DE SIMULATIETAAL COSMOS

D.L. Kettenis

In dit artikel wordt de simulatietaal COSMOS beschreven. Dit is een taal die geschikt is voor het beschrijven van continue, discrete en gecombineerde modellen. COSMOS maakt het mogelijk de formulering van het model en dat van de experimenten onafhankelijk van elkaar te geven. Hierdoor is het mogelijk modellen zonder experimenteerdetails te beschrijven, waardoor hergebruik van modellen in bijvoorbeeld een groter model mogelijk wordt. De taalelementen zijn zo gekozen, dat duidelijke en goed overdraagbare modellen ontstaan.
Voor wat continue modellen betreft, is het mogelijk discontinue overgangen te beschrijven. Aan de afhandeling van deze discontinuïteiten tijdens een simulatie experiment wordt zonder speciale aandacht van de modelmaker de grootste zorg besteed.
Voor de discrete simulatiemogelijkheden is gekozen voor de procesinteractie. Doordat COSMOS zowel discrete als continue submodellen toestaat, is een optimale modelkeuze mogelijk.

Inleiding

Zoals in het inleidende artikel over simulatie is weergegeven, wordt er bij simulatie een model gebouwd en experimenteert men met dat model. Bij computer simulatie wordt het model gevormd door een stel instructies. De instructies moeten aan de computer worden medegedeeld, waarvoor een aantal hulpmiddelen ter beschikking staat. Veelal past men een algemene programmeertaal, zoals Fortran of Basic, toe bij simulatie. Ook zijn er talen beschikbaar die specifiek bedoeld zijn voor simulatie. In het landbouwkundig onderzoek zijn daartoe onder andere CSMP en SIMULA in gebruik. Welke taal er wordt gekozen, hangt van een aantal factoren af, zoals beschikbaarheid. Er is een aantal redenen, dat pleit voor het gebruik van een simulatietaal boven een programmeertaal. Deze redenen zijn:

- de structuur van het model kan beter in overeenstemming gebracht worden met de structuur van het gemodelleerde systeem in werkelijkheid;
- de maker van het model kan zich op het model concentreren en behoeft zich niet bezig te houden met allerlei implementatie details;
- verzamelen van gegevens en weergeven van die gegevens is comfortabel te verwezenlijken;
- de modelmaker is vaak een materiedeskundige en geen professioneel programmeur. Ter voorkoming van communicatiestoornissen tussen de maker van het model en een programmeur zal de maker van het model in het algemeen zelf het computerprogramma schrijven, testen en onderhouden;
- in een simulatietaal is het mogelijk het model compacter en, mede daardoor, overzichtelijker te formuleren. Het eerste punt uit deze opsomming speelt daarbij ook een rol. Beide punten hebben tot gevolg dat een model in een simulatietaal geformuleerd beter onderhoudbaar is;
- een model geformuleerd in een simulatietaal is beter te begrijpen door andere wetenschappers dan door de maker zelf. Hierdoor is het eenvoudiger concepten over te dragen.

De uiteindelijke keuze van de te gebruiken taal zal altijd wel een kwestie van smaak blijven, of misschien veel erger, van het toeval afhangen. In het vervolg zal een aantal elementen uit de taal COSMOS beschreven worden. COSMOS is een taal die ontwikkeld is bij de vakgroep Informatica van de Landbouwuniversiteit. De naam is een acronym van COmbined System and MOdelling and Simulation.

COSMOS is geïmplementeerd op een VAX onder VMS en op PC/AT en hoger onder het MS/DOS operating systeem. Daar het gehele systeem is gemaakt op basis van de algemene programmeertaal C die goed overdraagbaar is, is het zeer goed mogelijk het systeem te implementeren op een groot aantal merken computers en besturingssystemen, zoals werkstations. In het najaar is COSMOS gebruikt in de cursus simulatietechniek van de vakgroep Informatica. Dit gebeurt in het kader van de beta test. Verwacht wordt dat de testfase op korte termijn afgesloten kan worden.

