Dendritica

Version 1.0

9 February 2001

Philipp Vetter, Arnd Roth and Michael Häusser

Table of contents

1. General Introduction

Dendritica is a program package for relating dendritic geometry and signal propagation. The programs are based on those used for the simulations described in the following paper:

> Vetter, P., Roth, A. & Häusser, M. (2001). Action potential propagation in dendrites depends on dendritic morphology. *Journal of Neurophysiology*, 85: 926-937.

Dendritica can functionally be divided into three main parts:

- interactive morphological analysis and electrophysiological simulation of single cells
- automated batch simulations across a set of morphologies using the same simulation parameters
- automated analysis of batch simulation runs

Dendritica requires NEURON 4.1.1 with some modifications described in Appendix 1. It was tested for NEURON 4.1.1 on Linux and SGI IRIX. Some modifications to the *Dendritica* code may be necessary in order to run it on older or newer versions of NEURON.

We are very grateful to Muki Rapp, Diana Smetters, Nelson Spruston, Greg Stuart, and the contributors to the Duke-Southampton Neuronal Morphology Archive (accessible via http://www.neuro.soton.ac.uk) for allowing us to use their neuronal reconstructions for this project. We also thank Alain Destexhe (Destexhe@iaf.cnrs-gif.fr) and Zach Mainen (mainen@cshl.org) for providing NEURON code. Please note that while the programs in *Dendritica* are freely available, they are protected by the GNU public licence, and we request that you acknowledge us if you use the programs for a publication. If you require further information, please do not hesitate to contact any of the authors: Philipp Vetter (p.vetter@ucl.ac.uk), Arnd Roth (roth@mpimf-heidelberg.mpg.de), or Michael H usser (m.hausser@ucl.ac.uk).

1.1 Directory structure

The directory structure is defined automatically when you untar the package. The structure must be respected for most of the functions to work properly.

The directory **`dendritica-1.0`** contains all files. There are three subdirectories:

```
batch_back/          batch_forward/          batch_forward2/
```

indicating different types of simulation runs, i.e. looking a backpropagating action potentials and forward propagating action potentials. The structure of the subdirectories is identical, however.

```
batch_back/back:
       aphalf.hoc                    gui.hoc
       batch1.hoc                    help.hoc
       batch2.hoc                    impedance.hoc
       batch3.hoc                    init.hoc
       batchrun                      mod/
      bp                             neuronprefs.hoc
       dendspike_p21                 output.hoc
        electrophysiology.hoc        parse.hoc
       figures.hoc                   referenceAP_p18@200um_act0
       forward.hoc                   settings.hoc
       geometry.hoc                  statistics.hoc
       graphics.hoc
```

The `*.hoc` files contain the code for the NEURON interpreter. It is split into several files according to what the procedures/functions do. The directory `mod/` contains all mod files necessary to create the `special` executable of NEURON. The mod files for the simulations are in the subdirectory `kvz_naz`. `dendspike_p21` and `referenceAP_p18@200um_act0` are saved waveforms that can be played in during simulations.

```
  batch_back/data:
      act0/        cells/       geometry/
```

`act0` contains simulation results that depend on the active model used, while geometry contains simulations results that are independent. `cells` contains the morphologies of all the cells used.

```
  batch_back/neuron_output:
```

This contains ascii files with correlation analysis from a batch run.

2. Getting Started

Most of the data presented in the paper are directly accessible through the graphical user interface (GUI). (Note that the optimization routine to find `halfdecay_max` is an exception).

2.1 A sample session

- To create the program `special` go to `dendritica-1.0/batch_back/back/mod/kvz_naz`
- To create `special` type
  `>nrnivmodl`
- Move `special` to `dendritica-1.0/batch_back/back/`, then load the GUI with
  `>special gui.hoc -`
  `Dialog box: Welcome to Progagation Geometry [Load Cell] [Statistics]`
- Choose `[Load Cell]`
  `Dialog box: Please pick a neuron and an active model [Neuron] [Conductances]`
- Specify Neuron `[Nigra]->[Nigra2]`, and Conductance `[standard conductances]` (`act0` is the setting used in the paper), and then `[Accept]`. The Morphology is loaded, and subsequently the panels `Electrophysiology` and `Main` (see below) are called.

