Modelleren van de Neocortex: Analyse & simulatie op grote schaal

Afstudeerproject
Applied Mathematics
Numerical Analysis and Computational Mechanics (NACM)
Applied Analysis and Mathematical Physics (AAMP)
Universiteit Twente; Faculteit EWI

Wilco Bouwhuis

onder begeleiding van
prof. dr. ir. B.J. Geurts (numerieke methoden/analyse)
prof. dr. S.A. van Gils (neurofysiologische aspecten)
S. Visser, MSc. (Hodgkin-Huxley model - Implementatie C++)

December 2009 - Juli 2010

Voorwoord

Bij de totstandkoming van deze scriptie en de uitvoering van het project is vanuit allerlei hoeken hulp verleend. Zonder hen had de beschrijving zoals deze nu voor u openslaat nooit in deze vorm kunnen verschijnen. Ik zou daarom graag een aantal mensen willen bedanken.

Directe begeleiders tijdens het onderzoek waren professor Bernard Geurts als afstudeerdocent, professor Stephan van Gils als opdrachtverlener (d.w.z.: het onderzoek bevond zich binnen zijn vakgebied) en promovendus Sid Visser, MSc. als directe vraagbaak. Ik ben zeer dankbaar voor de sturing, maar ook zeker voor de motivatie die zij mij wisten te geven in de uitvoering van het onderzoek. Ik heb gedurende de laatste fase van de Masteropleiding Applied Mathematics aan de Universiteit Twente te Enschede een boel kunnen doen en leren.

Professor Jaap van der Vegt, leerstoelhouder Numerical Analysis & Computational Mechanics (NACM), heeft gefunctioneerd als begeleider tijdens het cursorisch gedeelte van de Masteropleiding NACM. Hij heeft een zeer belangrijke rol gespeeld in de route naar het afstuderen, inclusief de stage.

Dank gaat ook uit naar dr. Hanneke Becht, literatuurdeskundige, ir. David Lopez-Penha voor zijn introductie in het Huygens-systeem, dr. Ruud van Damme voor zijn heldere beschrijving betreffende programmeren in C++-taal, en dr. Lilit Axner voor de hulp betreffende de Supercomputer te Amsterdam.

Last but not least zou ik ook zeker graag mijn familie en kennissen bedanken voor hun hulp; in het bijzonder mijn ouders, die mij de mogelijkheid hebben gegeven deze studie te volgen, en mijn vriendin Geerke, die ik tijdens de afstudeerperiode heb ontmoet, en me heeft gesteund en tevens sterk heeft weten te motiveren. De afgelopen tijd was op allerlei vlak een heel bijzondere (en leuke) tijd, die veel te snel voorbij is gevlogen!

Enschede, juli 2010

Samenvatting

Achtergrond

Deze scriptie is gebaseerd op werk binnen zowel de vakgroep Applied Analysis and Mathematical Physics (AAMP) als de vakgroep Numerical Analysis and Computational Mechanics (NACM) onder de afdeling Applied Mathematics van de Universiteit Twente te Enschede. De scriptie is een examenonderdeel voor het behalen van het diploma voor de mastertitel Applied Mathematics.

De omgeving 'GENESIS' bevat een computermodel van een hersengedeelte (ontwikkeld door Van Drongelen e.a.) en wordt onder meer gebruikt om de verschijnselen van epilepsie te bestuderen. Het model maakt gebruik van het beroemde Hodgkin-Huxley-model, dat een neuron (zenuwcel) opdeelt in een aantal compartimenten, waarbinnen vrij eenvoudige differentiaalvergelijkingen worden opgelost. Sid Visser, MSc. heeft het model binnen GENESIS herbouwd naar 'NeuroSim', een C++-variant. Met zijn code kan op zeer flexibele wijze neuronennetwerkgedrag beschouwd worden. Tot dusver is dit gebeurd voor een netwerk van ongeveer 700 neuronen, i.e. een hersenstukje van slechts 80 μm bij 80 μm bij 1.5 mm. Met behulp van parallelrekening (hier OpenMP) kan, binnen afzienbare simuleertijd, het gedrag van een veel groter netwerk (hier tot ca. 18.500 neuronen) beschouwd worden. Titel van deze scriptie is daarom 'Modelleren van de Neocortex: Analyse & simulatie op grote schaal'.

Doel

Doel van het onderzoek is een antwoord te vinden op de vraag hoe het gedrag van een populatie neuronen zich ontwikkelt bij een groeiend simulatieprobleem, d.w.z. bij een groeiend aantal neuronen binnen groeiende afmetingen. Voor een klein groepje neuronen zijn verschillende typen gedrag herkend: het virtuële Elektro-Encefalogram (EEG) toont toestanden van stilte (er worden groepsmatig amper actiepotentialen gevuurd), fluctuaties (er worden zeer veel actiepotentialen gevuurd op ongeordende momenten) en ritmische groepsgewijze oscillaties (epileptiform gedrag), afhankelijk van de netwerkexcitatie en -inhibitie. Vraag is hoe het populatiegedrag is bij een groot aantal neuronen. Wat zijn de ritmen van de (eventuele) groepsgewijze oscillaties en hoe ziet het excitatie/inhibitie-parametervlak eruit? Een belangrijke kwestie is of het OpenMP Single Neuron model wellicht te herschrijven is in een (minder complex) populatiemodel, dat het gedrag voor een gehele groep neuronen beschrijft, in plaats van per neuron.

Methode

Er wordt een OpenMP-implementatie toegepast op NeuroSim en een eenvoudige populatiegedragsclassificatie uitgewerkt. Op de verbindingsparameters valt bovendien enige theoretische analyse uit te voeren. We beschouwen hiermee de resultaten van het grote-schaal Single Neuron model 'NeuroSim'. Aan de hand van een kort literatuuronderzoek worden deze resultaten vergeleken met output van populatiemodellen.

Resultaten & Conclusies

Het valt op dat een grootschalig hersenmodel met onveranderde verbindingsparameters (zoals sterkte, verbindingskans en maximum verbindingsafstand) het netwerk eerst meer excitatoir, maar voor nóg grotere probleemafmetingen zeer inhibitoir maakt. Het totaalnetwerkgedrag loopt uiteindelijk naar een limiet toe, die al snel (asymptotisch) benaderd wordt als naar de genoemde $\sim$18.500 neuronen wordt toegewerkt. Dit wordt tevens teruggezien in de verscheidene E-I-vlakken met EEG's, die voor verschillende aantallen neuronen geconstrueerd zijn. Theoretische analyse op de verbindingsparameters bevestigt bovendien het verloop richting de genoemde limiet. Als men de gedragsregimes zou willen vertalen in een populatiemodel, zou dit in eerste instantie aan de hand van de activititeit (zeg, het aantal actiepotentiaalvuringen per populatie van een zekere grootte per zeker tijdsinterval) binnen een groep neuronen moeten plaatsvinden.

Abstract

Background

This thesis is based on work performed at as well the group of Applied Analysis and Mathematical Physics (AAMP) as the group of Numerical Analysis and Computational Mechanics (NACM), of the Department of Applied Mathematics at the University of Twente, Enschede. The thesis is a fundamental part of the examination for the master title Applied Mathematics.

The environment 'GENESIS' contains a computer model of a brain part (developed by Van Drongelen et al.) and has been used to study the phenomena of epileptic activity. The model applies the famous Hodgin-Huxley model, which divides a neuron into a number of compartments, in which relatively simple differential equations are solved. Sid Visser, MSc. has rebuilt the GENESIS model into 'NeuroSim', a C++-implementation. With his code one is very flexible to study network behaviour. So far this has been examined for a network of about 700 neurons, i.e. a brain part of mere 80 μm times 80 μm times 1.5 mm. With use of parallel calculation (OpenMP, in our case), it is possible to obtain results for a much larger neuronal network (say, $\sim$18.500 neurons). Therefore, the title of this thesis is 'Modelling the Neocortex: large scale Analysis & Simulation'.

Purpose

Purpose of the research is to find an answer on the question how neuronal population behaviour develops during increasing simulation scale, that is: an increasing number of neurons inside growing problem dimensions. For a small group of neurons, different types of groupwise behaviour are recognized: the virtual Electro-Encefalogram (EEG) shows states of silence (almost no action potentials are fired), fluctuations (there are very much action potentials at unordered moments) and rhythmic groupwise oscillations (epileptiform behaviour), depending on the network excitation and inhibition. What is the population behaviour in case of a large number of neurons? What are the rhythms of the groupwise oscillations (if there are any) and how looks the excitation-/inhibition-parameter plane? Could the OpenMP Single Neuron model be described with a (less complex) Population model? In that case the behaviour of a complete group of neurons can be considered, instead of the output per neuron.

Methods

An OpenMP implementation is applied on NeuroSim and a simple population behaviour classification is developed. Also a theoretical analysis can be applied on the connection parameters. With this, we consider the large scale results of the Single Neuron model 'NeuroSim'. Using a short literature investigation, these results are compared with output of neuronal population models.

Results & Conclusions

It stands that a large scale brain model with unchanged connection parameters makes the network at first more and more excitatory, but at still larger problem sizes very inhibitory. The groupwise behaviour eventually drops to a limit, which quickly is reached (asymptotically), if the number of neurons gradually is increased to the mentioned $\sim$18.500. This progress can also be observed by constructing several E-I-planes with EEG's for some numbers of neurons. Theoretical analysis on the connection parameters also confirms this course to the called limit. If one would translate the behavioural regimes into a population model, this should at first sight take place using the activity (say, number of action potential firings per population of certain size per certain time interval) within a group of neurons.

Inhoudsopgave

Hoofdstuk 1

Inleiding

1.1 Probleemstelling

De werking van het (menselijk) brein wordt in vele aspecten nog niet begrepen. Het functioneren van zo'n 10^{11} neuronen en daartussen $\sim 10^{14}$ verbindingen is ongelooflijk complex en mag gerust één van de wonderen der natuur genoemd worden. Het is dan ook een uitdaging om een zo gedetailleerd mogelijk wiskundig en numeriek model van de hersenen te maken, wat het doel is van het zgn. 'Blue Brain Project'. Dit model is zó ontzettend groot dat het onmogelijk is er enige analyse op los te laten, zowel in theoretisch als in numeriek opzicht. Het virtuële model runt uitsluitend op een speciale computer. GENESIS en 'NeuroSim', een remake van GENESIS in C++-taal, beschouwen een wat minder geavanceerd model, gebaseerd op het werk van Hodgkin & Huxley. GENESIS wordt onder meer gebruikt om de verschijnselen van epilepsie te bestuderen. Alhoewel de volledige werking van de hersenen misschien nooit op een computersysteem te modelleren zal zijn, geeft de digitale remake van het brein een goede kijk op netwerkgedrag van een zekere 'hersenkolom' met een 'geringe' hoeveelheid zenuwcellen. $\sim$700 neuronen vormen samen een aantal minikolommen - er wordt dan gepraat over een stukje hersen van (slechts) 80 micrometer bij 80 micrometer bij 1500 micrometer. Véél grotere netwerken vereisen een enorme rekentijd. Echter, parallelrekening over meerdere CPU's is een aangewezen methode om de berekeningen over een dergelijk groot zenuwnetwerk significant te versnellen. Centraal in dit project staat het beschouwen van de resultaten bij hersennetwerken van een (vooralsnog) 'groot' aantal neuronen - zeg, maximaal zo'n $\sim$18500 (afmetingen ongeveer 425 micrometer bij 425 micrometer bij 1500 micrometer). Er zouden allerlei effecten op kunnen treden naarmate het aantal 'virtuële' zenuwcellen wordt vergroot, waarvan sommige in bepaalde opzichten te voorspellen/formuleren zijn. Er zullen in de volgende paragraaf dan ook een aantal concrete onderzoeksvragen en hypothesen worden opgesteld.

De route in het project is dus een stappenproces. Allereerst wordt, met behulp van OpenMP, de code voor NeuroSim op een zo effectief mogelijke manier geparallelliseerd, om lange simulaties met een groot aantal neuronen tot een werkbare situatie te maken. Zodra deze doelstelling behaald is, wordt initieel gekeken naar het zgn. 'kunstmatige EEG', lees voor het gemak: totaalgedrag van het neuronennetwerk. Nu kunnen al bepaalde vragen over het netwerkgedrag ten opzichte van het aantal neuronen beantwoord worden. Terwijl het totaalgedrag nauwkeurig in de gaten wordt gehouden, kunnen bepaalde parameters worden gevarieerd om de gevolgen daarvan te observeren. Mogelijk moeten bepaalde netwerkeigenschappen gecorrigeerd of op een andere manier geïmplementeerd worden. Hierbij kan men zich, zo mogelijk, gaan richten op de theoretische uitwerking van bepaalde netwerkeigenschappen. Analyse van de gamma ritmen en plaatsafhankelijkheid van groepsgedrag (oftewel ruimtelijke structuren) kan daarna in feite rechtstreeks worden uitgevoerd door bijvoorbeeld de aandacht te verdelen over verschillende 'hersenblokken' (kolommen).

Merk op dat dergelijke onderzoekspunten al vaker zijn gepasseerd voor diverse modellen van het brein. Voor de C++-implementatie ligt het (geleidelijk) vergroten van het netwerk nog open.

1.2 Onderzoeksvragen en hypothesen

De vraag die we allereerst dienen te beantwoorden is hoe groot het gesimuleerde hersensysteem mag zijn, waarbij we, met behulp van OpenMP, binnen afzienbare tijd (de 'gewenste', lees: identiek aan als zijnde berekend met 1 CPU) output verkrijgen. Zouden we wellicht een hyperkolom van zo'n 100.000 neuronen kunnen simuleren of houdt het bij 10.000 of zelfs nog minder wel op? Het is lastig om daar voorspellingen over uit te spreken. Interessant zijn de zgn. 'Speed-up lines'; we houden het rekenkundig probleem identiek, verhogen stapsgewijs het aantal benutte CPU's en beschouwen het verloop van de benodigde kloktijden. Zodra we dergelijke simulatie-eigenschappen in beeld hebben, richten we ons op de onderzoeksvragen die hieronder beschreven worden.

Voor een 'klein' hersensysteem van $\sim$700 neuronen springen een aantal eigenschappen in het oog. Neuronen hebben bijvoorbeeld de neiging om (ritmisch) groepsgedrag te vertonen ('epileptiform' gedrag), althans bij realistische netwerkparameters en condities. In een bepaald (gamma) ritme (met een frequentie in de orde van grootte van 40 Hz) kan een grote groep SPyr-cellen (voor dergelijk kleine systemen vaak álle SPyr-cellen) in het model op (nagenoeg) hetzelfde moment een actiepotentiaal vuren. Dit is duidelijk zichtbaar in het kunstmatige EEG. Eén van de belangrijkste vragen is hoe het netwerkgedrag zal veranderen (althans, *als* het verandert) bij een groter wordend hersensysteem. Worden er, in verhouding tot het aantal neuronen, méér of minder actiepotentialen gevuurd (m.a.w.: verandert de excitatie en/of inhibitie van het model)? Veranderen de frequenties van de gamma ritmen? Zijn deze veranderingen 'gewenst', oftewel realistisch voor een *echt* brein? Hypothese voor wat betreft de gamma ritmen is dat deze ietwat in frequentie zullen afnemen, aangezien de gemiddelde reistijd zal groeien bij een groter wordend netwerk (er bestaat een maximum afstand tussen twee verbonden neuronen in het virtuële netwerk, maar voor een voldoende klein systeem kan een neuron in de ene uiterste hoek best verbinden met een neuron in de andere uiterste hoek van het gemodelleerde groepje neuronen; bij grotere netwerken speelt de maximale verbindingslengte een significantere rol en zullen limieten opspelen). Hoe de totaal-excitatie van het netwerk zal veranderen is een stuk lastiger te voorspellen. Omdat dergelijke voorspellingen fikse theoretische berekeningen vergen, die onderdeel uitmaken van het onderzoek, formuleren we hierover dan ook nog géén concrete hypothese. Verwacht wordt dat de gemiddelde reistijd van zenuwsignalen (actiepotentialen) tegen het aantal cellen naar een (min of meer) horizontale lijn zal afvlakken en het aantal verbindingen tegen het aantal cellen geleidelijk een lineair verband zal aannemen (en in het begin kwadratisch, wat al geconstateerd is voor een klein aantal neuronen). Bij wélk aantal neuronen dergelijke limieten opspelen is echter nog steeds de vraag.

Het valt voor te stellen dat actiepotentialen een zekere tijd nodig hebben om te reizen van, bijvoorbeeld, de ene kant naar de andere kant van het modelblokje met neuronen. Zodra de afmetingen groeien, wordt de gemiddelde afstand en dus reistijd van actiepotentialen tussen de neuronen groter en kan men wellicht al vertragingen in het groepsgedrag constateren. De neuronen aan de ene kant van het systeem vuren dan bijvoorbeeld in hetzelfde ritme als aan de andere kant, maar met een zekere tijdsvertraging (faseverschil), omdat het groepsgedrag zich over de verbindingen moet verplaatsen. Het is een (onderzoeks)vraag of we dergelijk ruimtelijk gedrag zullen herkennen en of een netwerk van het aantal cellen tot waar we komen met behulp van OpenMP genoeg cellen zijn om dergelijke ruimtelijke verplaatsingen terug te zien.

Tot slot is het een belangrijke kwestie of wellicht een groepsmodel opgesteld mag worden voor het (vergrote) neuronennetwerk. Is het mogelijk om complete populaties van zenuwcellen (gebaseerd op neurontype, neuronpositie, of beide) te beschrijven met een zekere functie of, als alternatief, met zeker voorspelbaar gedrag? Kunnen we, in plaats van met zeer veel differentiaalvergelijkingen in het Single Neuron model een veel eenvoudiger (kleiner) wiskundig model opstellen voor complete populaties met neuronen? Is de output van NeuroSim terug te koppelen aan (in de wetenschap bekende) eigenschappen van populatiemodellen? Groepsgedrag is dus absoluut een punt van onderzoek.

1.3 Opbouw van de scriptie

Dit verslag start met een samenvatting van de meest essentiële informatie betreffende neurofysiologie en breinmodellering: hoofdstuk 2 bevat beknopte informatie over de (virtuële) neuronen en verbindingen in het (menselijk) zenuwstelsel. Het Hodgkin-Huxley model wordt behandeld, waarop (onder meer) NeuroSim gebaseerd is. Verbindingen en hun eigenschappen blijken een essentiële rol te spelen in (de output van) het model. Aan de hand van de verbindingen tussen de neuronen in het model kunnen een aantal theoretische voorspellingen worden gedaan. In hoofdstuk 3 wordt hier aandacht aan besteed. Hoofdstuk 4 is een overzicht van de experimentele resultaten: onder meer de runtijden, netwerkexcitatie, gemiddelde reistijd van actiepotentialen en gamma ritmen worden beschouwd. Hoofdstuk 5, het laatste hoofdstuk, bevat de conclusies, een discussie van de resultaten en een toekomstvisie.

Hoofdstuk 2

Breinsimulatie

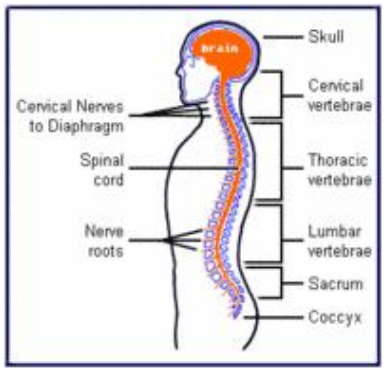
In 1952 publiceerden Alan Lloyd Hodgkin en Andrew Fielding Huxley een theorie over het modelleren van het gedrag van een neuron met behulp van een aantal elektrische vervangingscircuits. Er werd onder meer praktisch onderzoek gedaan met een gigantische axon van de (Atlantische) inktvis, zodat ionische stroompjes gemeten konden worden. Een inktvisaxon heeft slechts twee typen voltage-afhankelijke kanalen en heeft een diameter van maar liefst 1 mm; dit axon is dus min of meer zichtbaar voor het menselijk oog. In 1963 wonnen Hodgkin & Huxley de Nobelprijs ('in Physiology and Medicine') voor hun werk. Zij deelden hun prijs met John Carew Eccles, die een onderzoek had gedaan aan synapsen. Het waren Hodgkin & Huxley die het idee van de ionkanalen introduceerden. Hier ligt de oorsprong van de optie tot modelvorming van het brein. In deze paragraaf wordt achtereenvolgens het concept van een Neocortex geïntroduceerd, het basismodel van Hodgkin & Huxley toegelicht [1][5] en kort ingegaan op het Blue Brain Project [6], dat zeer veel gedetailleerde gegevens van de hersenen gebruikt om tot een zo realistisch mogelijk resultaat te komen. Hierop volgt het ietwat versimpelde model binnen GENESIS, ontwikkeld door Van Drongelen e.a. [7][8], met als belangrijkste toepassing het bestuderen van epilepsie. Het GENESIS-model is herbouwd in C++ door S. Visser, zodat men flexibel(er) de afhankelijkheid van parameters ten opzichte van het netwerkgedrag kan bestuderen. De vijfde paragraaf van dit hoofdstuk beschrijft enkele details met betrekking tot de C++-implementatie, die grotendeels werkt met een 'Linked List'-systeem. Merk dus op dat paragraaf 2.5 ingaat op enkele details van het programma, terwijl de twee paragrafen daarvóór uitsluitend (prestatiegerichte) info bevatten over het Blue Brain Project en de GENESIS-simulatie. Deze scriptie heeft immers van doen met (de versnelling van) de C++-versie. De laatste paragraaf bevat informatie in de richting van zgn. populatiemodellen in vergelijking met Single Neuron modellen zoals deze in paragraaf 2.2-2.5 besproken worden.

2.1 De gemodelleerde Neocortex

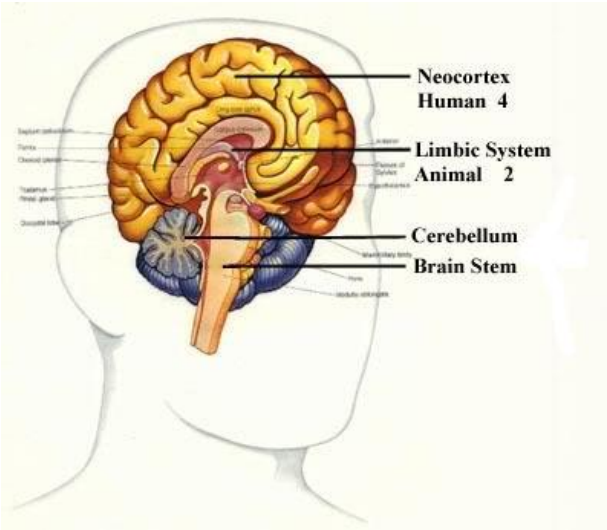
Binnen het Hodgkin-Huxley model komen een aantal aspecten met een neurofysiologische grondslag terug: het is uiteraard wenselijk dat de computermodelvorming van de hersenen tot op zekere hoogte realistisch is. In deze introducerende paragraaf wordt beknopt de stap van realiteit naar breinsimulatie beschreven. Voor een uitgebreidere kijk op neurofysiologie wordt verwezen naar [1].

2.1.1 Neurofysiologische achtergrond

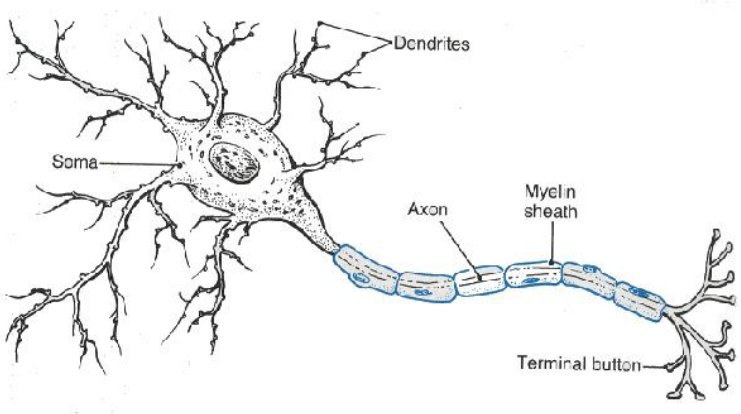
Het menselijk brein, de Cortex, vormt samen met het ruggenmerg en het netvlies het Centraal Zenuwstelsel (zie figuur 2.1). De Cortex is een 'processor' die aan de hand van input vanuit de zenuwen in het lichaam gedachten en acties stuurt. In figuur 2.2 wordt het menselijk brein gedetailleerder uitgebeeld. Er is te zien dat het grootste gedeelte van de Cortex bestaat uit een 'ineengeduwde verkronkelde massa', de Neocortex. Het 'Cerebellum' (gelegen onderin de hersenstam) behoort *niet* tot de Neocortex en het breinmodel dat beschouwd zal worden heeft dus geen



Figuur 2.1: Het Centraal Zenuwstelsel, bestaande uit de hersenen, het ruggenmerg en het netvlies. (http://www.vonkenvisie.be/typo3temp/pics/5fbabbefe9.gif)



Figuur 2.2: 'Indeling' van de hersenen. Het cerebellum is een 'op zichzelf staand' gedeelte van de Cortex en behoort niet tot de Neocortex.
(http://media.photobucket.com/image/neocortex/Primate_bucket/3brains.jpg)



Figuur 2.3: Structuur van een neuron. Het soma is het cellichaam, waarop de dendrieten uitmonden. Een axon geleidt signalen naar een volgend neuron.
(http://www2.cedarcrest.edu/academic/bio/hale/bioT_EID/lectures/tetanus-neuron.gif)

betrekking op dit kleine (ook anatomisch gescheiden) hersengedeelte.

De menselijke Neocortex bestaat uit zo'n 20.000.000.000 zenuwcellen, 'neuronen' genoemd. Kenmerkend voor neuronen is de prikkelbaarheid: zenuwcellen kunnen signalen ontvangen en deze *zonder verlies van signaalsterkte*doorgeven aan de hand van (actieve) regeneratie van een volgend signaal (naar een volgend neuron). Deze informatie-overdracht vindt plaats door middel van elektrochemische processen. De anatomie van een neuron wordt weergegeven in figuur 2.3. De kenmerkende onderdelen van neuronen zijn de axonen, die van de cel af geleiden (naar een volgende cel of naar de hersenen), en de dendrieten, die (meestal) naar het 'soma' (cellichaam) van de cel toe lopen. Signalen komen de cel dus binnen via de dunne, sterk vertakte dendrieten en verlaten het neuron (eventueel) via een dikkere, onvertakte (behalve aan het uiteinde) axon, bestaande uit gemyeliniseerde vezels, naar een volgend neuron.

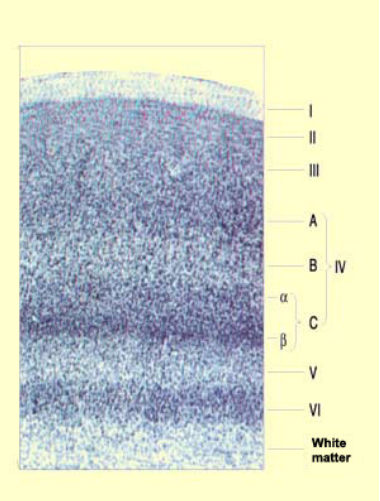
Neuronen zijn zgn. 'excitable cells': ze laten na een zekere stimulatie een bepaalde activiteit zien. In rusttoestand, dus in een toestand zonder (voldoende..) prikkels van andere neuronen, worden de cellen *typisch* gekarakteriseerd door een rust-membraanpotentiaal van -90 tot -60 milli-Volt. 'Membraanpotentiaal' duidt hier op de potentiaal over het celmembraan, het verschil tussen de lading binnen en buiten het neuron. Op het moment dat (via de dendriet) externe prikkels (stroompjes) van toepassing zijn, kan er een zgn. actiepotentiaal gegenereerd worden: een membraanpotentiaal van ongeveer +50 of +60 mV. Dit is een ja/nee-principe: wanneer een zekere voltagedrempel (in werkelijkheid afhankelijk van de duratie van de stimulatie) bereikt wordt, dan wordt er een actiepotentiaal gegenereerd door het betreffende neuron en wanneer deze drempel niet bereikt wordt, dan zendt het neuron geen nieuwe signalen uit. Een prikkel is dus niet sterk genoeg om een actiepotentiaal te ontwikkelen *of* deze komt juist volledig tot ontwikkeling. Het ontstaan van zo'n stroompuls, want dat is in feite het karakter van een actiepotentiaal, heeft een (elektro)chemische achtergrond en berust op een verhouding tussen verscheidene ionen, zoals natrium- en kalium-ionen.

Gedurende korte tijd na de generatie van een actiepotentiaal ('vuren'), is een zenuwcel niet opnieuw te prikkelen; dit wordt de 'absoluut refractaire periode' (a.r.p.) genoemd.

2.1.2 Indeling van de (gemodelleerde) Neocortex

Een Single Neuron model is niet veel anders dan een verzameling neuronen, gemodelleerd met behulp van het Hodgkin & Huxley formalisme, met verbindingen, variërend in eigenschappen als geleidingssnelheid en verbindingssterkte. De modelneuronen zijn gerangschikt naar ervaringen uit de realiteit: wanneer de Neocortex wordt 'uitgevouwen' en dit hersengedeelte als een horizontale 'brij' beschouwd wordt, blijkt dat deze brij typisch uit zes verschillende lagen (met, wanneer we gedetailleerder te werk gaan, bepaalde deellagen) met zenuwcellen bestaat [2], zoals getoond in figuur 2.4. Onder de bovenste laag (I) bevinden zich twee lagen met onder meer '(superficial) pyramidal cells', 'oppervlakkige piramidecellen. Een piramidecel is een bepaalde soort zenuwcel die er anatomisch inderdaad uitziet als een piramide. Deze cellen zijn 'excitatoir': ze vergroten de kans op de generatie van een actiepotentiaal in andere zenuwcellen en werken dus stimulerend. In laag IV blijken zich geen piramidecellen te bevinden, maar uitsluitend 'interneuronen': 'basketcellen', 'kaarsenmakerscellen' ('chandelier cells') en andere, minder voorkomende/belangrijke interneuronen. De meeste interneuronen zijn 'inhibitoir', oftewel remmend. Ze werken de generatie van actiepotentialen in neuronen waarmee ze een verbinding hebben tegen. In laag V en VI bevinden zich opnieuw excitatoire piramidecellen, de 'deep pyramidal cells'. De neuronen worden ook in het computermodel op een wijze verdeeld als zijnde zes lagen, waarbij de bovenste laag geen gemodelleerde neuronen bevat. De SPyr-cellen bevinden zich tussen 250 en 750 micrometer diepte, de DPyr-cellen tussen 1000 en 1500 micrometer diepte en de interneuronen verspreiden zich over de gehele z-as (250-1500 micrometer). De xy-verdeling en de z-spreiding binnen de betreffende regio's is uniform.

De celverdeling is ongeveer 80% excitatoire piramidecellen tegen 20% interneuronen; De 'uitgevouwen' hersenen hebben een oppervlakte van ongeveer 40 centimeter bij 40 centimeter en een diepte van 1.5 mm. In een minikolom van 80 micrometer bij 80 micrometer (uiteraard met dezelfde diepte) bevinden zich ca. 700 (model: 656) neuronen (256 SPyr, 256 DPyr, 36 Bask1, 36 Bask2, 36



Figuur 2.4: Verscheidene 'lagen' in het (menselijk) brein. Er zijn verschillen te zien in dichtheden van zenuwcellen. In laag II bevinden zich de Superficial Pyramid (SPyr) Cells, laag IV en V bevatten de Deep Pyramid (DPyr) Cells en interneuronen bevinden zich verspreid over laag II-VI. Onder interneuronen vallen Basket Cells en Chandelier Cells.
(http://thebrain.mcgill.ca/flash/i/i_02/i_02_cl/i_02_cl_vis/i_02_cl_vis_3a.jpg)

Bask3 en 36 Chand). De celdichtheid is dus ongeveer $68.3 \cdot 10^{12}$ cellen per kubieke meter hersen.

2.1.3 Verbindingen

In de Neocortex bevinden zich zo'n 10^{14} verbindingen tussen de $\sim 10^{11}$ neuronen [3]. Zonder deze verbindingen zouden AP's uiteraard nooit kunnen worden overgedragen. De termen 'excitatoir' en 'inhibitoir' zijn aan bod gekomen. Deze termen zijn geen eigenschappen van de verbindingen, maar van het neuron zelf. Wanneer men praat over een 'excitatoire verbinding', wordt een verbinding vanaf een excitatoir neuron bedoeld. Verbindingen tussen cellen variëren in diverse eigenschappen; een feit dat meegenomen dient te worden in het computermodel. Er wordt gepraat over voortbewegingssnelheden van AP's (vertragingen) en verbindingssterkten. *Of* er een verbinding tussen twee *virtuele* neuronen is, is bovendien een kanssommetje gebaseerd op onder meer de (kortste) afstand tussen de twee neuronen. Informatie omtrent verbindingen en hun eigenschappen tussen virtuele zenuwcellen is te vinden in hoofdstuk 3.

Merk op dat de hoeveelheid verbindingen zeer sterk met het aantal cellen toeneemt, althans bij realistische parameterwaarden. Een verdubbeling van het aantal cellen betekent in het algemeen véél meer dan een verdubbeling van het aantal verbindingen. De complexiteit van het model neemt dus zéér sterk toe met het aantal neuronen. Uitspraken als deze zijn van theoretische aard, maar worden praktisch bevestigd in paragraaf 4.2. Daar wordt ook ingegaan op een ander punt van onderzoek, te weten de verhouding tussen het aantal excitatoire verbindingen en het aantal inhibitoire verbindingen. Deze verhouding zal meer bepalend zijn voor het netwerkgedrag van zenuwcellen.

De tabel in hoofdstuk 3 slaat uitsluitend op het Hodgkin-Huxley model binnen GENESIS, dat alleen de 6 belangrijkste celsoorten omhelst, en is hooguit *gebaseerd* op de reële Neocortex (aan de hand van een fikse literatuurstudie). In deze tabel gaat het om de simulatie van de Neocortex van een muis.

2.2 Het Hodgkin-Huxley model

In de vorige paragraaf werd al enigszins gerefereerd naar een hoeveelheid ionkanalen binnen een neuron. Daar komt de condensatorwerking van het celmembraan bij; het membraan weet lading op te slaan en later vrij te laten. Ieder ionkanaal kan zelf als het ware (positieve, dan wel negatieve) stroom leveren, door middel van een verandering in de ladingverhouding ten gevolge van verplaatsing van bijvoorbeeld natrium- en kaliumionen. Dat betekent dat een elektrisch vervangingsschema van een neuron ook spanningsbronnen zal bevatten: de zgn. Nernstpotentiaal [1] voor de verschillende ionen maakt deel uit van het model. Verder hebben de kanalen een zekere (veranderbare - in elektrotechnische termen: regelbare) weerstand R. In het model wordt de reciproke van deze weerstanden gebruikt, namelijk de geleiding $g = 1/R$, omdat een kanaal dat *niet* geleidt in het andere geval ingesteld zou moeten worden als een kanaal met $R = \infty$. $g = 0$ werkt 'beter' in computertaal. Al de genoemde gegevens zijn uit te drukken in het vervangingscircuit van figuur 2.5. Ieder ionkanaal krijgt zijn eigen 'deelroute' van de parallelschakeling; de stroom door het membraan wordt verdeeld over de verschillende routes. De spanning over iedere deelroute is de membraanpotentiaal V_m. De 'richting' van de spanningsbronnen E_{Na} en E_K in figuur 2.5 heeft te maken met de functionaliteit van het betreffende ionkanaal (stromen de ionen naar binnen of naar buiten?). Men zou ervoor kunnen kiezen om meerdere kanalen te modelleren, bijvoorbeeld een chloridekanaal toe te voegen, doch, zoals eerder aangegeven, vormen natrium en kalium de meest belangrijke ionen. Wel wordt er een op zichzelf staand 'synaptisch' kanaal beschouwd, het specifieke 'voltage gated' kanaal, waarin synapsen (seinstoffen) vanuit andere neuronen terechtkomen. Synaptische kanalen zijn tevens te modelleren als kanalen met een regelbare weerstand (geleiding g_{syn}) en een spanningsbron met spanningslevering E_{syn}. De andere kanalen gezamenlijk zouden beschouwd kunnen worden als één rustkanaal met spanningsbron E_{rest} (Goldman-Hodgkin-Katz-vergelijking [1]) en geleiding g_{rest}. De 'physiologist's convention' zegt daarbij dat de ionische 'rust'-stroom, I_{rest}, positief is wanneer positieve lading uit het neuron (of compartiment, zie verderop) stroomt. Het is bovendien een optie om de synaptische spanningen *ten opzichte van* E_{rest} te modelleren, oftewel te definiëren dat $E_{rest} = 0$.

