NETMORPH manual

NETMORPH version 2011-06-24, manual updated 2014-04-03

Randal A. Koene, Frank Postma, Sander de Ridder, Sacha Hoedemaker, Andrew Carnell, Pieter Laurens Baljon, Jaap van Pelt and Arjen van Ooyen. *Department of Integrative Neurophysiology, VU University Amsterdam, The Netherlands*

Table of contents

1. Introduction

NETMORPH is a simulation framework for the developmental generation of 3D large-scale neuronal networks with realistic neuron morphologies. In NETMORPH, neuronal morphogenesis is simulated from the perspective of the individual growth cone. For each growth cone in a growing axonal or dendritic tree, its actions of elongation, branching and turning are described in a stochastic, phenomenological manner. In this way, neurons with realistic axonal and dendritic morphologies, including neurite curvature, are generated. Synapses are formed as neurons grow out and axonal and dendritic branches come in close proximity of each other. NETMORPH is a flexible tool that can be applied to a wide variety of research questions regarding neuronal morphology and synaptic connectivity. Research applications include studying the complex relationship between single neuron morphology and global synaptic connectivity. The assumptions and intrinsics of NETMORPH are described in Koene et al. (2009) (Ref. [1]).

NETMORPH is the result of the joint efforts of Randal Koene, Betty Tijms, Peter van Hees, Frank Postma, Sander de Ridder, Sacha Hoedemaker, Jaap van Pelt and Arjen van Ooyen of the Neuroinformatics Group at the Department of Integrative Neurophysiology, Center for Neurogenomics and Cognitive Research, VU University Amsterdam, De Boelelaan 1085, 1081 HV Amsterdam, The Netherlands. The work was supported by grants from NL NWO-CLS2003 CASPAN (635.100.005), EU MC-RTN NEURoVERS-it (019247) and EU BIO-ICT SECO (216593) awarded to Jaap van Pelt and Arjen van Ooyen.

2. Installing NETMORPH

Windows and 64-bit Linux users, please read platform specific notes below!

The following steps demonstrate how to obtain and install NETMORPH.

1. Download the NETMORPH source package. The most recent version of NETMORPH is available at: http://www.neurodynamics.nl
2. Save the archive file **netmorph-YYYYmmdd.HHMM.tar.gz** in a suitable directory or folder of your choice. The version identifier YYYYmmdd.HHMM will contain version numbers of the most recent version of NETMORPH, e.g. netmorph-20090224.1225.tar.gz
3. Extract the archive file
   a) **Linux users** may use tar, for example: tar zxvf netmorph-YYYYmmdd.HHMM.tar.gz
   b) **Windows users** may use WinRAR for Windows downloadable from http://www.rarlab.com
   (Windows users need to execute Cygwin Bash Shell at this point)
4. Change the active directory to the one that was created during extraction of the archive file:
   cd netmorph-YYYYmmdd.HHMM;
   NOTE: This directory is now the main NETMORPH directory:
   installation, running, output storage, etc. takes place within this directory
   Read the files README and INSTALL to determine if any information that is relevant to installation with the newest version of NETMORPH differs from the description in this manual:
   less README ; less INSTALL (If the program "less" is unavailable, try "more" or another text reader.)
5. Run the installation script within the NETMORPH directory:
   ./install.sh
6. If all went well, then test NETMORPH by running it with its default settings:
   ./netmorph

After these steps, you should have a working installation of NETMORPH and a running NETMORPH simulation that produces some characteristic results.

2.1. Platform specific notes

The instructions above suffice for installation of NETMORPH on 32/64-bit Linux and Mac OSX platforms, as well as for installation on Windows within Cygwin, if the Cygwin environment was ready for NETMORPH, for example due to a previous installation of an earlier version of NETMORPH. If you are installing NETMORPH in a Cygwin environment for the first time, follow the additional platform specific advice below.

Windows with Cygwin:
NETMORPH is tailored to a Unix operating environment. Windows users can install and use NETMORPH by providing such an environment within Windows through a collection such as Cygwin. Cygwin is free software available at http://www.cygwin.com/. Please follow the instructions there to install Cygwin, but note the points below about important programs to include during installation, so that NETMORPH will compile and will be fully functional. At some point during Cygwin installation, a list of possible download sites is presented, select one of the sites (one may choose a site from a nearby country). Subsequently a window is presented with packages that can be included in the Cygwin environment. When this opportunity is presented, *choose the option to display the full list* of available software by clicking the 'View' button and select the following packages for installation in addition to the Default packages included in the 'Curr' installation:

1. gcc-g++ : C++ compiler
2. make : The GNU version of the 'make' utility

3. Xfig : An interactive drawing tool
4. ImageMagick : Image manipulation software suite (utilities)

If Cygwin was already installed on your computer then the Cygwin Setup program is normally accessible either through a short-cut on the Windows desktop or from a Cygwin folder in the program menu. Please follow the instructions provided by the developers of Cygwin to determine the most up-to-date method of installing the necessary program packages!

In Windows, a suitable directory to which to copy the archive file **netmorph-YYYYmmdd.HHMM.tar.gz** may be the Windows desktop or your home directory, since those normally also appear under the installed root directory of Cygwin. You can use the Unix change directory command "cd" to navigate to the directory containing the archive file and proceed according to the usual installation steps. To use Xfig for the visualization of generated networks in Windows, type "startx" at the Cygwin prompt. An X-Windows terminal should appear, and X-Windows programs such as Xfig can be run by typing their names into that terminal.

2.2. Optimization

Various compilers, operating systems and processor architectures allow for optimization of the program at compile time through the use of compiler flags. For reasons of maximal support, these flags have been omitted from the current makefiles. Once performance becomes an issue, it is advisable to recompile NETMORPH after setting the parameter MACHOPT. Example settings (for -march, -mtune and -mfpmath) are commented in the current Makefile.

2.3. Quick start

After successful compilation, it is possible to grow single-neuron morphologies or networks of realistic neurons. The program has a large number of parameters that can be set to control the individual neuron's development, as well as 'administrative issues' during simulation. These parameters are all described in the manual that can be obtained from the same location as your NETMORPH distribution. In addition, two example scripts are provided that generate the example morphologies described in Koene et al. (2009) (Ref. [1]; Figures 8B and 8D). These scripts contain appropriate settings for all parts of the simulation. The scripts can be executed by the following command in the netmorph home directory:

./netmorph include=fig8b

and

./netmorph include=fig8d

respectively. This results in a series of files with prefix fig8b, followed by the timestamp, and then a description of the various outputs. You can investigate the effect of changing the various parameters by looking at changes in the output statistics. The manual describes more advanced ways of inspecting the output.

3. Running NETMORPH simulations

NETMORPH is a command line program, which means that its invocation at the command prompt provides the necessary parameters for a simulation run. The complete list of sources in which NETMORPH searches for commands is given in Table 1. Importantly, the order in which sources are parsed is the table order from top to bottom, and a redeclaration of a command replaces the value of earlier declarations.

Without any commands, running NETMORPH elicits a default simulation with some interesting morphogenesis results that can be used to test the program. If the current directory contains the NETMORPH program binary, type at the command prompt:
./netmorph
The directory reference "./" should be included unless the directory has been added to the PATH environment variable (as defined in Linux/Unix, Mac OSX and Cygwin under MS Windows). The default simulation is specified in the resource file .nibrrc.

The straightforward way to give simulation commands to NETMORPH is to append the commands at the command line. For example:
./netmorph neurons=100 days=21

Included scripts:
Commands can be stored in script files, where a semicolon (";") must separate each command. Such scripts are parsed if included with an include command. Parsing is performed in depth-first manner. Given the example command line:
./netmorph include=/netmorph-defs/standard-def dt=100 include=mydirectory/myscript days=7
NETMORPH parses commands on the command line. Consequently, the following command parsing steps are taken:
1. NETMORPH parses the file at /netmorph-defs/standard-def.
2. NETMORPH modifies the simulation step size (**dt**) to 100 seconds.
3. NETMORPH parses the file at mydirectory/myscript.
4. NETMORPH modifies the simulated duration of morphogenesis to 7 days.

Deeper nesting of commands is possible by using the include command within command line parameter (CLP) scripts. For additional advanced interface options see Chapter 9.

Table 1. Sources that provide model choices and parameter values for simulations with NETMORPH.

source	description
compiled defaults	These are the default choices and values specified in the NETMORPH executable program file.
the command line	Program input provided by instructions that follow the program name on the command line.
included CLP scripts	Command line parameters (CLP) stored in a text file and separated by semicolons (";"), included by the include command.

4. CONTEXT: morphogenesis

NETMORPH has many parameters. To simplify its use, we contextualize NETMORPH. That is, we customize the user experience to a specific application context. The context described in this manual is the use of NETMORPH to simulate morphogenesis of neurons and network formation based on the spatial proximity of developing neurons. This manual is intended to facilitate getting started with NETMORPH. We assume that the NETMORPH user intends to produce simulations with information, constraints and goals similar to those of simulation results presented in Ref. [1].

NETMORPH is customized to the morphogenesis context by separating the parameter space into USER and ADVANCED parameters. Any model options and parameters that are not necessary for simulations in the context described in the preceding paragraphs of this section are considered to

belong to the ADVANCED set of parameters. Explanations in this manual focus primarily on model options and parameters that belong to the USER set listed in Chapter 8 of this manual. The ADVANCED model and parameter commands are listed in Chapter 10 of this manual.

5. PROTOTYPING protocol using model schemas

Before describing the model parameters, it is important to understand the so-called prototyping protocol used in NETMORPH. That is the referencing method used to apply specific models and specific parameter values to components of the simulated network. In NETMORPH each new component inherits its model choices and parameter values from its logical parent or from a prototype if there is no parent. A prototype is a declaration of a specific model choice and associated parameter values, which can schematically apply to a number of components in a simulation.

Example 1: Neurite segments following a bifurcation inherit from the parent segment before the bifurcation.
Example 2: The initial segment of a dendrite has no preceding parent segment and therefore inherits from the most suitable prototype.

5.1. SET prefix labels

A set prefix label indicates that a model choice or parameter value applies to each member of a logical set. The set is composed of an optional user-defined "region" specification and one of the predefined "natural sets". Table 2 lists the "natural" sets that are predefined in NETMORPH.

The full prefix format in NETMORPH is [**region label.**]**[natural set.][contributing.]**, where the additional "contributing" identifier is an arbitrary label that is used only when multiple models are chained to work together to compute expected growth results for a component of a network, as described in Chapter 5.3. In the lists of commands in Chapters 8 and 10, wherever an asterisk "*****" precedes a command then that command can follow a prefix with the format introduced here.

Table 2. Predefined "natural" sets and their prefix labels.

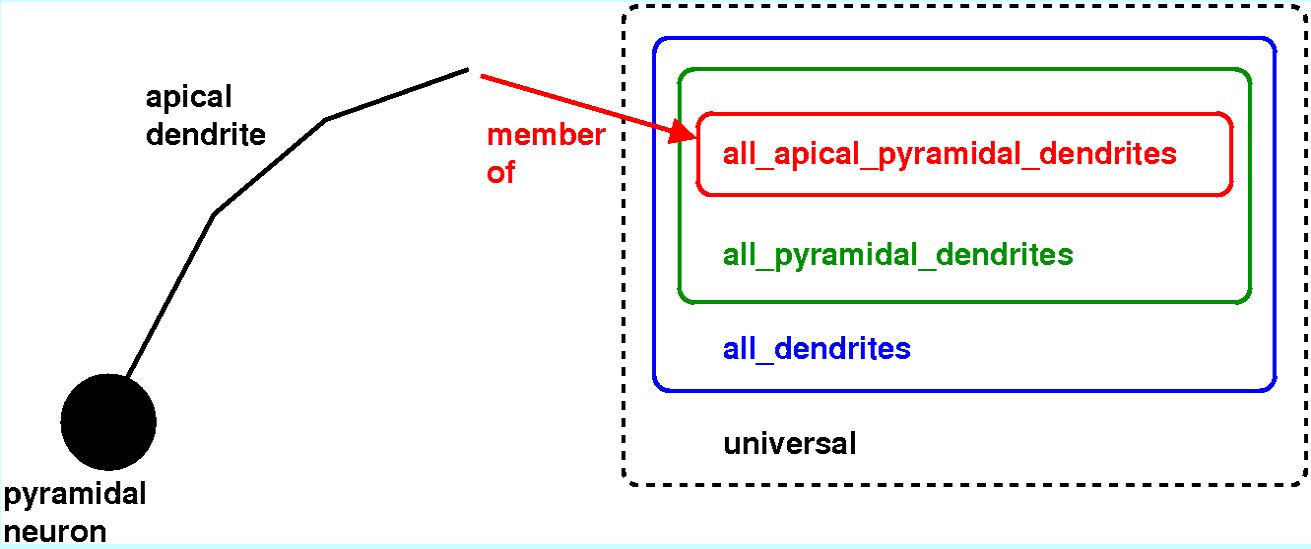
set prefix	set description
no prefix	The universal set, i.e. default settings.
all_axons.	The set of all axons, regardless of neuron type.
all_dendrites.	The set of all dendrites, regardless of neuron or dendrite type.
all_pyramidal_axons.	The set of all axons of pyramidal neurons.
all_pyramidal_dendrites.	The set of all dendrites of pyramidal neurons, regardless of dendrite type.
all_interneuron_axons.	The set of all axons of interneurons.
all_interneuron_dendrites.	The set of all dendrites of interneurons.
all_apical_pyramidal_dendrites.	The set of all apical dendrites of pyramidal neurons.

5.2. Schema matching by set logic

A component without a direct parent inherits from the smallest and most specific set for which a prototype has been declared. For example, using the natural sets in Table 2, a member of the set **all_apical_pyramidal_dendrites** can inherit model choices and parameter values from a prototype for the set **all_pyramidal_dendrites.** The example in Box 1 demonstrates the procedure for the growth cones of the apical dendrites of simulated pyramidal neurons in the case where an arbor elongation model is chosen and necessary parameters are set according to implicit and explicitly specified values.

Beware: It is important to note that if a model choice is specified for a given set then no further schema matching is done to locate any associated parameter values that were not explicitly specified for that set. Parameters with unspecified values receive default values that were compiled into the NETMORPH program. The default values are listed in the "default value" column of the command lists in Chapters 8 and 10 of this manual.

BOX 1. Explicitly specified or compiled default parameter values.



1. Is an arbor elongation model specified for all_apical_pyramidal_dendrites? - NO.
2. Is an arbor elongation model specified for all_pyramidal_dendrites? - YES.
3. Model chosen = van_Pelt.
4. Parameters: Is the F parameter explicitly specified for all_pyramidal_dendrites? - YES.
5. Set the F parameter as explicitly specified.
6. Parameters: Is the nu0 parameter explicitly specified for all_pyramidal_dendrites? - NO.
7. Use the compiled default value for the F parameter.

Example script:

```
arbor_elongation_model=van_Pelt;
growth_F=0.0;
growth_nu0=0.0000944;
aem.PDF=delta;
aem.PDF.mean=1;
all_pyramidal_dendrites.arbor_elongation_model=van_Pelt;
all_pyramidal_dendrites.growth_F=0.74;
all_pyramidal_dendrites.growth_nu0=0.0017940;
all_pyramidal_dendrites.aem.PDF=delta;
all_pyramidal_dendrites.aem.PDF.value=1;
```

In the above example, an arbor elongation model and its relevant parameter values are explicitly specified for the universal set. Additionally, an arbor elongation model and its relevant parameter values are explicitly specified for apical dendrites of pyramidal neurons. Any neurite arbors that are not apical dendrites of pyramidal neurons receive the universal model and universal parameter values. If the PDF model (delta) and its value (1) had not been explicitly specified for the set all_apical_pyramidal_dendrites, then the neurites of pyramidal neuron apical dendrites would use the default probability density function for arbor elongation models and corresponding default parameter values.

5.3. CONTRIBUTING identifiers

In some cases, the simulation of aspects of neurite growth such as elongation, branching, turning or direction of growth may be best achieved if the hypotheses underlying two or more growth model can be combined in some way. To enable this, NETMORPH permits weighted chaining of many of the developmental models within each category. The following example demonstrates how chaining two different growth cone direction models can produce a desired neurite curvature as well as a general trend of directed growth to connect different simulated layers of neurons.

Example script:

```
all_axons.direction_model=segment_history_tension;
  dirhistory_selection=none;
all_axons.dm_label=axondm;
  all_axons.axondm.dm_weight=1.0;
  all_axons.axondm.direction_model=cell_attraction;
veeranglemin=0.0;
veeranglemax=0.75;
```

In this example, a **segment_history_tension** direction model is placed at the head of a chain of direction models that is applied to members of the *all_axons* natural set in the *universal network* region (no region prefix). That direction model computes an expected direction of growth by taking into account the tensile influence of preceding neurite segment pieces up to the most recent branch point (or soma), without the possibility that the history of preceding directions is randomly truncated (**dirhistory_selection**=none). Specifying the label axondm indicates that another direction model should be chained into this computation of the expected direction. The reference weight of the model at the head of a chain is always 1.0, so that the second model is given equal weight by specifying a **dm_weight** of 1.0. The contributing direction model is a **cell_attraction** direction model. It computes an expected direction vector toward the centroid of a group of target neurons, for example the neurons of an adjacent region. The expected direction vectors computed by the two models after each turn in the neurite fiber are combined by the weighted sum $\mathbf{d} = \mathbf{d}_{\text{seghistension}} + 1.0\mathbf{d}_{\text{cellattraction}}$. The vector **d** is then modified by a perturbation that applies to the combined result of the chain of computations, within a range of perturbation angles from 0 to 0.75 radians.

