6c + 5d$ aangemerkt kan worden. Aannemende dat deze gde inderdaad gevonden wordt, zal zinnige uitvoer b.v. als volgt gepresenteerd moeten worden : $s_0 = 2a + 3c$, $e_1 = 4b + s_0$, $e_2 = 5d + 2s_0$. Er is nu een vermenigvuldiging en een optelling uitgespaard, een vrijwel minimale winst. Het is denkbaar dat zo'n som-gde, die als factor in een produkt optreedt, tot nieuw symbool reduceert, waardoor het oorspronkelijke produkt een andere structuur gaat krijgen. Dit zelfde kan natuurlijk gelden voor een als term optredend monoom. M.a.w. zoeken naar gde's moet een iteratief karakter dragen. Na het vinden van gde's moeten de matrices bijgewerkt worden. Deze hergroepering in combinatie met de opslag van de definiërende regels in de matrices voor de gde's leidt tot steeds spaarzamer bezette matrices, die na enige stappen al blijken geen relevante geheel dichtbezette deelmatrices meer te bevatten. Opslag van de gde's in de matrices is om twee redenen nodig. Allereerst dienen ze op consistente manier in het uitvoerproces opgenomen te worden. Verder moet het mogelijk zijn ze bij de mogelijke voortzetting van het optimalisatieproces te betrekken. Daarom worden minimale besparingen steeds meegenomen. De gde-structuur kan dwingen tot een stapsgewijze opsporing.

Het proces kan uiteengelegd gedacht worden in twee stappen. Allereerst een herhaald zoeken naar gde's, die tenminste een lengte 2 hebben. Tot slot een finishing touch, waarmee een verder opknappen van de verzameling toekenningsopdrachten plaatsvindt, die niet gebaseerd is op het zoeken van deelmatrices met rang 1. Voorbeelden hiervan zijn b.v. herhaald optredende identieke veelvouden van variabelen. De iteratie-stap van het eerste en belangrijkste deel van het optimalisatie-proces kan als volgt kort samengevat worden:

1. Toepassing van de commutatieve wet bij het opsporen van lineaire (deel)sommen en (deel)monomen. De strategie is gebaseerd op Breuer's heuristische grow factor algoritme [3,23,24]. Gde's worden door nieuwe namen vervangen en hun beschrijving wordt toegevoegd aan de matrices, implicerend dat samengestelde operanden tot eenvoudige gereduceerd kunnen zijn.
2. Informatie-uitwisseling tussen beide matrix-structuren, opdat tot nieuwe symbolen gereduceerde gde's, hun nieuwe rol naar behoren kunnen vervullen bij voortzetting van het optimalisatie-proces.
3. Toepassing van de distributieve wet, inhoudende dat uitdrukkingen als $ab + ac$ vervangen worden door $a(b + c)$, door op adekwate wijze de matrices te hermodelleren.

Uit experimenten is gebleken, dat deze heuristische technieken tot grote reductie in arithmetiek kunnen leiden. Bij een omvangrijke door Hearn en Gates behandelde uitdrukking bleek dat het aanvankelijke aantal van 20.534 optellingen en aftrekkingen gereduceerd werd tot 4.316, terwijl het aantal vermenigvuldigingen van 41.746 terugliep tot 4.919. Voorts bleken van de aanvankelijke 12.473 machtsverheffingen er maar 13 over te blijven en kwamen onder de 7990 andere operaties maar 60 echt verschillende voor.

5. Enige conclusies

Een pakket als GENTRAN kan in combinatie met de zojuist beschreven optimalisatie-technieken goede diensten bewijzen bij de automatisering van het oplossen van omvangrijke problemen. Onze experimenten met de generatie van Jacobianen en Hessianen suggereren, dat de komende jaren expert-systemen kunnen ontstaan als uitbreiding van bestaande computeralgebra-systemen. Deze expert-systemen zullen geschikt zijn voor de behandeling van problemen, die qua oplossingstechniek verwant zijn. Hiervoor is echter een verdere integratie nodig van de eerder besproken faciliteiten, terwijl de optimalisatie-strategie verder gegeneraliseerd moet worden, opdat volledige programma's behandeld kunnen worden.

Dit laatste vereist inzicht in de onderlinge relaties en afhankelijkheden tussen de variabelen, die gebruikt worden bij de beschrijving van probleem en oplossingsmethode. Zo'n analyse van afhankelijkheden e.d., waaraan we momenteel werken, is ook van belang bij vragen betreffende de parallelliseerbaarheid van problemen. Ook dit is, als eerder gezegd, een ontwikkeling waar de aandacht zich in toenemende mate op zal blijken te richten. Ditzelfde zal trouwens (moeten) gelden voor automatisering van foutanalyse.

Literatuur

1. A.T. Balaban (1985). Symbolic computations in chemistry. B. Buchberger (ed.). *Proceedings EUROCAL '85, Vol.1,* Springer LNCS series nr 203, Heidelberg: Springer Verlag, 68-79.
2. R.H. Berman (1984). Measuring the performance of a computational physics environment. V.E. Golden (ed.). *Proceedings 1984 MACSYMA User's Conference,* Schenectady: G.E., 244-290.
3. M.A. Breuer (1969). Generation of optimal code for expressions via factorization. *Comm. ACM 12,* 333-340.
4. W.S. Brown, A.C. Hearn (1979). Application of symbolic mathematical computations. *Comput. Phys. Comm. 17,* 207-215.
5. B. Buchberger, G.E. Collins, R. Loos (eds.) (1982). *Computer Algebra,* Computing Supplementum 4, Wien: Springer Verlag.
6. B. Buchberger, R. Loos (1982). Algebraic simplification, in [5], 11-44.
7. J. Calmet, J.A. van Hulzen (1982). Computer algebra applications, in [5], 245-258.
8. B.F. Caviness (1986). Computer algebra: past and future. *J. Symbolic Comput. 2,* 217-236.
9. B.W. Char, G.J. Fee, K.O. Geddes, G.H. Gonnet, M.B. Monogan (1986). A tutorial introduction to MAPLE. *J. Symbolic Comput. 2,* 179-200.
10. G.O. Cook jr. (1982). *Development of a Magnetohydrodynamic Code for Axisymmetric High-β Plasmas with Complex Magnetic Fields,* Ph.D. Thesis, Lawrence Livermore Nat. Lab., Cal.
11. A.A.M. Cuyt, L.B. Rall (1985). Computational implementation of the multivariate Halley method for solving nonlinear systems of equations. *ACM Trans. Math. Software 11,* 20-36.

12. J.H. DAVENPORT, Y. SIRET, E. TOURNIER (1988). *Computer Algebra*, London: Academic Press.
13. J. FITCH (1985). Solving algebraic problems with REDUCE. *J. Symbolic Comput. 1*, 211-238.
14. B.L. GATES (1985). GENTRAN: An automatic code generation facility for REDUCE. *ACM SIGSAM Bulletin 75*, 24-85.
15. B.L. GATES (1986). A numerical code generation facility for REDUCE. B.W. CHAR (ed.). *Proceedings SYMSAC '86*, New York: ACM, 94-99.
16. B.L. GATES, P.S. WANG (1984). A LISP-based RATFOR code generator. V.E. GOLDEN (ed.). *Proceedings 1984 MACSYMA User's Conference*, Schenectady: G.E., 319-329.
17. V.V. GOLDMAN, J.A. VAN HULZEN (1988). Automatic code vectorization of arithmetic expressions by bottom-up structure recognition. E. TOURNIER (ed.). *Proceedings CAP '88*, te verschijnen. Ook beschikbaar als Memoramdum INF-88-48, Universiteit Twente.
18. A.C. HEARN (1985). Structure: The key to improved algebraic computation. N. INADA, T. SOMA (eds.). *Proceedings RSYMSAC2*, Singapore: World Scientific Publ., 215-230.
19. A.C. HEARN (1986). Optimal evaluation of algebraic expressions. J. CALMET (ed.). *Proceedings AAECC-3*, Springer LNCS series nr 229, Heidelberg: Springer Verlag, 392-403.
20. P. VAN DEN HEUVEL, V.V. GOLDMAN, J.A. VAN HULZEN (1988). Automatic generation of FORTRAN-coded Jacobians and Hessians. J.H. DAVENPORT (ed.). *Proceedings EUROCAL '87*, te verschijnen. Ook beschikbaar als Memorandum INF-88-47, Universiteit Twente.
21. B.J.A. HULSHOF, J.A. VAN HULZEN (1984). Automatic error cumulation control. J. FITCH (ed.). *Proceedings EUROSAM '84*, Springer LNCS series 174, Heidelberg: Springer Verlag, 260-271.
22. B.J.A. HULSHOF, J.A. VAN HULZEN (1985). An expression compression package for REDUCE based on factorization and controlled expansion. B.F. CAVINESS (ed.). *Proceedings EUROCAL '85*, Vol. 2, Springer LNCS series nr 204, Heidelberg: Springer Verlag, 315-316.
23. J.A. VAN HULZEN (1981). Breuer's grow factor algorithm in computer algebra. P.S. WANG (ed.). *Proceedings SYMSAC '81*, New York: ACM, 100-104.
24. J.A. VAN HULZEN (1983). Code optimization of multivariate polynomial schemes: a pragmatic approach. J.A. VAN HULZEN (ed.). *Proceedings EUROCAL '83*, Springer LNCS series 162, Heidelberg: Springer Verlag, 286-300.
25. J.A. VAN HULZEN (1987). Program generation through symbolic processing. P.R.J. ASVELD, A. NIJHOLT (eds.). *Essays on Concepts, Formalisms and Tools*, CWI Tract 42, Amsterdam: CWI, 231-260.
26. J.A. VAN HULZEN (1988). *Formule Manipulatie m.b.v. REDUCE*, Collegedictaat Faculteit Informatica, Universiteit Twente.
27. J.A. VAN HULZEN, J. CALMET (1982). Computer algebra systems, in [5], 221-243.

28. J.A. VAN HULZEN, B.J.A. HULSHOF, B.L. GATES, M.C. VAN HEERWAARDEN. A code optimization package for REDUCE, ingezonden.
29. R.D. JENKS (1984). A primer: 11 keys to new SCRATCHPAD. J. FITCH (ed.). *Proceedings EUROSAM '84*, Springer LNCS series 174, Heidelberg: Springer verlag, 123-147.
30. H.G. KAHRIMANIAN (1953). *Analytic Differentiation by Computer*, MA Thesis, Temple University, Philadelphia, Pa.
31. Y. KANADA, T. SASAKI (1981). LISP based 'big float' system is not slow. *ACM SIGSAM Bulletin 58*, 13-19.
32. G. KEDEM (1980). Automatic differentiation of computer programs. *ACM Trans. Math. Software 6*, 150-165.
33. D.E. KNUTH (1970). An empirical study of FORTRAN programs. *Software Pract. and Experience 1*, 105-133.
34. D.H. LANAM (1981). An algebraic front-end for the production and use of numeric programs. P.S. WANG (ed.). *Proceedings SYMSAC '81*, New York: ACM, 223-227.
35. R. LOOS (1982). Introduction in [5], 1-10.
36. J.H.J. MOLENKAMP, J. MELLINK (1988). *Een onderzoek naar de bruikbaarheid van symbolisch-mumerieke oplossingstechnieken in de lineaire programmering*, Doctoraal verslag, Faculteit Informatica, Universiteit Twente.
37. *NAG FORTRAN Library Manual Mark 11* (1984). Numerical Algorithm Group, Oxford.
38. E.W. NG (1979). Symbolic-numeric interface: A review. E.W. NG (ed.). *Proceedings EUROSAM '79*, Springer LNCS series 72, Heidelberg: Springer Verlag, 330-345.
39. J. NOLAN (1953). *Differentiation on a Digital Computer*, MA Thesis, Math. Dept. MIT, Cambridge, Mass.
40. R. PAVELLE, M. ROTHSTEIN, J. FITCH (1981). Computer algebra. *Scientific American*, 102-113.
41. R. PAVELLE, P.S. WANG (1985). MACSYMA from F to G. *J. Symbol. Comput. 1*, 69-100.
42. K.M. PITMAN (1979). A FORTRAN → LISP translator. V.E. LEWIS (ed.). *Proceedings 1979 MACSYMA User's Conference*, Cambridge, Mass.: MIT, 200-214.
43. L.B. RALL (1981). *Automatic Differentiation: Techniques and Applications*, Springer LNCS series 120, Heidelberg: Springer Verlag.
44. L.B. RALL (1984). Differentiation in PASCAL-SC: type GRADIENT. *ACM Trans. Math. Software 10*, 161-184.
45. S.M. RUMP (1987). Algebraic computation, numerical computation and verified inclusion. R. JANSSEN (ed.). *Trends in Computer Algebra*, Springer LNCS series 296, Heidelberg: Springer Verlag, 177-197.
46. T. SASAKI (1979). An arbitrary precision real arithmetic package in REDUCE. E.W. NG (ed.). *Proceedings EUROSAM '79*, Springer LNCS series 72, Heidelberg: Springer Verlag, 358-368.
47. J. SMIT, J.A. VAN HULZEN (1982). Symbolic-numeric methods in microwave technology. J. CALMET (ed.). *Proceedings EUROCAM '82*, Springer

LNCS series 162, Heidelberg: Springer Verlag, 281-288.
48. G.L. STEELE JR. (1977). Fast arithmetic in MACLISP. *Proceedings 1977 MACSYMA User's Conference*, NASA-CP2012, 215-224.
49. S. STEINBERG, P. ROACHE (1984). Using VAXIMA to write FORTRAN code. V.E. GOLDEN (ed.). *Proceedings 1984 MACSYMA User's Conference*, Schenectady: G.E., 1-22.
50. P.S. WANG (1986). FINGER: A symbolic system for automatic generation of numerical programs in finite element analysis. *J. Symbol. Comput. 2*, 305-316.
51. P.S. WANG. Pers. comm.
52. P.S. WANG, N. SHARMA. Symbolic derivation and automatic generation of parallel routines for finite element analysis. P. GIANNI (ed.). *Proceedings ISSAC '88*, te verschijnen.
53. M.C. WIRTH (1980). *On the Automation of Computational Physics*, Ph.D. Thesis, Lawrence Livermore Nat. Lab. Cal.
54. C. WOOFF, D. HODGKINSON (1987). *muMATH: A microcomputer algebra system*, London: Academic Press.
55. D.Y.Y. YUN, D.R. STOUTEMYER (1980). Symbolic mathematical computation. *Encyclopedia of Computer Science and Technology Vol. 15*, New York - Basel: Marcel Dekker, 235-310.

Het Gebruik van de Computer in de Analyse van Niet-lineaire Systemen met een Vreemde Aantrekker

J. Grasman
Mathematisch Instituut
Rijksuniversiteit Utrecht

1. Inleiding

Niet-lineaire systemen met een vreemde aantrekker als oplossing laten zich vanwege de gevoelige afhankelijkheid van beginwaarden niet analyseren met de gebruikelijke numerieke oplossingsmethoden. In de praktijk wordt de oplossing geïntegreerd over een zo groot tijdsdomein dat de benadering in geen enkele decimaal correct is. Wel beweegt de numerieke oplossing zich in de toestandsruimte in de directe omgeving van de limietverzameling, de vreemde aantrekker, en volgt deze over kortere tijdsintervallen wel met een zekere nauwkeurigheid. Uit het gelineariseerde systeem, dat meeloopt met de oplossing, kan men informatie over de eigenschappen van de vreemde aantrekker verkrijgen. Uit de Lyapunov-exponenten, vergelijkbaar met de Floquet-exponenten voor periodieke oplossingen, kan het volgende afgeleid worden. De Lyapunov-exponenten worden genummerd zodanig dat

$$\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_k \geqslant 0 \geqslant \lambda_{k+1} \geqslant \cdots \geqslant \lambda_n.$$

De grootste Lyapunov-exponent λ_1 geeft aan met welke snelheid het aantal significante decimalen uit de oplossing van het beginwaardeprobleem verdwijnt. De som van alle positieve Lyapunov-exponenten, de Kolmogorov-entropie, is een maat voor de snelheid waarmee informatie, welke af te leiden is uit de begintoestand (b.v. de kans om het systeem in een deel van de toestandsruimte aan te treffen), verdwijnt. Ook de (fractale) waarde van de dimensie van de vreemde aantrekker kan in de Lyapunov-exponenten uitgedrukt worden. In par. 3 worden deze resultaten besproken. In par. 4 behandelen we de reconstructie van een vreemde aantrekker op basis van een signaal $\{y(t_1), y(t_2), \ldots, y(t_N)\}$. Dit is mogelijk door het introduceren van een systeem van voldoend hoge dimensie d met als toestandsvariabelen

$$x(t_k) = (x_1(t_k),...,x_d(t_k)) = (y(t_k),y(t_{k-1}),...,y(t_{k-d+1})).$$

In zowel par. 3 als par. 4 komt aan de orde welke de invloed van ruis is op een vreemde aantrekker. In par.2 beginnen we ons overzicht van het gebruik van de computer voor de analyse van niet-lineaire systemen met de constructie van het bifurcatiediagram voor een systeem met één of meerdere parameters. Eén van de scenario's voor het ontstaan van een vreemde aantrekker, de periodeverdubbeling, is in zo'n diagram goed te analyseren.

2. Niet-lineaire dynamische systemen en chaos

De dynamica van een fysisch proces laat zich in vele gevallen beschrijven door een differentiaalvergelijking van het type

$$\frac{dx}{dt} = f(x), \tag{2.1}$$

waarin x een n-dimensionale vector is die van de tijd t afhangt. Fysische en biologische problemen dragen veelal een niet-lineair mechanisme in zich, waardoor het interessant is de studie van (2.1) toe te spitsen op het geval dat $f(x)$ essentieel niet-lineair is.

2.1. Het beginwaardeprobleem

Als het systeem een gegeven begintoestand

$$x(0) = x_0 \tag{2.2}$$

heeft, dan bestaat er een oplossing $x(t)$ die aan (2.1)-(2.2) voldoet voor alle t. Verondersteld hierbij is, dat f een geschikte vorm heeft die inderdaad zo'n oplossing toestaat. Op grond hiervan kunnen we dus de toekomstige waarden van $x(t)$ bepalen en daarmee een voorspelling doen van de toekomstige toestand van een fysisch systeem, waarvan (2.1) een model is. Als voorbeeld kan hier genoemd worden de atmosferische circulatie voor een komende periode met de daarmee gepaard gaande weerssituatie. Dit brengt ons direct tot een aantal praktische problemen waarin de computer uitkomst biedt. Het volledige fysische probleem, dat zich laat beschrijven met de Navier-Stokes vergelijkingen, is niet exact oplosbaar. Echter een goede benadering van de vorm (2.1) is te maken. Gekozen kan worden voor een ruimtelijke discretisatie (eindige differenties of eindige elementen) of voor spectrale functies. In beide gevallen wordt slechts een zinvolle kwantitative benadering gevonden als n van de grootte-orde 1000 of hoger is. Ook van dit benaderde systeem is geen exacte oplossing te leveren en zal met numerieke methoden een tijdsintegratie uitgevoerd moeten worden. De keuze van een differentie-schema is bepalend voor de rekensnelheid en de stabiliteit van de numerieke oplossing. De geheugencapaciteit van de computer is bepalend voor de grootte van n. We kiezen n uiteraard zo groot mogelijk. Voor ons meteorologisch probleem dienen we echter ook de begintoestand met overeenkomstige precisie te bepalen. Dit blijkt een beperkende factor te zijn: meten is een kostbare zaak en vooral meetpunten op en boven zee, waar het weer van de komende dagen tot

ontwikkeling komt, zijn dun gezaaid. Nog niet zo lang geleden bestond de gedachte dat als n steeds groter gekozen kan worden en ook het aantal meetpunten toeneemt, het weer steeds langer vooruit voorspeld kan worden. Recente resultaten in de studie van de dynamica van niet-lineaire processen hebben ons geleerd dat dit niet het geval is. Zo is aangetoond (Lorenz [11]) dat het weer een eindige voorspelbaarheidshorizon heeft, welke in de grootte-orde van veertien dagen ligt, zie ook Lighthill [9] voor een discussie van dit verschijnsel.

2.2. Limietoplossingen

We gaan nu in op oplossingen van (2.1) welke gevonden worden voor $t \to \infty$. Dit zijn limietoplossingen, welke in geval van dissipatieve systemen, slechts een beperkt deel van de n-dimensionale toestandsruimte vullen. Een fysisch systeem is dissipatief als, b.v. door wrijving, energie verdwijnt. In het systeem (2.1) uit zich dit als volgt. Nemen we op tijdstip $t = 0$ een verzameling punten die een n-dimensionale bol vullen, dan zal na verloop van tijd het volume van de verschoven verzameling blijvend afgenomen zijn. Bekende limietgedragingen van een systeem van het type (2.1) zijn evenwichten en periodieke oplossingen. In de toestandsruimte hebben zij resp. dimensie 0 en 1. Periodieke oplossingen laten zich generaliseren: beschrijft het systeem 2 onafhankelijke periodieke bewegingen met perioden in een irrationele verhouding, dan vult zo'n oplossing een torus en heeft dimensie 2. Voor k onafh. periodieke bewegingen wordt dit een k-torus. Waar het systeem ook start in de n-dimensionale toestandsruimte, het nadert voor $t \to \infty$ altijd een lager dimensionale limietoplossing.

2.3. Vreemde aantrekker

Naast de bovengenoemde limietoplossingen blijkt nog een type limietoplossing te bestaan met zeer bijzondere eigenschappen: de vreemde aantrekker. Karakteristiek voor de vreemde aantrekker is dat de dimensie van de oplossingscurve geen geheel getal is, zoals in bovenstaande gevallen. Verder vertoont de oplossing een sterke afhankelijkheid van de beginwaarden, d.w.z. twee oplossingen die dicht bij elkaar starten, blijken al snel de vreemde aantrekker langs volledig verschillende trajecten te volgen.

2.4. Hausdorff-dimensie

De dimensie van een vreemde aantrekker laat zich bepalen met behulp van een gegeneraliseerde definitie van dimensie; de Hausdorff-dimensie. Als voor een verzameling punten in $\mathbb{R}^n$ $N(\epsilon)$ cellen in de vorm van een kubus met ribbe ϵ nodig zijn om de verzameling te bedekken en het aantal neemt toe volgens $N(\epsilon) \sim \epsilon^{-D}$ voor $\epsilon \to 0$, dan is D de Hausdorff-dimensie van de verzameling

$$D = \lim_{\epsilon \to 0} \frac{\log N(\epsilon)}{\log 1/\epsilon}. \tag{2.3}$$

2.5. De Lorenz-vergelijkingen

Het was de meteoroloog Lorenz [10], die als eerste een stelsel differentiaalvergelijkingen formuleerde dat in de limiet voor $t \to \infty$ een vreemde aantrekker nadert. De vergelijkingen zijn afgeleid uit een spectraalmodel voor de atmosferische circulatie en zijn van de vorm

$$\frac{dx_1}{dt} = -\sigma x_1 + \sigma x_2, \tag{2.4a}$$

$$\frac{dx_2}{dt} = -x_1 x_3 + r x_1 - x_2, \tag{2.4b}$$

$$\frac{dx_3}{dt} = x_1 x_2 - b x_3. \tag{2.4c}$$

Voor $r = 28$, $\sigma = 10$ en $b = 8/3$ wordt voor $t \to \infty$ de dynamica van het systeem bepaald door een vreemde aantrekker. We kunnen de Hausdorff-dimensie op een praktische manier bepalen door de doorsnijdingspunten van de oplossingskromme met een transversaal vlak te nemen, b.v. met $x_3 = 30$. Het aantal vierkanten met zijden ϵ nodig om de verzameling in het x_1x_2-vlak te bedekken wordt uitgezet tegen ϵ op een dubbel-logaritmische schaal. De helling van de lijn die raakt voor $\epsilon \to 0$ geeft de dimensie van de verzameling. De dimensie van de vreemde aantrekker in $\mathbb{R}^3$ wordt vervolgens gevonden door bij de richtingscoëfficiënt van de lijn 1 op te tellen: $D = 2.06$ voor de bovenvermelde keuze van de parameters.

