Sharon Calor
Faculteit Onderwijs en Opvoeding
Hogeschool van Amsterdam
s.m.calor@hva.nl

Sonia Abrantes Garcêz Palha
Faculteit Onderwijs en Opvoeding
Hogeschool van Amsterdam
s.abrantes.garcez.palha@hva.nl

Laura Kubbe
Faculteit Onderwijs en Opvoeding
Hogeschool van Amsterdam
l.kubbe@uva.nl

Onderwijs

Studenten denken algoritmisch bij Dynamische Systemen

Dynamische Systemen is een van de vakken van het wiskundeprogramma van de eerstegraadslerarenopleiding wiskunde van de Hogeschool van Amsterdam. De opleiding wil graag meer aandacht geven aan ICT in het wiskundeonderwijs, algoritmisch denken en programmeren, ook vanwege de toegenomen aandacht voor digitale geletterdheid in het voortgezet onderwijs. Het vak Dynamische Systemen leent zich hier goed voor als het gaat om numerieke benaderingen van oplossingen van differentiaalvergelijkingen. In dit artikel beschrijven docenten Sharon Calor, Sonia Abrantes Garcêz Palha en Laura Kubbe een ontwerp van een aantal opdrachten waarmee getracht is algoritmisch denken bij studenten te ontwikkelen. Om in kaart te kunnen brengen hoe studenten algoritmisch denken, is een coderingschema gemaakt en toegepast bij hardop denkende studenten.

Digitale geletterdheid krijgt de laatste tijd veel aandacht in het voortgezet onderwijs. De steeds voortgaande digitalisering van de maatschappij geeft aanleiding tot nadenken over wat leerlingen op school zouden moeten leren op het gebied van ICT. De plannen voor herzieningen van de programma's in het voortgezet onderwijs, waaronder de voorstellen van Curriculum.nu uit 2019, laten dit ook zien. Een onderdeel van deze voorstellen is de formulering van een leergebied *Digitale geletterdheid*. Dit gebied wordt onderverdeeld in vier afzonderlijke domeinen: ICT-basisvaardigheden, Informatievaardigheden, Mediawijsheid en Computational Thinking (CT). Om meer aandacht te kunnen geven aan ICT en onze studenten beter te kunnen voorbereiden op de ontwikkelingen in het onderwijs, hebben we bij de masteropleiding leraar wiskunde van de Hogeschool van Amsterdam (HvA) lesmateriaal bij het vak Dynamische Systemen ontwikkeld dat toegespitst is op het domein CT.