5.4. REGION specific populations of neurons of selected type

When **regions** are used to specify the volumes in which neuron cell bodies are placed during network initialization, then it is possible to indicate specific populations of neurons that should be placed within a region, in addition to the randomized selection of a number of neurons from the general (network-wide) populations of neurons. The additional populations are specified through the value of a parameter that is formed by the combination of a region label and a neuron type identifier. It is possible to set the number of neurons in the general population to zero, so that the numbers of each type of neuron to be placed in each region are defined entirely independently.

```
neurons=0;
regions=IV V VI;
IV.pyramidal=15;
V.pyramidal=10;
V.interneuron=5;
VI.interneuron=15;
```

Example script excerpt: Used in a complete script, the parameter specifications in the example above produce a network with no general population neurons, but three regions containing specific numbers of pyramidal neurons and interneurons.

5.5. REGION specific models and parameter values

When **regions** are used to specify the volumes in which neuron cell bodies are placed during network initialization then it is possible to define models and parameter values that apply exclusively to the neurons that are members of a specific region. Note that regions are allowed to overlap or occupy the same space in a 3D simulation, which makes it possible to define neurons of the same type (e.g., pyramidal neurons) with different growth models and parameter values that are distributed in the same spatial volume.

Using a region identifier, as well as a natural set label (see Table 2), model choices and parameter values specified are applied as a schema. Region-specific schemas are applied first, before seeking best matches among the universally applicable schemas.

The network-wide universal schemas are defined as shown in the preceding examples. The following piece of an example script demonstrates model specification in a simulation with three different regions.

Example script:

```
IV.pyramidal=15;

V.pyramidal=10;
V.interneuron=5;
V.all_interneuron_axons.arbor_elongation_model=van_Pelt;
V.all_interneuron_axons.growth_nu0=0.0001;
V.all_interneuron_axons.growth_F=0.16;

VI.interneuron=15;
VI.all_interneuron_axons.arbor_elongation_model=van_Pelt;
VI.all_interneuron_axons.growth_nu0=0.0001;
VI.all_interneuron_axons.growth_F=0.16;
```

The three layers (IV, V and VI) are composed of neuron populations with different numbers of specific neuron types. Pyramidal neurons are placed in regions IV and V, interneurons are placed in regions V and VI. Within the regions that contain interneurons, region-specific schemas define elongation rates of the axons of interneurons that differ from the elongation rates defined by network-wide schemas applied to all other neurons.

5.6. How to generate pyramidal neurons with a 3D pyramidal apical dendrite

A special terminal segment elongation model, **pyrAD_BESTLNN**, exists to aid in the "prototyping" of three dimensional pyramidal neurons. In situ, pyramidal neurons have a distinct shape, characterized to a large degree by a long apical dendrite with specific features: linear extension up to several hundred micrometers, oblique branches at right angles to the main trunk of the apical dendrite, and strong arborization at the end of the linear extension.

When using **pyrAD_BESTLNN** in NETMORPH, the characteristic shape of the trunk portion of the apical dendrite is determined through the usual set of models and parameters that apply to pyramidal apical dendrites, e.g. all_pyramidal_apical_dendrites.direction_model (and its parameters), all_pyramidal_apical_dendrites.branching_model (and its parameters), etc. In this way, it is still possible to control the likelihood that the trunk of the apical dendrite bifurcates into two trunk branches, the curviness of the trunk, and more. It is even possible to target the growth of the apical dendrite at another population of neurons by including a cell attraction model in the chain of direction models (see the description of parameters and models in the following sections).

The usual parameters for growth of the trunk involve very little or no bifurcation (B_{inf}=0), very little curvature and a much greater mean elongation rate than that of basal dendrites. The pyrAD_BESTLNN model inherits its elongation function from the non-normalizing BESTL model, so that a probability distribution determines absolute elongation rates.

The **pyrAD_BESTLNN** model is special, in that it contains parameters that describe the appearance of oblique branches to the trunk, as well as all of the models to use for growth of oblique branches and for growth ("tufting") at the tip, once the apical trunk reaches its full extent. The models that can be specified uniquely for the tuft and oblique branches of an apical dendrite (identified by a unique prefex) are: **terminal segment elongation models, elongation rate initialization models, terminal segment branching models, branch angle models** and **direction models.** It is even possible to use unique arbor elongation model parameters by specifying a special terminal segment elongation model (**van_Pelt_specBM**). Details concerning the use of these models are included in the next section, which contains a fully described sample simulation script.

Note: It is possible to create the trunk with oblique branch points as a near-initial state by setting a very large elongation rate for the apical dendrite, which results in the completion of the trunk after only a few simulation steps.

6. Writing and running a simple sample simulation script

In this example script we will only focus on getting the program running and learning how to write a script that is able to be simulated in NETMORPH.

6.1. Creating an editable textfile in the NETMORPH folder

For Cygwin users:
- The default folder is "C:\Cygwin\home\(username)\netmorph"
- Right Click in the folder and mouse over 'New' and press 'Text Document'.
- Name it for instance 'test.txt'

For LINUX users:
Open the editable text file using any text editor.
In the first few notes it is recommended to write some information about the simulation you are

currently writing. Do this by adding "#" (or // in C++) to every comment line.

Example:

```
# This is a test file for writing a simple sample simulation script
```

6.2. Adding parameters regarding size and duration of the simulation

`days=15;` During this simulation the neurons have been growing for 15 days.

`randomseed=0;` By setting 'randomseed' to 0, each time a simulation is run a different random seed will be used in the pseudo-random generator, therefore leading to different simulation results.

`dt=100;` The simulated time interval between fixed-step updates of the developing network is set to 100 seconds.

`neurons=1;` This parameter defines the number of neurons to grow out. It is possible to specify a virtual space in which the neurons grow; this feature is described in Chapter 8.

6.3. Adding growth and bifurcation parameters

In NETMORPH it is possible to change a great number of parameters, but fortunately not every parameter has to be described in a sample script. If parameters are not used in the script NETMORPH will use its own default values for those parameters. In this example we will create one neuron using only growth and bifurcation parameters.

```
arbor_elongation_model=van_Pelt;                                    *
growth_nu0=0.0001;
growth_F=0.16;
elongation_rate_initialization_model=nonnorm_BESTL_length_distribution;
eri.PDF=normal;
eri.PDF.mean=0.0003;
eri.PDF.std=0.0001;
E=0.3;
E_competes_with=whole_neuron;
```

```
branching_model=van_Pelt;                                           *
B_inf=20;
S=2;
tau=1000000;
```

* These USER and ADVANCED parameters are described in Chapters 8 and 10.

6.4. Adding output parameters

After adding all the neuron parameters, information is needed about the output of your data. For statistical analysis it is recommended to save text files of the output. Also, NETMORPH has a built-in visualizer that can be used to view the sample simulation.

In this sample script only the use of the text file output is required. The other parameters are set to 'false' or default value.

```
outattr_make_full_Txt=true;
outattr_Txt_sequence=false;
outattr_Txt_separate_files=true;
outattr_track_nodegenesis=true;
outattr_show_progress=true;
outattr_show_figure=false;
outattr_show_stats=false;
figattr_make_full_Fig=false;
figattr_make_zoom_Fig=false;
```

6.5. Running the sample script

- Open the Cygwin Bash Shell executable
- Locate script file in the NETMORPH default folder, e.g.: cd C:/Cygwin/home/username/netmorph/
- Execute NETMORPH and include the sample script: ./netmorph include=test.txt

6.6. Visualizing the sample script (for Cygwin users)

NETMORPH has an analysis and visualization tool for Windows. The visualization tool is called 'startWinMovie.jnlp'
- Double-click to start this program.
- Click Start Visualization!
- Locate the .continuationnodes output file of the sample simulation script (NETMORPH default folder).
- Click Open.
- The sample neuron(s) are now viewable in the window (Fig. 1)

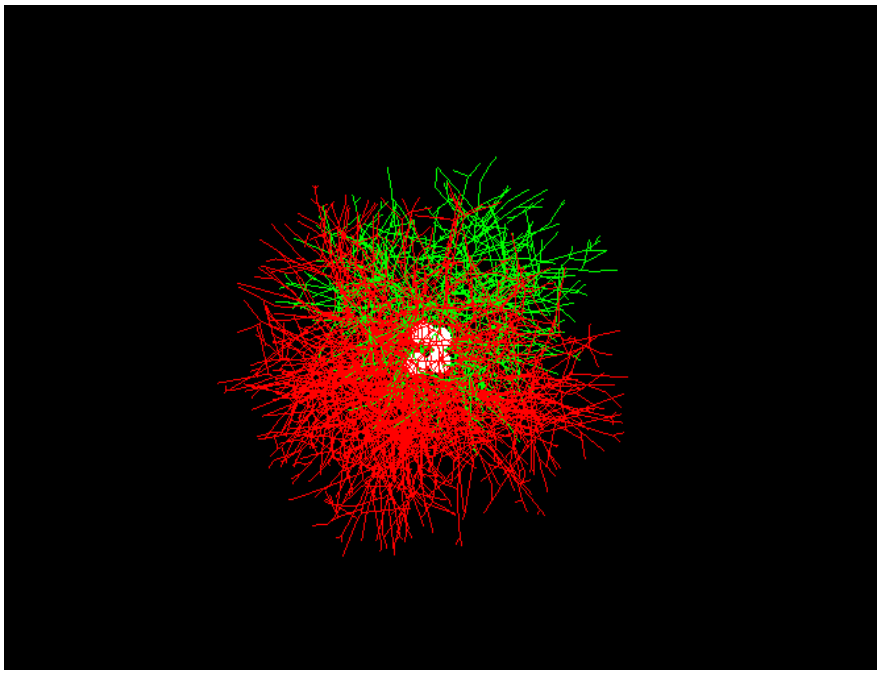
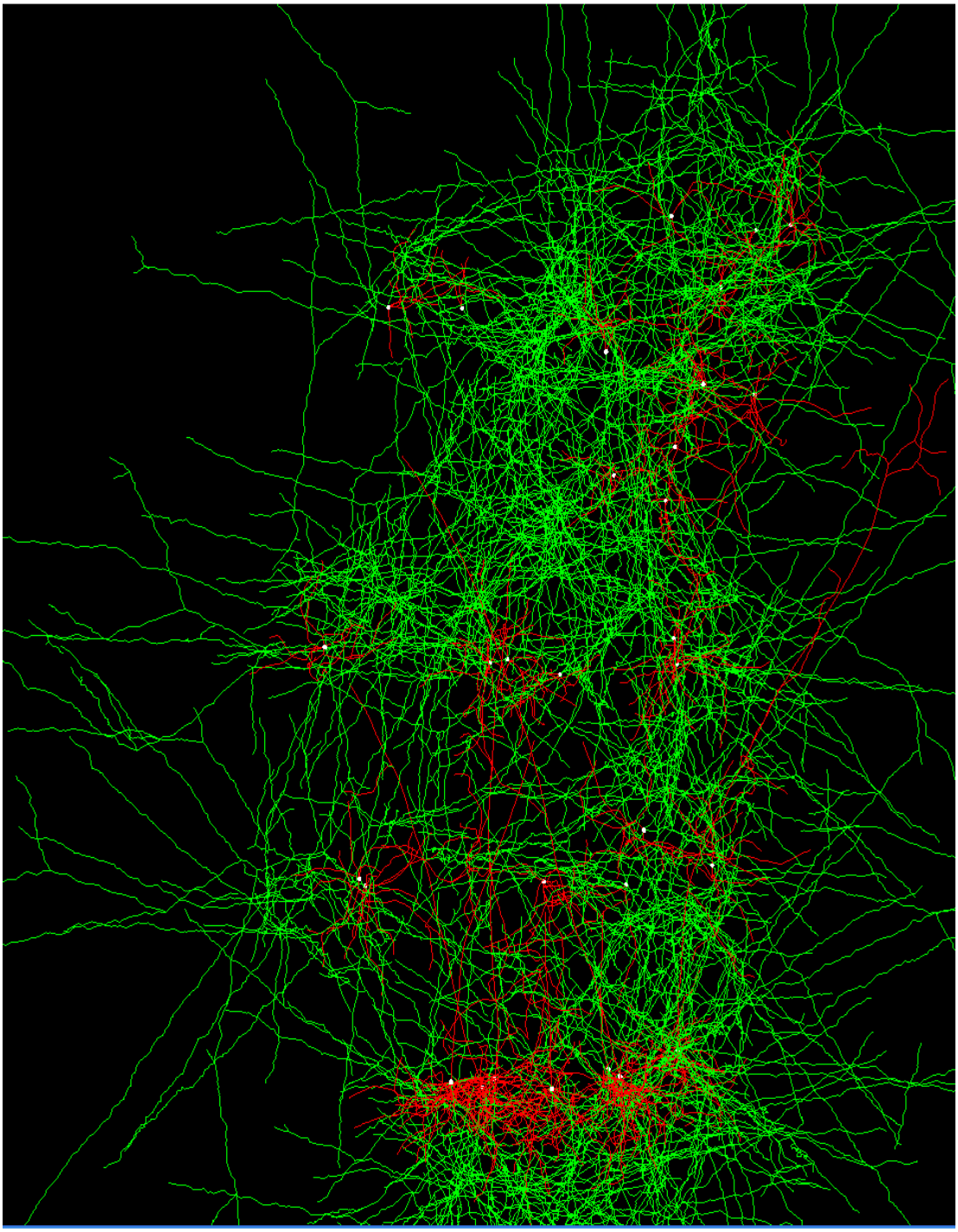


Fig. 1. The visual result of a neuron generated using the sample script described in Chapter 6.

7. A fully described sample simulation script

The following is a complete sample simulation script separated into logical paragraphs that explain each portion of the script. The script defines neuronal populations that are arranged spatially in layers roughly intended to resemble the layers of the human cerebral cortex. Only the deepest layer in this simulation explicitly contains pyramidal neurons, and the script includes model parameters for the simulated growth of apical dendrites from those pyramidal neurons through successive layers. The resulting network after 21 simulated days of development is shown in Fig. 2.



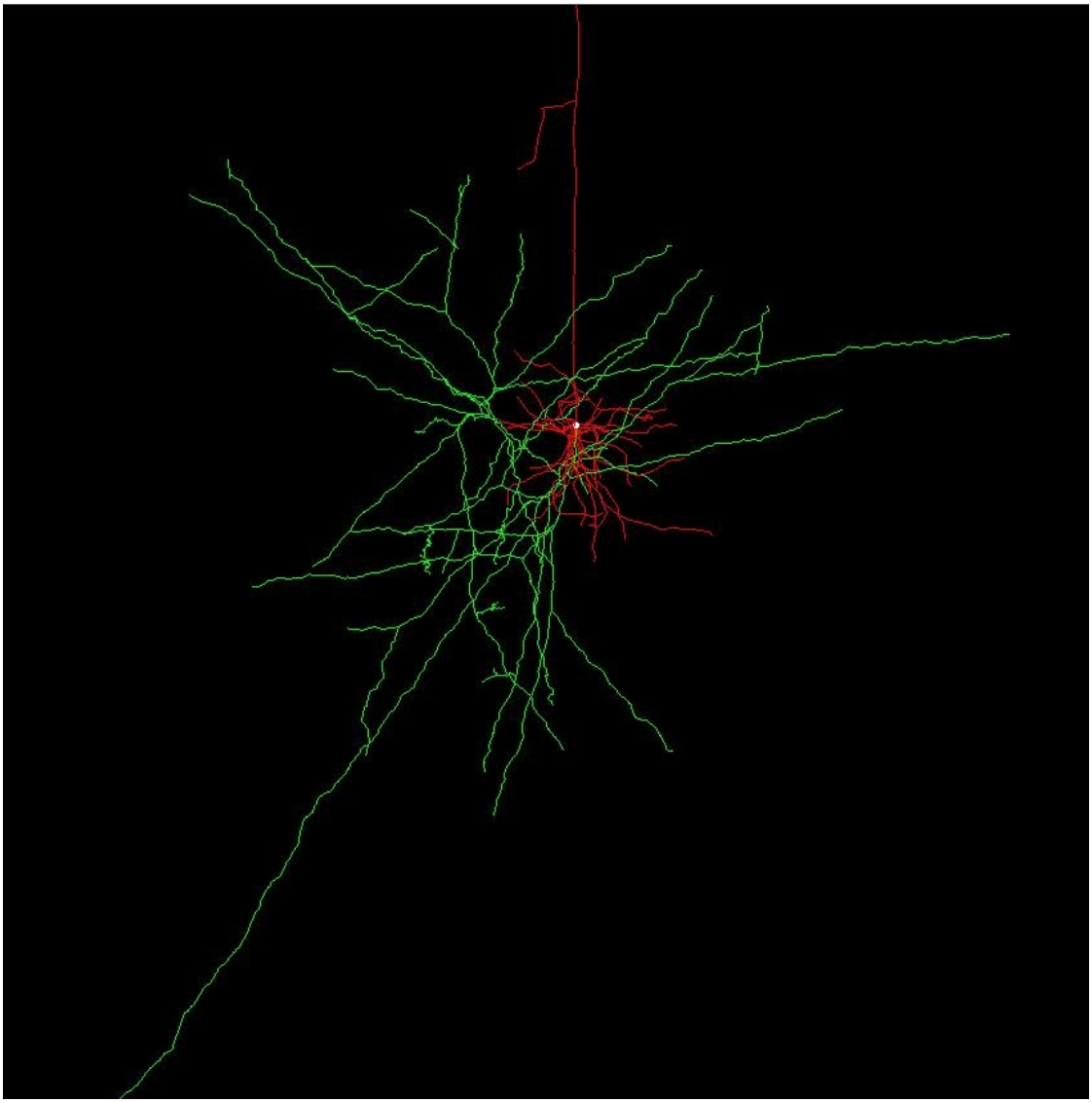


Fig. 2. (top) Network generated with the full sample script after 21 simulated days of development. (bottom) Single pyramidal cell of the network. Visualization by NEURON3D.