2.6. Bifurcatiediagram

Een fysisch systeem kan afhankelijk van externe parameters verschillend limietgedrag vertonen. We beschouwen daarom de dynamica van (2.1) in afhankelijkheid van een parameter μ:

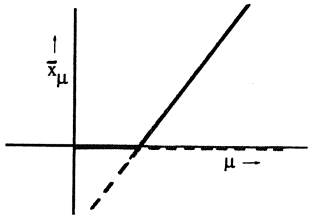
$$\frac{dx}{dt} = f_\mu(x). \tag{2.5}$$

De parameter μ is zoiets als de intensiteit waarmee het systeem geforceerd wordt: in de Lorenz-vergelijkingen vervult de parameter r deze rol, het is een maat voor de opwarming van het aardoppervlak. In eerste instantie zetten we de mogelijke evenwichtsoplossingen $x = \bar{x}_\mu$ van (2.5) uit tegen μ. In zo'n bifurcatiediagram wordt ook de stabiliteit van de evenwichtsoplossingen bijgehouden: een evenwichtsoplossing is stabiel als alle eigenwaarden van de matrix

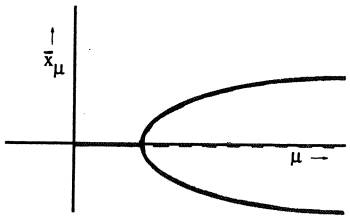
$$\{\frac{\partial f_\mu}{\partial x}(\bar{x}_\mu)\}_{n \times n} \tag{2.6}$$

negatieve reële delen hebben. Het referentiepunt van een bifurcatiediagram is de parameterwaarde, zeg $\mu = 0$, waarvoor het systeem in rust is. Als voor toenemende μ een reële eigenwaarde positief wordt, dan doorsnijdt een andere evenwichtsoplossing de betreffende tak en wordt stabiliteit uitgewisseld (a) of er ontstaan een tweetal stabiele evenwichtsoplossingen (b), zie fig.1. In beide

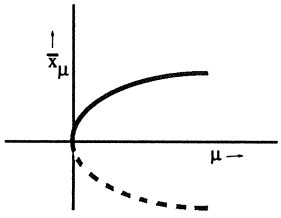
gevallen ondergaat de rusttoestand een kwalitatieve verandering. Ook is het mogelijk dat er geen enkel evenwicht bestaat en dat voor de referentieparameterwaarde twee evenwichten ontstaan: de één stabiel en de ander onstabiel (c). Als tenslotte voor een zekere μ-waarde twee toegevoegd complexe eigenwaarden de imaginaire-as kruisen (d), dan splitst zich een periodieke oplossing af (Hopf-bifurcatie).



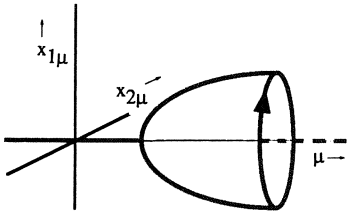
(a) Transversale bifurcatie



(b) Stemvorkbifurcatie



(c) Keerpunt



(d) Hopf-bifurcatie

FIGUUR 1. De elementaire bifurcaties; getrokken lijnen geven aan dat de limietoplossing stabiel is, gestippelde lijnen duiden op instabiliteit van de stationaire oplossing.

Er bestaan diverse software pakketten voor het construeren van een bifurcatiediagram behorende bij een vergelijking van het type (2.5). Een pakket, dat zeer goed aansluit bij de behoeften en wensen van een toegepast wiskundige, is dat van Doedel [3]. Beginnend bij de rustoplossing voor $\mu = 0$ loopt de routine alle takken af. Ook een periodieke oplossing ontstaan uit een Hopf-bifurcatie kan gevolgd worden met berekening van de bijbehorende Floquet-exponenten tot het moment dat ook deze oplossing bifurceert. Spelen er in een model twee parameters μ_1 en μ_2 een rol in het bifurcatiepatroon, dan is het met dit pakket ook mogelijk in het μ_1,μ_2-vlak de lijnen te volgen waar bij kruising een bifurcatie van het type (a) t/m (d) plaatsvindt.

2.7. Periodeverdubbeling

Een periodieke oplossing kan gevolgd worden tot het moment dat voor een zekere parameterwaarde eigenwaarden de imaginaire as kruisen. Als dit twee toegevoegd complexe eigenwaarden zijn, dan takt in het algemeen een oplossing af die een torus rond de periodieke oplossing vult. Deze oplossing en de verdere bifurcatie ervan is met de bestaande software niet te volgen. Dit bifurcatieprobleem is nog in het stadium dat de theoretische analyse ervan pas onlangs systematisch onderzocht is (Braaksma et al. [2]). Als echter reële eigenwaarden door nul gaan, dan is een verdere analyse wel mogelijk. Het is dan aantrekkelijk een Poincaré-afbeelding P te construeren die aan een punt in een transversaalvlak V het punt toevoegt dat gevonden wordt door het eerste punt als beginwaarde voor de differentiaalvergelijking te nemen en te stoppen als het transversaalvlak de eerst volgende keer in dezelfde richting doorsneden wordt. Dit levert het beeldpunt. Een periodieke oplossing correspondeert met een vast punt van deze afbeelding: $\bar{x}_\mu = P_\mu(\bar{x}_\mu)$. De oplossing is stabiel als de $n-1$ eigenwaarden van de matrix

$$\{\frac{\partial P_\mu}{\partial x}(\bar{x}_\mu)\}_{(n-1)\times(n-1)} \tag{2.7}$$

zich binnen de eenheidscirkel in het complexe vlak bevinden. Treden twee toegevoegd complexe eigenwaarden buiten de eenheidscirkel, dan takt de eerder genoemde oplossing op een torus zich af. In het transversaalvlak is dit een gesloten kromme, welke zich uit het vaste punt ontwikkelt. Gaat een reële eigenwaarde door 1, dan snijden twee takken van vaste punten elkaar en vindt eventueel uitwisseling van stabiliteit plaats. Gaat een reële eigenwaarde door -1, dan takt er een periodieke oplossing met de dubbele periode zich af. Volgen we deze periodieke oplossing dan kan bij een zekere parameterwaarde de periode ervan zich weer verdubbelen in een aftakkende oplossing. We zijn dan in het scenario van chaos door middel van periodeverdubbeling aangeland zoals dat door Feigenbaum [6] beschreven is. Een overzicht van de diverse routes die tot chaos leiden, wordt gegeven door Eckmann [4].

3. Lyapunov-exponenten en dimensies van een vreemde aantrekker

Het gedrag van oplossingen in de omgeving van een vreemde aantrekker $s(t)$ laat zich analyseren aan het gelineariseerde systeem, dat verkregen wordt na substitutie van

$$x(t) = s(t)+v(t) \tag{3.1}$$

in (2.1):

$$\frac{dv}{dt} = \frac{\partial f}{\partial x}(s(t))v. \tag{3.2}$$

3.1. Lyapunov-exponenten

Starten we met een bol met straal 1 in de oorsprong, dan zal onder (3.2) deze bol vervormd worden tot een ellipsoïde. Deze ellipsoïde loopt mee met de vreemde aantrekker. Essentieel voor een vreemde aantrekker is dat in minstens één richting de bol opgerekt wordt ondanks het feit dat het volume wel afneemt i.v.m. het dissipatieve karakter van het systeem. De hoofdassen van de ellipsoïde worden genummerd in de volgorde van de mate van strekking: op tijdstip t hebben zij een lengte $p_1(t)>p_2(t)>...>p_n(t)$ ($p_i(0) = 1$). Merk op dat hun richtingen, die op tijdstip $t = 0$ een orthogonaal stelsel vormen, verdraaien. De Lyapunov-exponenten worden gedefinieerd als

$$\lambda_i = \lim_{t\to\infty} \frac{1}{t} \log p_i(t). \tag{3.3}$$

Daar de hoofdassen op den duur alle in de richting van grootste toename zullen wijzen, dient regelmatig een orthogonalisatieprocedure plaats te vinden. Ook dient de lengte van de hoofdassen geschaald te worden, i.v.m. informatieverlies ten gevolge van het rekenen met te grote en te kleine getallen (Gram-Schmidt reorthonormalisatie). In Wolf et al. [15] wordt een numerieke procedure gegeven (incl. Fortranprogramma) om de Lyapunov-exponenten van (2.1) te berekenen. Het blijkt dat over een lang tijdsinterval geïntegreerd moet worden omdat de convergentie van (3.3) erg langzaam gaat. Uiteindelijk blijft de waarde binnen een interval fluctueren. Voor eenvoudige systemen laat λ_i zich tot op drie decimalen bepalen. De eerste Lyapunov-exponent geeft informatie over de foutengroei. Stel dat een initiële waarde gegeven is tot op ϵ nauwkeurig, dan groeit deze fout als $\epsilon e^{\lambda_1 t}$ en is van grootte-orde 1 op het tijdstip $t = 1/\lambda_1 \log 1/\epsilon$.

3.2. Dimensies

Uit de Lyapunov-exponenten kan meer informatie over de vreemde aantrekker verkregen worden. Daartoe generaliseren we eerst de dimensie-definitie van Hausdorff. We delen de toestandsruimte $\mathbb{R}^n$ op in $M(\epsilon)$ cellen met volume ϵ^n en genereren een tijdreeks uit de vreemde aantrekker $s(t_1),s(t_2),s(t_3),...,s(t_N)$ met $t_j = j\tau$ en τ voldoende klein. We definiëren de waarschijnlijkheid p_i om de aantrekker in cel i te vinden als

$$p_i = \lim_{N\to\infty} \frac{k(i)}{N}, \quad \sum_{i=1}^{M} p_i = 1, \tag{3.4}$$

waarin k het aantal keren is dat de aantrekker in de $i$de cel is. Voor $\epsilon\to 0$ levert dit een continue verdeling. Het is nu mogelijk om op basis van de waarschijnlijkheidsverdeling een oneindig aantal dimensies $D^{(m)}$ te definiëren welke gerelateerd zijn aan de m-de macht van p_i:

$$D^{(m)} = \lim_{\epsilon\to 0} \frac{1}{m-1} \frac{\log \sum_{i=1}^{M} p_i^m}{\log \epsilon}, \quad m = 0,2,3,.... \tag{3.5}$$

Deze formule geldt ook in de limiet $m\to 1$.

Het is eenvoudig na te gaan dat de Hausdorff-dimensie overeenkomt met $D^{(0)}$. De volgende in de rij, $D^{(1)}$, wordt wel de informatie-dimensie genoemd. De uitdrukking (3.5) kan voor $m=1$ geschreven worden als

$$D^{(1)} = \lim_{\epsilon\to 0} \frac{-I(\epsilon)}{\log\epsilon}, \quad I(\epsilon) = \sum_{i=1}^{M} p_i \log 1/p_i, \tag{3.6}$$

waarin $I(\epsilon)$ een maat is voor de informatie dat de aantrekker zich in de betreffende cellen bevindt. $D^{(2)}$ is de correlatie-dimensie en voldoet aan

$$D^{(2)} = \lim_{\epsilon\to 0} \frac{\log C(\epsilon)}{\log\epsilon} \tag{3.7}$$

met

$$C(\epsilon) = \lim_{N\to\infty} \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} H(\epsilon - |s(t_i) - s(t_j)|) \tag{3.8}$$

de correlatie-integraal, welke equivalent is aan de waarschijnlijkheid dat twee willekeurig gekozen punten op de aantrekker binnen een afstand ϵ van elkaar liggen. Er geldt dat in het algemeen de $D^{(m)}$'s dicht bij elkaar liggen en dat

$$D^{(0)} \geqslant D^{(1)} \geqslant D^{(2)}. \tag{3.9}$$

Opgemerkt kan worden dat de dimensies statische grootheden zijn: ze zeggen iets over de geometrie van de aantrekker. Uitgaande van de definitie (3.5) is het vrijwel onmogelijk om de dimensies van aantrekkers in hoger dimensionale toestandsruimten te berekenen. Het tellen van cellen in een steeds fijner rooster vereist veel rekentijd. Een uitzondering vormt de correlatie-integraal en daarmee de correlatie-dimensie. Voor de Hausdorff-dimensie bestaat een alternatief in de vorm van het Kaplan-Yorke vermoeden: dit legt een relatie tussen Lyapunov-exponenten en Hausdorff-dimensie:

$$D^{(0)} = j + \frac{1}{|\lambda_{j+1}|} \sum_{i=1}^{j} \lambda_i \tag{3.10}$$

met

$$0 < \sum_{i=1}^{j} \lambda_i < -\lambda_{j+1}. \tag{3.11}$$

Voor laag-dimensionale systemen, waarvoor het tellen van cellen nog te doen is, blijkt dit te kloppen.

3.3. Invloed van ruis op een vreemde aantrekker

Een vreemde aantrekker gedraagt zich op het oog als een stochastisch gestoord systeem. De vraag is nu: wordt door ruis de vreemde aantrekker aangetast? In eerste aanzet laat deze vraag zich het best beantwoorden voor een gegeven dynamisch systeem dat gestoord wordt met b.v. witte ruis.

Het eenvoudigste systeem met chaotisch gedrag is de logistische afbeelding; gestoord met witte ruis ξ, wordt deze

$$x(n) = ax(n-1)\{1-x(n-1)\}+\xi(n).$$

De Lyapunov-exponent van het systeem zonder ruis voldoet aan

$$\lambda = \lim_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \log|f'(x(i))|$$

met

$$f(x) = ax(1-x).$$

We kunnen deze definitie uitbreiden tot het gestoorde systeem: de functie $\lambda(a)$ blijkt door de ruis niet zo sterk te veranderen. Alleen de smalle periodieke vensters binnen het chaotisch gebied ziet men verdwijnen.

Zet men voor een chaotisch systeem een terugkeerafbeelding uit, b.v. opéénvolgende maxima van x_3 van de Lorenz-vergelijking dan ziet men dat onder invloed van ruis de punten zich concentreren alsof het systeem gestoord periodiek is, zie Grasman en Roerdink [8] voor een vergelijkbare analyse van stochastisch gestoorde relaxatietrillingen in het chaotisch regime. In genoemd artikel is ook het spectrum van dergelijke oscillaties onderzocht. Dit zou in het geval van chaos met ruis weer pieken moeten gaan vertonen. Een dergelijk effect is nauwelijks waar te nemen. Wel werd gevonden dat λ_1 bij toenemende ruis iets toeneemt. Dit is in overeenstemming met de constatering dat in de logistische afbeelding de periodieke vensters opgevuld worden.

4. Reconstructie van een vreemde aantrekker uit een signaal

In de vorige paragraaf leidden we uit de oplossing van een gegeven differentiaalvergelijking van het type (2.1) een tijdreeks $\{s(t_j)\}$, $j = 1,2,3,...$ af met $t_j = j\tau$ en τ voldoende klein. We veronderstellen nu dat niet de vector $s(t_j)$ afgelezen kan worden maar een enkele variabele $y(t_j)$, welke op een deterministische, doch verder onbekende wijze, gerelateerd is aan de toestandsvector $s(t_j)$.

4.1. Vertraagde coördinaten

Kunnen we op basis van het signaal $y(t_j)$ eigenschappen van de aantrekker reconstrueren? Dit blijkt inderdaad het geval te zijn. Wolf et al. [15] geven hiervoor een Fortranprogramma en laten zien op welke ideeën het algoritme gebaseerd is. Uit de opéénvolgende waarden van y kan de d-dimensionale vector $z(t)$ geconstrueerd worden

$$\begin{aligned} z_1(t) &= y(t), \\ z_2(t) &= y(t-\tau), \\ z_3(t) &= y(t-2\tau), \\ &\text{---------} \\ z_d(t) &= y(t-(d-1)\tau). \end{aligned} \tag{4.1}$$

De waarde van d dient voldoende groot te zijn, opdat de dimensie D van de aantrekker die zich beweegt in de originele toestandsruimte $\mathbb{R}^n$ ook binnen $\mathbb{R}^d$

past. In de mathematische analyse die Takens [14] van dit probleem geeft, wordt een voldoende voorwaarde voor d afgeleid. De tijdreeks $y(t_j)$ levert de verzameling afbeeldingen $z(t_j) \mapsto z(t_{j+1})$.

4.2. Lyapunov-exponenten

Gezien het feit dat een aantrekker vele malen dicht in de buurt komt van posities, die in het verleden ingenomen zijn, kan men bij het volgen van $z(t_j)$, $j = 0,1,2,...$ ook de dynamica van punten in de directe omgeving reconstrueren. Men zoekt in de tijdreeks een aantal nabij gelegen punten: $z(t_{j_1}), z(t_{j_2}),...$ en weet waar zij een tijdstap verder afgebeeld worden. Op basis hiervan kunnen de Lyapunov-exponenten bepaald worden. Volgt men $z(t_j)$, $j = 0,1,2,...$ dan kan locaal het uitéénlopen van oplossingen bepaald worden uit $z(t_j+1)$ en $z(t_{j_1}+1)$. Dit levert na middeling λ_1. De strekking van een oppervlakte-element volgt uit $z(t_j+1)$, $z(t_{j_1}+1)$ en $z(t_{j_2}+1)$ en bepaalt $\lambda_1+\lambda_2$, waaruit λ_2 volgt. Op deze wijze kunnen λ_i, $i = 1,2,...,j$ bepaald worden, waarvoor $0<\Sigma_{i=1}^{j}\lambda_i<-\lambda_{j+1}$. Dit levert als onder- en bovenwaarde voor de dimensie van de vreemde aantrekker resp. j en $j+1$. De Lyapunov-exponenten zijn invariant onder een transformatie van de toestandsvariabelen, zodat uit het systeem in z inderdaad conclusies getrokken kunnen worden over het systeem in x.

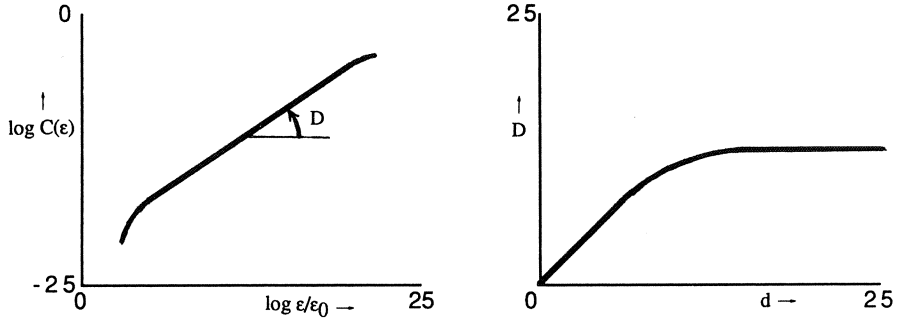
4.3. Voorspellen

Op basis van de tijdreeks $y(t_j)$, $j = 0,1,2,...$ kan met behulp van het model met vertraagde coördinaten z een voorspelling gemaakt worden van de waarde van y in de toekomst; zeg op het tijdstip $t+T$. We doorlopen de gegeven tijdreeks over het interval $t_j \in [0,t)$ en registreren de waarden $z(t_{j_i})$, $i = 1,2,...,k$ van de k het dichtst bij $z(t)$ liggende waarden. Een geschikte interpolatie van $z(t_{j_i}+T)$, $i = 1,2,...,k$ levert een voorspelling van $z(t+T)$. De kracht van een dergelijk voorspellingsmodel ligt in het behoud van het niet-lineaire karakter in het voorspelsysteem, n.l. de oplossingscurven uit een verleden. Een dergelijk voorspellingsmodel kan dus in principe plotselinge toekomstige veranderingen verwerken. Echter gezien het sterke divergente karakter van oplossingen kan het voorkomen dat $z(t_{j_i}+T)$, $i = 1,...,k$ zich over twee ver uit elkaar liggende subdomeinen van de toestandsruimte uitspreidt, zodat het antwoord sterk van de gekozen interpolatietechniek afhangt: een 'gemiddelde' oplossing kan dan in een domein terecht komen dat ver van de aantrekker verwijderd is. Farmer en Sidorowich [5] geven een methode om tot een geschikt interpolatieschema te komen.

4.4. Signaal-analyse van niet-lineaire fysische en biologische systemen

Op basis van een signaal kan met behulp van de correlatie-integraal een schatting van de dimensie van een vreemde aantrekker gemaakt worden. De methode is als volgt. Het systeem $z(t)$ in vertraagde coördinaten wordt eerst zeer laag dimensionaal genomen, $d = 1,2,3$. Dit is voor de vreemde aantrekker een te krap jasje. In deze fase vindt men uit het signaal $D^{(2)} = d$. Neemt d

verder toe dan treedt verzadiging op, zie fig.2. De asymptotische waarde voor toenemende d levert een schatting voor $D^{(2)}$. Als schatting voor de dimensie van een vreemde aantrekker is hiermee het volgende gevonden: het weer en ook het klimaat hebben $D = 3$ à 4 (Fraedrich [7]). De hersenactiviteit heeft $D = 6$ à 7 (slapend) en $D \approx 4$ (met ogen dicht), zie Mayer-Kress en Holzfuss [12]. De waarde die aan deze uitkomsten moet worden toegekend is iets dat nog verdere studie vereist. Door de beschikbaarheid van moderne computers voor het doorrekenen van grote datasets, waarvan de correlatie-integraal te bepalen is, dreigt de methode te bezwijken onder de 'successen' die er mee geboekt worden. De vraag is of een fysisch of biologisch probleem echt van de vorm $\dot{x} = f(x)$ is, dus eindig-dimensionaal, tijdsonafhankelijk en zonder vertragingen. Stel dat dit zo is, dan nog blijft er het effect van stochastische storingen, hetgeen in de vorige paragraaf voor expliciet gegeven systemen al tot complicaties leidde.

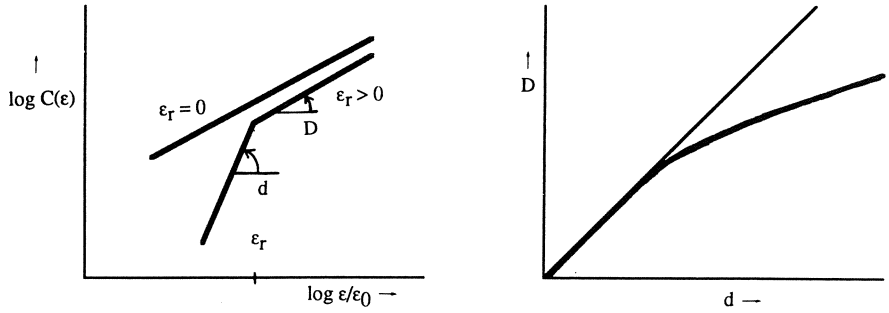


(a) Bepaling correlatiedimensie bij vaste d (b) Verzadiging bij toenemende d

FIGUUR 2. Bepaling van de dimensie uit de correlatie-integraal

4.5. Ruis en de dimensie van de aantrekker

Zoals onderzocht is door Ben-Mizrachi et al. [1] voor de Hénon-afbeelding heeft ruis van intensiteit ϵ_r bij het bepalen van de correlatie-integraal het volgende effect. De ruis maakt dat in de d-dimensionale toestandsruimte het systeem een bol met straal ϵ_r rond de punten van de aantrekker vult. Het gevolg is dat voor $\epsilon < \epsilon_r$ als dimensie voor de aantrekker de waarde d gevonden



(a) De Hénon-aantrekker in een systeem met $d = 3$

(b) De correlatiedimensie bij een meer gecompliceerd systeem met ruis: er treedt geen verzadiging op

FIGUUR 3. Bepaling van de dimensie bij ruis

wordt. Voor $\epsilon >> \epsilon_r$ heeft de random storing geen effect. Echter de correlatiedimensie volgt uit (3.7) voor $\epsilon \to 0$. Er is dus sprake van tegenstrijdigheid in de eisen die aan ϵ opgelegd worden. In fig.3a is de ideale situatie uit het experiment met de Hénon-afbeelding geschetst. In fig.3b is het eindresultaat gegeven voor de dimensie van een vreemde aantrekker in een signaal afgeleid uit de atmosferische circulatie over een periode van 15 jaar (Fraedrich [7]): er treedt geen volledige verzadiging op.

LITERATUUR

1. A. BEN-MIZRACHI, I. PROCACCIA, P. GRASSBERGER (1984). Characterization of experimental (noisy) strange attractors. *Phys. Rev. A. 29*, 975-977.
2. B.L.J. BRAAKSMA, H.W. BROER, G.B. HUITEMA (1988). *Toward a Quasi-periodic Bifurcation Theory*, Report Univ. Groningen.
3. E. DOEDEL (1986). *Auto: Software for Continuation and Bifurcation Problems in Ordinary Differential Equations*, Report Caltech, Pasadena.
4. J.P. ECKMANN (1981). Roads to turbulence in dissipative dynamical systems. *Rev. Modern Phys. 53*, 643-654.
5. J.D. FARMER, J.J. SIDOROWICH (1987). Predicting chaotic time series. *Phys. rev. Letters. 59*, 845-848.
6. M.J. FEIGENBAUM (1978). Quantitative universality for a class of nonlinear transformations. *J. Stat. Phys. 21*, 669.
7. K. FRAEDRICH (1986). Estimating the dimensions of weather and climate attractors. *J. Atmos. Sci. 43*, 419-432.
8. J. GRASMAN, J.B.T.M. ROERDINK (1988). *Stochastic and Chaotic Relaxation Oscillations*, preprint 540, Dept. of Math., Utrecht Univ. Te verschijnen in *J. Stat. Phys.*
9. J. LIGHTHILL (1986). The recently recognized failure of predictability in Newtonian dynamics. *Proc. Roy. Soc. London A. 407*, 35-50.
10. E.N. LORENZ (1963). Deterministic nonperiodic flow. *J. Atm. Sc. 20*, 130-141.

11. E.N. LORENZ (1984). Some aspects of atmospheric predictability. D.M. BURRIDGE, E. KALLEN (eds.). *Problems and Prospects in Long and Medium Range Weather Forcasting*, Springer-Verlag, Berlin, 1-20.
12. G. MAYER-KRESS, J. HOLZFUSS (1987). Analysis of the human electyro-encephalogram with methods from nonlinear dynamics. L. RENSING, U. VAN DER HEIDEN, M.C. MACKEY (eds.). *Temporal Disorder in Human Oscillatory Systems*, Springer-Verlag, Berlin, 57-68.
13. H.G. SCHUSTER (1984). *Deterministic Chaos*, Physik-Verlag, Weinheim.
14. F. TAKENS (1981). Detecting strange attractors in fluid turbulence. D. RAND, S. YOUNG (eds.). *Dynamical Systems and Turbulence*, Springer-Verlag, Berlin.
15. A. WOLF, J.B. SWIFT, H.L. SWINNEY, J.A. VASTANO (1985). Determining Lyapunov exponents from a time series. *Physica 160*, 285-317.