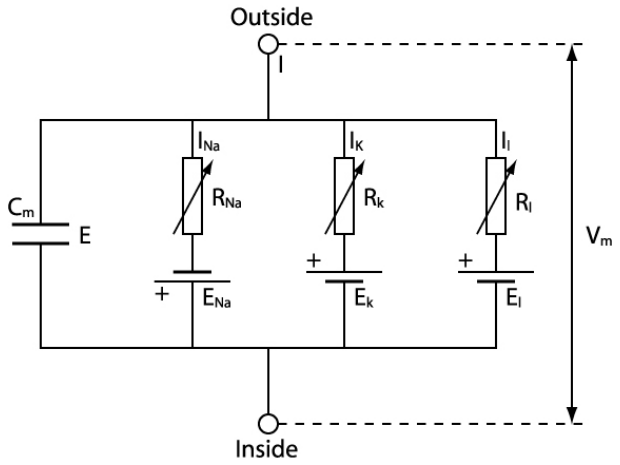
Nu werd figuur 2.5 geïntroduceerd als een vervangingsschema van een compleet neuron, wat ook een mogelijkheid is, maar een neuron zou in meer detail gemodelleerd kunnen worden wanneer de cel opgedeeld wordt in meerdere 'compartimenten' met allen hun eigen functie en werking, zie figuur 2.6. Wanneer een klein netwerk wordt geanalyseerd, is het nauwkeuriger om met meerdere compartimenten per cel te werken. Men kiest dan bijvoorbeeld voor twee compartimenten voor de dendrieten, één compartiment voor het soma, enzovoorts. Binnen het Hodgkin-Huxley model wordt voor iedere celsoort (SPyr, DPyr, Bask1, Bask2, Bask3 en Chand) een zekere compartimentale verdeling toegekend. Dit gaat bij NeuroSim, hoewel uiteraard vrij instelbaar, om zo'n 5 à 6 compartimenten per neuron. Zodra het netwerk erg groot wordt, zou overdacht kunnen worden wat er gebeurt in het geval van één compartiment per neuron, wat natuurlijk minder 'rekenwerk', maar ook minder nette resultaten levert.

Met behulp van de wet van Ohm en figuur 2.5 vinden we:

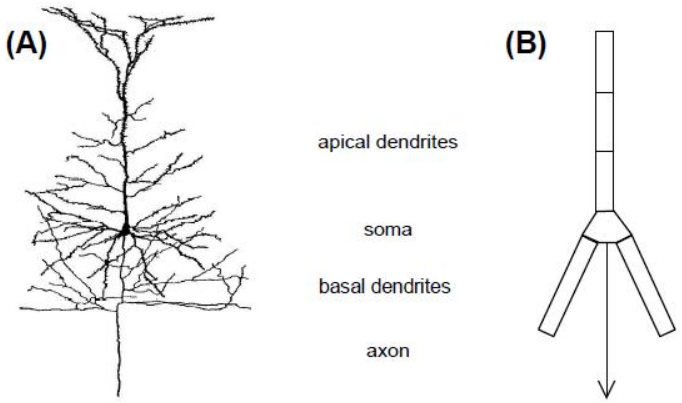
$$I_K = g_K(V_m - E_K); \tag{2.1}$$
$$I_{Na} = g_{Na}(V_m - E_{Na}); \tag{2.2}$$
$$I_L = g_L(V_m - E_L). \tag{2.3}$$

Door middel van analyse op het openen en sluiten (geleiding) van de ionkanalen, oftewel op het gedrag van g_K en g_{Na} waren Hodgkin & Huxley in staat een verzameling van niet-lineaire empirische vergelijkingen op te stellen:

$$I_K = g_K(V_m - E_K) = \overline{g_K} n^4 (V_m - E_K) \tag{2.4}$$
$$I_{Na} = g_{Na}(V_m - E_{Na}) = \overline{g_{Na}} m^3 h (V_m - E_{Na}) \tag{2.5}$$
$$I_L = g_L(V_m - E_L) \tag{2.6}$$



Figuur 2.5: Elektrisch vervangingscircuit van een neuroncompartiment, hier met een natrium-, een kalium-, en een leak-kanaal. Een synaptisch compartiment zou tevens een synaptisch kanaal bezitten. De capaciteit van de lipide bilaag van het celmembraan is gemodelleerd met de condensator C_m. De variabele geleiding van de ionkanalen wordt aangegeven met het pijlsymbool door de weerstanden. Met 'inside' en 'outside' wordt binnen, resp. buiten het neuron bedoeld.
(http://alford.bios.uic.edu/Images/586%20images/circuit%20model)



Figuur 2.6: Een complexe (pyramide) zenuwcel (met dendrieten, soma en axon) wordt gesimplificeerd (gemodelleerd) naar een compartimentaal neuronmodel.
(uit: [5])

$\overline{g_K}$ en $\overline{g_{Na}}$ zijn de maximum geleidingen van kalium en natrium, respectievelijk. n, m en h zijn 'gating variables' die het openen (en sluiten) van de 'voltage gated' kanalen beschrijven - liggen dus allemaal tussen 0 en 1. Het kaliumkanaal functioneert daarbij iets anders dan het natriumkanaal - gebaseerd op de genoemde experimenten. Het gedrag van n, m en h is tijdsafhankelijk en te beschrijven aan de hand van de volgende niet-lineaire 'gewone' (ordinary) differentiaalvergelijkingen:

$$\frac{dn}{dt} = \alpha_n(V_m)(1-n) - \beta_n(V_m)n \tag{2.7}$$
$$\frac{dm}{dt} = \alpha_m(V_m)(1-m) - \beta_m(V_m)m \tag{2.8}$$
$$\frac{dh}{dt} = \alpha_h(V_m)(1-h) - \beta_h(V_m)h \tag{2.9}$$

α_i en β_i zijn 'non-permissive to permissive (gesloten naar ontvankelijk / open)' en 'permissive to non-permissive' snelheidsconstantes, respectievelijk. Het zijn exponentiële functies van de membraanpotentiaal V_m. Hodgkin & Huxley hebben expliciete uitdrukkingen gevonden voor deze variabelen:

$$\alpha_n(V_m) = 0.01\frac{10 - V_m}{exp(\frac{10-V_m}{10}) - 1} \tag{2.10}$$
$$\beta_n(V_m) = 0.125exp(\frac{-V_m}{80}) \tag{2.11}$$
$$\alpha_m(V_m) = 0.1\frac{25 - V_m}{exp(\frac{25-V_m}{10}) - 1} \tag{2.12}$$
$$\beta_m(V_m) = 4exp(\frac{-V_m}{18}) \tag{2.13}$$
$$\alpha_h(V_m) = 0.07exp(\frac{-V_m}{20}) \tag{2.14}$$
$$\beta_h(V_m) = \frac{1}{exp(\frac{30-V_m}{10}) + 1} \tag{2.15}$$

Bovenstaande vergelijkingen voor n, m en h komen van een meer doorzichtige (maar lastiger te meten) schrijfwijze, zoals in de meeste tekstboeken te vinden, te weten:

$$\dot{p_i} = \frac{(p_{i\infty}(V_m) - p_i)}{\tau_{p_i}(V_m)} \tag{2.16}$$

met

$$p_{i\infty} = \frac{\alpha_{p_i}}{\alpha_{p_i}(V_m) + \beta_{p_i}(V_m)} \tag{2.17}$$
$$\tau_{p_i} = \frac{1}{\alpha_{p_i} + \beta_{p_i}} \tag{2.18}$$

en, evenals in 2.4 en 2.5:

$$g_k = \overline{g_k}\prod_i p_i \tag{2.19}$$

met g_k de normalisatieconstante, de constante die de geleiding bepaalt wanneer alle deelkanalen, geïndexeerd met i, (volledig) open zijn. k is de kanaalindex (natrium, kalium, ...). τ_{p_i} is de tijdsconstante voor deelkanaal i.

De complete verzameling Hodgkin-Huxley vergelijkingen leidt tot een differentiaalvergelijking, geldend per neuron:

$$C_m \frac{dV_m}{dt} = I_{in} - \overline{g_K} n^4 (V_m - E_K) - \overline{g_{Na}} m^3 h (V_m - E_{Na}) - ... - g_L (V_m - E_L) \tag{2.20}$$

Hierin is I_{in} de inkomende stroom vanuit andere compartimenten en/of een (eventuele) extern toegepaste (elektrode)stroom.

'Compartmental modeling' is het (numeriek) oplossen van vergelijking 2.20 voor ieder compartiment in het breinmodel. Een neuronaal simulatieprogramma lost een verzameling gekoppelde vergelijkingen op door de differentiaalvergelijking te vervangen door een discrete differentievergelijking die opgelost wordt op discrete tijdsintervallen. Dit kan expliciet en impliciet, waarbij in het algemeen de impliciete methode complexer, maar stabieler is. Een axon wordt geïmplementeerd als een eenvoudige vertraging in tijd voor de voortbeweging van AP's. Voor wat betreft deze AP's, die natuurlijk per cel gegenereerd worden en niet per compartiment of kanaal, zijn er twee bijzondere aspecten: het (eenvoudig te implementeren) 'alles-of-niets'-principe, oftewel het drempelprincipe voor de membraanpotentiaal met betrekking tot generatie van AP's, en de a.r.p., de 'absolute refractory period': kort na het genereren van een AP kan de betreffende zenuwcel niet opnieuw een prikkel genereren. Meesimulatie van deze a.r.p. in het model vereist meer werk: per kanaal moet een 'Event List' (lijst met 'momenten' van AP-generaties) worden bijgehouden.

2.3 Het Blue Brain Project

Het doel van het Blue Brain Project [6] was en is het ontwerpen van een computermodel dat zoveel mogelijk (in de wetenschap bekende) details meeneemt binnen de simulatie van het brein. Het Brain Mind Institute en IBM lanceerden het Blue Brain Project op 1 juli 2005, met als doel de hersenen van zoogdieren met een grote biologische precisie te modelleren. Het project was gebaseerd op het Hodgkin-Huxley model dat in de vorige paragraaf is besproken, met in het begin zo'n 20 compartimenten ('takken') per cel. Later werd de hoeveelheid compartimenten per neuron nog vele malen groter. Gebruik van minder compartimenten betekende (te veel) verlies van biologisch realisme: er bestond en bestaat een zekere minimale grootte van een microcircuit en een minimale complexiteit van de morfologie van een zenuwcel binnen de discretisatie. Het model is zó groot geworden dat het alleen op een speciale computer runt en dat het niet geschikt is voor analyse. Het Blue Brain Project runt op meerdere processoren en past dus parallelrekening toe.

De grootste begrenzingen voor computers binnen de simulatie van biologische processen zijn de vereiste tijds- en plaatsafhankelijke resolutie van neuronen en de limieten van de gebruikte algoritmen. Het menselijk brein is een opgerolde massa van ongeveer 40 bij 40 centimeter en bevat zo'n 20.000.000.000 zenuwcellen. Dit aantal is natuurlijk ontzettend groot voor simulatie (in genoemd detail) en er wordt daarom meestal slechts een klein stukje van de hersenen gesimuleerd. 'FPGA-based chips' hebben een hogere computatiesnelheid, waardoor het mogelijk in de toekomst een optie zal zijn om het *gehele* menselijk brein te simuleren, zelfs met de huidige technologie.

2.4 GENESIS; Kunstmatige Elektro-Encefalografie

Van Drongelen e.a. ontwierpen een beperkter (maar nog altijd zeer groot en geavanceerd) breinsimulatiemodel met het softwarepakket GENESIS [7][8], met als belangrijk doel het bestuderen van epilepsieverschijnselen. Het grootste verschil met het Blue Brain Project is dat GENESIS, dat staat voor *Gene Network Evolution Simulation Software*, het brein modelleert met veel minder compartimenten per neuron. De output is dus ook minder nauwkeurig, doch voldoende voor de gewenste analyse [5][9].

GENESIS modelleert het menselijk brein met uitsluitend de vier meest voorkomende celtypen, genoemd in paragraaf 2.1: SPyr-, DPyr-, Chand- en Bask-cellen, waarbij de Bask-cellen drie subtypen hebben. Ook GENESIS werkt met parallelrekening. In [8] wordt geconcludeerd dat gebruik van meerdere processoren significante (als niet ideale) versnelling betekent, oftewel 'bigger is better'. Het ging hier om een simulatie van maar liefst 18.500 neuronen met een insteltijd van

2000 seconden en een simulatietijd van slechts 400 seconden (voor 10.000 tijdstappen; dat is een hersensimulatie van 1 seconde).

'EEG' staat voor 'Elektro-EncefaloGrafie'. Wanneer een sensor op de hoofdhuid geplaatst wordt, dan meet deze een (gewogen) totaalsignaal van alle elektrische activiteit binnen de hersenen. Binnen GENESIS kunnen de elektrische stromen makkelijk opgeteld worden (over honderdduizenden cellen of een gedeelte daarvan), wat (op iedere tijdstap in de simulatie) in een zeker kunstmatig EEG-signaal resulteert. Dit EEG-signaal is eigenlijk één van de weinige simulatie-outputs die rechtstreeks vergeleken kunnen worden met de werkelijkheid, voor wat betreft de mens, maar laat niet veel details zien van de onderliggende activiteiten (op celniveau). Merk op dat bij het construëren van het kunstmatige EEG alleen de output van de SPyr-cellen invloed heeft. De Dpyr-cellen liggen te diep in de hersenen. Bovendien zijn de stromen in inhibitoire interneuronen in het algemeen zwakker dan de stromen in de excitatoire piramidecellen. Een ander praktisch meetsysteem voor de activiteit binnen neuronen is 'MEG', 'Magneto-EncefaloGrafie'.

GENESIS modelleert de Neocortex in de zes horizontale lagen als geïntroduceerd in deelparagraaf 2.1.2 en met verticale rangschikking in 'modules' of '(micro-)kolommen'. De ruimte tussen piramidecellen is ongeveer 5 μm, de verhouding tussen piramidecellen en interneuronen ('inhibitors') is $\sim$4.5:1, en de celdichtheid is ongeveer 100.000 cellen per kubieke millimeter hersen (hoewel deze celdichtheid natuurlijk varieert over de diepte(lagen)).

2.5 C++-implementatie

Om netwerkgedrag van neuronen ten gevolge van verandering van parameters nóg een stapje flexibeler te kunnen beschouwen, is het neuronaal model van de hersenen uit GENESIS herbouwd in C++-programmeertaal ('NeuroSim'). Het programma functioneert principieel hetzelfde als GENESIS, is dus ook gebaseerd op het beschreven Hodgkin-Huxley model, en produceert output voor de membraanpotentialen en -stromen van een instelbare hoeveelheid excitatoire en inhibitoire zenuwcellen voor een zekere tijdsperiode. De C++-implementatie werkt met 'classes' ('klassen') om op een handige manier de cellen, kanalen, verbindingen, 'events' (momenten van AP-generaties per cel), enzovoorts, op te slaan. Voor geheugendoeleinden wordt gewerkt met 'pointers' (zeg: adresgegevens of geheugenlocaties) en om snel lijsten te kunnen bewerken wordt het principe van de zgn. 'Linked List' toegepast. 'Classes', 'pointers' en 'Linked Lists' laten we hier onbeschreven; dit zijn programmeertechnische termen. Ook wordt niet op de complete C++-code ingegaan. Er is echter een selectie aan achtergrondinformatie betreffende de implementatie te vinden in bijlage A.

2.6 Van Single Neuron modellen naar populatiemodellen; output & ondervindingen

Met het Hodgkin-Huxley model kan het gedrag van een neuronennetwerk zeer gedetailleerd gemodelleerd worden. Met behulp van parallelrekening is het mogelijk dit netwerk groot te maken en een behoorlijk gedeelte van de Neocortex te simuleren binnen afzienbaar tijdsbestek. Ieder virtueel neuron heeft (voor zover van toepassing) zijn eigen (instelbare) eigenschappen en er is een behoorlijke hoeveelheid aan fysiologische parameters die keurig in het Hodgkin-Huxley model te verwerken en dus te variëren zijn. Het model laat herkenbare (lees: vanuit realistische breinmetingen weerlegbare) output zien. Dit is meteen de belangrijkste vorm van modelvalidatie: het totaalgedrag (somsignaal, niets meer en niets minder) van een grote groep virtuële neuronen als output van een computationeel model vertoont veel overeenkomsten met medische EEG's, gemeten met op het hoofd geplaatste elektroden. Breinactiviteit gedurende een epileptische aanval, bijvoorbeeld, is overtuigend weerlegbaar met computermodellen. In dit geval vuurt een grote groep neuronen groepsgewijs actiepotentialen, allen op ongeveer hetzelfde moment.

Nadelen van het gedetailleerde Single Neuron Hodgkin-Huxley model zijn echter júíst de vele details; het is zeer moeilijk (zo niet, dan onmogelijk) bepaalde vormen van analyse uit te voeren

op het model, dat wil zeggen: de complete output van het model enigszins wiskundig te kunnen voorspellen. Bovendien zal het simuleren van een complete Neocortex opgedeeld in Single Neurons altijd een gigantische rekenklus blijven.

Eigenlijk is het, hoewel afhankelijk van waarnaar precies computationeel onderzoek wordt gedaan, volstrekt onbelangrijk wat er precies met één afzonderlijk neuron gebeurt. We zullen op dit moment veel meer belangstelling hebben voor de vraag hoe een grote groep neuronen zich in zijn geheel gedraagt, ook wanneer we onderzoek doen naar bijvoorbeeld epilepsie. Het totaalgedrag is (voorlopig) immers het enige vergelijkingscriterium tussen een medisch EEG en een computationeel EEG over een virtueel neuronennetwerk. Zijn Single Neuron modellen misschien wat 'overdone' en kunnen we door gehele populaties van neuronen te modelleren, wat in het algemeen veel minder rekenwerk zou kosten, dezelfde output voor het totaalgedrag van een neuronennetwerk verkrijgen? Het antwoord is ja, zo zullen we in deze paragraaf zien. De indeling van de neuronengroepen ('populaties') is hierbij gebaseerd op neurontype (excitatoir/inhibitoir) en eventueel, bij een grootschalig model, locatie van de neuronen. Onder meer totaalactiviteit binnen een populatie als functie van zekere input en verplaatsingen van AP's kunnen door populatiemodellen duidelijk in kaart gebracht worden. Er worden per neuronenpopulatie differentiaalvergelijkingen opgesteld en is er sprake van slechts een paar modelparameters, met weinig fysiologische relevantie, in tegenstelling tot de vele (wél fysiologische) parameters in Single Neuron modellen. Populatiemodellen laten enige theoretische analyse toe, waaronder bifurcatie-analyse.

Onderzoek aan populatiemodellen is momenteel dus zeer voornaam en zal bondig behandeld worden in deelparagraaf 2.6.2. In deelparagraaf 2.6.1 wordt specifiek het zgn. 'Integrate-and-fire' model besproken, als het meest simpele neuronreplica en tevens vergelijkingscriterium voor uitgebreidere modellen en populatiemodellen. In de gehele paragraaf 2.6 komen bovendien de belangrijkste neuronennetwerkverschijnselen als output van veelvuldig uitgevoerde experimenten met zowel NeuroSim als andere neuronale simulatiecodes naar voren.

Deze paragraaf bevat dus informatie over allerhande computationele neurofysica, waarbij verwezen wordt naar diverse literatuur.

2.6.1 Het 'Integrate-and-fire' model; Benadering AP firing rate

Bij Single Neuron Modeling wordt elk afzonderlijk neuron individueel beschouwd, zoals besproken in het begin van dit hoofdstuk. Ieder neuron kan zijn eigen output produceren voor bijvoorbeeld de membraanpotentiaal, de membraanstroom en de momenten (tijdstippen) van actiepotentiaalvuring. Men kiest dus minimaal één compartiment per neuron. Een elementair 'Single Neuron Single Compartment model' (1 compartiment per neuron) kan bijvoorbeeld gebruikt worden om de relatie tussen synaptische inputs en de 'firing rate' (frequentie van AP-vuring) van een neuron of een groepje neuronen te onderzoeken [1]. In het 'Integrate-and-fire (Single Neuron Single Compartment)' model (figuur 2.7) wordt onder meer aangenomen dat er geen actieve membraangeleidingen aanwezig zijn (onder meer geen Hodgkin&Huxley-kanalen). Het virtuële neuron bestaat slechts uit een weerstand en een condensator, die parallel geschakeld zijn. Dit model valt slechts terug op het drempelprincipe voor wat betreft actiepotentialen. Het standaard Integrate-and-fire model is in 1907 geïntroduceerd door Lapicque. Inderdaad, dat was al ver vóór het moment dat het Hodgkin&Huxley model zijn intrede deed.

Kirchoff geeft, beschouwende figuur 2.7:

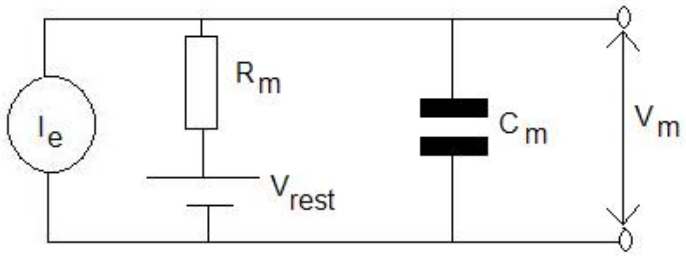
$$I_e = \frac{V_m - V_{rest}}{R_m} + C_m \frac{dV_m}{dt}. \tag{2.21}$$

Deze (differentiaal)vergelijking valt te herschrijven in:

$$\tau_m \frac{dV_m}{dt} = -V_m + V_{rest} + I_e R_m, \tag{2.22}$$

met $\tau_m = R_m C_m$ de zgn. membraantijdsconstante. Oplossen resulteert in:

$$V_m(t) = I_e R_m (1 - exp(-t/\tau_m)) + V_{rest}. \tag{2.23}$$



Figuur 2.7: Integrate-and-fire model: 1 (eenvoudig) compartiment per neuron en daarmee de meest simpele representatie van een zenuwcel.

V_{rest} is de waarde van de membraanpotentiaal op tijdstip $t = 0$ ('het moment dat de condensator zal beginnen met opladen'), oftewel de rustmembraanpotentiaal. We zijn geïnteresseerd in de 'firing rate' (aantal gevuurde AP's per seconde) van dit neuron, gegeven een constante inputstroom I_e. Stel: $t = t_1$ is het tijdstip waarop, na een AP-vuring, V_m de reset-waarde V_{reset} bereikt:

$$V_{reset} = I_e R_m(1 - exp(-t_1/\tau_m)) + V_{rest}. \tag{2.24}$$

Het tijdstip waarop er op z'n vroegst opnieuw een AP gevuurd kan worden, $t_2 = t_1 + T$, vindt nu plaats op het moment dat V_m gelijk is aan V_{th}, de drempelwaarde:

$$V_{th} = I_e R_m(1 - exp(-t_2/\tau_m)) + V_{rest}. \tag{2.25}$$

Dit geeft:

$$T = t_2 - t_1 = -\tau_m \cdot ln(\frac{-V_{th} + V_{rest} + I_e R_m}{I_e R_m}) + \tau_m \cdot ln(\frac{-V_{reset} + V_{rest} + I_e R_m}{I_e R_m}), \tag{2.26}$$

wat gelijk is aan:

$$T = \tau_m \cdot ln(\frac{-V_{reset} + V_{rest} + I_e R_m}{-V_{th} + V_{rest} + I_e R_m}). \tag{2.27}$$

De (maximale) AP firing rate f_r is de reciproke van deze (absoluut refractaire) periode:

$$f_r = \frac{1}{T} = \frac{1}{\tau_m}(ln(\frac{-V_{reset} + V_{rest} + I_e R_m}{-V_{th} + V_{rest} + I_e R_m}))^{-1}. \tag{2.28}$$

Uitdrukking 2.28 is alleen geldig als $I_e R_m > V_{th} - V_{rest}$, anders $f_r = 0$. Gebruik de linearisatie $ln(1 + z) \sim z$ voor $z << 1$ en vind:

$$f_r = \frac{1}{\tau_m}(ln(\frac{-V_{reset} + V_{rest} + I_e R_m - V_{th} + V_{th}}{-V_{th} + V_{rest} + I_e R_m}))^{-1} \sim \frac{V_{rest} - V_{th} + R_m I_e}{\tau_m(V_{th} - V_{reset})}. \tag{2.29}$$

Voor grote I_e groeit de AP firing rate dus lineair met de inputstroom. Natuurlijk is in werkelijkheid I_e een somstroom van meerdere outputs van andere (presynaptische) neuronen (en zal I_e in het algemeen niet constant zijn). Het Integrate-and-fire model laat zien dat neuronen de sterkte van een prikkel vertalen in de frequentie van de verzonden AP's.

Voor gedetailleerde simulaties, zoals die binnen GENESIS en NeuroSim, is het Integrate-and-fire model niet voldoende. Het gedrag van de neuronen is feitelijk niet gedetailleerd genoeg; daar zijn meerdere compartimenten per neuron voor nodig. Echter, in de stap naar populatiemodellen wordt output van een eenvoudig Integrate-and-fire model vaak gebruikt als vergelijkingsmateriaal - populatiemodellen geven immers al helemáál geen gedetailleerd beeld van de neuronen op zichzelf.

Een voorbeeld van een dergelijke vergelijking is het werk van Nykamp en Tranchina in 1999 [10], dat we in paragraaf 2.6.2. en 2.6.4. opnieuw zullen aanhalen.

Op dit moment beschouwen we het model dat beschreven is in deze deelparagraaf als de meest simpele replica van een neuron. Rangan en Cai ontwierpen snelle numerieke methoden om op grote schaal berekeningen uit te voeren over netwerken met Integrate-and-fire neuronen [11]. Hun zgn. 'Arbored Coupling Method' bevat onder meer een integratiefactor (gebaseerd op de neurofysiologie) die het mogelijk maakt de ontwikkelingen per neuron te beschouwen, zonder last van tijdstaprestricties die veroorzaakt zijn door eventuele instabiliteiten in het model.

2.6.2 Populatiegedrag en -modellen; Oscillaties

In de vorige deelparagraaf vonden we een uitdrukking voor de firing rate van een enkel neuron bij een zekere constante inputstroom. In realiteit, kijkende naar een neuron in een netwerk, zullen inputstromen niet constant zijn. Het gedrag van twee verbonden neuronen valt in te denken: neuron 1 vuurt met regelmaat (frequentie f_r) AP's naar neuron 2, dat de AP's, eventueel met een iets andere frequentie (instellingen mogen, en zullen, natuurlijk verschillen per neuron), maar wel met (ongeveer) dezelfde amplitude, terugzendt naar neuron 1, dat op zijn buurt weer reageert op deze input. Mogelijk synchroniseren de twee neuronen hun vuringen. Op deze manier kan ook een complete groep met neuronen gezamenlijk ritmisch gedrag laten zien. Gray toonde in 1994 aan dat oscillaties in neuronale systemen zich kunnen synchroniseren binnen het gehele brein van een zoogdier [12]. Het gedrag van een klein groepje neuronen in een menselijk brein kan soortgelijk zijn (zie bijvoorbeeld [13] en [14]) en ook computermodeloutput bevestigt deze neiging van synchronisatie. De frequenties en ritmen van groepsgewijze oscillaties zijn veelvuldig geanalyseerd aan de hand van (onder meer) EEG-plots, gebaseerd op experimentele realistische output en modeloutput (zie bijvoorbeeld [15]), en onderverdeeld in zekere ritmen [16]; de belangrijkste zijn het alfa-ritme ($\sim$10 Hz, voor een wakker persoon, maar in zeer ontspannen toestand), het beta-ritme ($\sim$16-38 Hz, voor een persoon in gespannen of nerveuze toestand: intense activiteit van het zenuwstelsel) en het gamma-ritme ($\sim$38-80 Hz, sterke mentale activiteit: waarneming of angst). De beta- en gamma-ritmen komen veelvuldig voor in het brein ten tijde van attentie, perceptie en herkenning. Andere voorbeelden van ritmen zijn het theta-ritme ($\sim$4-8 Hz, ook wel het 'creatieve' ritme genoemd: komt vaak voor vlak voor het slapen gaan of bij het wakker worden) en het delta-ritme ($\sim$0.5-4 Hz, vooral voorkomend bij baby's in diepe slaap). Bij nog lagere EEG-ritmen is er sprake van de 'hersendood': het brein functioneert niet meer als normaal, maar laat inderdaad nog wel groepsgewijze elektrische pulsen zien (later zal hiernaar worden gerefereerd als 'het stilte-regime'). Oscillaties in het (bijna) complete bereik 12-80 Hz blijken vooral gelinkt te zijn aan het gedrag van de inhibitoire interneuronen in het centraal zenuwstelsel [17].

Let op: het is wat betrekkelijk om over dergelijke ritmen te praten, want belangrijk is daarbij op hoeveel neuronen precies gedoeld wordt. Een grote groep neuronen dat samen een gamma-ritme vertoont, is abnormaal te noemen. Een gezamenlijk ritme van een grote groep neuronen is doorgaans het geval tijdens een epileptische aanval. Het EEG van een patiënt tijdens een epileptische aanval toont vaak abnormale 'bursting' activiteiten (een 'burst' is een snelle opeenvolging van gevuurde actiepotentialen). Een kleine groep neuronen (zeg, $\sim$700) zal bij een gezond persoon juist geregeld gamma-ritmen laten zien. Beta-ritmen synchroniseren dan ook anders dan gamma-ritmen [18]. De beta-frequenties zijn in staat om zich over grote afstanden uit te breiden, wat correspondeert met signalen die zich een significante afstand in het brein verplaatsen. Dit lange-afstandsverschijnsel wordt met gamma-ritmen niet bereikt, omdat er tijd benodigd is om de actiepotentialen te verspreiden in de ruimte. Ritmisch vuren over een grote populatie neuronen is dus wel mogelijk, maar met wat vertraging tussen de verschillende delen van het brein (model). Ritmisch vuren van een groepje neuronen heet 'epileptiform' gedrag (wat dus in feite niets heeft te maken met de ziekte epilepsie!). Het gamma-domein [19] bevat de meest centrale (meest voorkomende) ritmen op een klein stukje hersen.

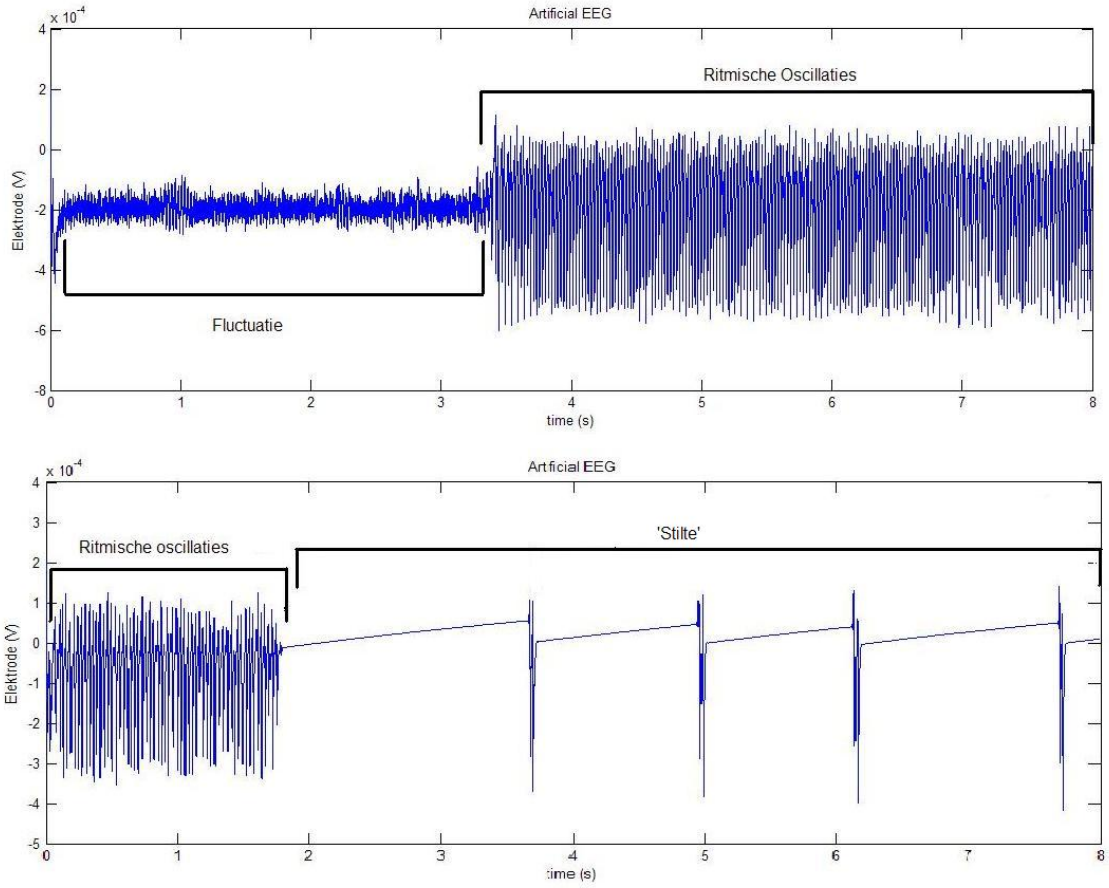
Dynamica van populaties in het model kunnen over het algemeen onderverdeeld worden in drie regimes: random fluctuaties (geen synchrone vuring), de zojuist besproken ritmische groepsgewijze oscillaties, en stilte (géén vuringen, met uitzondering van groepsgewijze pulsen met relatief zeer

lage frequentie) [20] (zie figuur 2.8 voor bijbehorende voorbeelden gegenereerd met NeuroSim). Natuurlijk zijn er veel meer regimes te definiëren aan de hand van bijvoorbeeld de frequentie van eventuele populatie-oscillaties, het type golven (lopend, spiraliserend, etc.), die we hier buiten beschouwen laten. Wanneer de excitatie-/inhibitieverhouding van het netwerk (waarop o.a. hoofdstuk 3 zal ingaan) langzaam verlaagd wordt, en het totaalnetwerk dus meer inhibitoir wordt, gaat het grote-schaal netwerkgedrag van het fluctuërende regime via het ritmische regime naar het stilteregime. Prince & Jacobs stelden in 1998 al de hypothese dat een verhoogd aantal inhibitoire of een verlaagd aantal excitatoire synapsen het netwerk in het epileptische regime kan brengen [21] en dat bevestigen meerdere modellen, waaronder ook NeuroSim. GENESIS-output geeft, voor 1000 neuronen, een excitatie-inhibitie-vlak zoals weergegeven in figuur 2.9, afkomstig uit [7]. In dit vlak zijn de verschillende gedragsregimes zichtbaar als functie van de globale excitatie en inhibitie in het netwerk, maar Van Drongelen e.a. geven in [7] al duidelijk aan dat bij dezelfde parameters zich verschillende scenario's kunnen afspelen - zie ook deelparagraaf 2.6.3.