Simulatietalen

In de simulatie zijn er twee wereldbeelden: een continu en een discreet. Bij continue modellen worden de toestandsveranderingen beschreven door middel van differentiaal- of gewone vergelijkingen. De toestandsveranderingen in een discreet model zijn discreet - er is sprake van een plotselinge verandering - en vinden ook op discrete momenten plaats. De twee werelden hebben hun eigen simulatietalen naar voren gebracht. De laatste tijd zijn er hulpmiddelen gekomen om gecombineerde modellen te beschrijven. Dat is een zeer belangrijke ontwikkeling, omdat het niet altijd duidelijk is of men een systeem uit de werkelijkheid nu moet afbeelden op een continu of op een discreet model. Bovendien zal het vaak zo zijn dat men bepaalde deelprocessen graag continu wil afbeelden en andere deelprocessen discreet. Veel van de hulpmiddelen die ter beschikking staan voor gecombineerde modellen komen uit de discrete hoek en staan helaas niet toe continue modellen op comfortabele wijze te beschrijven en/of schieten te kort voor wat betreft numerieke integratiealgoritmen.

COSMOS is een nieuwe simulatietaal, die specifiek is ontwikkeld voor het beschrijven van continue, discrete en gecombineerde modellen. De taalelementen in COSMOS zijn zo gekozen dat duidelijke en goed onderhoudbare pro-

gramma's geschreven kunnen worden. Een van de belangrijkste aspecten daarbij is de scheiding van formulering van model en experiment. Om een goed onderhoudbaar model te maken, is het van het grootste belang dat het model in structuur in overeenstemming is met de werkelijkheid. In werkelijkheid zal men in het algemeen aan een systeem niet ontdekken dat er, in het systeem zelf, voorzieningen zijn voor het verzamelen van gegevens om later een plot of een statistisch overzicht te maken. Daarom horen statements en datastructuren om dat soort zaken te doen niet thuis in een modelbeschrijving. In een model hoort ook geen oplossingstechniek, zoals een numeriek integratieproces of integratiestap, geplaatst te worden. In de figuren 1, 2 en 3 worden COSMOS programma's gegeven, zodat een beeld gevormd kan worden van de mogelijkheden van deze taal.

Continue modellen

Continue modellen worden veelal toegepast in vakgebieden zoals scheikunde, natuurkunde en regeltechniek. Ook in het landbouwkundig onderzoek wordt dit soort modellen zeer uitgebreid gebruikt. De basis van veel continue modellen is een differentiaalvergelijking. In COSMOS is het alleen mogelijk zogenaamde eerste orde differentiaalvergelijkingen te beschrijven. Dit beperkt de toepasbaarheid van de taal geenszins, omdat het altijd mogelijk is een hogere orde differentiaalvergelijking om te zetten in een stelsel van eerste orde differentiaalvergelijkingen. In COSMOS wordt de differentiaalvergelijking:

```
dv/dt = -g
```

geformuleerd als:

```
v':= -g;
```

Zoals vermoedelijk wel bekend is, is een eerste orde differentiaalvergelijking slechts compleet oplosbaar als er een beginconditie gegeven is. De beginvoorwaarde behoort echter tot de experimentele gegevens, het model legt alleen het dynamisch gedrag vast. Dus in COSMOS ontbreekt de beginconditie in het model, in tegenstelling tot andere simulatietalen waar die voorwaarde een intrinsiek onderdeel is van de modelbeschrijving. De beginvoorwaarde wordt in COSMOS gegeven in de beschrijving van het experiment, waardoor het model in meerdere experimenteeromgevingen bruikbaar is.

De modelwereld blijkt meestal niet volledig continu. Het is zelfs mogelijk dat door het model enigszins te vereenvoudigen een discontinuïteit ontstaat die in het werkelijke systeem niet aanwezig is.

Zo'n discontinue overgang kan optreden bij de luchtstroming langs een druppel water. De stroming kan zich netjes langs de druppel bewegen en wordt dan laminair genoemd. De stroming kan ook wervelingen vertonen en loslaten van de druppel, in dat geval is de stroming turbulent. De weerstand die een druppel ondervindt van de luchtstroming is afhankelijk van het Reynolds getal.

De luchtweerstandscoëfficiënt (cw) kan in COSMOS als volgt beschreven worden:

```
IF reynolds < 2.0
THEN cw:= reynolds/24;
          (*laminaire stroming*)
ELSE cw:= 18.5/reynolds ** 0.6
          (*turbulente stroming*)
END IF;
```

Een andere vorm van een discontinuïteit treedt op bij een stuiterende bal. Op het moment dat de bal de grond raakt, zal een deel van de energie worden omgezet in warmte en de richting van de snelheid wordt omgedraaid. De manier waarop dat in COSMOS kan worden beschreven, staat in figuur 1.