Stochastiek en de computer

Statistische Informatica in de Praktijk

W.J. Keller
Centraal Bureau voor de Statistiek, Hoofdafdeling Automatisering
Postbus 959, 2270 AZ Voorburg
Vrije Universiteit, F.E.W.E.C.
Postbus 7161, 1007 MC Amsterdam

De rol die de statistische informatica, met name op het CBS, speelt bij het statistische produktieproces wordt besproken aan de hand van een nieuw CBS-onderzoek: de Enquête Beroepsbevolking. Vier deelprocessen van het statistische produktieproces komen aan de orde: de verzameling van gegevens, het controleren en corrigeren van de gegevens, de transformatie van de individuele gegevens naar schattingen voor populatiekenmerken en het analyseren en het presenteren van de uitkomsten. Tevens wordt het door het CBS ontwikkelde computergestuurde enquêteersysteem BLAISE kort besproken.

1. Inleiding

Statistische Informatica is een voor de hand liggende (en wat modieuze) titel voor het grensgebied van statistiek en informatica. (Ik prefereer de term informatica boven informatiekunde omdat de laatste term mijns inziens een soort zwaktebod is.) Gezien mijn achtergrond hoop ik dat U begrip heeft voor mijn keuze om me in deze voordracht te beperken tot de *beschrijvende* statistiek zoals die bij het CBS onderwerp van studie is. De wiskundige statistiek zal dus, vrees ik, in het hierna volgende wat minder aan bod komen. Ook de liefhebber van een goed statistisch pakket op de PC moet ik teleurstellen; de keuze daarvan heb ik bij een andere gelegenheid al besproken (zie o.a. Keller, 1986).

Om tot een (CBS) statistiek te komen, zijn een groot aantal stappen nodig, stappen waarbij de informatica een minstens even grote rol speelt als de statistiek. Ik zal mij in dit verhaal beperken tot 4 deelprocessen: het verzamelen van gegevens (meestal op steekproefbasis), het 'editten' van de gegevens (ook wel genoemd het controleren en corrigeren van de gegevens), het transformeren van individuele gegevens naar schattingen van populatiekenmerken (vaak het 'ophogen' genoemd) en tenslotte het analyseren en het presenteren van de uitkomsten. Daarnaast speelt de gegevensbeheersing ('datamanagement') bij dit alles een belangrijke rol.

Hieronder zal ik de rol van de informatica bij al deze vijf processen beschrijven aan de hand van een gloednieuw CBS onderzoek: de Enquête BeroepsBevolking (kortweg de EBB). We beginnen met een beschrijving van de eerste twee fases van het statistische produktieproces: gegevensverzameling en de daarop volgende controle/correctie. Aangezien we ons in concreto op de

EBB zullen richten, is het referentiekader dat van de persoons- en gezinsenquêtes.

2. Gegevens verzamelen en corrigeren

Zoals we zullen zien, kunnen we bij de beschrijving van het gehele statistische produktieproces dankbaar gebruik maken van een onderscheid tussen 'record-based' en 'file-based' processing, of, in wat beter Nederlands gezegd: transactie- en bulkverwerking. Bij transactieverwerking is er sprake van een cyclus per record (pakweg: per respondent), bij bulkverwerking van een cyclus per bestand. Het aardige is dat het type computer dat voor beide 'slagen' het beste geschikt is, verschilt: transactieverwerking gaat beter op micro's terwijl bulkverwerking (nog?) op de mainframecomputer plaatsvindt.

De traditionele werkwijze, die op alle statistische bureaus (ook het CBS) het meest wordt toegepast, is de volgende: Na het definiëren, het ontwerpen en het drukken van de (papieren) vragenlijst, worden de gegevens op de vragenlijsten door de enquêteur (of zijn er alleen enquêtrices?) thuis bij de respondent ingevuld, waarna de vragenlijsten naar het bureau worden verstuurd voor verwerking. Deze verwerking, die uiteindelijk een zogenaamd 'schoon' bestand moet opleveren, is traditioneel een merkwaardige mengeling van transactie- en bulkverwerking. Na handmatige controle van de vragenlijsten (o.a. op compleetheid en netheid van invullen), de typering van open antwoorden (beroep 'directeur' = code '72436') en de daaruit voortvloeiende (handmatige) correcties, worden de bewerkte vragenlijsten op het data-invoercentrum door datatypistes ingevoerd.

Het daaruit resulterende bestand is nog bij lange na niet schoon. Vandaar dat met behulp van omvangrijke (Cobol) programma's fouten worden gesignaleerd die op lange lijsten worden uitgeprint. Aan de hand van deze lijsten en de originele formulieren worden verbeteringen op de lijsten aangebracht welke opnieuw door de datatypistes worden ingevoerd. Deze verbeteringen worden opnieuw gecontroleerd, wat wederom resulteert in een foutenlijst, die weer aanleiding is tot nieuwe verbeteringen, etc. Gewoonlijk wordt de cyclus 'invoeren-fouten signaleren-verbeteren' zo'n drie maal herhaald voordat het aantal resterende fouten acceptabel klein is. Aangezien het gehele bestand iedere keer een cyclus doorloopt, wordt de produktietijd dan ook bepaald door de som van de cyclustijden en dus door het aantal cycli en hun duur. Met andere woorden, het traditionele verwerkingsproces is een repeterend en sequentieel proces, en zoiets kost tijd, veel tijd.

Hoe gaat het nou bij de EBB? Wel, we hebben ons op het CBS gerealiseerd dat het boven omschreven proces twee problemen omvatte: het vertaalprobleem (van antwoord naar registratie, naar waarde, naar toetsaanslag, naar uiteindelijk een schoon en machineleesbaar gegeven) en een procesbeheersingsprobleem, voornamelijk veroorzaakt door het sequentiële- en bulkkarakter van het verwerkingsproces. Beide problemen worden zoveel mogelijk vermeden bij de EBB: door thuis direct de gegevens van de respondent in een draagbare computer in te laten voeren door de enquêteur en ter plekke de computer de controles te laten uitvoeren, verkorten we het

vertaaltraject en vermijden we de cycli: aan het eind van het interview beschikken we over schone en machineleesbare gegevens. Merk op dat de controlecyclus per interview (per record) gebeurt in plaats van per bestand: ieder afgerond interview is schoon, op de typering na. Tot slot worden de interviewgegevens iedere nacht automatisch per telefoon doorgegeven aan een netwerk van microcomputers op het CBS, terwijl tevens de batterijen van de computer worden opgeladen. Daarbij neemt de computer van de enquêteur het initiatief (elk op een ander tijdstip) en wordt de verbinding met behulp van een zgn. auto-dial modem volautomatisch tot stand gebracht. (Voor historici: de experimenten met 'handheld' computers op het CBS stammen uit 1984, zie Bemelmans-Spork e.a. [1] en Bemelmans-Spork e.a. [2].)

Het coderen van open antwoorden (typeren) van de gegevens gebeurt interactief en parallel: op het PC-netwerk op het CBS krijgen typeurs (ons woord voor codeurs) de relevante open antwoord-teksten aangeboden tesamen met suggesties voor de code ('directeur' = '72436'). Naast tekst-herkenningsroutines ('darecteur' = 'directeur') spelen hierbij menu's een belangrijke rol, wat voor de hand ligt gezien de hiërarchische structuur van de meeste classificaties (1ste digit, 2de digit, etc.). Het gebeurt interactief omdat automatische tekstherkenningsmethoden nooit 100% succesvol zijn en omdat de meeste van onze coderingen (helaas) niet context-vrij zijn: de codering van het beroep hangt alleen af van de omschrijving van het beroep maar ook van de opgegeven opleiding of inkomen.

Vergelijken we nu interactief controleren en corrigeren achter de microcomputer met de traditionele methode, dan valt op welke grote winst hier met de (micro)computer kan worden behaald. Alles gaat sneller, efficiënter en de bewerkers vinden het leuker. Door de parallelle (in plaats van sequentiële) verwerking kan ook hier de totale doorlooptijd van een statistiek eenvoudig bekort worden door meer bewerkers en meer micro's in te zetten. Ik heb hieruit een belangrijke les geleerd: probeer niet altijd alles van voor tot achter bulksgewijs te automatiseren, doch streef naar parallelle, zonodig interactieve processen waarbij de computer de mens ondersteunt doch niet noodzakelijkerwijs vervangt. Zie ook Bethlehem [5].

Alhoewel het zoals hierboven beschreven EBB-verwerkingsproces tot één van de modernste ter wereld behoort, is het toch nog niet ideaal. Iedere wijziging in de vragenlijst leidt tot wijzigingen in de programmatuur voor de draagbare computer, de datacommunicatie, de decodering van de gegevens en de typering, terwijl ook de programmatuur voor de verdere verwerking (ophogen, tabelleren, analyse) bij wijzigingen aangepast moet worden. De wijzigingen in de draagbare computer zijn relatief het eenvoudigst omdat de 'handheld' enigszins vragenlijst-gestuurd werkt: een interpretator in de computer leest keer op keer de vragenlijst, die in een wat rudimentaire vragenlijst-taal (QUEST) is gespecificeerd. Alhoewel bij de EBB aanpassingen van de vragenlijst in QUEST niet eenvoudig zijn vanwege de buitengewoon complexe structuur van deze vragenlijst, is dit toch beter mogelijk dan aanpassing van de programmatuur op het netwerk die niet vragenlijst-gestuurd is. Dit brengt ons bij de volgende generatie CBS-programmatuur, genaamd BLAISE. Alhoewel al eerder

over BLAISE is gerapporteerd, kan ik het niet nalaten hier nog even apart bij stil te staan.

3. CAPI, CATI EN CADI MET BLAISE

BLAISE is ontstaan (zie ook Denteneer e.a. [9]) uit onvrede met het traditionele controle/correctie proces, zoals beschreven aan het begin van de vorige paragraaf, en als vervolg op onze ervaringen met QUEST. Terwijl QUEST uitsluitend bedoeld was voor het enquêteren met draagbare computers ('Computer Assisted Personal Interviewing' ofwel CAPI), was BLAISE in eerste instantie bedoeld voor de verwerking van de duizenden (papieren) formulieren die dagelijks bij het CBS binnenkomen. De redenatie was als volgt: als we nu in plaats van de traditionele handbewerking gevolgd door bulkcycli, de informatie van het formulier direct door de handbewerkers laten overtypen in een microcomputer, die net als bij QUEST alles van de vragenlijst weet, zodat 'formulier klaar' betekent 'gegevens schoon'. Op deze wijze kunnen we bijna dezelfde winsten boeken als bij de handheld, doch zonder iedere enquêteur een draagbare computer te hoeven geven. 'Alles van de vragenlijst weten' betekent net als bij QUEST dat de computer de vragen, de antwoordmogelijkheden (geslacht=man, vrouw, weetniet), de route door de vragenlijst (ALS respondent-heeft-werk DAN vraag-werk-vragen ANDERS andere-vragen) en de relatiecontroles (ALS leeftijd<16 DAN burgstaat<>ongehuwd) kent en dus fouten kan signaleren. Omdat BLAISE in eerste instantie bedoeld was voor intelligente-interactieve-data-invoer-van-papier, hebben we deze vorm van verwerking CADI oftewel 'Computer Assisted Data Input' genoemd. CADI is sinds begin 1987 in produktie op het CBS en heeft zijn waarde reeds bewezen: bijv. de ESM'87 (Enquête Slachtoffers Misdrijven, ca. 15000 personen) ging 2 maanden later dan gewoonlijk in bewerking en kwam er, dankzij CADI, een paar maanden eerder uit.

Een belangrijk verschil tussen QUEST en BLAISE is de vragenlijsttaal, die bij BLAISE veel leesbaarder en ook veel krachtiger is. Daarnaast genereert de BLAISE-compiler direct executeerbare code, die veel sneller is dan de interpretator van QUEST. Behalve voor CADI zijn momenteel ook BLAISE compilers voor CAPI en CATI ('Computer Assisted Telephone Interviewing') beschikbaar, zodat dezelfde vragenlijst in BLAISE zonder aanpassingen gebruikt kan worden voor alle drie de vormen van gegevensverzameling. Terzijde: BLAISE draait momenteel op MS-DOS computers, zodat de tijd van dure mini's voor CATI binnenkort voorgoed voorbij is. Voor CAPI wordt gebruik gemaakt van een lichte (minder dan drie kilo wegende), draagbare MS-DOS computer met een goed scherm, een floppy disk en een 'non-volatile' RAM-geheugen.

De aanleiding om over BLAISE te beginnen, was het niet volledig vragenlijst-gestuurd zijn van de EBB-verwerking. Wel, om eerlijk te zijn, ook BLAISE kan momenteel niet vanuit de vragenlijst-definitie de programmatuur voor het gehele EBB-proces genereren. Wel kunnen we vanuit de vragenlijst een SPSS-setup aanmaken, zodat bij eenvoudige surveys direct na het invoeren

in een BLAISE CAPI/CADI/CATI-systeem getabelleerd en geanalyseerd kan worden, zonder dat de vragen, de waarden, de labels, de weet-niet-coderingen etc. opnieuw gespecificeerd moeten worden. Dit is de kern van BLAISE: geef één keer goed en helder aan hoe de vragenlijst in elkaar zit en leid daaruit de data-definities/dictionairies voor alle volgende verwerkingsstappen af. Setup-generatoren voor STATA (een fijn en krachtig pakket op de micro) en Paradox (een databasemanagement pakket) zijn ook beschikbaar in de nieuwste versie 2.0 van BLAISE.

In BLAISE 2.0 zit ook een typeer-module. De gedachte is wederom dat de typeer-programmatuur automatisch gegenereerd wordt vanuit de BLAISE vragenlijst, waarin de typeerkennis gespecificeerd is. De gegenereerde typeerprogrammatuur ondersteunt zowel hiërarchische menu-typering (kies 1ste digit, bijv. voeding, 2de digit, bijv. brood, 3de digit, bijv. gesneden witbrood) als tekstherkenning mogelijk maken (geef alle codes met 'brood'). Vanzelfsprekend kan hier dankbaar gebruik worden gemaakt van de mogelijkheid om in BLAISE met modules te werken, zodat bijv. een beroepen-classificatie slechts één keer gespecificeerd hoeft te worden en daarna in alle enquêtes kan worden meegenomen ('ge-linked'). Deze modulaire opbouw van BLAISE komt trouwens ook de standaardisering van enquêtes ten goede: waarom een nieuwe huishoudbox-definitie ontwerpen als er al een dergelijke module op de disk staat? Meer over BLAISE in Bethlehem e.a. [6,7].

BLAISE en QUEST zijn beide voorbeelden van interactieve, transactiegerichte systemen waar door parallelle inzet van veel kleine computers (EBB: ca. 400 draagbare CP/M systemen van ieder 2 kilo) zelfs bij gigantische surveys (EBB: ca. 200 000 personen per jaar) grote winsten kunnen worden geboekt. (Begrijpt U nu waarom ik soms toch van een volkstelling droom?)

4. Ophogen en tabelleren

Wanneer we, met of zonder hulp van systemen als QUEST of BLAISE, uiteindelijk een 'schone' tape voor ons op het bureau hebben liggen, komt de vraag 'Wat nu?'. Wel, voor een Bureau voor de Statistiek is die vraag eenvoudig te beantwoorden: tellen, totaliseren en tabelleren. Echter, aangezien de gegevens zelden uit een integrale telling zijn verkregen (zei iemand iets over een volkstelling?) en de non-respons mogelijk selectief kan zijn, kunnen we de onzuiverheid en de variantie van de betreffende populatieschatters, gegeven een aantal veronderstellingen, verbeteren door de gegevens op te hogen, dat wil zeggen ieder record zodanig van een gewicht te voorzien dat de gewogen verdeling van de steekproef, met betrekking tot bekende kenmerken als leeftijd, geslacht, burgelijke staat en regio, de populatie weerspiegelt. Aangezien het woord representatieve steekproef toch al zo vaak misbruikt is, zal ik het ophogen voorstellen als het rechttrekken van de representativiteit met betrekking tot een aantal kenmerken waarvan de populatieverdeling bekend is.

Ophogen kunnen we het eenvoudigst voorstellen aan de hand van twee tabellen, één voor de steekproef en één voor de populatie; bijvoorbeeld:

AANTAL PERSONEN NAAR LEEFTIJD, GESLACHT EN REGIO

Steekproef (fantasie)

	mannen			vrouwen		
	1-19	20-44	45-99	1-19	20-44	45-99
Friesland	0	10	5			
Drenthe	etc.					
Groningen						
Rest. Ned.						

Populatie (×1000)

	mannen			vrouwen		
	1-19	20-44	45-99	1-19	20-44	45-99
Friesland	100	200	400			
Drenthe	etc.					
Groningen						
Rest. Ned.						

Uit deze tabellen kunnen de ophooggewichten worden bepaald voor iedere combinatie van de drie variabelen door de populatie-aantallen te delen door de steekproefaantallen. Dus het ophooggewicht voor (Friesland, mannen 20-44) is 200000/10=20000. Een probleem bij het ophogen is de geringe celvulling. In het bovengenoemde voorbeeld zien we bijv. dat in de regio Friesland door het toeval (het gaat immers om een steekproef) geen mannelijke respondenten onder de twintig zijn aangetroffen. Als we nu willen ophogen naar de combinatie Leeftijd × Geslacht × Regio (kortweg L×G×R) dan lukt dat niet, aangezien het gewicht voor mannelijke respondenten in Friesland oneindig is. Te bewijzen is tevens dat schatters gebaseerd op de andere gewichten geen goede schatters opleveren voor bijv. het totaal. In de praktijk wordt opgehoogd met honderden soms duizenden cellen, zodat zelfs bij grote surveys dit probleem zich kan voordoen. Het onaangename is dat dan de programmatuur, afhankelijk van hoe de steekproef bij toeval is uitgevallen, stukloopt.

Een ander probleem bij de ophoging is dat soms van de populatie niet de volledige kruising (bijv. L×G×R) bekend is, doch slechts de tweeweg-tabellen (bijv. L×G, G×R en L×R). Beide problemen (slecht gevulde cellen en onvolledige populatie-informatie) kunnen worden opgelost door het ophogen te zien als een stap van een 'model-based' schattingsprocedure, waarbij de afhankelijke variabele (hier het aantal werklozen) een lineaire relatie heeft met de zgn. hulpvariabelen, hier L, G, en R. Door nu in een dergelijk ANOVA model hogereorde interacties in de design matrix weg te laten (dus bijv. L×G+G×R+L×R als model te kiezen), kunnen ook daar waar populatie-informatie ontbreekt en waar steekproefcellen slecht gevuld zijn, goede schatters worden geconstrueerd (zie ook Bethlehem en Keller [3] en Bethlehem en Keller [8]).

Soms echter geeft het, zoals wij dat noemen, ophogen naar de marginalen (met een niet volledig-verzadigd ANOVA model) niet altijd de oplossing. Het kan zijn dat juist de drieweg-interactie tussen L, G en R van betekenis is voor het niveau van de werkloosheid, zodat we misschien beter mannelijke respondenten jonger dan twintig in bijv. Friesland en Groningen bij elkaar kunnen nemen om lege cellen te voorkomen. Echter, in 10 dimensies heeft iedere cel legio buren, terwijl bij nominale gegevens als burgelijke staat zelfs het begrip 'buur' moeilijk te definiëren is. Voordat je het weet, is zoiets simpels als ophogen dan uitgegroeid tot een complete multivariate (cluster?) analyse, waarbij als afhankelijke variabele gekozen kan worden uit enerzijds de werkloosheid en anderzijds de mate van respons (beide bepalen namelijk de kwaliteit van de ophoging).

Na de ophoging staat meestal als eerste het tabelleren op het menu: Den Haag wil altijd zo snel mogelijk wat kerncijfers zien. Tabellen lijken eenvoudige dingen, maar nadere bestudering laat zien dat met name de lay-out van de rand, de 'statistics' in de cel, de wijze van percenteren, de plaatsing van totalen, de verwerking van 'multipele respons scores' en nog vele andere zaken het leven hier moeilijk maken. Programmatuur om te tabelleren verschilt onder andere in de aansturing (TPL is een belangrijke tabel-specificatie taal waarop de meeste pakketten zoals SPSS/Tables, SAS en P-Stat hun tabelleerprogramma hebben gebaseerd), de wijze van tellen (in intern geheugen of op disk) en de aan te sturen printers (SPSS/PC+/Tables kan bijvoorbeeld camera-ready tabellen op de laserprinter maken). Voor het maken van veel tabellen van grote bestanden zijn er pakketten zoals TAU die Cobol programma's genereren, die op hun beurt na gecompileerd te zijn, zeer efficiënt stapels papier produceren. Om even een tabelletje uit te proberen, zijn deze krachtpatsers minder geschikt. Daarnaast vraag ik me toch af of al die batch-georiënteerde tabelleerprogramma's het eeuwige monopolie zullen hebben in een tijd van meer en meer interactieve programmatuur zoals spreadsheets (toch een soort tekstverwerker voor tabellen, nietwaar?) en Desk Top Publishing op de PC.

Als het zo eenvoudig was dat het EBB-proces na de dataverzameling en het typeren verder alleen nog maar bestond uit ophogen en tabelleren op een rechthoekig bestand, dan waren we gauw klaar. Dat wil zeggen, als we de vragenlijst in BLAISE hadden gespecificeerd, want dan wordt automatisch een (SPSS-) beschrijving van het bestand gegenereerd. Deze kunnen we gebruiken als input voor onze home-made standaardprogrammatuur voor het ophogen, genaamd LINWEIGHT, en voor het tabelleren met bijv. SPSS. Helaas is de EBB (nog) niet met BLAISE gedaan, en betreft het geen simpel rechthoekig bestand en geen simpele ophoging. Op het eerste onderwerp kom ik dadelijk terug, het tweede ligt buiten de scope van dit artikel. Ik volsta met te vermelden dat de EBB eigenlijk een longitudinaal onderzoek is vanwege de vele retrospectieve vragen ('wanneer bent U dan van baan veranderd?'). De schatting van de verandering in de werkloosheid is hierdoor zeer complex: niet alleen hebben we bijv. gegevens over de maand maart van mensen die in maart zijn geënquêteerd, maar ook retro-informatie over maart van de mensen die in

april zijn ondervraagd, alsook informatie van de ondervraagden over de verandering van, zeg, februari op maart. Combineren van al deze gegevens leidt tot zeer gecompliceerde schattingsmethoden.

5. De gegevensopslag

Zoals uit bovenstaand exposé over ophogen volgt, kan voor het berekenen van de ophooggewichten worden volstaan met geaggregeerde gegevens. We hebben voldoende aan één of meer (meerdimensionale) tabellen, zowel wat steekproef als populatie betreft. Dit brengt ons meteen op een interessant probleem in de statistische informatica: de opslag van hoogdimensionale tabellen. Wanneer we, bijv. bij de EBB, zouden willen ophogen naar 10 hulpvariabelen ieder met 10 categorieën, dan praten we over maar liefst 10^{10} cellen, een aantal dat zelfs voor onze microcomputers met extended memory (EMS: 8 MByte) iets te veel van het goede is (om over een oude mainframe maar te zwijgen: de Cyber 180-855 onder NOS geeft het al op boven 1 MB!). Opmerkelijk is dat de opslag van het microdata materiaal (20 000 records per maand bij 10 variabelen) nog best in het RAM geheugen van ons PC-tje met EMS zou passen, en zonder veel problemen (afgezien van een extra koffiepauze) met STATA versie 2.0 verwerkt kan worden. En aangezien design matrices slechts bestaan uit nullen en enen, vaak ook nog op een buitengewoon ritmische wijze aan elkaar geplakt, is hier werk aan de winkel voor informatici met capita-selecta in optimale opslagmethoden. (Hetzelfde probleem heb je trouwens bij hoogdimensionale log-lineaire modellen, maar dat is afdeling analyse.) Zie ook Denteneer en Verbeek [9] en Kelderman [11].

Een veel gebruikt alternatief, als het gaat om de opslag van weinig variabelen en veel records, is het sorteren en intellen van records. Alhoewel het aantal mogelijke patronen bij 10 categorale variabelen met gemiddeld 10 categorieën ca. 10^{10} is, komt het vaak voor dat bepaalde combinaties in het geheel niet voorkomen, zodat het aantal verschillende antwoordpatronen bij 10 vragen veel minder is dan het aantal records. Door nu het bestand te sorteren en opeenvolgende gelijke records samen te nemen en met een gewicht gelijk aan het aantal gelijke records op te slaan, kan zonder informatieverlies het bestand worden gecomprimeerd. Door het tellen van gelijke records direct in het sorteer-algoritme mee te nemen, kan een dergelijk indikprogramma bovendien buitengewoon efficiënt worden geïmplementeerd.