Part 1 – Comments identifying the simulation script and its purpose:

```
# Sample.txt (NETMORPH versions 20080722+)
# Randal A. Koene
# edited by: Sacha Hoedemaker 20090206
#
# This simulation demonstrates the growth of pyramidal neurons and
# interneurons in three layers. Features demonstrated include:
# 1. Region specific numbers of specific neuron types.
# 2. Region specific parameter specification per neuron type.
# 3. Apical dendrite development with specific models and
#    functions for oblique branches and tuft branches.
```

Part 1 is a block of comment lines. Any line of text provided to NETMORPH in a script (or directly on

the command line) that is preceded by the hash character "#" or by the C++ comment line convention "//" is considered a line of comment that is ignored by the command parser. Here, the comment block describes the version of NETMORPH with which the script is guaranteed to work, author and date stamp, and the purpose of the simulation script.

Part 2 – Temporal duration, temporal granularity and deterministic or non-deterministic simulation:

```
days=21;
randomseed=0;
//randomseed=1196091470;
dt=100;
```

Part 2 specifies how many days of neural development NETMORPH should simulate. By setting randomseed to 0, each time a simulation is run a different random seed will be used in the pseudo-random generator, therefore leading to different simulation results. The comment line shows that we can use a specific random seed instead to reproduce identical simulation results on every run. The simulated time interval (**dt**) between fixed-step updates of the developing network is set to 100 seconds.

Part 3 – Initializing soma morphology:

```
# SOMA INITIALIZATION PARAMETERS
pia_attraction_repulsion_hypothesis=true;
use_specified_basal_direction=true;
specified_basal_direction_relx=0.001;
specified_basal_direction_rely=0;
specified_basal_direction_relz=-1;
pyramidal.min_basal=4;
pyramidal.max_basal=8;
pyramidal.basal.minangle=0;
pyramidal.basal.maxangle=1.5;
pyramidal.basal.force_model=surface_division;
all_axons.L0=15,25;
```

In this simulation, we want our pyramidal cell morphologies to resemble those seen in cortical slices, and we want to arrange initial directions of growth of the apical dendrites to resemble the arrangement in cortical slices. For this reason, part 3 includes a number of ADVANCED parameters (Chapter 10):

1. The initial somatic arrangement assumes that apical dendrite grow toward the pia and that the apical dendrite is at the opposite end of the soma in relation to the axon and basal dendrites (**pia_attraction_repulsion_hypothesis**).
2. Instead of randomizing the somatic locations of the root segments of basal dendrites, we specify a vector from the soma center through the center of mass of the initial dendrite roots (**use_specified_basal_direction** and **specified_basal_direction_rel<x/y/z>**).
3. For each pyramidal cell, the number of basal dendrites is determined by a uniform random selection between 4 and 8.
4. The angular deviation from the center of mass vector at which dendrites can emerge is constrained between 0 and 1.5 radians.
5. Use a model for the distribution of basal dendrite roots, which constrains placement by

computing minimal angular difference such that the area between the basal minangle and maxangle is divided into equally large areas, one for each basal dendrite (**pyramidal.basal.force-model**=surface_division). Effectively, the basal dendrites appear to repel each other within the available area on the soma.

Finally, USER parameters (Chapter 8) in part 3 specify the ranges in which initial lengths of the root segments are selected by a uniform random function. Dendrites have initial lengths between the default minimum 9 µm and default maximum 11 µm, while axons have initial lengths between 22 µm and 25 µm.

Part 4 – Comments describing the model architecture:

```
# SOMA PLACEMENT PARAMETERS
// Selected excerpts of the following cortical laminar pattern
// are reproduced:
// a) Molecular layer I is not explicitly modeled here, but may
//    emerge as the result of physical constraints.
//    (Mainly extensions of apical dendrites and horizontally-
//    oriented axons.)
// b) II = external granular layer II (small pyramidal neurons
//    and numerous stellate neurons)
// c) III = external pyramidal layer III (small and medium-size
//    pyramidal neurons, as well as non-pyramidal neurons with
//    vertically-oriented intracortical axons)
// d) IV = internal granular layer IV (different types of
//    stellate and pyramidal neurons)
// e) V = internal pyramidal layer V (large pyramidal neurons)
// f) VI = multiform layer VI (few large pyramidal neurons and
//    many small spindle-like pyramidal and multiform neurons)
// The connectivity of minicolumns is not reproduced here.
// (See http://en.wikipedia.org/wiki/Cerebral_cortex)
```

The second comment block in part 4 of the script describes the successive cortical layers and their prevailing neurons, which are used here as a guide for the geometric placement of specific types of neurons with specific growth models. The simulation includes a number of neurons with soma in geometric regions that represent successive layers II to VI over a total distance of about 2000 µm.

Part 5 – Defining spatial regions:

```
shape=regions;
regions=II III IV V VI;
```

Part 5 specifies that the geometric placement of soma at the onset of simulated network development (**shape**) is done in multiple regions. The regions are declared with the labels II, III, IV, V and VI.

```
II.shape=disc;
II.neurons=0;
II.pyramidal=0;
II.interneuron=6;
II.minneuronseparation=100;
II.centerZ=-400;
II.shape.radius=600;
II.shape.thickness=50;

III.shape=disc;
III.neurons=0;
III.pyramidal=0;
III.interneuron=6;
III.minneuronseparation=100;
III.centerZ=-800;
III.shape.radius=600;
III.shape.thickness=50;

IV.shape=disc;
IV.neurons=0;
IV.pyramidal=0;
IV.interneuron=6;
IV.minneuronseparation=100;
IV.centerZ=-1200;
IV.shape.radius=600;
IV.shape.thickness=50;

V.shape=disc;
V.neurons=0;
V.pyramidal=0;
V.interneuron=6;
V.minneuronseparation=100;
V.centerZ=-1600;
V.shape.radius=600;
V.shape.thickness=50;

VI.shape=disc;
VI.neurons=0;
VI.pyramidal=7;
VI.interneuron=0;
VI.minneuronseparation=100;
VI.centerZ=-2000;
VI.shape.radius=600;
VI.shape.thickness=50;
```

Part 6 – Initializing region volumes and their contents:

Each region that contains neuron somata has a basic disc shape. The number of pyramidal neurons and the number of interneurons are set explicitly in each region. Here, we simulate the development of pyramidal neurons only in region VI, and their interaction with developing interneurons in the other regions. By setting the total number of neurons in each region to 0 (e.g., **II.neurons**=0), we insure that the specified pyramidal neurons and interneurons define all neurons placed in each region. Had this number been greater than the total of pyramidal neurons and interneurons in a region then the remaining neurons in that region would receive types according to global proportions specified (see **populationsize<type>** and **approxproportion<type>** in Chapter 8 and 10). The regions specified are shown in Fig. 3.

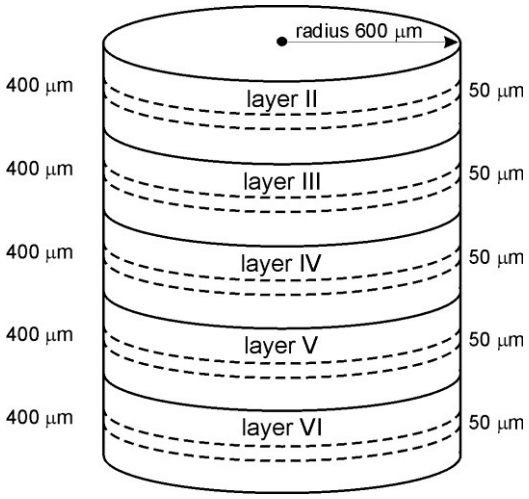


Fig. 3. Model regions II through VI accommodating cell bodies in a cortical layer-like spatial arrangement. Regions have a thickness of 50 μm, are spaced 400 μm from each other and have a radius of 600 μm.

Part 7 – Region specific, neuron type specific model parameters:

```
III.all_interneuron_axons.arbor_elongation_model=van_Pelt;
III.all_interneuron_axons.growth_nu0=0.0001;
III.all_interneuron_axons.growth_F=0.16;

V.all_interneuron_axons.arbor_elongation_model=van_Pelt;
V.all_interneuron_axons.growth_nu0=0.0001;
V.all_interneuron_axons.growth_F=0.16;
```

In part 7, we specify elongation model parameters that apply only to the growth cones on axons of interneurons in regions III and V. An arbor elongation model based on the phenomenological model functions published by van Pelt and Uylings (2003; Ref. [2]) governs the resources available to the elongating axon fibers. The parameter of the elongation rate is initialized to 0.0001 μm/s, which is slower than the rate specified for other axons (see below).

Part 8 – Environment physics:

```
environment_physics=pia;
pia.physical_boundary=spherical;
pia.spherical_center=0,0,-50000;
pia.spherical_radius=50000;
pia.spherical_maxrange=100;
pia.spherical_c_repulse=0.2;
```

In order to constrain fiber growth to the model cortical layers, and to force the tufts of rising apical dendrite fibers to spread out horizontally and fill in the volume that would be occupied by cortical

layer I, we specify physical constraints of the environment, each of which receives a label (**environment_physics**). Here, the only label is "pia". The type of environment physics to be applied is a spherical boundary (**pia.physical_boundary**). The sphere, which represents the confines of the cranial cavity has a radius of 5 cm, centered so that the boundary appears above region II, defining the top of the implicitly modeled cortical layer I. To achieve the desired shape of the bending fibers near the boundary, a repulsion coefficient of 0.2 (**pia.spherical_c_repulse**) is specified to apply within 100 µm of the boundary (**pia.spherical_maxrange**).

Part 9 – Elongation model parameters, applied to the universal set:

```
arbor_elongation_model=van_Pelt;
growth_F=0.39;
growth_nu0=0.00013889;
F_competes_with=same_arbor;

terminal_segment_elongation_model=BESTL;
tsem.PDF=delta;

elongation_rate_initialization_model=length_distribution;
```

All growth cones that do not belong to a smaller set for which arbor elongation model parameters are explicitly specified use the phenomenological model functions of elongation published by van Pelt and Uylings (2003; Ref. [2]), which include competition for resources at the level of axon or dendrite arbors (**arbor_elongation_model**, **growth_F** and **F_competes_with**). For these growth cones, an initial elongation rate of 0.00013889 µm/s is specified (**growth_nu0**), i.e., 12 µm per day. At each growth cone, the individual elongation rate is controlled by a specific variant of the van Pelt et al. (2001; Ref. [3]) model functions (**terminal_segment_elongation_model**), which was designed to take into account the position of a growth cone in a dendritic tree. In prior work, that model did not alter elongation rates between bifurcation points, so that we set the perturbation parameter to a delta function with the default value of 0 (**tsem.PDF**). After each bifurcation, the initial elongation rates of the new branches are selected from a normal distribution with a mean value equal to the mean value of elongation rates at all other growth cones (**elongation_rate_initialization_model**), and are assigned so that the greater of the two new elongation rates is given to the branch with the longer initial length (**length_distribution**).

Part 10 – Branching model parameters, applied to the universal set:

```
branching_model=van_Pelt;
min_node_interval=2.0;

branch_angle_model=Balanced_Forces;
```

All growth cones that do not belong to a smaller set for which a branching model is explicitly specified use the phenomenological model functions of branching published by van Pelt and Uylings (2003; Ref. [2]) with default parameter values (**branching_model**). Branching and turning events are further constrained at the universal set level by enforcing a minimum node interval of 2 µm. Branch angles are determined by a model of balanced forces that takes into account the elongation rates at the two growth cones after a bifurcation (**branch_angle_model**).

Part 11 – Turning model parameters:

```
TSTM=linear_rate;
turn_separation=5.0;
```

Stochastic turn events at which growing fibers change direction occur with a mean separation along the neurite fiber of 5 μm.

Part 12 – Direction model parameters, applied to the universal set:

```
direction_model=segment_history_tension;
veeranglemin=0.0;
veeranglemax=0.75;
dirhistory_selection=none;
```

All growth cones that do not belong to a smaller set for which a direction model is explicitly specified experience tensile influence on their direction of growth that is exerted by a weighted history of the directions, lengths and distances of preceding pieces of neurite fiber (**direction_model**). All the history contributes, without selecting a limited range (**dirhistory_selection**). An expected direction that is computed with the specified model is perturbed by a random angle up to 0.75 radians.

Part 13 – Elongation models parameters, applied to axons:

```
all_axons.arbor_elongation_model=van_Pelt;
all_axons.growth_nu0=0.0005208333;
all_axons.growth_F=0.16;
all_axons.F_competes_with=same_arbor;
all_axons.B_inf=13.21658;
all_axons.S=-0.20538;
all_axons.E=0.319251;
all_axons.elongation_rate_initialization_model=nonnorm_BESTL_length_distribution;
all_axons.eri.PDF=normal;
all_axons.eri.PDF.mean=0.00021367265496506256;
all_axons.eri.PDF.std=0.0003978;
all_axons.tau=1681541;
```

Axon growth cones in regions other than region III and region V, which received more specific elongation model parameters in part 7 of the script, also elongate according to model functions published by van Pelt and Uylings (2003; Ref. [2]), but with a parameter of the elongation rate that is initialized to the greater rate of 0.000520833 μm/s, i.e., 45 μm per day. Axons in regions II, IV and VI will become significantly larger than other axons and basal dendrites over the simulated time interval.

Part 14 – Direction model parameters with cell attraction, applied to axons:

```
all_axons.direction_model=segment_history_tension;
all_axons.dm_label=axondm;
all_axons.axondm.dm_weight=0.05;
all_axons.axondm.direction_model=cell_attraction;
```

In addition to a direction model with tensile history and default parameters, axon direction is guided by another chained-in direction model with the label “axondm”. The influence of that direction model is equal to the influence of the tensile model at the head of the chain of direction models, due to a relative weighting of 0.05 (**all_axons.axondm.dm_weight**). The contributing model (**all_axons.axondm.direction_model**) computes an expected direction vector according to the geometric centroid location of neurons in another region that attract the axon growth cones. By default, a region's axon growth cones are attracted to the cell somata in the next region defined.

Part 15 – Elongation model parameters, applied to dendrites:

```
all_dendrites.terminal_segment_elongation_model=nonnorm_BESTL;
all_dendrites.tsem.PDF=delta;
all_dendrites.tsem.PDF.value=0;
all_dendrites.tsem.branch.PDF=normal;
all_dendrites.tsem.branch.PDF.mean=6;
all_dendrites.tsem.branch.PDF.std=5;
all_dendrites.elongation_rate_initialization_model=nonnorm_BESTL_length_distribution;
all_dendrites.eri.PDF=normal;
all_dendrites.eri.PDF.mean=0.0000914464;
all_dendrites.eri.PDF.std=0.0000365786;
```

The elongation of growth cones of basal dendrites is governed by a non-normalizing terminal segment elongation model (**all_dendrites.terminal_segment_elongation_model**) that is based on the model functions published by van Pelt et al. (2001; Ref. [3]). This is a growth cone local phenomenological model that is not affected by explicit arbor-wide resource limitations, i.e. it does not request resources from an arbor elongation model. In accordance with this model, elongation rates are not perturbed between bifurcation nodes (**all_dendrites.tsem.PDF**=**delta** with value 0). The actual elongation rate of each growth cone is selected immediately after branching, in terms of absolute values (**nonnorm_BESTL_length_distribution**) generated according to the probability density function of the elongation rate initialization model (**all_dendrites.elongation_rate_initialization_model**). The mean rate chosen for this simulation is 0.0000914464 µm/s, i.e., 7.9 µm per day. Development models used in this simulation assume that branching is an event that takes some finite amount of time, so that we may assign to each branch an initial length selected with a normal probability density function with mean 6 µm and standard deviation 5 µm (**all_dendrites.tsem.branch.PDF**).

Part 16 – Branching model parameters, applied to dendrites:

```
all_dendrites.branching_model=van_Pelt;
all_dendrites.B_inf=2.52;
all_dendrites.tau=259680;
all_dendrites.E_competes_with=same_arbor;
all_dendrites.E=0.73;
all_dendrites.TSBM=van_Pelt;
all_dendrites.S=0.5;
```

Growth cones of basal dendrites branch according to the phenomenological model published by van Pelt and Uylings (2003; Ref. [2]), with specific parameter values that result in a constrained number of branches and competition between the growth cones within each dendrite (**all_dendrites.B_inf**,

all_dendrites.E and **all_dendrites.E_competes_with**). Most of the branching takes place during a short period of development (**all_dendrites.tau**). The probability of branching at individual growth cones is also influenced by the location of the growth cone, in accordance with the S parameter of the van Pelt et al. (2001; Ref. [3]) model functions (**all_dendrites.TSBM** and **all_dendrites.S**).

Part 17 – Command name substitution:

```
substitute=APD:all_apical_pyramidal_dendrites;
```

We substitute the short-hand **APD** for the set identifier **all_apical_pyramidal_dendrites** in all following commands.