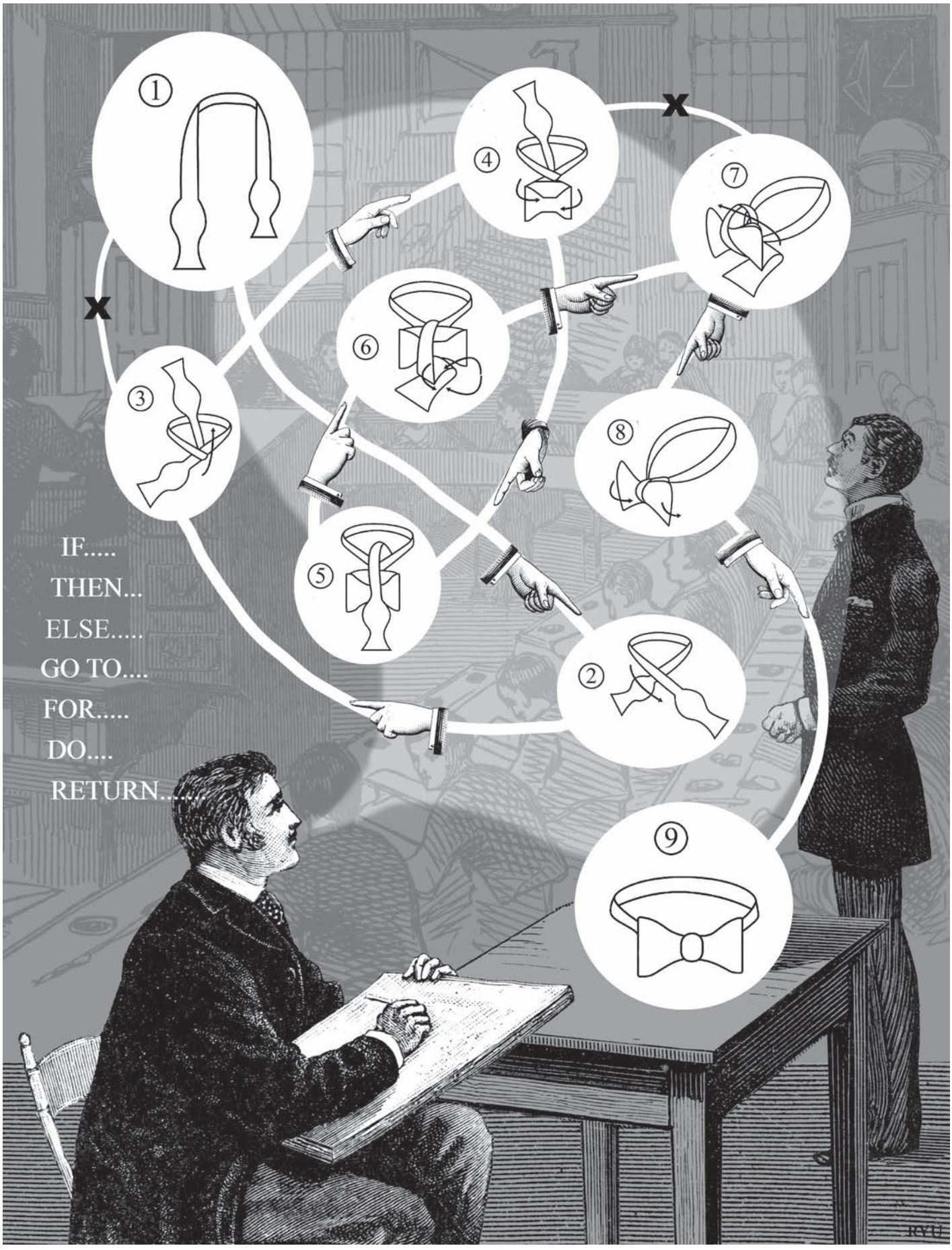
De masteropleiding leraar wiskunde van de HvA is bedoeld voor tweedegraadswiskundeleraren die hun eerstegraadsbevoegdheid willen behalen en in de bovenbouw van het voortgezet onderwijs willen lesgeven. De opleiding wordt in deeltijd aangeboden en studenten combineren de opleiding met hun baan op school. Het curriculum bevat (vak)didactische en pedagogische onderdelen en een praktijkdeel. Daarnaast krijgen de studenten een wiskundeprogramma met vakken als analyse, meetkunde en statistiek. Het vak Dynamische Systemen staat ook op het programma. Studenten leren bij dit vak verschillende gewone differentiaalvergelijkingen op te lossen en toe te passen bij modelleerproblemen. Daarnaast krijgen ze een onderdeel voor verdere verdieping. De opleiding kan daarbij kiezen uit een aantal onderwerpen, zoals *chaostheorie en fractalen* en *numerieke methoden* [6]. Tot voor kort was het eerste onderwerp onderdeel van het vak, maar om meer aandacht te geven aan CT, algoritmes en ICT, zijn we ons gaan richten op numerieke methoden.

Er is tot nog toe geen consensus in de literatuur te vinden over wat CT is. De meest geciteerde definitie van CT is die van Wing [5]: "The thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent." Over het algemeen wordt CT geassocieerd met abstractie, generalisatie, decompositie, algoritmes en debugging [1, 4]. Bij algoritmes gaat het om de vaardigheid van het bedenken van een verzameling stap-voor-stapbewerkingen of acties om een probleem te kunnen oplossen [3]. Daarbij onderscheidt men twee onderdelen:

1. *Sequentie:* de vaardigheid om acties in de correcte volgorde te plaatsen.
2. *Flow of Control:* de manier waarop instructies/acties in volgorde worden uitgevoerd [3].