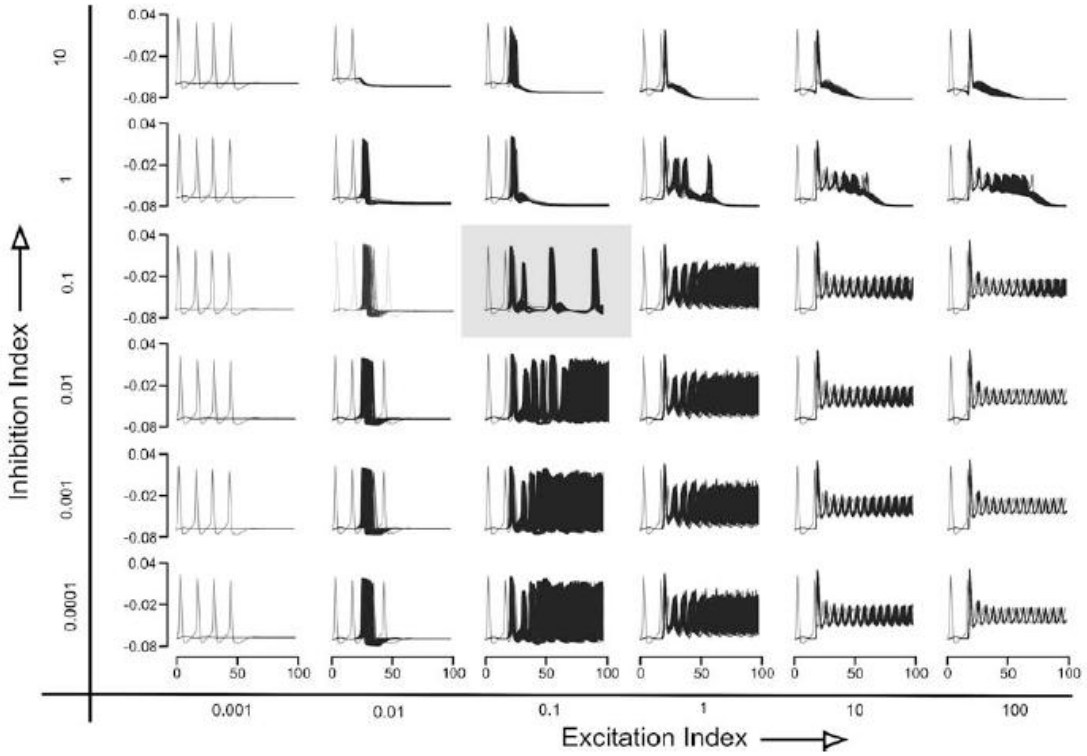
De hoeveelheid gevuurde actiepotentialen binnen een groep neuronen, oftewel de activiteit binnen een populatie, is het belangrijkste onderzoeksgebied van een populatiemodel; deze activiteiten zijn dan ook juist de eigenschappen die in een populatiemodel beschreven worden met vergelijkingen. Bij Single Neuron modellen kan de gemiddelde 'firing rate' bepaald worden en bij populatiemodellen wordt de 'population firing rate' beschouwd. Neuronenpopulatiemodellen stammen uit 1972, uit ideeën van Wilson & Cowan [22](zie ook [23]) (voor een recent artikel in de context van epilepsie: [24]). Een hoeveelheid excitatoire neuronen wordt gedefinieerd en $E(t)$ wordt de proportie neuronen hieruit die AP's vuren per tijdseenheid, op tijdstip t genoemd. $I(t)$ is de proportie per tijdseenheid van een groep inhibitoire neuronen die AP's vuren, op tijdstip t. De toestand $E(t) = I(t) = 0$ is de rusttoestand: een toestand waarin uitsluitend enige achtergrondactiviteit voorkomt. Naar $E(t)$ en $I(t)$ wordt gerefereerd als de 'totale' activiteiten in de respectievelijke subpopulaties. Het blijkt inderdaad dat met slechts twee (partiële) differentiaalvergelijkingen al excitatoire en inhibitoire activiteiten verklaard kunnen worden (bijvoorbeeld in [25] voor de 'virtual cortex'). Huidig onderzoek binnen neuropopulatiemodellen is onder meer het uitzetten van minimaal twee populaties (E en I) met beide hun eigen (doch gekoppelde) differentiaalvergelijking. Minimaal, inderdaad, want er geldt: hoe meer gemodelleerde populaties met neurontypen, des te realistischer de output wordt. De differentiaalvergelijkingen kunnen zgn. 'sigmoïdale' functies (een speciale vorm van de logistieke functie) bevatten (recent voorbeeld: [26]). In tegenstelling tot 'neural-mass models', die uitsluitend de gemiddelde activiteit van neuronale populaties beschouwen, zijn er ook zgn. 'mean-field models' ('population density models'), die de distributie van populatie-activiteit (dus de verschillen in activititeit over de ruimte bínnen een populatie) in beeld brengen, toegepast en beschouwd (actueel voorbeeld: [27]). Zie ook figuur 2.10, deze figuur koppelt de resultaten van een Single Neuron model aan die van een 'population density model'.

Recentelijk (2010) onderzocht onder meer Coombes het brein op grote schaal [28]. Met het model van Liley e.a. [29] werden ritmen gesignaleerd die consistent waren met de ritmen in het menselijke EEG-spectrum. Wendling e.a. introduceerden, net als Liley, in 2002 een 'neural mass model' [30], recentelijk (2009) verder uitgewerkt door Ursino [31], die een vierde populatie (een $GABA_{fast}$-interneuronpopulatie) toevoegde, waardoor de resultaten nog een stap realistischer werden. De staat van ontwikkeling in neuropopulatiemodellen momenteel is dus (al..) het stapsgewijs toevoegen van nieuwe populaties aan de bestaande verzameling populaties.

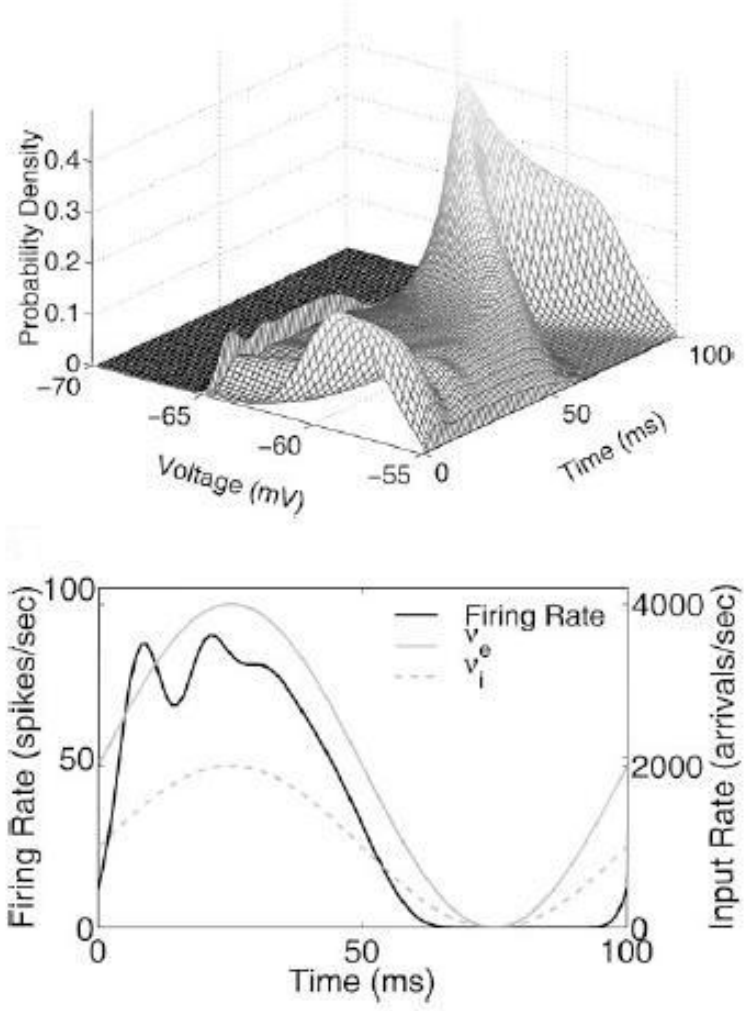
Kenmerkend voor populatiemodellen is de zgn. 'neural oscillator' [32]. Een neural oscillator is een paar van op elkaar inwerkende excitatoire (E) en inhibitoire (I) populaties. De term 'oscillator' is wat verwarrend, want de bijbehorende EEG's hoeven helemaal geen (groepsgewijze) oscillaties te laten zien. Het ritmisch gedrag varieert van de evenwichtstoestand (geen oscillaties) tot chaotisch dynamisch gedrag. De term 'oscillator' duidt in dit geval uitsluitend op de interacties tussen de E-populatie(s) en de I-populatie(s).



Figuur 2.8: Afbeelding van de drie belangrijkste verschillende typen neuronennetwerkgedrag. Het betreft 8-seconden-EEG's over 656 neuronen, gegenereerd door NeuroSim. De bovenste afbeelding bevat een synchronisatie: vanaf ∼3.2 seconden vuren de 656 neuronen synchroon actiepotentialen, vóór die tijd vuren ze a-synchroon. De onderste afbeelding bevat stiltegebieden waarbij de hele populatie 'niets' doet: er worden in het gehele netwerk geen AP's gevuurd, met uitzondering van de zichtbare groepsgewijze pulsen. Dit gebeurt met een zeer lage frequentie en in de tekst wordt dit gedrag gekoppeld aan een 'hersendood' persoon. Bovenste afbeelding toont typisch excitatoir (vanaf ∼3.2 seconden epileptiform) gedrag ('veel AP's'), onderste afbeelding typisch inhibitoir gedrag ('weinig AP's').



Figuur 2.9: Excitatie-/inhibitievlak, afkomstig uit [7], gebaseerd op GENESIS-output voor een netwerk van 1000 neuronen. Horizontaal staat een maat voor de globale excitatie, verticaal een maat voor de globale inhibitie in het netwerk. Weergegeven zijn (korte) EEG-metingen (over de SPyr-cellen) bij de betreffende globale excitatie- en inhibitieparameters. In het vlak zijn verschillende gedragsregimes zichtbaar. Merk op dat deze afbeelding niet 'heilig' is: er zijn *bij dezelfde parameters* meerdere scenario's mogelijk. Dit is de multistabiliteit, waarop we terug zullen komen in paragraaf 2.6.3.



Figuur 2.10: De AP firing rate wordt gekoppeld aan de kansdichtheid van de membraanpotentiaal over de neuronen. Bron: [10]. De bovenste afbeelding toont de verdeling van neuronen over de membraanpotentiaaltoestanden, als functie van de tijd. De onderste afbeelding toont de resulterende 'firing rate' van de populatie (linkerschaal) en de synaptische 'input rates' $v_e(t)$ en $v_i(t)$ (rechterschaal).

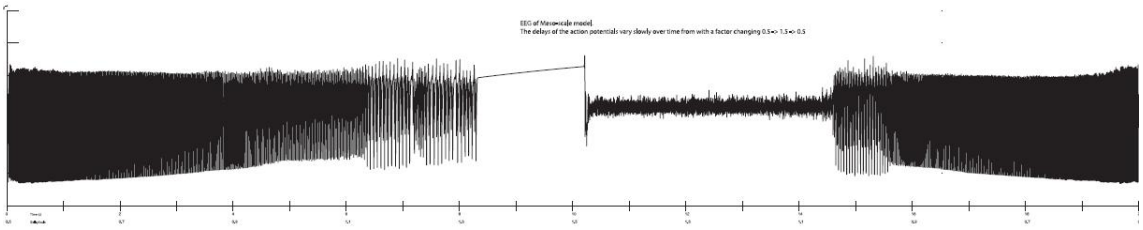
2.6.3 Multistabiliteit

Nu we drie duidelijke gedragsregimes hebben gedefinieerd (die overigens weer onder te verdelen zijn), kunnen we ons afvragen bij wélke parameters wélk groepsgedrag vertoond wordt. Experimenten met Single Neuron modellen & populatiemodellen en theoretische analyses bij populatiemodellen [33] laten zien dat deze vraag niet eenduidig beantwoord mag worden. Bij exact dezelfde parameters kan het neuronengroepsgedrag in verschillende regimes terechtkomen, afhankelijk van de begincondities. Er is sprake van multistabiliteit. Figuur 2.11, de EEG-output (NeuroSim) bij langzame variatie van de gemiddelde AP-reistijd van kort naar lang en weer terug, laat dit duidelijk zien. Het parameterverloop is in deze afbeelding symmetrisch ten opzichte van het midden van de simulatie, maar het EEG is dit duidelijk niet. Theoretische analyse bij populatiemodellen houdt bifurcatie-analyse in, waarin o.a. Izhikevich zich de laatste jaren verdiept heeft [34].

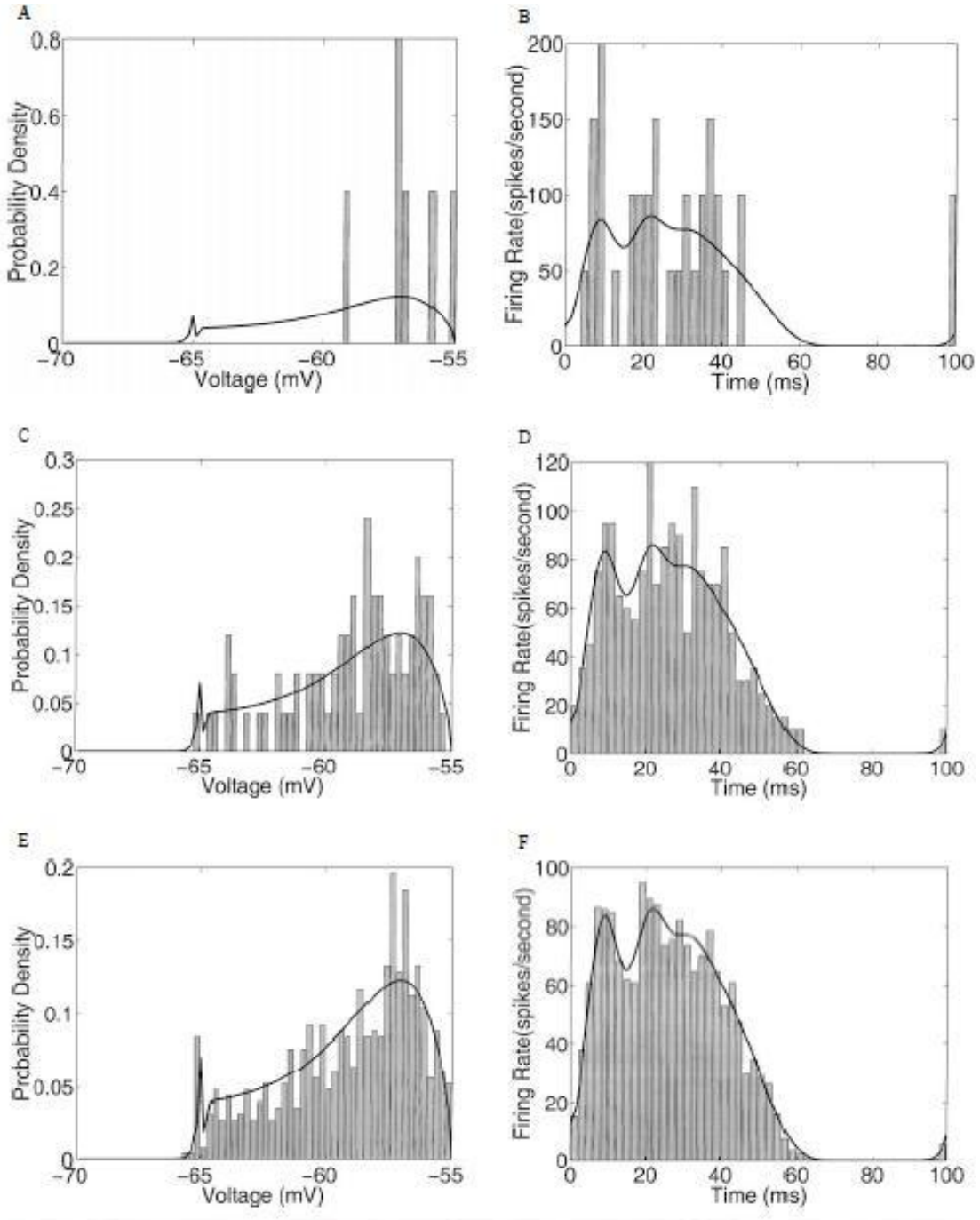
2.6.4 Samenvattend...

Complexe Single Neuron modellen versus 'simpele' populatiemodellen, die, voor wat betreft het groepsgedrag, exact dezelfde resultaten weten te leveren. Twee heel verschillende filosofieën eigenlijk, die momenteel nog steeds nader vergeleken worden (voorbeeld: [35] en, mede uitgelijnd in figuur 2.12, [10]). Wat is nu de link tussen beide soorten modellen? Wanneer we het voorafgaande samenvatten, merken we dat Single Neuron modellen, aan de hand van Hodgkin & Huxley, al veel eerder uitgediept zijn dan populatiemodellen. Validatie van uitgebreide Single Neuron modellen is zuiver gebeurd aan de hand van medische EEG's, oftewel aan de hand van het groepsgedrag uit de realiteit. Medische EEG's vertoonden dezelfde verschijnselen als computationele EEG's, óók tijdens een aanval van epilepsie. Bij dat laatste oscilleerden de (al dan niet virtuéle) neuronen groepsgewijs met een zeker ritme. De gedragsregimes die in deze deelparagraaf naar voren zijn gekomen, en zéker dit groepsgewijze gedrag, zijn veelvuldig geanalyseerd, zowel medisch als in computationele vorm. Wanneer belandde het neuronensysteem in het oscillerende, dus epileptische, regime? Als op deze vraag een concreet antwoord gevonden zou zijn, zou misschien op een zekere manier (bijvoorbeeld met behulp van zgn. stimuli) in de praktijk een epileptische aanval voorkomen kunnen worden!

Aan de hand van het groepsgedrag in Single Neuron modellen zijn modellen opgesteld die dit groepsgedrag met slechts een paar differentiaalvergelijkingen beschrijven, ..in plaats van met miljoenen of misschien wel miljarden. De sleutel tussen de twee modellen was dus voornamelijk dit genoemde groepsgedrag, oftewel de totaalactiviteit binnen een neuronenpopulatie, of, nog concreter: het aantal gevuurde actiepotentialen binnen een neuronenpopulatie per seconde, genormeerd naar de grootte van de populatie.



Figuur 2.11: EEG bij 656 neuronen, gegenereerd door NeuroSim. Tijdens de simulatie wordt de (gemiddelde) reistijd van actiepotentialen in het netwerk, oftewel de delays in het model, langzaam gevarieerd van 1.5 maal naar 0.5 maal het standaardgemiddelde en weer terug. Op de helft van de simulatie vindt dus een keerpunt plaats in de simulatie en is de gemiddelde reistijd gelijk aan 0.5 maal de gebruikelijke gemiddelde reistijd. In het geval zonder hysterese zou men dus symmetrie verwachten rond dit punt. Op de horizontale as staat de simulatietijd weergegeven, en dus ook de delay-parameter lopende van 1.5 naar 0.5 naar 1.5. Op de verticale as staat de waarde van de gewogen somspanning over alle SPyr-cellen, als bij een normale (kunstmatige) EEG-meting. We zien het groepsgedrag van het oscillerende regime naar het stilteregime gaan, waarop een fluctuerend regime en vervolgens weer een oscillerend regime volgt. De oscillerende regimes zijn bovendien te scheiden in twee gedeelten, waarvan het ene gedeelte ongeveer een dubbele periode heeft ten opzichte van het andere gedeelte. In deze afbeelding is geen symmetrie te onderscheiden. De afbeelding is een bewijs van het feit dat een neuronennetwerksysteem zich bij bepaalde parameters in meerdere toestanden kan bevinden, afhankelijk van de begincondities. Er is sprake van hysterese. Met nog andere woorden: het systeem moet in een bepaalde toestand *gebracht* worden.



Figuur 2.12: Vergelijking output Single Neuron model en populatie(dichtheids)model: aan de hand van activiteiten ('firing rates'). Bron: [10]. Input: links, output: rechts.

Hoofdstuk 3

Verbindingsparameters

3.1 Toelichting

Figuur 3.1 is een tabel afkomstig uit [4] die de verbindingsinformatie laat zien waarmee gewerkt is binnen de GENESIS- en originele C++-implementatie van het Hodgkin-Huxley model. De parameters in de tabel zijn gebaseerd op gegevens uit de ('medische') literatuur en bestemd voor een neuronennetwerk van ca. 700 neuronen. Dit aantal is vast: vergroten van het netwerk met behoud van de parameters in de tabel zal afwijkende resultaten geven. Dit is hetgene waar dit hoofdstuk zich op zal richten; gewenst is de afhankelijkheden van de parameters in de verbindingstabel in beeld te krijgen, voor variabele netwerkafmetingen. Het gaat in deze tabel om de Neocortex van een muis. In het algemeen is iedere Neocortex en dus ieder mens of dier uniek, vandaar enige randomness in de netwerkparameters.

De eerste kolom van de tabel bevat een referentienummer van het type verbinding die weergegeven staat in de tweede kolom. Omdat er 6 soorten cellen gebruikt worden (SPyr, DPyr, Bask1, Bask2, Bask3 en Chand), zijn er in theorie $6 \cdot 6 = 36$ verschillende soorten verbindingen (cellen mogen ook met een cel van een gelijke soort verbinden - ofschoon, binnen de huidige implementatie van NeuroSim, niet met zichzelf). Echter, Chandl-cellen verbinden uitsluitend met SPyr- & DPyr-cellen en de SPyr→SPyr- en SPyr→DPyr-verbindingen kennen twee varianten. 'Exc' = Excitatoir, 'Inh' = Inhibitoir, 'Dinh' = Disinhibitoir (reductie op inhibitoire effecten - gedrag is meer impulsief, noem het voor het gemak ook 'Inhibitoir'. De disinhibitoire verbindingen zijn de verbindingen tussen de interneuronen). De volgende kolom, *Destination*, D, beschrijft de straal om de zendende cel heen (dimensie meters, uitsluitend gebaseerd op de (x, y) bovenaanzichtslocatie van de neuronen (2D), Euclidiaans) waarbinnen deze verbindingen (van het betreffende type) kan maken met andere cellen. Buiten deze straal zal dit niet gebeuren, i.e.: twee cellen die in bovenaanzicht verder dan de waarde in de betreffende rij van de kolom *Destination* uit elkaar liggen, zullen geen verbinding maken. 'Source hole' is een 'gat' in het D-gebied, waarbinnen de cel alsnog geen synapsen maakt met andere cellen (momenteel nog niet geïmplementeerd in het C++-model). Binnen de gedefinieerde verbindingscirkel om een neuron heen neemt de kans op een verbinding exponentieel af met de afstand tussen de cellen. De startkans en de exponentiële afname waarmee dit gebeurt staan weergegeven in de kolommen *Probability* en *Exponential*. Een Bask1→SPyr-verbinding, bijvoorbeeld, heeft 4% kans op bestaan wanneer de Bask1-cel en de SPyr-cel zich op 0 (micro)meter afstand van elkaar bevinden, en neemt dan met een 'tempo' gedefinieerd als 300 μm af naarmate de afstand tussen deze cellen groeit. Netter gezegd: het kansverloop is gedefinieerd als: $P(x, y) = P_{max} \cdot e^{(-\rho_2(x,y)/E_P)}$, met $\rho_2(x, y)$ de rechtlijnige (oftewel de kortste - Euclidische) afstand in horizontale richting (2D), zolang $\rho_2(x, y) \leq D$. P_{max} en E_P zijn de waarden uit de kolommen *Probability* en *Exponential*, respectievelijk. De kolom *Velocity* bevat de voortbewegingssnelheden van actiepotentialen. De snelheid is eigenlijk altijd 0.08 m/s (met, in GENESIS, een eventuele afwijking van maximaal 25%), met uitzondering van de tweede soort SPyr→Spyr- en SPyr→Dpyr-connecties. Ook deze twee verbindingstypen zijn

Overview of the connectivity parameters in the simulation

No. and Ref.	Synapse	Destination, μm	Source Hole, μm	Probability	Exponential ln x, y only, μm	Velocity $\pm$, m/s	max W	min W	Exponential ln 3D, μm
1	Exc: S_PYR → S_PYR	500	5, 5, 1500*	0.10	300	0.08 (25%)	6	0	200
2	Exc: D_PYR → S_PYR	500	—	0.10	300	0.08 (25%)	6	0	1,500
3	Exc: S_PYR → D_PYR	500	—	0.10	300	0.08 (25%)	6	0	1,500
4	Exc: D_PYR → D_PYR	500	5, 5, 1500	0.10	300	0.08 (25%)	6	0	200
5	Exc: S_PYR → BASK1	100	—	0.25	10,000	0.08 (25%)	1	1	10,000
6	Exc: S_PYR → BASK2	100	—	0.25	10,000	0.08 (25%)	1	1	10,000
7	Exc: S_PYR → BASK3	100	—	0.25	10,000	0.08 (25%)	1	1	10,000
8	Exc: D_PYR → BASK1	100	—	0.25	10,000	0.08 (25%)	1	1	10,000
9	Exc: D_PYR → BASK2	100	—	0.25	10,000	0.08 (25%)	1	1	10,000
10	Exc: D_PYR → BASK3	100	—	0.25	10,000	0.08 (25%)	1	1	10,000
11	Exc: S_PYR → S_PYR	<1cm, every 1 mm, 200 μm	5, 5, 1500	0.10	—	15 (95%)	3	3	1,000,000
12	Exc: S_PYR → D_PYR	<1cm, every 1 mm, 200 μm	5, 5, 1500	0.10	—	15 (95%)	3	3	1,000,000
13	Exc: S_PYR → CHAND	100	—	0.25	10,000	0.08 (25%)	.5	.5	10,000
14	Exc: D_PYR → CHAND	100	—	0.25	10,000	0.08 (25%)	.5	.5	10,000
a	Inh: BASK1 → S_PYR	300	—	0.04	300	0.08 (25%)	6	6	10,000
b	Inh: BASK2 → S_PYR	600	—	0.01	600	0.08 (25%)	6	6	10,000
c	Inh: BASK3 → S_PYR	900	—	0.005	900	0.08 (25%)	6	6	10,000
d	Inh: BASK1 → D_PYR	300	—	0.04	300	0.08 (25%)	6	6	10,000
e	Inh: BASK2 → D_PYR	600	—	0.01	600	0.08 (25%)	6	6	10,000
f	Inh: BASK3 → D_PYR	900	—	0.005	900	0.08 (25%)	6	6	10,000
g	Inh: CHAND → S_PYR	300	—	0.04	300	0.08 (25%)	1	0	10,000
h	Inh: CHAND → D_PYR	300	—	0.04	300	0.08 (25%)	1	0	10,000
I	Dinh: BASK1 → BASK1	300	5, 5, 1500	0.04	300	0.08 (25%)	1	1	10,000
II	Dinh: BASK1 → BASK2	300	5, 5, 1500	0.04	300	0.08 (25%)	1	1	10,000
III	Dinh: BASK1 → BASK3	300	5, 5, 1500	0.04	300	0.08 (25%)	1	1	10,000
IV	Dinh: BASK2 → BASK1	600	5, 5, 1500	0.015	600	0.08 (25%)	1	1	10,000
V	Dinh: BASK2 → BASK2	600	5, 5, 1500	0.015	600	0.08 (25%)	1	1	10,000
VI	Dinh: BASK2 → BASK3	600	5, 5, 1500	0.015	600	0.08 (25%)	1	1	10,000
VII	Dinh: BASK3 → BASK1	900	5, 5, 1500	0.007	900	0.08 (25%)	1	1	10,000
VIII	Dinh: BASK3 → BASK2	900	5, 5, 1500	0.007	900	0.08 (25%)	1	1	10,000
IX	Dinh: BASK3 → BASK3	900	5, 5, 1500	0.007	900	0.08 (25%)	1	1	10,000

Figuur 3.1: Tabel met verbindingsgegevens voor een virtueel neuronennetwerk. Afkomstig uit [4], oorspronkelijk gebaseerd op gegevens uit de literatuur. De tabelinhoud geldt in principe uitsluitend voor een kleinschalig netwerk (∼700 neuronen).

(nog) niet opgenomen in NeuroSim. De signaalgeleidingssnelheid geeft automatisch de vertraging (delay) tussen de prikkelgeneratie en het synaptische effect bij een volgend neuron, door de afstand tussen deze twee neuronen (de verbindingslengte) te delen door deze geleidingssnelheid. Synaptische geleiding wordt bepaald door een exponentiële functie en een maximum geleiding. De verbindingssterkte, W, is daarmee tevens gebaseerd op de afstand tussen de broncel en de doelcel: $W = minW + (maxW - minW) \cdot e^{(-\rho_3(x,y,z)/E_W)}$, met $\rho_3(x, y, z)$ de 3-dimensionale Euclidiaanse afstand tussen de neuronen. De variabelen $minW$, $maxW$ en E_W in deze formule zijn te vinden in de laatste drie kolommen (respectievelijk) van de tabel. N.B.: over deze laatste formule bestaat wat verwarring; zoals de formule in deze tekst staat, is deze geïmplementeerd in NeuroSim, maar mogelijk is dit foutief in vergelijking met GENESIS. De gevolgen hiervan zijn echter niet significant groot.

In de rest van dit hoofdstuk zullen we aan de hand van de gegevens uit de tabel in deze paragraaf stapsgewijs een aantal theoretische voorspellingen voor wat betreft de verbindingen in het neuronennetwerk doen. Achtereenvolgens beschouwen we uitdrukkingen voor het verwachte aantal verbindingen van een bepaald type per neuron, het verwachte totaalgewicht van een bepaald verbindingstype per neuron en de verwachte actiepotentiaalreistijd per verbinding van een bepaald type.

3.2 Theoretische bepalingen met betrekking tot het virtuële neuronennetwerk

We weten nu dat verbindingseigenschappen verschillen per neurontype *waarvanaf* en ook per type *waarnaartoe* de (eventuele) verbinding loopt. Voor enige theoretische benadering zullen we dus *per verbindingstype* (per rij uit de tabel in figuur 3.1) de invloeden afzonderlijk moeten beschouwen en type-afhankelijk sommeren. We zullen bij de algemene uitdrukkingen in deze deelparagraaf dan ook doorgaans spreken van een $A \rightarrow B$-verbinding, met A en B zijnde neurontype-aanduidingen. Figuur 3.1 opent de volledige wereld aan theoretische bepalingen voor wat betreft het neuronennetwerk, waarmee het type output van de simulaties deels verklaard/voorspeld zou kunnen worden. Men kan zich bijvoorbeeld richten op de verwachte gemiddelde reistijd van actiepotentialen, het totaal aantal verbindingen uiteraard, maar nog veelzeggender (althans, dat zal later worden bevestigd in de resultaten) is de verhouding tussen de totale excitatie en de totale inhibitie binnen het netwerk. Het is nu bekend dat een netwerkverbinding óf excitatoir, óf inhibitoir is, afhankelijk van het zendende neuron. Iedere verbinding heeft bovendien een zekere sterkte, een 'gewicht'. Het is dus mogelijk om de totale excitatie (het totale gewicht van alle excitatoire verbindingen) en de totale inhibitie (soortgelijke som van alle inhibitoire verbindingsgewichten) van het netwerk te bepalen. Interessant is vervolgens de ratio van deze twee gegevens; dit is een maat voor de totaal-excitatie/-inhibitie in het model. Deze verhouding blijkt (zowel theoretisch als praktisch, zie hoofdstuk 4) te veranderen, naarmate het aantal cellen en/of de grootte van het systeem wordt aangepast. Logisch, want de connectieparameters zoals zojuist geïntroduceerd zijn afhankelijk van bijvoorbeeld de afstanden tussen de neuronen. In deze paragraaf proberen we geleidelijk limieten voor dit type netwerkeigenschappen theoretisch uit te schrijven. Merk op dat het in deze theoretische paragraaf een vraag blijft *vanaf wanneer* we de limietberekeningen mogen toepassen, met andere woorden: bij hoeveel neuronen mogen we het hersensysteem als 'oneindig groot' benaderen en deze limietberekeningen als geldige benadering gebruiken? Gedraagt het breinmodel zich bij zo'n 2.000.000 neuronen als een oneindig systeem of is dat bij ∼18.500 neuronen of minder al het geval? Deze vraag zullen we in hoofdstuk 4 beantwoorden aan de hand van de praktijk.

3.2.1 Aantal netwerkverbindingen

Definieer een netwerk met in totaal N neuronen, waaronder $N_A \in [0, N]$ van type A en $N_B \in [0, N]$ van type B. De neuronen zijn in horizontale richting uniform verdeeld in de ruimte. In verticale richting is de distributie afhankelijk van het geïntroduceerde lagensysteem (zie hoofdstuk 2). Definieer twee neuronen $n_A \in [0, N_A]$ en $n_B \in [0, N_B]$ van type A en B, respectievelijk. $(x_{n_A}, y_{n_A}, z_{n_A})$ is

de Cartesiaanse positie van neuron n_A, $(x_{n_B}, y_{n_B}, z_{n_B})$ die van neuron n_B. $\rho_2(x_{n_A}, y_{n_A}, x_{n_B}, y_{n_B})$ is de horizontale afstand (dus gezien vanuit het bovenaanzicht, een afstand bepaald over 2 dimensies) tussen neuron n_A en neuron n_B:

$$\rho_2(x_{n_A}, y_{n_A}, x_{n_B}, y_{n_B}) = \sqrt{(x_{n_A} - x_{n_B})^2 + (y_{n_A} - y_{n_B})^2}. \tag{3.1}$$

Voor de kans op een verbinding tussen neuron n_A en neuron n_B was in de toelichting van de tabel in de vorige paragraaf aangehaald:

$$P(n_A, n_B) = P_{max}(A, B) \cdot e^{-\frac{\rho_2(x_{n_A}, y_{n_A}, x_{n_B}, y_{n_B})}{E_P(A,B)}}, \tag{3.2}$$

althans wanneer $\rho_2(x_{n_A}, y_{n_A}, x_{n_B}, y_{n_B}) \leq D(A, B)$, met $D(A, B)$ de waarde in de kolom *Destination* van de tabel bij de betreffende $A \to B$-verbinding. Wanneer $\rho_2(x_{n_A}, y_{n_A}, x_{n_B}, y_{n_B}) > D(A, B)$, dan $P(n_A, n_B)$=0.

$P_{max}(A, B)$ en $E_P(A, B)$ komen uit de desbetreffende rij en kolommen (*Propability* en *ExpontentialP*) van de verbindingstabel.

Om $< P > (n_A, B)$, de verwachte verbindingskans tussen specifiek neuron n_A en neuronen van type B, liggende binnen de $D(A, B)$-cirkel om neuron n_A, te bepalen, worden álle mogelijkheden voor $P(n_A, n_B)$, variërende de positie van neuron n_B ten opzichte van neuron n_A, gemiddeld:

$$< P > (n_A, B) = \frac{1}{\pi D(A, B)^2} \int_0^{2\pi} \int_0^{D(A,B)} P_{max}(A, B) \cdot e^{-\frac{r}{E_P(A,B)}} \cdot r dr d\theta, \tag{3.3}$$

overstappende op poolcoördinaten, zodat integratie over de cirkel eenvoudiger is. Zie figuur 3.2. De tweede r in bovenstaande expressie (de r buiten de e-macht) komt vanuit het integreren in poolcoördinaten; dit is de zgn. Jacobiaan. In 3.3 bevindt zich de aanname dat de horizontale verdeling van alle typen neuronen uniform is in de gehele (x, y)-ruimte.

Vergelijking 3.3 is gelijk aan:

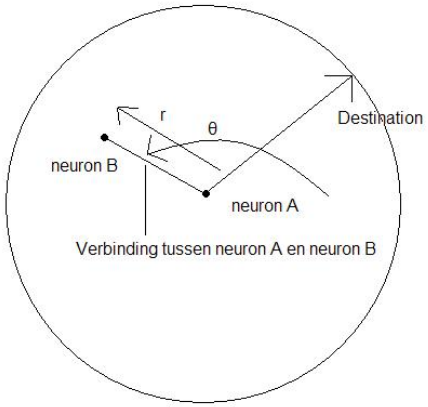
$$< P > (n_A, B) = \frac{2}{D(A, B)^2} \int_0^{D(A,B)} P_{max}(A, B) \cdot e^{-\frac{r}{E_P(A,B)}} \cdot r dr. \tag{3.4}$$

Merk op dat in bovenstaande bepalingen is aangenomen dat het neuron n_A niet in de buurt van de rand van het gedefinieerde complete model ligt, zodat de $D(A, B)$-cirkel inderdaad getrokken mag worden. Wanneer n_A op een afstand van de modelrand ligt die kleiner is dan $D(A, B)$, geldt dat er als het ware sprake is van 'half een cirkel, half een vierkant'. Een deel van de $D(A, B)$-cirkel om neuron n_A valt buiten het model. In dat geval wordt de middeling in 3.3 veel moeilijker. Later in dit hoofdstuk zullen we berekeningen doen voor limietgevallen, dus gevallen van een oneindig groot model/systeem, en is er dus geen sprake van deze situatie, maar in eindige systemen zullen er altijd neuronen aan de rand van het 'blokje' hersenen liggen. Zelfs bij een netwerk van, zeg, 2.000.000 neuronen zou de rand van de modelruimte nog steeds een belangrijke rol kunnen spelen! Of dat zo is, kan eigenlijk alleen maar praktisch worden beschouwd, en in hoofdstuk 4 zal hier dus op worden teruggekomen.

Wanneer we $< P > (n_A, B)$ vermenigvuldigen met het verwachte aantal neuronen van type B binnen de $D(A, B)$-cirkel om n_A heen, dan vinden we rechtstreeks het verwachte aantal verbindingen van type $A \to B$ vanaf (specifiek) neuron n_A. Let op: we vermenigvuldigen hier twee verwachtingswaarden, wat we ook mogen doen omdat deze twee verwachtingswaarden onafhankelijk van elkaar zijn. Het verwachte aantal B-neuronen in de D-cirkel, noem dit $< M_B > (A, B, \gamma_B)$, is de 2D-neuronendichtheid van het type-B-neuron γ_B (aantal B-neuronen per m^2 bovenaanzicht) maal de oppervlakte van de $D(A, B)$-cirkel:

$$< M_B > (A, B, \gamma_B) = \gamma_B \cdot \pi D(A, B)^2. \tag{3.5}$$

Het woord 'dichtheid' is hier wat misplaatst, omdat γ_B het aantal (B-)cellen per oppervlakte in plaats van per inhoud betreft. Substitutie in 3.4 geeft nu voor het verwachte aantal verbindingen van type $A \to B$ vanaf specifiek neuron n_A, noem deze $< \Psi > (n_A, B, \gamma_B)$,:



Figuur 3.2: Bepalen van $< P > (n_A, B)$ door alle mogelijkheden voor de positie van neuron B binnen de $D(A,B)$-cirkel te variëren.

$$< \Psi > (n_A, B, \gamma_B) = 2\pi \cdot \gamma_B \cdot \int_0^{D(A,B)} P_{max}(A,B) \cdot e^{-\frac{r}{E_P(A,B)}} \cdot r dr. \tag{3.6}$$

Samenvattend: men kiest een neurontype A, daarna één van de neuronen van dat type binnen het netwerk, n_A, en berekent vervolgens het verwachte aantal verbindingen van type $A \to B$ vanaf dit specifieke gekozen neuron.