Part 18 – Branching model parameters, applied to the trunk fibers of apical dendrites:

```
APD.branching_model=van_Pelt;
APD.B_inf=0.1;
APD.tau=400000;
APD.E=0;
APD.E_competes_with=same_arbor;
```

Branching of the trunk of an apical dendrite is very unlikely (**APD.B_inf**=0.1).

Part 19 – Elongation model parameters, applied to the trunk fibers of apical dendrites:

```
APD.terminal_segment_elongation_model=pyrAD_BESTLNN;

APD.tsem.trunklength.PDF=normal;
APD.tsem.trunklength.PDF.mean=80;
APD.tsem.trunklength.PDF.std=2;
APD.tsem.prefix=pyr1;


APD.tsem.PDF=delta;
APD.tsem.PDF.value=0;
APD.tsem.branch.PDF=normal;
APD.tsem.branch.PDF.mean=6;
APD.tsem.branch.PDF.std=1;

APD.elongation_rate_initialization_model=nonnorm_BESTL_length_distribution;
APD.eri.PDF=normal;
APD.eri.PDF.mean=0.0010208333;
APD.eri.PDF.std=0.000256;
```

The local elongation model at the growth cones of apical dendrites is explicitly intended for the apical dendrites of pyramidal neurons (**pyrAD_BESTLNN**) and enables the specification of a probability density function for the apical trunk length (**APD_tsem.trunklength.PDF** with a mean of 80 µm and standard deviation 2 µm), as well as the transition to separate model sets to be used for oblique branches and for the tuft of branching fibers at the end of the apical dendrite trunk. We specify the label "pyr1" to be used to identify model and parameter choices for the oblique branches and for the apical tuft (**APD.tsem.prefix**), as shown in the descriptions of parts 20 to 23 of the script.

As in part 14 of the script, the growth cone local phenomenological model specified is not affected by arbor-wide resource limitations and elongation rates are not perturbed between bifurcation nodes (**APD.tsem.PDF**=**delta** with value 0). After branching into obliques or the tuft, each branch receives an initial length selected by a normal probability density function with mean 6 μm and standard deviation 1 μm (**APD.tsem.branch.PDF**).

The apical dendrites, which appear at pyramidal neurons placed only in region VI of this simulation, elongate at a rapid rate so that we can see them extend through all the layers of the model cortex during the simulated time interval of neural development. To achieve this, the elongation rate intialization model (**APD.elongation_rate_initialization_model**) selects high rates of elongation with a mean value of 0.0010208333 μm/s (i.e., 88 μm/day).

Part 20 – Direction model parameters, applied to the trunk fibers of apical dendrites:

```
APD.direction_model=segment_history_tension;
APD.veeranglemin=0.0;
APD.veeranglemax=0.1;
```

The direction model that is applied to growth cones in the trunk of an apical dendrite (**APD.direction_model**) takes into account the tensile history of a fiber segment and is perturbed only by small random angles up to 0.1 radians.

Part 21 – Model parameters governing the placement of oblique branches on apical dendrites:

```
APD.tsem.obliques.PDF=normal;
APD.tsem.obliques.PDF.mean=5;
APD.tsem.obliques.PDF.std=1;
APD.tsem.obliqueangle.PDF=normal;
APD.tsem.obliqueangle.PDF.mean=0;
APD.tsem.obliqueangle.PDF.std=1;
```

A mean of 5 oblique branches appear on the trunks of apical dendrites (**APD.tsem.obliques.PDF**), branching at a mean angle that is orthogonal to the trunk (**APD.tsem.obliqueangle.PDF**).

Part 22 – Elongation model parameters, applied to the tuft fibers of apical dendrites:

```
pyr1.tuft.terminal_segment_elongation_model=nonnorm_BESTL;
pyr1.tuft.tsem.branch.PDF=normal;
pyr1.tuft.tsem.branch.PDF.mean=6;
pyr1.tuft.tsem.branch.PDF.std=1;
pyr1.tuft.tsem.PDF=delta;
pyr1.tuft.tsem.PDF.value=0;
pyr1.tuft.elongation_rate_initialization_model=nonnorm_BESTL_length_distribution;
pyr1.tuft.eri.PDF=normal;
pyr1.tuft.eri.PDF.mean=0.0003889;
pyr1.tuft.eri.PDF.std=0.000004;
```

At the tuft of each apical dendrite, growth cones change direction and the rate of branching increases.

To achieve this different part of the development of apical dendrites, growth cones in the tuft use a separate set of models. As in part 14 of the script, growth cones in the tuft are governed by a local phenomenological model of elongation that is not affected by explicit arbor-wide resource limitations. Elongation rates are not perturbed between bifurcation nodes (**pyr1.tuft.tsem.PDF**=**delta** with value 0). The elongation rate of each growth cone is selected in terms of absolute values (**nonnorm_BESTL_length_distribution**) by a probability density function of the elongation rate initialization model (**pyr1.tuft.elongation_rate_initialization_model**) with a mean rate of 0.0003889 µm/s (about 33.6 µm/day). The initial length of branches is selected by a normal probability density function with mean 6 µm and standard deviation 1 µm (**pyr1.tuft.tsem.branch.PDF**).

Part 23 – Branching model parameters, applied to the tuft fibers of apical dendrites:

```
pyr1.tuft.TSBM=van_Pelt_specBM;
pyr1.tuft.S=1;
pyr1.tuft.branching_model=van_Pelt;
pyr1.tuft.B_inf=25;
pyr1.tuft.tau=400000;
pyr1.tuft.E=0.3;
pyr1.tuft.E_competes_with=same_arbor;
```

Branching in the tuft of an apical dendrite is modeled with the functions published by van Pelt and Uylings (2003; Ref. [2]), and is subject to dendrite-wide resource limitations, but does not reference the same arbor elongaiton model parameters as growth cones in the trunk of the apical dendrite. Instead, the terminal segment branching model (**pyr1.tuft.TSBM**) specifies the use of a specific branching model (**van_Pelt_specBM**) for which the parameters are obtained through commands with the **pyr1.tuft** prefix. In all other respects, the terminal segment branching model computes expected branching probabilities in the same way as the TSBM **van_Pelt** model specified in part 15 of the script, with an S parameter value of 1. The specific arbor branching model (**pyr1.tuft.branching_model**) results in an increased number of branches over a greater time interval (**pyr1.tuft.B_inf** and **pyr1.tuft.tau**), while there is some competition within the apical dendrite (**pyr1.tuft.E** and **pyr1.tuft.E_competes_with**).

Part 24 – Elongation model parameters, applied to the oblique branch fibers of apical dendrites:

```
pyr1.oblique.terminal_segment_elongation_model=nonnorm_BESTL;
pyr1.oblique.tsem.branch.PDF=normal;
pyr1.oblique.tsem.branch.PDF.mean=6;
pyr1.oblique.tsem.branch.PDF.std=1;
pyr1.oblique.tsem.PDF=delta;
pyr1.oblique.tsem.PDF.value=0;
pyr1.oblique.elongation_rate_initialization_model=nonnorm_BESTL_length_distribution;
pyr1.oblique.eri.PDF=normal;
pyr1.oblique.eri.PDF.mean=0.0002;
pyr1.oblique.eri.PDF.std=0.000001;
```

As in the case of the tuft, the oblique branches of an apical dendrite also develop according to a separate set of growth models. The terminal segment elongation model selected for obliques here (**pyr1.oblique.terminal_segment_elongation_model**) is the same as that for the tuft, while the elongation rate initialization model has a smaller mean rate of elongation of 0.0002 µm/s, i.e., 17.28 µm per day.

Part 25 – Branching model parameters, applied to the oblique branch fibers of apical dendrites:

```
pyr1.oblique.TSBM=van_Pelt_specBM;
pyr1.oblique.S=1;
pyr1.oblique.branching_model=van_Pelt;
pyr1.oblique.B_inf=1.5;
pyr1.oblique.tau=500000;
pyr1.oblique.E=0.3;
pyr1.oblique.E_competes_with=same_arbor;
```

In the oblique branches of an apical dendrite, as in the tuft, the terminal segment branching model (**pyr1.oblique.TSBM**) specifies an S parameter of 1 and the use of a specific arbor branching model (**pyr1.oblique.branching_model**) with a small number of branches over an extended time interval (**pyr1.oblique.B_inf** and **pyr1.oblique.tau**) and some competition within the apical dendrite (**pyr1.oblique.E** and **pyr1.oblique.E_competes_with**).

Part 26 – General simulation parameters:

```
fibreswithturns=true;
branchatinitlength=false;
branchinsegment=true;
Abranchesatturns=false;
Dbranchesatturns=false;

candidate_synapses=false;
```

In part 25 of the script, we specify some parameters of simulated development that apply to all neurons. Branching is not enforced at the root length of axons and dendrites that was specified in part 3 of the script (**branchatinitlength**). When branches occur, they can appear at any point in the piece of elongated fiber segment that was grown during the preceding development time interval (**branchinsegment**). Remaining neurite fiber after the branch point is redistributed to the branches and is involved in the branch angles and allocation of elongation rates as described for parts 9 and 10 of the script. Growth cones can make turns during growth between branch points (**fibreswithturns**) with turning and direction models as described in parts 10 and 11 of the script. Nodes at which growth cones make turns during preceding development cannot become branch points through later probabilistic selection (**Abranchesatturns** and **Dbranchesatturns**). In this simulation, we do not seek potential sites of synapses in the developing network (**candidate_synapses**).

Part 27 – Simulation runtime and sampling parameters:

```
sample_dt=4320;
statsattr_collect_statistics=true;
```

During the stepwise simulation of neuronal development in the network, we collect sample data at intervals of 4320 seconds (**sample_dt**), i.e. every 72 simulated minutes. That data is used to produce statistical output about the developing network (**statsattr_collect_statistics**). When development is investigated through a sequence of structural output (in part 27 of the script we opt not to do this), or is visualized as a sequence of network figures, or by animation, as described in part 29 of the script, then

the sample rate also determines the intervals at which structure data is written to a text file or at which frames are produced for visualization.

Part 28– Textual data parameters:

```
outattr_show_progress=true;
outattr_make_full_Txt=true;
outattr_Txt_sequence=false;
outattr_Txt_separate_files=true;
outattr_track_synaptogenesis=true;
outattr_track_nodegenesis=true;
outattr_show_stats=true;
```

A textual indicator displays progress during the simulation of network development (**outattr_show_progress**). In the structural data, we opt to keep track of the simulation time points at which branching and turning nodes are generated (**outattr_track_nodegenesis**), or at which synapses are generated (**outattr_track_synaptogenesis**). We also opt to produce a textual file containing the full structural data of the generated network (**outattr_make_full_Txt**).

Part 29 – Graphical visualization parameters:

```
outattr_show_figure=false;
figattr_make_full_Fig=true;
figuresequence=false;
```

In this example no animated visualization will be used. Otherwise, use the following two parts of this manual:

Of the many possible visualizations of the resulting network, we choose only to produce an XFig compatible .fig file of the entire network (**figattr_make_full_Fig**). In the figure of the network structure, we choose to show the neuron somata (**figattr_neurons**), the axon fibers (**figattr_presynaptic**), the dendrite fibers (**figattr_postsynaptic**) and any synapses (**figattr_synapses**). We do not show abstract connections (**figattr_connections**) or spatial partitions created to search for candidate synaptic sites (**figattr_partitions**). We also choose to display a scale bar (**figattr_show_scale**) and axis arrows to indicate the orientation of the network visualization (**figattr_show_axis**). We draw the simulation time into the visualizations as a line of text (**figattr_progress**). Visualizations are prepared for presentation in color (**figattr_use_color**) with filled circles for the soma of each neuron (**figattr_fill_soma**). We depict terminal segments up to their growth cones instead of to the most recent branching or turning node (**figattr_tsupd_visibly**). Finally, we specify a color table for the elements of the visualization that differs from the default table (commands with the **CT_** prefix).

Part 30 – Animated visualization parameters:

In order to create an animated visualization of simulated network development, we opt to produce a sequence of figures at the chosen time intervals as set in part 26 of the script (**figuresequence**). The animation will contain an excerpt of the network environment with a specific center location and a specific distance from the center to each box wall of the excerpt volume (**sequence_zoom_center<x/y/z>** and **sequence_zoom_disttoedge**). The specified excerpt has a volume of about 10.648 cubic millimeters. Only fibers that are spatially located within the excerpt volume are shown (**figattr_fibres_nobox**), and this is done regardless whether the neuron soma to

which such fiber belongs lies within the volume (**figattr_box_fibre_independently**). Combination of the figure sequence into an animated file format is done externally (**combinesequence**). A geometric rotation is applied to each figure in the sequence, so that the network is rotated three times around the Z-axis (**autorotatesequence** and **ROT_interval_<x/y/z>**) from a specified initial angle of rotation around the Y-axis (**ROT_<x/y/z>**). The resulting animation is available at: http://netmorph.org/.

8. USER parameters

Change these parameters to modify the results of simulated morphogenesis. Table rows with a yellow background signify commands that choose models. The rows that follow them list associated model parameters.

8.1. Global simulation context

parameter label	units and format	default value	description	reference
seconds	(s), decimal	(see days)	Morphogenesis is simulated from 0.0 to this number of seconds. Note: Use only the **seconds** or **days** command. Seconds takes precedence over days.	t in Ref. [1]
days	(days), decimal	21	Morphogenesis is simulated from 0.0 to this number of days. Note: Use only the **seconds** or **days** command. Seconds takes precedence over days.	t in Ref. [1]
dt	(s), decimal	100	Fixed time step size between successive model calculations.	Δt in Ref. [1]
randomseed	integer	0	The "master" value used to seed pseudo random generators. Reuse a **randomseed** value to reproduce identical simulation results. The special value 0 implies that the system clock provides the seed value, producing essentially unpredictable simulation results.	
include	file path	none	Include commands stored in the specified file.	Chapter 2.3 of this manual.

Example script (as in **.morphogenesis.user.clp**):

```
days=21;
dt=100;
randomseed=0;
```

8.2. Neuronal populations and embedding space

parameter label	units and format	default value	Description
neurons	integer	9	The total number of neurons in the simulation.
populationsize<type>	integer		The number of neurons of a specified type (see <type> labels listed below table). Note: See for default value the description of the way in which NETMORPH determines population sizes below this table.
***L0**	(µm) decimal range	9, 11	Range of the minimum and maximum length of initial dendrite and axon segments at the onset of simulation.
shape	shape name	regions	Spatial constraints for the placement of somata. Here we describe only the specification of the regions shape and its parameters. For more options see Chapter 10.
regions	space separated list	pyrlayer	A list of region labels, each separated from the next by a space character.
<region.>shape	shape name	disc	Spatial constraints for the placement of somata within <**region.**>. Shape names are: **disc**, **box**, **sphere**.
<region.>neurons	integer	0	The number of neurons from the general pool to be placed in the <**region.**>.
<region.>minneuronseparation	(µm), decimal	75	A placement constraint specifying the minimum distance between the centers of neighboring somata.
<region.>center<X/Y/Z>	(µm), decimal	(0,0,0)	The center coordinates of the <**region.**>.
[region.]shape.radius	(µm), decimal	700	Radius of a disc or sphere shaped region.
[region.]shape.thickness	(µm), decimal	500	Thickness of a disc shaped region. Note that this parameter also allows a disc to have the appearance of a cylinder.
[region.]shape.<r.extent>	(µm), decimal	(see above)	Extent (length) of the box region shape in the <r.extent> dimension (see <r.extent> labels listed below table).

<type>: pyramidal, interneuron, bipolar, multipolar_nonpyramidal, principal_neuron†, untyped_neuron†
(† Neuron base classes that are not intended to be used in simulations.)
<r.extent>: width (i.e. x-axis), height (i.e. y-axis), depth (i.e. z-axis).

How NETMORPH determines population sizes:
Three types of commands control the population numbers of each kind of neuron in a simulation: **neurons**, **populationsize<type>** and **approxproportion<type>** (see Chapter 10). The total number of **neurons** is used in conjunction with the **approxproportion<type>** commands and is ignored when **populationsize<type>** commands are given.

When at least one **populationsize<type>** command is provided with a number greater than 0, then absolute population sizes are assumed to be specified. In that case, the sizes of populations not specified are assumed to be 0 and the total number of neurons in the simulation is derived from the sum of the absolute population sizes.

Otherwise, the total number of neurons is given by the **neurons** command or its default value 9.

Allocation of those neurons to populations of neuron types is then done according to approximate proportions, which can be specified in **approxproportion<type>** commands, as ratios that should add up to 1.0. Ratios not specified use default values. If nothing is specified, then the default allocation of 70% pyramidal neurons and 30% interneurons is used.

Example script:

```
neurons=100;
populationsizepyramidal=100;
L0=0,0;
```

8.3. Morphological development: general

parameter label	units and format	default value	description
fibreswithturns	flag	true	This flag selects whether turning models are applied to the dendrite and axon fibres throughout a simulation. The choice has a significant effect on computational resource requirements.

8.4. Morphological development: growth cone bifurcation

parameter label	units and format	default value	description
***branching_model**	model label	van_Pelt	The model used to determine the probability that branching occurred at any of the growth cones of an axon/dendrite arbor during the most recent simulation time interval. The following rows describe the parameters used by the van_Pelt model. For more options see Chapter 10.
***B_inf**	decimal	4.75	The asymptotic value of the expected number of branching events at an isolated segment.
***tau**	(s), decimal	319680	The exponential time coefficient of the function for the expected number of branching events at an isolated segment.
***E**	decimal	0.5	The parameter governing competition between growth cones for branching events.
***E_competes_with**	competition label	whole_neuron	The number of growth cones that are included in competition if E>0. Possible choices are: whole_neuron, all_axons, all_dendrites, same_arbor.