Algoritmisch Denken (AD) is in deze optiek een onderdeel van CT, waarbij het gaat om het formuleren van oplossingen van wiskundige problemen in een stappenplan. Curriculum.nu vraagt zelfs specifiek aandacht voor AD bij wiskunde in het voortgezet onderwijs [7] en noemt het belang van het hebben van enige kennis over de talrijke benaderingstechnieken binnen numerieke wiskunde.

Het vermogen om algoritmisch te kunnen denken is vanzelfsprekend belangrijk



Illustratie: Ryu Tajiri

bij wiskunde, maar de richting van de ontwikkelingen van een nieuw programma voor de middelbare school geeft aan dat het zinvol is om aankomende docenten wiskunde toe te rusten op het gebied van AD. Door ons te richten op numerieke methoden bij Dynamische Systemen vonden we een natuurlijke manier om AD een plek te geven binnen de opleiding. Het geeft ook aanleiding om aandacht te geven aan programmeren, een belangrijk aspect van AD en iets waar onze studenten weinig tot geen ervaring mee hebben. We denken dat AD bovendien het begrip van differentiaalvergelijkingen kan versterken.

Bij het ontwerpen van dit onderdeel hebben we ons niet alleen beziggehouden met de vraag hoe we AD vorm kunnen geven, maar ook of AD daadwerkelijk is te herkennen bij onze studenten als zij het onderwerp numerieke methoden bestuderen.

Algoritmisch denken

In het college Dynamische Systemen leren studenten verschillende soorten differentiaalvergelijkingen exact op te lossen. Met het programma GeoGebra kunnen ze richtingsvelden en oplossingen tekenen van differentiaalvergelijkingen van de eerste orde. Bij bijvoorbeeld het prooi- en roofdiermodel van Lotka–Volterra lukt dat exact oplossen echter niet, maar de oplossingskrommen kunnen met behulp van GeoGebra en computerprogramma's numeriek worden benaderd. Van richtingsvelden en het idee van een lineaire benadering van een functie is het een kleine stap naar de methode van Euler. Voor het beginwaardeprobleem

$$f(x,y) = \frac{dy}{dx}, \quad y(x_0) = y_0,$$

geeft de methode van Euler de bekende formules

$$x_{n+1} = x_n + h$$

en

$$y_{n+1} = y_n + h \cdot f(x_n, y_n).$$

Het bijbehorende algoritme geeft aanleiding om de studenten kennis te laten maken met eenvoudige algoritmen en manieren om deze op te schrijven. Daarna ligt het voor de hand om naar betere benaderingsmethoden te kijken en verdere stappen te nemen richting programmeren.

Bij het onderwijzen van numerieke methoden om de oplossing van differentiaalvergelijkingen te benaderen is het van belang dat studenten de numerieke algoritmes (en wiskundige concepten zoals complexiteit, stabiliteit en convergentie) begrijpen. Studenten moeten ook de numerieke algoritmes kunnen implementeren met een programma en het programma kunnen testen en debuggen. Met andere woorden, studenten moeten de CT-vaardigheden leren, in het bijzonder algoritmisch leren denken, om van een differentiaalvergelijking naar een betrouwbare benaderde oplossing te komen.

Categorieën in algoritmisch denken

Om meer inzicht te krijgen in AD in de context van het vak Dynamische Systemen, hebben we ons gericht op twee onderdelen van AD: het algoritme zelf (in programmeertaal of in *pseudocode*) en de *Flow of Control* [2]. Pseudocode is een manier om een algoritme in woorden te omschrijven, een soort representatie tussen het algoritme en de programmeercode in. Flow of Control is de wijze waarop commando's in volgorde worden uitgevoerd. We spreken ook van volgorde als er lussen in het algoritme voorkomen. Voor de analyse van de redeneringen van studenten bij het maken van een aantal opdrachten, hebben we een codeerschema gemaakt dat hiervan uitgaat. Het schema bestaat uit twee categorieën die verwijzen naar de twee hoofdaspecten van AD: *Algoritme en Pseudocode* en *Algoritme Flow of Control* met daarbij een aantal subcategorieën. Zie Figuur 1.