`[Voltage Clamp]` runs a Voltage clamp simulation with the Electrode standardly located at the soma. This takes about a minute on a PentiumII, depending also on the morphology being simulated. The position can be changed manually by clicking on `[Location]`. The size of the waveform can be changed by entering a value under `[scaling]`. The standard waveform is `somatic AP (p18)`, but others can be chosen from the pull-down menu `[waveform]`.

`[Current Clamp]` clamps a constant current, standardly at the soma. The electrode location can be changed manually using `[Location]`. The magnitude of the current can be specified by changing `[Amplitude]`.

`[Synapse]` simulates a synapse. The parameters and location of the synapse can be altered by clicking on `[Location]` and/or `[gmax]`.

`[Input Resistance]` calculates the input resistance at the soma.

`[g_na threshold]` calculates the Na-channel density for full backpropagation. This simulation can take >30 minutes!

- Press `[Voltage Clamp]`. After the simulation is over, 4 figures are plotted:

(1) Voltage traces at soma, node and dendrite
(2) The AP amplitude as a function of distance from the soma
(3) Simulation settings
(4) Plot of rate of change of peak voltage as a function of distance from the soma
The same plots are obtained when simulating `[Current Clamp]` or `[Synapse]`.

2.2 Options available from the GUI

```
[Load Neuron]
[Clear Screen]
[Graphs]
```

```
   ->[New Graph]
   ->[Which Sections]
   ->[Which Lengths]
   ->[Simulation]
   ->[Geometric Values]
   ->[Functional Values]
   ->[Simulation X against distance]
   ->[Geometric individual]
   ->[Impedance individual]
[Other Panels]
   ->[Electrophysiology]
   ->[Statistics]
   ->[Channels]
   ->[Simulation settings]
   ->[Geometry]
[Miscellaneous]
   ->..
[Quit]
```

- Choose `[Graphs]->[Which Sections]->[all]`
  to plot figures (2) & (4) with data from all segments of the Morphology.
- Choose `[Graphs]->[Which Lengths]->[electrotonic]`
  to plot figures (2) & (4) in electrotonic space (X-axis!).
- Choose `[Other Panels]->[Simulation settings]`
  to call up a panel allowing to change simulation duration, and time step.
- Choose `[Other Panels]->[Conductances]`
  to call up a panel which allows setting of the active membrane properties.
- To re-run the simulation, simply press `[Voltage Clamp]` in the panel `Electrophysiology`. Units as in the paper.
- Choose `[Other Panels]->[Geometry]`
  to call up a panel which allows the Axon to be removed `[Remove Axon]` or added to the morphology `[Connect Axon]`.
- `[Graphs]->[Geometric Values]` Plots 5 figures that have been calculated in a batch simulation (see next section)

(1) Branchpoint and Termination histogram as a function of distance from the soma
(2) Cumulative membrane area as a function of distance from the soma
(3) Rate of change in membrane against distance from the soma
(4) Rall ratio distribution of branchpoints (smoothed)
(5) number of sections at a given distance from the soma.

2.3 On-line help

An on-line help can be accessed from the command line. All functions and procedures of the package can be listed with the command

```
oc>hlp()
parse.hoc
get()
get_somadist()
connect_axon()
   ...
```

The listing is sorted according to the **.hoc** files the functions and procedures are defined in. To get more information about a particular function, e.g. the function **get**, type

```
oc>hlp("get")
get cell $s1
use ActiveModel $s2
load data if numarg=3
```

2.4 Basic commands for running simulations

There are four basic commands to run simulations from the command line (see **hlp()** for details)

• **get()**	loads a morphology, it s simulation results, gets it ready for simulations
• **sim()**	runs a simulation
• **fig()**	plots vectors
• **spaceplot()**	dumps a spaceplot on disk

2.5 What happens when loading a cell

get() loads morphologies from **../data/cells/<name of cell>**, specifies name and **spine_density** in **neuronprefs.hoc** (structure **MyCell**) inserts passive membrane properties and channels **parse.hoc**, then sets the parameters as specified in **settings.hoc**. To facilitate analysis, the morphology is split into soma and dendrites (note Purkinje cells have two types of dendrites), and SectionLists are specified in parse.hoc accordingly.

```
dist_switch()
if (n == 1) distlist = trunk
if (n == 2) distlist = all
if (n == 3) distlist = branchpoints
if (n == 4) distlist = terminations
if (n == 5) distlist = branchpt_noend
if (n == 6) distlist = all_noend
```

Simulation results (calculated previously) are loaded from **../data/act0/** and **../data/geometry**" into vectors. These vectors can be printed using **pt(<vectorname>)**, or plotted using **fig(<vectorname>)**. Vectors can be plotted against each other as **fig(<vectorx>,<vectory>)** (see **hlp("fig")**).