3.2.2 Netwerkexcitatie en -inhibitie

Zoals vermeld in de introductieparagraaf van dit hoofdstuk varieert het gewicht per verbinding, afhankelijk van de Euclidiaanse afstand tussen de 2 neuronen (gezien in 3D), oftewel gebaseerd op de lengte van de verbinding, en bepaalde parameters uit de tabel. Het gewicht van een verbinding van type $A \to B$ (tussen neuronen n_A en n_B) is (in de huidige implementatie) gedefinieerd als:

$$W(n_A, n_B) = minW(A,B) + (maxW(A,B) - minW(A,B)) \cdot e^{-\frac{\rho_3(x_{n_A}, y_{n_A}, z_{n_A}, x_{n_B}, y_{n_B}, z_{n_B})}{E_W(A,B)}}, \tag{3.7}$$

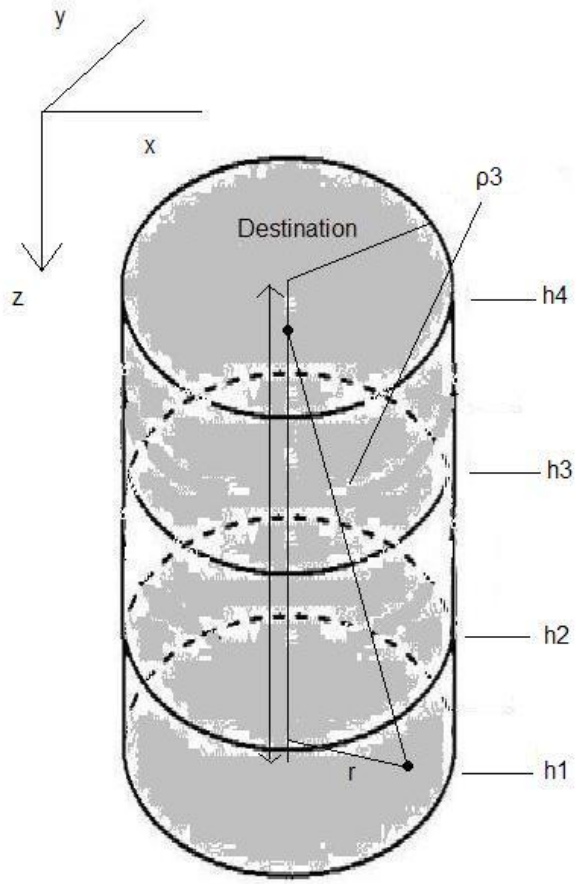
waarbij $minW(A,B)$, $maxW(A,B)$ en $E_W(A,B)$ terug te vinden zijn in de laatste drie kolommen van de verbindingstabel, bij het betreffende verbindingstype. $\rho_3(x_{n_A}, y_{n_A}, z_{n_A}, x_{n_B}, y_{n_B}, z_{n_B})$ is de rechtlijnige verbindingslengte, de (op 3 dimensies gebaseerde) afstand tussen de twee neuronen voor een (eventuele) verbinding:

$$\rho_3(x_{n_A}, y_{n_A}, z_{n_A}, x_{n_B}, y_{n_B}, z_{n_B}) = \sqrt{(x_{n_A} - x_{n_B})^2 + (y_{n_A} - y_{n_B})^2 + (z_{n_A} - z_{n_B})^2}, \tag{3.8}$$

wat in poolcoördinaten herschreven wordt in

$$\rho_3(x_{n_A}, y_{n_A}, z_{n_A}, x_{n_B}, y_{n_B}, z_{n_B}) = \sqrt{r^2 + (z_{n_A} - z_{n_B})^2}. \tag{3.9}$$

De z-richting speelt nu een rol, dus zullen we bij middeling moeten integreren in 3 dimensies, namelijk over een cilinder met neuron n_A op de middenas van de cilinder en neuron n_B op een willekeurige (x,y)-locatie en een willekeurige, maar type-afhankelijke (laag-afhankelijke) z-locatie, binnen de cilinder. Zie figuur 3.3. r is nog steeds de bovenaanzichtsafstand (dus 2D, gebaseerd op (x,y)-positie) tussen neuron n_A en neuron n_B. Het probleem wordt nu een stuk geavanceerder dan in de vorige deelparagraaf, aangezien de neurontypen onderverdeeld zijn in het lagensysteem over de z-richting. De SPyr-cellen bevinden zich in laag II en III (van bovenaf gezien), de DPyr-cellen



Figuur 3.3: Bepalen van $W^{tot}(n_A, B)$ door alle mogelijkheden voor de positie van neuronen A en B binnen de $D\&h$-cilinder te variëren.

in laag V en VI, en de interneuronen zijn verdeeld over laag II, III, IV, V en VI (laag I, de bovenste laag van 250 μm, is leeg). Voor de SPyr→DPyr-verbindingen, bijvoorbeeld, zullen we dus op een bijzondere manier moeten middelen; er bevindt zich immers altijd een (verticale) tussenruimte tussen een SPyr- en een DPyr-cel. De invulling valt echter mee en is volledig naar onze hand te zetten door over de juiste regionen met de juiste grenzen te middelen (integreren). Definieer de cilinder als in figuur 3.3. De straal van de cilinder is $D(A,B)$. Neuron n_A bevindt zich op de middenas van de cilinder tussen hoogte h_3 en hoogte h_4. Neuron n_B bevindt zich tussen hoogte h_1 en hoogte h_2 op een willekeurige straal van de cilinder. Voor het bepalen van het verwachte totale verbindingsgewicht van type $A \to B$ per neuron n_A van type A, $< W^{tot} > (n_A, B, \gamma_B)$, moet dus gemiddeld worden over alle mogelijke hoogteposities van neuron n_A, alle mogelijke hoogteposities van neuron n_B, alle mogelijke stralen r van neuron n_B ten opzichte van neuron n_A en alle mogelijke hoeken θ van neuron n_B ten opzichte van neuron n_A:

$$< W^{tot} > (n_A, B, \gamma_B) = X \cdot \int_{h_1}^{h_2} \int_{h_3}^{h_4} \int_0^{2\pi} \int_0^{D(A,B)} Y(r) \cdot Z(r) \cdot r dr d\theta dz_1 dz_2, \tag{3.10}$$

met

$$X = \frac{\gamma_B \cdot \pi \cdot D(A,B)^2}{\pi \cdot D(A,B)^2 \cdot (h_4 - h_3) \cdot (h_2 - h_1)} = \frac{\gamma_B}{(h_4 - h_3) \cdot (h_2 - h_1)}; \tag{3.11}$$

$$Y(r) = P_{max} \cdot e^{-\frac{r}{E_P(A,B)}}; \tag{3.12}$$

$$Z(r) = minW(A,B) + (maxW(A,B) - minW(A,B)) \cdot e^{-\frac{\sqrt{(z_{n_A} - z_{n_B})^2 + r^2}}{E_W(A,B)}}. \tag{3.13}$$

h_1, h_2, h_3 en h_4 variëren per verbindingstype. Ter verduidelijking een aantal voorbeelden. Voor de SPyr→SPyr-verbindingen is h_1 gelijk aan h_3, namelijk 750 μm, en h_2 gelijk aan h_4, 1250 μm, want de SPyr-cellen bevinden zich in laag II en III (samen 1250-750=500 μm) van de cilinder. Voor de DPyr→DPyr-verbindingen geldt h_1=h_3=0 μm en h_2=h_4= 500 μm, maar dit zal exact dezelfde uitkomst geven als voor de SPyr→SPyr-verbindingen, omdat dit feitelijk het identieke probleem is, maar gedraaid in verticale richting. Voor de SPyr→DPyr-verbindingen, en ook andersom, vullen we in: h_1=0 μm, h_2= 500 μm, h_3 = 750 μm en h_4 = 1250 μm. Voor een SPyr→Interneuron-verbinding geldt: h_1=0 μm, h_2=h_4=1250 μm en h_3=750 μm.

Voor de compleetheid kunnen we tevens een uitdrukking introduceren voor het verwachte gewicht per verbinding van type $A \to B$, $< W > (n_A, B, \gamma_B)$, in plaats van het verwachte totaalgewicht van type $A \to B$ per neuron van type A, zoals gedefinieerd in 3.10:

$$< W > (n_A, B, \gamma_B) = \frac{X \cdot \int_{h_1}^{h_2} \int_{h_3}^{h_4} \int_0^{2\pi} \int_0^{D(A,B)} Y(r) \cdot Z(r) \cdot r dr d\theta dz_1 dz_2}{2\pi \cdot \gamma_B \cdot \int_0^{D(A,B)} Y(r) \cdot r dr}. \tag{3.14}$$

We onderscheiden in de tabel, voor wat betreft de berekening van het verwachte verbindingsgewicht voor zekere typen zend- (A) en ontvangst- (B) neuronen, twee gevallen: het geval $minW = maxW$ en het geval $minW \neq maxW$. In dat laatste geval geldt automatisch $minW = 0$ (zie tabel). Wanneer $minW$ gelijk is aan $maxW$, dan hebben we geluk, want dan geldt $Z = minW = maxW$, en daarmee

$$< W^{tot} > (n_A, B, \gamma_B) = X \cdot minW(A,B) \cdot \int_{h_1}^{h_2} \int_{h_3}^{h_4} \int_0^{2\pi} \int_0^{D(A,B)} Y(r) \cdot r dr d\theta dz_1 dz_2. \tag{3.15}$$

Dit is simpelweg gelijk aan:

$$< W^{tot} > (n_A, B, \gamma_B) = \gamma_B \cdot 2\pi \cdot minW(A,B) \cdot \int_0^{D(A,B)} P_{max}(A,B) \cdot e^{-\frac{r}{E_P(A,B)}} \cdot r dr, \tag{3.16}$$

een relatief eenvoudige integraal die prima uit te rekenen blijkt te zijn door middel van partiële integratie. Wanneer $minW \neq maxW$, dan wordt de situatie lastiger. De e-macht in Z blijft dan staan; Z wordt gelijk aan $maxW(A,B) \cdot e^{-\frac{\sqrt{(z_{n_A} - z_{n_B})^2 + r^2}}{E_{W^{(A,B)}}}}$. De integrant is nu van de vorm $r \cdot e^{-r-\sqrt{\alpha^2+r^2}}$. $< W^{tot} > (n_A, B, \gamma_B)$ is nu niet 'rechtstreeks' uit te rekenen, ook niet numeriek! We zullen verderop in deze paragraaf een Riemann-benadering uitvoeren.

3.2.3 Excitatie-inhibitieratio in een oneindig groot NeuroSim-netwerk

Neem nu een oneindig groot aantal neuronen aan - of, logischer, misschien: een complete Neocortex, wat in ons geval al als een oneindig aantal neuronen mag functioneren. Deze neuronen zijn nog altijd per type uniform (met een zekere neuronendichtheid die verschilt per z-positie) verdeeld over de 1.5 mm diepte en oneindige oppervlakte in (x, y)-ruimte.

In de vorige deelparagrafen hebben we uitdrukkingen gevonden voor het verwachte totaalgewicht vanaf een zendend neuron van type A naar ontvangende neuronen van type B binnen de $D(A,B)$-cirkel rond neuron n_A, gebaseerd op de tabel, zie uitdrukking 3.10. Natuurlijk, wanneer we dit verwachte totaalgewicht per type per zendend neuron vermenigvuldigen met het aantal A-neuronen in het netwerk, N_A, dat oneindig is, is het verwachte totaalgewicht over alle $A \rightarrow B$-verbindingen in het netwerk, wat op deze manier afgeleid wordt, ook oneindig. Het is niet onlogisch dat in een netwerk met een oneindig aantal neuronen het aantal verbindingen (en dus ook het totaalgewicht over deze verbindingen) oneindig is. Let op: niet alleen de excitatoire verbindingen leveren samen een oneindig totaalgewicht, maar natuurlijk ook de inhibitoire verbindingen. Hun ratio zal daarmee (in limietgeval) níét oneindig zijn.

We definiëren de 'excitatie-inhibitieratio', $\frac{E}{I}$, als zijnde de ratio van het totale gewicht over álle excitatoire verbindingen in het netwerk, over het totale gewicht over álle inhibitoire verbindingen in het netwerk. De verwachte excitatie-inhibitieratio, $< \frac{E}{I} >$, is dus een maat voor de netwerkexcitatie in verhouding tot de netwerkinhibitie:

$$< \frac{E}{I} >= \frac{\sum_{A=1}^{2} \sum_{B=1}^{6} N_A \cdot < W^{tot} > (n_A, B, \gamma_B)}{\sum_{A=3}^{6} \sum_{B=1}^{6} N_A \cdot < W^{tot} > (n_A, B, \gamma_B)}, \tag{3.17}$$

waarbij we aanhouden:

$$type\ 1 = \ SPyr - cellen;\ Oppervlakkige\ piramidecellen\ (excitatoir); \tag{3.18}$$
$$type\ 2 = \ DPyr - cellen;\ Diepe\ piramidecellen\ (excitatoir); \tag{3.19}$$
$$type\ 3 = \ Bask1 - cellen;\ Basketcellen,\ type\ 1\ (inhibitoir); \tag{3.20}$$
$$type\ 4 = \ Bask2 - cellen;\ Basketcellen,\ type\ 2\ (inhibitoir); \tag{3.21}$$
$$type\ 5 = \ Bask3 - cellen;\ Basketcellen,\ type\ 3\ (inhibitoir); \tag{3.22}$$
$$type\ 6 = \ Chandl - cellen;\ Kandelaarscellen\ (inhibitoir). \tag{3.23}$$

De somratio zoals deze staat gedefinieerd in definitie 3.17 is duidelijk en rechtstreeks afkomstig uit de vorige deelparagrafen, maar nog niet uitrekenbaar, omdat in het limietgeval N_A (voor alle A, A=1..6) immers oneindig is. Op welke manier deelt deze oneindigheid nu weg? Substitueer:

$$N_A = \gamma_A \cdot \Delta, \tag{3.24}$$

oftewel: het totaal aantal neuronen van type A in het model is gelijk aan het (uniforme) aantal A-neuronen per oppervlakte, γ_A, maal de totale horizontale oppervlakte van het model, Δ (men zou kunnen zeggen: Δ is de oppervlakte van de uitgevouwen hersenen). In een oneindig groot model is Δ oneindig, maar γ_A natuurlijk niet. Substitueer dit in 3.17 en vind:

$$< \frac{E}{I} >= \frac{\sum_{A=1}^{2} \sum_{B=1}^{6} \gamma_A \cdot \Delta \cdot < W^{tot} > (n_A, B, \gamma_B)}{\sum_{A=3}^{6} \sum_{B=1}^{6} \gamma_A \cdot \Delta \cdot < W^{tot} > (n_A, B, \gamma_B)} = \frac{\sum_{A=1}^{2} \sum_{B=1}^{6} \gamma_A \cdot < W^{tot} > (n_A, B, \gamma_B)}{\sum_{A=3}^{6} \sum_{B=1}^{6} \gamma_A \cdot < W^{tot} > (n_A, B, \gamma_B)}, \tag{3.25}$$

met $< W^{tot} > (n_A, B, \gamma_B)$ als in 3.10.

Nu is γ_A voor A=1 en voor A=2 hetzelfde, en ook voor A=3 t/m 6. Verhouding van de celdichtheid voor excitatoire cellen tot de celdichtheid voor inhibitoire cellen is in NeuroSim 256:36 (of 64:9). Invullen in 3.25 levert:

$$< \frac{E}{I} >= \frac{\sum_{A=1}^{2}\sum_{B=1}^{6} < W^{tot} > (n_A, B, \gamma_B)}{\sum_{A=3}^{6}\sum_{B=1}^{6} \cdot < W^{tot} > (n_A, B, \gamma_B)} \cdot \frac{256}{36}. \tag{3.26}$$

De afhankelijkheid van de parameters uit de tabel is nu duidelijk in beeld verschenen. Er is ook een keerzijde: wat voegt deze limiet nu eigenlijk toe? Wat hebben we daadwerkelijk berekend? Kunnen we het model aanpassen (lees: corrigeren) aan de hand van de beschrijvingen die we zojuist gedaan hebben en de berekeningen die we nu gaan doen? Dit kunnen we alleen door middel van de praktijk bepalen. Misschien is een bepaling van de limiet over *het aantal* excitatoire en inhibitoire verbindingen, noem dit $< \frac{E}{I} >'$, wel veel significanter dan de gewichtsratio. In dat geval is het gewicht van de verbinding niet langer belangrijk en vinden we:

$$< \frac{E}{I} >'= \frac{\sum_{A=1}^{2}\sum_{B=1}^{6} < \Psi > (n_A, B, \gamma_B)}{\sum_{A=3}^{6}\sum_{B=1}^{6} \cdot < \Psi > (n_A, B, \gamma_B)} \cdot \frac{256}{36}, \tag{3.27}$$

met $< \Psi > (n_A, B, \gamma_B)$ als gedefinieerd in 3.6.

De laatste stap voor wat betreft de bepaling van de excitatie-inhibitieratio is natuurlijk het uitrekenen van $< W^{tot} > (n_A, B, \gamma_B)$ voor alle mogelijke combinaties van A en B (dus A=1,2 & B=1..6 en A=3..6 & B=1..6) en correct in te vullen in 3.26. In dat geval nemen we de 'standaard' *globale* excitatie- en inhibitiewaarde 1.0 en 1.0, respectievelijk, aan - voor andere waarden worden de $maxW$'s en de $minW$'s aangepast. Natuurlijk gebruiken we de parameterwaarden uit figuur 3.1.

We vonden een probleem voor 3.10, in het geval dat $minW \neq maxW$; er bleef dan een lastige drievoudige integraal staan (de integraal over θ kan vervangen worden met een vermenigvuldiging met 2π), die we moeten benaderen in plaats van berekenen. We kiezen hierbij voor een Riemann-benadering, waarin we de drie variabelen waarover geïntegreerd wordt (in ons geval r, z_1 en z_2) in $(\sim 300) \cdot (\sim 300) \cdot (\sim 300)$ blokjes opdelen, als in:

$$\int_a^b \int_c^d \int_e^f f(x,y,z)dzdydx \approx \Delta x \Delta y \Delta z \sum_{i=0}^{\sim 300}\sum_{j=0}^{\sim 300}\sum_{k=0}^{\sim 300} f(x_i, y_j, z_k), \tag{3.28}$$

met

$$\Delta x = x_{i+1} - x_i; \tag{3.29}$$

$$\Delta y = y_{j+1} - y_j; \tag{3.30}$$

$$\Delta z = z_{k+1} - z_k. \tag{3.31}$$

x_0 is hierbij gelijk aan a, $x_{\sim 300}$ aan b, y_0 aan c, $y_{\sim 300}$ aan d, z_0 aan e en $z_{\sim 300}$ aan f. We nemen voor het gemak een niet variërende stap-/blokgrootte aan. Het aantal Riemann-blokjes mag natuurlijk naar keuze gevarieerd worden, maar te weinig blokjes geeft een (te) onnauwkeurige benadering en te veel blokjes brengt memoryproblemen met zich mee.

Het geval $minW \neq maxW$ komt in de tabel (slechts) 6 maal voor, namelijk bij de SPyr→SPyr- & DPyr→DPyr-verbindingen, bij de SPyr→DPyr- & DPyr→SPyr-verbindingen en bij de Chandl→SPyr- en Chandl→DPyr-verbindingen. Bovenstaande Riemann-benadering wordt slechts bij deze 6 bepalingen van $< W^{tot} > (n_A, B, \gamma_B)$ toegepast. Bij de andere verbindingstypen is de integratie handmatig (exact) uit te voeren.

We sommeren over álle verbindingen, beter gezegd, over het *verwachte* aantal excitatoire en inhibitoire verbindingen, in het modelnetwerk. Wanneer we in de tabel kijken, zien we dat bijvoorbeeld de SPyr→SPyr-verbindingen exact dezelfde eigenschappen hebben als de DPyr→DPyr-verbindingen. Er hoeft dus niet voor elk uniek verbindingstype (rij in de tabel) een opzichzelfstaande berekening uitgevoerd te worden; er kunnen typen gecombineerd worden.

Wanneer we alle gegevens, nu allemaal voor ons bekend, invullen in 3.26, levert dit:

$$< \frac{E}{I} >= \underline{8.013...}. \tag{3.32}$$

Over de correctheid hiervan doen we nog geen uitspraken; deze limiet zullen we in hoofdstuk 4 proberen te bevestigen aan de hand van de praktijk.

N.B.: nogmaals, deze berekende waarde is uitsluitend geldig in het limietgeval van een oneindig groot netwerk. Bij een 'zeer groot' netwerk zal deze waarde benaderd worden. Wat 'zeer groot' inhoudt, moeten we op dit moment nog in het midden laten.

3.2.4 Actiepotentiaalreistijd

Wanneer we de verwachting van de gemiddelde reistijd in ons model (de gemiddelde tijd die een actiepotentiaal nodig heeft om van het ene neuron naar het andere neuron te reizen) willen uitrekenen, zullen we de gemiddelde afstand (3 dimensies) tussen twee willekeurige verbonden neuronen moeten benaderen & middelen, en deze moeten delen door de voortbewegingssnelheid van een actiepotentiaal, oftewel door de waarde in de kolom $Velocity$ bij de betreffende verbinding, die voor iedere geïmplementeerde connectie 0.08 m/s is.

De verwachte *totale* verbindingslengte van type $A \rightarrow B$ per zendend neuron n_A, $< \rho_3^{tot} >$, wordt gevonden via dezelfde weg als voorheen in dit hoofdstuk:

$$< \rho_3^{tot} >= X' \int_{h_1}^{h_2} \int_{h_3}^{h_4} \int_0^{2\pi} \int_0^{D(A,B)} Y'(r) \cdot Z'(r) \cdot r dr d\theta dz_1 dz_2, \tag{3.33}$$

met

$$X' = X = \frac{\gamma_B}{(h_4 - h_3) \cdot (h_2 - h_1)}; \tag{3.34}$$

$$Y'(r) = Y(r) = P_{max} \cdot e^{-\frac{r}{E_P(A,B)}}; \tag{3.35}$$

$$Z'(r) = \sqrt{(z_{n_A} - z_{n_B})^2 + r^2}. \tag{3.36}$$

Hierbij wordt de kans op een verbinding maal de lengte van deze verbinding gemiddeld over alle mogelijkheden voor de posities van de twee verbonden neuronen, en vermenigvuldigd met het aantal neuronen binnen de $D(A,B)$-cirkel om neuron n_A heen. De integraal(verzameling) in definitie 3.33 blijkt prima exact uit te rekenen zijn, bijvoorbeeld met Maple.

De verwachte *gemiddelde* verbindingslengte voor de $A \rightarrow B$-verbindingen vanaf een specifiek neuron van type A, $< \rho_3 >$, is gelijk aan de verwachte *totale* verbindingslengte voor alle $A \rightarrow B$-verbindingen vanaf dit neuron gedeeld door het verwachte aantal verbindingen van dit type vanaf dit neuron n_A:

$$< \rho_3 >= \frac{2\pi \cdot X' \int_{h_1}^{h_2} \int_{h_3}^{h_4} \int_0^{D(A,B)} Y'(r) \cdot Z'(r) \cdot r dr dz_1 dz_2}{2\pi \cdot \gamma_B \cdot \int_0^{D(A,B)} Y'(r) \cdot r dr}. \tag{3.37}$$

Om tot de verwachte gemiddelde reistijd, noem deze $< \tau >$, te komen, moet men $< \rho_3^{tot} >$ voor alle verbindingstypen afzonderlijk uitrekenen en naar verhouding (gebaseerd op het verwachte aantal verbindingen per type) sommeren, zodat de gemiddelde verbindingslengte over alle verbindingstypen in het netwerk overblijft. Deze gemiddelde lengte moet gedeeld worden door de gegeven snelheid van 0.08 m/s:

$$< \tau > [s] = \frac{\sum_{A=1}^{6} \sum_{B=1}^{6} < \rho_3^{tot} > \cdot N'_A[m]}{\sum_{A=1}^{6} \sum_{B=1}^{6} < \Psi > (n_A, B, \gamma_B) \cdot N'_A} \frac{1}{0.08[m/s]}, \tag{3.38}$$

met N'_A 256 of 36, afhankelijk van het type zendende neuron (A) in de betreffende verbinding (neuron A excitatoir: N'_A=256, neuron A inhibitoir: N'_A=36). Deze N'_A is van toepassing omdat

$< \rho_3^{tot} >$ en $< \Psi > (n_A, B, \gamma_B)$ *per zendend neuron* gedefinieerd waren. Dit is een zelfde soort doorvoering als de factor $\frac{256}{36}$ in 3.26. Voor de gemiddelde reistijd maakt het natuurlijk niet uit of een verbinding excitatoir of inhibitoir is.

Wanneer we alle gegevens in formule 3.38 uitrekenen en invullen, vinden we:

$$< \tau >= \underline{7.17...} ms. \tag{3.39}$$

Over de correctheid hiervan doen we wederom nog geen uitspraken; deze limiet zullen we in hoofdstuk 4 proberen te bevestigen aan de hand van de praktijk.

N.B.: nog steeds geldt dat deze berekende waarde uitsluitend geldig is in het limietgeval van een oneindig groot netwerk. Bij een 'zeer groot' netwerk zal deze waarde benaderd worden. Wat 'zeer groot' inhoudt, moeten we op dit moment nog in het midden laten.

3.2.5 Gemiddelde afstand tussen twee neuronen (binnen de *Destination*-cirkel)

Vooralsnog is in deze paragraaf uitsluitend naar verbindingen gekeken; dat is ook eigenlijk het enige wat van toepassing is op dit moment, omdat we geïnteresseerd zijn in de connectie-eigenschappen, dus de verplaatsing van activiteit in de ruimte. Wanneer men bepaalde andere fysica van het neuronennetwerk zou willen bestuderen, kunnen andere zaken van toepassing zijn dan de gemiddelde verbindingskans, bijvoorbeeld alleen de gemiddelde horizontale afstand, $< \rho_2 > (n_A, B)$, of Euclidiaanse afstand (3D) tussen twee neuronen (al dan niet verbonden) van type A en B, respectievelijk. Formule 3.3 verandert in het 2D-geval in:

$$< \rho_2 > (n_A, B) = \frac{1}{\pi \cdot D(A,B)^2} \int_0^{2\pi} \int_0^{D(A,B)} |r| \cdot r dr d\theta = \frac{2}{D(A,B)^2} \int_0^{D(A,B)} r^2 dr, \tag{3.40}$$

oftewel: de verbindingskansverdeling zelf is niet langer belangrijk. 3.40 is de definitie voor, zogezegd, de gemiddelde horizontale (2D-)tussenafstand van het verbindingstype $A \rightarrow B$ voor alle ontvangende neuronen van type B binnen de $D(A,B)$-cirkel rond het zendende A-neuron, per specifiek neuron n_A. Ook in 3D kan het van belang zijn gemiddelde afstanden te berekenen. Het bepalen van de gemiddelde afstand tussen twee (willekeurige) punten in een ruimte (1D, 2D, 3D of zelfs nD, $n > 3$) is een bekend probleem, genaamd het 'Line picking' probleem. Omdat deze kwestie voor de netwerkeigenschappen waarin wij geïnteresseerd zijn voorlopig weinig betekenis heeft, laten we het Line picking probleem op dit moment voor wat het is.

Hoofdstuk 4

Resultaten

4.1 Simulatieruntijd

Op neurofysisch vlak is de runtijd van een simulatie met NeuroSim niet bepaald een significant punt van onderzoek. Deze zal in het algemeen natuurlijk toenemen met de systeemgrootte en initieel afnemen met het aantal beschikbare processoren. Op programmatechnisch vlak is het echter van belang de 'speed-up lines' (verband tussen runtijd en aantal CPU's) in beeld te brengen. Op deze manier wordt bijvoorbeeld zichtbaar in hoeverre het (naar verwachting) zinvol zou zijn om véél meer CPU's te reserveren voor een 'enorme' simulatie. In deze paragraaf worden de aspecten betreffende simulatieruntijd bekeken.

4.1.1 Versnelling ten gevolge van gebruik van OpenMP

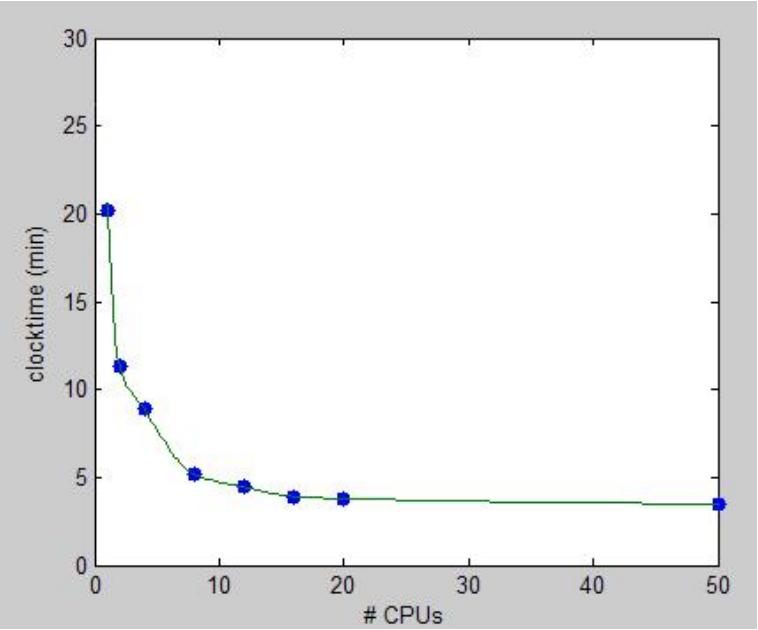
Wanneer 'NeuroSim' een grootschalige (en dus langdurige) simulatie uitvoert, oftewel een groot gedeelte van het brein (veel neuronen) modelleert, dan is het wenselijk dit in een zo kort mogelijk tijdsbestek (doch met 'correcte' resultaten, dus als zijnde berekend met 1 CPU) te laten plaatsvinden. Dit was de reden van de snelle parallellisatie van de 'Step' in NeuroSim, met behulp van OpenMP. In deze paragraaf wordt de versnelling van NeuroSim ten opzichte van het aantal CPU's beschouwd bij gebruik van OpenMP.

In figuren 4.1, 4.2 en 4.3 worden de speed-up lines, de verbanden tussen de rekentijd en het aantal CPU's, voor een steeds identieke situatie (het identieke probleem, dus dezelfde hoeveelheid neuronen en verbindingen, parameters, randvoorwaarden, beginvoorwaarden, etc.), getoond voor een klein systeem (656 neuronen - 1 seconde), medium systeem (8000 neuronen - 0.5 seconde) en groot systeem (18500 neuronen - 0.2 seconde). We zien in alle drie de situaties een (snel!) afvlakkende lijn: het kloktijdverschil tussen 20 en 50 CPU's is relatief niet meer zo groot.

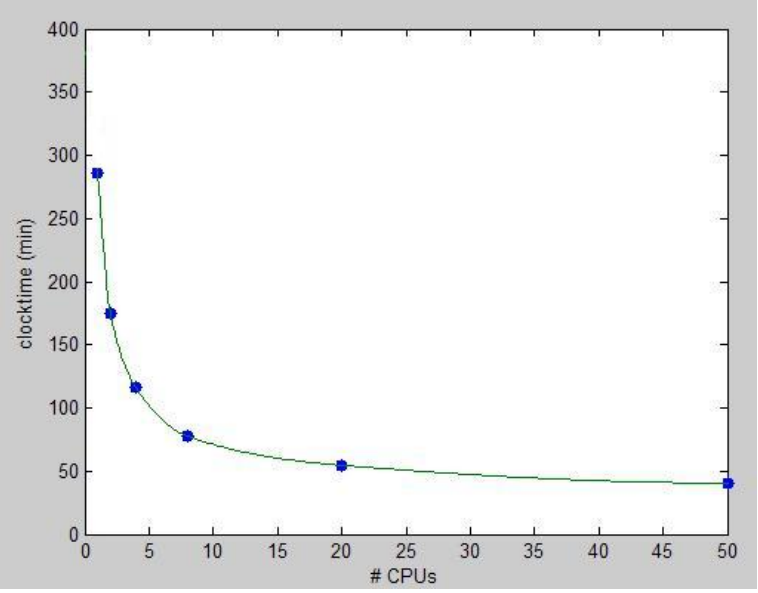
4.1.2 Probleemgrootte-afhankelijkheid

Vergroten van het systeem met behouden parameters en hetzelfde aantal CPU's blijkt voor wat simulatieruntijd betreft wat verraderlijk. Dit is gekoppeld aan een latere paragraaf binnen de resultaten, waarin de uitkomst is dat, in de originele situatie, de excitatie-/inhibitie-verhouding verandert met de systeemgrootte. Gedurende geleidelijke vergroting van het probleem wordt het netwerk eerst meer excitatoir, en daarna (bij systemen van ca. 10.000 neuronen) sterk inhibitoir. Simulatieruntijd is niet alleen afhankelijk van deze systeemgrootte (en het aantal CPU's), maar ook in sterke mate van de parameters binnen het model. In het algemeen vereist een sterk excitatoir model véél meer werk dan een sterk inhibitoir model. Logisch, want bij sterke excitatie worden in het algemeen een boel actiepotentialen gevuurd, wat veel bewerkingen, opslag en communicatie vraagt.

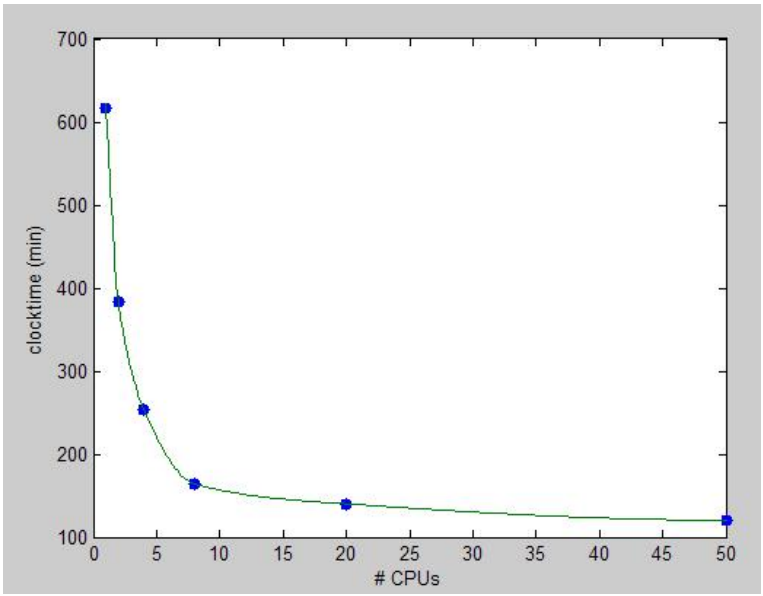
Wanneer we, voor wat betreft de rekentijd, de afhankelijkheid van de probleemgrootte beschouwen, zien we in het begin (voor 'kleine' problemen) een niet-lineair verband. Zie figuur 4.4, die de



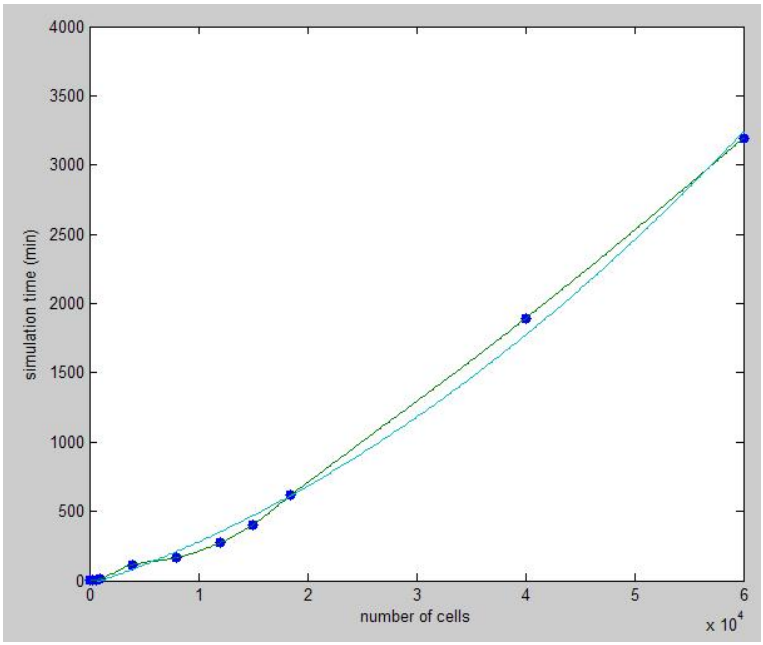
Figuur 4.1: SpeedUp-line voor klein probleem: 656 neuronen (256 + 256 + 36 + 36 + 36 + 36), ruimtelijke structuur 80 μm, simuleertijd 1 seconde, excitatie: 1, inhibitie: 1



Figuur 4.2: SpeedUp-line voor medium probleem: 8000 neuronen (3122 + 3122 + 439 + 439 + 439 + 439), ruimtelijke structuur 270 μm, simuleertijd 0.5 seconde, excitatie: 1, inhibitie: 1



Figuur 4.3: SpeedUp-line voor groot probleem: 18500 neuronen (7220 + 7220 + 1015 + 1015 + 1015 + 1015), ruimtelijke structuur 400 μm, simuleertijd 0.2 seconde, excitatie: 1, inhibitie: 1



Figuur 4.4: Simulatieruntijd tegen het aantal neuronen bij constante celdichtheid voor een hersensimuleertijd van 0.2 seconden bij 1 CPU. E:I=1.0:1.0. De punten zijn de gemeten data; de lijnen vormen de 'best fit' van een *tweedegraads* polynoom en een 'shape-preserving' interpolatie. We zien dat (op de betreffende schaal) de simulatietijd (en daarmee ook de complexiteit van het model) (bij een voldoende groot probleem) lineair lijkt te schalen. De 'helling' van deze lineaire fit is echter groot, ongeveer 3.75 seconde per cel. Wanneer het probleem 2 maal zo groot wordt, wordt de simulatietijd méér dan 2 maal zo groot.

simuleertijd voor een 0.2-seconde-simulatie bij gebruik van 1 processor laat zien voor verschillende probleemgroottes en een constante (handmatige) excitatie-/inhibitieschaal. Voor kleine problemen lijkt het gedrag enigszins kwadratisch. Voor grotere problemen laat interpolatie echter een lineair verband zien.