Algoritme en Pseudocode

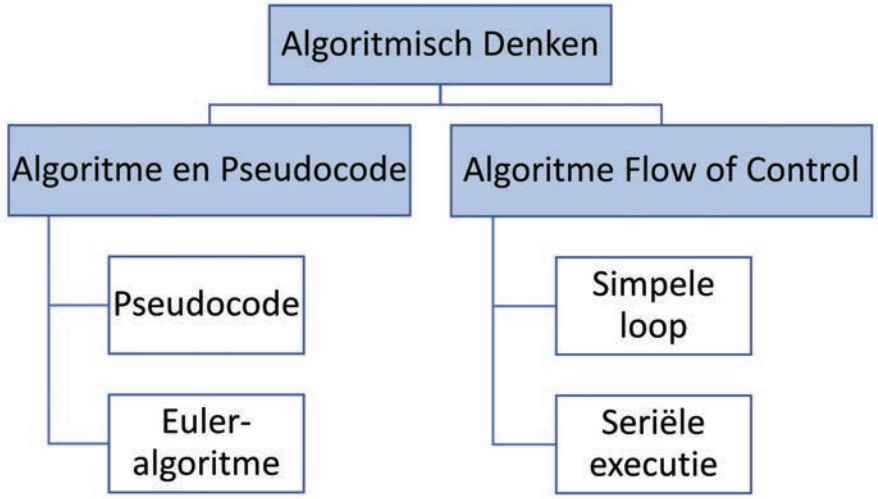
De categorie *Algoritme en Pseudocode* bestaat uit de subcategorieën: *Pseudocode* en *Euleralgoritme*. Bij Pseudocode laat de student zien dat hij/zij het Euleralgoritme in pseudocode kan beschrijven zoals hieronder:

Herhaal n maal
  $d = f(x,y)$
  $y = y + h \cdot d$
  $x = x + h$
Uitvoer: x en y
Einde

De tweede subcategorie is Euleralgoritme, waarin de student schriftelijk of in woorden laat zien dat hij/zij de aspecten van het Euleralgoritme kan benoemen (maar niet per se hoeft te kunnen toepassen). De student ziet bijvoorbeeld dat het volgende punt steeds op de raaklijn ligt in het vorig punt, benoemt de beginwaarde en vertelt dat de volgende stappen een herhaling zijn van de berekeningen. De twee subcategorieën verwijzen naar twee verschillende aspecten van begrip van een algoritme. Zo kan iemand een algoritme in pseudocode zetten zonder dat hij/zij begrijpt wat het Euleralgoritme is en het omgekeerde kan ook gebeuren.

Algoritme en Flow of Control

Deze categorie bestaat uit de subcategorieën *Simpele loop* en *Seriële executie*. Seriële executie is van toepassing als de student laat zien dat hij/zij begrijpt hoe het programma in serie uitgevoerd is. Dit is bijvoorbeeld te herkennen doordat de student zegt "in de volgende cel komt deze formule te staan" of "eerst x_0, dan x_1,...". De andere subcategorie is Simpele loop, waarin de student laat zien dat hij/zij begrijpt hoe een simpele loop werkt. Bijvoorbeeld door te



Figuur 1 Categorieën codeerschema AD.

zeggen: “Dus het moet n keer worden herhaald” of “de berekening van het volgende punt n keer…”. De subcategorieën kunnen los van elkaar geobserveerd worden (een simpele loop is een herhaling van een serie uitgevoerde executies en een seriële executie hoeft geen loop te hebben).

Opdracht bij de methode van Euler

De methode van Euler wordt in het college geïntroduceerd als de studenten verschillende manieren hebben gezien om gewone differentiaalvergelijkingen van de eerste orde op te lossen. Ze zijn op dat moment ook bekend met het gebruik van GeoGebra om richtingsvelden en oplossingen te tekenen van differentiaalvergelijkingen van de eerste orde. Ze maken kennis met het idee van een lineaire benadering van een functie, waarna het een kleine stap is naar de methode van Euler. De student krijgt als opdracht dit stap voor stap na te volgen.