2.6 What happens during simulations

Most functions to do with simulations are in **electrophysiology.hoc** (see **hlp()** for details). Simulations come in three flavours - voltage clamp/current clamp/synapse, which is set by the flag simMode. **sim()** brings the cell to resting potential with **rest()**, then inserts the appropriate PointProcess. The unused PointProcesses are parked on a dummy section. **sim** then calls **simcore()** which is equivalent to **run()**. Because some values, like the AP half-width require knowledge of the AP-waveform, **simcore()** has to be called twice, so that these values can be calculated.

To be able to plot them, type

```
>sim_calc()
```

which creates the following vectors that can be plotted against distance from the soma.

- **vpk** - peak voltage
- **amp** - AP amplitude
- **vmax** - maximum velocity

- **plat** - peak latency
- **olat** - onset latency
- **half** - half distance
- **dvdr** - spatial derivative of peak voltage

e.g.

```
>sim()
>dist_switch(2)    //  all  sections  [optional]
>L_switch(0)         //  physical  lengths  [optional]
>sim_calc()
>fig(dist,vpk)
```

3. Batch Simulations

Batch simulations allow for the automated generation and saving of simulation results across a wide range of morphologies using the same set of simulation parameters. Because these calculations are computationally intensive, it is more efficient to invoke `batch()` without using the GUI. Simulation runs take >24 hours on a PentiumII 450 MHz.

3.1 Examples

```
>batch(17,act0)
```

performs all the calculations in conjuction with action potential backpropagation, using the standard active model “act0”.

```
>batch(18,act0)
```

performs all the calculations when the action potential is generated at a dendritic location 200 um from the soma.

```
>batch(19,act0)
```

performs all the calculations when the action potential is generated at the dendritic location from where the action potential has the greatest halfdecay distance.

3.2 How batch simulations are done

The procedure `batch()` is a loop which applies a function to all cells in turn. The specifics of this are defined in `output.hoc` Basically, calculations done in `electrophysiology.hoc`, `geometry.hoc` and `impedance.hoc` are saved as numbers or vectors in the directories `../data/act0` and `../data/geometry.` The convention is that the directory name is the same as the vector, and the filename is the same as that of morphological data of the cell in `../data/cells`.

4. Batch Analysis

4.1 A sample session

The results of the batch simulations can be analysed using the graphical user interface.

- Go to directory `dendritica-1.0/batch_back/back/` and type
  ```
  >special gui.hoc -
  Dialog box: Welcome to Progagation Geometry [Load Cell]
  [Statistics]
  ```
- Choose `[Statistics]`
  ```
  Dialog box: Select dataset to analyse
  [Conductances]
  [] equivalent
  [] backpropagation
  [] forward200
  [] forwardhdecay
  ```
- Select `[Conductances]->[standard]`
- Select `[x] backpropagation`
- Press `[Load]`
  The simulation results are loaded into memory, and the panels `Main` and `Statistics` are opened
  ```
  [Get_Data][Legend]
  [Average][Single][Double][Triple][]Powers
  [Y]
  [X1]
  [X2]
  [X3]
  ```
- Choose `[Y]->[1]->[nathresholdvclamp]`
- Press `[Average]`
  This does 3 things

(1) plots a bar chart with cell-type averaged Na thresh values under voltage clamp
(2) prints numerical values on command line
(3) saves numerical values in ascii in
`dendritica-1.0/batch_back/neuron_output/nathresholdvclamp`

- Choose `[X1]->[3]->[d2area_max]`
- Press `[Single]`
- Press `[Legend]` This correlates the maximum rate of rise in membrane area as a function of distance from the soma with nathresholdvclamp and shows a legend colour-coding the cell types. Again, 3 things are done

(4) correlation plot
(5) numerical values on command line
(6) numerical values saved in
`/neuron_output/nathresholdvclamp vs branchpoints_num (act0)`

- Choose `[] Powers`
- Press `[Single]`
  This does the same as before, but maximizes the correlation nathresholdvclamp and d2area_max^exponent, by varying the exponent.
- Choose `[X2]->[3]->[diam_mean]`

- Press `[Double]`
  This maximizes the correlation between **`nathresholdvclamp`** and **`(d2area_max^a * diam_mean^b)`**
- Press `[Clear Screen]`
- Choose `[X1]->[geometric]`
- Deselect (optional) **`Powers`**
- Press `[Single]`
  This plots the 6 best correlations of geometric parameters against nathresholdvclamp, and plots a ranked list of correlations on the command line
- Press `[Clear Screen]`
- Type

```
>make_figures()
```

  This creates all the average and correlation plots shown in the paper.