Het is een belangrijke veronderstelling dat de rekentijd van de simulatie van meer zaken afhangt dan alleen de hoeveelheid neuronen (en het aantal CPU's). Variatie van de excitatie- en/of inhibitieschaal beïnvloedt (de output en) de rekentijd aanzienlijk, wat al bij het 'kleine' hersensysteem van 656 neuronen in het oog was gesprongen. Omdat deze excitatie-/inhibitieschaal 'uit zichzelf' al verandert bij aanpassing van de probleemgrootte (zoals blijkt), is dit eigenlijk slechts de enige geldige conclusie die we mogen trekken uit deze paragraaf! In de volgende paragraaf gaan we dieper in op deze automatische (in bepaald opzicht foutieve!) parameterverandering.

4.2 Netwerkeigenschappen

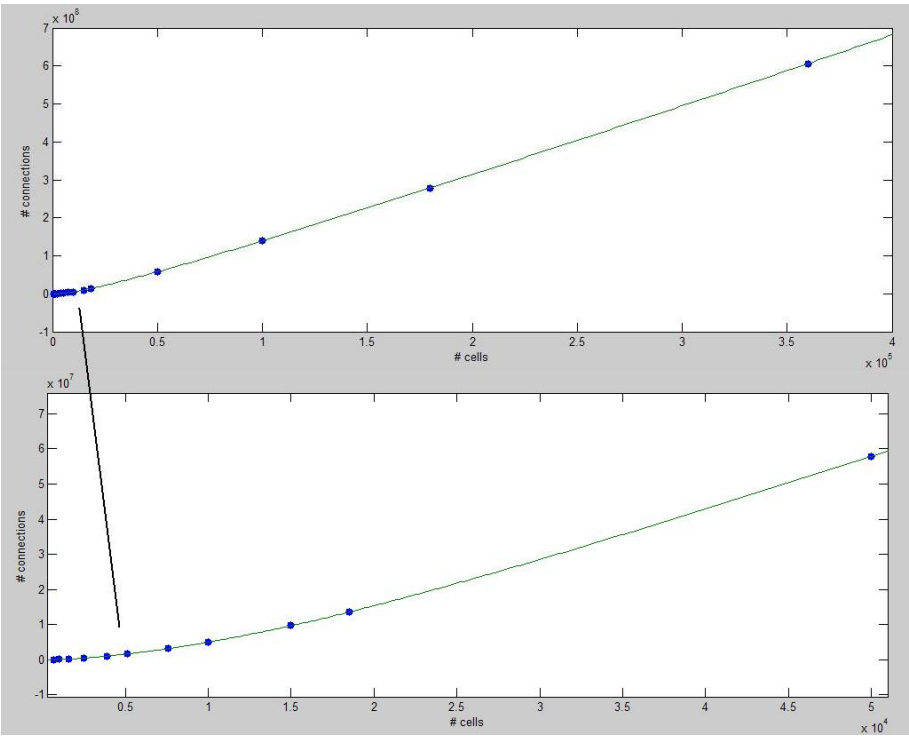
De hypothese was dat het totaal aantal neuronverbindingen in het breinmodel tegen het aantal neuronen bij geleidelijke vergroting van het probleem, met constante celdichtheid, van een (ongeveer) kwadratisch verband over zou gaan naar een (ongeveer) lineair verband. Dit konden we verklaren aan de hand van hoofdstuk 3: er is een zekere maximale afstand waarover twee neuronen kunnen verbinden. Voor een klein systeem wordt deze maximum afstand *niet* of *niet zo vaak* overtroffen en neemt het aantal verbindingen kwadratisch toe (aantal verbindingen gaat met het aantal cellen gekwadrateerd); voor een groot systeem speelt deze maximum afstand *wel* een rol: een neuron in de ene hoek zal niet verbinden met een neuron in de andere hoek van de Neocortex. Het aantal verbindingen heeft dan een bepaalde limiet bereikt en neemt lineair toe met het aantal neuronen (zie opnieuw hoofdstuk 3). Figuur 4.5 bevestigt dit gedrag. Een kanttekening hierbij is dat deze figuur terug te koppelen is aan figuur 4.4: de simulatieruntijd linkt sterk terug met het aantal verbindingen en dus de 'geavanceerdheid' (complexiteit) van het neuronennetwerk.

Het totaal aantal verbindingen zegt niet bijzonder veel over het netwerkgedrag; dit aantal zou hooguit enkele informatie kunnen geven over de runtijden uit de vorige paragraaf. Veel significanter is de verhouding tussen de excitatoire verbindingen en de inhibitoire verbindingen in het model, oftewel de excitatie-/inhibitieratio die geïntroduceerd is in hoofdstuk 3. Zoals intussen al bondig is aangehaald, verandert deze als vanzelf bij probleemvergroting, een duidelijk gevolg van de implementatie van figuur 3.1. Het netwerk krijgt initieel steeds meer fluctuërende neigingen en bij 'echt' grotere schaal netwerken juist inhibitoire verschijnselen, onder meer zichtbaar in de volgende paragraaf. De gemeten excitatie-/inhibitieratio (totaal-excitatie gedeeld door totaal-inhibitie) tegen het aantal neuronen is geplot in figuur 4.6, voor grote systemen. Hier zien we het groeiende netwerk op deze schaal duidelijk steeds minder excitatoir worden. In het begin verandert dit snel (en drastisch), aan het eind gaat dit langzamer - lopend naar een limiet/asymptoot van ongeveer 8, zoals in hoofdstuk 3 is benaderd. Merk op dat uit de EEG's de gedragsveranderingen tijdens geleidelijke systeemvergroting eruitzien als een route naar iets meer excitatoir naar heel inhibitoir, wat níét wordt herkend in figuur 4.6, ook niet bij inzoomen op deze kleine schaal.

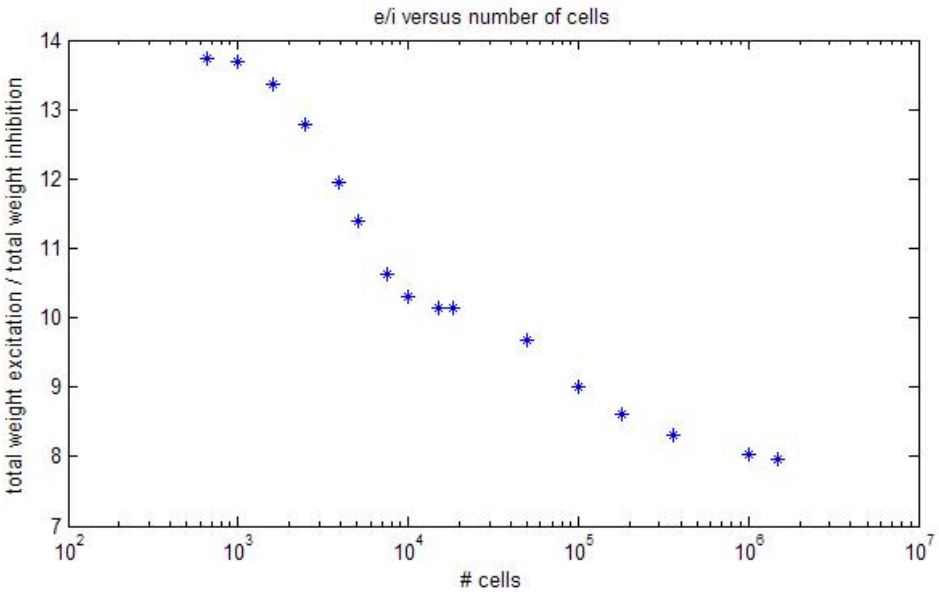
We verwachtten dat de gemiddelde reistijd van actiepotentialen naar een limiet toeliep, en wel naar ongeveer 7.17 ms (zie hoofdstuk 3). De hypothese is dat de gemiddelde reistijd eerst groeit met het aantal neuronen en langzaam naar deze 7.17 ms afvlakt (dus geleidelijk horizontaal wordt langs een limietwaarde). Figuur 4.7 laat zien dat deze hypothese juist is.

4.3 Globale variatie van excitatie en inhibitie

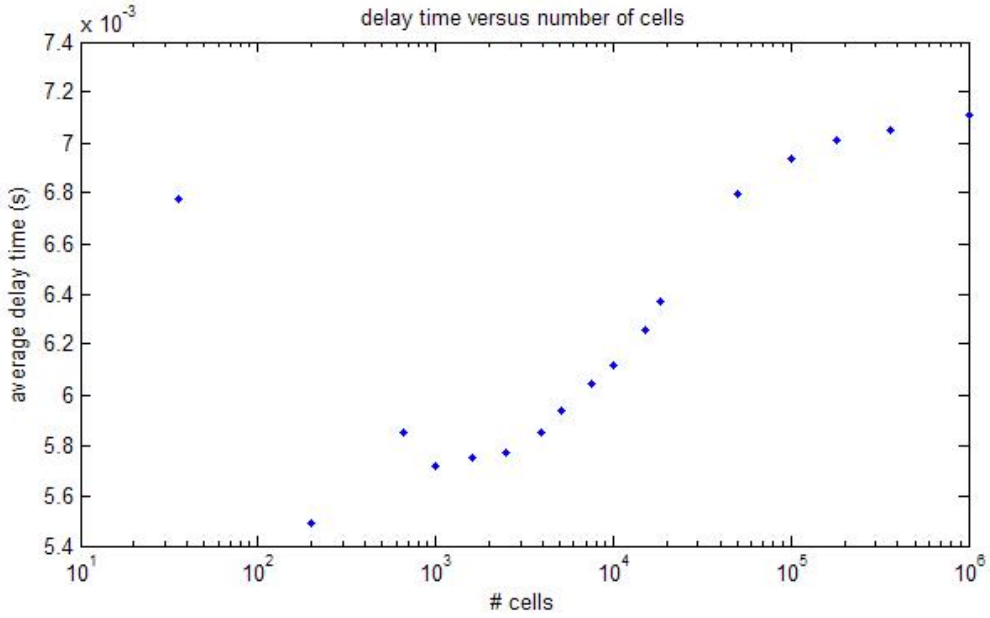
De sterkte van de verbindingen tussen de virtuële neuronen kunnen natuurlijk handmatig worden gevarieerd. Men zou alleen de excitatoire verbindingen sterker of zwakker kunnen maken, of juist alleen de inhibitoire - om maar een voorbeeld van de mogelijkheden voor gewichtsvariatie te noemen. In figuur 2.9 stond een E-I-ruimte opgenomen voor 1000 neuronen. Deze ruimte fixeerde zich nog al op het opstartgedrag. In figuren 4.8-4.13 worden dergelijke ruimten getoond voor een



Figuur 4.5: Simulatie-output voor het totaal aantal verbindingen (excitatoir + inhibitoir) tegen het aantal neuronen bij continue celdichtheid (aantal cellen per kubieke meter). De onderste afbeelding is een zoom-in op het eerste gedeelte (kleine systemen) van de bovenste afbeelding. Op kleine schaal is het verband bij benadering kwadratisch; bij grotere problemen wordt geleidelijk overgegaan op lineair gedrag.



Figuur 4.6: Ratio van de totale excitatie tegen de totale inhibitie van het netwerk, oftewel: de gewichtensom van alle excitatoire verbindingen gedeeld door de gewichtensom van alle inhibitoire verbindingen in het netwerk, uitgezet tegen de systeemgrootte (aantal cellen).De figuur betreft simulatie-output. Hoe hoger de plek in het diagram, des te meer excitatoir het netwerk is. We zien een snel afvlakkende lijn naar een zekere vaste (limiet)waarde, ongeveer 8, zoals benaderd in hoofdstuk 3. Wat bovendien opvalt is de 'kronkel', ver voor de limiet bereikt wordt, bij ongeveer 10.000 cellen. Deze kronkel oogt wat vreemd, maar is in theorie óók weerlegbaar aan de hand van de tabel in hoofdstuk 3.



Figuur 4.7: Gemiddelde reistijd ('delay') van actiepotentialen van álle (excitatoire en inhibitoire) verbindingen in het neuronennetwerk, tegen het aantal neuronen. De figuur betreft simulatie-output. Het verband vlakt af naar een (redelijk) horizontale lijn. We vinden een limiet van ongeveer 7.10 ms. Dit afvlakkende verloop is als verwacht. De metingen uiterst links in het diagram geven aan dat de betrouwbaarheid van een dergelijk klein probleem te klein is, omdat deze metingen te ver buiten de verwachtingen vallen.

opklimmend aantal neuronen, maar dan voor output van NeuroSim, en gericht op het limietgedrag: het gedrag na een bepaalde (stabilisatie)tijd. Hieraan kunnen we immers de totaalnetwerkexcitatie koppelen; het opstartgedrag zegt ons daar niet zoveel over. De parameters E:I=1.0:1.0 betekenen een ongewijzigd netwerk voor wat betreft de gemiddelde verbindingssterkten; 1.0:20.0 betekent dat de *verwachte* sterkte van alléén de inhibitoire verbindingen met 20 is vermenigvuldigd ten opzichte van de normale verbindingssterkte (door $minW$ en $maxW$ uit de tabel voor de betreffende (inhibitoire) verbindingstypen met 20 te vermenigvuldigen), en 0.2:1.0 dat de *verwachte* sterkte van alléén de excitatoire verbindingen met 0.2 is vermenigvuldigd ten opzichte van de normale sterkte. Op deze manier kunnen we het netwerk behoorlijk bespelen, dus sterk in de excitatoire of inhibitoire richting pushen. Let op: 2.0:2.0 betekent wel degelijk iets anders dan 1.0:1.0 (bij wijze van voorbeeld); de verhouding tussen de excitatie- en inhibitieparameters vertelt ons niet genoeg om de netwerkeigenschappen te ontrafelen. Dit komt doordat excitatoire verbindingen nu eenmaal een ander effect op neuronen hebben dan inhibitoire verbindingen.

Bedenk dat het enige voorbereiding en overwegingen kost om tot het construeren van dergelijke vlakken te komen. Hoe variëren we het aantal neuronen (welke waarden voor N kiezen we?), tussen welke waarden variëren we de excitatie E en tussen welke waarden de inhibitie I, en op welke manier verwerken we de multistabiliteit in de vlakken? Twee even grote systemen met dezelfde verbindingen en parameters kunnen immers in verschillende toestanden terechtkomen, afhankelijk van (in dat geval uitsluitend) de begincondities.

De belangrijkste ('overall') toestanden die we over het algemeen in NeuroSim EEG's herkennen zijn '(bijna complete) stilte', 'stilte met af en toe een groepsgewijze puls', 'groepsgewijze oscillaties' en 'fluctuaties (ongeordend gedrag)', vaak afgewisseld in 1 EEG, bijvoorbeeld 'fluctuaties→oscillaties' of 'afwisselend stilte & oscillaties'. Hierbij moet, zoals vermeld, duidelijk onderscheid gemaakt worden in opstartgedrag en limietgedrag. Multistabiliteit verwerken in de vlakken is in feite onmogelijk. Om zeker te weten of een systeem nóóít in een andere eindtoestand terecht kan komen dan die we 'almaar' krijgen, zouden we letterlijk oneindig veel experimenten moeten uitvoeren. Wel geldt: hoe vaker we de experimenten herhalen (bij steeds andere begincondities natuurlijk), des te meer kans we zullen verkrijgen op meerdere (wellicht alle mogelijke) soorten limietgedrag (gedrag na een bepaalde tijd - dus ná het opstartgedrag). Toch hebben we de experimenten behorend bij de E-I-vlakken in deze paragraaf maar één keer uitgevoerd, vooral wegens een praktisch probleem: benodigdheid van zeer veel simulatietijd. Het gedrag blijkt overigens niet zeer veel te variëren bij andere begincondities. In het 'vervelendste' geval zal een systeem in experiment 2 wel groepsgewijs gaan oscilleren en in experiment 1 niet (of andersom), vanuit fluctuaties dan wel stiltegedrag. De kans om in één systeem zowel stilte (als oscillaties,) als fluctuaties te verkrijgen is klein, maar niet nihil, zoals we een enkele keer zien in de EEG's in deze paragraaf. We moeten ons op dit moment dus tevreden stellen met de opmerking dat er veel meer scenario's mogelijk zijn dan weergegeven in de E-I-vlakken, ook afhankelijk 'vanaf welk ander punt in het E-I-vlak we komen' (hysterese). Waar het nu om gaat is aan de hand van deze figuren al duidelijk in beeld gebracht: het verloop naar excitatoir (overwegend (oscillaties en) fluctuaties) en daarna juist terug naar zwaar inhibitoir totaalgedrag (overwegend stilte (en oscillaties)), bij vergroting van het aantal neuronen.

De aantallen neuronen waarbij de figuren in deze paragraaf geplot zijn, zijn (vooral) gebaseerd op figuur 4.6. Daar zien we dat vanaf een neuron of 10.000 het netwerkgedrag een limiet bereikt en het niet veel zin heeft om het aantal neuronen veel verder te vergroten; daar zullen ongeveer dezelfde (inhibitoire) resultaten uitkomen. Het was ook een mogelijkheid geweest om (ongeveer) *lineair* van ∼1000 naar ∼20.000 neuronen te lopen.

Schaling van E is gebaseerd op eerdere output van NeuroSim voor 656 neuronen. Bij E:I=0.2:1.0 was hier stilte met af en toe een puls te zien, bij 1.0:1.0 fluctuaties gaande naar oscillaties en bij 2.0:1.0 uitsluitend fluctuaties. Het hele gebied inhibitoir→excitatoir werd nu netjes bestreken, vandaar het gekozen bereik E=0.2-2.0. Het I-interval hadden we tussen dezelfde grenzen kunnen kiezen, maar deze zijn bewust iets breder gekozen, om wellicht een tikkeltje meer gedrag naar voren te kunnen halen. De E- en I-intervallen komen in feite niet zo precies - men had eventueel ook E van 0.02 naar 20.0 kunnen laten lopen. Wanneer we E of I echter *te* groot kiezen, dan is

er sprake van een verzadiging (zowel in theorie als programmatechnisch gezien): de betreffende verbindingen worden dan zó sterk, dat nóg groter maken niet zo veel verschillen in netwerkgedrag oplevert. Een dergelijk principe geldt ook in de andere richting: de verbindingen lopen tegen een soort gedragslimiet aan als de gemiddelde verbindingssterkte *te* klein wordt (in zo'n geval hadden we de betreffende verbindingssterkten evengoed gewoon gelijk aan 0 kunnen kiezen).

Voorop staat telkens wanneer we van linksboven naar rechtsonder in het E-I-parametervlak lopen, de route inhibitoir→excitatoir: we lopen van (bijna) complete stilte naar stilte met een groepsgewijze puls naar stilte afgewisseld met groepsgewijze oscillaties naar groepsgewijze oscillaties en fluctuaties naar uitsluitend fluctuaties. Wanneer het aantal neuronen vergroot wordt (656-5248), groeit het fluctuatiegebied initieel. Vergroten we nog verder (5248-20992), vergroot juist het stilte-met-puls-regime zich sterk. Een conclusie die we op voorhand al voorspeld hadden, want we wisten immers dat het netwerk uiteindelijk zeer inhibitoire neigingen zou krijgen. Kortom: figuren 4.8-4.13 bevestigen nogmaals het verloop dat in paragraaf 4.2 werd geconstateerd.

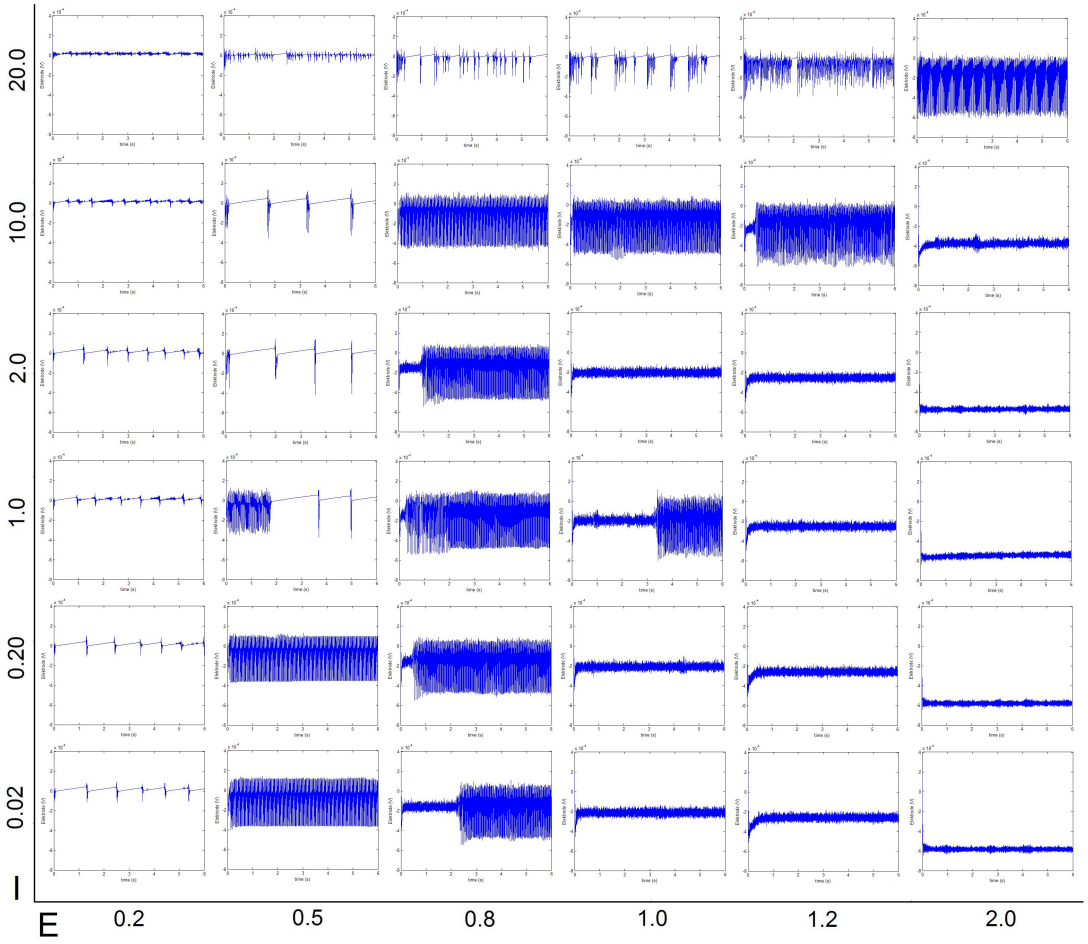
4.4 Ruimtelijk gedrag

Vooralsnog is, voor wat betreft de output voor grote schaal breinmodellen in NeuroSim, uitsluitend gekeken naar de totaalnetwerkexcitatie. Het valt echter voor te stellen dat er méér is. Zien we bijvoorbeeld verplaatsingen in de ruimte van actiepotentialen, van bursts met opeenvolgingen van actiepotentialen of van compleet groepsgedrag? Voor een klein systeem van 656 neuronen werden al verticale verplaatsingen geconstateerd (van de DPyr-cellen naar de SPyr-cellen via de interneuronen, bijvoorbeeld), maar nog geen overtuigende horizontale, omdat de horizontale afmetingen wat te klein waren om dergelijke zaken te onderscheiden (zie figuur 4.15).

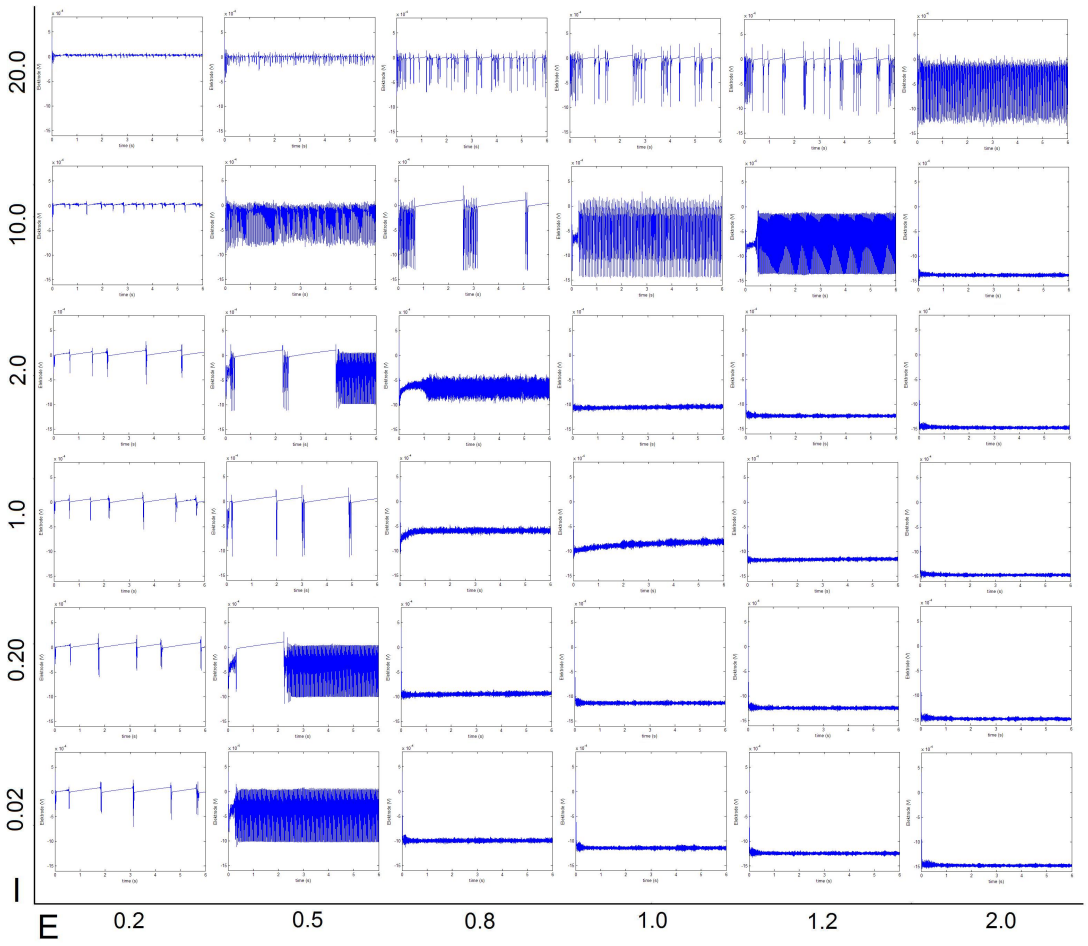
In het geval van fluctuërende AP-vuring valt eigenlijk niets over tijdsverschillen tussen locaties in het model te zeggen (omdat alle neuronen immers door elkaar vuren), maar als sprake is van groepsmatige pulsen of oscillaties, dan ligt dat anders. In dat geval gedragen de groepen met neuronen verdeeld over het model zich wel netjes ongeveer hetzelfde, maar op een *net* iets ander moment, (mede) afhankelijk van de locaties en de tussenruimten van de neuronen. Het ene hersengedeelte heeft bij een grote schaal breinmodel wat vertraging ten opzichte van het andere hersengedeelte. Onder meer om dergelijk gedrag te kunnen herkennen, kunnen de SPyr-cellen die het EEG leveren worden onderverdeeld in bijvoorbeeld 16 stukken, gebaseerd op de (x, y)-posities (bovenaanzicht) van de betreffende neuronen. Zie figuur 4.14. We kunnen nu doorgaans het EEG van stuk 11 (linksboven) met het EEG van stuk 44 (rechtsonder) vergelijken, of het groepsgedrag van stuk 11 met dat van het dichterbij liggende stuk 12 (zie figuur (4.15 en) 4.16). Inderdaad.., voor systemen die groot genoeg zijn nemen we kleine tijdsverschillen waar.

4.5 Gamma ritmen

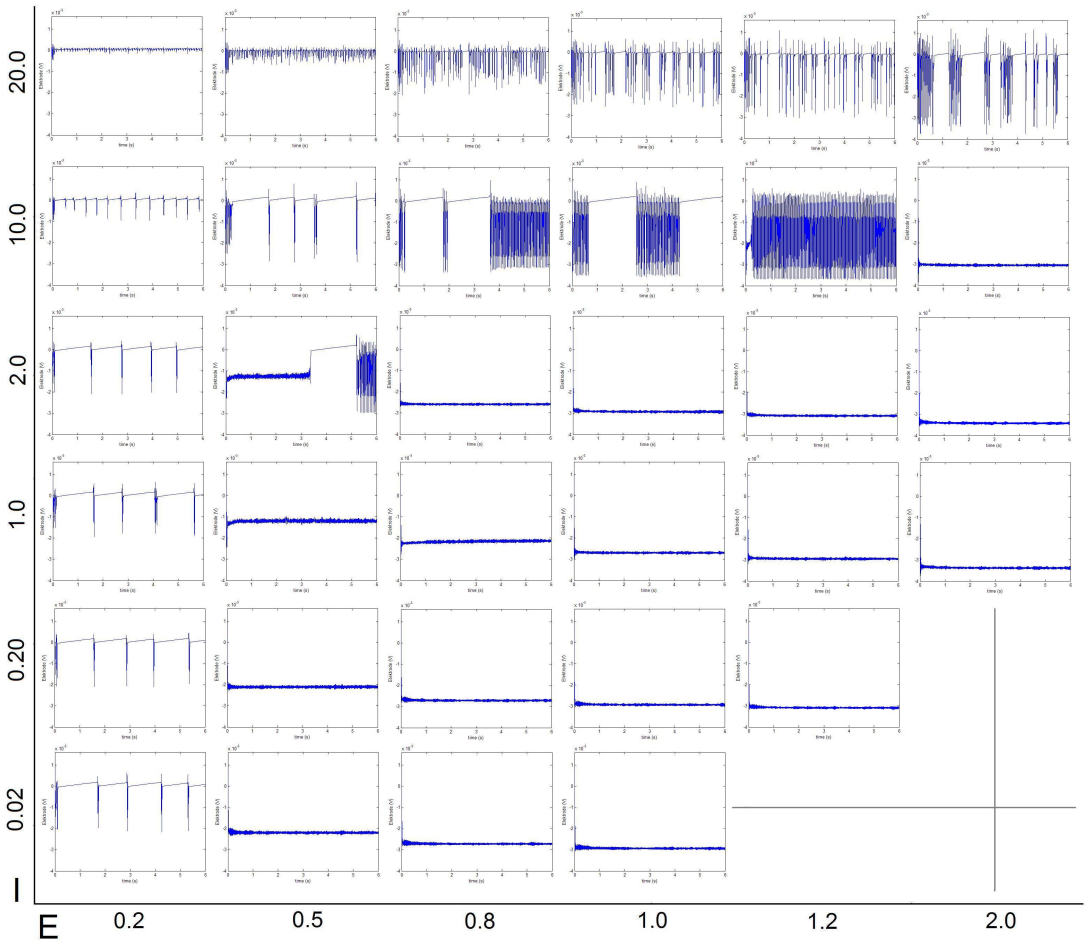
De hypothese was dat de gamma ritmen in het geval van groepsgewijze oscillaties een lagere frequentie zouden hebben (liggende vermoedelijk in het beta domein) bij grotere systemen. Dit blijkt kortgezegd inderdaad het geval te zijn (zie figuur 4.17), maar er is meer aan de hand. Merk op dat de ritmen kunnen variëren voor hetzelfde hersensysteem (van hetzelfde aantal neuronen), bijvoorbeeld zichtbaar in figuur 2.11, afhankelijk van bepaalde parameters. Dit maakt precieze analyse van de groepsritmen lastiger. Bij 656 neuronen blijken de groepsgewijze oscillaties allemaal te variëren tussen 37 en 44 Hz (gebaseerd op de EEG's in figuur 4.8), wat een vrij hoog ritme is, hoewel voor een dergelijk klein systeem niet abnormaal (zie deelparagraaf 2.6.2). Bij 1312 neuronen valt er een soort scheiding te trekken tussen de wat excitatoire oscillaties (gepaard met fluctuaties) en de wat inhibitoire oscillaties (gepaard met stilte), zogezegd. De frequenties bij de parameters E:I=2.0:20, 0.5:10 en 0.8:10 zijn laag, tussen 30 en 36 Hz; de frequenties bij de parameters E:I=1.2:10, 0.8:2.0, 0.8:0.2 en 0.8:0.02 zijn juist hoog, allemaal rond de 46 Hz (zie figuur 4.9). Een stap verder, 2624 neuronen, waar amper oscillaties in het E-I-vlak te vinden zijn (zie figuur 4.10), ligt de frequentie voortdurend rond de 35 Hz, maar is het EEG een stuk



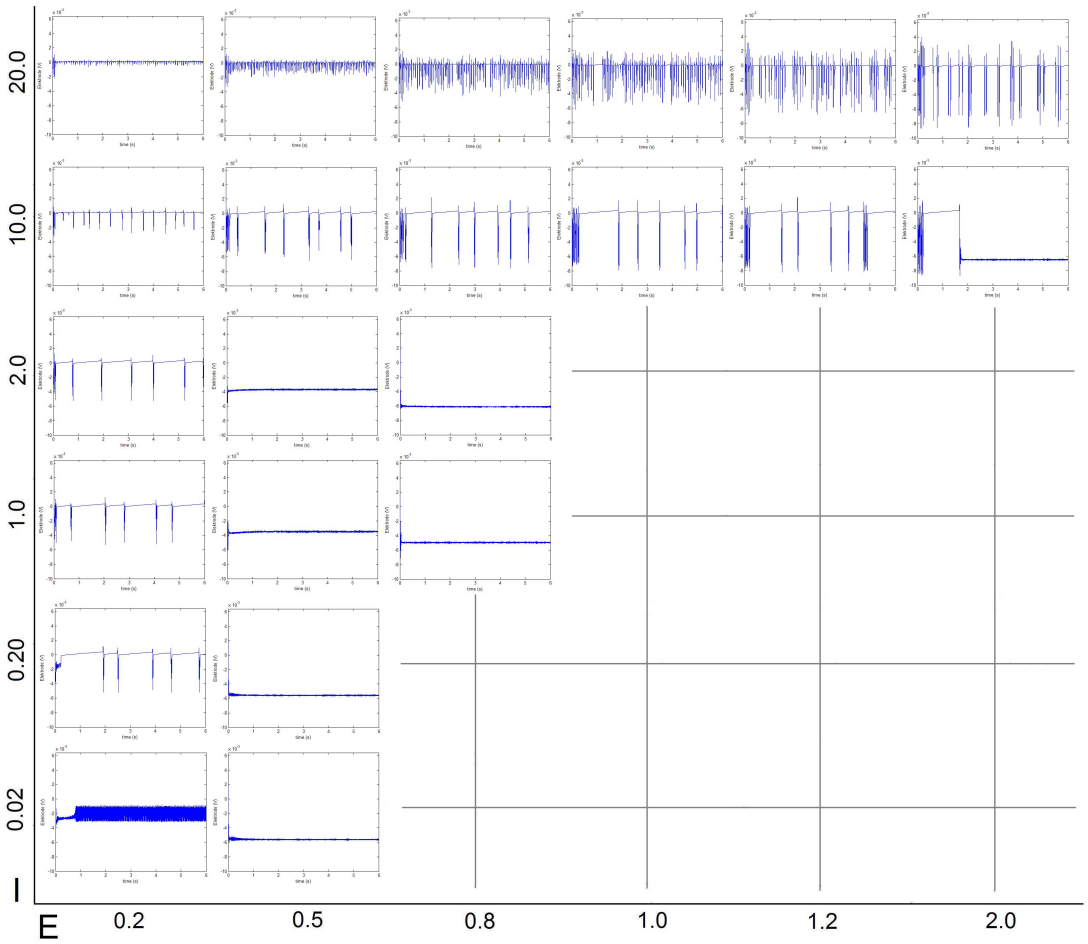
Figuur 4.8: Excitatie-/inhibitieparametervlak met de EEG's voor 656 neuronen (256 SPyr-cellen).