Aangezien voor ophogen, tabelleren en analyse vaak slechts een beperkte subset van de variabelen nodig is, terwijl de verwerking van statistische gegevens vaak sequentieel is, is de zgn. 'inverted-file' structuur in sommige statistische bureaus in trek. Hierbij wordt, simpel gezegd, de data-matrix op zijn kant gezet: in plaats van records achter elkaar, worden variabelen achter elkaar gezet, soms zo dat per variabele een aparte file wordt gecreëerd. Het gevolg is dat bijv. voor een tabel op drie variabelen slechts drie kleine files sequentieel geheel worden gelezen en ingeteld. Een voordeel is tevens dat de aldus geconstrueerde files homogener zijn: bijv. bij geslacht = man, vrouw kunnen de waarden efficiënt als een stroom bits worden opgeslagen.

Zoals hierboven is betoogd, is de micro niet geheel machteloos als het gaat

om de analyse van erg grote gegevensbestanden. Een analyse van pakweg 25 variabelen uit de jaar-EBB (met ca. 200 000 records) kan, als het gaat om lineaire modellen, makkelijk in een gewone PC met 640 Kb plaatsvinden als de X'X matrix maar op het mainframe berekend wordt en daarna als matrix wordt overgeheveld naar de micro. Zijn alle gegevens categoraal en voorgesteld als dummies, dan is X'X niets anders dan een wat uit de kluiten gegroeide tabel met zeg 250 $\times$ 250 cellen (als iedere variabele gemiddeld 10 categorieën heeft). Opslag hiervan kost op een PC zo'n 125 Kbytes, zodat er nog best wat te analyseren valt. Bij continue gegevens met 8 byte reals kost de opslag van X'X slechts 50 KBytes.

Tenslotte, zoals al eerder gezegd, is het gegevensbestand vaak niet rechthoekig, en zo ook bij de EBB. Met een aantal potentiële vragen dat in de honderden loopt (gelukkig hoeven de meeste respondenten maar een deel daarvan te beantwoorden), is het opslaan van het gegevensbestand in de vorm van een 'plat' bestand (de datamatrix) niet praktisch vanwege de hiërarchische structuren in de gegevens. Voorbeelden zijn de relatie adres-gezin, gezin-persoon, persoon-werkkringen (incl. vorige werkkring), persoon-opleidingen, etc. Daarnaast zorgt de route door de vragenlijst voor bijv. personen jonger dan 16 jaar voor een zo goed als blanco record. Het zal duidelijk zijn dat alleen al voor de efficiënte opslag van dergelijke bestanden zgn. data-base-management systemen van pas komen. De meest recente versies van statistische pakketten zoals SPSS, STATA en SAS, evenals BLAISE 2.0, bieden hiervoor ook enige faciliteiten in de vorm van 'joins', waardoor records uit bijv. het huishoudbestand op sleutel aan een persoonsrecord kunnen worden verbonden. Eén en ander gebeurt conform de theorie van zgn. relationele databases, waar de records pas tijdens de daadwerkelijke selectie bij elkaar gezocht worden.

Het voordeel is een zeer grote flexibiliteit in de zoekpaden, doch de prijs is vaak een matige prestatie, speciaal als de sleutels sequentieel afgezocht moeten worden zonder hulp van snelle zoekmethoden als zgn. B+ bomen (over datastructuren als deze, zie bijv. Kruse [13]). Voor de statistiek, waar de verwerkingsgang toch vaak sequentieel ('van boven naar beneden') is in plaats van willekeurig ('random') zou het fijn zijn als rekening gehouden wordt met de sortering van de bestanden. Het is dan ook de vraag of de vervanging van Cobol door vierde generatie talen als Oracle en Ingres voor de statistici dezelfde belofte inhoudt als elders vaak wordt beweerd.

6. De analyse

Naast het publiceren van kerncijfers in de vorm van tabellen, houdt het CBS zich ook bezig met de analyse van de databestanden. De bedoeling hiervan is achterliggende relaties bloot te leggen, het inzicht in het materiaal te verdiepen en op die wijze tevens te bewerkstelligen dat eventuele vervolgonderzoekingen qua vraagstelling en opbouw steeds beter worden. Voor een ieder die denkt dat dit loze kreten zijn, benadruk ik dat dat niet het geval is: geen meten zonder weten!

De analyse van CBS gegevens wijkt enigszins af van wat er in de statistiekboekjes staat: ten eerste gaat het bij ons vaak over duizenden records

in plaats van enkele tientallen, en ten tweede wordt de dominante positie, die continue (niet-categorale) gegevens innemen in de leerboeken en aan de universiteiten, niet weerspiegeld in het CBS materiaal. Bij persoons- en gezinsenquêtes zijn namelijk om meettechnische redenen bijna alle vragen categoraal, ook zelfs die welke in principe continue verschijnselen betreffen (denk aan leeftijds- en inkomensgroepen). Daarnaast is een belangrijk en afwijkend kenmerk dat ieder record van één (of meer) ophooggewichten is voorzien. Tenslotte is de analyse op het CBS vaak meer beschrijvend (exploratief) dan hypothese-toetsend (inferentieel), zoals bij de wiskundige statistiek. Trouwens, met onze aantallen gegevens is ieder verschil significant, zodat hypothese-toetsen niet echt zinvol is.

Een erg eenvoudige analysemethode op categorale gegevens is standaardisering van tabellen. Beschouw bijv. de volgende denkbeeldige EBB tabel:

Werkloosheid als percentage van de beroepsbevolking naar Regio en Leeftijd (fantasie)

	1-19	20-44	45-99	Gemiddeld
Friesland	4	9	14	11
Rest Ned.	5	10	15	10

Populatie (×1000) naar Regio en Leeftijd (fantasie)

	1-19	20-44	45-99	Totaal
Friesland	100	200	400	700
Rest Ned.	4000	4000	4000	12000

Op het eerste oog lijkt de werkloosheid in Friesland het hoogst, doch het is de vraag of dat betekent dat je in Friesland moeilijk een baan vindt. Als we namelijk tevens naar de leeftijdsopbouw kijken, dan zien we dat in Friesland veel ouderen wonen, en dat onder ouderen de werkloosheid het hoogst is. Als we de tabel standaardiseren naar de Nederlandse leeftijdsopbouw, blijkt de werkloosheid lager dan in de rest van Nederland te zijn. (Over standaardisatie, zie ook Israëls en de Ree [10].)

Categorale gegevens kunnen nominaal of ordinaal zijn. In beide gevallen is het vaak zinvol de categorieën zo te schalen dat met een categorale variabele gewerkt kan worden alsof het een continue is, zij het met een beperkt aantal waarden. De schaling gebeurt bijna altijd door de schaalwaarden zo te kiezen dat één of andere canonieke correlatie wordt gemaximeerd. Technieken als correspondentie-analyse en HOMALS zijn dan ook gebaseerd op eigenwaarden problemen toegepast op tabellen. Wanneer er sprake is van te verklaren categorale variabelen enerzijds en verklarende variabelen anderzijds, wordt gebruik gemaakt van ANOTA (Analysis of Tables, zie Keller e.a. [12]), een soort OLS op dummie-variabelen. Naast deze lineaire modellen staat

vanzelfsprekend het log-lineaire model, met als meest bekende schattingsmethode het 'iteratief proportioneel fitten' (IPF). Zoals we hierboven reeds zagen, is het probleem bij IPF dat bij veel variabelen de hoogdimensionale tabel niet meer in het geheugen past. Iets minder ambitieus (en toch familie) is het logit/probit model, waar een afhankelijke categorale variabele verklaard wordt uit continue en categorale (=dummie) variabelen, gelukkig meestal met maar weinig interacties. Een probleem bij deze multiplicatieve methoden is vaak de uitleg van de coëfficiënten aan leken, inclusief de valkuil van de normalisatie (laatste dummie weglaten), een reden voor ons om soms toch ANOTA te prefereren, ondanks wat theoretische bezwaren. Theoretische bezwaren zijn er natuurlijk ook bij de clusteranalyse, en ik geloof dan ook dat ik U gerust kan stellen: clusteranalyse wordt niet echt vaak op het CBS toegepast. Theoretische problemen zijn er ook bij het gebruik van ophooggewichten bij multivariate methoden (vooral als het gaat om de berekening van de standaardfouten) doch daar zal ik hier niet verder op in gaan.

Tot slot wil ik nog een belangrijk aspect van de analyse op het CBS noemen: grafische methoden. De microcomputer heeft het mogelijk gemaakt om achter je bureau de meest opwindende plaatjes te voorschijn te toveren. In navolging van Tukey, Tufte, Chambers en Cleveland is ook deze cultuur op het CBS sterk in ontwikkeling. Dat hier ook qua algoritmes geld te verdienen is, wordt duidelijk als men zich verdiept in methodes als 'cubic splines' en 'LOWESS', zoals onder andere geïmplementeerd in het CBS pakketje 'P-Wolk' (zie Bethlehem [4]), dat evenals vele andere juweeltjes van de hand van Jelke Bethlehem het aanzien meer dan waard is.

7. Slot

In het bovenstaande is in vogelvlucht stil gestaan bij enkele informatica aspecten in de statistiek. Daarbij lag de nadruk op de informatica bij de beschrijvende statistiek, zoals die op het Centraal Bureau voor de Statistiek wordt bedreven. De vraag 'wat is statistische informatica?' heb ik niet expliciet beantwoord, doch hopelijk weet U nu wel hoe leuk het is.

Referenties

1. M.E.J. Bemelmans-Spork, D. Sikkel, H.M. van Sintmaartensdijk, (1984). *Verslag van het experiment 'Prijswaarneming met draagbare computers',* Intern CBS-rapport, Centraal Bureau voor de Statistiek, Voorburg.
2. M.E.J. Bemelmans-Spork, W.J. Keller, D. Sikkel (1985). Use of handheld micro computers at the Netherlands Central Bureau of Statistics. *Proceedings of the Bureau of the Census First Annual Research Conference,* U.S. Department of Commerce, 47-52.
3. J.G. Bethlehem, W.J. Keller (1983). A generalized weighting procedure based on linear models. *Proceedings of the Section on Survey Research Methods of the American Statistical Association,* 70-75.
4. J.G. Bethlehem (1987). *Het spreidingsdiagram opnieuw bekeken,* Intern CBS-rapport, Centraal Bureau voor de Statistiek, Voorburg.
5. J.G. Bethlehem (1987). The data editing research project of the

Netherlands Central Bureau of Statistics. *Proceedings of the Third Annual Research Conference of the Bureau of the Census,* U.S. Department of Commerce, 194-203.

6. J.G. BETHLEHEM, D. DENTENEER, A.J. HUNDEPOOL, M.H. SCHUERHOFF (1989). *BLAISE 2.0 Een eerste kennismaking,* Intern CBS-rapport, Centraal bureau voor de Statistiek, Voorburg.
7. J.G. BETHLEHEM (1989). *The BLAISE System for Computer Assisted Collection and Editing of Survey Data,* Intern CBS-rapport, Centraal Bureau voor de Statistiek, Voorburg.
8. J.G. BETHLEHEM, W.J. KELLER (1987). A general method for weighting sample surveys. Te verschijnen in *Journal of Official Statistics* (Sweden).
9. D. DENTENEER, A. VERBEEK (1986). A fast algorithm for iterative proportional fitting in log-linear models. *Computational Statistics & Data Analysis,* 251-264.
10. A.Z. ISRAËL, S.J.M. DE REE (1982). *Standaardisatietechnieken,* Intern CBS-rapport, Centraal Bureau voor de Statistiek, Voorburg.
11. H. KELDERMAN (1987). *Quasi-loglinear Models for Test and Item Analysis,* Proefschrift, Technische Universiteit Twente.
12. W.J. KELLER, A. VERBEEK, J.G. BETHLEHEM (1985). *ANOTA: Analysis of Tables,* Intern CBS-rapport, Centraal Bureau voor de Statistiek, Voorburg.
13. R.L. KRUSE (1984). *Data Structures and Program Designs,* Prentice hall, New Jersey.

The Fast Fourier Transform Algorithm in Applied Probability Theory

Rudolf Grübel
Department of Mathematics
Imperial College, London

1. Introduction

Transforms in various guises have always played an important role in applied probability. Students meet them as probability generating functions, Laplace transforms, moment generating functions or characteristic functions and learn that such transforms can be used to identify distributions, to obtain moments, to obtain the distribution of the sum of independent random variables and to find the limit in distribution of a sequence of random variables. Once the Uniqueness Theorem and the Continuity Theorem for characteristic functions have been established, for example, it is difficult to beat the simplicity of the characteristic functions proof of the Central Limit Theorem. Also, transforms often provide an easy access to quantities of interest in a stochastic model: the extinction probability of a branching process is a classical example.

The main intention of this lecture is to show that another point can be added to the above list: transform methods can often be used to *calculate* quantities of interest. With the Fast Fourier Transform (FFT) algorithm this can be done most efficiently so that, when they apply, these methods will usually give a precision far beyond that of the ubiquitous simulation procedures.

Applied probability is, of course, an enormously varied subject and we cannot seriously hope to cover any substantial portion of it. We will concentrate on classical stochastic models as discussed, for example, in the two volumes *An Introduction to Probability Theory and Its Applications* by W. Feller, and we will deal with two main structural elements in this field - renewal theory, and the theory of Wiener-Hopf factors of random walks. These provide the key to a variety of models such as simple repair systems, storage processes, queueing models and many others.

The lecture is organized as follows: after a brief section on transforms, the

method is demonstrated in detail in the renewal theoretic context in Section 3. Section 4 deals with Wiener-Hopf factors and explains the approach via harmonic renewal measures, leading to a reformulation of the Spitzer-Baxter identities. An application to the stationary waiting time of the G/G/1 queueing model is indicated in Section 5. A final section collects some concluding remarks and also comments on the literature.

2. Transforms

Let μ be a probability distribution, or, more generally, a finite complex-valued measure on the Borel subsets $\mathfrak{B}$ of the real line. Its *Fourier transform* is the function $\hat{\mu}:\mathbf{R}\to\mathbf{C}$ given by

$$\hat{\mu}(\theta) = \int e^{i\theta x}\mu(dx) \tag{1}$$

where the integral is the usual Lebesgue integral. Such distributions can be approximated with respect to weak convergence by a discrete measure which is concentrated on a finite number of equally spaced points; after a simple shift and rescaling procedure we may even assume the support to be the set $\{0,\ldots,N-1\}$. For measures of this type (1) reduces to

$$\hat{\mu}(\theta) = \sum_{k=0}^{N-1} a_k e^{i\theta k} \quad \text{where} \quad a_k = \mu(\{k\}),\ k=0,\ldots,N-1 \tag{2}$$

and such a *discrete Fourier transform* is completely specified by its values at the *Fourier frequencies*

$$\theta_k = 2\pi k/N, \quad k=0,\ldots,N-1. \tag{3}$$

From a more abstract point of view we are exchanging locally compact Abelian groups; the above statement reflects the fact that the dual group of $G=\mathbf{Z}/N\mathbf{Z}$ is (isomorphic to) G. Weak convergence implies pointwise convergence of the associated Fourier transforms; as shift and rescaling are easy to handle the approximation of integrals as appearing in (1) essentially leads to sums of the form (2).

On first sight it seems that calculating (2) at all the Fourier frequencies given in (3) requires about N^2 multiplications - the FFT algorithm, however, can do this in about $N\log N$ multiplications, if N is a power of 2. We will not describe the algorithm but refer to Brigham [3] who also gives a concise and most interesting account of the history of the algorithm; Press et al. [14] include **FORTRAN** and **Pascal** programs.

With the notation as in (2) and (3) it is easily seen that

$$a_l = N^{-1}\sum_{k=0}^{N-1} \hat{\mu}(\theta_k)e^{-i\theta_l k} \tag{4}$$

and, consequently, a simple modification of the algorithm can be used to reobtain the sequence $a_0,\ldots,a_{N-1}$ from the values of its Fourier transform at the Fourier frequencies.

We close this section with a general discussion of the types of errors

involved, more detailed information will be given along with the discussion of specific models. First, on the way from the measure to its transform, we will have a *truncation error* as we replace μ by its restriction to some interval $(-a,b)$. Evidently, this error can be made arbitrarily small by choosing a and b large enough in absolute value. Further, lumping the mass of an interval into an atom at a certain point will introduce a *discretization error* which will be small if the length of these intervals is chosen small enough. Both these requirements can lead to large values of N which underlines the importance of a suitable algorithm. Also, using (4) to go back from a transform to the underlying measure, induces an *aliasing error*: if (4) is used on a Fourier series

$$\hat{\mu}(\theta) = \sum_{-\infty}^{\infty} a_k e^{i\theta k} \tag{5}$$

we will not obtain the coefficients a_k but the values

$$\tilde{a}_k = \sum_{-\infty<l<\infty} a_{k+lN}, \quad k=0,\ldots,N-1 \tag{6}$$

instead. These are taken as approximations for a_k if the values with negative indices are negligible, otherwise we use $\tilde{a}_{N-k}$ as an approximate value for a_{-k}, $k=1,\ldots,N/2$. Again, for summable sequences the aliasing error will be small if N is chosen large enough. Finally, the values so obtained will be regarded as approximations to the mass which the distribution or measure of interest assigns to some interval. This means that an error of discretization type occurs again; its magnitude will depend on the stochastic system in question but, if this system is in a 'stable' region, will automatically be of the same order as the initial discretization error.

3. Renewal theory

The standard example of renewal theory involves 'an electric bulb, fuse or other piece of equipment with a finite life span. As soon as the piece fails, it is replaced by a new piece of the like kind, which in due time is replaced by a third piece, and so on' (Feller [6], p.311). Formally, we have a sequence $(X_i)_{i\in\mathbb{N}}$ of independent and identically distributed random variables representing the successive lifetimes and

$$N(t) = 1+\sup\{n\in\mathbb{N}_0: \sum_{i=1}^{n} X_i \leqslant t\},$$

is the number of renewals up to time t.

The importance of this model goes significantly beyond this simple example: a similar structure often underlies more complex stochastic models and can provide the key to the understanding of, for example, the asymptotic properties of such constructions - in fact, many books on applied probability contain a chapter dealing exclusively with this setup. The reason for this is that stochastic models often have 'renewal points' where the process 'starts from scratch', and the occurrence of such points forms a pattern of the type described above. For a more detailed discussion, including a range of examples, see Feller [6], Chapter XI.

Of particular interest is the *renewal measure* which associates with a Borel subset of the real line the expected number of partial sums with values in this set; $U(t)=EN(t)$, the expected number of renewals in the interval $[0,t]$, is called the *renewal function.* Our aim in this section is to show how the renewal measure can be obtained numerically - subject to the errors described in the previous section - from the *lifetime distribution,* i.e. the distribution μ of X_1.

As a first step, we discretize μ and obtain a sequence

$$p_0 = \mu([0,h/2)), \quad p_k=\mu([(k-1/2)h,(k+1/2)h)) \text{ for } k\in\mathbb{N}$$

which we regard as a distribution on $\mathbb{N}_0$. The renewal measure associated with this sequence is evidently concentrated on $\mathbb{N}_0$ again and, therefore, characterized by the *renewal sequence* $u_n=u_n^{(h)}=U^{(h)}(n)-U^{(h)}(n-)$. Summing these values we obtain $U^{(h)}$ and, for every continuity point of the latter, it holds that $U^{(h)}([t/h])$ tends to $U(t)$ as $h\to 0$. This shows that both discretization errors involved will be small if h is chosen small enough.

In the second step we analyze the dependence of a discrete renewal sequence u on the underlying lifetime probability mass function p. The Discrete Renewal Theorem tells us that u_n tends to a positive limit if p has finite mean, i.e. if $\Sigma np_n<\infty$, hence its Fourier transform in the sense of Section 2 does not exist and we have to 'renormalize' first. To achieve this we employ the *renewal equation*

$$u_n = d_n + \sum_{m=0}^{n} u_m p_{n-m}$$

where $d_n=1$ for $n=0$ and 0 otherwise. We assume that p has finite mean and is *aperiodic,* i.e.

$$\gcd\{n\in\mathbb{N}:p_n>0\} = 1,$$

and we define two new sequences r and q by

$$r_n = \sum_{m>n} p_m \text{ for } n\in\mathbb{N}_0;\ q_0=1,\ q_n=u_n-u_{n-1} \text{ for } n\in\mathbb{N}.$$

It then follows from the renewal equation that r and q are convolution inverse, i.e.

$$\sum_{m=0}^{n} r_m q_{n-m}=d_n \text{ for all } n\in\mathbb{N}_0.$$

Further, aperiodicity yields

$$\hat{r}(\theta) = \sum_{n=0}^{\infty} r_n e^{in\theta}\neq 0 \text{ for all } \theta\in\mathbb{R}$$

and Wiener's Lemma (see, e.g., Rudin [15], p.266) implies that q is a summable sequence with

$$\hat{q}(\theta) = 1/\hat{r}(\theta) \text{ for all } \theta\in\mathbb{R}.$$

This argument is the core of one of the proofs given by Erdös, Feller and Pollard [5] for the Discrete Renewal Theorem and it is quite interesting to see that it is also important in the present numerical context!

We now proceed as follows: choose c large enough for $\mu((c, \infty))$ to be negligible (see also Comment 1 on p.6). Choose N as a power of 2 and put $h = c/N$. Define $r_0, \ldots, r_{N-1}$ by $r_n = \mu([(n+1/2)h, \infty))$ and use FFT to evaluate (a truncated version of) $\hat{r}(2\pi n/N)$ for $n = 0, \ldots, N-1$. (Note that we used a shortcut which avoids the introduction of p and the subsequent summation.) Calculate $\hat{q}(2\pi n/N) = 1/\hat{r}(2\pi n/N)$ for $n = 0, \ldots, N-1$ and apply the inverse FFT to obtain, with additional aliasing errors, the values $q_0, \ldots, q_{N-1}$ (in this simple case one could even skip the unscrambling parts of the FFT; see Press et al. [14], p. 395). From these, the values $u_0, \ldots, u_{N-1}$ are obtained by summation and a second summation now yields our approximate values for $U(h/2)$, $U(3h/2)$, $U(5h/2), \ldots, U((2N-1)h/2)$, i.e. the function U on $(0,c)$ on a grid of width h.

That all this is, essentially, a very simple affair, is perhaps seen best if we condense the whole procedure into a little **Pascal** program:

```
PROGRAM RenFFT;
CONST                                    (see Comment 1 below)
  c = 6.0;
  n2 = 1024;
  datfile = 'RenFFT.dat';
TYPE
  gldarray = ARRAY [1..n2] OF real;
VAR
  a,b,z,t,h :real;
  i,j,n,fact,coeff :integer;
  x :gldarray;
  file1 :text;
FUNCTION tail(y : real): real;
BEGIN
  IF y<1 THEN tail:=1-y ELSE tail:=0
END;
PROCEDURE fft(VAR data: gldarray; nn, isign: integer);
                                         (see Comment 2 below)
```

```
BEGIN
  n := n2 DIV 2;
  h := c/n;
  FOR i:=1 TO n DO
  BEGIN
    x[2*i] := 0;
    x[2*i-1] := tail((i-0.5)*h)
  END;
  fft(x,n,1);
  FOR i:=1 TO n DO
  BEGIN
    z := sqr(x[2*i-1])+sqr(x[2*i]);
    x[2*i-1] := x[2*i-1]/z;
    x[2*i]   := -x[2*i]/z
  END;
  fft(x,n,-1);
  FOR i:=1 TO n DO x[2*i-1]:=x[2*i-1]/n;
  FOR i:=2 TO n DO x[2*i-1]:=x[2*i-1]+x[2*i-3];
  FOR i:=2 TO n DO x[2*i-1]:=x[2*i-1]+x[2*i-3];
  FOR i:=1 TO n DO                          (see Comment 3 below)
  BEGIN
    t := (i-0.5)*h;
    a := exp(t);
    fact := 1;
  coeff := 1;
  FOR j:=1 TO trunc(t) DO
  BEGIN
   fact := j*fact;
   coeff := -coeff;
   a := a + coeff*exp(j*ln(t-j)+(t-j))/fact
  END;
  x[2*i] := a;
  END;
  assign(file1,datfile);                    (see Comment 4 below)
  rewrite(file1);
  FOR i:=1 TO n DO
  BEGIN
  t := (i-0.5)*h;
  writeln(file1,t:5:3,x[2*i-1]:10:8,x[2*i]:10:8);
  END;
  close(file1)
END.
```

COMMENTS:

1. The constant c represents the range on which the renewal function will be considered. If c is not large enough a serious aliasing error might arise. However, it is a straightforward matter to change this value and to observe the corresponding changes in $U(t)$ for a fixed t. This should be accompanied by a corresponding change in n2, which must be a power of 2 and corresponds to $2N$ in the discussion preceeding the program, so that h remains fixed - the error might otherwise be masked by a change in the discretization error.
2. A subroutine which calculates the FFT is not given. It has to do the following: the complex numbers $z_0, \ldots, z_{N-1}$ enter in the form $z_k = \text{data}[2k+1] + i\,\text{data}[2k+2]$; on return, we must have the values $\hat{z}(\theta_k) = \text{data}[2k+1] + i\,\text{data}[2k+2]$ where

$$\hat{z}(\theta) = \sum_{l=0}^{N-1} z_l \exp(\pm l\theta), \quad \theta_k = 2\pi ik/N$$

   with $\pm$ specified by isign $= \pm 1$. Note that we have to divide by N in the main program after the inverse FFT has been carried out. Structure and names of variables are such that the procedure **FOUR1** in Press et al. [14] can be used without any changes.
3. The above program calculates (our approximation for) the renewal function for uniformly distributed lifetimes. In this special case the renewal function is known explicitly (see Feller [6], p.385),

$$U(t) = \sum_{k=0}^{n} (-1)^k e^{t-k} \frac{(t-k)^k}{k!} \quad \text{for} \quad n \leqslant t \leqslant n+1.$$

   These values are obtained on a grid of width h and put into the array elements with even index (sofar these contain the imaginary parts of the sequence q, a fact which can be used to check the choice of constants).
4. File-handling is not standardized in **Pascal**, so some of the associated commands are specific for the implementation used here.