***TSBM**	model label	van_Pelt	The model used to determine the branching probability for a specific growth cone, which may involve local data, such as the centrifugal order of a growth cone or environmental influences in the vicinity of a growth cone. The following rows describe the parameters used by the **van_Pelt** and **van_Pelt_specBM** models. The van_Pelt_specBM terminal segment branching model behaves like the van_Pelt TSBM, but is intended to be specified as the TSBM of pyramidal apical dendrite tuft and oblique branches. While this TSBM also expects branching to be governed by arbor-wide constraints, it references a separate arbor branching model for which the model selection and parameter values are expected to be specified, preceded by **<prefix>.tuft.** or by **<prefix>.oblique.** labels. See also parts 22 & 24 of the example in Chapter 7 of this manual.
***S**	decimal	0	The parameter for the dependence of the branching probability on the centrifugal order of a specific growth cone. (This parameter is used by the **van_Pelt** and **van_Pelt_specBM** terminal segment branching models.)
***branch_angle_model**	model label	Balanced_ Forces	The model used to determine the branch angles when a bifurcation occurs at a specific growth cone. The following rows describe the parameters used by the Balanced_Forces model. For more options see Chapter 10.
***bam.bfbam.PDF**	PDF label	normal (mean = π/2, std=0.5, trunc = π-0.1)	A parameter of the Balanced_Forces model. A PDF that is used to determine the angle in the parallelogram of forces between daughter branches. This therefore also determines the angle between daughter branches, prior to perturbation according to ***bam.PDF**.
***bam.PDF**	PDF label	normal (mean=0, std=0.3, trunc=1)	Probability density function used for the random perturbation of the calculated balanced forces angles. Note: A delta PDF removes perturbation.

(Ref. [1])

Note about distributing the remainder of elongated fiber length after a bifurcation: To determine the random values X_1 and X_2 for the branch ratio, only a uniform PDF is currently implemented. When branching occurs, the actual branching location can be anywhere along the elongated fiber that was created during the last simulation interval. Fiber that protrudes beyond the determined branching location is redistributed to the new daughter branches. The ratios $X_1/(X_1+X_2)$ and $X_2/(X_1+X_2)$ determine the respective proportions of the remaining fiber given to daughter branch 1 and daughter branch 2.

8.5. Morphological development: growth cone elongation

parameter label	units and format	default value	Description
***arbor_elongation_model**	model label	van_Pelt (for universal, **all_axons, all_apical_pyramidal_dendrites**)	The model used to determine total elongation resources available to an axon/dendrite arbor at each time point. The following rows describe the parameters used by the van_Pelt model. Note: The arbor elongation model and all its parameters are ignored by growth cones that use the **nonnorm_BESTL** or **pyrAD_BESTLNN** terminal segment elongation models. For more options see Chapter 10.
***growth_F**	decimal	0.39 (universal) 0.16 (**all_axons**) 0.5 (**all_apical_pyramidal_dendrites**)	Elongation competition factor that determines the effect of the number of branches on the rate of elongation at each terminal segment. *F* in Ref. [1]
***growth_nu0**	(µm/s), decimal	0.00013889 (universal) 0.0005208333 (**all_axons**) 0.00013889 (**all_apical_pyramidal_dendrites**)	Elongation speed/rate. If F=0, then this is the rate applied to each terminal segment. If F=1, then it is the rate for the elongation of a whole dendrite/axon arbor. v_0 in Ref. [1]
***F_competes_with**	competition label	same_arbor (universal, **all_axons, all_apical_pyramidal_dendrites**)	The number of terminal segments that are counted for competition if F>0. Possible choices are: whole_neuron, all_axons, all_dendrites, same_arbor. $n(t)$ in Ref. [1]
***aem.PDF**	PDF label	normal (universal, **all_axons, all_apical_pyramidal_dendrites**)	Probability density function used for the random perturbation of the computed arbor elongation. PDF labels and PDF specific parameters are described below this table. Note: A delta PDF removes perturbation.
***terminal_segment_elongation_model**	model label	BESTL (universal)	The model used to determine the ratio of elongation resources allocated to specific terminal segments at each time point. The following rows describe the parameters used by the **BESTL**, **nonnorm_BESTL** and **pyrAD_BESTLNN** models. The nonnorm_BESTL model does not apply normalization to elongation speeds with regard to other growth cones. Note: Growth cones that use the nonnorm_BESTL or pyrAD_BESTLNN models ignore arbor elongation models and their parameters, in effect implying independent elongation *(***growth_F=0**)*! The pyrAD_BESTLNN model behaves like the nonnorm_BESTL model, but is explicitly designed to enable the simulation of pyramidal apical dendrite development.

			For more options see Chapter 10.
***tsem.branch.PDF**	(µm), PDF label	normal (mean=2.0, std=1.0)	Probability density function used for the random selection of initial length at BESTL branches.
***tsem.PDF**	PDF label	delta (universal)	Probability density function used for the random perturbation of the expected quota. Note: A delta PDF removes perturbation. Beware: The perturbation is added to 1.0 before multiplication with the calculated elongation speed, so that a delta PDF with a value greater than zero can cause a ramp up to infinite speed with unpredictable results!
***tsem.trunklength.PDF**	(µm), PDF label	normal (mean=700.0, std=100.0)	Probability density function used to select the trunk length for an apical dendrite in terms of the expected distance from the soma center at which the apical dendrite begins to tuft. (A parameter used by the **pyrAD_BESTLNN** model)
***tsem.obliques.PDF**	PDF label	normal (mean=7, std=3)	Probability density function used to select the number of oblique branches of an apical dendrite. (A parameter used by the **pyrAD_BESTLNN** model)
***tsem.prefix**	prefix label	pyrAP	A label that is used as a prefix to identify models and parameters specified for use in the *tuft* or *oblique branches* of an apical dendrite. (A parameter used by the **pyrAD_BESTLNN** model)
<prefix>.oblique.<model/parameter command>	miscellaneous	models and parameter values of the *universal network* region *all_pyramidal_dendrites* set	The prefix specified by ***tsem.prefix** and the **.oblique.** keyword precede a collection of model and parameter specifications for use in oblique branches of an apical dendrite. The models for which model choices and parameter values can be specified in this manner are: terminal segment elongation model, elongation rate initialization model, terminal segment branching model, branch angle model, direction model. (A parameter used by the **pyrAD_BESTLNN** model. Also note the **van_Pelt_specBM** model described as an option for ***TSBM**.) See parts 23 & 24 of the example in Chapter 7 of this manual.
<prefix>.tuft.<model/parameter command>	miscellaneous	models and parameter values of the *universal network* region *all_pyramidal_dendrites* set	The prefix specified by ***tsem.prefix** and the **.tuft.** keyword precede a collection of model and parameter specifications for use in the tuft of an apical dendrite. The models for which model choices and parameter values can be specified in this manner are: terminal segment elongation model, elongation rate initialization model, terminal segment branching model, branch angle model, direction model. (A parameter used by the **pyrAD_BESTLNN** model. Also note the **van_Pelt_specBM** model described as an option for ***TSBM**.) See parts 21 & 22 of the example in Chapter 7 of this manual.
***elongation_rate_ini**	model label	length_distribution	The model used to determine the initial ratio of

tialization_model		(universal)	elongation resources expected after a bifurcation. The following rows describe the parameters used by the length_distribution model. For more options see Chapter 10.
***eri.PDF**	PDF label	normal (mean=0, std=1.0, trunc=3.0)	Probability density function used to draw random numbers for new relative elongation rates of the two daughter branches. Note: A delta PDF leads to identical ratios.

(Ref. [1])

PDF modifying parameters:

delta: **<label>.PDF.value**
uniform: (none)
linear: **<label>.PDF.max_x**, **<label>.PDF.height_b**
spline_normal: **<label>.PDF.max_x**, **<label>.PDF.significance_threshold**, **<label>.PDF.proportion_significant**
spline_normal_with_min: **<label>.PDF.max_x**, **<label>.PDF.significance_threshold**, **<label>.PDF.proportion_significant**, **<label>.PDF.min_x**
normal: **<label>.PDF.mean**, **<label>.PDF.std**, **<label>.PDF.trunc**
exponential: (none)

Note about elongation rate initialization by length distribution: Two random values X_1 and X_2 are drawn according to **eri.PDF**. Both are converted. If X<0 then X=2/(1+exp(-X)), a sigmoidal remapping of negative values into the range from 0 to 1. If X>=0 then X=(X/2)+1, an optimization of positive values from a standard normal distribution. An example of the conversion is shown in Figs. 4, 5 for values drawn from a standard normal distribution. The resulting values are multiplied with the mean resource quota requested by growth cones at other elongating terminal segments, so that a value of 0 becomes the mean quota. If the standard deviation of **eri.PDF** is changed, then variability from the mean quota is modified. If the mean of **eri.PDF** is less than 0 then new daughter branches tend to elongate more slowly than existing terminal segments. If the mean of **eri.PDF** is greater than 0 then new daughter branches tend to elongate more rapidly than existing terminal segments. Finally, the greater of the two calculated relative elongation quotas is given to the daughter branch that received the greater proportion of remaining elongated fiber after a bifurcation (see note below).

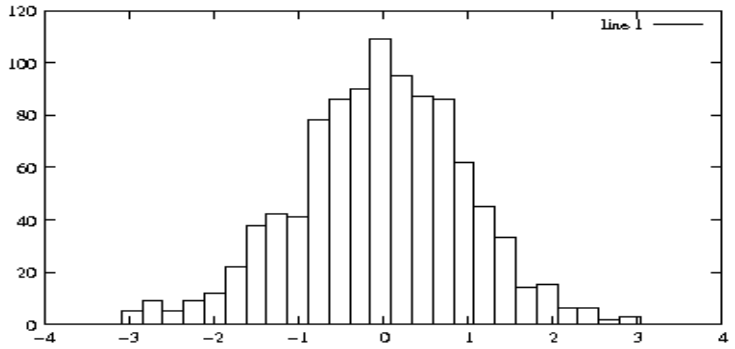


Fig. 4. Standard normal distribution of random values used to determine X_1 and X_2.

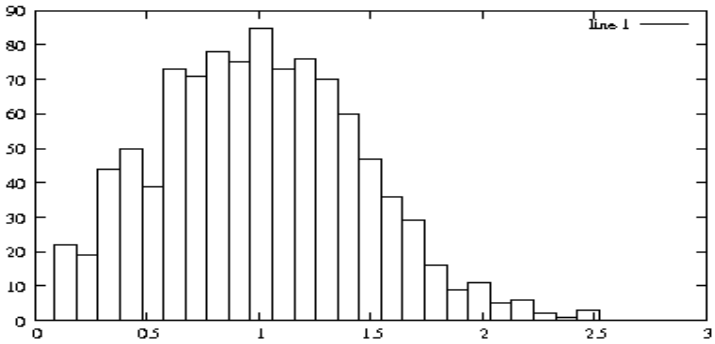


Fig. 5. Constrained to positive values to be multiplied with the mean relative elongation rate of older growth cones. Half of the random values are mapped to the interval from 0 to 1.

Example script (as in **.morphogenesis.user.clp**):

```
arbor_elongation_model=van_Pelt;
growth_F=0.39;
growth_nu0=0.00013889;
F_competes_with=same_arbor;
all_axons.arbor_elongation_model=van_Pelt;
all_axons.growth_F=0.16;
all_axons.growth_nu0=0.0005208333;
all_axons.F_competes_with=same_arbor;
all_apical_pyramidal_dendrites.arbor_elongation_model=van_Pelt;
all_apical_pyramidal_dendrites.growth_F=0.5;
all_apical_pyramidal_dendrites.growth_nu0=0.00013889;
all_apical_pyramidal_dendrites.F_competes_with=same_arbor;
terminal_segment_elongation_model=BESTL;
tsem.PDF=delta;
elongation_rate_initialization_model=length_distribution;
```

8.6. Morphological development: growth cone direction

parameter label	units and format	default value	Description
***TSTM**	model label	linear_rate	The model that is used to determine the probability of a turn, a change in the direction of elongation at growth cones. The following rows describe the parameters used by the **linear_rate** model. For more options see Chapter 10.
***turn_rate**	(1/µm), decimal	0.1	A **linear_rate** TSTM parameter. Mean number of turns per µm. (Note: When compiled with the ENABLE_FIXED_STEP_SIMULATION directive, then the units are per µm, otherwise the units are per second.
***turn_separation**	(µm), decimal	10	A **linear_rate** TSTM parameter. Mean length of neurite fiber segment pieces between consecutive turns. (Note: When compiled with the

			ENABLE_FIXED_STEP_SIMULATION directive, then the units are µm, otherwise the units are seconds.
***direction_model**	model label	segment_history_tension	The model that is used to determine the new direction of growth after a turn event at a growth cone. The following rows describe the parameters used by the segment_history_tension model. For more options see Chapter 10.
***veeranglemin**	(radians), decimal	π/16	The expected pitch of the new direction computed by direction models is probabilistically perturbed between these minimum and maximum veer angles.
***veeranglemax**		π/4	
***history_power**	decimal	2	This parameter is used by the **segment_history_tension** direction model. It is the power exponent of the distance denominator that determines the influence that neurite segment pieces in the history of a neurite from the last bifurcation on have on the predicted direction of growth. A value equal to 0 means that all pieces have the same influence irrespective of distance (though modulated by the lengths of the pieces). A value greater than 0 means that influence decreases with distance, while a value smaller than 0 means that influence increases with distance.

(Ref. [1])

8.7. Morphological development: neurite fiber diameter

parameter label	units and format	default value	description
neurite_diameter_model	model label	none	The model that is used to determine the diameter of axon and dendrite fiber. The following rows describe the parameters used by the **Rall** model. For more options see Chapter 10.
ndm.e_power.PDF	PDF label	normal (mean=1.47, std=0.3)	The PDF used to select a new power law exponent at each bifurcation point. (This is a parameter of the **Rall** model.)
ndm.d_term.PDF	PDF label	normal (mean=0.7, std=0.3)	The PDF used to select a neurite diameter at the growth cone of each terminal segment. (This is a parameter of the **Rall** model.)

8.8. Connectivity development: synapse formation

parameter label	units and format	default value	Description
synapse_formation.PDF	PDF label	uniform	The PDF used to determine if an actual synapse is formed

			at a candidate site. The drawn value is compared with a computed likelihood threshold. If the value is smaller than the threshold then a synapse is formed. Note: A delta PDF with value 0 can be used to make synapse formation at a candidate site a certainty.
D_synmax.<pre>.<post>	(µm), decimal	1 (pyramidal to pyramidal)	The maximum distance between fibers at a synapse between pre- and postsynaptic neuron types. The **<pre>** and **<post>** placeholders can be any of: **principal, interneuron, multipolar, bipolar, pyramidal, untyped**.

(Ref. [1]; Method of synapse formation as described in Ref. [6])

8.9. Simulation output: network data

parameter label	units and format	default value	description
outattr_make_full_Txt	flag	false	Create .txt files that describe details of the generated network structure. Note: The file format is described in Chapter 11 of this manual.
outattr_Txt_sequence	flag	false	Create a sequence of .txt files that describe details of the generated network structure at successive sample time points.
outattr_Txt_separate_files	flag	false	Store the detail data in separate files for each type of data.

9. Advanced invocation of NETMORPH simulations

In addition to the standard invocations described in Chapters 2 and 3, NETMORPH simulations can be specified, controlled and run through a web interface and a local or remote server. NETMORPH searches for commands in web form data (provided by a GET method QUERY_STRING or by POST method standard input). The resulting complete list of simulation command sources and their parsing order is given in Table 3. A detailed description of that interface method is beyond the scope of this manual.

Table 3. Complete list of sources that provide model choices and parameter values for simulations with NETMORPH.

source	description
compiled defaults	These are the default choices and values specified in the NETMORPH executable program file.
.nibrrc	This resource file is automatically read by the 3D version of NETMORPH (netmorph).
.nibr2Drc	This resource file is automatically read by the 2D version of NETMORPH (netmorph2D).
the command line	Program input provided by instructions that follow the program name on the command line.
web form input	Program input provided by GET and POST methods via a web interface (e.g. HTML form input). Not presently used.
included CLP scripts	Command line parameters (CLP) stored in a text file and separated by semicolons (";"), included by the include command.

10. ADVANCED parameters

The following parameters are used in simulations that required additional capabilities of NETMORPH that exceed the requirements of basic morphogenesis simulations, such as those shown in Ref. [1].

10.1. Global simulation context

parameter label	units and format	default value	description
events_seconds	(s), decimal	(see events_append_seconds)	Events (e.g. neuronal spiking) are simulated from 0.0 to this many seconds.
events_append_seconds	(s), decimal	0.0	Events are simulated from 0.0 to seconds+this or (days*86400)+this many seconds.
maxrandomspikeinterval	(s), decimal	100	Maximum number of seconds between randomly allocated spontaneous spiking events.