Om een idee te krijgen van wat een algoritme is, bestuderen de studenten eerst enkele eenvoudige algoritmen met en zonder lussen. Deze worden in pseudocode gegeven en studenten moeten ook zelf pseudocode opschrijven. De studenten wordt vervolgens gevraagd de methode van Euler in pseudocode te schrijven. Na deze opdracht wordt het algoritme toegepast in GeoGebra om een benadering van een oplossingskromme van een gegeven differentiaalvergelijking te tekenen (Figuur 2).

Opdracht bij de methode van Runge–Kutta

Een veel betere benadering krijg je met de methode van Runge–Kutta. Hierbij wordt steeds een volgend punt (x_n, y_n) berekend met behulp van een soort gemiddelde helling. Hiertoe worden de hellingen genomen van een viertal hulppunten.

De student kan de serie ingewikkeldere berekeningen invoegen op de plek van het Euleralgoritme in de eerder gevonden pseudocode. Weer tekenen de studenten een benadering van een oplossingskromme met behulp van GeoGebra, om te ontdekken dat deze zeer nauwkeurig is. Een leuk weetje is dat de makers van GeoGebra zelf het algoritme van Runge–Kutta gebruiken voor het tekenen van oplossingskrommen.

Opdracht bij een complexer model

Als de studenten een paar algoritmes hebben gezien, willen we een complexer systeem nemen en gaan programmeren in Python. Voor het stelsel differentiaalvergelijkingen van het Lotka–Volterra-model

$$\frac{dx}{dt} = \alpha \cdot x(t) - \beta \cdot x(t) \cdot y(t),$$
$$\frac{dy}{dt} = -\delta \cdot y(t) + \gamma \cdot x(t) \cdot y(t),$$

kunnen studenten bijvoorbeeld een pseudocode schrijven met zowel de methode van Euler als met die van Runge–Kutta voor zekere waarden van α, β, γ en δ. Dan gaat het om twee parallel verlopende berekeningen. Om de stap naar een programmeertaal te zetten, laten we de studenten een programma in Python bekijken (en uitvoeren) waar de oplossingen van een Lotka–Volterra-model worden benaderd met de methode van Euler. Vervolgens moeten de studenten de methode van Euler vervangen door die van Runge–Kutta en het programma testen. Dit onderdeel is nog in ontwikkeling en komt in een volgende ronde aan bod.

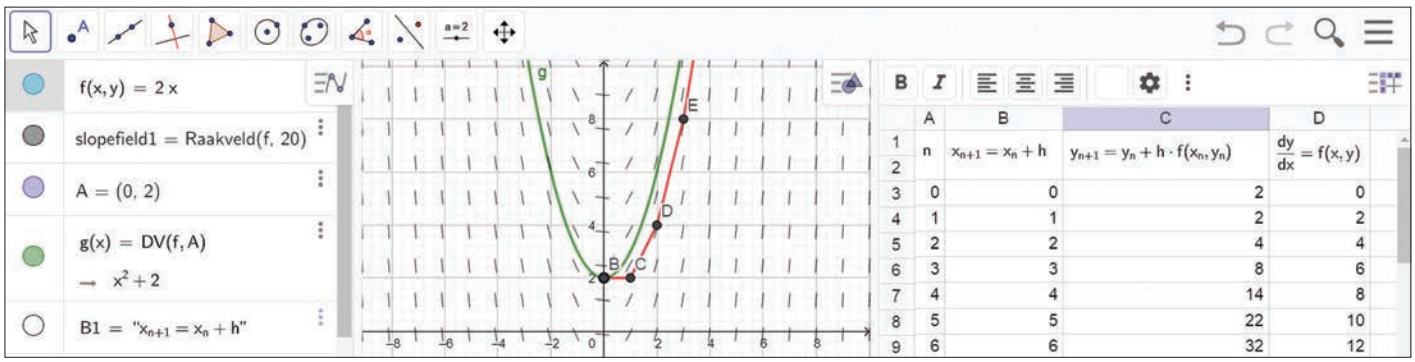
Ervaringen met de opdrachten

We hebben de opdrachten door de studenten laten maken in studiejaar 2020–2021. Dat was tijdens twee online sessies vanwege de lockdown die was ingesteld gedurende de coronapandemie. Twee van de studenten zijn elk door een onderzoeker apart genomen in een onlinevergadering om de opdrachten hardop denkend te maken. De rest van de studenten hebben college van de docent Dynamische Systemen gekregen waarin dezelfde stof werd behandeld. De lesstof maakte deel uit van het tentamen dat alle studenten aan het einde van het vak hebben gemaakt.