No attempt has been made to optimize the code. After execution of the program the file **RenFFT.dat** contains the triplets $(t, U_1(t), U_2(t))$ for $t = h/2, 3h/2, \ldots, (2N-1)h/2$. Here U_1 is the true renewal function and U_2 the approximation found with the FFT algorithm. With the above parameters the maximum error was found to be 0.000184. In view of the following points this could be regarded as a remarkably good result: firstly, we have to sum twice to get from q to U, which means that any errors for small time values will propagate through the whole array. Secondly, no sophisticated numerical analysis tool was used (application of Richardson's deferred approach to the limit is currently being investigated). Finally, the value of this error should be seen in connection with the fact that normally simulations are used in such a context: simulation errors are decreasing at rate $n^{-1/2}$ only, which means that an untolerably large number n of repetitions is necessary to obtain a comparable result.

4. Wiener-Hopf factors

Let again $(X_i)_{i\in\mathbb{N}}$ be a sequence of independent and identically distributed random variables; we now assume that these take positive and negative values. The sequence $(S_n)_{n\in\mathbb{N}_0}$ of partial sums,

$$S_0\equiv 0, \quad S_n = \sum_{i=1}^{n} X_i \quad \text{for} \quad n\in\mathbb{N},$$

is called a *random walk,* the distribution μ of the X_i's is the *step distribution* of the random walk.

Put $\tau=\inf\{n\in\mathbb{N}:S_n>0\}$, $\tau=\infty$ if $\{\cdots\}=\varnothing$, and on $\{\tau<\infty\}$ let $Y=S_\tau$. These are the *first ascending ladder epoch* and *height* respectively; we call their joint distribution P_+ the *right Wiener-Hopf factor* associated with μ. These quantities are of some interest in a variety of stochastic models, a queueing theory example is given in the next section.

Decisive for our approach is the following concept: for each measure P on $\mathbb{N}\times\mathbb{R}$ with total mass $\leqslant 1$ (a *sub-probability*) we define the associated *harmonic renewal measure* $\nu(P)$ by

$$\nu(P) = \sum_{n=1}^{\infty} \frac{1}{n} P^{*n}$$

where * denotes convolution. Some care is needed with the transforms of such possibly infinite (if $P(\mathbb{N}\times\mathbb{R})=1$) measures. In the present case we have

$$\nu(P)\in\mathcal{Q} := \{Q: Q \text{ measure on } \mathbb{N}\times\mathbb{R}, \sup_{n\in\mathbb{N}} Q(\{n\}\times\mathbb{R})\leqslant 1\} \tag{7}$$

which can be seen as follows:

$$\begin{aligned}\nu(P)(\{n\}\times\mathbb{R}) &= \sum_{k=1}^{\infty} \frac{1}{k} P^{*k}(\{n\}\times\mathbb{R}) \\ &\leqslant \sum_{k=1}^{\infty} (p^{*k})_n \leqslant u_n \leqslant 1\end{aligned}$$

where $p_n=P(\{n\}\times\mathbb{R})$ denotes the first marginal of P and u the associated renewal sequence. Measures in $\mathcal{Q}$ are conveniently characterized (i.e., a corresponding uniqueness theorem is easily proved) by their transforms $\hat{Q}$ which are defined as follows,

$$\begin{aligned}\hat{Q}(z,\theta) &= \int z^n e^{i\theta x} Q(dn,dx) \\ &= \sum_{n=1}^{\infty} z^n \int e^{i\theta x} Q(\{n\}\times dx), \quad |z|<1,\ \theta\in\mathbb{R}.\end{aligned}$$

Also, we have the following important relation,

$$\nu(P)\hat{}(z,\theta) = -\log(1-\hat{P}(z,\theta)) \quad \text{for all } |z|<1,\ \theta\in\mathbb{R}. \tag{8}$$

To prove (8) keep $|z|<1$ fixed. Then $\theta \to \hat{Q}(z,\theta)$ is the characteristic function of the finite measure $A \to \Sigma z^n Q(\{n\}\times A)$ and the condition in (7) permits the necessary interchanges of sum and integral.

Equation (8) implies, in particular, that P can be recovered from $\nu(P)$: if $\nu(P_1)=\nu(P_2)$ for two sub-probabilities on $\mathbb{N}\times\mathbb{R}$, then $P_1=P_2$.

Our intention in introducing this concept is to provide a simple link between μ and P_+, and, indeed, their harmonic renewal measures are related in the simplest possible way: consider the 'time-space' steps $(1,X_i)$ with distribution $P=\delta_1\otimes\mu$, where δ_x denotes the unit mass in $x\in\mathbb{R}$ and $\otimes$ denotes product measure. As P_+ is concentrated on $\mathbb{N}\times(0,\infty)$ and as this set is closed with respect to addition, $\nu(P_+)$ is also concentrated on $\mathbb{N}\times(0,\infty)$. Apart from this obvious condition on the support, $\nu(P)$ and $\nu(P_+)$ coincide, i.e., $\nu(P_+)$ is the restriction of $\nu(P)$ to this set in the following sense:

$$\nu(P_+)(A\times B) = \nu(P)(A\times(B\cap(0,\infty))) \quad \text{for all } A\subset\mathbb{N},\ B\in\mathcal{B}.$$

Given the previous definitions and properties, especially (8), this is easily seen to be, essentially, a reformulation of the famous Spitzer-Baxter identities (see e.g. Feller [6], p. 605),

$$1-Ez^\tau e^{i\theta Y} = \exp\Big(-\sum_{n=1}^{\infty}\frac{1}{n}z^n\int_{(0,\infty)} e^{i\theta x}\mu^{*n}(dx)\Big).$$

We will mainly be interested in the distribution μ_+ of Y. Taking $A=\mathbb{N}$ in (8) we obtain

$$\nu(\mu_+)(B) = \nu(\mu)(B\cap(0,\infty)) \quad \text{for all } B\in\mathcal{B}. \tag{9}$$

If we wish to use (9) for numerical purposes some of the same problems as in the renewal theory context of Section 3 arise; in particular, an appropriate normalization is needed so that we can work with finite (signed) measures. Again, we discretize first. The relevant result is then given in Grübel [10]: let $p=(p_n)_{n\in\mathbb{Z}}$ be an aperiodic and non-defective distribution and let $h=\Sigma_1^\infty \delta_n/n$ be the harmonic renewal sequence associated with δ_1. Then $\Sigma|n|p_n<\infty$, $\Sigma np_n>0$ (i.e., first moment exists and is positive) is equivalent to $\nu(p)-h$ being summable, and both imply

$$\widehat{(\nu(p)-h)}(\theta) = \log\widehat{\Sigma p}(\theta) \tag{10}$$

where

$$(\Sigma p)_n = \begin{cases} \sum_{m>n} p_m, & \text{if } n\geqslant 0; \\ -\sum_{m\leqslant n} p_m, & \text{if } n<0. \end{cases}$$

The transition $n\to -n$ easily gives the corresponding statement for step distributions with negative mean, as required for the queueing model in the next section.

5. The stationary waiting time in a G/G/1 queue

We give a brief description of the model first: customers arrive at times Z_1, Z_2, Z_3,... and require service times S_1, S_2, S_3,...; there is only one server and the queueing discipline is first in/first out. In the G/G/1 model (GI/G/1 with some authors) the interarrival times T_1, T_2, T_3,... where $T_1 = Z_1$, $T_k = Z_k - Z_{k-1}$ for $k > 1$, are assumed to be independent and identically distributed with distribution μ_T, also the service times are independent and identically distributed with distribution μ_S and service and arrival times are independent of each other.

Let W_n denote the waiting time for the n-th customer, not including his service time. Assume that the mean interarrival time m_T and the mean service time m_S are finite and that $m_T > m_S$. This prevents explosion and, as a consequence, W_n converges in distribution to W, the stationary waiting time.

It is well known (see, e.g., Asmussen [1], Chapter VIII) that W has the same distribution as the maximum of the random walk with steps $X_i = S_i - T_i$. Moreover, this maximum is the sum of all ascending ladder heights of the random walk. These ladder heights are independent; all have the same distribution μ_+ which is defective because of $m_T > m_S$. This implies the following relation,

$$Ee^{i\theta W} = \frac{1-\hat{\mu}_+(0)}{1-\hat{\mu}_+(\theta)}. \tag{11}$$

The overall structure of the dependence of the stationary waiting time distribution μ_W on the service time distribution μ_S and the customer interarrival time distribution μ_T can now be summarized in the following diagram:

$$\mu_S,\ \mu_Y \to \mu_X \to \mu_+ \to \mu_W.$$

It is easy to handle the first step on the transform side as $\hat{\mu}_X(\theta) = \hat{\mu}_S(\theta)\hat{\mu}_T(-\theta)$ (the transition from μ to $\Sigma\mu$ only causes a minor complication). For the second step we apply the theory developed in Section 4: use (10) to obtain the transform of the (normalized) harmonic renewal measure associated with μ_X, transform back, use (9) to obtain the harmonic renewal measure of μ_+, transform, and use (10) again to obtain $\hat{\mu}_+$. Staying on the transform side we use (11) for the last step and transform back to obtain μ_W. Clearly, the FFT algorithm is employed for switching back and forth between measures and their transforms: essentially, we proceed in the same way as detailed in Section 3 for the easier renewal theory context.

Details, including a computer program, are given in Grübel [9]. In the special case of exponential distributions for service and interarrival times, i.e., for M/M/1 queues, the distribution of W is known explicitly: it is a mixture of an atom of size $1 - m_S/m_T$ in 0 and an exponential distribution with parameter $m_T - m_S$. The table below gives the exact values and the approximations obtained by the above procedure in a special case.

0	0.50495271	0.50497508	1	0.00980174	0.00980215
2	0.00960767	0.00960805	3	0.00941744	0.00941780
4	0.00923098	0.00923132	5	0.00904821	0.00904852
10	0.00818724	0.00818744	50	0.00367906	0.00367886
100	0.00135359	0.00135338	300	0.00002480	0.00002479

TABLE 1
Result of the algorithm and theoretical value of
$P((i-0.5)0.02<W<(i+0.5)0.02)$ for a sample of i-values
(W is the stationary waiting time for an M(1)/M(2)/1 queue)

6. CONCLUDING REMARKS

Somewhat surprisingly, the literature on applications of the FFT in applied probability seems to be sparse. Bohman [2] considers the problem of obtaining a distribution function from the corresponding characteristic function; this interesting paper also employs the Richardson extrapolation. Tijms [16] briefly mentions FFT as a means to re-obtain probabilities from generating functions.

In Grübel [11] the method described in Section 3 was used to obtain numerically renewal densities of mixtures of distributions in order to illustrate certain limit theorems; an explicit description is given here for the first time. It has since then been applied to other stochastic models: Pang [13] considered a simple repair model and compared various procedures, simulation, simulation plus some standard variance reduction techniques, the approach of Grübel [11] and, finally, the FFT method. From a numerical point of view the latter was found to be far superior to all the other methods. We also mention Brown et al. [4] for an interesting variance reduction technique in this context.

The approach to ladder variates via harmonic renewal measures has been used in Grübel [8] for distance estimates, in Grübel [10] for tail probability expansions and in Grübel and Pitts [12] to investigate robustness properties; see also Greenwood et al. [7]. It is a simple and useful technique. The explicit description in Section 4 of the link with the Spitzer-Baxter identities appears here for the first time.

In statistics, the FFT algorithm is routinely applied for density estimation and in time series analysis. In both cases the sequences of interest can be taken to be absolutely convergent. In contrast, in our applications given above, we deal with sequences which are not summable: a suitable normalization has to be applied - and, of course, to be found first! This, perhaps, explains why the FFT algorithm doesn't seem to have been applied in these areas before. This difficulty also lends some charm to the method as it involves a non-trivial amount of mathematics.

REFERENCES

1. S. ASMUSSEN (1987). *Applied Probability and Queues,* Wiley, New York.
2. H. BOHMAN (1977). Numerical Fourier inversion. *TIMS Studies in the Management Sciences 7,* 3-11.

3. E. ORAN BRIGHAM (1974). *The Fast Fourier Transform,* Prentice-Hall, Englewood Cliffs., N.J.
4. M. BROWN, H. SOLOMON, M.A. STEPHENS (1981). Monte Carlo simulation of the renewal function. *J. Appl. Prob. 18,* 426-434.
5. P. ERDÖS, W. FELLER, H. POLLARD (1949). A property of power series with positive coefficients. *Bull. Am. Math. Soc. 55,* 201-204.
6. W. FELLER. *An Introduction to Probability Theory and Its Applications,* Vol. I, 3rd ed. (1968) and Vol. II, 2nd ed. (1971) Wiley, New York.
7. P. GREENWOOD, E. OMEY, J.L. TEUGELS (1982). Harmonic renewal measures. *Z. Wahrsch. verw. Gebiete 59,* 391-409.
8. R. GRÜBEL (1986). A note on the distance of ladder height distributions. *Stoch. Proc. Appl. 23,* 339-341.
9. R. GRÜBEL (1987). G/G/1 via FFT. Submitted for publication.
10. R. GRÜBEL (1988) Harmonic renewal sequences and the first positive sum. *J. London Math. Soc. 38,* 179-192.
11. R. GRÜBEL (1989). Stochastic models as functionals: some remarks on the renewal case. *J. Appl. Prob.,* to appear.
12. R. GRÜBEL, S.M. PITTS (1989). Perturbation and rearrangement aspects of renewal and harmonic renewal sequences. *J. London Math. Soc.,* to appear.
13. S.K.W. PANG (1987). *Alternating Renewal Processes,* MSc Thesis, Imperial College, London.
14. W.H. PRESS, B.P. FLANNERY, S.A. TEUKOLSKY, W.T. VETTERLING (1987). *Numerical Recipes,* Cambridge University Press, Cambridge.
15. W. RUDIN (1974). *Functional Analysis,* Tata McGraw-Hill, New Delhi.
16. H.C. TIJMS (1986). *Stochastic Modelling and Analysis* Wiley, New York.

Didactiek en de computer

Dynamische Simulatie in de Klas

Piet van Blokland, Douwe Kok
Faculteit Wiskunde en Informatica
Vrije Universiteit Amsterdam

1. Inleiding

De computer kan op verschillende manieren gebruikt worden binnen het wiskunde-onderwijs. Met een grafisch pakket als VU-grafiek kan men van elke functie snel de grafiek onder ogen krijgen. De computer is dan een soort elektronisch schoolbord. Ook kan men de computer benutten als 'learning-machine': een vaardigheid als het vinden van een functie-voorschrift bij een gegeven grafiek, kan met behulp van de computer ingeoefend worden.

In deze lezing gaat het om weer een ander type gebruik: de computer als een instrument dat van pas komt bij het bouwen van modellen en bij het uittesten ervan.

Je hebt dan wel speciale software nodig. Binnen onze vakgroep is daarom het pakket VU-dynamo [1] ontwikkeld. Het biedt de gebruiker de mogelijkheid te experimenteren met zgn. 'dynamische modellen'.

De ontwikkeling en het gebruik van dit pakket sluit aan bij de belangrijkste trend in het denken over het wiskunde-onderwijs op het voortgezet onderwijs. Mathematiseren en modelvorming worden steeds belangrijker geacht. Ook wordt veelvuldig de nadruk gelegd op de toepassingen van de wiskunde. HEWET op het VWO en HAWEX op de HAVO laten dat duidelijk zien.

Reden genoeg om iets te tonen van de mogelijkheden van VU-dynamo. Er komt nog iets bij: sinds een half jaar wordt er door ons in enkele 5VWO-klassen geëxperimenteerd met het onderwerp Dynamische Simulatie, waarbij VU-dynamo als software gebruikt wordt. Ons is gevraagd na te gaan of dit onderwerp een geschikt onderdeel zou kunnen vormen van het VWO-B programma.

Deze lezing heeft als opbouw:

- informatie over Dynamische Simulatie;
- informatie over VU-dynamo;

- enkele ervaringen in de klas;
- een 'echt' model: BOOMSTERFTE.

2. WAT IS DYNAMISCHE SIMULATIE ?

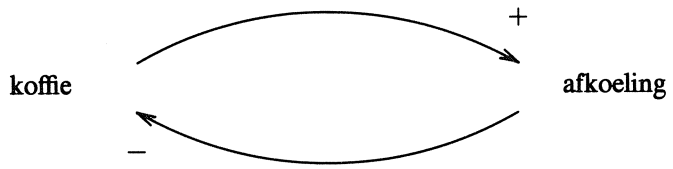
Dynamische simulatie is een simulatiemethode die de nadruk legt op de structuur en het gedrag van systemen die zijn samengesteld uit een of meerdere feedback-loops.

Een voorbeeld:

Hoe verloopt de afkoeling van een kopje koffie? Voor dit proces zijn onder andere van belang:

- de temperatuur van de koffie;
- de temperatuur van de kamer;
- de vorm van het kopje en het materiaal waarvan het is gemaakt.

Volgens de afkoelingswet (van Newton) geldt dat de koelsnelheid evenredig is met het verschil tussen de kamertemperatuur en die van de koffie. Hoe hoger de koffietemperatuur, hoe sneller verloopt de afkoeling. Maar aan de andere kant geldt: des te sneller de afkoeling, des te lager de koffietemperatuur. Deze situatie kan met onderstaand oorzaak-gevolg-diagram beschreven worden.



Essentieel in het proces is de terugkoppeling, de feedback.

De methode van Dynamische Simulaties is aan het einde van de jaren vijftig ontwikkeld aan het beroemde Massachusetts Institute of Technology, het MIT [2]. De eerste toepassingen lagen op het gebied van het management. De systeemdynamische aanpak van allerlei problemen is aan het eind van de zestiger jaren algemeen bekend geworden door toepassingen op macro-niveau, met name door het wereld-model ('Grenzen aan de groei') en door modellen van steden [3,4].

Dynamische simulatie bleek een krachtige techniek te zijn om het gedrag van grote complexe systemen - ondernemingen, industrie, steden, de ecologie van de aarde - te bestuderen. De onderzoeker kan zo de invloed van verschillende beheerspolitieken nagaan.

Enkele voorbeelden van dynamische simulaties uit de literatuur zijn:

- World2. De eerste geslaagde poging om een samenhangend model te maken van de wereldeconomie, bevolkingsgroei, milieu en voedsel vanuit systeemdynamisch perspectief.
- Economie. Er zijn veel vergeefse pogingen ondernomen de prijzen van grondstoffen als olie, suiker, jute, koffie te stabiliseren. Met behulp van dynamische modellen kan men inzicht krijgen waarom deze pogingen mislukten. Ook kan men het effect van andere pogingen simuleren.
- Heroïne. Een model over het verband tussen heroïne, misdaad en politie.

Het model maakt duidelijk dat het enige effect van een heroïne-vangst daarin bestaat, dat de prijs van heroïne toeneemt en dus ook het aantal misdaden die de junks plegen om het geld te verdienen waarmee de heroïne betaald moet worden.

- Kaibab. Een ecologisch model over herten, tijgers en voedsel. Door de natuurlijke vijand van de herten te doden, groeit de herten populatie explosief. Hierdoor ontstaat overbegrazing, met dramatische gevolgen voor de hertenstand. Op dit model komen we verderop nog terug.

3. Enkele opmerkingen over VU-dynamo

De taal DYNAMO is ontwikkeld door J.W. Forrestor; oorspronkelijk voor grote computersystemen. Aan Micro-dynamo, een afgeleide versie voor de PC, is dat nog wel te zien [5]. Met VU-dynamo is geprobeerd een leerling-vriendelijke versie van deze taal te maken, die tevens optimaal gebruik maakt van de grafische mogelijkheden van de PC.

Als voorbeeld kiezen we weer de afkoeling van een kopje koffie. Dat proces kan beschreven worden via:

$$\text{koffie\_nieuw} = \text{koffie\_oud} - \Delta t * \text{koelsnelheid}$$

$$\text{koelsnelheid} = \text{constante} * (\text{koffie\_oud} - \text{kamer})$$

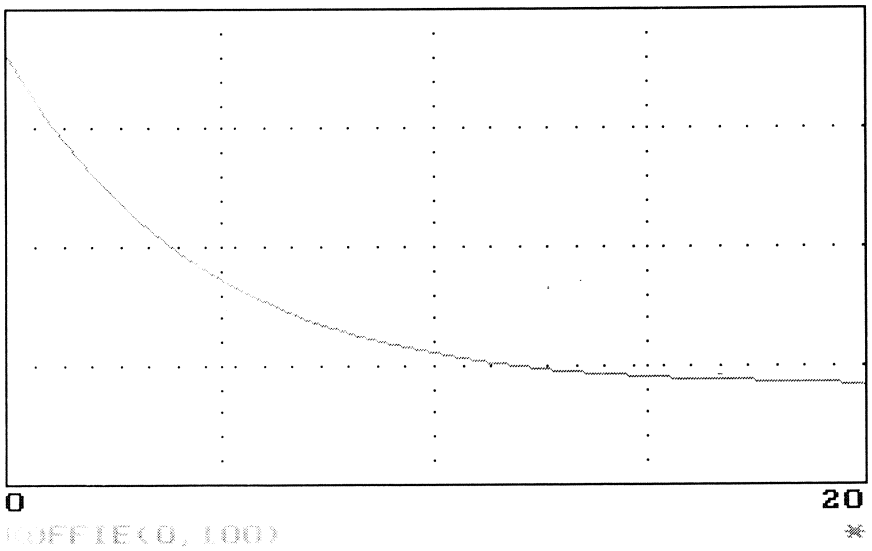
We kiezen als beginwaarden: koffie = 90 ,kamer = 20. Δt nemen we 1 (minuut) en de waarde voor de constante stellen we op 0.2. Dit levert de volgende tabel op

tijd	koffie	koelsnelheid
0	90	14
1	76	11.2
2	64.8	..

Een programma in VU-dynamo dat het afkoelingsproces beschrijft luidt:

```
note   KOFFIE
note   hoe verloopt de afkoeling van koffie?
note   een model gebaseerd op de afkoelingswet van Newton

l   koffie = koffie -Δt*koelsnelheid      ; koffie=koffietemperatuur
r   koelsnelheid = constante*(koffie -    ; kamer  = kamertemperatuur
    kamer)
                                          ; koelsnelheid is afkoeling
                                            van de koffie (in graden
                                            per minuut)
n   koffie = 90
n   kamer = 20
c   constante = 0.2
```



FIGUUR 1

spec Δt = 1/eindtijd=20/prtper=1
print koffie, koelsnelheid
plot koffie(0,100)

TIJD	KOFFIE	KOELSNEL
0.0000	90.0000	14.0000
1.0000	76.0000	11.2000
2.0000	64.8000	8.9600
3.0000	55.8400	7.1680
4.0000	48.6720	5.7344
5.0000	42.9376	4.5875
6.0000	38.3501	3.6700
7.0000	34.6801	2.9360
8.0000	31.7441	2.3488
9.0000	29.3952	1.8790
10.0000	27.5162	1.5032
11.0000	26.0130	1.2026
12.0000	24.8104	0.9621
13.0000	23.8483	0.7697
14.0000	23.0786	0.6157
15.0000	22.4629	0.4926
16.0000	21.9703	0.3941
17.0000	21.5763	0.3153
18.0000	21.2610	0.2522
19.0000	21.0088	0.2018
20.0000	20.8070	0.1614

Enkele opmerkingen over dit programma:

De letter L komt van LEVEL, Engels voor niveau. In deze regel wordt aangegeven hoe een hoeveelheid, in dit geval koffie, in de loop van de tijd verandert.

De letter R is afkomstig van RATE. Dit woord heeft in het Engels heel veel betekenissen. Het duidt op de mate van verandering, de snelheid, het tempo. In dit model is het de koelsnelheid die bepaalt hoe snel de temperatuur van de koffie verandert. De snelheid van de verandering dus. Koelsnelheid is in dit model een stroom-variabele.

N komt van iNitial. In een regel die met N begint, wordt altijd een beginwaarde opgegeven, in dit geval de beginwaarde van de koffietemperatuur.

C is de eerste letter van CONSTANT. Het gaat hier om een constante waarde, een waarde dus die tijdens het doorrekenen van het model niet verandert.

SPEC is een afkorting van SPECIFICATION. In deze regel wordt aangegeven in welke stappen het model wordt doorgerekend. Behalve de stapgrootte Δt, moet hier ook de eindtijd opgegeven worden. Daarnaast kan men ook de printperiode opgeven. Bij $\Delta t = 0.1$ en PRTPER = 1 worden dus elke tiende keer dat het model wordt doorgerekend, de gevraagde waarden afgedrukt.