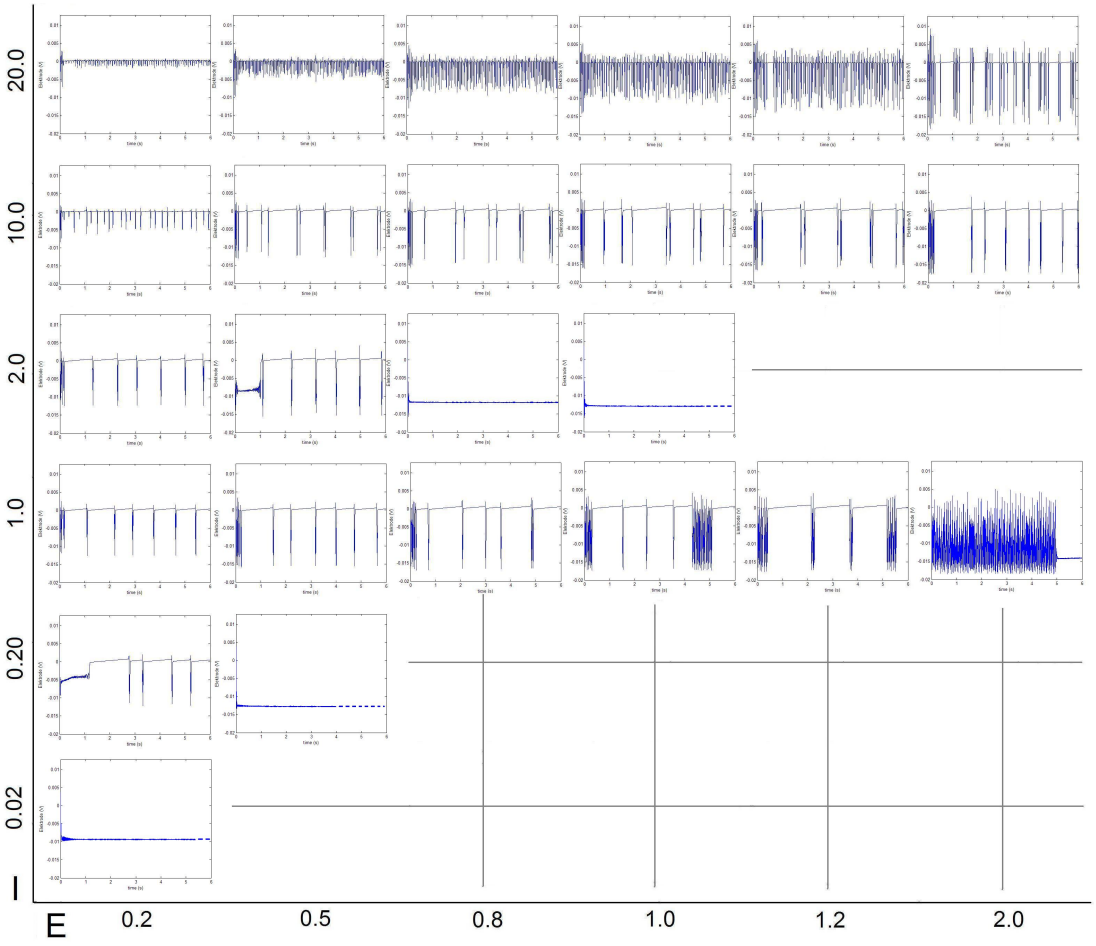
Figuur 4.9: Excitatie-/inhibitieparametervlak met de EEG's voor 1312 neuronen (512 SPyr-cellen).



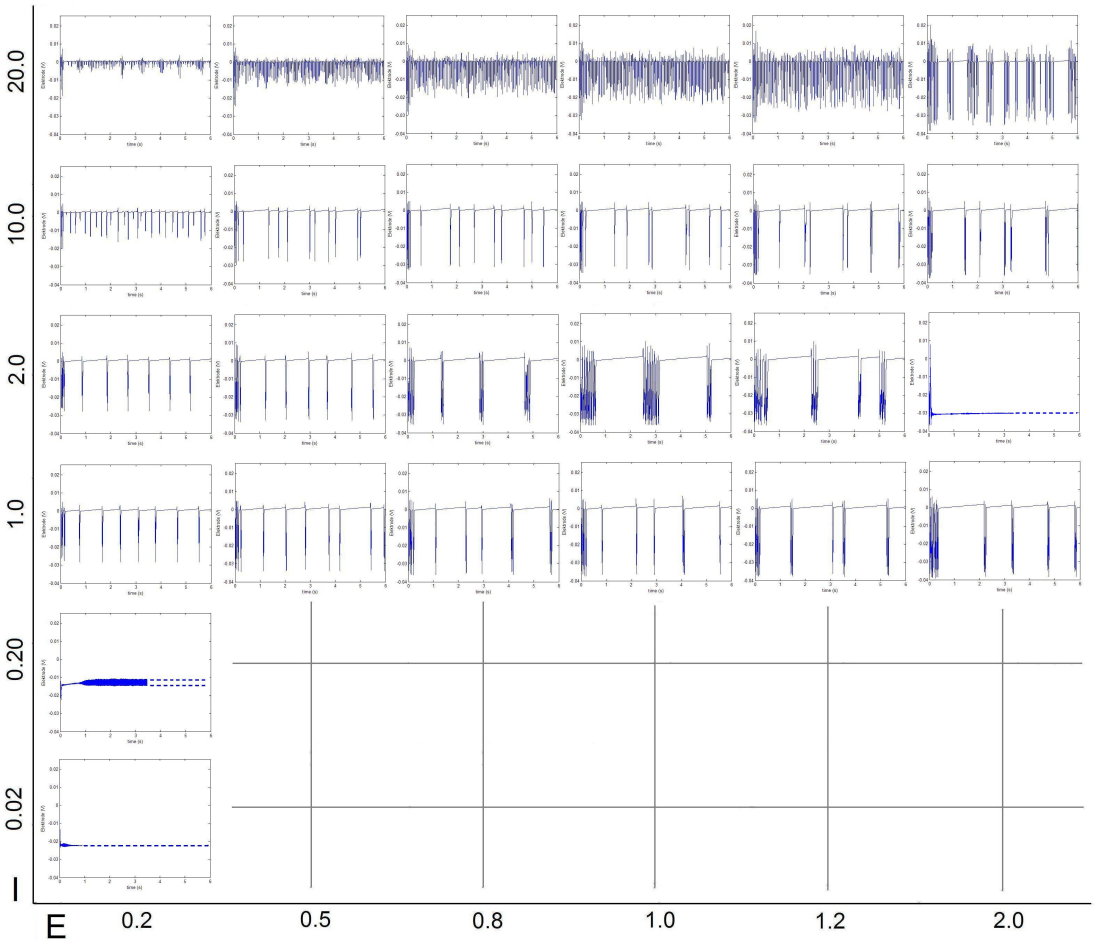
Figuur 4.10: Excitatie-/inhibitieparametervlak met de EEG's voor 2624 neuronen (1024 SPyr-cellen).



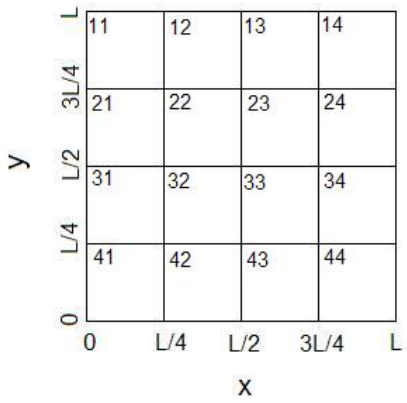
Figuur 4.11: Excitatie-/inhibitieparametervlak met de EEG's voor 5248 neuronen (2048 SPyr-cellen).



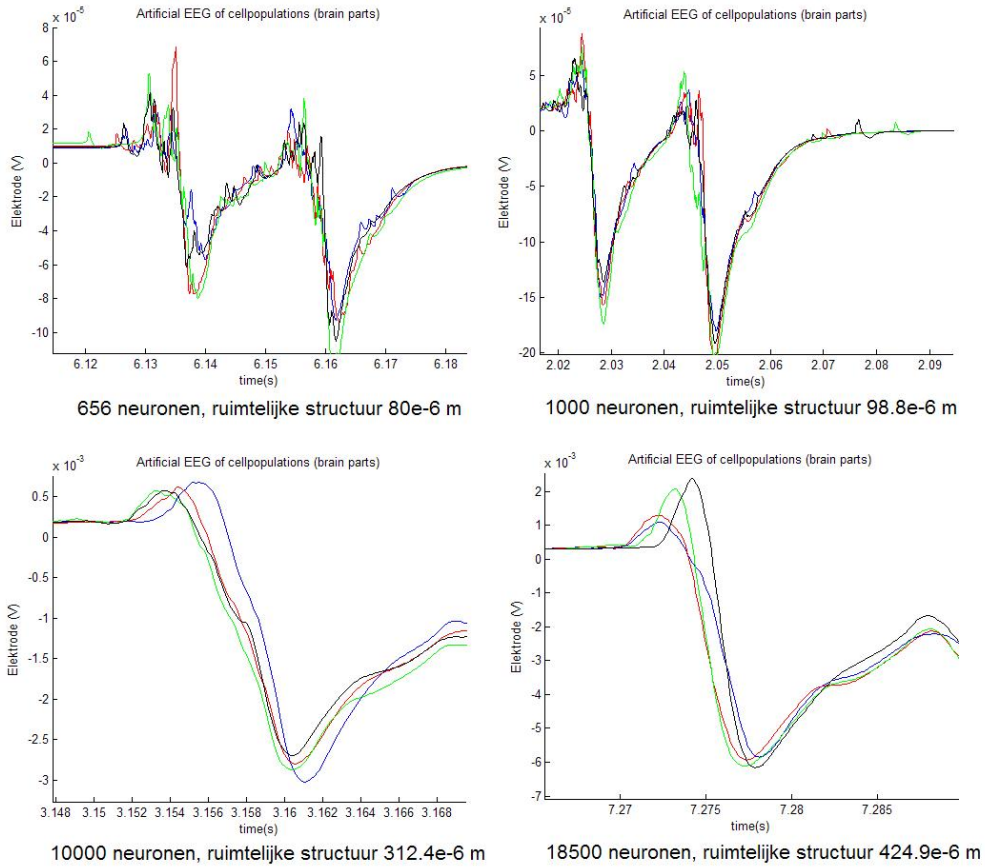
Figuur 4.12: Excitatie-/inhibitieparametervlak met de EEG's voor 10496 neuronen (4096 SPyr-cellen).



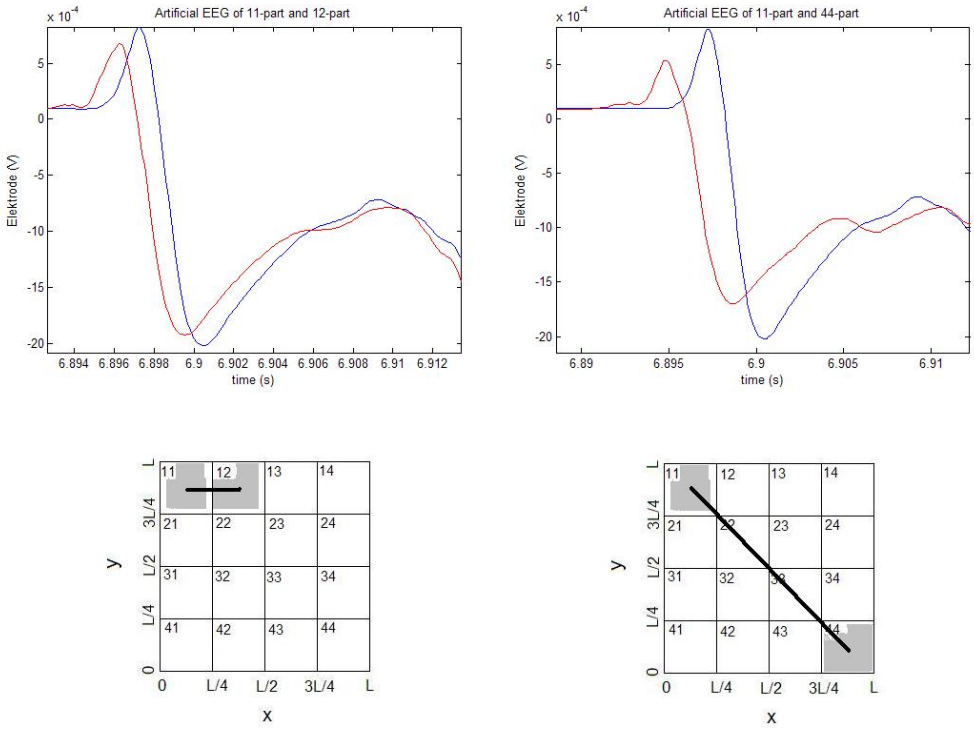
Figuur 4.13: Excitatie-/inhibitieparametervlak met de EEG's voor 20992 neuronen (8192 SPyr-cellen).



Figuur 4.14: Verdeling van de neuronen gebaseerd op (x, y)-locatie (de afbeelding is het bovenaanzicht van het gemodelleerde stukje Neocortex), met als doel het zichtbaar maken van (eventuele) gedragsverschillen in de ruimte. Hoe groter het systeem, des te meer van toepassing wordt deze verdeling in zestien (of meer) stukken.



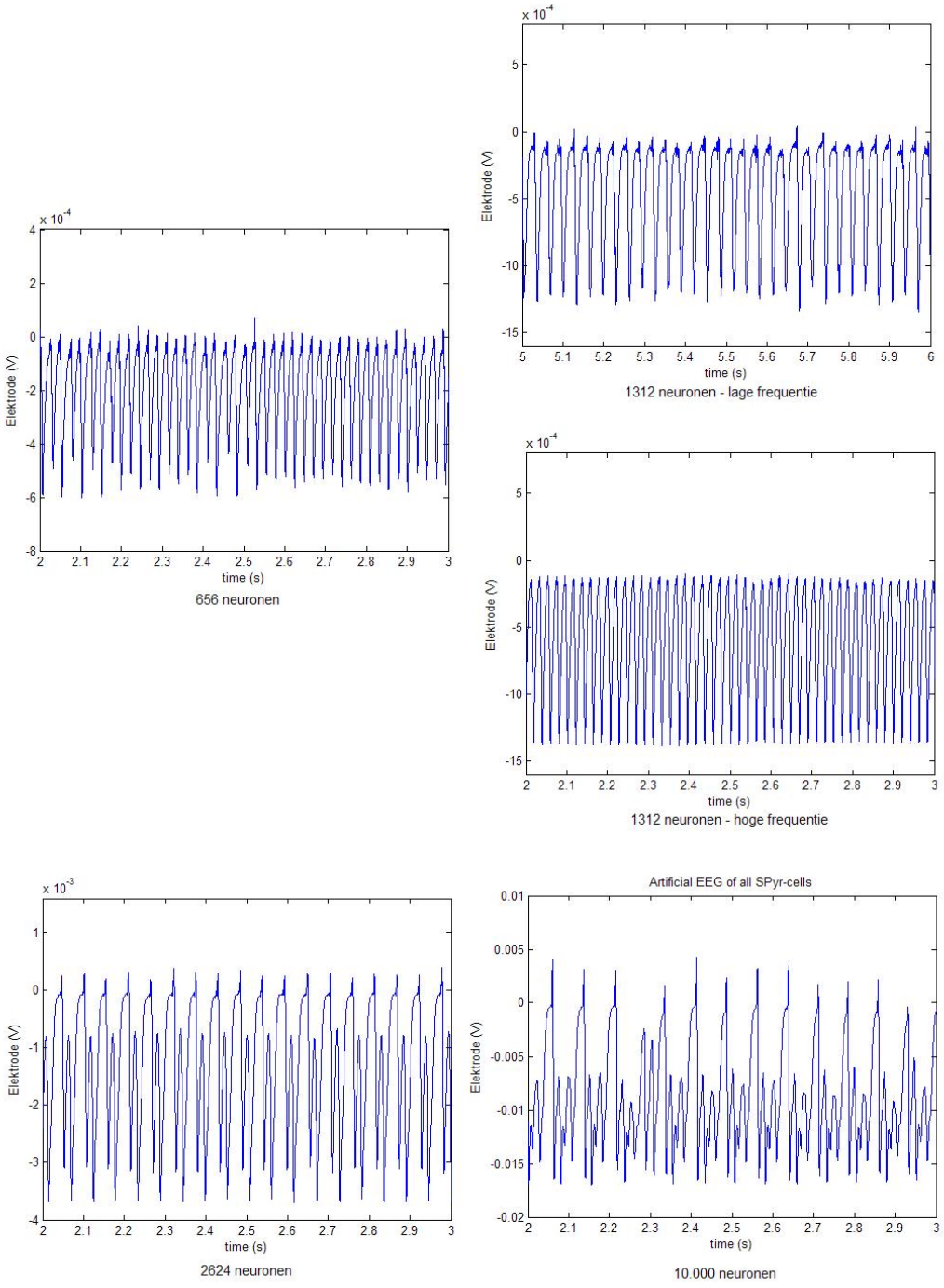
Figuur 4.15: Plots van een sterke zoom op een groepspuls in het EEG voor de 4 verticale (complete) stroken uit figuur 4.14 voor 656 neuronen (linksboven), 1000 neuronen (rechtsboven), 10000 neuronen (linksonder) en 18500 neuronen (rechtsonder). Elke lijn per plot is het EEG van alle SPyr-cellen binnen zo'n strook (blauw is de meest linker strook, daarna volgen rood, groen en zwart, respectievelijk). Bij 656 en 1000 neuronen is het gedrag nog wat chaotisch, hoewel bij 1000 al minder dan bij 656. Hier is het ruimtelijke verplaatsingsgedrag nog niet goed te herkennen. Niet alleen wegens de lichte chaos, maar ook omdat de afmetingen domweg te klein zijn om dergelijke verschillen duidelijk te kunnen zien, als ze er al zijn. Bij 10000 neuronen en, beter nog, bij 18500 neuronen is dat niet meer het geval, de AP-vuringen lijken beter gerangschikt (lees: een puls van een afzonderlijk neuron beïnvloedt het somsignaal niet zoveel meer), en zien we de lijnen elkaar (met heel lichte vertragingen) opvolgen met een gering tijdsverschil. Anders gezegd: ze liggen niet meer netjes over elkaar heen. Bij 10000 neuronen lijkt het gedrag zich als het ware van rechts naar links te verplaatsen over de SPyr-cellen (gezien de lijnkleuren), bij 18500 neuronen juist andersom.



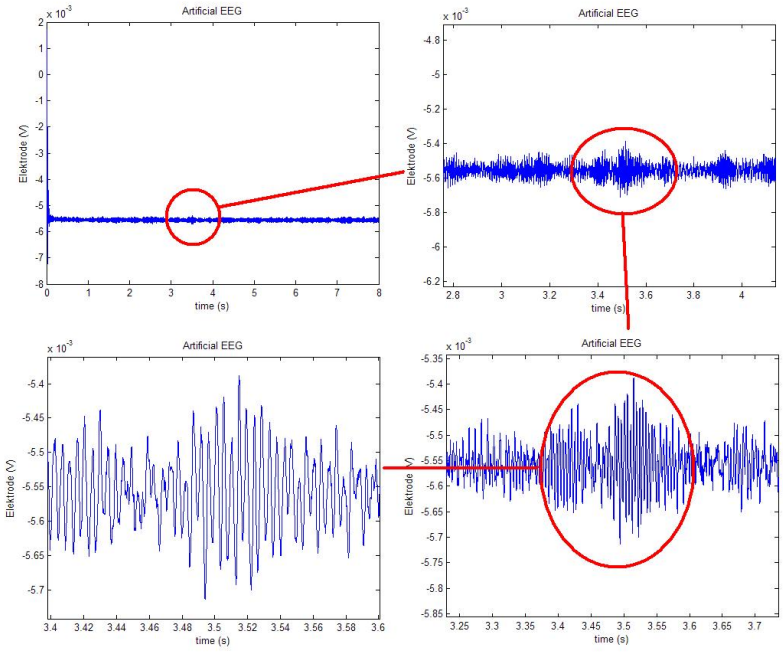
Figuur 4.16: Blokgedragsvergelijkingen (zie ook figuur 4.14) bij 18500 neuronen. Vergeleken is een puls in het EEG op het 11- (blauw) met die op het 12-blok (rood) (links) en dezelfde puls op het 11- (blauw) met die op het 44-blok (rood) (rechts). De puls in blok 11 loopt dus achter ten opzichte van de bijbehorende (*bijna* hetzelfde moment) in blok 12 en die in blok 44. De rechterafbeelding vergelijkt tussen twee blokken die een stuk verder uit elkaar liggen dan de twee uit de linkerafbeelding. Dat zien we ook terug in de vertraging tussen de twee pulsen: het tijdsverschil links is ongeveer 0.92 ms en rechts ongeveer 2.41 ms. Het is niet zo dat deze verschillen in vertragingen overtuigend rechtlijnig schalen met de afstand tussen de twee blokken.

chaotischer geworden. De groepspulsen zijn niet meer duidelijk (de één is hoog, de ander laag, de frequentie varieert wat: het lijkt alsof er steeds een burst van 2 pulsjes herhaald wordt). Bij een groot aantal neuronen, bijvoorbeeld 10.000, is de frequentie nog steeds van deze grootte, en is het EEG nóg chaotischer geworden.

Houd in gedachten dat de meeste neuronen best zo'n 100 AP's per seconde kunnen vuren (soms wordt zelfs wel 1000 gesuggereerd). Frequenties van honderden Hz zijn dan ook geregeld terug te vinden in realistische of kunstmatige EEG's. Een netwerkje van slechts een handjevol (minimaal twee) (vurende) (dicht bij elkaar zijnde) neuronen hebben een grote kans om samen met een zeer hoge frequentie regelmatig te gaan vuren. Ook in onze fluctuërende EEG's vinden we bij ver inzoomen af en toe een periodiciteit met zeer hoge frequentie terug. Zie bijvoorbeeld figuur 4.18. Hoe groter het netwerk wordt, des te moeilijker (en onnatuurlijker) het wordt om deze frequenties vast te houden en ook naar voren te krijgen. Figuur 4.18 laat zien dat de term 'fluctuaties' feitelijk al gevaarlijk is, omdat er dus wel degelijk een zekere vorm van structuur te herkennen valt binnen de chaos. Echter, deze structuur vindt maar plaats op zeer kleine schaal (een paar neuronen, is de schatting).



Figuur 4.17: EEG-output in het geval van groepsgewijze oscillaties, bij verschillende systeemgrootten: naarmate het systeem groter wordt, wordt het pulsgedrag geleidelijk steeds chaotischer. De ritmen verlagen van ongeveer 40 Hz naar ongeveer 35 Hz. Ook ritmen van slechts $\sim$30 Hz en van wel $\sim$50 Hz worden geconstateerd. N.B.: gedoeld wordt op de pulsen richting onder!



Figuur 4.18: EEG-meting voor 5100 neuronen, excitatie-/inhibitieparameters 1.0:1.0. Naarmate we ver inzoomen herkennen we wel degelijk een vorm van structuur binnen de fluctuaties; een structuur met zeer hoge frequentie.

Hoofdstuk 5

Conclusies & discussie

5.1 Gebruik van OpenMP

De vorm van de grafieken in figuren 4.1, 4.2 en 4.3 is keurig volgens de Wet van Amdahl (afvlakkend), maar de prestaties zijn enigszins teleurstellend te noemen: het lineaire gebied van de speed-up lines is erg klein. De curve begint eigenlijk meteen bij 2 CPU's al sterk af te vlakken. Blijkbaar heeft het (voor deze implementatie) weinig zin om te testen hoe snel het programma is bij, zeg, 1000 CPU's: extrapolatie zou laten zien dat een dergelijk aantal processoren de simulatie nog amper zou versnellen of mogelijk zelfs juist vertragen in verband met de vele communicatie.

Hoe zou men het gedrag van figuren 4.1, 4.2 en 4.3 kunnen verklaren? Merk allereerst op dat parallelrekening soms wat onbegrepen resultaten geeft - er kunnen allerlei achtergrondzaken ten grondslag liggen aan het snelle afvlakken van de speed-up lines. Het rekenwerk per tijdstap zal behoorlijk verschillen per cel in het systeem. Een virtuële zenuwcel die op een zeker tijdstip t een AP vuurt, vereist misschien wel minstens vier keer zo veel calculeer-/opslag/...-werk als een zelfde cel die op dat moment 'in rust' is. Bij de huidige implementatie van NeuroSim wordt, en dat is vanzelfsprekend, parallel gerekend over de cellen en niet over de totale hoeveelheid werk(, wat ook zeer complex, misschien wel onmogelijk, om toe te passen zou zijn). Zodra één CPU nog niet 'klaar' is met de bijbehorende 'thread', moeten álle andere CPU's wachten. Hoe groter het aantal CPU's is, hoe meer van belang dergelijke wachttijd wordt (omdat dan álle CPU's minus één op ééntje moeten wachten).

Een andere conclusie die aan de drie speed-up lines verbonden kan worden is dat het afvlakgedrag doorgaans hetzelfde is: bij élke probleemgrootte wordt nagenoeg hetzelfde verband tussen de runtijd en het aantal CPU's geconstateerd. Inzet van zo'n 20 CPU's betekent een simulatiesnelheid van ongeveer 5 maal de 1-CPU-snelheid.

Ondanks dat de versnelling misschien niet erg bevredigend is voor een groot aantal CPU's, zien we *wel* een significante versnelling in rekentijd. Voor 18500 neuronen is een rekentijd van 2 uur ten opzichte van de oorspronkelijke ruim 10 uur nog altijd een sterke verbetering! Een 'stuk' brein van 18500 neuronen is voor NeuroSim al behoorlijk groot (ruimtelijke structuur: 0.425 mm X 0.425 mm), dat moet niet onderschat worden. Wanneer echter naar nóg grotere gedeelten van de Neocortex gekeken zou moeten worden, zal implementatie van MPI in plaats van OpenMP absoluut een goede investering zijn. Met MPI kan (moet) immers communicatie tussen de cellen gecoördineerd worden, in tegenstelling tot de situatie met OpenMP.

Conclusies

De benodigde tijd van een simulatie met NeuroSim schiet omlaag bij toepassing van OpenMP, maar de SpeedUp lines vlakken vrij snel af naar een horizontale lijn. Het rekentijdsverschil tussen een simulatie met 20 CPU's en één met 50 CPU's is nog maar weinig. Het stijle gebied van de relatie tussen de simulatietijd en het aantal processoren is dus klein.

Het doel van gebruik van OpenMP was echter niet het zo snel mogelijk maken van NeuroSim.

Oogmerk was naar grotere hersenkolommen te kunnen gaan binnen afzienbare rekentijd en dat is zéker gelukt met OpenMP. Bij gebruik van 2 CPU's is de runtijd al bijna 2 maal verkleind en bij gebruik van 20 CPU's is dit minstens 5 maal. We stellen als maximum werkbare systeemgrootte zo'n 18.500 neuronen op, met de kanttekening dat dit aantal neuronen in het geval van een hoge netwerkexcitatie eveneens een paar dagen rekentijd kost, ook met veel ingezette CPU's. Met overtuiging wordt geconstateerd dat voor echt grote hersensystemen (één of meerdere hyperkolommen) méér nodig is dan OpenMP. MPI is dan de voordehandliggende optie.

5.2 Resultaten bij grote schaal netwerken

De 'kronkel' in figuur 4.4 valt op een 'speelse' manier te verklaren: vanaf ongeveer 4000 neuronen wordt het netwerk vanzelf behoorlijk inhibitoir, wat gunstig is voor de rekentijd. De grafiek stijgt natuurlijk nog wel, maar minder stijl. Bij ongeveer 8000 neuronen komt de totaalnetwerkexcitatie in de buurt van een limiet (nog meer inhibitoir zal het netwerk niet meer worden vanaf dan) en herstelt de grafieklijn zich naar ongeveer parallel aan vóór de kronkel om verder weer lineair toe te nemen.

Houd in gedachten dat figuur 4.4 slechts een 1-CPU-voorbeeld is van een verloop bij bepaalde parameters en beginvoorwaarden. Zoals inmiddels is uitgelegd, is in het algemeen (voor het originele model), en zéker bij toepassing van parallelrekening, geen vastliggend verband op te stellen. Een 8-seconden-simulatie van 5100 neuronen duurt, zonder handmatige parameteraanpassingen tijdens het vergroten van het probleem, met 50 CPU's bijna een etmaal (ca. 22.5 uur), een simulatie van 10000 neuronen een uur of 10 en ééntje van 18500 zo'n 14 uur! Dit vreemde verloop heeft te maken met het feit dat bij 5100 neuronen (zoals blijkt in de bijbehorende output) er zeer veel AP's worden gevuurd, in tegenstelling tot bij 10000 of 18500 neuronen, waarbij het systeem zich behoorlijk inhibitoir is gaan gedragen. Bij 5100 neuronen wordt er dus een gigantische hoeveelheid tijd aan communicatie tussen de 50 CPU's gespendeerd. Figuur 4.4 is het resultaat voor dezelfde berekeningen bij 1 CPU, waarbij deze communicatie geen rol speelt - daarom is het verloop in deze figuur relatief gezien vrij egaal.

Het meer inhibitoir worden van het netwerk bij sterke vergroting is een algemeen resultaat dat automatisch plaatsvindt, onafhankelijk van de handmatige invoer van de excitatie en inhibitie. Eigenlijk is dat ongewenst, want men zou immers graag willen dat de netwerkeigenschappen ongeveer gelijk blijven bij verandering van de systeemgrootte, m.a.w.: in figuur 4.6 zou men een horizontale lijn willen zien. Hiertoe zouden er zaken moeten worden aangepast in de 'centrale' verbindingstabel uit hoofdstuk 3, als functie van de systeemgrootte. Dat is een bijzondere handeling; op deze manier worden modelparameters aangepast om naar de verwachte output toe te werken. De EEG's zouden op dat moment niet gezien worden als experimentele output, maar als afstellingsmiddel om *wel* de juiste resultaten te kunnen verkrijgen. Helaas gaat deze vorm van correctie niet op; er zitten natuurlijk veel meer afhankelijkheden in het model dan de zojuist besprokene. De belangrijkste hiervan is de gemiddelde reistijd van actiepotentialen tussen neuronen (zie figuur 4.7).

De output in figuren 4.5, 4.6 en 4.7 zijn allemaal precies zoals voorspeld aan de hand van de theorie: het aantal verbindingen neemt eerst (ongeveer) kwadratisch, daarna (ongeveer) lineair toe met het aantal cellen, de excitatie-/inhibitieratio gaat naar een limiet van ongeveer 8, en de gemiddelde reistijd van het netwerk wandelt naar ongeveer 7.17 ms.

Wat ondervinden we zoal in figuren 4.8-4.13? We zien zeer veel verschijnselen terug, waaronder alfa-ritmen ($\sim$13 Hz) in het geval van zeer lage excitatie en een groot netwerk. We kunnen in praktijk véél meer concluderen dan alleen de dingen die we in deze sectie zullen beschrijven. De belangrijkste observaties (in het kader van dit project) worden hieronder naar voren gehaald.

Wat betreft de EEG's zelf, valt allereerst op dat epileptiform gedrag weer 'terug kan vallen' naar chaotisch gedrag (zie 10.496 neuronen, E:I=2.0:1.0). Dat komt niet zo vaak voor, gezien het voor het systeem veel 'moeilijker' (en onlogischer) is om van geordend naar chaotisch te gaan dan andersom. Het kunstmatige EEG voor dit proces ziet er uit als dat van een epileptische aanval die stopt (wat ook op deze wijze vertaald mag worden, gezien het grote aantal neuronen waarbij dit gebeurt). Verder blijkt het mogelijk dat in slechts 1 EEG zowel het stiltegedrag als het

oscillerende groepsgedrag als chaotisch gedrag naar voren komt. 2624-0.5:2.0, 5248-2.0:10, 10496-0.5:2.0 en 10496-0.2:0.2 zijn 'ongebruikelijke' (en voorafgaand aan het onderzoek onverwachte) EEG's.

Het E-I-diagram voor 10.496 neuronen is een voorbeeld van het feit dat we in grote mate te maken hebben met multistabiliteit. Dat we bij het *verhogen* (verdubbelen zelfs) van de sterkte van de inhibitoire verbindingen van stiltegedrag naar fluctuaties gaan, is niet logisch. Het bewijst dat de hele rij voor I=2.0 evengoed uit stilte met pulsen had kunnen bestaan. Wellicht is de multistabiliteit zelfs zodanig groot dat de E-I-vlakken in deze paragraaf kant noch wal raken, omdat er *te* veel andere mogelijkheden zouden zijn. Echter, we zien duidelijk linksboven zeer stil gedrag en rechtsonder een zeer hoge activiteit. We plaatsen de (enigszins riskante) uitspraak dat dit voor elke beginconditie het geval zal zijn. Multistabiliteit speelt de zichtbare rol vooral in het pad tussen deze twee extreemste regimes.

Merk op dat de schaling van de verticale as van de EEG's in figuren 4.8-4.13 ongeveer lineair met het aantal neuronen heeft plaatsgevonden. Dat betekent dat het aantal neuronen dat meewerkt aan de groepsgewijze oscillaties ook lineair vergroot wordt, aangezien de extremen in de EEG's van ongeveer hetzelfde formaat blijven. Het is dus *niet* het geval dat bij een groter systeem steeds minder neuronen meewerken aan de groepsoscillaties (omdat de rest er 'te veel moeite voor zou moeten doen', lees: niet over de drempelwaarden heen zouden komen). De membraanstroom van de meeste neuronen trilt nog altijd netjes mee met het systeem. 'Bursting' blijft dus absoluut van toepassing.

Wat verder in het oog springt, is dat het netwerk van neuronen bij vergroting steeds minder de neiging lijkt te krijgen tot het groepsgewijs oscilleren. In het gehele E-I-vlak voor 5248 neuronen is géén epileptiform gedrag te herkennen (althans niet binnen de uitgevoerde 22 simulaties), hoewel er een zekere regelmaat is te constateren in het EEG van E:I=0.2:0.02. Bij 10.496 en 20.992 neuronen zijn weer wat vaker oscillaties te herkennen in het EEG, maar niet zoveel als bij de eerste EEG's. Dit is verklaarbaar: zoals in paragraaf 4.5 duidelijker is geworden (in praktijk), is het voor een groot netwerk erg moeilijk een groepsgewijs gamma-ritme te behouden. Voor een zeer groot aantal neuronen kunnen er wel groepsoscillaties zijn, maar in dat geval met een wat lager ritme (in het beta- of misschien alfa-domein)), en, vooral, met een grotere hoeveelheid aan chaos.

De verschijnselen die zichtbaar worden in figuren 4.15 en 4.16 zijn misschien wel de leukste en grappigste resultaten van het project. Zekerheid dat 18500 neuronen er al genoeg waren om dergelijk ruimtelijk gedrag in beeld te brengen hadden we immers niet.

In figuur 4.16 loopt blok 11 achter ten opzichte van blok 12 en ook van blok 44. Bij de puls die in beeld is gebracht is dat toevallig het geval, maar dit is in het algemeen niet in het gehele EEG zo. Bij een andere groepsgewijze puls in het zelfde EEG kan juist het gedrag in blok 11 dat in bijvoorbeeld blok 12 en in blok 44 'aansporen'. Ook de tijdsverschillen variëren nog al.