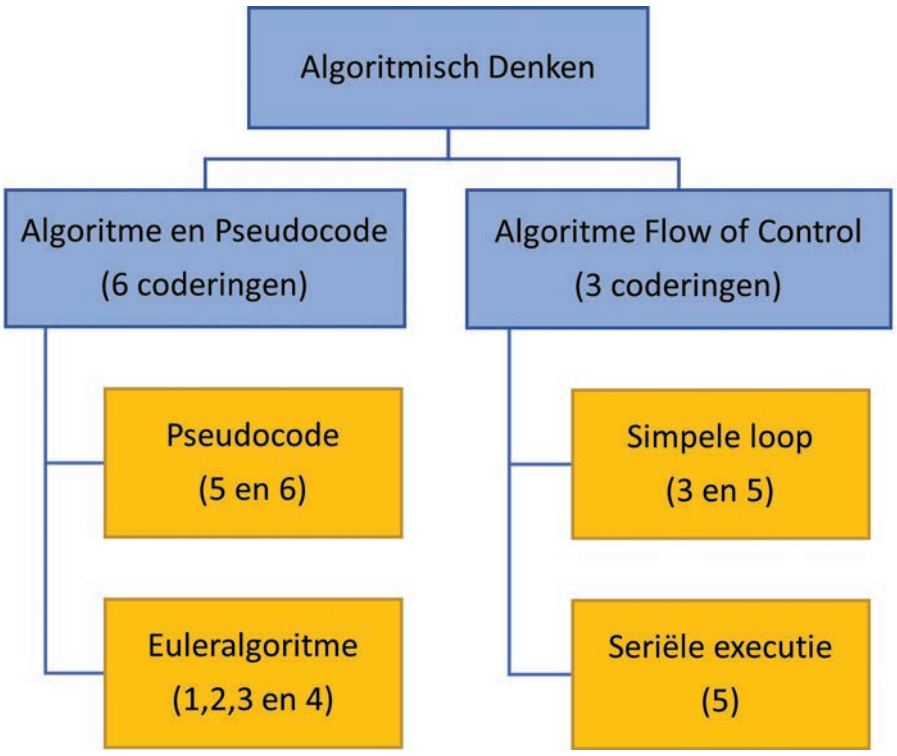
We hebben geobserveerd hoe de twee studenten aan de slag gingen met de opdrachten en gekeken in hoeverre de opdrachten bij de studenten tot AD leidden. Beide studenten waren uiteindelijk in staat om het Euleralgoritme uit te leggen en de pseudocode op te stellen. We zagen dat de studenten afwisselend de instructie lazen, nadachten over hun oplossingen en regelmatig terugkeken. Daarnaast zagen we dat de studenten vaak aan het reflecteren waren tijdens het oplossingsproces. Figuur 3 laat een weergave zien van de codering van het AD van een van de studenten, al werkend aan de opdracht met de methode van Euler.

Van de verschillende fragmenten vonden we er zes waarin AD was te herkennen. Sommige coderingen van de fragmenten vielen onder meerdere categorieën. Er waren negen coderingen in totaal waarvan er zes onder Pseudocode en Euleralgoritme kunnen worden geschaard en drie onder Flow of Control. Een voorbeeld van een fragment dat is gecodeerd als Flow of Control met simpele loop:

> “Ik zou bij stap 5 dan invoeren hoe die uitvoer gegeven moet worden, dat is door die formule neer te zetten van $x_{n+1} = x_n + h$, of uh, $y_{n+1} = y_n + h$ maal $f(x_n, y_n)$, maar dan wel even de n’s vervangen door nullen en dan zeggen dat die dat moet gaan herhalen en dat die zelf een uh… dat werd net een lus genoemd geloof ik. Dat die zelf een lus maakt.”