Wiskundig bekeken gebeurt er dit: Men stelt eerst een aantal differentievergelijkingen op. Vervolgens worden die met de computer doorgerekend. Als men de stapgrootte Δt steeds kleiner neemt dan nadert de oplossing tot de oplossing van het continue probleem. Aldus kan men naderen tot de oplossing van de differentiaalvergelijking.

3. Dynamische simulatie in de klas

Als gevolg van het toenemende maatschappelijke belang van de informatica en de daardoor veroorzaakte introductie van de computer binnen het voortgezet onderwijs wordt de laatste jaren de roep om een examenvak informatica van verschillende kanten vernomen.

Echter, vorig jaar is de politieke beslissing genomen dat (voorlopig?) een dergelijk zelfstandig examenvak er niet komt. Men heeft in Den Haag gekozen voor 'integratie van elementen van informatie-technologie binnen verschillende schoolvakken'. Tot die vakken behoren wiskunde, natuurkunde en economie.

In dat kader zijn er voor wiskunde twee experimenten gestart. Aan de universiteit van Groningen gaat men de mogelijkheid onderzoeken onderwijs in de statistiek te koppelen aan computergebruik, met behulp van een op het onderwijs toegesneden statistisch pakket. Men wil dat pakket ook voorzien van een zgn. SQL-laag.

Aan de Vrije Universiteit zijn we vorig jaar begonnen met een experiment rond dynamische simulatie. We werken daartoe samen met twee leraren van het Waterland college in Amsterdam. Het ligt in de bedoeling een leerpakket te ontwikkelen voor ongeveer 20 lesuren.

De leerlingen brengen veel tijd door achter de computer, maar ook klassikale momenten zijn essentieel. Het gaat ons er om leerlingen ervaringen op te laten

doen met allerlei modellen bij verschijnselen, die zich zonder computer niet lenen voor kwantitatieve analyse.

De eerste modellen zijn eenvoudig en van de leerlingen wordt slechts gevraagd de parameters te wijzigen of een eenvoudige verfijning aan te brengen. Gaandeweg worden de modellen ingewikkelder. Ook wordt er steeds meer aan het initiatief van de leerling overgelaten. Allerlei modellen passeren de revue:

Bibliotheek. Hoe ontwikkelt zich het boekenbestand, als men rekening houdt met zaken als wegraken van boeken, subsidie en prijsontwikkeling.

Vervuiling. Een beschrijving van het afbreken van bepaald afval in open water. Wat zijn de effecten van maatregelen ter bescherming van het milieu?

Radio-actief verval. Over het opstellen van een model dat een complete vervalketen beschrijft.

Duidelijk moet worden, en daarbij speelt het klassegesprek een essentiële rol,
- dat veel van de modellen een zelfde structuur bezitten;
- wat de kenmerken zijn van een labiel en een stabiel evenwicht;
- dat er een groot onderscheid is tussen een lineaire en exponentiële ontwikkeling.

Al doende leert men heel wat van de taal VU-dynamo, maar we willen er voor waken dat het programmeer-aspect te veel nadruk krijgt.

In de tweede serie van 10 lessen ligt de nadruk op het maken van een ingewikkeld model. Daarom wordt een schema behandeld dat beschrijft hoe deskundigen zo'n model ontwikkelen.

Als complex model hebben we gekozen een model voor het Kaibab-plateau. Dit uit de literatuur bekende model [6] beschrijft de ontwikkeling van een herten-populatie, die bedreigd leek door de tijgers, maar bijna ten onder ging door goed bedoelde, maar verkeerd uitpakkende maatregelen van natuurbeschermers.

Het model wordt in vijf fasen opgebouwd. Eerst is er alleen sprake van herten die geboren worden en dood gaan. Vervolgens komt er een constant aantal tijgers in het model. Deze tijgers leven van de jacht op herten. Als derde wordt het aantal tijgers variabel: er worden immers tijgers geboren en er gaan tijgers dood. Bovendien zet de overheid een premie op het doden van de tijger. In de vierde fase komt het voedsel binnen het model. In de vijfde fase van het model, tenslotte, wordt ook in rekening gebracht dat de hoeveelheid aanwezig voedsel variabel is. De groei van de hoeveelheid voedsel hangt immers ook af van de hoeveelheid voedsel die er is.

Bij het opstellen van het Kaibab-model komt men in contact met enkele krachtige hulpmiddelen van de taal Dynamo.
- DELAY. Sommige verschijnselen hebben een natuurlijke vertraging. Een verminderd voedselaanbod heeft pas na enige tijd invloed op de gemiddelde levensduur.
- STEP. Sommige maatregelen worden op een zeker moment genomen en oefenen dan een geheel nieuw effect uit. Bijvoorbeeld het met ingang van een bepaalde datum uitloven van een premie op het neerschieten van tijgers.

- TABEL. Soms is het verband tussen twee grootheden alleen bekend voor bepaalde waarden. De tabelfunctie berekent dan de tussenliggende waarden via lineaire interpolatie.

Er is een sterk verband tussen de vergelijkingen in een dynamo-model en de analyse. Neem onderstaande level-vergelijking :

$$L \quad \text{afstand} = \text{afstand} + \text{snelheid} \star \Delta t$$

In de dagelijkse schoolpraktijk schrijven we meestal

$$s(t+\Delta t) = s(t) + v(t) \star \Delta t$$

En hier staat uiteraard

$\frac{\Delta s}{\Delta t} = v(t)$. Ofwel: je vindt de afstand door te zoeken naar een primitieve van de snelheid.

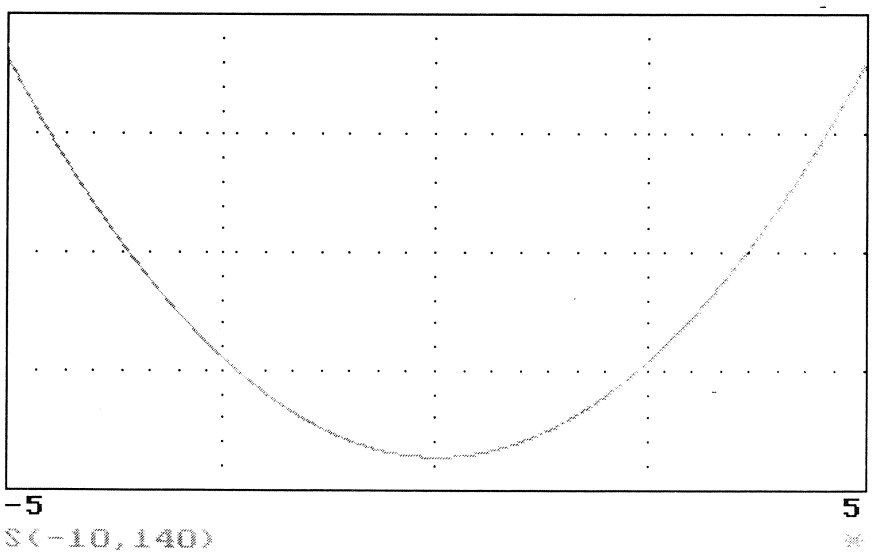
Dit blijkt ook uit het volgende programma.

```
l       s=s+v*Δt
r       v=10*t

n       s=125

spec Δt=0.01/begintijd=—5/eindtijd=5/prtper=1
plot s
print s
```

TIJD	Y
−5.0000	125.000
−4.0000	79.950
−3.0000	44.900
−2.0000	19.850
−1.0000	4.800
−0.0000	−0.250
1.0000	4.700
2.0000	19.650
3.0000	44.600
4.0000	79.550
5.0000	124.500



FIGUUR 2

Inderdaad: als Δt maar voldoende klein genomen wordt dan nadert s tot aan de formule $5t^2$.

Op een vergelijkbare manier kunnen differentiaalvergelijkingen onderzocht worden die niet analytisch oplosbaar zijn. We hebben alleen nog geen gegevens over wat leerlingen op dit terrein kunnen. Wel valt er iets te melden over de eerste 10 lessen over het onderwerp dynamische simulatie.

Die ervaringen zijn positief. Er werd enthousiast gewerkt. Dit geldt zowel voor de leerlingen als voor de leraar. Het enthousiasme van de leerlingen werd duidelijk uit hun werkhouding tijdens de lessen maar bleek ook uit het feit dat ze in de tussenuren naar het computerlokaal gingen.

De lessen droegen het karakter van een practicum, een voor de wiskundeleraar nieuwe situatie. Leerlingen werken samen, helpen elkaar, nemen resultaten van elkaar over. Het is in zo'n situatie niet eenvoudig uit te maken welke kennis persoonlijk eigendom is en welke gemeenschappelijk bezit.

De meeste leerlingen werkten gemotiveerd. Sommigen accepteerden bijvoorbeeld niet dat de docent de zaak wilde 'versnellen', ze wilden het zelf ervaren. Ons viel op dat het werken met VU-dynamo op relatief weinig problemen stuitte.

Wel problematisch waren de pogingen van de docent om een klassegesprek te houden over zaken als modelvorming. Dan verslapte de aandacht snel. Hierbij speelt mee dat ook in de 'gewone' wiskundeles klassegesprekken zeldzaam zijn.

Het is nog veel te vroeg om conclusies te trekken. Hoogstens kan men vaststellen dat leerlingen bereidt zijn met nieuwe en toch tamelijk moeilijke leerstof aan de slag te gaan, als ze zich maar uitgedaagd voelen. Dit laatste

geldt overigens ook voor de docenten die bij het experiment betrokken zijn. Ook voor hen is het onderwerp immers vrijwel nieuw.

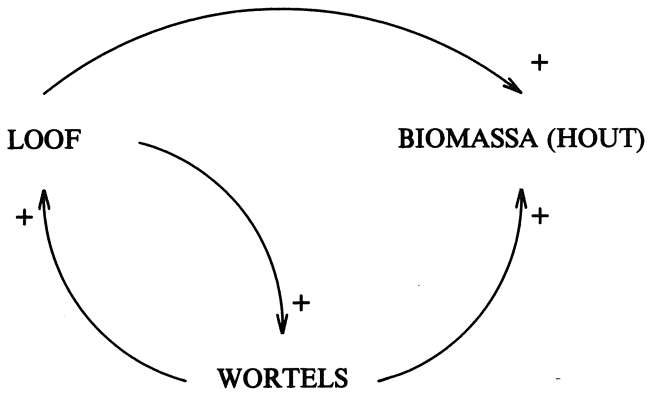
Met de aanwezigheid van de computer in het voortgezet onderwijs is er geen reden meer ons tot die differentiaalvergelijkingen te beperken die analytisch oplosbaar zijn. De computer voert niet alleen de numerieke benadering uit maar is ook in staat de numerieke methode grafisch toe te lichten. VU-dynamo is voor beide zaken geschikt. Maar het is minstens zo belangrijk dat de nadruk verschuift van het oplossen naar het opstellen van differentiaalvergelijkingen. En dat laatste gebeurt als leerlingen een dynamisch model opstellen. Ook vanuit het standpunt van de informatica is deze aandacht voor modelvorming van belang. Leerlingen ervaren dat je niet zomaar kunt beginnen met een model te maken. Een analyse vooraf is noodzakelijk.

4. Een ingewikkeld model: BOOMSTERFTE

Tot slot willen we een nogal ingewikkeld model demonstreren dat ook deel uitmaakt van het leerlingen-materiaal. Het model BOOMSTERFTE is niet geschikt om door leerlingen te worden opgesteld, maar we denken dat het voor hen wel leerzaam is er mee geconfronteerd te worden.

Door met verschillende waarden van de parameters te experimenteren krijgen de leerlingen enig gevoel voor de situatie, hopen we. Kunnen ze ervaren hoe een kleine wijziging in een parameter een dramatisch effect kan hebben op het verdere verloop.

Door verschillende wetenschappelijke instituten wordt er gewaarschuwd dat als er geen heel drastische maatregelen genomen worden, binnen enkele tientallen jaren de meeste bomen in Nederland dood zullen zijn. Kijken we echter om ons heen dan lijkt er weinig aan de hand te zijn. Pas op enkele plekken kun je zien dat onze bossen er slecht voor staan. Hoewel het nog onbekend is waardoor deze sterfte precies wordt veroorzaakt, is het al wel duidelijk dat de luchtverontreiniging de grote boosdoener is. Onduidelijk is of de schadelijke stoffen direct op het assimilatieproces werken of indirect via de verzuring van de bodem op de wortels. Bossel (1987) heeft een aantal modellen gemaakt om dit probleem van bossterfte te bestuderen. BOOMSTERF is een model voor een enkele naaldboom.

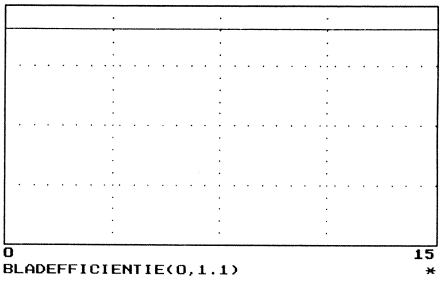


Enerzijds zorgt het loof door assimilatie (de omzetting van CO_2 door fotosynthese in organische stoffen) voor voedingsstoffen, die de wortels nodig hebben om te groeien. Aan de andere kant zorgen de wortels weer voor de aanvoer van vocht en mineralen die het loof weer nodig heeft om te kunnen functioneren.

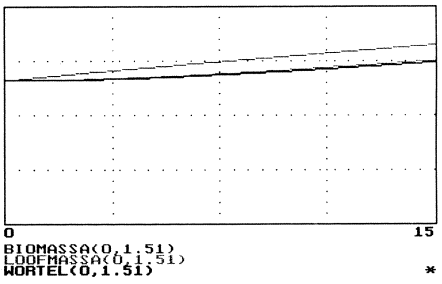
Bossel ging uit van de volgende veronderstellingen m.b.t. de energieverdeling:

- Een boom heeft in ieder geval 0.3 van zijn biomassa per jaar nodig om te kunnen functioneren. Wat over is aan voedingsstoffen wordt verdeeld tussen vruchten, wortels en loof.
- Bij voldoende voedingsstoffen blijven de naalden 8 jaar aan de boom hangen.
- De wortels hebben voedsel nodig vooral voor de vervanging van de kleine haarworteltjes. Het restant van de energie wordt omgezet in houtgroei.

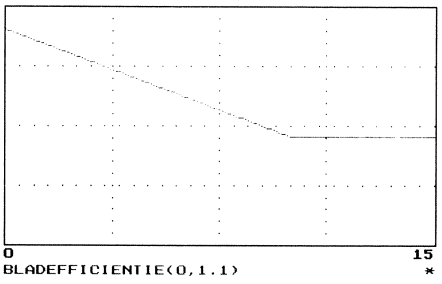
Als er geen sprake is van luchtverontreiniging is de 'bladefficiëntie' maximaal (gelijk aan 1). In het model is er dan inderdaad sprake van groei van hout, loof en wortels. Ook bij een bladefficiëntie van 0.6 is er nog steeds sprake van groei. Echter bij een bladefficiëntie van 0.5 zien we een dramatische omslag en sterft de boom zeer snel.



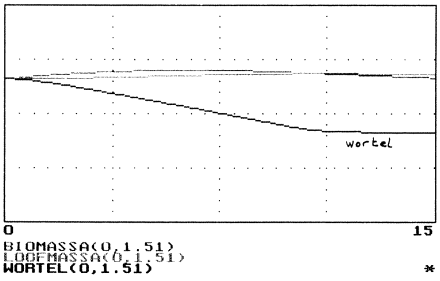
FIGUUR 3



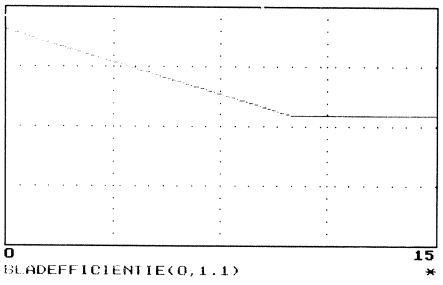
FIGUUR 4



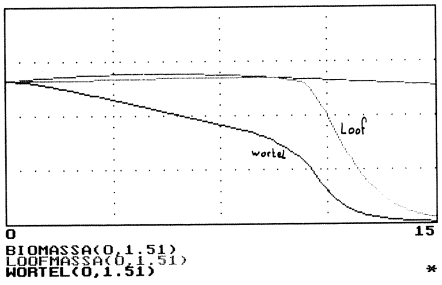
FIGUUR 5



FIGUUR 6



FIGUUR 7



FIGUUR 8

BOOMSTERFTE is een voorbeeld van een terugkoppelingsproces dat leidt tot een zichzelf versterkende ineenstorting van het systeem.

LITERATUUR

1. PIET VAN BLOKLAND. VU-dynamo. Een simulatie-programma voor het onderwijs. Faculteit Wiskunde en Informatica. Vrije Universiteit. De Boelelaan 1081. Amsterdam.
2. J.W. FORRESTOR ET AL. (1958). *Principles of Systems,* Cambridge, Massachusetts, USA; Wright Allen Press.
3. J.W. FORRESTOR (1971). *World Dynamics,* Cambridge, Massachusetts, USA; Wright Allen Press.
4. D.H. MEADOWS (1972). *Limits to Growth.*
5. G.P. RICHARDSON, A.L. PUGH III (1981). *Introduction to System Dynamics Modelling with DYNAMO,* MIT.
6. N. ROBERTS, ET AL. (1983). *Introduction to Computer Simulation,* Addison Wesley.

Modelvorming en Computergebruik bij Wiskunde-onderwijs

Heleen B. Verhage
Centrum voor Didactiek van Wiskunde en Natuurwetenschappen
Vakgroep OW & OC
Rijksuniversiteit Utrecht

1. Inleiding

De meeste ontwikkelingen op het gebied van computergebruik in het onderwijs hebben de afgelopen jaren plaatsgevonden in het kader van het NIVO-project [1]. De standaardisatie van apparatuur heeft een zekere rust gebracht. Daar was ook dringend behoefte aan na de chaotische start van de beginjaren met een enorme verscheidenheid aan inmiddels alweer antieke apparatuur.

Het NIVO-project heeft tevens op grote schaal nascholing voor docenten in gang gezet. Het betreft zowel basiscursussen informatica als vakgerichte cursussen over de computer als hulpmiddel binnen schoolvakken. Eén van de schoolvakken waarvoor een cursus ontwikkeld is, is wiskunde. Voor die cursus zijn negen verschillende thema's uitgewerkt en één daarvan is het thema Besliskunde. In deze bijdrage wil ik op dit thema verder ingaan. Verder zal ik ook de thema's *spreadsheet* en *dynamische simulaties* kort aanstippen.

Onderwerpen uit de matrixrekening en besliskunde (in het bijzonder lineair programmeren) maken deel uit van het examenprogramma voor wiskunde A op het VWO. Het mag geen toeval heten dat juist bij deze onderwerpen ten tijde van de experimenten met wiskunde A al nagedacht werd over het gebruik van de computer daarbij. Matrices worden in wiskunde A vooral gebruikt als hulpmiddel bij het maken van wiskundige modellen om er situaties en processen mee te beschrijven. In de praktijk wordt zo'n matrix dan al gauw vrij groot en is het bijkomende rekenwerk met de hand niet meer te doen. Een computer kan dan uitkomst bieden. Daarmee komt de aandacht vrij voor het *opstellen van modellen* en het *interpreteren van de uitkomsten.* Voor lineair programmeren geldt hetzelfde verhaal, computergebruik ligt daar voor de hand bij het verwerken van het standaardalgoritme voor LP-problemen, de simplexmethode.

Het type software dat bij dit soort computergebruik nodig is, zouden we

kunnen aanduiden met *instrumentele software.* De programmatuur is een hulpmiddel om de leerling rekenwerk uit handen te nemen en een didactische inkleuring is in eerste instantie nauwelijks aanwezig. Professionele software uit de beroepspraktijk zou bij wijze van spreken voldoen, ware het niet dat het leren omgaan met een professioneel pakket vermoedelijk een te grote tijdsinvestering vraagt ten opzichte van het beperkte aantal lessen dat leerlingen met zo'n programma bezig kunnen zijn. Om die reden is het dan toch wenselijk dat voor het onderwijs eenvoudige versies van dergelijke programmatuur ontwikkeld worden.

Aan de hand van enkele voorbeelden wil ik u iets laten zien van thans voor het onderwijs beschikbare programmatuur en het mogelijk gebruik daarvan in de klas.

2. Een model uit de biologie

Sinds de invoering van wiskunde A is elke VWO-docent vertrouwd geraakt met populatievoorspellingsmatrices (ook wel lesliematrices genaamd). Een voorbeeld louter daarover zal dus wellicht niet veel nieuws brengen. Toch start ik met een dergelijk probleem in de hoop dat het vervolg van mijn voorbeeld, waar lesliematrices en lineair programmeren bij elkaar komen, wel een bruikbare nieuwe suggestie voor de schoolpraktijk zal bevatten.

Het algemene populatievoorspellingsmodel zoals dat in wiskunde A behandeld wordt, luidt:

$$X(t+1) = PX(t)$$

$$\text{met } P = \begin{bmatrix} f_0 & f_1 & f_2 & \cdots & f_{n-1} & f_n \\ p_0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & p_1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & & & \\ 0 & 0 & 0 & & p_{n-1} & 0 \end{bmatrix} \text{ en } X(t) = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Uitgaande van een gegeven beginpopulatie $X(0)$ kan door het herhaald vermenigvuldigen van de matrix met de populatievector het model doorgerekend worden. In 't algemeen luidt de probleemstelling: *onderzoek hoe de populatie zich ontwikkelt.*

Door wat te experimenteren met de vruchtbaarheidscijfers f_i en de overlevingskansen p_i kunnen de leerlingen groei, periodiciteit en stabiliteit ontdekken. Bij dit experimenteren is de computer een onontbeerlijk hulpmiddel. Sterker nog, zonder computer kan dit onderwerp op deze manier nauwelijks behandeld worden, want de achterliggende wiskundige theorie over eigenwaarden en eigenvectoren behoort niet tot de wiskunde A stof.

Ter illustratie rekenen we een lesliematrix voor een populatie met drie leeftijdsklassen door. Een matrix en populatievector met beginwaarden:

$$P = \begin{bmatrix} 0 & 9 & 12 \\ 1/3 & 0 & 0 \\ 0 & 1/2 & 0 \end{bmatrix} \text{ en } X(0) = \begin{bmatrix} 33.3 \\ 33.3 \\ 33.3 \end{bmatrix}.$$

Wie niet zo gecharmeerd is van de decimale dieren, kan de beginpopulatie als een verdeling van de totale populatie genormeerd op 100 opvatten.

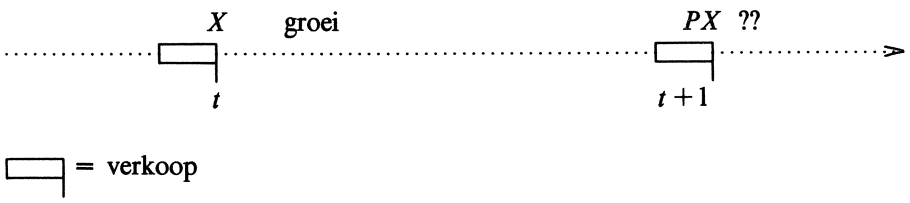
Voor het doorrekenen van het model is gebruik gemaakt van een spreadsheetprogramma, en wel het door NIVO verspreide programma PCcalc. In het WISCOM-materiaal [2] zijn toepassingen van PCcalc voor het wiskunde-onderwijs uitgewerkt. Enkele van de in dit kader gemaakte templates voor PCcalc betreffen het rekenen met matrices. In de figuur hieronder staat een beeldschermafdruk met de uitvoer voor de lesliematrix en beginpopulatie van hierboven.

	A	B	C	D	E	F	G	H	I	J
1										
2				*** MATRIX 3×3 ***					af begin rekenen	
3										
4		(	.000	9.000	12.000)		(	33.3)		
5	L =	(	.333	.000	.000)	B =	(	33.3)		
6		(	.000	.500	12.000)		(	33.3)		
7										
8	tijd	0	1	2	3	4	5	6	7	8
9										
10	1:	33.3	699.3	300.3	2163.3	2298.0	7083.6	11209.2	25820.4	47747
11	2:	33.3	11.1	232.9	100.0	720.4	765.2	2358.8	3732.7	8598
12	3:	33.3	16.7	5.6	116.5	50.0	360.2	382.6	1179.4	1866
13										
14	totale	100	727	539	2380	3068	8209	13951	30733	58211
15	groei		7.270	.741	4.416	1.289	2.676	1.699	2.203	1.894
16										
17	1:	33%	96%	56%	91%	75%	86%	80%	84%	82%
18	2:	33%	2%	43%	4%	23%	9%	17%	12%	15%
19	3:	33%	2%	1%	5%	2%	4%	3%	4%	3%
20										
21	F3=3 decimalen		F4=6 dec.		F5=rekenen	F6=verder rekenen		F7=schoonmaken		

Ter toelichting: L is de populatievoorspellingsmatrix, B de beginpopulatie. Na enig doorrekenen blijkt dat een verdeling over de leeftijdsklassen van 82.8% jong, 13.8% middel en 3.4% oud, stabiel is, althans relatief gezien. In absolute zin verdubbelt de populatie zich iedere tijdseenheid. Vervolgens voegen we een nieuw element aan de probleemstelling toe door aan te nemen dat de lesliematrix de ontwikkeling van een populatie fokdieren voorstelt en ons te verplaatsen in de huid van de fokker. Het probleem luidt nu:

Een fokker fokt een bepaalde diersoort en wil elke tijdseenheid een aantal dieren aan de populatie onttrekken voor de verkoop. Wat is een goede fok-strategie?