Kunnen we de geconstateerde tijdsverschillen koppelen aan modelparameters? Het ligt voor de hand dat de snelheid waarmee het gedrag (lees: een burst) zich in horizontale richting 'verplaatst' in de ruimte afhankelijk is van de voortbewegingssnelheid van actiepotentialen door een verbinding tussen twee neuronen. Vermenigvuldigen we het gevonden tijdsverschil in het rechtergedeelte van figuur 4.16, ongeveer 2.41 ms, met de AP-voortbewegingssnelheid van 0.08 m/s, dan verkrijgen we een afstand van ongeveer 192.8 μm. De horizontale gemiddelde afstand tussen een SPyr-cel in blok 11 en een SPyr-cel in blok 44 is bij benadering $\sim \sqrt{2 \cdot (\frac{3}{4} \cdot (424.9))^2} \sim 450\ \mu m > 192.8\ \mu m$ (eigenlijk is dit een volgend Line picking probleem!). De orde van grootte klopt dus wel redelijk, maar we vinden nog niet linea recta een terugkoppeling naar de voortbewegingssnelheid van 0.08 m/s. We moeten dan ook onthouden dat het tijdsverschil tussen twee pulsen niet voortdurend constant is, deze varieert flink. Het kiezen van de zestien blokjes is dan ook niet zo effectief voor te kleine systemen. Voor nu laten we het dan ook even bij de opmerking dat we inderdaad ruimtelijk gedrag zien, maar de analyse van dit gedrag laten we open voor later. Overigens werkt deze analyse natuurlijk veel effectiever zodra we een veel grotere ruimte en dus een groter model hebben.

Een logische verklaring voor het wat chaotische gedrag van het kunstmatige EEG voor een groot systeem in het geval van groepsoscillaties (zie figuur 4.17) zou zijn dat het het somsignaal van een veel groter aantal (SPyr-)neuronen betreft en dat ruimtelijk gedrag weer opspeelt. De kleine pulsen

zouden eventueel plaatsvinden in een ander bepaald gedeelte van de Neocortex dan die van de grote pulsen. Een neuron die meewerkt aan zo'n lage puls, werkt dan wellicht niet mee aan de hoge puls, hoewel ze wel een gevolg van elkaar zouden zijn. Deze speculatie is echter onjuist: zo'n beetje élk neuron (uitzonderingen nagelaten) geeft ongeveer hetzelfde verloop van de output in de tijd tijdens grootschalige groepsgewijze oscillaties: een grote puls, wat chaotische pulsjes (die ook meetellen in de frequentiebepaling..), een grote puls, enzovoorts (dit viel in feite ook al op in figuur 4.16). Bij het systeem van 10.000 neuronen, bijvoorbeeld, geeft het gros van de individuele SPyr-cellen zelf al een membraanstroom-verloop dat eruitziet als de betreffende groepsplot in figuur 4.17. Wat in dit geval dus opspeelt is dat een neuron zó veel verbindingen met andere neuronen heeft, sommige genererend/versterkend, andere weer remmend, dat het somsignaal van al deze input zodanig ingewikkeld wordt dat het steeds moeilijker is om tot een eenduidig groepsritme te komen. We houden het bij de conclusie dat gamma ritmen bij grote systemen inderdaad verlagen, maar niet op de manier zoals we misschien gehoopt hadden: een 'frequentie-versus-systeemgrootte-diagram' komt niet in beeld.

Conclusies

Het belangrijkste resultaat bij geleidelijke vergroting van het rekenprobleem is dat het netwerk eerst wat meer excitatoir wordt en daarna sterke inhibitoire (geremde) neigingen heeft. Dit resultaat blijkt in theorie vrij logisch te zijn: de verbindingsparameters die doorgaans gebruikt worden gelden in principe alléén voor het kleine netwerk en kunnen niet klakkeloos voor een veel groter netwerk worden toegepast. We hebben gezien dat het inhibitoire limietgedrag van het netwerk exact klopt met de voorspellingen naar aanleiding van de theorie. Ook de theoretische voorspellingen van de gemiddelde vertragingstijden voor verbindingen komen uit in de praktijk. Gamma ritmen dalen ietwat in frequentie (van zo'n 40 Hz naar ruim 30 Hz, ritmen komen dus terecht in het beta-domein), maar niet op de manier zoals we gehoopt hadden: een 'frequentie-versus-systeemgrootte-diagram' komt niet in beeld, omdat frequenties behoorlijk sterk variëren en de frequenties vooral dalen als gevolg van méér chaos in het groepsgedrag. Het valt dan ook in te denken dat het voor een heel groot netwerk moeilijker is om dit groepsgedrag 'af te stemmen' dan voor een klein netwerkje. Het totaal aantal verbindingen in het netwerk neemt voor grote systemen lineair toe met het aantal neuronen, omdat een asymptoot aanwezig is in verband met de maximum verbindingsafstand tussen twee neuronen.

Het meer inhibitoir worden van het netwerk bij vergroting is in feite *fout*. Als functioneel naar grotere netwerken wordt gekeken, zullen dan ook andere verbindingsparameters gebruikt moeten worden dan die in de tabel in hoofdstuk 3, want deze zijn bestemd voor een kleiner netwerk. Door middel van de geïntroduceerde theoretische route is het ook mogelijk om het netwerk zodanig aan te passen dat de totaalexcitatie en totaalinhibitie meer overeenkomen met die van het kleinere hersensysteem van rond de 656 à 1000 neuronen. Het is echter niet gegarandeerd dat daarmee het netwerk volledig 'gecorrigeerd' is - vermoedelijk niet zelfs, omdat er zich duidelijk meerdere afhankelijkheden bevinden in het model.

Er wordt ruimtelijk gedrag geconstateerd: bij grote systemen valt vaak een klein tijdsverschil tussen twee gedeelten die een zekere afstand van elkaar verwijderd zijn te bemerken (in het geval van duidelijk groepsgewijs gedrag). Deze vertraging is wel in de orde van grootte van deze afstand gedeeld door de voortbewegingssnelheid van AP's, maar is hier nog niet linea recta via een vastliggend verband aan terug te koppelen. *Verwacht* wordt dat vergroting van deze reistijd (onder meer) een preciezere analyse van het ruimtelijk gedrag mogelijk maakt, door de bredere scheidingen in tijd.

5.3 Terugkoppeling naar populatiemodellen

De hamvraag is of we kunnen stellen dat de netwerkeigenschappen die we gezien hebben te schalen zijn naar een populatiemodel. Oftewel: is de limiet van alle afzonderlijke differentiaalvergelijkingen in het model een voorspelbaar iets, dat altezamen slechts in een paar differentiaalvergelijkingen beschreven had kunnen worden, geldig voor een paar te definiëren neuronenpopulaties in het gehele

model? Natuurlijk is het hierbij belangrijk naar wát precies gekeken wordt. Gedetailleerde Single Neuron modellen richten zich veel meer op specifieke actiepotentiaalverplaatsingen dan populatiemodellen, die de totale activiteit binnen een groep neuronen modelleren. Populatiemodellen zouden dus gekoppeld zijn aan 'verplaatsing van activiteit' en niet aan 'verplaatsing van een AP'. Bepaalde praktische resultaten in dit verslag zijn in het geval van populatiemodellen misschien niet eens aan de orde, zoals bijvoorbeeld de gamma ritmen. Bij populatiemodellen is eigenlijk het voornaamste dat met vergelijkingen beschreven kan worden het aantal AP-vuringen (per tijdseenheid) binnen een populatie van een bepaald aantal neuronen, dus de activiteit in deze groep neuronen. Deze activiteit is dan ook de enige link naar Single Neuron modellen: in het geval van groepsgewijze oscillaties zien we bij een groot neuronensysteem een lichte vertraging tussen twee verschillende regio's binnen het stukje brein. Dat betekent dat er inderdaad sprake is van verplaatsing van activiteit in de ruimte. De activiteit die op het moment t in het ene gedeelte aanwezig is, is een moment later, tijdstip $t + \tau$, in het andere gedeelte te vinden. Dit is dezelfde (geschaalde) activiteit, die zich dus hooguit aan het verplaatsen is in de ruimte. Dit duidt op de mogelijkheid tot activiteitmodellering met een partiële differentiaalvergelijking, met plaats- en tijdsafhankelijkheid. In het geval van fluctuaties is de activiteit redelijk verdeeld over het stukje brein. In het geval van de 'stilte (en af en toe een groepsgewijze puls)'-toestand is de activiteit natuurlijk zeer laag.

In de vorige alinea hebben we vooral gesproken over activiteitbewegingen over en activiteitverschillen tussen locaties in het brein. Populatiemodellen zijn in het algemeen gebaseerd op neurontype: excitatoir of inhibitoir. Een neuropopulatiemodel bevat doorgaans minstens één excitatoire populatie en minstens één inhibitoire populatie, elk met een eigen (gekoppelde) differentiaalvergelijking die de activiteiten binnen de populaties beschrijft. Over deze vorm van modellering kunnen we helaas niet veel praktische uitspraken doen, want de groepsgedragmetingen (EEG's) die in hoofdstuk 4 beschreven staan laten het somsignaal van alléén de (excitatoire) SPyr-cellen zien. Wat er op dat moment nu eigenlijk met de interneuronen en met de DPyr-cellen gebeurt, wordt vooralsnog in het midden gelaten. Voor NeuroSim is dat interessant toekomstonderzoek, voor wat betreft de hyperkolom.

Teruggaand naar de basis mogen we concluderen dat, nu de link tussen Single Neuron modellen en populatiemodellen voor ons bekend is geworden, we het antwoord op de vraag of we veel van de resultaten in deze scriptie niet veel eenvoudiger hadden kunnen bereiken door middel van een veel sneller op te lossen populatiemodel, weten. Dit antwoord is *ja*, althans voor een nog in grote lijnen op te stellen groepsmodel. We zien duidelijke verlopen van neurale activiteit bij verandering van de systeemgrootte.

5.4 Toekomstvisie en aanbevelingen

Om door te stromen naar grotere (en dus ook meerzeggende) Single Neuron netwerken, zal men aandacht moeten schenken aan parallelrekening (in het geval van populatiemodellen is dit minder van toepassing). Voor een groot virtueel neuronensysteem, bijvoorbeeld een hyperkolom van zo'n 100.00 neuronen, valt implementatie van MPI sterk aan te raden.

Voor wat betreft de praktische output van NeuroSim voor grote schaal modellen zou bijvoorbeeld het ruimtelijk gedrag beter geanalyseerd kunnen worden door de delays in het netwerk aan te passen of voor een inhibitoir netwerk (in verband met de benodigde rekentijd) naar een groter systeem dan het huidig maximum van 18.500 neuronen te gaan.

Een actie die nu wat voor de hand ligt, is trachten de netwerkexcitatie te corrigeren voor grotere netwerken, d.w.z.: een geldige verbindingstabel proberen op te stellen als die in hoofdstuk 3, maar dan voor een systeem van 1000 neuronen, eentje voor 2000 neuronen, eentje voor 3000 neuronen, enzovoorts, zodanig dat de totaalnetwerkexcitatie bij netwerkvergroting ongeveer gelijk blijft aan die van het oorspronkelijke netwerk van 656 neuronen. Dit kan met behulp van de opgestelde theorie in hoofdstuk 3. Echter, dit zal helaas niet zo makkelijk zijn als het lijkt, allicht zelfs onmogelijk. Als we de gewichten van de verbindingen zodanig aanpassen dat de berekende excitatie-/inhibitieratio voor een neuron of 10.000 *wel* ongeveer gelijk is aan die bij 656 neuronen,

zien we tóch enkele verschillen in excitatie. Dit bevestigt ons vermoeden dat er meer gevarieerd zou moeten worden dan alleen de genoemde verbindingssterkten.

Ander toekomstwerk aan het model is natuurlijk het her en der verbeteren van de techniek in en de eigenschappen & geavanceerdheid van NeuroSim, wat momenteel dan ook gebeurt. Tijdens de uitvoering van het nu beschreven onderzoek was NeuroSim dan ook nog volop in ontwikkeling. In deze periode zijn gap-junctions toegevoegd, is een analyse gedaan op de delaytimes met betrekking tot hysterese, en is gedacht aan een herstart en alternatieve integratiemethodieken. Een voorbeeld van een wat diepere eventuele toekomstontwikkeling: in werkelijkheid verschijnen en verdwijnen de biologische neuronale verbindingen in het menselijk brein voortdurend. Een verschijnsel wat bij lange na nog niet is verwerkt in NeuroSim; zover zijn de vorderingen nog niet. Misschien is dat voor wat betreft epilepsie dan ook minder belangrijk.

Het meeste toekomstonderzoek zal, als we terugkijken naar breinmodellering als geheel, in het gebied van populatiemodellen liggen, want het is aantrekkelijk om een groot model met slechts een op de vingers van één hand telbaar aantal vergelijkingen te modelleren...

Bibliografie

[1] M.J.A.M. van Putten. Essentials of Neurophysiology (Basic Concepts and Clinical Applications for Scientists and Engineers) *Springer, 2009*

[2] Canadian Institutes of Health Research; 'The Brain from Top to Bottom' (http://thebrain.mcgill.ca/flash/i/i_02/i_02_cl/i_02_cl_vis/i_02_cl_vis.html) (gezien op 18-05-2010)

[3] R.W. Williams, K. Herrup. The Control of Neuron Number *Annual review of Neuroscience, 11: 423-453, 1988*

[4] W. van Drongelen, H. Koch, F.P. Elsen, H.C. Lee, A. Mrejeru, E. Doren, C.J. Marcuccilli, M. Hereld, R.L. Stevens, J. Ramirez. The Role of Persistent Sodium Current in Bursting Activity of Mouse Neocortical Networks (Bursting Activity in Neocortical Networks) *Journal of Neurophysiology, 96: 25642577, 2006*

[5] J.M. Bower, D. Beeman, M. Nelson, J. Rinzel, I. Segev. Tutorial Genesis chapters 2 (Compartmental Modeling), 4 (The Hodgkin-Huxley Model), 6 (Temporal Interactions Between Postsynaptic Potentials)

[6] H. Markram. The Blue Brain Project *Nature Reviews Neuroscience, 7(2): 153-159, 2006*

[7] W. van Drongelen, H.C. Lee, M. Hereld, D. Jones, M. Cohoon, F. Elsen, M.E. Papka, R.L. Stevens. Simulation of Neocortical Epileptiform Activity Using Parallel Computing *Neurocomputing, 58: 1203-1209, 2004*

[8] M. Hereld, R.L. Stevens, J. Teller, W. van Drongelen, H. Lee. Large neural simulations on large parallel computers *International Journal of Bioelectromagnetism, 7: 44-46, 2005*

[9] D. Beeman, University of Colorado, Boulder, USA, 2005; 'GENESIS Modeling Tutorial Electrical and Computer Engineering Department' (http://www.brains-minds-media.org/archive/220) (gezien op 18-05-2010)

[10] D.Q. Nykamp, D. Tranchina. A Population Density Approach That Facilitates Large-Scale Modeling of Neural Networks: Analysis and an Application to Orientation Tuning *Journal of Computational Neuroscience 8: 19-50, 2000*

[11] A.V. Rangan, D. Cai. Fast Numerical Methods for Simulating Large-scale Integrate-and-fire Neuronal Methods *Journal of Computational Neuroscience 22: 81-100, 2007*

[12] C.M. Gray. Synchronous Oscillations in Neuronal Systems: Mechanisms and Functions *Journal of Computational Neuroscience 1: 11-38, 1994*

[13] E. Rodriguez, N. George, J. Lachaux, J. Martinerie, B. Renault, F. J. Varela. Perception's Shadow: Long-distance Synchronization of Human Brain Activity *(letters to) Nature, 397: 430-433, 1999*

[14] R. Llinás, U. Ribari, D. Jeanmonod, R. Cancro, E. Kronberg, J. Schulman, M. Zonenshayn, M. Magnin, A. Morel, M. Siegmund. Thalamocortical dysrhythmia I. Functional and imaging aspects *Thalamus & Related Systems, vol. 1, nr. 3: 237-244, 2001*

[15] A. Garenne, J. Henry, C.O. Tarniceru. Analysis of Synchronization in a Neural Population by a Population Density Approach *Mathematical Modelling of Natural Phenomena, vol. 5, nr. 2: 5-25, 2010*

[16] Farlex, The Free Dictionary; 'Beta Rhythm' (http://medical-dictionary.thefreedictionary.com/beta+rhythm) (gezien op 18-05-2010)

[17] M.A. Whittington, R.D. Traub, N. Kopell, B. Ermentrout, E.H. Buhl. Inhibition-based Rhythms: Experimental and Mathematical Observations on Network Dynamics *International Journal of Psychophysiology, 38: 315-336, 2000*

[18] N. Kopell, G.B. Ermentrout, M.A. Whittington, R.D. Traub. Gamma Rhythms and Beta Rhythms have Different Synchronization Properties *PNAS, vol. 97, nr 4: 1867-1872, 2000*

[19] R.D. Traub, N. Spruston, I. Soltesz, A. Konnerth, M.A. Whittington, J.G.R. Jefferys. Gamma-frequency Oscillations: a Neuronal Population Phenomenon, Regulated by Synaptic and Intrinsic Cellular Processes and Inducing Synaptic Plasticity *Progress in Neurobiology, 55: 563-575, 1998*

[20] D. Volk. Population Oscillations in a Discrete Model of Neural Networks of the Brain *BioSystems 63: 35-41, 2001*

[21] D.A. Prince, K. Jacobs. Inhibitory function in two models of chronic epileptogenesis *Epilepsy Research 32: 83-92, 1998*

[22] H.R. Wilson, J.D. Cowan. Excitatory and Inhibitory Interactions in Localized Populations of Model Neurons *Biophysical Journal, 12(1): 1-24, 1972*

[23] L.H. Zetterberg, L. Kristianssion, K. Mossberg. Performance of a Model for a Local Neuron Population *Biological Cybernetics, 31: 15-26, 1978*

[24] S. Verduzco-Flores, B. Ermentrout, M. Bodner. From Working Memory to Epilepsy: Dynamics of Facilitation and Inhibition in a Cortical Network *Chaos: An Interdisciplinary Journal of Nonlinear Science, 19: 15115 (1), 2009*

[25] U. Ernst, A. Etzold, M.H. Herzog, C.W. Eurich. Dynamics of Neuronal Populations Modeled by a Wilson-Cowan System Account for the Transient Visibility of Masked Stimuli *Neurocomputing, vol. 52, nr. 54: 747-753, 2003*

[26] A.C. Marreiros, J. Daunizeau, S.J. Kiebel, K.J. Friston. Population Dynamics: Variance and the Sigmoid Activation Function *NeuroImage, 42: 147-157, 2008*

[27] A.C. Marreiros, J. Kiebel, K.J. Friston. A Dynamic Causal Model Study of Neuronal Population Dynamics *NeuroImage 51: 91-101, 2010*

[28] S. Coombes. Large-scale Neural Dynamics: Simple and Complex *NeuroImage, article in press, 9 pages. doi: 10.1016/j.neuroimage.2010.01.045*

[29] D.T.J. Liley, P.J. Cadusch, M.P. Dafilis. A Spatially Continuous Mean Field Theory of Electrocortical Activity *Network-Computation in Neural Systems 13: 67-113, 2002*

[30] F. Wendling, F. Bartolomei, J. J. Bellanger, P. Chauvel. Epileptic Fast Activity Can Be Explained by a Model of Impaired GABAergic Dendritic Inhibition *European Journal of Neuroscience, vol. 15, nr. 9: 1499-1508, 2002*

[31] M. Ursino, F. Cona, M. Zavaglia. The Generation of Rhythms within a Cortical Region: Analysis of a Neural Mass Model *NeuroImage, article in press, 15 pages. doi: 10.1016/j.neuroimage.2009.12.084*

[32] F.C. Hoppensteadt, E.M. Izhikevich. Weakly Connected Neural Networks *Springer, Applied Mathematical Sciences, vol. 126, 1997*

[33] E.M. Izhikevich. Dynamical Systems in Neuroscience: The Geometry of Excitability and Bursting *The MIT Press, 2007*

[34] E.M. Izhikevich. Neural Excitability, Spiking and Bursting *International Journal of Bifurcation and Chaos, vol. 10, nr. 6: 1171-1266, 2000*

[35] A. Hutt. Oscillatory Activity in Excitable Neural Systems *Contemporary Physics, vol. 51, nr. 1: 3-16, 2009 (eerste editie)*

[36] D. Wanrooy, Software Innovators; 'De Wet van Amdahl' (http://www.software-innovators.nl/2009/03/23/de-wet-van-amdahl) (gezien op 18-05-2010)

[37] B. Chapman, G. Jost, R. van der Pas, D. J. Kuck. Using OpenMP (Portable Shared Memory Parallel Programming) *The MIT Press, 2008*

[38] B. Barney, Lawrence Livermore National Laboratory; 'Message Passing Interface' (https://computing.llnl.gov/tutorials/mpi) (gezien op 18-05-2010)

Bijlage A

Programmastructuur NeuroSim

Ontwerp NeuroSim: S. Visser, gebaseerd op neuronaal simulatieprogramma GENESIS (ontworpen door Van Drongelen e.a.).

A.1 Pointers

Een Pointer betekent een 'wijzer' naar een zekere geheugenplek in de computer. Wanneer aan de variabele *andy* de waarde 25 wordt toegekend, wordt op een zekere plek in de computer opgeslagen dat $andy = 25$. Deze plek wordt het adres genoemd van de variabele *andy*. Dit adres is bekend voor de computer (dit wordt de 'referentie' genoemd), dus zou ook opgevraagd kunnen worden. We kunnen bijvoorbeeld de variabele *ted* het adres van *andy* als waarde geven via het commando $ted = \&andy$ (in C-taal). Ted slaat dan het adres op van *andy*, wat bij de meeste huidige computers één of ander hexadecimaal getal zal zijn. Zie figuur A.1.

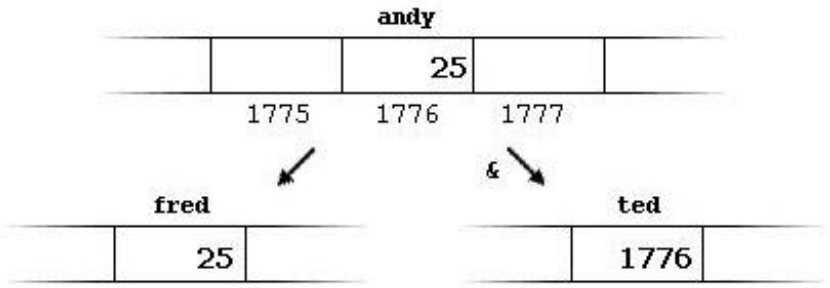
Wanneer variabelen binnen een functie gebruikt worden, worden deze 'weggegooid' zodra de functie doorlopen is en kunnen ze niet meer gebruikt worden binnen het programmagedeelte buiten de functie. Natuurlijk kunnen er ook globale variabelen gedefinieerd worden, die blijven bestaan zolang het programma actief is, maar ook door elke functie kunnen worden veranderd ..wat tot (ongemerkte!) fouten zou kunnen leiden. Door het gebruik van Pointers kan de programmeur ervoor zorgen dat geheugen (en dus de informatie in dat geheugen) beschikbaar blijft totdat deze het zelf weer vrijgeeft. Gebruik van Pointers is dus eigenlijk nooit een vereiste, maar een handigheid. Pointers worden echter in de meeste professionele codes gebruikt!

A.2 Linked Lists

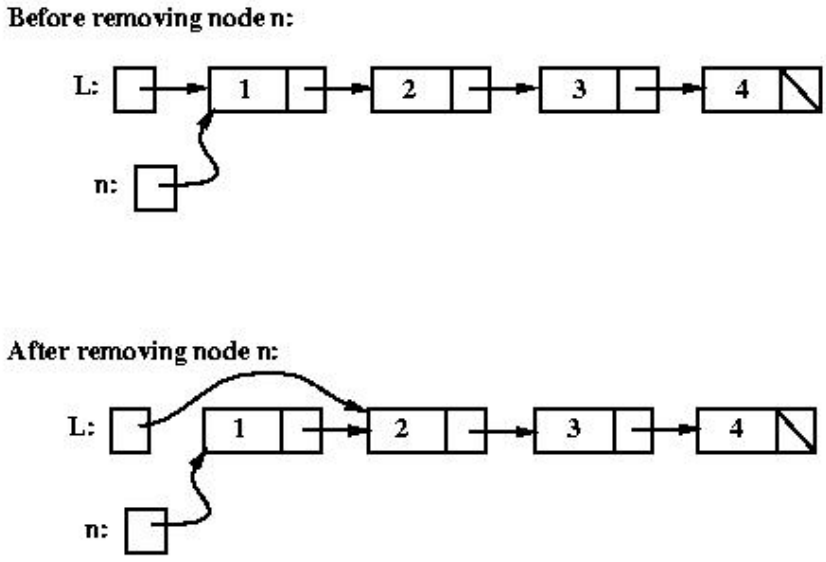
In C-taal kost het veel werk om op een willekeurige plek in een array een element toe te voegen of te verwijderen. Er moet dan eigenlijk een geheel nieuwe array gemaakt worden, waarin de elementen opnieuw worden opgeslagen. Bovendien moet van ieder array de lengte vooraf gedefinieerd worden om geheugen te reserveren - ook wanneer niet bekend is hoe lang het array zal zijn/worden. Soms werkt dit nogal omslachtig en is het makkelijker om de zgn. Linked List te gebruiken. Een Linked List bekijkt een array per element en slaat steeds de waarde van het betreffende element en het adres van het volgende (Single Linked List) en eventueel het adres van het vorige (Double Linked List) element op. Zodra een Pointer op (het adres van) een array-element gericht staat, ziet deze dus automatisch wat (het adres van) het volgende element is. Element n verwijderen uit een array gaat nu heel makkelijk door de rechterbuur van element $n-1$ te veranderen in het adres van element $n+1$. Zie figuur A.2. In C-taal kan vervolgens met behulp van het commando 'delete' eventueel het geheugen dat gebruikt werd voor element n verwijderd worden. Op dezelfde manier kan ook eenvoudigerwijs een element toegevoegd worden op een willekeurige plaats in een array.

Merk op dat het gebruik van de Linked List niet altijd in het voordeel werkt. Wanneer vooraf duidelijk vast staat hoe groot het array zal worden en er zullen naderhand geen elementen op

```
1 andy = 25;
2 fred = andy;
3 ted = &andy;
```



Figuur A.1: Principe van de Pointer: opslaan van het geheugenadres van een gegeven in plaats van het gegeven zelf.
(http://www.cplusplus.com/doc/tutorial/pointers)



Figuur A.2: Principe van de Linked List: opslaan per element van een array in plaats van een gehele array. Uitbeelding van verwijdering van een element (hier het eerste) uit de gelinkte lijst door verandering van de gegevens over het buurelement. Boven: vóór verwijdering; Onder: ná verwijdering van element 1 uit de lijst.
(http://www.it-c.dk/research/theory/SearchEngineF2004/mirrors/wiscdocs/notes/LINKED-LIST-FIGURES/removeFirst.gif)

willekeurige plaatsen worden toegevoegd, is het niet aan te raden een Linked List systeem te gebruiken. Nadeel van de Linked List is immers dat wanneer de Pointer ongunstig staat, zeg op element 1, en er moet iets gebeuren met/op element 100.000.000, de Pointer zich dan 100.000.000-1 adressen moet verplaatsen, wat vermoedelijk alsnog meer tijd zal kosten dan de gebruikelijke methode.

A.3 Klassen

Een 'class' (klasse) is een groep objecten van dezelfde 'soort'. Men zou 'integers' en 'doubles' als twee (verschillende) klassen kunnen zien, want voor iedere integer gelden bepaalde eigenschappen, die weer anders (kunnen) zijn dan voor een double. Iedere double heeft wel weer dezelfde eigenschappen als een andere double.

Voor een klasse definieert men bepaalde eigenschappen (al dan niet handmatig - voor integers gelden natuurlijk automatisch zekere eigenschappen) die gelden als sjabloon voor alle objecten binnen deze klasse. Voor ieder object in de betreffende klasse gelden dezelfde eigenschappen. We halen bijvoorbeeld de klasse 'APFiringCell' aan, waarin de eigenschappen voor alle zenuwcellen (bijvoorbeeld AP-threshold en AP-generatieduur), algemeen gezien, opgeslagen staan. Men zou hierbij zes 'subklassen' kunnen definiëren: de SPyr-, DPyr-, Bask1-, Bask2-, Bask3- en de Chand-klasse. Voor iedere celsoort gelden dan de eigenschappen van het bijbehorende sjabloon; hierin bevindt zich informatie over bijvoorbeeld de hoeveelheid compartimenten, het type (klasse!) van deze compartimenten en het type (klasse!) verbindingen naar andere neuronen. Andere voorbeelden van klassen zijn de compartimentklassen, de kanaalklassen en de verbindingsklassen. Programmeren in klassen heet 'objectgeoriënteerd programmeren'. Objectgebaseerde talen definiëren programma's geheel in termen van objectklassen. Objecten ontstaan als kopieën van 'basisobjecten' en gebruiken uitbreiding (subklassen) om informatie toe te voegen of van definitie te veranderen.

A.4 De code

Nu volgt een zeer beknopte uitleg over de code in C++-programmeertaal van 'NetworkSimulator.cpp' binnen NeuroSim. Het betreft *de originele implementatie*, dus de versie voorafgaand aan de implementaties t.b.v. de resultaten in deze scriptie. NetworkSimulator.cpp is de hoofdcode: runnen van deze C++-file betekent runnen van de gehele simulatie. Voor compleet begrip en eventuele creatie van eigen implementaties zal deze beknopte uitleg bij lange na niet genoeg zijn. Doel van deze bijlage is dan ook hoogstens verduidelijking van de opbouw binnen NeuroSim.

Bovenin de code wordt, behalve de gebruikelijke 'include'-basiscommando's (*iostream*, *cmath*, enz.), het commando *include "LinkedLists.cpp"* uitgevoerd. LinkedLists.cpp bevat de toevoegingsfuncties en verwijderingsfuncties voor elementen in het sjabloon van de Linked List. Ook wordt er een functie toegevoegd die een genormeerd random nummer genereert - later in de code wordt deze functie toegepast. Let op: het programma initialiseert (naar keuze) de 'random-seeds'. Dat betekent dat de random getallen die gebruikt worden in de simulatie wel 'random' zijn, maar bij iedere gelijke simulatie (met dezelfde random-seeds) hetzelfde. Dit heet 'pseudo-random'. Simulatieresultaten zijn dus reproduceerbaar!

Elke klasse binnen de netwerksimulator heeft haar eigen C++-implementatie en is verdeeld in een cpp-bestand met Definitions en een cpp-bestand met bijbehorende Functions. Onder Definitions staat aangegeven wat de klasse is van een zeker object of van een zekere functie; de bewerkingen zelf staan onder Functions. In 'chronologische' volgorde worden aan NetworkSimulator.cpp meegegeven (via '#include') de cpp's SimConstants (simulatieconstanten: simulatietijd, hoeveelheid cellen van iedere soort, resolutie, enz.), PhysConstants (fysiologische constanten: geleidingssnelheden, kansdichtheden, enz.), Declarations (basisdeclaraties), CommunicationDefinitions, ChannelDefinitions, CompartmentDefinitions, APGeneratorDefinitions, ChannelFunctions, CompartmentFunctions, APGeneratorFunctions, NetworkSetupDefinitions, NetworkSetupFunctions, NetworkGenerator (die van NetworkSetupDefinitions en -Functions gebruik maakt om een

netwerk te genereren: cellocaties en -verbindingen worden gegenereerd), SPyrDefinitions, DPyr-Definitions, Bask1Definitions, Bask2Definitions, Bask3Definitions en ChandDefinitions.

De volgende stap, en de laatste vóór het 'main'-gedeelte van NetworkSimulator, is het reserveren van geheugen voor de verbindingstabellen. Er is een TableType (int), een TableSelfConnect (double) (niet van toepassing - we beschouwen geen celverbindingen met zichzelf), een TableDistance (double), een TableProbability (double), een TableDistanceDecay (double), een TableVelocity (double), een TableMinWeight (double), een TableMaxWeight (double) en een TableWeightDecay (double). Al deze tabellen/tabelnamen spreken voor zich en zijn gebaseerd op de (kolommen uit de) tabel in hoofdstuk 3. Dit is het codegedeelte dat het meest gebaseerd is op het model in GENESIS.

Het eerste wat na 'int main()' gebeurt is het gebruiksvriendelijk laten weergeven van de simulatiestartdatum en -tijd, het inlezen van de tabellen (apart onder een directory Data toegevoegd in .dat-bestanden) en het interpreteren van twee externe variabelen voor de totale excitatie en inhibitie van het netwerk vanuit een bestandje Params.in. De tabellen MinWeight en MaxWeight worden hiermee aangepast, namelijk genormeerd met de opgegeven schalingsconstanten vanuit Params.in.

Het netwerk wordt gegenereerd en de code produceert bestandjes *CellPositions.nwk*, *ConnectionCount.nwk*(, *IncomingConnections.nwk*) en *ConnectionList.nwk*, met daarin de gegevens over celposities (x,y en z-coördinaten), hoeveelheid verbindingen per cel en specifieke informatie (naar welke cel, naar welk kanaal, de vertraging, het gewicht (in kansverdeling)) per verbinding. Vervolgens wordt per cel de specifieke informatie (SPyr, DPyr, ...) toegevoegd in een CellPointerList en worden de verbindingen toegevoegd.

Door (bijvoorbeeld) de eerste paar (in dit geval 50) milliseconden een stroompje (vanaf een denkbeeldige externe elektrode) toe te voegen aan de eerste paar SPyr-cellen, krijgt het systeem een 'schop' en zal, afhankelijk van de hoeveelheid excitatie en inhibitie, het aantal cellen, enzovoorts, het zenuwsysteem bepaald gedrag gaan vertonen. Na een bepaald aantal milliseconden wordt deze stroom weer uitgeschakeld, omdat naar de *natuurlijke* frequenties binnen de hersenen gekeken zou moeten worden, dat wil zeggen naar het limietgedrag en niet naar het opstartgedrag. Ook zouden eventueel een aantal extra actiepotentialen op verscheidene cellen kunnen worden ingelezen/toegevoegd vanuit een externe file APInput.in.

Na enkele initialisaties van (namen van) de outputbestanden *Vm.dat* en *Im.dat* (bestanden met de membraanpotentialen per cel en de membraanstromen van de SPyr-cellen, respectievelijk, per tien tijdstappen) kan de simulatietijd-loop beginnen. Per tijdstap van 0.01 milliseconde (ook gebaseerd op GENESIS) wordt een 'Step' en een 'Update' over iedere cel en daarmee ieder compartiment en kanaal binnen dit compartiment uitgevoerd. De 'Step' houdt in: het uitvoeren van de integratie, gebaseerd op de geformuleerde differentiaalvergelijkingen per compartiment. Dit gebeurt vooralsnog met een expliciete Euler integratie. Bij de 'Update' wordt de informatie van iedere cel daadwerkelijk geupdated (lees: weggeschreven naar de kanaalvariabelen). Per tiende deel van de simulatie verschijnt een extra puntje in beeld, zodat de uitvoerder van NetworkSimulator enigszins kan zien hoever de simulatie is. Per (simulatie)seconde wordt overgestapt op een volgend outputbestand voor V_m en I_m, omdat de outputbestanden anders zeer groot zullen worden (bij een groot aantal cellen). Na het doorlopen van de simulatietijd-loop is de simulatie afgerond en wordt de eindtijd van de simulatie gereproduceerd.

Het parallelliseren van de code met behulp van OpenMP gebeurt eenvoudigerwijs door boven in NetworkSimulator.cpp *#include ⟨omp.h⟩* te plaatsen (zodat de compiler weet dat er OpenMP gebruikt wordt (beter gezegd: zou kunnen worden)) en boven de Step-regel, dat een for-loop is over iedere cel in het systeem, de regel '#pragma omp parallel for' toe te voegen. De integratiestap wordt dan parallel uitgevoerd. Verscheidene CPU's (extern op te geven) nemen een zeker gedeelte van de zenuwcellen voor hun rekening. Bij de Update-regel moet dit parallelliseren *niet* plaatsvinden. Er zal anders sprake zijn van zgn. 'data-racing'. De update-momenten voor de cellen zijn dan niet netjes uitgelijnd, waardoor er zich ongewenste random verschijnselen in het hersensimulatieprogramma zullen bevinden. Zie voor meer informatie over de toepassing van OpenMP bijlage B en natuurlijk [37] voor een algemene handleiding.

Voor de daadwerkelijke resultaten uit hoofdstuk 4 is de output voor V_m en I_m aangepast: deze

bevatten niet meer de membraanpotentialen en -stromen per tien tijdstappen. Voor een zodanig grote hoeveelheid cellen wordt de hoeveelheid outputdata dan veel te groot - en overbodig, omdat naar populaties gekeken wordt. Voor een grote hoeveelheid cellen worden uitsluitend de momenten (tijdstippen) van AP-generaties per cel (uitsluitend de SPyr-cellen) opgeslagen en het kunstmatige EEG in een speciaal bestand geproduceerd. Aan de hand van de bestanden met AP's zouden alsnog gamma ritmen beschouwd kunnen worden. Ook (gewogen) totaalsignalen kunnen deels afgeleid worden, maar via een benaderingsmethode. Dergelijke data-uitlezing gebeurt met Matlab, wat ook in de niet opgeschaalde code het geval is.