Een van de belangrijkste elementen om een programma onderhoudbaar te maken, is het op te delen in een aantal modulen. Dat is ook zeer sterk geldig voor modellen. In COSMOS is het mogelijk een model op te delen in submodellen door dat submodel te declareren als een procestype. Een procestype is niets anders dan een definitie, een actieve kopie zal expliciet aangemaakt moeten worden. Het is mogelijk dat van een procestype meerdere versies tegelijk in het model aanwezig zijn. Ook kan men gedurende een simulatie-experiment elementen van zo'n proces aanmaken of weer verwijderen uit de simulatie. De in het model aanwezige actieve processen kunnen met elkaar communiceren. Hierop wordt in dit artikel niet verder ingegaan.

```
PROGRAM bouncing_ball IS
  STATE v (* vertical velocity *),
        h (* height *);
  CONSTANT
    REAL g (* gravity *) := 32.2,
         kr := 0.7
         (* coefficient of restitution *)
  SYSTEM
    CONTINUOUS
      v' := -g;
      h' :=  v;
      AS SOON AS h<0 v:= -kr * v; END AS;
    END CONTINUOUS;
  END SYSTEM;
  EXPERIMENT
    INITIALIZE STATE v := 0; h:= 2.0;
    PLOT BY PLOTTER h VERSUS time;
    INTEGRATE BY RUNGE KUTTA;
    SIMULATE UNTIL (h<=0) AND (v<=0.001);
  END EXPERIMENT;
END PROGRAM;
```

Figuur 1: Stuiterende bal

Discrete modellen

Discrete modellen worden veelal toegepast bij organisatorische vraagstukken, zoals die voorkomen in de bedrijfskunde en operations research. Een produktiestraat, zoals een slachtlijn, is een voorbeeld daarvan. In discrete modellen beschrijft men de gebeurtenissen die de toestand van het model veranderen en de momenten waarop die gebeurtenissen moeten plaatsvinden. De gebeurtenissen moeten op de een of andere manier gegroepeerd worden in modulen in het model. Dat kan op vele manieren gebeuren. Eén van de mogelijkheden is alle gebeurtenissen die op één tijdstip moeten plaatsvinden, in een module te plaatsen. Deze methode staat bekend als de event georiënteerde beschrijvingswijze. Een andere manier is alle gebeurtenissen die betrekking hebben op één component in één model module weer te geven. Deze beschrijvingswijze wordt proces interactie genoemd. Omdat op de proces georiënteerde wijze alle gebeurtenissen die betrekking hebben op één component in één module in het model terechtkomen is het programma overzichtelijk en goed in overeenstemming met het werkelijke systeem. Bij COSMOS is voor deze aanpak gekozen. De basiselementen voor discrete simulatie zijn de discrete processen, eerder is reeds de continue tegenhanger beschreven. Ook een discreet procestype is een declaratie, er moet een actieve versie van aangemaakt worden. Ook van dit procestype kunnen meerdere kopieën gemaakt worden en kunnen actieve kopieën weer uit de simulatie verwijderd worden.

Voor dit artikel voert het te ver om alle mogelijkheden voor discrete simulatie te behandelen. Er zal slechts een summier overzicht gegeven worden, iets dat ook voor de continue mogelijkheden van COSMOS geldt. In discrete simulatietoepassingen spelen stochastische variabelen een grote rol. Een stochastische variabele wordt in COSMOS weergegeven in een random variabele. In het experiment wordt aangegeven uit welke verdelingsfunctie de random getallen getrokken moeten worden. Een andere, in discrete simulatie veel gebruikte, variabele is de wachtrij. In COSMOS zijn er verschillende soorten wachtrijen aanwezig. De verschillen worden bepaald door de prioriteitsregels. In COSMOS zijn dat first-in-first-out, last-in-first-out en twee varianten waarbij de prioriteiten afhangen van de waarde van een variabele van de componenten, die in een wachtrij geplaatst kunnen worden.

Naast de bij de talen voor de procesinteractie beschrijvingswijze gebruikelijke activeer- en passiveer-opdrachten, staat er in COSMOS een WAIT UNTIL contructie ter beschikking. Men kan op die manier een proces laten wachten totdat er aan een bepaalde conditie is voldaan. Een andere belangrijke constructie is het interrupt mechanisme, met prioriteiten afhandeling.