Example script (as in **.morphogenesis.system.clp**):

```
events_append_seconds=0;
```

10.2. Complete simulated network

parameter label	units and format	default value	description
approxproportion<type	decimal	(see	The approximate proportion of the neurons that

>	[0,1]	populationsize<type>)	are of a specific type.
random_orientation	(radians), true/decimal	(see use_specified_basal_direction)	Should the neuronal axis through the axon be randomly oriented at initialization? If not true, then by how many radians can the axis be randomly perturbed around the fixed direction? Note: If use_specified_basal_direction is true, then this parameter is only used if a decimal value is given, which then provides a maximum deviation of the apical dendrite.
use_specified_basal_direction	flag	false	Specify the general direction in which the center of mass of all initial basal dendrite segments is oriented.
specified_basal_direction_<angle>	(radians), decimal	(see specified_basal_direction_<axis>	The angular direction in spherical (3D) or polar (2D) coordinates (see <angle> labels listed below table).
specified_basal_direction_<axis>	decimal	(0, 1, 0)	A relative direction vector (see <axis> labels listed below table).
apical_specified_overrides_centroid	flag	false	Ignore the centroid of the allocated basal dendrites and instead align an apical dendrite with a negated vector along the direction specified with specified_basal_direction_<angle> or specified_basal_direction_<axis>. This command has no effect if use_specified_basal_direction is false.

<angle>: theta, the "roll" angle (here around the y-axis) in the interval; phi, the "pitch" angle (with the y-axis). <axis>: relx, rely, relz displacement relative to the x,y and z axes.

Example script (as in **.morphogenesis.user.clp**):

```
use_specified_basal_direction=true;
specified_basal_direction_relx=0;
specified_basal_direction_rely=1;
specified_basal_direction_relz=0;
```

10.3. Network placement constraints

parameter label	units and format	default value	description
electrodes	true/false	false	Include a hexagonal grid of electrode nodes in the x-y plane that resembles a typical multi electrode array (MEA) used with cultures of dissociated neurons. In the current version of NETMORPH this is an optional visual reference that can be included in graphical representations of a simulation.
pia_attraction_repulsion_hy	true/false	true	Assumes that apical dendrites are attracted

pothesis			into the direction of the pia and that when there is an apical dendrite, then the axon emerges from the opposing location on the soma.
<neuron-type>.min_basal	integer	2/2/2/1/4/2	Minimum number of basal dendrites at each pole of a neuron.
<neuron-type>.max_basal	integer	5/4/5/1/8/5	Maximum number of basal dendrites at each pole of a neuron.
<neuron-type>.basal.minangle	(radians), decimal	0.1	Minimum angle between the somatic location of a basal dendrite and the polar center.
<neuron-type>.basal.maxangle	(radians), decimal	½π/2π/2π/¼π/½π/½π	Maximum angle between the somatic location of a basal dendrite and the polar center.
<neuron-type>.basal.force_model	model label	unrestricted	The model that determines how basal dendrite root nodes are distributed within an available surface area on a soma. Options are: **unrestricted**, **surface_division**.
multipolar.max_axons	integer	1	Maximum number of axons of a multipolar neuron.
[label.]environment_physics	space separated list	(none)	A set of specified physical constraints to be applied to a simulation, each with a corresponding ID label.
<label>.physical_boundary	boundary type name	(none)	Boundary type: spherical (that a neuron can be inside of or outside of), point_attractor.
<label>.spherical_center[X/Y/Z]	(µm), decimal/vector	(0,0,0)	Coordinates of the center location of a spherical boundary.
<label>.spherical_radius	(µm), decimal	0	Radius of a spherical boundary.
<label>.spherical_maxrange	(µm), decimal	0	On the inside of a spherical boundary, the maximum distance from that boundary at which boundary effects act.
<label>.spherical_c_repulse	decimal	0.0	Proportion of the approach speed/rate that contributes to the repulsive force of a spherical boundary.
<label>.attractor_point[X/Y/Z]	(µm), decimal/vector	(0,0,0)	Coordinates of an attractor_point in the environmental physics.
<label>.attractor_maxrange	(µm), decimal	0	Maximum distance from an attractor_point at which the attraction effects act.
<label>.attractor_c_attract	decimal	0.0	Coefficient of the attractive force of an attractor_point in the environmental physics.
shape	shape name	regions	Spatial constraints for the placement of somata. Some spatial shape constraints are only available in 3D simulations. Shape names are: regions (3D), box (3D), circle, rectangle, hexagon. Non-region shapes that place neurons in a regular grid can affect the number of neurons that are actually included in a simulation, since the number is reduced as necessary to best allocate neurons to a regular grid. Note: The default shape for 2D simulations is circle.

			Also note: In 3D, the use of shapes that are not regions is deprecated and the neurons in such a shape are automatically allocated to a logical region called "net".
shape_<extent>	(µm), decimal	(see **regions**)	Extent (length) of the shape in the <extent> dimension (see <extent> labels listed below table).
shape_sidelength	(µm), decimal	(see **regions**)	Length of each side of a hexagon shape.
shape_radius	(µm), decimal	(see **regions**)	Radius of a circle shape.
shape_randompolar	true/false	false	Randomize polar coordinates during 2D placement.
[region.]<neuron-type>	integer	0	The number of neurons of the specific type <**neuron-type**> to be added to any general pool neurons that may already belong to the <**region.**>.

<extent>: horizontal (i.e. x-axis), vertical (i.e. y-axis), depth (i.e. z-axis).
<r.extent>: width (i.e. x-axis), height (i.e. y-axis), depth (i.e. z-axis).
<neuron-type>: principal, interneuron, multipolar, bipolar, pyramidal, untyped.

Example script (as in **.morphogenesis.user.clp**):

```
electrodes=false;
pyramidal.min_basal=4;
pyramidal.max_basal=8;
pyramidal.basal_minangle=0;
pyramidal.basal_maxangle=1.5;
shape=regions;
regions=pyrlayr;
pyrlayr.shape=disc;
pyrlayr.neurons=100;
pyrlayr.minneuronseparation=75;
pyrlayr.centerX=0;
pyrlayr.centerY=0;
pyrlayr.centerZ=0;
pyrlayr.shape.radius=700;
pyrlayr.shape.thickness=500;
```

10.4. Morphological development: general

parameter label	units and format	default value	description
branchatinitlength	flag	false	Specify if the first branching must occur at arbor lengths determined by L0 (see ***L0** parameter).
Dbranchesatturns	flag	false	Allow turning points in existing dendrite fiber to be converted into new branch points.
Abranchesatturns	flag	false	Allow turning points in existing axon fiber to be converted

			into new branch points.
branchinsegment	flag	true	Upon bifurcation the branch point is determined to lie somewhere within the most recent strentch of elongated fiber, grown during the preceding interval from t-Δt to t.

(Ref. [1])

10.5. Morphological development: growth cone elongation

parameter label	units and format	default value	description
***arbor_elongation_model**	model label	van_Pelt (for universal, **all_axons, all_apical_pyramidal_dendrites**)	The model used to determine total elongation resources available to an axon/dendrite arbor at each time point. Implemented models and the parameters they use are: van_Pelt **growth_F, growth_nu0, F_competes_with, aem_PDF, aem_weight, aem_label** polynomial_O1 **growth_F, growth_nu0, growth_nu1, F_competes_with, aem_PDF, aem_weight, aem_label** polynomial_O3 **growth_F, growth_nu0, growth_nu1, growth_nu2, F_competes_with, aem_PDF, aem_weight, aem_label**
***growth_F**	decimal	0.39 (universal) 0.16 (all_axons) 0.5 (all_apical_pyramidal_dendrites)	Elongation competition factor that determines the effect of the number of branches on the rate of elongation at each terminal segment. This is used by van_Pelt, polynomial_O1 and polynomial_O3 models.
***growth_nu0**	(μm/s), decimal	0.00013889 (universal) 0.0005208333 (all_axons) 0.00013889 (all_apical_pyramidal_dendrites)	Elongation speed/rate. If F=0, then this is the rate applied to each terminal segment. If F=1, then it is the rate for the elongation of a whole dendrite/axon arbor. This is used by van_Pelt, polynomial_O1 and polynomial_O3 models.
***F_competes_with**	competition label	same_arbor (universal, all_axons, all_apical_pyramidal_dendrites)	The number of terminal segments that are counted for competition if F>0. Possible choices are: whole_neuron, all_axons, all_dendrites, same_arbor. This is used by the van_Pelt model.
***growth_nu1**	($\mu m^2/s^2$), decimal	(use van_Pelt)	Elongation acceleration. This is used by the polynomial_O3 model.
***growth_nu2**	($\mu m^3/s^3$), decimal	(use van_Pelt)	The change of elongation acceleration. This is used by the polynomial_O3 model. The [set.] possibilities include the natural sets (see <schema.> below this table).
***aem.PDF**	PDF label	normal (universal, all_axons, all_apical_pyramidal_d	Probability density function used for the random perturbation of the computed arbor elongation. PDF labels are: delta, uniform, linear,

		endrites)	spline_normal, spline_normal_with_min, normal, exponential. Note: A delta PDF removes perturbation. Each PDF has a specific set of modifying parameters (e.g., mean, std), which are listed below this table.
***aem_weight**	decimal	(none)	Weight given to the model output produced by a [**contributing.**] model in a model chain. Weights are relative to the influence of the model at the head of the chain, which has the implicit weight value 1.0.
***aem_label** or ***arbor_elongation_model_label**	arbitrary label	(none)	Specify a label in order to add another elongation model to the chain of elongation models that is used to compute arbor elongation. When chaining, also specify an **aem_weight** at this level. At the next level, the level of the new contributing model, the existing [**contributing.**] specifier is extended with the chosen label, in order to access model parameters specific to that model. For example, a 2 model chain, in which the second model is accessed through the "target_layer_attraction." prefix might be expanded to a 3 model chain with the full prefix "target_layer_attraction.following_filaments".
***terminal_segment_elongation_model**	model label	BESTL (universal)	The model used to determine the ratio of elongation resources allocated to specific terminal segments at each time point. Implemented models are: simple, inertia, second_order, constrained_second_order, decaying_second_order, initialized_cso, BESTL, nonnorm_BESTL.
***tsem_inertia**	decimal	(use BESTL)	First order (speed) inertia in the second_order and decaying_second_order TSEM.
***tsem_decay**	(s), decimal	(use BESTL)	Decay time constant in the decaying_second_order TSEM.
***tsem_initial_acceleration**	decimal	(use BESTL)	Initial acceleration of the ratio/quota of elongation a terminal segment requests initialized_cso TSEM.
***tsem.PDF**	PDF label	delta (universal)	Probability density function used for the random perturbation of the expected quota. PDF labels are: delta, uniform, linear, spline_normal, spline_normal_with_min, normal, exponential. Note: A delta PDF removes perturbation. Each PDF has a specific set of modifying parameters (e.g. mean, std), which are listed below this table.
***aem_weight**	decimal	(none)	Weight given to the model output produced by a [**contributing.**] model in a model chain. Weights are relative to the influence of the model at the head of the chain, which has the implicit weight value 1.0.
***tsem_label** or ***terminal_segment_elongation_model**	arbitrary label	(none)	Specify a label in order to add another terminal segment elongation model to the chain of TSEMs that is used to compute terminal segment

_label			elongation ratios/quotas. When chaining, also specify a **tsem_weight** at this level.
***elongation_rate_initialization_model**	model label	length_distribution (universal)	The model used to determine the initial ratio of elongation resources expected after a bifurcation. Implemented models are: length_distribution, pure_stochastic, zero, unitary, continue_defaults. Note: There is a dependence between the initial rate determined by the distribution of remaining fiber lengths after a bifurcation point, and the two daughter branch elongation ratios, as computed by the BESTL terminal segment elongation model.
***eri_initialquota**	decimal	(use length_distribution)	Elongation quota to initialize each terminal segment to in the unitary ERI model.
***eri.PDF**	PDF label	(none)	Probability density function used for the random perturbation of the initial ratio computed. PDF labels are: delta, uniform, linear, spline_normal, spline_normal_with_min, normal, exponential. Note: A delta PDF removes perturbation. Each PDF has a specific set of modifying parameters (e.g. mean, std), which are listed below this table.
***eri_weight**	decimal	(none)	Weight given to the model output produced by a [contributing.] model in a model chain. Weights are relative to the influence of the model at the head of the chain, which has the implicit weight value 1.0.
***eri_label** or ***elongation_rate_initialization_model_label**	arbitrary label	(none)	Specify a label in order to add another elongation rate initialization model to the chain of ERI models that is used to compute initial elongation ratios after a bifurcation. When chaining, also specify an **eri_weight** at this level.

natural sets of <schema.>:
(empty universal set), all_axons., all_dendrites., all_pyramidal_axons.,
all_pyramidal_dendrites., all_interneuron_axons.,
all_interneuron_dendrites., all_apical_pyramidal_dendrites.
PDF modifying parameters:
delta: **<label>.PDF.value**
uniform: (none)
linear: **<label>.PDF.max_x**, **<label>.PDF.height_b**
spline_normal: **<label>.PDF.max_x**, **<label>.PDF.significance_threshold**, **<label>.PDF.proportion_significant**
spline_normal_with_min: **<label>.PDF.max_x**, **<label>.PDF.significance_threshold**, **<label>.PDF.proportion_significant**, **<label>.PDF.min_x**
normal: **<label>.PDF.mean**, **<label>.PDF.std**, **<label>.PDF.trunc**
exponential: (none)

Example script (as in **.morphogenesis.user.clp**):

```
arbor_elongation_model=van_Pelt;
growth_F=0.39;
growth_nu0=0.00013889;
F_competes_with=same_arbor;
all_axons.arbor_elongation_model=van_Pelt;
all_axons.growth_F=0.16;
all_axons.growth_nu0=0.0005208333;
all_axons.F_competes_with=same_arbor;
all_apical_pyramidal_dendrites.arbor_elongation_model=van_Pelt;
all_apical_pyramidal_dendrites.growth_F=0.5;
all_apical_pyramidal_dendrites.growth_nu0=0.00013889;
all_apical_pyramidal_dendrites.F_competes_with=same_arbor;
terminal_segment_elongation_model=BESTL;
tsem.PDF=delta;
elongation_rate_initialization_model=length_distribution;
```

10.6. Morphological development: growth cone bifurcation

parameter label	units and format	defaultvalue	description
***branching_model**	model label	van_Pelt	The model used to determine the probability that branching occurred at any of the growth cones of an axon/dendrite arbor during the most recent simulation time interval. Implemented models and the parameters they use are: van_Pelt **min_node_interval, B_inf, tau, E, E_competes_with, bm_weight, bm_label** polynomial_O1 **min_node_interval** polynomial_O3 **min_node_interval**
***min_node_interval**	(µm), decimal	0	The minimum length of pieces of axon or dendrite fiber that constrains how close together turning and branching points can be.
***B_inf**	decimal	4.75	The asymptotic value of the expected number of branching events at an isolated segment.
***tau**	(s), decimal	319680	The exponential time coefficient.
***E**	decimal	0.5	The parameter governing competition between growth cones for branching events.
***E_competes_with**	competition label	whole_neuron	The number of growth cones that are included in competition if E>0. Possible choices are: whole_neuron, all_axons, all_dendrites, same_arbor.
***bm_weight**	decimal	(none)	Weight given to the model output produced by a [contributing.] model in a model chain. Weights are

			relative to the influence of the model at the head of the chain, which has the implicit weight value 1.0.
***bm_label**	arbitrary label	(none)	
***TSBM**	model label	van_Pelt	The model used to determine the branching probability for a specific growth cone, which may involve local data, such as the cetrifugal order of a growth cone or environmental influences in the vicinity of a growth cone. The following rows describe the parameters used by the van_Pelt model.
***S**	decimal	0	The parameter for the dependence of the branching probability on the cetrifugal order of a specific growth cone. (This parameter is used by the **van_Pelt** and **van_Pelt_specBM** terminal segment branching models.)
***tsbm_weight**	decimal	(none)	Weight given to the model output produced by a [contributing.] model in a model chain. Weights are relative to the influence of the model at the head of the chain, which has the implicit weight value 1.0.
***tsbm_label**	arbitrary label	(none)	
***branch_angle_model**	model label	Balanced_Forces	The model used to determine the branch angles when a bifurcation occurs at a specific growth cone.
***bam.bfbam.PDF**	PDF label	normal (mean = π/2, std=0.5, trunc = π-0.1)	A parameter of the Balanced_Forces model. A PDF that is used to determine the angle in the parallelogram of forces between daughter branches. This therefore also determines the angle between daughter branches, prior to perturbation according to ***bam.PDF**.
***bam.PDF**	PDF label	normal (mean=0, std=0.3, trunc=1)	Probability density function used for the random perturbation of the calculated balanced forces angles. Note: A delta PDF removes perturbation.
***bam_weight**	decimal	(none)	Weight given to the model output produced by a [contributing.] model in a model chain. Weights are relative to the influence of the model at the head of the chain, which has the implicit weight value 1.0.
***bam_label**	arbitrary label	(none)	

(Ref. [1])

10.7. Morphological development: growth cone direction

parameter label	units and format	default value	description
***TSTM**	model label	linear_rate	The model that is used to determine the probability of a turn, a change in the direction of elongation at growth cones. The following rows describe the parameters used by the **none**, **each_dt**, **linear_rate** and **branch_coupled** models.