Bijlage B

Parallelcalculatie

NeuroSim modelleert (initieel) het brein met behulp van één computer, oftewel met één CPU (Central Processing Unit). Er wordt slechts één rekenkundige bewerking tegelijk uitgevoerd. Het is natuurlijk al een prestatie dat dergelijke modellering op een computer mogelijk is, maar het valt in te denken dat bepaalde simulatieprogramma's significant versneld kunnen worden zodra er meerdere bewerkingen tegelijk uitgevoerd kunnen worden, oftewel zodra er parallel gerekend kan worden op meerdere CPU's tegelijkertijd. Parallelcalculatie is een actuele ontwikkeling die al veelvuldig wordt toegepast bij grootschalige simulaties.

B.1 Parallelrekening en de Wet van Amdahl

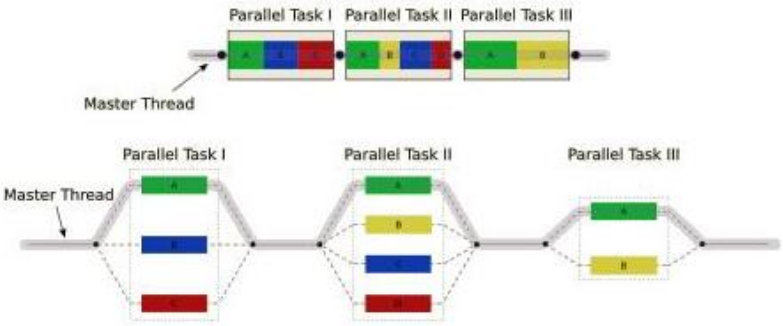
Een belangrijke beperking van een CPU is dat deze slechts één rekenkundige bewerking tegelijk kan uitvoeren. Een CPU voert bewerkingen ná elkaar uit. Dit gebeurt zeer snel, wat van vroeger uit het ultieme voordeel van een computer is. Een zeer grootschalige simulatie zal natuurlijk ook een grote hoeveelheid rekentijd in beslag nemen (doch in het algemeen uiteraard nog altijd vele malen minder dan wanneer de mens zelf dergelijke berekeningen moest uitvoeren..).

Wanneer bij wijze van spreken 32 maal het sommetje $16 - 3$ berekend moet worden, dan gaat dit in theorie 16 maal sneller met 16 processoren dan met 1 processor (dra zal blijken dat dit in de praktijk niet het geval is. We nemen tenminste aan dat er over de output van deze sommetjes gecommuniceerd zal moeten worden. De complete sessie bewerkingen zal wellicht wel sneller verlopen, maar helaas niet de theoretische 16 maal).

Niet iedere bewerking kan zomaar geparallelliseerd worden. De bewerkingen moeten van elkaar onafhankelijk zijn; de ene calculatie mag de ander niet beïnvloeden. Tijdstappen, bijvoorbeeld, kunnen in het algemeen niet parallel plaatsvinden. Wat *laat* gebeurt en afhankelijk is van *eerdere* gebeurtenissen, kan niet tegelijkertijd met deze eerdere gebeurtenissen berekend worden. Een voorbeeld van een zeer eenvoudige handeling die *wel* parallel kan verlopen is het vullen van een lange array met (van elkaar onafhankelijke) getallen. Neem bijvoorbeeld een array met $N = 1600$ elementen, die in z'n geheel gevuld wordt met enen. Wanneer dit gebeurt met, bijvoorbeeld, 20 CPU's, dan zou CPU 1 element 0 t/m 79 voor zijn rekening kunnen nemen, CPU 2 element 80 t/m 159, ..., en CPU 20 element 1520 t/m 1599.

Het parallelliseren van een programmacode vindt in het algemeen plaats door de meest tijdrovende onafhankelijke bewerkingen op meerdere CPU's te laten uitvoeren. Hiertussendoor kan gecommuniceerd worden. Een dergelijke deelbewerking (per CPU) heet een 'thread'. Een geparallelliseerd programma bestaat uit een 'master-thread', de hoofd-thread, die serieel (als in een gebruikelijke code, zeg 'chronologisch') verloopt en een hoeveelheid 'slave-threads' of 'worker-threads', die parallel hun handelingen uitvoeren. Tussen de parallelle blokjes door vindt communicatie plaats in naam van de master-thread. Zie figuur B.1.

Wat is nu de snelheid van een programmacode als functie van de hoeveelheid processoren? Lo-



Figuur B.1: Master-slave constructie voor een geparallelliseerde programmacode.

gisch zou klinken, zoals eerder aangehaald, dat bij een verdubbeling van de hoeveelheid CPU's de benodigde kloktijd halveert. Dit zou wáár zijn op het moment dat de gehele code parallelliseerbaar is. In de praktijk komt dit helaas weinig (of niet - er zal altijd sprake blijven van de nodige communicatietijd) voor. De rekentijd (voor een voortdurend identiek probleem) tegen de hoeveelheid processoren gedraagt zich volgens de zgn. Wet van Amdahl [36], die als volgt wordt afgeleid:

Stel: P is de fractie van de totale hoeveelheid werk die parallel uitgevoerd kan worden tegen de totale hoeveelheid werk W van een programma:

$$P = \frac{werk_{parallel}}{werk_{totaal}} = \frac{W_{parallel}}{W} \tag{B.1}$$

In het meest ideale (misschien onmogelijke) geval is P dus 1. Noem N het aantal beschikbare processoren (een integer dus), T de benodigde tijd wanneer het programma niet parallel zou verlopen, en T' de benodigde kloktijd wanneer het programma geparallelliseerd wordt met gebruik van alle N CPU's. Er geldt in het algemeen dat de seriële rekentijd T de complete hoeveelheid werk W is, gedeeld door de (reken)snelheid van de CPU's, die we v noemen:

$$T = \frac{W}{v} \tag{B.2}$$

Per definitie van P geldt dat

$$W_{parallel} = PW \tag{B.3}$$

en, omdat $W = W_{parallel} + W_{serieel}$, dat:

$$W_{serieel} = (1 - P)W \tag{B.4}$$

Wanneer we het bovenstaande substitueren in T', vinden we:

$$T' = T_{serieel} + T_{parallel} = \frac{W_{serieel}}{v} + \frac{W_{parallel}}{Nv} \tag{B.5}$$

$T_{serieel}$ en $T_{parallel}$ zijn hier de benodigde seriële en parallelle, respectievelijk, rekentijden behorend bij het parallelle verloop van het programma. Het parallelle gedeelte wordt *wel* lineair versneld ten opzichte van de hoeveelheid processoren. Bovenstaande is gelijk aan:

$$T' = \frac{(1 - P)W}{v} + \frac{PW}{Nv} \tag{B.6}$$

en daarmee verkrijgen we:

$$\frac{T}{T'} = \frac{\frac{W}{v}}{\frac{(1-P)W}{v} + \frac{PW}{Nv}} = \frac{1}{(1-P) + \frac{P}{N}} \tag{B.7}$$

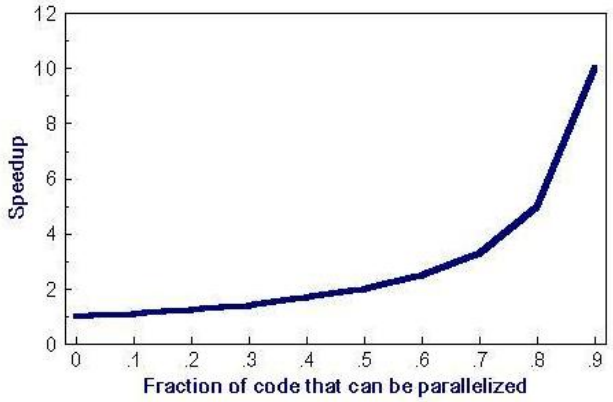
B.7 is de wet van Amdahl (zie figuren B.2 en B.3). Volgens de wet van Amdahl vlakt de lijn in een diagram van de rekentijd tegen het aantal CPU's af naar een zekere waarde. Parallelliseren van 1 naar 2 CPU's, of van 2 naar 4, lijkt vaak nog bij benadering lineair te gaan (afhankelijk van de probleemgrootte), maar bij een grote waarde voor N worden de voordelen steeds kleiner. Dit heeft tevens te maken met de communicatie tussen de CPU's. Als het aantal CPU's groot is, dan heeft de code een boel tijd nodig om de resultaten vanuit de verschillende slave-threads naar de master-thread (en/of eventueel andersom) te communiceren. De code wordt dan simpelweg een 'babbelbox'. Wat blijkt uit de praktijk is dat bij opschaling van het rekenkundig probleem (dus bij het vergroten van het probleem) het domein met een lineair verband tussen de rekentijd en het aantal CPU's vergroot wordt; de communicatie-invloed neemt dan immers af.

B.2 OpenMP

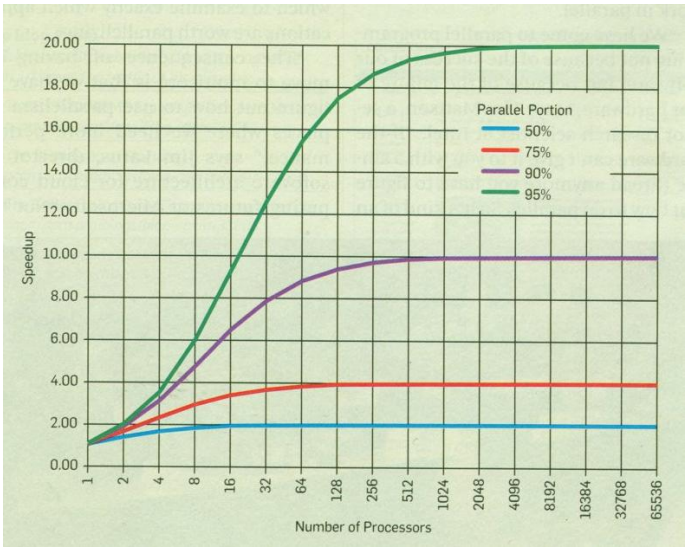
In de vorige paragraaf staat enigszins logische theorie over de versnelling ten gevolge van parallelrekening. Maar hoe geeft men nu in de praktijk aan een programmacode mee welke 'stukjes' van deze code parallel kunnen/mogen verlopen en hoe de code het rekenwerk het meest effectief zou moeten verdelen over een hoeveelheid CPU's? Hiervoor zijn een paar 'interfaces' ontworpen, waarvan de belangrijkste twee MPI (Message Passing Interface) en OpenMP zijn. Beide zijn (voor wat betreft functionering) programmeertaal-onafhankelijk. MPI [38] is het dominante model dat gebruikt wordt in hedendaagse high-performance computersimulaties en is bedoeld om de communicatie tussen de processen te synchroniseren. MPI kan gebaseerd zijn op 'Point-to-point basics' (het ene proces zendt een bericht, variabele of resultaat naar het andere proces) of 'Collective basics' (communicaties die álle processen binnen een zekere groep aan gaan). Bij gebruik van MPI wordt in principe de gehele seriële code opnieuw geschreven met extra routines betreffende de initialisaties, topologiën en functionaliteiten binnen en tussen de parallelle processen. Dit is, vergeleken met OpenMP, een vrij ingewikkelde en gespecialiseerde, maar zeer effectieve, methode. OpenMP ('a portable programming interface for shared memory parallel computers') is eenvoudiger toe te passen (de 'Open' in OpenMP staat voor het vrije gebruik van parallelrekening onder niet-experts): per programmacode wordt (vaak in slechts één enkele programmeerregel!) aangegeven of het volgende programmastukje al dan niet parallel mag/moet verlopen. Er wordt een 'pragma' (noem het 'richtlijn') meegegeven. Zie figuur B.4. Door toevoeging van slechts één regel aan de code wordt in deze figuur de for-loop geparallelliseerd. Voor programmeertechnische details over OpenMP wordt verwezen naar [37].

In het algemeen is uit talloze toepassingen in het verleden gebleken dat OpenMP ten opzichte van MPI zeer eenvoudig te begrijpen, maar ook te implementeren is in een code. De programmeur bekijkt eerst per deelfunctie van de hoofdcode hoeveel rekentijd het seriële programma voor deze deelfunctie kwijt is en parallellisseert de meest tijdrovende stukjes naar aanleiding van deze resultaten door toevoeging van enkele regels OpenMP-code. Wat er dus *niet* gebeurt, is dat iedere 'do-loop' in een code zomaar geparallelliseerd wordt. Dit zal de code (bij wijze van voorspelling) significant minder effectief versnellen, misschien zelfs juist vertragen, omdat communicatie dan wederom een grote rol zal gaan spelen. Toepassing van MPI is in principe een effectiever, maar veel ingewikkelder proces, omdat de gehele code dan herschreven dient te worden in MPI-routines.

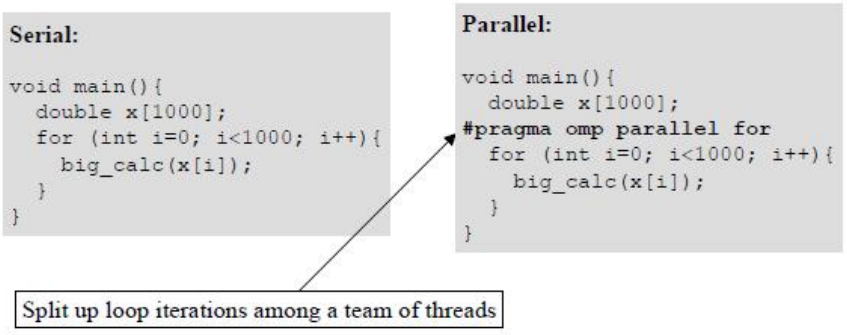
Voor het versnellen van NeuroSim is binnen dit project gekozen voor OpenMP. Ook in dit geval kon door toevoeging van (letterlijk) slechts één programmeerregel (plus natuurlijk de benodigde extra declaratie(s) en initialisaties) een groot deel van de code (doch bij lange na niet de gehele code!) parallel uitgevoerd worden. Toepassing van OpenMP verhoogt de snelheid van de simulaties aanzienlijk, maar 'beperkt' de output alsnog tot een zeker aantal neuronen: bij een 1-seconde modellering van 100.000 neuronen over 50 processoren kan NeuroSim alsnog meer dan 40 dagen aan het rekenen zijn, afhankelijk van de parameters en beginvoorwaarden. Op dergelijke (runtijd)resultaten wordt ingegaan in paragraaf 4.1.



Figuur B.2: Wet van Amdahl: versnelling tegen de parallelliseerbare fractie P bij constant aantal processoren.
(http://www.hku.hk/cc/sp2/workshop/html/parallel-intro/gif/amdahl1.gif)



Figuur B.3: Wet van Amdahl: versnelling tegen het aantal processoren bij constante parallelliseerbare fractie P.
(http://www.cs.umbc.edu/∼squire/images/amdahl.jpg)



Figuur B.4: Voorbeeld van parallellisatie met behulp van OpenMP van een for-loop in C-taal. Neem aan dat de bewerking 'big_calc(x[i])' een bewerking is die onafhankelijk is van de data op andere indices dan index i. Door toevoeging van slechts één regel aan de code wordt de for-loop geparallelliseerd. Instelling van het aantal CPU's/threads en de initialisatie voor OpenMP gebeurt extern van deze loop.
(http://www.osc.edu/supercomputing/training/openmp/openmp_0311.pdf)

B.3 Toepassing van OpenMP (op NeuroSim)

B.3.1 Tijdmeting

Om het effect van gebruik van OpenMP te onderzoeken, maar ook om parallelrekening toe te passen op de *juiste* (meest efficiënte) stukjes code, is het van toepassing de runtijd van een programmacode (voldoende) nauwkeurig te kunnen meten. De term 'runtijd' is hier wat betrekkelijk, want men kan kijken naar o.m. de 'wallclocktime' (de klokduur per run van het programma; eindtijd minus starttijd) en de 'CPU-tijd' (de totale calculeertijd opgeteld over alle gebruikte CPU's). Het doel van parallelrekening is het zo ver mogelijk verkleinen van de kloktijd. De uitkomst zal zijn dat hierdoor de totale CPU-tijd juist toeneemt. Dit heeft vooral te maken met de benodigde communicatie tussen de CPU's.

Het meten van de runtijd kan natuurlijk door de meting simpelweg toe te voegen aan de code. Denk aan het gebruik van *time_t* en *ctime*. Ook door in Unix of de Opdrachtprompt het commando *time NetworkSimulator.exe* in plaats van 'gewoon' *NetworkSimulator.exe* te geven, worden de duurgegevens van *NetworkSimulator.exe* na afloop weergegeven. Er verschijnt in dat laatste geval een 'real' tijd in beeld (de kloktijd, de werkelijke duur), een 'user' tijd (de totale CPU-tijd), en een 'sys' tijd (een achtergrondtijd benodigd voor o.m. het lezen van de code en eventuele wachttijd). Voor parallel-programmeurs (mensen die *met* een supercomputer werken) is eigenlijk alleen de 'real' tijd van toepassing. Voor mensen die *aan* een supercomputer werken zou ook de totale CPU-tijd interessant kunnen zijn. Naar de 'sys' tijd wordt in eenvoudige projecten eigenlijk niet zoveel gekeken.

De genoemde tijdmetingen kunnen variëren ten gevolge van bijvoorbeeld andere gelijktijdige belastingen op de betreffende CPU('s). De weergegeven runtijd kan en zal (licht) variëren per (identiek) experiment. De tijd die men meet dient dan ook meer als benadering van de orde van grootte tijdsduur opgevat te worden en niet als exacte meting. In principe is het meten van de rekentijd van een programmacode als een fysisch experiment: men zou het een aantal keren moeten herhalen en de uitkomst als een gemiddelde plus een zekere $fout$ of *spreiding* moeten geven.

Zoals inmiddels aangehaald, is het niet aan te raden om bij herprogrammering van een seriële code naar een parallelle code in OpenMP-stijl álle code die te parallelliseren is binnen een programma ook meteen te parallelliseren. Dit versnelt het programma niet efficiënt, omdat er dan veel communicatie om de hoek zal komen kijken. Uiteindelijk zou de parallelle code dan nog trager kunnen worden dan de oorspronkelijke code! De juiste methode is het meten van de tijdsduur (in percentage) van alle deelfuncties van de te parallelliseren code en, zo mogelijk, parallellisatie

stapsgewijs toe te passen vanaf de meest/langst gebruikte deelfunctie. Bij iedere stap dient de programmeur de resultaten nauwkeurig te observeren, want de experimentele output zou onafhankelijk moeten zijn van het aantal CPU's dat ingezet wordt.

Het meten van een dergelijke gesorteerde functielijst gebeurt (bijvoorbeeld) met het Unix-commando/-programma *gprof*. In de compileerregel voegt men toe $-g-pg$. Er wordt dan een file *gmon.out* aangemaakt, die naderhand ingelezen kan worden met *gprof*. De output is een gesorteerde opsomming van alle gebruikte deelfuncties: het laat per functie zien hoeveel maal en hoeveel tijd deze is gebruikt door de gemeten hoofdcode (zie hieronder). N.B.: *gprof* is een systeem dat werkt op de gebruikte supercomputer. Er zijn meerdere methoden om tijd en functielijsten te meten.

We geven een voorbeeld van een runtijdmeting over een seriële simulatie (gebruik van slechts 1 CPU) van 656 neuronen binnen *NetworkSimulator.cpp*. Simulatietijd (de hersenfunctioneringstijd die gesimuleerd wordt) is 1 seconde. De runtijd en 'function time listing' worden gemeten met behulp van *time* en *gprof*, respectievelijk. De zgn. batch-file (ter bevoeging voor een supercomputer waarop de code zal runnen) ziet er uit als boven in figuur B.5. Onder in figuur B.5 staat (een deel van) de output die zal verschijnen in de 'error-file', de output van de commandotoevoeging 'time'. Het blijkt dat de benodigde kloktijd van deze simulatie ongeveer 20 minuten was, de CPU-tijd (die bijna hetzelfde zou moeten zijn omdat er immers slechts 1 CPU gebruikt wordt) een minieme hoeveelheid tijd minder en de sys-tijd minder dan een seconde. Ietwat ingewikkelder, maar ook interessanter, is de output van *gprof*, die bestaat uit twee tabellen. Deze zijn weergegeven in figuren B.6 en B.7.

De eerste tabel heet 'Flat profile' en toont een lijst op volgorde van hoeveelheid tijd die de deelfunctie gebruikt. Blijkbaar wordt bij dit voorbeeld 51.07% van de tijd gebruikt voor de functie *AddEvent* onder klasse *ClassLinkChannel*. Deze functie wordt 1075427728 maal aangeroepen. Het is bekend dat de functie *AddEvent* de gebeurtenis van een AP-generatie toevoegt aan een EventList en blijkbaar gebeurt dit regelmatig. Op de tweede plaats in de lijst staat *Step*, de integratiestap-functie, onder de compartimentklasse. Daarna volgt de *Step* onder het natriumkanaal - als deelfunctie ('kindfunctie') van de *Step* op de tweede plaats. Dan volgt $Update(Tree)$ onder de compartimentklasse. Logisch dat de Step- en Update-functies hoog in de lijst staan - dit zijn immers in zekere zin de hoofdbestanddelen van de netwerksimulator: alle deelfuncties werken in principe onder *Step* en *Update*. De rest van de lijst bevat, gesorteerd naar percentage tijdsduur van de totaaltijd, steeds minder/korter gebruikte deelfuncties.

Beknopte toelichting voor de 'Flat profile' tabel (uitleg tevens weergegeven in de gprof-output):
*) time: percentage van de totale runtijd van de tijd die gebruikt wordt door deze functie;
*) cumulative seconds: som van het aantal seconden gebruikt door deze functie en degenen erboven in de lijst;
*) self seconds: aantal seconden gebruikt door deze functie: sorteringscriterium voor deze lijst;
*) calls: aantal keren dat deze functie werd aangeroepen - wanneer dit tenminste is gebeurd (anders blanco);
*) self ms/call: gemiddelde aantal milliseconden besteed aan deze functie per aanroep
*) total ms/call: gemiddelde aantal milliseconden besteed aan deze functie en z'n voorgangers in de lijst, per aanroep
*) name: naam van de functie.

De tweede tabel heet 'Call graph' en beschrijft de 'aanroepboom' van de code, gesorteerd naar de totale hoeveelheid tijd besteed aan iedere functie en bijbehorende kind-functies. Natuurlijk is 100 procent van de totaaltijd besteed aan de functie $main()$ (1 maal aangeroepen) met kinderen $AP-generator.Update()$, $GenerateNetwork()$, enzovoorts: allemaal functies die aangeroepen zijn binnen $main()$. 99.4 % van de tijd binnen $main()$ is gebruikt voor de functie $Update()$ onder de celklasse $AP-generator$ (dus het updaten van een complete cel - compartimenten, kanalen, etc.), met kinderen *UpdateTree* onder de compartimentklasse en *AddEvent* onder het/een synaptisch kanaal in het Hodgkin-Huxley-model.

```
# commentaar
#
# request # processors:
# @ tasks_per_node = 1
#
# @ notification = never
# @ input = /dev/null
# @ output = out.$(jobid)
# @ error  = err.$(jobid)
# @ job_type = parallel
# @ wall_clock_limit = 120:00:00
# @ queue

cd $HOME/Afstuderen/code

xlc++ -O3 -g -pg -qsmp=omp -o NetworkSimulator NetworkSimulator.cpp
export OMP_NUM_THREADS=1

time ./NetworkSimulator

gprof NetworkSimulator > gprof.output
```

```
real    20m8.426s
user    20m6.898s
sys     0m0.830s
```

Figuur B.5: 1) Batchfile voor tijdmeting via *time* en *gprof*; 2) output *time*. Gemeten wordt over de seriële code (1 CPU) van NetworkSimulator.cpp. Aantal cellen is 656 en simulatietijd is 1 seconde. Parameters zijn excitatie: 1, inhibitie: 1

Flat profile:

Each sample counts as 0.01 seconds.

% time	cumulative seconds	self seconds	calls	self ms/call	total ms/call	name
51.07	72.06	72.06	1075427728	0.00	0.00	.ClassLinkChannel::AddEvent(SynapseEvent)
20.57	101.09	29.03	336000000	0.00	0.00	.HHCompartment::Step(double, double)
3.20	105.60	4.51	303532837	0.00	0.00	.HHSodiumChannel::Step(double, double)
2.45	109.06	3.46	65600000	0.00	0.00	.ClassCompartment::UpdateTree()
2.45	112.52	3.46	65600000	0.00	0.00	.ClassCompartment::StepTree(double, double)
2.41	115.92	3.40	45874850	0.00	0.00	.DoubleExpSynapticChannel::GetCurrent(double)
2.20	119.02	3.10	303532837	0.00	0.00	.DoubleExpSynapticChannel::Step(double, double)
1.86	121.65	2.63	45874850	0.00	0.00	.HHPotassiumChannel::Step(double, double)
1.67	124.00	2.35	57067280	0.00	0.00	.DoubleExpSynapticChannel::Update()
1.61	126.27	2.27	54711352	0.00	0.00	.HHLeakChannel::Update()
1.60	128.53	2.26	57067280	0.00	0.00	.HHLeakChannel::Step(double, double)
1.30	130.36	1.83	336000000	0.00	0.00	.HHLeakChannel::GetCurrent(double)
1.19	132.04	1.68	427627365	0.00	0.00	.HHCompartment::Update()
1.13	133.63	1.59	324280930	0.00	0.00	.DoubleExpSynapticChannel::Step(double, double)
1.02	135.07	1.44	324280930	0.00	0.00	.HHPotassiumChannel::GetCurrent(double)
1.01	136.50	1.43	324280930	0.00	0.00	.HHSodiumChannel::GetCurrent(double)
1.01	137.92	1.42	324280930	0.00	0.00	.HHSodiumChannel::Update()
0.60	138.76	0.84	82964632	0.00	0.00	.HHPotassiumChannel::Update()
0.45	139.40	0.64	2277857	0.00	0.00	.PersistentSodiumChannel::Step(double, double)
0.38	139.94	0.54	89338480	0.00	0.00	.DoubleExpSynapticChannel::AddEvent(SynapseEvent)
0.36	140.45	0.51	65600000	0.00	0.00	.PersistentSodiumChannel::Update()
0.21	140.75	0.30				.APGenerator::Update()
0.21	141.04	0.29	89338480	0.00	0.00	.main
0.01	141.06	0.02	360000	0.00	0.00	.PersistentSodiumChannel::GetCurrent(double)
0.01	141.07	0.01	2560000	0.00	0.00	.ChandCell::GetSoma()
0.01	141.08	0.01	2560000	0.00	0.00	.DPyrCell::GetSoma()
0.01	141.09	0.01	463826	0.00	0.00	.SPyrCell::GetSoma()
0.01	141.10	0.01	1	10.00	462.44	.NetworkConnection::NetworkConnection()
0.00	141.10	0.00	928896	0.00	0.00	.GenerateNetwork()
0.00	141.10	0.00	927480	0.00	0.00	.CellGroup::GetGroupSize()
0.00	141.10	0.00	430336	0.00	0.00	.CellGroup::GetNumberChannel()
0.00	141.10	0.00	429680	0.00	0.00	.FindGroupNumber(int, CellGroup*, int)
0.00	141.10	0.00	429680	0.00	0.00	.myDistance(CellPosition, CellPosition)
0.00	141.10	0.00	429680	0.00	0.00	.xyzDistance(CellPosition, CellPosition)
0.00	141.10	0.00	420644	0.00	0.00	.FindFirstColumnNumber(int, CellGroup*)
0.00	141.10	0.00	420644	0.00	0.00	.MakeConnection(NetworkConnection*, int, int, double, double)
0.00	141.10	0.00	420644	0.00	0.00	.NetworkConnection::SetReceiver(unsigned short, unsigned char)
0.00	141.10	0.00	360000	0.00	0.00	.Bask1Cell::GetSoma()
0.00	141.10	0.00	360000	0.00	0.00	.Bask2Cell::GetSoma()
0.00	141.10	0.00	360000	0.00	0.00	.Bask3Cell::GetSoma()

Figuur B.6: output *gprof* (tabel 1: Flat profile) behorende bij figuur B.5.

```
                     Call graph (explanation follows)


granularity: each sample hit covers 4 byte(s) for 0.01% of 141.10 seconds

index % time    self  children    called     name
                                             <spontaneous>
[1]    100.0    0.30  140.80                 .main [1]
                0.51  139.79 65600000/65600000     .APGenerator::Update() [2]
                0.01    0.45       1/1           .GenerateNetwork() [24]
                0.02    0.00  360000/360000      .ChandCell::GetSoma() [29]
                0.01    0.00 2560000/2560000     .SPyrCell::GetSoma() [31]
                0.01    0.00 2560000/2560000     .DPyrCell::GetSoma() [30]
                0.00    0.00   43182/463826      .NetworkConnection::NetworkConnection() [32]
                0.00    0.00  360000/360000      .Bask3Cell::GetSoma() [40]
                0.00    0.00  360000/360000      .Bask2Cell::GetSoma() [39]
                0.00    0.00  360000/360000      .Bask1Cell::GetSoma() [38]
                0.00    0.00   43182/43182       .NetworkConnection::GetValues(unsigned short*, unsigned char*, unsigned short*, unsigned short*) [43]
                0.00    0.00   43182/43182       .APGenerator::AddConnection(SynapseConnection) [41]
                0.00    0.00   12287/12287       .DPyrCell::GetSynapticChannelList() [45]
                0.00    0.00   12280/12280       .SPyrCell::GetSynapticChannelList() [46]
                0.00    0.00    4731/4731        .Bask3Cell::GetSynapticChannelList() [48]
                0.00    0.00    4713/4713        .Bask1Cell::GetSynapticChannelList() [49]
                0.00    0.00    4596/4596        .ChandCell::GetSynapticChannelList() [50]
                0.00    0.00    4575/4575        .Bask2Cell::GetSynapticChannelList() [51]
                0.00    0.00       8/8           .void ReadTable<double>(char const*, double*, int, int) [59]
                0.00    0.00       1/1           .void ReadTable<int>(char const*, int*, int, int) [61]
-----------------------------------------------
                0.51  139.79 65600000/65600000     .main [1]
[2]     99.4    0.51  139.79 65600000             .APGenerator::Update() [2]
                3.46  135.69 65600000/65600000     .ClassCompartment::UpdateTree() [3]
                0.64    0.00 2277857/2277857       .DoubleExpSynapticChannel::AddEvent(SynapseEvent) [22]
-----------------------------------------------
                                270400000           .ClassCompartment::UpdateTree() [3]
                3.46  135.69 65600000/65600000     .APGenerator::Update() [2]
[3]     98.6    3.46  135.69 65600000+270400000 .ClassCompartment::UpdateTree() [3]
                1.83  133.86 336000000/336000000     .HHCompartment::Update() [4]
                              270400000           .ClassCompartment::UpdateTree() [3]
-----------------------------------------------
```

Figuur B.7: output *gprof* (tabel 2: Call graph) behorende bij figuur B.5.

Beknopte toelichting voor de 'Call graph' tabel (uitleg tevens weergegeven in de gprof-output): Iedere rij in de tabel bestaat uit verschillende regels. De regel met de index op de linkermarge lijnt de betreffende functie uit. De regels erboven geven de functies aan die deze functie *hebben* aangeroepen (de 'moederfuncties') en de regels eronder geven de functies aan die deze functie *heeft* aangeroepen (de 'kind-functies'). De tabel bestaat uit:
*) index: een uniek getal gegeven aan iedere rij van de tabel. Indices zijn numeriek gesorteerd naar percentage tijdsgebruik binnen de hoofdcode;
*) % time: percentage van de totale tijd gespendeerd aan deze functie en bijbehorende kind-functies;
*) self: de totale hoeveelheid tijd gespendeerd aan deze functie;
*) children: de totale hoeveelheid tijd getransporteerd in deze functie door kind-functies;
*) called: aantal keren dat de functie is aangeroepen;
*) name: naam van de huidige functie + index.
De functienamen worden onder de tweede tabel, dus onderaan in *gprof* (ter verduidelijking nogmaals) geïndexeerd.

B.3.2 Implementatie

OpenMP is een 'portable programming interface for shared memory parallel computers' [37] en kan worden toegepast binnen de C-taal, maar evenzo binnen andere talen, zoals Fortran. Binnen C++ dient boven de *main*() in ieder geval het commando $\#include\,\langle omp.h\rangle$ te staan, zodat de compiler 'is voorbereid' op gebruik van OpenMP. *omp.h* bevat (sub)routines die de wensen van de gebruiker vertalen in programmeercode en deze dus omzetten in de juiste handelingen (per CPU). Een codegedeelte (do-loop) dat geparallelliseerd wordt, wordt nu voorafgegaan door '*#pragma omp parallel ...*' De '...' is optioneel/eventueel nodig en implementeert zgn. 'clauses': extra informatie waarmee de CPU's rekening dienen te houden (voorbeeld: welke variabelen zijn 'shared' en welke zijn 'private'?). Clauses blijven hier onbesproken. Voor de parallellisatie in $NetworkSimulator.cpp$ is alleen de for-clause gebruikt: er wordt aangegeven dat een for-loop geparallelliseerd wordt over de index waarover de for-loop itereert. De enige regel die dus binnen de code is toegevoegd is de regel '*#pragma omp parallel for*' boven het Step-commando in het hoofdprogramma. En that's it. Het Update-commando, dat de berekende waarden en eigenschappen uit de Step-regel daadwerkelijk toekent aan alle objecten, moet niet geparallelliseerd worden - de Updatemomenten zijn dan niet netjes uitgelijnd. Dit is een aantal maal in de praktijk geprobeerd en er was daarbij sprake van 'data-racing'. De output bleef niet identiek bij verandering van het aantal CPU's. Dit aantal CPU's wordt extern meegegeven (zie figuur B.5) en de 'master-thread' zal zelf de indices over de CPU's verdelen. In dit geval worden *alleen* de indices en *niet* de hoeveelheden werk eerlijk verdeeld over de CPU's. Een stap op cel i is misschien wel tientallen malen zoveel werk als een stap op cel $i+1$. Dit is typisch een gevolg van gebruik van OpenMP in plaats van MPI. Het eind van het parallelle stukje code is automatisch het eind van de for-loop. Dit is een impliciete barrière voor alle slave-threads. Zolang niet *iedere* thread bij het einde van de for-loop is aangekomen, gaat $NetworkSimulator.cpp$ niet verder met het runnen van de code ná de betreffende for-loop.

Een basisvoorbeeld voor de toepassing van OpenMP is te vinden in figuur B.8. Dit voorbeeld is ook veelvuldig te vinden op het web.

B.3.3 OpenMP versus MPI

MPI is de voornaamste parallel-programmering 'directive'. Het is een investering om een grootschalige simulatiecode te programmeren in MPI: dit kost meestal veel werk (omdat de gehele code herschreven dient te worden), maar het zal efficiëntere resultaten geven dan het relatief veel eenvoudigere OpenMP. Kortom: MPI levert méér, maar in (veel) langere werktijd voor de programmeur. Voor een handleiding over MPI wordt verwezen naar [38]. (Bijna) hetzelfde voorbeeld als in figuur B.8, doch in MPI-taal, is te vinden in figuur B.9.

Een ander voorbeeld van een parallellisatieroutine is Pthreads; deze routine blijft onbesproken.

```
#include<iostream>
using namespace std;
#include<fstream>
#include<stdio.h>
#include<omp.h>

int omp_get_num_threads(void);

int main(int argc, char * argv[])
{
       int NTHREADS, TID;
//Fork a team of threads giving them their own copies of variables
       #pragma omp parallel private(TID)
       {

//Obtain thread number
              TID = omp_get_thread_num();
              printf("Hello world from thread %i\n", TID);

              #pragma omp barrier
//Only master thread does this
              if (TID==0)
              {
                     NTHREADS=omp_get_num_threads();
                     printf("Number of threads = %i\n", NTHREADS);
              }
       }
       return 0;
}
```

Figuur B.8: Standaardvoorbeeld van de toepassing van OpenMP. Deze code laat iedere CPU 'Hello World from thread *_thread#_*' op het scherm printen. Deze berichten komen dus (in willekeurige volgorde!) van iedere CPU, waarvan elke zijn eigen nummer (ID) heeft. Onderaan het lijstje (vanwege de handmatig ingevoerde barrier) met *'Hello World'*s komt te staan 'Number of threads = *_# threads_*'.

```
#include <stdio.h>
#include "mpi.h"

int main( argc, argv )
int  argc;
char **argv;
{
    int rank, size;
    MPI_Init( &argc, &argv );
    MPI_Comm_size( MPI_COMM_WORLD, &size );
    MPI_Comm_rank( MPI_COMM_WORLD, &rank );
    printf( "Hello world from process %d of %d\n", rank, size );
    MPI_Finalize();
    return 0;
}
```

Figuur B.9: Standaardvoorbeeld van de toepassing van MPI. Deze code laat iedere CPU 'Hello World from process *_process#_*' (te vergelijken met 'thread#' uit figuur B.8) op het scherm printen. Deze berichten komen dus (in willekeurige volgorde!) van iedere CPU, waarvan elke zijn eigen nummer (ID) heeft.