Groei en verkoop van de dieren schematisch weergegeven:



Een mogelijke oplossing zou kunnen zijn: neem als uitgangspunt de stabiele verdeling van dieren zoals hiervoor bepaald. Verkoop na elke tijdseenheid voor elke leeftijdsklasse het aantal dieren waarmee de omvang van die klasse is toegenomen. In het voorbeeld betekent dit:

$$\text{als } X(t)=\begin{bmatrix}82.8\\13.8\\3.4\end{bmatrix} \text{ dan } X(t+1)=\begin{bmatrix}165.0\\27.6\\6.9\end{bmatrix} \text{ zodat } X(t+1)-X(t)=\begin{bmatrix}82.8\\13.8\\3.4\end{bmatrix}$$

De populatie verdubbelt zich elke tijdseenheid, zodat de fokker steeds net zoveel dieren kan verkopen als hij al had. Bij een beginpopulatie van 100 dieren zijn dit 82.8 jonge dieren, 13.8 middel-dieren en 3.4 oude dieren, dus 100 dieren in totaal. Een sterk punt van deze oplossing is, dat de fokker in elk geval niet inteert op z'n kapitaal, de beginpopulatie. Aangezien de populatie elke tijdseenheid weer toeneemt, is de verkoop van dieren op deze manier gegarandeerd.

Het is echter de vraag of deze strategie *optimaal* is. Zou een andere verdeling over de leeftijdsklassen niet lucratiever kunnen zijn? Om hier iets over te kunnen zeggen, is het nodig meer te weten van de verkoopopbrengsten van de fokker. Laten we om te beginnen eens aannemen dat alle dieren evenveel opbrengen. Dan kan het heel goed zijn dat er een andere relatieve verdeling is die een hogere opbrengst geeft dan de verdeling die bij de stabiele situatie hoort. Bijvoorbeeld:

$$\text{als } X(t)=\begin{bmatrix}72\\20\\8\end{bmatrix} \text{ dan } X(t+1)=\begin{bmatrix}276\\24\\10\end{bmatrix} \text{ zodat } X(t+1)-X(t)=\begin{bmatrix}204\\4\\2\end{bmatrix}.$$

Voor de verkoop zijn nu 210 dieren beschikbaar, zodat deze tweede verdeling voor de fokker heel wat gunstiger is dan de eerder geprobeerde stabiele verdeling. De vraag die rijst, luidt:
Bij welke relatieve verdeling van het aantal dieren per leeftijdsklasse is de toename van het totaal aantal dieren per tijdseenheid maximaal ?

We nemen ook nog aan dat de fokker voor elke leeftijdsklasse een minimum aantal dieren wil aanhouden. Doet hij dit niet, dan wordt immers de kip met de gouden eieren geslacht: een fokker zonder fokdieren verliest zijn broodwinning. Aan deze voorwaarde wordt voldaan, als de fokker steeds per leeftijdsklasse alleen de *toename* van het aantal dieren verkoopt.

Om tot een wiskundige formulering van de probleemstelling te komen, is het

handig een winstvector in te voeren, waarin voor elke leeftijdsklasse de opbrengst per dier genoteerd wordt: $W=(w_0,w_1,\ldots,w_n)$.

3. Lineair programmeren

De verbinding met het onderwerp lineaire programmering kan nu gelegd worden, want het probleem van de fokker is een optimaliseringsprobleem met een lineaire doelfunctie en lineaire nevenvoorwaarden. Het LP-model ziet er zo uit:

max $W(X(t+1)-X(t))$	(de opbrengst bij de verkoop van $X(t+1)-X(t)$ dieren)
Onder de voorwaarden:	
$X(t+1)\geqslant X(t)$	('interen' moet voorkomen worden)
$X(t)\geqslant 0$	(biologische vanzelfsprekendheid)
$\Sigma x_i(t)=N$	(het totaal aantal dieren wordt genormeerd op N)

Deze laatste voorwaarde is nodig omdat het probleem anders onbegrensd is. Na verkoop van $X(t+1)-X(t)$ dieren heeft de fokker weer het oorspronkelijke aantal van $X(t)$ dieren tot zijn beschikking, zodat het proces van groei en verkoop in iedere tijdseenheid op dezelfde wijze kan verlopen. Substitutie van $X(t+1)=P\cdot X(t)$ in het model en enig herschrijven geeft:

$$\max W(P-I)X(t)$$

Onder de voorwaarden:

$$\begin{aligned}(I-P)X(t) &\leqslant 0\\ X(t) &\geqslant 0\\ \Sigma x_i(t) &= N\end{aligned}$$

Hiermee staat het probleem in de standaardvorm voor lineaire programmering. Uitschrijven van het model voor het concrete voorbeeld

met $P=\begin{bmatrix}0 & 9 & 12\\ 1/3 & 0 & 0\\ 0 & 1/2 & 0\end{bmatrix}$, $W=(1,1,1)$ en $N=100$ geeft:

$$\max -\tfrac{2}{3}x_0+8.5x_1+11x_2$$

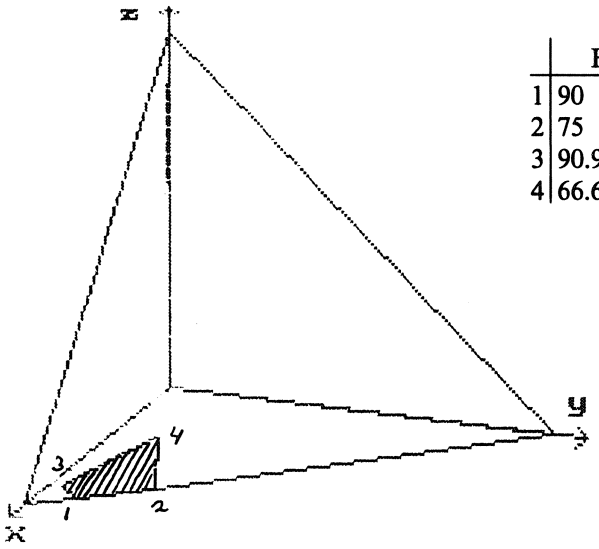
Onder de voorwaarden:

$$\begin{aligned}x_0-9x_1-12x_2 &\leqslant 0\\ -\tfrac{1}{3}x_0+x_1 &\leqslant 0\\ -\tfrac{1}{2}x_1+x_2 &\leqslant 0\\ x_0+x_1+x_2 &= 100\end{aligned}$$

Omwille van de overzichtelijkheid is $x_i(t)=x_i$ geschreven.

Dit probleem met drie beslissingsvariabelen kan opgelost worden met standaardprogrammatuur voor LP. Terugkerend naar de schoolpraktijk zijn hiervoor verschillende programma's te gebruiken. Omdat het een probleem met drie variabelen betreft, is een mogelijke oplossingsweg de grafische methode. De grafische methode houdt in dat eerst het toelaatbare gebied (dat altijd convex is) volledig bepaald wordt. Vervolgens wordt gekeken in welk hoekpunt (eventueel: langs welke ribbe of grensvlak) de doelfunctie optimaal is. De grafische methode is echter in de praktijk met de hand nauwelijks uitvoerbaar vanwege het vele tekenwerk. Het tijdrovende tekenen van toelaatbare gebieden zou dan de overhand krijgen, hetgeen niet de bedoeling van lineair programmeren kan zijn. Dit praktische probleem neemt niet weg dat het *idee* van de grafische methode wel aantrekkelijk is. Het maakt immers het principe om langs de ribben van een convex gebied te wandelen aanschouwelijk.

Om toch de grafische methode te kunnen toepassen, is het programma LINPROG ontwikkeld [3]. In dat programma wordt het toelaatbare gebied stap voor stap opgebouwd, waardoor de ligging van elke beperking in de driedimensionale ruimte goed te zien is. We verwerken het probleem van de fokker met dit programma. In de figuur hieronder is het toelaatbare gebied gearceerd en in de tabel naast de figuur staan de coördinaten van de hoekpunten van het toelaatbare gebied en de bijbehorende waarden van de winstfunctie.



	Hoekpunten			Winst
1	90	10	0	25
2	75	25	0	75
3	90.90	6.06	3.03	24.24
4	66.67	22.22	11.11	266.67

Uit de tabel blijkt dat een populatie van $X=(66.7\ 22.2\ 11.1)$ een optimale verdeling van dieren over leeftijdsklassen geeft. Eén tijdseenheid later heeft de fokker dan (333 22.2 11.1) dieren, zodat er 266 jonge dieren verkocht kunnen worden. Het aantal dieren in de andere leeftijdsklassen is precies op peil gebleven, wat één van de randvoorwaarden was! De strategie van de fokker is in dit geval: fok zoveel mogelijk jonge dieren en verkoop die. De opbrengst is

nu heel wat groter dan in het geval dat de fokker uitgaat van de stabiele verdeling (82.8 13.8 3.4) die eerder bepaald is. In dat geval schieten er per tijdseenheid immers slechts 100 dieren over.

Het wordt nog anders als de oude dieren veel meer waard zouden zijn dan de jonge. De winstfunctie moet dan aangepast worden. Bijvoorbeeld bij een winstvector $W=(0,10,20)$ wordt het doel:

$$\max \tfrac{10}{3}x_0 - 20x_2$$

Onder dezelfde beperkende voorwaarden als hiervoor.

In dit geval blijkt de verdeling van dieren $X=(90,10,0)$ optimaal te zijn. Eén tijdseenheid later zijn er dan (90, 30, 5) dieren, zodat de fokker 20 middel- en 5 oude dieren kan verkopen zonder in te teren, wat een opbrengst geeft van 300.

4. Het harvesting model en het lesliematrix model

De verkoop van dieren kan ook met een matrixmodel beschreven worden. Noem H de matrix met op de hoofddiagonaal per leeftijdsklasse de fractie dieren die overblijft na verkoop (de *harvesting matrix*):

$$H = \begin{bmatrix} h_0 & 0 & 0 & \cdots & 0 \\ 0 & h_1 & 0 & \cdots & 0 \\ 0 & 0 & h_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & h_n \end{bmatrix}$$

Als na verkoop de populatie weer zijn oorspronkelijke omvang moet hebben en de groei beschreven wordt door $X(t+1)=PX(t)$, betekent dit dat $HX(t+1)$ weer gelijk moet zijn aan $X(t)$. Er is dus evenwicht tussen groei en verkoop als $X(t)=HPX(t)$.

De matrix H is te bepalen als bij een gegeven winstvector W de optimale populatie X is vastgesteld. In het voorbeeld met $W=(0,10,20)$ en $X=(90,10,0)$ wordt dat:

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1/3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \text{ zodat } HP = \begin{bmatrix} 0 & 9 & 12 \\ 1/9 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

De interpretatie: van de jonge dieren wordt er geen enkele verkocht, van de middel-dieren 2/3 deel en de oude dieren allemaal.

We hebben nu twee lineaire modellen uit de biologie met elkaar gecombineerd: het *harvesting model* en het *lesliematrix model.* Bovendien zijn twee computerprogramma's de revue gepasseerd: een programma voor het doorrekenen van matrixmodellen en een programma voor ruimtelijk lineair programmeren. Beide getoonde programma's zijn alleen geschikt voor modellen met drie variabelen. Een programma voor grotere matrices verschilt niet

wezenlijk van het hier gepresenteerde, maar voor lineair programmeren met meer dan drie variabelen moet wel een andere oplossing gezocht worden dan de grafische methode. We komen dan terecht bij het standaardalgoritme voor lineair programmeren: de simplexmethode. Een geschikt programma hiervoor is SIMOPT [4]. Aan de hand van een tweede voorbeeld demonstreren we in- en uitvoer van dit programma.

In 1830 waren er in het westen van de VS veertig miljoen bizons. In 1887 waren er nog maar 200 over, want de dieren werden ongelimiteerd afgeslacht.

Van de Amerikaanse bizon is bekend dat:
- *100 volwassen vrouwtjes elk jaar gemiddeld 90 kalveren werpen, waarvan 48 mannetjes en 42 vrouwtjes;*
- *60% van de kalveren het eerste levensjaar overleeft;*
- *75% van de tweejarige dieren volwassen wordt;*
- *elk jaar 5% van de volwassen dieren sterft.*

>> *Bij welke leeftijdsopbouw is de relatieve verdeling van de bizons stabiel?*

De matrix voor de bizonpopulatie is:

$$A = \begin{bmatrix} 0 & 0 & 0.42 & 0 & 0 & 0 \\ 0.6 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.75 & 0.95 & 0 & 0 & 0 \\ 0 & 0.42 & 0.48 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.75 & 0.95 \end{bmatrix}$$

De stabiele leeftijdsverdeling is $X=(0.112, 0.061, 0.294, 0.128, 0.069, 0.336)$. De groeifactor voor de bizonpopulatie is ongeveer 1.105. Stel nu dat men kuddes dieren wil houden en dat alleen de volwassen dieren verkoopwaarde hebben. Wat is dan de beste samenstelling voor een kudde?

Geheel analoog aan het vorige voorbeeld kan er weer een LP-model opgesteld worden met de matrix hierboven en $W=(0,0,1,0,0,1)$ als vertrekpunt. Invoer van het model:

```
max      0.75vmiddel - 0.05vvolw +                0.75mmiddel - 0.05mvolw
                       0.42vvolw                                          ≥ vkalf
0.6vkalf                                                                  ≥ vmiddel
         0.75vmiddel + 0.95vvolw                                          ≥ vvolw
                       0.48vvolw                                          ≥ mkalf
                                    0.6Mmkalf                             ≥ mmiddel
                                                 0.75mmiddel + 0.95mvolw ≥ mvolw
  vkalf +    vmiddel +    vvolw +     mkalf +     mmiddel +    mvolw =1000
```

en verwerking met het programma SIMOPT geeft als uitvoer:

Summary of Results

Value Objectfunction : 145.492

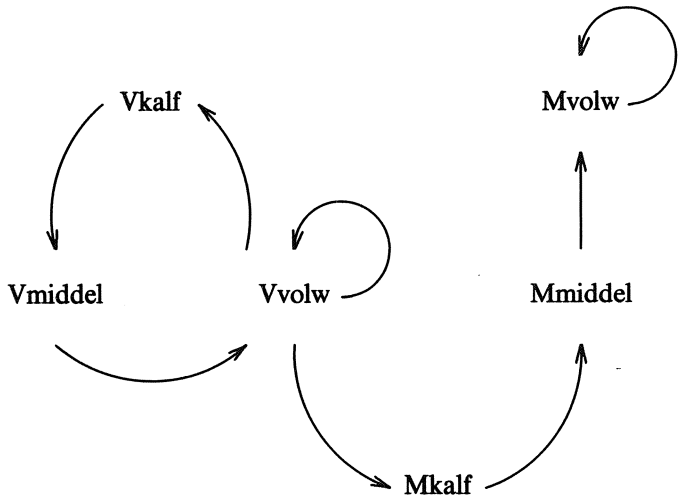
		Activity Level
vmiddel	:	103.279
vvolw	:	409.836
mmiddel	:	118.033
mvolw	:	0.000
vkalf	:	172.131
mkalf	:	196.721

Een optimale verdeling over de klassen is nu $X=(17.2, 10.3, 42, 19.7, 11.8, 0)$. Toepassen van de populatievoorspellingsmatrix geeft de populatie een tijdseenheid later: (17.2, 10.3, 46.7, 19.7, 11.8, 8.9). Van de volwassen vrouwtjes wordt 12.2% verkocht en de volwassen mannetjes worden allemaal verkocht. Dit model gaat voorbij aan het feit dat er ook mannetjes nodig zijn voor de reproduktie, vandaar die verkoop van 100%. Er zullen dus toch enkele mannetjes gehouden moeten worden ten behoeve van de voortplanting.

5. Dynamische simulatie

In het NIVO-nascholingsmateriaal voor wiskunde is ook het thema *dynamische simulaties* opgenomen. Bij dit thema staat het maken van wiskundige modellen centraal. Het computerprogramma dat in de cursus gebruikt wordt, is VU-dynamo [5]. De werking van dit pakket en de mogelijkheden voor het onderwijs worden besproken in de bijdrage van D. Kok en P. van Blokland elders in deze bundel. Op deze plaats illustreren we de relatie tussen matrixmodellen en het modelleren met VU-dynamo aan de hand van het bizonprobleem.

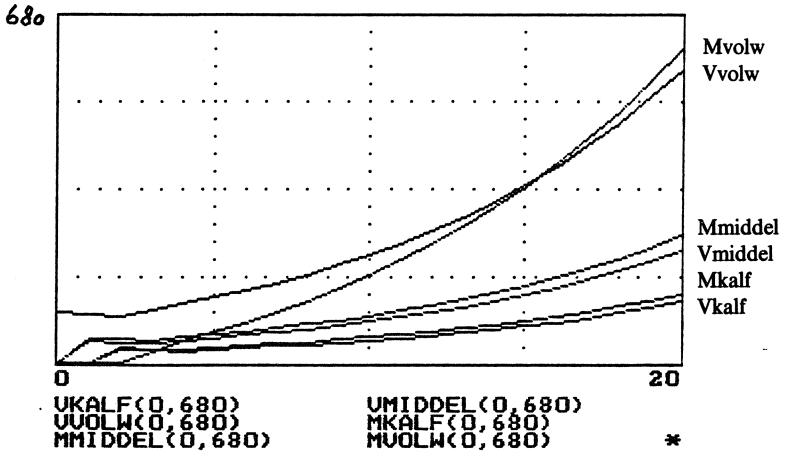
Een veelgebruikt hulpmiddel bij modelformulering van dynamische processen is het schematisch weergeven van verbanden tussen variabelen in een diagram. Het diagram voor de bizons ziet er zo uit:



De volgende stap is het opstellen van de modelvergelijkingen volgens de regels van het computerprogramma. Het bizonmodel in VU-dynamo is:

```
L vkalf = 0.42 * vvolw
L vmiddel = 0.6 * vkalf
L vvolw = 0.95 * vvolw + 0.75 * vmiddel
L mkalf = 0.48 * vvolw
L mmiddel = 0.6 * mkalf
L mvolw = 0.95 * mvolw + 0.75 * mmiddel
N vkalf=0
N vmiddel=0
N vvolw=100
N mkalf=0
N mmiddel=0
N mvolw=0
SPEC dt=1/eindtijd=10/prtper=1
PRINT vkalf/vmiddel/vvolw/mkalf/mmiddel/mvolw
PLOT vkalf,vmiddel,vvolw,mkalf,mmiddel,mvolw.
```

In feite staat hier niet veel anders dan de uitgeschreven matrixvermenigvuldiging van het matrixmodel, gevolgd door een rijtje beginwaarden. Omdat het bizonmodel lineair is, komen beide aanpakken eigenlijk op hetzelfde neer. VU-dynamo kan echter ook niet-lineaire modellen aan. Bovendien is het goed mogelijk om een model stapsgewijs op te bouwen en uit te breiden, een belangrijk voordeel als de modellen ingewikkelder worden. Een grafiek van de uitvoer nadat het model tien tijdseenheden is doorgerekend:



Begonnen is met een populatie van uitsluitend 100 volwassen vrouwtjes. In de grafiek is te zien hoe na wat aanloophobbels in het begin exponentiële groei ontstaat.

6. Conclusie

Bij het hier gepresenteerde computergebruik ligt het accent sterk op software die ingezet wordt als rekengereedschap. Dit is zeker niet de enige mogelijkheid, maar het is wel een richting die naar mijn smaak perspectief biedt. Bovendien past deze vorm van computergebruik goed in de sfeer van met name wiskunde A: de tendens om leerlingen realistische contexten aan te bieden en aandacht te besteden aan het opstellen en interpreteren van modellen.

Voegt het gebruik van de computer nu daadwerkelijk iets toe aan het wiskunde-onderwijs? De discussie over deze vraag kan pas goed gevoerd worden als we niet al te veel last meer hebben van de ruis van allerlei praktische problemen. Nu is het nog vaak zo dat er van een docent die de computer in de les wil gebruiken heel wat organisatie- en improvisatievermogen gevraagd wordt. Toch valt er al wel iets te zeggen over de potentiële mogelijkheden van het type computergebruik dat uit de voorbeelden spreekt:

- het 'spelen' en experimenteren met wiskundige modellen wordt haalbaar voor het onderwijs, wiskunde kan daardoor een dynamischer vak worden;
- het aanbieden van grotere problemen wordt mogelijk, en daarmee wellicht de integratie van verschillende onderwerpen;
- algoritmen en oplossingsmethoden kunnen stapsgewijs op de computer uitgevoerd worden, zodat de leerling als het ware kan meekijken zonder zelf al het rekenwerk te hoeven doen;
- in sommige gevallen kan het oplossingsproces ook gevisualiseerd worden, zoals bij het ruimtelijk lineair programmeren het geval was.

Uiteindelijk zullen ervaringen uit de schoolpraktijk moeten uitwijzen of computers inderdaad een aanwinst voor het onderwijs zijn.

Een noodzakelijke voorwaarde om daar achter te komen, is dat docenten voldoende zijn nageschoold om met apparatuur en programmatuur uit de voeten te kunnen. Het is dan ook heel terecht dat het NIVO-project heeft voorzien in nascholing van docenten.

NOTITIES

1. NIVO: Nieuwe Informatietechnologie in het Voortgezet Onderwijs.
2. WISCOM - voorbeelden van computergebruik bij wiskunde, Vakgroep OW&OC, Rijksuniversiteit Utrecht, 1987.
3. Het programma LINPROG is ontwikkeld bij OW&OC te Utrecht, door Kees Henzen en Heleen Verhage.
4. Het programma SIMOPT is ontwikkeld aan de VU te Amsterdam, door Erwin Kalvelagen.
5. Het programma VU-dynamo is ontwikkeld aan de VU te Amsterdam, door Piet van Blokland.

Discussie

Wiskunde en de Computer

P.C. Baayen
Centre for Mathematics and Computer Science
P.O. Box 4079, 1009 AB Amsterdam, The Netherlands

Computers have influenced mathematics in two quite different ways: they have made mathematics more powerful than ever before, and they have altered the very nature of the mathematical sciences.
Lynn Arthur Steen [23]

0. Afpaling van het thema

In wetenschap en techniek komt men zowel het gebruik van wiskunde als de toepassing van computers alom tegen. Verbanden tussen wiskunde en de computer bestaan dan ook op veel niveaus en in allerlei vormen. Niet alle staan centraal in dit symposium.

Alleen al de *naam* computer - of rekenmachine, of logische automaat - verwijst naar de wiskunde. Een halve eeuw geleden betekende 'computer' nog 'rekenaar (m/v)'; zo b.v. bij Alan Turing [24], waar het werk van zo'n humane computer voor het eerst volledig ontleed wordt in atomaire handelingen. Deze analyse voerde niet alleen tot de beschrijving van (Turing-) berekenbaarheid, maar ook tot het inzicht dat universele programmeerbare logica-machines bestaan; in de abstracte vorm van een universele Turing-machine zijn ze zelfs vrij eenvoudig te beschrijven. Doel van Turing was de behandeling van het Entscheidungsproblem voor de eerste-orde logica, en zijn werk was dus stevig geworteld in een wiskundig onderzoeksprogramma, het grondslagenprogramma van D. Hilbert (vgl. [6], alsook [1]).

Maar dit *historisch-genetische* verband tussen wiskunde en de computer (waarbij Turing staat in een rij wiskundige grondleggers van de logica-machine die zich uitstrekt van Pascal en Leibniz tot Von Neumann en Goldstine) wordt niet bedoeld door de organisatoren van dit congres, en het zal hier daarom verder buiten beschouwing blijven. Wie juist in de historie geïnteresseerd is zij gewezen op [1], [18], [19].

Het thema is evenmin *Wiskunde en Computer Science.* Op zichzelf zou dat ook de moeite waard zijn geweest. De ontwikkeling van en de theorievorming over zowel apparatuur als programmatuur voeren tot moeilijke en (soms) interessante wiskundige problemen. Voorbeelden zijn complexiteitstheorie, de ontwikkeling van nieuwe, beter in silicon realiseerbare algoritmen voor

bekende operaties en functies, algoritmiek ontwikkeld voor programmatransformaties, nieuwe meetkundige vraagstellingen ten behoeve van het topologisch ontwerp van VLSI chips, weer andere 'computational geometry' problemen rondom robots en andere intelligente autonome automaten (zie o.m. [3]), ontwikkelingen in de tralietheorie vanuit de semantiek van programmeertalen, tijdslogica en andere niet-klassieke logica voor de beschrijving en interpretatie van concurrente processen, en zo nog veel meer. Maar iedere volwassen wetenschap voert tot nieuwe wiskunde, of op zijn minst tot nieuw gebruik van wiskunde, en de Computer Science neemt hierbij geen bijzondere positie in.