***turn_rate**	(1/µm), decimal	0.1	A **linear_rate** TSTM parameter. Mean number of turns per µm. (Note: When compiled with the ENABLE_FIXED_STEP_SIMULATION directive, then the units are per µm, otherwise the units are per second.
***turn_separation**	(µm), decimal	10	A **linear_rate** TSTM parameter. Mean length of neurite fiber segment pieces between consecutive turns. (Note: When compiled with the ENABLE_FIXED_STEP_SIMULATION directive, then the units are µm, otherwise the units are seconds.
***tf**	decimal	9.25	A **branch_coupled** TSTM parameter. The "turn factor" by which the branching probability computed by a TSBM is multiplied to determine the probability that a turn even occurs.
***tstm_weight**	decimal	(none)	Weight given to the model output produced by a [**contributing.**] model in a model chain. Weights are relative to the influence of the model at the head of the chain, which has the implicit weight value 1.0.
***tstm_label** or ***TSTM_label**	arbitrary label	(none)	Specify a label in order to add another turning model to the chain of TSTMs that is used to compute terminal segment turn events. When chaining, also specify a **tstm_weight** at this level. At the next level, the level of the new contributing model, the existing [**contributing.**] specifier is extended with the chosen label in order to access model parameters specific to that model. For example, a 2 model chain, in which the second model is accessed through the "obstacle_detection." prefix might be expanded to a 3 model chain with the full prefix "obstacle_detection.EF_detection".
***direction_model**	model label	segment_history_	The model that is used to determine the new direction of growth after a turn event at a growth cone. Implemented models are: segment_history_tension: cell_attraction: radial: vector:
***veeranglemin**	(radians), decimal	tension π/16	The expected pitch of the new direction computed by direction models is probabilistically perturbed between these minimum and maximum veer angles.
Veeranglemax		π/4	
***dm_weight**	decimal	(none)	
***dm_label**	arbitrary label	(none)	
***Samsonovich_hypothesis**	flag	false	This parameter is used by the **radial** direction model. If false then the expected direction is a perturbation of the current direction of growth. If true then the expected direction is radial, i.e., along a normal vector from the soma center to the growth cone, with angular perturbation.

			See also Samsonovich and Ascoli (2003; Ref. [4]).
***history_power**	decimal	2	This parameter is used by the **segment_history_tension** direction model (Ref. [1]). It is the power exponent of the distance denominator that determines the influence that neurite segment pieces in the history of a neurite have on the predicted direction of growth. A value equal to 0 means that all pieces have the same influence irrespective of distance (though modulated by the lengths of the pieces). A value greater than 0 means that influence decreases with distance, while a value smaller than 0 means that influence increases with distance.
***direction**	vector	(0,0,0)	This parameter is used by the **vector** direction model. A specific direction vector is specified and used as a constant expected direction.
dirhistory_selection	model label	none	A model of changes in the fiber segment history that is taken into account in the **segment_history_tension** model. Implemented models are: none: random_truncation:
general_rtdhs_probability	decimal		The probability that the history of segment pieces is truncated.
general_rtdhs_minfraction	decimal		The minimum fraction maintained after truncation.
general_rtdhs_maxfraction	decimal		The maximum fraction maintained after truncation.
turnanglemin	(radians), decimal	π/16	Changing these minimum and maximum constraints modifies the default values for all ***veeranglemin** and ***veeranglemax.**
turnanglemax		π/4	

(Ref. [1])

10.8. Morphological development: neurite fiber diameter

parameter label	units and format	default value	description
neurite_diameter_model	model label		Options: none: diameter is fixed. asymptotic: **ndm.asymptotic_proportion**, **ndm.asymptotic_lambda** rall: **ndm.e_power.PDF,** **ndm.d_term.PDF**
ndm.asymptotic_proportion	decimal		
ndm.asymptotic_lam	decimal		

bda			
ndm.e_power.PDF	PDF label	normal (mean=1.47, std=0.3)	With the Rall model, the PDF used to draw a power law exponent at each bifurcation point.
ndm.d_term.PDF	PDF label	normal (mean=0.7, std=0.3)	With the Rall model, the PDF used to draw a neurite diameter at each terminal segment.

(Ref. [1])

10.9. Connectivity development: synapse formation

parameter label	units and format	default value	description
candidate_synapses	flag	true	If true then candidate synapses are sought and synapse formation is simulated.
no_autapses	flag	true	If true than no synaptic connections can be generated from a neuron to itself.
partition_to_synapses	flag		
synapses_during_development	flag	true	
partitionsubsetsize	integer		
max_spatial_segment_subsets	integer		
min_spatial_segment_span	(µm), decimal		
max_spatial_segment_coordinate	(µm), decimal		
synapse_formation.PDF	PDF label	uniform	The PDF used to determine if an actual synapse is formed at a candidate site. The drawn value is compared with a computed likelihood threshold. If the value is smaller than the threshold then a synapse is formed. Note: A delta PDF with value 0 can be used to make synapse formation at a candidate site a certainty.
D_synmax.<pre>.<post>	(µm), decimal	1 (pyramidal to pyramidal)	The maximum distance between fibers at a synapse between pre- and postsynaptic neuron types. The **<pre>** and **<post>** placeholders can be any of: **principal, interneuron, multipolar, bipolar, pyramidal, untyped**.
P_receptor.<pre>.<post>.<receptor_type>	decimal		This table of probability values determines the likelihood that synapses of a specific receptor type are formed at synapses between specific pre- and postsynaptic neuron types. The **<pre>** and **<post>** placeholders can be any of: **principal, interneuron, multipolar, bipolar, pyramidal, untyped**. The **<receptor_type>** placeholder can be any of: **AMPAR, NMDAR, GABAR**.

			Default values (short-hand notations): All values equal to -9.9 except: P_AMPAR(pr –> pr, in, mu, bi, py)=1.0 P_AMPAR(mu-> pr, in, mu, bi, py)=1.0 P_AMPAR(bi –> pr, in, mu, bi, py)=1.0 P_AMPAR(py –> pr, in, mu, bi, py)=1.0 P_NMDAR(pr -> py; mu -> py; bi -> py)=0.4 P_NMDAR(py -> py)=0.9 P_GABAR(in –> pr, in, mu, bi, py)=1.0 Example: change via P_receptor.bipolar.pyramidal.GABAR=1.0

(Ref. [1]; Method of synapse formation as described in Ref. [6])

10.10. Simulation output: network data

parameter label	units and format	default value	description
outattr_make_full_Txt	flag	false	Create .txt files that describe details of the generated network structure.
outattr_Txt_sequence	flag	false	Create a sequence of .txt files that describe details of the generated network structure at successive sample time points.
outattr_Txt_separate_files	flag	false	Store the detail data in separate files for each type of data.
outattr_track_synaptogenesis	flag	false	Include creation times of synapses in the detail data.
outattr_track_nodegenesis	flag	false	Include creation times of branching and turning points in the detail data.
outattr_make_full_X3D	flag	false	Create an X3D (standard 3D visualization) file that shows the generated network structure.
outattr_make_full_Catacomb	flag	false	Create a Catacomb (.ccm) script file that represents the abstracted network connectivity in an integrate-and-fire neural network.
outattr_synapse_distance_frequency	flag	false	Create an Octave/Matlab script (netdata-synapsedistance.m) with frequency histogram data for the number of synapses by distance.
outattr_connection_distance_frequency	flag	false	Create an Octave/Matlab script (netdata-connectiondistance.m) with frequency histogram data for the number of connections by distance.
outattr_distance_frequency_distbinsize	□□□), decimal	50	The bin size for frequency by distance histogram data.

10.11. Simulation output: histological slice generation

parameter label	units and format	default value	description
slice	label	none	If a label is specified then histological slice output will be created for the generated network. Note that <label> in the following options can point to a hierarchical level, e.g. “1.2.3.4”, or can be a specified ID.
<label>.batchsize	integer		
<label.<id>>.slice	new ID		
<label>.<vertex>[x/y/z]	decimal [, decimal, decimal]		
<label>.relativecoords	subindex		
<label>.batchrelativecoords	subindex		
<label><.nalpha/.ntransverse>	decimal		
<label><.nbeta/.ncoronal>	decimal		
<label><.ngamma/.nsagittal>	decimal		
<label>.origin	vertex		
<label>.width	decimal		
<label>.height	decimal		
<label>.depth	decimal		

10.12. Simulation output: statistical data

parameter label	units and format	default value	description
sample_dt	(s), decimal	86400 (one day)	The sample time interval for sampled output. This is an important parameter that is essential for statistics and (graphical) structure output at regular sample intervals during simulated development.
statsattr_collect_statistics	flag	true	Collect statistical data with which to create a stats.m Octave/Matlab file.
statsattr_D/Alength	flag	true	Include in stats.m statistics about the total length of fiber in each dendrite or axon arbor.
statsattr_D/Atermsegsperarbor	flag	true	Include in stats.m statistics about the number of terminal segments (i.e., growth cones) in each dendrite or axon arbor.
statsattr_D/Aterminallength	flag	true	Include in stats.m statistics about the lengths of the last straight fiber pieces at the terminal segments of each dendrite or axon arbor.
statsattr_D/Aintermedia	flag	true	Include in stats.m statistics about the lengths of the

telength			straight fiber pieces that are not included in **statsattr_D/Aterminallength** data.
statsattr_D/Alengthbetweenbifurcations	flag	true	Include in stats.m statistics about the length of fiber of intermediate segments between successive branching points.
statsattr_D/Atermlensincebifurcation	flag	true	Include in stats.m statistics about the length of fiber of terminal segments from the last branching point to the growth cone.
statsattr_D/Aturnsbetweenbifurcations	flag	true	Include in stats.m statistics about the number of turns (and therefore the number of straight pieces) between successive branching points.
statsattr_D/Abranchangles	flag	true	Include in stats.m statistics about branching angles in each dendrite or axon arbor.
statsattr_D/Aturnangles	flag	true	Include in stats.m statistics about turn angles in each dendrite or axon arbor.
statsattr_D/Atermlensincesoma	flag	true	Include in stats.m statistics about fiber path lengths from soma to growth cones in each dendrite or axon arbor.
statsattr_D/Acartratiobetweenbifurcations	flag	true	Include in stats.m statistics about the ratio of Cartesian distance / fiber length between successive branching points.
statsattr_D/Acartratiobifurcationtoterm	flag	true	Include in stats.m statistics about the ratio of Cartesian distance / fiber length between growth cones at the last branching points on the corresponding terminal segments.
statsattr_D/Acartratiosomatoterm	flag	true	Include in stats.m statistics about the ratio of Cartesian distance / fiber length from soma to growth cones.
statsattr_D/Asevenmicronbranchangles	flag	false	Include in stats.m statistics about branching angles as computed by taking points 7 µm before a branching point and 7 µm after a branching point on each daughter branch. These measurements are compatible with manually registered measurement made in images of cultured neurons (validation data by G.Ramakers).
statsattr_autoplot	flag	false (except if called from the HTML FORM interface)	Use a preset format for graphical Octave output of statistical data in the stats.m file. This is generally only used by utilities that automate the presentation of results, such as when an HTML FORM interface is used to control NETMORPH simulation.
statsattr_store_raw_data	flag	false	Store the raw data at each sample time during simulation. This is the raw sample data that is used to calculate the statistics normally returned in the stats.m file.
statsattr_fan_in_analysis	none / axons / dendrites	none	Perform a “fan-in” analysis of the angles of neurite outgrowth. If certain types of fiber structure should not be included in the analysis, e.g. apical dendrites, then those should be effectively removed from the simulation by setting relevant parameters. For example, setting the elongation rate of apical dendrites to 0 removes them from a simulation and its analysis. See McMullen et al. (2005; Ref. [5]).
outattr_show_stats	flag	true	Write statistical data to the stats.m file. In the case of NETMORPH control via a HTML FORM interface, also

			present the Octave graphical representation of the statistical data in a resulting HTML page.
outattr_count_segments	flag	inactive	Not in use.
outattr_count_synapse_search	flag	inactive	Not in use.
outattr_synapse_genesis_and_loss	flag	inactive	Not in use.

10.13. Simulation output: runtime options

parameter label	units and format	default value	description
warnings_on	selection label	file	Where warnings are sent to: **off**, **stdout**, **stdoutfile** (i.e. both to stdout and the file), **file**. The file has the same path as all file output of a specific simulation run, preceding the label "_warnings".
reports_on	selection label	file	Where report messages are sent to: **off**, **stdout**, **stdoutfile** (i.e. both to stdout and the file), **file**. The file has the same path as all file output of a specific simulation run, preceding the label "_report".
progress_on	selection label	stdout	Where progress messages are sent to: **off**, **stdout**, **stdoutfile** (i.e. both to stdout and the file), **file**. The file has the same path as all file output of a specific simulation run, preceding the label "_progress".
outattr_show_progress	flag	false	If true then show verbose information about simulation progress.
outattr_directory	directory path	empty	Specify an alternate output directory. If NETMORPH command input includes an **include** script file and this output redirection is specified, then the resulting output files reside in the specified directory and have as a prefix the script file name (preceding any terminating "." extension). Without the redirection, output files are placed in the current directory by default, or into the directory of an **include** script file, if one appears in the NETMORPH command input. Note that if NETMORPH is controlled by an HTML FORM interface then the default outattr_directory is "../nibr/output/". In any case, all output files receive a date and time stamp "YYYYmmddHHMM_".
outattr_URL	URL		The absolute URL that precedes Octave graphical representations of statistical output when NETMORPH is controlled by HTML FORM input and is run on a remote server. The URL is used to display the results in an HTML page.
figattr_tsupd_visibly	flag	false	If true then convert from terminal segment angular coordinates to fiber segment cartesian coordinates at each sample time point, so that visualized sample output

			correctly shows terminal segment growth cone locations.

10.14. Simulation output: graphical visualization

parameter label	units and format	default value	description
outattr_show_figure	flag	true	If true then include the generated results figures in an HTML page if NETMORPH is controlled by HTML FORM input.
figattr_use_color	flag	true	Use color in network visualization. Graphics without color, but with different line types may be useful in some publication formats.
figattr_fill_somas	flag	true	Draw somata as filled circles.
figattr_neurons	flag	true	Show neuron soma.
figattr_connections	flag	false	Show lines that indicate abstract connections between neurons.
figattr_connections_threshold	(list of) decimal	(empty)	A list of threshold values. Only connections with a strength greater than the threhold value are shown. The smallest value is used for visualization in a full figure. Separate figure files with "-threshold-" specifications in the file name are created when a list of threshold values is given.
figattr_presynaptic	flag	true	Show axons.
figattr_postsynaptic	flag	true	Show dendrites.
figattr_synapses	flag	true	Show synapses.
figattr_<receptor>	flag	[true, true, true]	Show synapses with receptors of a specific type. The <receptor> types are: **AMPAR, NMDAR, GABAR**.
figattr_partitions	flag	false	Show spatial segment subset partitions that were generated for the candidate synapse search strategy.
figattr_connection_eval	flag	false	Show the spatial segment subset partitions in which candidate synaptic sites were sought since the last sample time point.
figattr_progress	indicator label	none	Include a visual progress indicator in graphical visualizations. The indicator labels are: none, text, bar.
figattr_show_scale	flag	true	Draw a scale bars that indicates a length of 100 µm.
figattr_make_full_Fig	flag	false	Create an XFig (net.fig) graphics file of the complete network without any boundary constraints.
figattr_make_zoom_Fig	flag	false	Create an XFig (zoom.fig) graphics file of a bounded cube excerpt of the 3D space that shows parts of the generated network that reside in that excerpt of space.
figattr_make_connections_Fig	flag	false	Create an XFig (connection-example.fig) graphics file showing the connections into and out of a specific nenuron.
figattr_make_abstract_Fig	flag	false	Create an XFig (abstract.fig) graphics file in which circular neurons are arranged along the circumference of a circle, and in which abstract connections are shown as lines through the circle. A depiction in this format can be

			useful for the identification of network connectivity properties, such as a small-world network property.
figattr_make_neurons_Figs	flag	false	Create an XFig (neuron-ID.fig) graphics file for each neuron (with index ID) in a generated network, so that each file contains a visualization of an individual neuron.
net_zoom_disttoedge	(µm), decimal	1	The distance from the center location of an excerpt visualization to each of the boundaries.
net_zoom_center[X/Y/Z]	index / (µm), decimal	spatial center of neurons in network	The center location for excerpt visualization. This may be indicated by the index of one of the neurons in the network or by a specific 3D coordinates.
figattr_fibres_nobox	flag	false	If true then for all neuron soma with center points that are within the zoom volume, all fibers are drawn regardless whether fiber pieces are within the volume or not. Otherwise, only those pieces that are within the zoom volume are drawn.
figattr_box_fibre_independently	flag	true	If true, then fiber pieces that are within the zoom volume are drawn, even if the soma center points of the neurons that they belong to are not within the volume boundaries. This gives the appearance of a slice of transparent tissue with neural fiber.
fibrediameter	flag	false	If true then show a simple representation of the fiber diameter in XFig compatible output files.
figattr_show_axis_arrows	flag	true	Draw 3D axis arrows that indicate the axial directions of each dimension.
figureTeXwidth	(inches), decimal	6.5	Scale the values in output XFig files, so that the width of the network appears as **figureTeXwidth** inches wide at a resolution of 1200dpi.
figattr_Fig_rescale	flag	true	Rescale .fig output as needed, based on the actual coordinates that will be sent to the .fig file, in order to sufficiently approximate the desired **figureTeXwidth**.
color_table	table label	default	Use one of the predefined color tables for visual output. Options are: default, culturestain.
<CT_id>	integer / hexadecimal	according to chosen table	Set the color of an individual component with an RGB value (hexadecimal should be preceded by "0x"). The component identifiers **CT_id** are: **CT_background, CT_neuron_untyped, CT_neuron_principal, CT_neuron_interneuron, CT_connection_excitatory, CT_connection_inhibitory, CT_synapses, CT_dendrites, CT_axon_excitatory, CT_axon_inhibitory, CT_partition_boundary, CT_partition_overfull, CT_partition_evaluated, CT_progress_text, CT_AMPAR, CT_NMDAR, CT_GABAR**.
***color**	integer / hexadecimal	default initialized according to the CT table	Set the display color of a specific component, as identified by the shema protocol. For example, set **all_apical_pydamidal_dendrites.color**.