Gecombineerde modellen

In het voorgaande is een beknopt overzicht gegeven van de taalelementen die beschikbaar zijn voor continue en discrete simulatie. Door nu een deel van het model weer te geven door middel van een continu proces, en een ander deel door middel van een discreet proces, kan een gecombineerd model worden gebouwd. Hierdoor kan men een optimale keuze maken welke onderdelen van het model continu en welke onderdelen discreet gemodelleerd worden.

Zoals reeds eerder opgemerkt is, is het mogelijk van de processen op ieder gewenst moment een kopie te maken of deze uit de simulatie te verwijderen. Hierdoor wordt het mogelijk een model te vervaardigen met een variabele structuur. Discrete simulatietalen hebben meestal al de mogelijkheid modellen van variabele structuur te maken, maar continue simulatietalen gaan er meestal van uit dat een model een vaste structuur heeft! Een bekend industrieel voorbeeld van een model met een variabele structuur is een oven die in een staalfabriek zogenaamde ijzerbroodjes opwarmt. Als de capaciteit van de oven te klein is om het aantal aankomende ijzerbroodjes te verwarmen, zullen de ijzerbroodjes die geen plaats vinden in de oven afkoelen. Als in het model de temperatuursverandering van de ijzerbroodjes wordt beschreven door middel van een differentiaalvergelijking, dan ontstaat een gecombineerd model, waarin het totaal aantal differentiaalvergelijkingen dat actief is in het model varieert afhankelijk van het aantal ijzerbroodjes dat wacht voor de oven en in de oven aanwezig is. De formulering van dit probleem geeft een goede

```
PROGRAM harvesting IS
  STATE biom;
  REAL harvst;
  PARAMETER REAL rgr,biohst;

  DISCRETE PROCESS harvest IS
    DISCRETE
      harvst := biom;
      biom := 0.0;
    END DISCRETE;
  END PROCESS;

  SYSTEM
    INITIAL
      CREATE harvest;
    END INITIAL;
    CONTINUOUS
      biom' := rgr *biom;
      AS SOON AS biom >= biohst DO
        ACTIVATE harvest;
      END AS;
    END CONTINUOUS;
  END SYSTEM;

  EXPERIMENT
    PARAMETER rgr := 0.2;biohst := 190.0;
    INITIALIZE STATE biom := 100.0;
    PLOT BY PRINTER biom,harvst VERSUS
time;
    PLOT COMMUNICATION INTERVAL := 0.2;
    INTEGRATE BY RUNGE KUTTA;
    SIMULATE UNTIL time = 3.8;
```

Figuur 2: Oogsten

indruk van de mogelijkheden van een gecombineerde simulatietaal, de afmetingen van de beschrijving is echter te groot om in dit artikel op te nemen. Belangstellende kunnen deze beschrijving opvragen bij de auteur.

In figuur 2 is een wat kleiner gecombineerd model gegeven. Hierin wordt de groei van een gewas beschreven, weergegeven door de variabele biom, dat nadat de biomassa een bepaalde waarde heeft bereikt, geoogst moet worden. Het oogsten wordt in figuur 2 door een discreet proces gedaan, waarvan de activering vanuit het continue model wordt verzorgd.

Experiment beschrijving

In het voorgaande is reeds gesteld dat het zeer belangrijk is model- en experimentbeschrijving gescheiden te houden. Eén van de redenen daarvoor die nog niet genoemd is, is dat het op die manier mogelijk is modellen aan verschillende experimenten te onderwerpen. Daarenboven kan men op deze wijze modellen hergebruiken in andere modellen. In figuur 3 wordt daarvan een voorbeeld gegeven. Op dit voorbeeld zal later nog worden teruggekomen.

In het experiment beschrijft men de begincondities voor een model. Tevens worden parameterwaarden gegeven en wordt gespecificeerd uit welke verdelingsfunctie een stochastische variabele getrokken moet worden. Daarnaast wordt gespecificeerd van welke variabelen men een plot of een histogram wenst, en van welke variabelen men statistische gegevens wenst te verzamelen. Indien het model continue (sub)modellen bevat, kan het numerieke integratie algoritme waarmee de differentiaalvergelijkingen opgelost dienen te worden, gespecificeerd worden.