Figuur 2 GeoGebra-werkblad.



Figuur 3 Visuele weergave AD van een student.

Het is interessant om te zien dat de student nog twijfels over zijn oplossing heeft, nadat hij de pseudocode uiteindelijk had gevonden. Dit wijt de student zelf aan een gebrek aan ervaring met coderen. We denken dat het er ook te mee maken heeft dat hij niet gewend is om zelf een algoritme uit te werken.

Conclusie en discussie

De opdrachten die we hebben ontworpen voor het vak Dynamische Systemen zetten de studenten aan het denken over algoritmes en over manieren om ze weer te geven. We zien dat studenten nog wel even moeten puzzelen voordat ze een simpele loop kunnen opschrijven in pseudocode, maar dat ze er uiteindelijk uitkomen. De opdrachten zijn toegankelijk en laten studenten op een laagdrempelige manier ervaring opdoen met algoritmes en algoritmisch denken.

Algoritmisch denken is niet nieuw binnen de wiskunde; je kunt wel zeggen dat het onmogelijk is om wiskunde te bedrijven zonder algoritmisch te denken. We weten echter ook hoe moeilijk het is om denken en redeneren in de schoolpraktijk vorm te geven. Het expliciet benoemen van AD en CT als zelfstandige activiteiten, kan ertoe leiden dat er in het onderwijs meer aandacht komt voor deze processen. Wellicht dat docenten hierdoor ook ruimte in het curriculum maken om aandacht te besteden aan denken en redeneren en een didactiek kunnen toepassen die dit ook ondersteunt.

Het lijkt ons dat de lerarenopleiding een uitstekende plek is om aankomende docenten wiskunde hierop voor te bereiden. Het schema dat we hebben gemaakt om het AD van studenten in beeld te brengen, heeft ons geholpen om scherper te krijgen welke aspecten van de opdracht leiden tot AD. En we zien dat we het schema nog kunnen verbeteren. Zo vertonen de subcategorieën Pseudocode en Euleralgortime enige overlap. Een student kan bijvoorbeeld blijk geven van begrip van het Euleralgoritme door een goede pseudocode op te schrijven. Dit geldt ook voor de subcategorieën Simpele loop en Seriële executie.

Verder denken we dat we in de opleiding bij andere onderdelen van wiskundevakken eenvoudige algoritmes kunnen introduceren, voordat Dynamische systemen aan bod komt, en dat we de studenten hier wat langer dan twee colleges mee moeten laten werken. In feite moeten we meer ruimte maken bij de masteropleiding leraar wiskunde van de Hogeschool van Amsterdam om studenten algoritmisch te laten denken.

Referenties

1 C. Angeli, J. Voogt, A. Fluck, M. Webb, M. Cox, J. Malyn-Smith en J. Zagami, A K-6 computational thinking curriculum framework: Implications for teacher knowledge, *Education Technology and Society* 19(3) (2016), 47–57.

2 S. Grover, R. Pea en S. Cooper, Designing for deeper learning in a blended computer science course for middle school students, *Computer Science Education* 25 (2015), 199e237.

3 C.C. Selby, *How Can the Teaching of Programming be Used to Enhance Computational Thinking Skills?*, doctoral dissertation, University of Southampton, 2014

4 V.J. Shute, C. Sun en J. Asbell-Clarke, Demystifying computational thinking, *Educational Research Review* 22 (2017), 142–158.

5 J.M. Wing, Computational thinking: What and why?, Carnegie Mellon University, 2010, https://www.cs.cmu.edu/~CompThink/papers/TheLinkWing.pdf.

6 10voordeleraar, Vereniging Hogescholen, *Kennisbasis Eerstegraadslerarenopleiding Wiskunde*, 2018, https://10voordeleraar.nl.

7 https://www.curriculum.nu/voorstellen/rekenen-wiskunde/uitwerking-rekenen-wiskunde, Grote opdracht 13: Algoritmisch denken.