Een ander op zichzelf belangrijk onderwerp dat niet de kern raakt van het thema van deze dag, is dat van de *Wiskundige en de Computer.* Natuurlijk, het leven en werken van de wiskundige wordt beïnvloed door de opkomst van de elektronische informaat, en zal daar nog veel sterker door worden beïnvloed. De wiskundige als deelneemster aan de westerse cultuur ontmoet computers (of zij het weet of niet) in wasmachines, compact disc spelers en smart cards. Als burger ontdekt zij dat de overheid vroeger of later allerlei gegevensbestanden koppelt, en het Sofi-nummer toch gebruikt voor zaken waar het niet voor bedoeld was (en dat derden, en niet alleen 'hackers', op de een of andere manier, tegen alle afspraken in, ook toegang tot die gegevensbanken weten te verwerven). Ze is, met andere burgers, blij dat bij road-pricing en elektronische betalingscircuits gebruik wordt gemaakt van wiskundige crypto-systemen, zodat nog enige privacy blijft. De wiskundige als auteur gebruikt computers voor tekstverwerking en voor het componeren en bewaren van tekstontwerpen; raadpleegt daarbij gegevensbestanden en elektronische abstract-services; de wiskundige als deelneemster aan een (formeel of informeel) netwerk profiteert van e-mail, van elektronische kalenders en van electronic conferencing. Misschien realiseert ze zich dankbaar dat de noeste werkers van Euromath haar daarbij enige tijd een voorsprong hebben verschaft ten opzichte van auteurs en onderzoekers uit andere vakgebieden; maar die voorsprong is allengs ingehaald: uiteindelijk zijn deze verworvenheden natuurlijk niet specifiek voor de wiskunde. In de rubriek 'Computers and Mathematics', in de Notices van de AMS sinds mei 1988, geredigeerd door Jon Barwise [2], komen zulke gebruikersapecten geregeld naar voren.

Wat is dan wèl de kern van het thema 'Wiskunde en de Computer'? Ik meen dat die treffend is samengevat in het in de aanhef aangehaalde citaat van Lynn Steen: de computer heeft de wiskunde krachtiger en bruikbaarder gemaakt dan ooit eerder in de geschiedenis, en door de computer wordt de wiskunde wezenlijk veranderd.

1. Toelichting van het thema

Aan de vergroting door de computer van de *power,* de *effectiviteit* van de wiskunde zijn een aantal aspecten te onderscheiden. De computer is onmisbaar geworden als rekenmachine bij het numeriek oplossen van vergelijkingen, met velerlei toepassingen in het mathematisch modelleren en simuleren; als informatieverwerkende automaat bij formule-manipulatie en symbolisch

rekenen; als logica-machine bij het ondersteunen, controleren of zelfs genereren van bewijzen.

De laatste jaren is (voornamelijk in de USA) een aanzienlijk aantal publicaties verschenen, waarin wordt betoogd hoezeer zo de beschikbaarheid van moderne elektronische rekenmachines de toepasbaarheid, de 'power' van wiskundige methoden heeft vergroot, met als gevolg dat velerlei nieuwe toepassingsgebieden werden ontsloten. Was het befaamde Lax-report [16] niet specifiek gericht op de wiskunde, bij latere rapporten zoals [20], [21], [7] en [9] is dit wel het geval.

Wellicht is een korte samenvatting van de inhoud nuttig voor diegenen, die van deze publicaties geen kennis hebben genomen. Zo bevat [20] een hoofdstuk met voorbeelden van het belang van 'computational modelling' voor twaalf verschillende wetenschapsgebieden, zoals hydrodynamische systemen, niet-destructief toetsen en tomografische reconstructie, elektronische componenten en 'device simulation'. Dan volgt een hoofdstuk over rekentechnische en wiskundige problemen: vrijheidsgraden; tijds- en lengteschalen; singulariteiten in coefficiënten, gegevens of toestanden; randvoorwaarden; bifurcatie en chaos; 'ill-posed' inverse problemen; effectieve media; validatie, foutberekening en analyse van gevoeligheid. Het laatste hoofdstuk behandelt tevens de diverse in gebruik zijnde numerieke methoden.

In [21] wordt meer aandacht geschonken aan onderzoeksmogelijkheden, waarbij onderwerpen aan de orde komen uit: niet-lineaire elliptische differentiaalvergelijkingen; lineaire en niet-lineaire vergelijkingen; niet-lineaire hyperbolische vergelijkingen; adaptieve roosters; ill-posed problems; computational statistics; stochastiek en computational probability; patroonherkenning en symbolische berekening; parallel rekenen; en diverse toepassingen in de zuivere wiskunde. Voor het onderzoek in de mathematische statistiek geeft [7] verdere informatie, met aanbevelingen voor de ondersteuning in hardware en software van het onderzoek. In [9] worden in het kader van een studie betreffende de wiskundige systeem- en regeltheorie vignetjes geschilderd over computational methods for control en over optimal control algorithms.

Naast de toegenomen effectiviteit van de wiskunde stelde ik, in navolging van Lynn Steen, dat door de computer *de wiskunde naar zijn aard verandert.* Steen is niet de eerste die dit aan de orde stelt. In zijn voordracht 'Experimental Mathematics', gehouden op het CWI-symposium in 1983, werkte M. Hazewinkel deze stelling uit aan de hand van een gevarieerd aantal voorbeelden en met rijke verwijzing naar de literatuur [13]. Op de First International Conference on Industrial and Applied Mathematics (ICIAM '87, Parijs) behandelde Peter D. Lax (in beknoptere vorm) hetzelfde thema [17]; ook daar treft U waardevolle verdere literatuurverwijzingen aan.

Het is niet zinvol de betogen van Hazewinkel en Lax hier te herhalen of zelfs ze samen te vatten. Laat ik volstaan met een paar illustraties. Dank zij de computer is duidelijk geworden hoe gevoelig niet-lineaire systemen dikwijls zijn voor (zelfs uiterst kleine) veranderingen van de randvoorwaarden. De z.g. chaos-theorie spreekt tot de verbeelding, juist ook van niet-wiskundigen (en geeft zelfs aanleiding tot cultus-verschijnselen, met Benoit Mandelbrot als

guru); zie ook het recente populariserende werk van James Gleick [11], en vooral de bespreking hiervan door John Franks [10]. Het gebruik van computer graphics voert tot doorbraken bij meetkundig onderzoek; zie b.v. de bijdrage van Th.F. Banchoff in [8],1-14, David Hoffman [14], alsook [15]. De oplossing (in positieve zin) van het vier-kleuren-probleem door W. Haken en K. Appel is het bekendste voorbeeld van een computer-afhankelijk bewijs, maar niet het enige (zie b.v. [24]). De complexiteitstheorie is ontstaan en snel tot grote bloei gekomen door de mogelijkheden (wellicht nog meer: door de beperkingen) van de computer, en de mathematische cryptographie is een prominent toepassingsgebied. In allerlei wiskundig onderzoek is het algoritmische aspect veel belangrijker geworden dan voorheen, en soms is het ontwikkelen van snellere algoritmen zelfs een van de primaire doelen geworden. Ook dit leidt tot nieuw onderzoek, soms ook tot herleving van oude methoden, zoals die van de meetkunde der getallen in het werk van A.K. Lenstra, H.W. Lenstra, Jr. en L. Lovász (zie [18]).

2. DISCUSSIESTELLINGEN

1. Voor alle wiskundig onderzoek biedt de computer waardevolle ondersteuning.
2. Door de computer verandert wiskundig onderzoek qualitatief.
3. Voor de wiskundige onderzoekers moeten geavanceerde computervoorzieningen (inclusief supercomputers) 'laag-drempelig' beschikbaar zijn.
4. Door de computer verandert de werkstijl; groepswerk wordt bv. steeds belangrijker.
5. Het wetenschappelijk onderwijs aan wiskundigen is onvoldoende ingesteld op de relatie 'wiskunde en computer'.
6. De computer maakt een wezenlijk andere organisatie van de wiskundige kennis mogelijk; deze belangrijke mogelijkheid wordt nog nauwelijks gebruikt.

REFERENTIES

1. WILLIAM ASPRAY. The Mathematical Reception of the Modern Computer: John von Neumann and the Institute for Advanced Study Computer. In: [19], 166-194.
2. JON BARWISE (ed.) Computers and Mathematics. Feature Column in *Notices of the AMS,* from May/June 1988, Vol. 35 (693-694) onwards.
3. JOHN CANNY (1988). *The Complexity of Robot Motion Planning,* Cambridge, Mass.
4. EDWARD E. DAVID, JR. (chairman) (1984). *Renewing U.S. Mathematics, Critical Resource for the Future,* Report of the Ad Hoc Committee on Resources for the Mathematical Sciences, National Academy Press, Washington D.C.
5. EDWARD E. DAVID, JR. (1988). Renewing U.S. Mathematics, An Agenda to Begin the Second Century. *Notices of the American Mathematical Society, 35, nr 8.*
6. MARTIN DAVIS (1988). Mathematical Logic and the Origin of Modern

Computers. In: [19], 137-165.
7. WILLIAM F. EDDY (chairman) (1986). *Computers in Statistical Research.,* Report of a Workshop on the Use of Computers in Statistical Research. Institute for Mathematical Statistics.
8. R.E. EWING, K.I. GROSS, C.F. MARTIN (1986). The Merging of Disciplines: New Directions in C.F. Martin Pure, Applied and Computational Mathematics. *Proceedings symposium held in honor of Gail S. Young,* Univ. of Wyoming, Aug. 8-10, 1985, Berlin.
9. WENDELL H. FLEMING (chairman) (1988). *Future Directions in Control Theory, A Mathematical Perspective,* Report of the Panel on Future Directions in Control Theory, SIAM Reports on Issues in the Mathematical Sciences, Philadelphia.
10. JOHN FRANKS (1988). Review of James Gleick, Chaos Making a New Science, with Reply by James Gleick, and Response by John Franks. *The Mathematical Intelligencer 11,* 65-71.
11. JAMES GLEICK (1987). *Chaos Making a New Science,* New York.
12. JUDITH V. GRABINER (1986). The Centrality of Mathematics in the History of Western Thought. *Proceedings of the International Congress of Mathematicians,* Berkeley, California, USA.
13. M. HAZEWINKEL (1986). Experimental mathematics. In *Mathematics and Computer Science,* Proceedings of the CWI Symposium, November 1983, Amsterdam.
14. DAVID HOFFMAN (1987). The Computer-Aided Discovery of New Embedded Minimal Surfaces. *The Mathematical Intelligencer 9,* 8-21.
15. PAUL HOFFMAN (1988). *Archimedes' Revenge,* New York & London.
16. PETER D. LAX (1982). Report of the Panel on Large Scale Computing in Science and Engineering, Sponsored by DOD and NSF in cooperation with DOE and NASA, National Science Foundation.
17. PETER D. LAX (1988). Mathematics and Computing. In ICIAM '87: *Proceedings of the First International Conference on Industrial and Applied Mathematics,* SIAM Philadelphia.
18. L. LOVÁSZ (1988). Geometry of Numbers: An Algorithmic View. In ICIAM '87: *Proceedings of the First International Conference on Industrial and Applied Mathematics,* SIAM Philadelphia.
19. MICHAEL S. MAHONEY (1988). The History of Computing in the History of Technology. *Annals of the History of Computing 10,* 113-125.
20. SEYMOUR V. POLLACK (ed.) (1982). Studies in Computer Science. *Studies in Mathematics,* Vol. 22, MAA.
21. WERNER C. RHEINBOLDT (chairman) (1984). Computational Modeling and Mathematics Applied to the Physical Sciences, Report of the Committee on the Applications of Mathematics, Office of Mathematical Sciences and Commission on Physical Sciences, Mathematics, and Resources, National Research Council. Washington D.C.

22. WERNER C. RHEINBOLDT (chairman) (1985). Future Directions in Computational Mathematics, Algorithms, Scientific Software, Report of the Panel on Future Directions in Computational Mathematics, Algorithms and Scientific Software, Philadelphia.
23. LYNNA ARTHUR STEEN (1989). Mathematics for a New Century. *Notices AMS 36,* 133-138.
24. ALAN TURING. On Computable Numbers with an Application to the Entscheidungsproblem. *Proc. London Math. Soc., Ser. 2, 42,* 230-267.
25. PAUL WALLICH (1989). Beyond Understanding? Computers are changing the spirit of mathematics. *Scientific American 260,* 13.

CWI SYLLABI

1 Vacantiecursus 1984: *Hewet - plus wiskunde.* 1984.

2 E.M. de Jager, H.G.J. Pijls (eds.). *Proceedings Seminar 1981-1982. Mathematical structures in field theories.* 1984.

3 W.C.M. Kallenberg, et al. *Testing statistical hypotheses: worked solutions.* 1984.

4 J.G. Verwer (ed.). *Colloquium topics in applied numerical analysis, volume 1.* 1984.

5 J.G. Verwer (ed.). *Colloquium topics in applied numerical analysis, volume 2.* 1984.

6 P.J.M. Bongaarts, J.N. Buur, E.A. de Kerf, R. Martini, H.G.J. Pijls, J.W. de Roever. *Proceedings Seminar 1982-1983. Mathematical structures in field theories.* 1985.

7 Vacantiecurus 1985: *Variatierekening.* 1985.

8 G.M. Tuynman. *Proceedings Seminar 1983-1985. Mathematical structures in field theories, Vol.1 Geometric quantization.* 1985.

9 J. van Leeuwen, J.K. Lenstra (eds.). *Parallel computers and computations.* 1985.

10 Vacantiecursus 1986: *Matrices.* 1986.

11 P.W.H. Lemmens. *Discrete wiskunde: tellen, grafen, spelen en codes.* 1986.

12 J. van de Lune. *An introduction to Tauberian theory: from Tauber to Wiener.* 1986.

13 G.M. Tuynman, M.J. Bergvelt, A.P.E. ten Kroode. *Proceedings Seminar 1983-1985. Mathematical structures in field theories, Vol.2.* 1987.

14 Vacantiecursus 1987: *De personal computer en de wiskunde op school.* 1987.

15 Vacantiecursus 1983: *Complexe getallen.* 1987.

16 P.J.M. Bongaarts, E.A. de Kerf, P.H.M. Kersten. *Proceedings Seminar 1984-1986. Mathematical structures in field theories, Vol.1.* 1988.

17 F. den Hollander, H. Maassen (eds.). *Mark Kac seminar on probability and physics. Syllabus 1985-1987.* 1988.

18 Vacantiecursus 1988. *Differentierekening.* 1988.

19 R. de Bruin, C.G. van der Laan, J.R. Luyten, H.F. Vogt. *Publiceren met LATEX.* 1988.

20 R. van der Horst, R.D. Gill (eds.). *STATAL: statistical procedures in Algol 60, part 1.* 1988.

21 R. van der Horst, R.D. Gill (eds.). *STATAL: statistical procedures in Algol 60, part 2.* 1988.

22 R. van der Horst, R.D. Gill (eds.). *STATAL: statistical procedures in Algol 60, part 3.* 1988.

23 J. van Mill, G.Y. Nieuwland (red.). *Proceedings van het symposium wiskunde en de computer.* 1989.

MC SYLLABI

1.1 F. Göbel, J. van de Lune. *Leergang besliskunde, deel 1: wiskundige basiskennis.* 1965.

1.2 J. Hemelrijk, J. Kriens. *Leergang besliskunde, deel 2: kansberekening.* 1965.

1.3 J. Hemelrijk, J. Kriens. *Leergang besliskunde, deel 3: statistiek.* 1966.

1.4 G. de Leve, W. Molenaar. *Leergang besliskunde, deel 4: Markovketens en wachttijden.* 1966.

1.5 J. Kriens, G. de Leve. *Leergang besliskunde, deel 5: inleiding tot de mathematische besliskunde.* 1966.

1.6a B. Dorhout, J. Kriens. *Leergang besliskunde, deel 6a: wiskundige programmering 1.* 1968.

1.6b B. Dorhout, J. Kriens, J.Th. van Lieshout. *Leergang besliskunde, deel 6b: wiskundige programmering 2.* 1977.

1.7a G. de Leve. *Leergang besliskunde, deel 7a: dynamische programmering 1.* 1968.

1.7b G. de Leve, H.C. Tijms. *Leergang besliskunde, deel 7b: dynamische programmering 2.* 1970.

1.7c G. de Leve, H.C. Tijms. *Leergang besliskunde, deel 7c: dynamische programmering 3.* 1971.

1.8 J. Kriens, F. Göbel, W. Molenaar. *Leergang besliskunde, deel 8: minimaxmethode, netwerkplanning, simulatie.* 1968.

2.1 G.J.R. Förch, P.J. van der Houwen, R.P. van de Riet. *Colloquium stabiliteit van differentieschema's, deel 1.* 1967.

2.2 L. Dekker, T.J. Dekker, P.J. van der Houwen, M.N. Spijker. *Colloquium stabiliteit van differentieschema's, deel 2.* 1968.

3.1 H.A. Lauwerier. *Randwaardeproblemen, deel 1.* 1967.

3.2 H.A. Lauwerier. *Randwaardeproblemen, deel 2.* 1968.

3.3 H.A. Lauwerier. *Randwaardeproblemen, deel 3.* 1968.

4 H.A. Lauwerier. *Representaties van groepen.* 1968.

5 J.H. van Lint, J.J. Seidel, P.C. Baayen. *Colloquium discrete wiskunde.* 1968.

6 K.K. Koksma. *Cursus ALGOL 60.* 1969.

7.1 *Colloquium moderne rekenmachines, deel 1.* 1969.

7.2 *Colloquium moderne rekenmachines, deel 2.* 1969.

8 H. Bavinck, J. Grasman. *Relaxatietrillingen.* 1969.

9.1 T.M.T. Coolen, G.J.R. Förch, E.M. de Jager, H.G.J. Pijls. *Colloquium elliptische differentiaalvergelijkingen, deel 1.* 1970.

9.2 W.P. van den Brink, T.M.T. Coolen, B. Dijkhuis, P.P.N. de Groen, P.J. van der Houwen, E.M. de Jager, N.M. Temme, R.J. de Vogelaere. *Colloquium elliptische differentiaalvergelijkingen, deel 2.* 1970.

10 J. Fabius, W.R. van Zwet. *Grondbegrippen van de waarschijnlijkheidsrekening.* 1970.

11 H. Bart, M.A. Kaashoek. H.G.J. Pijls, W.J. de Schipper, J. de Vries. *Colloquium halfalgebra's en positieve operatoren.* 1971.

12 T.J. Dekker. *Numerieke algebra.* 1971.

13 F.E.J. Kruseman Aretz. *Programmeren voor rekenautomaten; de MC ALGOL 60 vertaler voor de EL X8.* 1971.

14 H. Bavinck, W. Gautschi, G.M. Willems. *Colloquium approximatietheorie.* 1971.

15.1 T.J. Dekker, P.W. Hemker, P.J. van der Houwen. *Colloquium stijve differentiaalvergelijkingen, deel 1.* 1972.

15.2 P.A. Beentjes, K. Dekker, H.C. Hemker, S.P.N. van Kampen, G.M. Willems. *Colloquium stijve differentiaalvergelijkingen, deel 2.* 1973.

15.3 P.A. Beentjes, K. Dekker, P.W. Hemker, M. van Veldhuizen. *Colloquium stijve differentiaalvergelijkingen, deel 3.* 1975.

16.1 L. Geurts. *Cursus programmeren, deel 1: de elementen van het programmeren.* 1973.

16.2 L. Geurts. *Cursus programmeren, deel 2: de programmeertaal ALGOL 60.* 1973.

17.1 P.S. Stobbe. *Lineaire algebra, deel 1.* 1973.

17.2 P.S. Stobbe. *Lineaire algebra, deel 2.* 1973.

17.3 N.M. Temme. *Lineaire algebra, deel 3.* 1976.

18 F. van der Blij, H. Freudenthal, J.J. de Iongh, J.J. Seidel, A. van Wijngaarden. *Een kwart eeuw wiskunde 1946-1971, syllabus van de vakantiecursus 1971.* 1973.

19 A. Hordijk, R. Potharst, J.Th. Runnenburg. *Optimaal stoppen van Markovketens.* 1973.

20 T.M.T. Coolen, P.W. Hemker, P.J. van der Houwen, E. Slagt. *ALGOL 60 procedures voor begin- en randwaardeproblemen.* 1976.

21 J.W. de Bakker (red.). *Colloquium programmacorrectheid.* 1975.

22 R. Helmers, J. Oosterhoff, F.H. Ruymgaart, M.C.A. van Zuylen. *Asymptotische methoden in de toetsingstheorie; toepassingen van naburigheid.* 1976.

23.1 J.W. de Roever (red.). *Colloquium onderwerpen uit de biomathematica, deel 1.* 1976.

23.2 J.W. de Roever (red.). *Colloquium onderwerpen uit de biomathematica, deel 2.* 1977.

24.1 P.J. van der Houwen. *Numerieke integratie van differentiaalvergelijkingen, deel 1: eenstapsmethoden.* 1974.

25 *Colloquium structuur van programmeertalen.* 1976.

26.1 N.M. Temme (ed.). *Nonlinear analysis, volume 1.* 1976.

26.2 N.M. Temme (ed.). *Nonlinear analysis, volume 2.* 1976.

27 M. Bakker, P.W. Hemker, P.J. van der Houwen, S.J. Polak, M. van Veldhuizen. *Colloquium discretiseringsmethoden.* 1976.

28 O. Diekmann, N.M. Temme (eds.). *Nonlinear diffusion problems.* 1976.

29.1 J.C.P. Bus (red.). *Colloquium numerieke programmatuur, deel 1A, deel 1B.* 1976.

29.2 H.J.J. te Riele (red.). *Colloquium numerieke programmatuur, deel 2.* 1977.

30 J. Heering, P. Klint (red.). *Colloquium programmeeromgevingen.* 1983.

31 J.H. van Lint (red.). *Inleiding in de coderingstheorie.* 1976.

32 L. Geurts (red.). *Colloquium bedrijfsystemen.* 1976.

33 P.J. van der Houwen. *Berekening van waterstanden in zeeën en rivieren.* 1977.

34 J. Hemelrijk. *Oriënterende cursus mathematische statistiek.* 1977.

35 P.J.W. ten Hagen (red.). *Colloquium computer graphics.* 1978.

36 J.M. Aarts, J. de Vries. *Colloquium topologische dynamische systemen.* 1977.

37 J.C. van Vliet (red.). *Colloquium capita datastructuren.* 1978.

38.1 T.H. Koornwinder (ed.). *Representations of locally compact groups with applications, part I.* 1979.

38.2 T.H. Koornwinder (ed.). *Representations of locally compact groups with applications, part II.* 1979.

39 O.J. Vrieze, G.L. Wanrooy. *Colloquium stochastische spelen.* 1978.

40 J. van Tiel. *Convexe analyse.* 1979.

41 H.J.J. te Riele (ed.). *Colloquium numerical treatment of integral equations.* 1979.

42 J.C. van Vliet (red.). *Colloquium capita implementatie van programmeertalen.* 1980.

43 A.M. Cohen, H.A. Wilbrink. *Eindige groepen (een inleidende cursus).* 1980.

44 J.G. Verwer (ed.). *Colloquium numerical solution of partial differential equations.* 1980.

45 P. Klint (red.). *Colloquium hogere programmeertalen en computerarchitectuur.* 1980.

46.1 P.M.G. Apers (red.). *Colloquium databankorganisatie, deel 1.* 1981.

46.2 P.G.M. Apers (red.). *Colloquium databankorganisatie, deel 2.* 1981.

47.1 P.W. Hemker (ed.). *NUMAL, numerical procedures in ALGOL 60: general information and indices.* 1981.

47.2 P.W. Hemker (ed.). *NUMAL, numerical procedures in ALGOL 60, vol. 1: elementary procedures; vol. 2: algebraic evaluations.* 1981.

47.3 P.W. Hemker (ed.). *NUMAL, numerical procedures in ALGOL 60, vol. 3A: linear algebra, part I.* 1981.

47.4 P.W. Hemker (ed.). *NUMAL, numerical procedures in ALGOL 60, vol. 3B: linear algebra, part II.* 1981.

47.5 P.W. Hemker (ed.). *NUMAL, numerical procedures in ALGOL 60, vol. 4: analytical evaluations; vol. 5A: analytical problems, part I.* 1981.

47.6 P.W. Hemker (ed.). *NUMAL, numerical procedures in ALGOL 60, vol. 5B: analytical problems, part II.* 1981.

47.7 P.W. Hemker (ed.). *NUMAL, numerical procedures in ALGOL 60, vol. 6: special functions and constants; vol. 7: interpolation and approximation.* 1981.

48.1 P.M.B. Vitányi, J. van Leeuwen, P. van Emde Boas (red.). *Colloquium complexiteit en algoritmen, deel 1.* 1982.

48.2 P.M.B. Vitányi, J. van Leeuwen, P. van Emde Boas (red.). *Colloquium complexiteit en algoritmen, deel 2.* 1982.

49 T.H. Koornwinder (ed.). *The structure of real semisimple Lie groups.* 1982.

50 H. Nijmeijer. *Inleiding systeemtheorie.* 1982.

51 P.J. Hoogendoorn (red.). *Cursus cryptografie.* 1983.