Er zijn voorzieningen aanwezig om op comfortabele wijze simulatieruns uit te voeren voor een groot aantal parameterwaarden. Daarnaast is het mogelijk een herhaling van simulatieruns zelf te programmeren, zodat bijvoorbeeld iteratief een zogenaamd randvoorwaardenprobleem opgelost kan worden. Een voorbeeld van een randvoorwaardenprobleem staat beschreven in figuur 3. Het experiment beschreven in figuur 3 bestaat uit 2 verschillende experimenteeromgevingen, frame 1 en frame 2. Het eerste frame beantwoordt de vraag waar de kogel die wordt weggeschoten terecht komt. Het tweede frame beschrijft een experiment dat voor artilleristen wat interessanter is, namelijk met welke snelheid moet de kogel worden weggeschoten, zodat een bepaald doel wordt bereikt. Een differentiaalvergelijking alleen bepaalt een hele grote verzameling oplossingen. Uit deze verzameling wordt er 1 gekozen op grond van condities. Die condities kunnen gegeven zijn aan het begin van de simulatie (beginwaarden probleem), aan het einde van de simulatie (eindwaarden probleem) of een mengvorm daarvan (randwaarden probleem). In het algemeen is het moeilijk direct een randwaarden probleem op te lossen; een randvoorwaarden probleem wordt meestal door middel van een iteratie teruggebracht tot een beginwaarde probleem. In dit geval gebeurt dat door een schatting te maken voor de ontbrekende beginsnelheid, een simulatie run te maken, te zien waar de kogel terecht komt en, als het doel nog niet dicht genoeg genaderd is, een nieuwe schatting te maken voor de beginsnelheid.

```
PROGRAM missile_trajectory IS
STATE x,dy,y;
REAL alphas,v0;
CONSTANT REAL g:= 10.0;
SYSTEM
  CONTINUOUS
    x':= v0*COS(alphas);
    dy':= -g;
    y' := dy;
  END CONTINUOUS;
END SYSTEM;

EXPERIMENT
REAL af,eps,k,distance,v0g;
  TYPE phase IS first_run,other_run;
  phase exp_phase;
  FRAME 1
    INITIALIZE STATE y,x := 0.0;
    SIMULATE UNTIL (y<=0.0) AND (dy<0);
    PRERUN
      alphas := ARCTAN(1.0);
      v0 := 10.0;
      dy := v0 * SIN(alphas);
    END PRERUN;
    POSTRUN WRITE "THE DISTANCE" x;
    END POSTRUN;
  END FRAME 1;

  FRAME 2
    INITIALIZE STATE y,x:=0.0;
    SIMULATE UNTIL (y<=0.0) AND (dy<0.0);
    PREFRAME
      exp_phase := first_run;
      af:= 500; eps := 0.01;k:= 0.1;
    END PREFRAME;

    PRERUN
      CASE exp_phase OF
        WHEN first_run THEN
          v0g := 10.0;
exp_phase:=other_run;
        WHEN other_run THEN
          v0g := v0g - k*(distance-af);
      END CASE;
      alphas := ARCTAN(1.0);
      dy:=v0g*SIN(alphas);
      v0:= v0g;
    END PRERUN;

    POSTRUN
      IF ABS(x-af) < eps THEN
        WRITE "THE REQUESTE VELOCITY" v0;
        FINISH;
      ELSE RERUN;
           distance := x;
      END IF;
    END POSTRUN;
  END FRAME 2;
```

Figuur 3: Kogelbaan

De bekende parameter-optimalisatiemodule ontwikkeld door L.G. Birta voor bijvoorbeeld CSMP, is in COSMOS aanwezig, zodat ook optimalisatie studies gemaakt kunnen worden. Bij optimalisatie is er meestal sprake van het optimaliseren van een kostfunctie. De berekening van de kostfunctie is in wezen een zaak die niet in het model thuishoort. Daarom zijn er in COSMOS voorzieningen getroffen om de berekening van de kostfunctie in het experiment te specificeren, waardoor het model ongewijzigd kan blijven.

Tot slot

COSMOS is ontwikkeld om op eenvoudige wijze modellen te specificeren, zodat de modellen onderhoudbaar zijn. Zoals uit de in dit artikel gegeven voorbeelden blijkt, is het mogelijk zeer duidelijke modellen te formuleren. De kracht van COSMOS zal echter pas duidelijk blijken bij toepassing op complexe systemen. De constructies zijn zo gekozen dat het vervaardigen van betrouwbare modellen wordt gestimuleerd. ❑

Ir. D.L. Kettenis is universitair hoofddocent bij de vakgroep Informatica van de Landbouwuniversiteit, Dreijenplein 2, 6703 HB te Wageningen, tel. 08370 - 83773.