10.15. Simulation output: sequences and animation

parameter label	units and format	default value	description
figuresequence	flag	false	If true then a sequence of output figures is created, one at each sample time point.
combinesequence	flag	false	If true then the sequence of output figures is combined into one animation.
combinetype	type extension	gif	Specify the animation file format. Options are: gif and any other formats recognized by the "convert" program of ImageMagick.
sequence_total_time	(s), decimal	25	The number of seconds for one full cycle of animation.
sequence_zoom_disttoedge	(μm), decimal	1	When creating a figure sequence, the distance from the center location of the cube excerpt visualization to each of the boundaries. Note: This is not used if a value greater than 0 is set in the sequence_zoom_width parameter.
sequence_zoom_<width/height/depth>	(μm), decimal	inactive (uses _disttoedge by default)	When creating a figure sequence, the bounded volume width, height and depth of the excerpt visualization.
sequence_zoom_center[X/Y/Z]	index / (μm), decimal	[0,0,0]	When creating a figure sequence, the center location for excerpt visualization. This may be indicated by the index of one of the neurons in the network or by specific 3D coordinates.
autorotatesequence	flag	false	If true then rotation is applied to successive figures in a sequence.
ROT[_x/_y/_z]	(radians), vector / decimal	[0,0,0]	Initial rotated angles around the center of coordinate axes.
ROT_interval[_x/_y/_z]	(radians), vector / decimal	[2π,2π,2π]	The total angles through which figures in a sequence are rotated.
camera[_x/_y/_z]	(μm), vector / decimal	inactive (uses ROT by default)	The location of a projection camera.
camera_ROT[_x/_y/_z]	(radians), vector / decimal	[0,0,0]	The rotation of a projection camera.
viewer[_x/_y/_z]	(μm), vector / decimal	[0,0,0]	The location of a projection viewer.
max_res_samples	flag	false	If true then output is stored for each time step of the simulation.
combinemagnification	decimal	1.0	This is a magnification factor used when an animation is generated.
camera	flag	false	If true then projection camera specifications are used to create 3D visualizations.

10.16. Outgrowth in a qualified environment

parameter label	units and format	default value	description
a1	decimal	0.0	Sensitivity of directed outgrowth to the external field.
a3	decimal	0.0	Sensitivity of branching rate to the external field.

11. Output files generated during NETMORPH simulations

11.1. stats.m

When **statsattr_collect_statistics** is true then specified statistics about the generated morphologies of neurons in a simulated network development are stored in a file located in the designated output directory, with the script name and time stamp prefix, with the terminating designation "_stats.m".

The stats.m file conforms to Octave and Matlab script conventions and can be read in those programs by typing the file name at the prompt. The first line of the file header that precedes a number of comment lines (identified by the first character #) is:
#! /usr/bin/octave

In Linux/BSD/Mac OSX/Unix/Cygwin environments, that line identifies the file as an executable script. Running the script at a terminal prompt automatically calls the Octave program, which is assumed to reside in the directory /usr/bin.

The collected statistics immediately follow the file header and are arranged as numerical matrices in the Octave and Matlab format.

- Successive rows signify samples taken at successive simulation time points, as specified by the **sample_dt** parameter.
- Five columns specify the statistics:

1. N, the sample size
2. mean value
3. standard deviation
4. minimum value
5. maximum value

The first three matrices do not convey statistics, but are always included to indicate:

1. T: The simulation time points of the samples taken.
2. Darborssampled: The number of dendrite trees included in the sample statistics at each simulation time point.
3. Aarborssampled: The number of axon trees included in the sample statistics at each simulation time point.

The following matrices are included if the corresponding **statsattr_<specific>** flag is set:

1. Dlength & Alength: The sum of all lengths of neurite fiber per dendrite or axon.
2. Dtermsegsperarbor & Atermsegsperarbor: The number of terminal segments (i.e. the number of growth cones) per dendrite or axon.
3. Dterminallength & Aterminallength: The lengths of terminal fiber segment pieces in dendrites or axons, measured from each growth cone to the preceding turn or branch point in the neurite.
4. Dintermediatelength & Aintermediatelength: The lengths of intermediate fiber segment pieces in

dendrites or axons, measured between consecutive nodes that are the points at which a turn or bifurcation occurs in a neurite.
5. Dlengthbetweenbifurcations & Alengthbetweenbifurcations: The intermediate lengths of neurite fiber, as measured along the fiber between consecutive bifurcations or from the root node (at the soma surface) to the first bifurcation point. This measurement continues through turns.
6. Dtermlensincebifurcation & Atermlensincebifurcation: The terminal lengths of neurite fiber, as measured from a growth cone to the nearest preceding bifurcation or to the root node at the soma surface if there are no branch points. This measurement continues through turns.
7. Dtermlensincesoma & Atermlensincesoma: The lengths of neurite fiber along a path that is measured from a growth cone to the root node at the soma surface . This measurement continues through turns and through preceding branch points.
8. Dturnsbetweenbifurcations & Aturnsbetweenbifurcations: The number of turning points in segments of neurite fiber that stretch between two successive branch points, or in neurite fiber from a root node at the soma surface to the first branch point.
9. Dbranchangles & Abranchangles: The pitch angles of the branches after a bifurcation, i.e. the angle of divergence from the parent direction of growth.
10. Dturnangles & Aturnangles: The pitch angle of the growth cone direction as compared to the direction of growth before a turn.
11. Dcartratiobetweenbifurcations & Acartratiobetweenbifurcations: The ratio of the cartesian distance (also called the Euclidean distance) between successive bifurcation points in 3D and the distance as measured along the neurite fiber between those bifurcation points, or the ratio of the cartesian distance from a root node location to the first bifurcation point in 3D and the distance as measured along the neurite fiber from that root node location at the surface of the soma to the first bifurcation point. This is a measure of tortuosity.
12. Dcartratiobifurcationtoterm & Acartratiobifurcationtoterm: The ratio of the cartesian distance (also called the Euclidean distance) between a growth cone and the preceding bifurcation point in 3D and the distance as measured along the neurite fiber between that growth cone and the preceding bifurcation point, or the ratio of the cartesian distance from a growth cone to the root node location in 3D and the distance as measured along the neurite fiber from that growth cone to the root node location at the surface of the soma. This is a measure of tortuosity.
13. Dcartratiosomatoterm & Acartratiosomatoterm: The ratio of the cartesian distance (also called the Euclidean distance) between a growth cone and the root node at the surface of the soma in 3D and the distance as measured along the neurite fiber between that growth cone and the root node. This is a measure of tortuosity.

11.2. Textual output files containing the generated network structure

If the parameter **outattr_make_full_Txt** is set to **true** then NETMORPH simulations produce an output file with a filename that ends in "_net.txt" and contains a textual encoding of the generated network morphology. The following describes the content of the text file. Please note that if the option **outattr_Txt_separate_files** is set to true then the data described in each of the following sections appears in a separate file with the appropriate label appended to the file name: .header, .neurons, .rootnodes, .continuationnodes, .bifurcationnodes, .growthcones.

11.2.1. Txt header

The header notes that the file is a "NETMORPH Txt" file containing network structure, then lists the data matrices that will follow: neurons, synapses, fiber structure root nodes (for both axons and dendrites), continuation nodes (turn points), bifurcation nodes (branch points), terminal growth cones (end points), apical dendrite tuft root nodes (branch points at which the tuft of an apical dendrite begins), apical dendrite oblique root nodes (branch points at which an oblique branch of an apical dendrite begins). The last two matrices in the list contain reference labels to specific entries in the matrix of bifurcation nodes and can be useful during quantitative analyses of the tuft and oblique branch structures of apical dendrites. The header then lists descriptive labels for the columns of each

data matrix.

That information is followed by several comment lines that were generated during the simulation, recording a number of parameter settings. This is the same information that is also stored as comment text in XFig figure output produced by NETMORPH.

The header is followed by data sections for each of the data matrices, containing one line of data for each object, and headed by the matrix labels: **neurons**, **synapses**, **fiber structure root nodes**, **fiber continuation nodes**, **fiber bifurcation nodes**, **terminal fiber growth cones, apical dendrite tuft root nodes, apical dendrite oblique root nodes.**

11.2.2. Txt neurons

Each line contains data for one neuron. The columns are:
1. An enumerating neuron index.
2. A unique neuron label that is used as a reference to this neuron throughout the data.
3. The name of the neuron type.
4. The name of the network region that the neuron is allocated to.
5. Soma center position X coordinate.
6. Soma center position Y coordinate.
7. Soma center position Z coordinate.

11.2.3. Txt synapses

Each line contains data for one synaptic site. The columns are:
1. An enumerating synapse index.
2. The name of the synapse type.
3. X, Y and Z coordinates of the presynaptic location of the synaptic site (3 columns).
4. X, Y and Z coordinates of the postsynaptic location of the synaptic site (3 columns).
5. A reference to a uniquely labeled axon.
6. A reference to a uniquely labeled dendrite.
7. A reference to a uniquely labeled presynaptic neuron.
8. A reference to a uniquely labeled postsynaptic neuron.
9. If the **outattr_track_synaptogenesis** flag is set then the 13th column contains the simulation time at the time step when the synapse was created.

11.2.4. Txt fiber structure root nodes

Each line contains data for the node that specifies the location at which one root fiber segment emerges from a neuron soma. The columns are:
1. An enumerating node index (shared with continuation nodes, bifurcation nodes and growth cones.)
2. A unique root node label that is used as a reference to fiber structure that begins with this root node throughout the data.
3. A label string that indicates if this node belongs to an axon or a dendrite.
4. X, Y and Z coordinates of the root node on the surface of the soma of a neuron (3 columns).
5. A reference to a uniquely labeled neuron to which the fiber structure belongs.
6. If the **outattr_track_nodegenesis** flag is set then the 8th column contains the simulation time at the time step when the root node was created.

11.2.5. Txt fiber continuation nodes

Each line contains data for one node at which a growth cone made a turn and where the resulting two fiber segment pieces meet. The columns are:

1. An enumerating node index (shared with root nodes, bifurcation nodes and growth cones).
2. A unique continuation node label that is used as a reference to the fiber segment piece that ends in this continuation node throughout the data.
3. A label string that indicates if this node belongs to an axon or a dendrite.
4. X, Y and Z coordinates of the continuation node where two fiber segment pieces meet (3 columns).
5. A reference to a uniquely labeled neuron to which the fiber structure belongs.
6. A reference to the enumerating index of the parent node, i.e. the node that precedes this one in the neurite fiber.
7. The diameter of the neurite at this node in µm.
8. If the **outattr_track_nodegenesis** flag is set then the 10th column contains the simulation time at the time step when the continuation node was created.

11.2.6. Txt fiber bifurcation nodes

Each line contains data for one node at which a growth cone bifurcated into two branches and where the resulting three fiber segment pieces meet. The columns are:
1. An enumerating node index (shared with root nodes, continuation nodes and growth cones).
2. A unique bifurcation node label that is used as a reference to the fiber segment piece that ends in this bifurcation node throughout the data.
3. A label string that indicates if this node belongs to an axon or a dendrite.
4. X, Y and Z coordinates of the bifurcation node where three fiber segment pieces meet (3 columns).
5. A reference to a uniquely labeled neuron to which the fiber structure belongs.
6. A reference to the enumerating index of the parent node, i.e. the node that precedes this one in the neurite fiber.
7. The diameter of the neurite at this node in µm.
8. If the **outattr_track_nodegenesis** flag is set then the 10th column contains the simulation time at the time step when the bifurcation node was created.

11.2.7. Txt terminal fiber growth cones

Each line contains data for one growth cone at which a terminal segment of neurite fiber ends. The columns are:
1. An enumerating node index (shared with root nodes, continuation nodes and bifurcation nodes).
2. A unique growth cone label that is used as a reference to the terminal fiber segment piece that ends in this growth cone throughout the data.
3. A label string that indicates if this growth cone belongs to an axon or a dendrite.
4. X, Y and Z coordinates of the growth cone (3 columns).
5. A reference to a uniquely labeled neuron to which the growth cone belongs.
6. A reference to the enumerating index of the parent node, i.e. the node that precedes this growth cone in the neurite fiber.
7. The diameter of the neurite at this growth cone in µm.
8. If the **outattr_track_nodegenesis** flag is set then the 10th column contains the simulation time at the time step of the last elongation at this growth cone.

11.2.8. Txt apical dendrite tuft root nodes

Each line contains a reference label for one bifurcation node that is the root of the tuft structure of an apical dendrite. The column is:
1. A reference to a uniquely labeled bifurcation node (see **Txt fiber bifurcation nodes** item 2).

11.2.9. Txt apical dendrite oblique root nodes

Each line contains a reference label to one bifurcation node that is the root of an oblique branching structure of an apical dendrite. The column is:

1. A reference to a uniquely labeled bifurcation node (see **Txt fiber bifurcation nodes** item 2).

12. References

[1] Koene, R. A., Tijms, B., Van Hees, P., Postma, F., De Ridder, A., Ramakers, G. J. A, Van Pelt, J., and Van Ooyen, A. (2009). NETMORPH: a framework for the stochastic generation of large scale neuronal networks with realistic neuron morphologies. *Neuroinformatics* 7: 195-210.

[2] Van Pelt, J., and Uylings, H. B. M. (2003). Growth functions in dendritic outgrowth. *Brain and Mind* 4: 51-65.

[3] Van Pelt, J., Van Ooyen, A., and Uylings, H. B. M. (2001). Modeling dendritic geometry and the development of nerve connections. In: De Schutter, E., ed. *Computational Neuroscience: Realistic Modeling for Experimentalists*. Boca Raton: CRC Press, pp. 179-208.

[4] Samsonovich, A. V., and Ascoli, G. A. (2003). Statistical morphological analysis of hippocampal principal neurons indicates cell-specific repulsion of dendrites from their own cell. *J. Neurosci. Research* 71: 173–187.

[5] McMullen, N. T., Velenovsky, D. S., and Holmes, M. G. (2005). Auditory thalamic organization: cellular slabs, dendritic arbors and tectothalamic axons underlying the frequency map. *Neuroscience* 136: 927-943.

[6] Van Pelt, J., Carnell, A., De Ridder, S., Mansvelder, H. D., and Van Ooyen, A. (2010). An algorithm for finding candidate synaptic sites in computer generated networks of neurons with realistic morphologies. *Frontiers in Computational Neuroscience* doi: 10.3389/fncom.2010.00148.

13. Studies using NETMORPH

McAssey, P. M., Bijma, F., Tarigan, B., Van Pelt, J., Van Ooyen, A., and De Gunst, M. A. (2014). A morpho-density approach to estimating neural connectivity. *PloS ONE* 9(1): e86526. doi:10.1371/journal.pone.0086526.

Van Ooyen, A., Carnell, A., De Ridder, S., Tarigan, B., Mansvelder, H. D., Bijma, F., De Gunst, M., Van Pelt, J. (2014). Independently outgrowing neurons and geometry-based synapse formation produce networks with realistic synaptic connectivity. *PloS ONE* 9(1): e85858. doi:10.1371/journal.pone.0085858.

Mäki-Marttunen, T., Aćimović, J., Ruohonen, K., and Linne, M.-L. (2013). Structure-dynamics relationships in bursting neuronal networks revealed using a prediction framework. *PLoS ONE* 8(7): e69373. doi:10.1371/journal.pone.0069373.

Van Pelt, J., and Van Ooyen, A. (2013). Estimating neuronal connectivity from axonal and dendritic density fields. *Frontiers in Computational Neuroscience* doi: 10.3389/fncom.2013.00160.

Aćimović, J., Mäki-Marttunen, T., Havela, R., Teppola, H., and Linne, M.-L. (2011). Modeling of neuronal growth in vitro: comparison of simulation tools NETMORPH and CX3D. *EURASIP Journal on Bioinformatics and Systems Biology* doi:10.1155/2011/616382.

Mäki-Marttunen, T., Aćimović, J., Nykter, M., Kesseli, J., Ruohonen, K., Yli-Harja, O., and Linne, M.-L. (2011). Information diversity in structure and dynamics of simulated neuronal networks. *Frontiers in Computational Neuroscience* doi: 10.3389/fncom.2011.00026.

Mäki-Marttunen, T., Aćimović, J., Ruohonen, K., and Linne, M.-L. (2011). Effects of local structure of neuronal networks on spiking activity in silico. *BMC Neuroscience 2011, 12 (Suppl 1):P202 http://www.biomedcentral.com/1471-2202/12/S1/P202*. Abstract poster.

Aćimović, J., Mäki-Marttunen, T., and Linne, M.-L. (2011). Computational study of structural changes in neuronal networks during growth: a model of dissociated neocortical cultures. *BMC Neuroscience 2011, 12 (Suppl 1):P203 http://www.biomedcentral.com/1471-2202/12/S1/P203.* Abstract Poster

Van Pelt, J., Carnell, A., De Ridder, S., Mansvelder, H. D., and Van Ooyen, A. (2010). An algorithm for finding candidate synaptic sites in computer generated networks of neurons with realistic morphologies. *Frontiers in Computational Neuroscience* doi: 10.3389/fncom.2010.00148.

Laakso, R. (2010). Algorithmic description of pyramidal neuron populations from mouse neocortex. *Thesis.* Aalto University, School of Science and Technology, Finland.

Krieger, P. and Laakso, R. (2010). Modeling realistic morphologies of genetically labelled layer 5 pyramidal neurons. *Front. Neurosci. Conference Abstract: Neuroinformatics 2010* . doi: 10.3389/conf.fnins.2010.13.00106