- Type

```
>multi_correlation()
```

  This will save all good single and multiple correlations into the file

```
dendritica-1.0/batch_back/neuron_output/backpropagation
```

4.2 Basic commands

There are six key commands for ANALYSIS/STATISTICS

(1) **`get_data()`**	loads simulation results for the whole batch of cells
(2) **`averages()`**	prints/plots cell-type average for any parameter
(3) **`cplot()`**	correlates two parameters with each other
(4) **`single_corr()`**	correlates one parameter with all geometric parameters
(5) **`single_corrf()`**	correlates one parameter with all functional parameters
(6) **`writevecs()`**	writes vectors to disk

4.3 Settings

FLAGS that have to be set (use before calling `get()`):

`equiv`	1= equivalent cylinder mode
`hdecay`	1= morphology is cut in two, where the halfdecay distance is maximal; distal part removed
`forward`	1= morphology cut in two 200 um from soma, distal part removed
`simMode`	0= do voltage clamp when sim() is called
	1= do current clamp when sim() is called
	3= do synapse when sim() is called
`electrotonicL`	0= physical lengths
	1= electrotonic lengths

(Note that usually, **`equiv=hdecay=forward=simMode=0`**)

4.4 What happens during correlation analysis

During correlation analysis, all simulation results are read from disk into the vector

```
data[i][j][k]
```

- i ={0,1,2} and specifies, respectively, a functional,physically-geometric,electrotonically-geometric parameter
- j ={0..30} for the different parameters
- k ={0,1,2} 0 = parameter (normal), 1 =exp(parameter), 2 = ln(parameter)

It s a nuisance to specify one vector with three numbers, so there is a one-number shorthand

```
1000*i + 100*k + j   {if i==0 add 3000 }
```

`dissect()` turns shorthand into ci,cj,ck, **`antidissect()`** does the opposite. Because they are all vectors they can be plotted and manipulated as mentioned above.

To get averages of a parameter for a given cell-type

```
>averages(3014)   // gets nathreshold (voltage clamp mode) averages
```

N.B. This writes the numerical values into a correctly named file

```
../neuron_output/nathesholdvclamp
```

To make the correlations, the appropriate **`data[][][]`** vectors are copied into **`vecx`** and **`vecy`**, and **`Rcorrelation()`** is applied. To correlate two parameters

```
>cplot(3014,1000)
```

N.B. This writes the numerical values into a correctly named file in
`../neuron_output/nathresholdvclamp_vs_area_max_act0`

N.B.II All such data relevant for the figures is generated automatically using **`make_figures()`**

Many correlations are possible (just loop through the indices i,j,k)

and in order to make sense of the data (**`single_corr()/single_corrf(),double_corr()`**. To make the data more easy to read, they are ranked according to their correlation coefficient in **`good_corr()`**.

To look at a mix of these correlations (with and without powers | normal or equivalent cable geometries etc)

4.5 List of functional parameters

r = distance from soma

Δr = incremental distance

Parameter	**#**	**Description**
st_intensity	3001	Current needed to elicit a nodal AP in the absence of somatic/dendritic sodium channels
Nathreshold	3000	g_na that leads to a depolarization >0mV in all sections during current clamp at st_intensity
Nathresholdvclamp	3014	g_na that leads to a depolarization >0mV in all sections during voltage clamp with AP waveform
nathresholdvclamp2	3021	g_na that leads to a depolarization >0mV in terminal sections during voltage clamp with AP waveform
AP200	3010	AP amplitude 200 um from soma / AP amplitude at soma
AP200_pass	3011	AP amplitude 200 um from soma / AP amplitude at soma (g_na = 0)
AP200_half	3016	Sigmoidal fit of AP200 = f(g_na)
		AP200 = AP200_basis + AP200_range/ { 1+exp[-(g_na - AP200_half)/AP200_steep)]}
AP200_steep	3017	See above
AP200_range	3018	See above
AP200_basis	3019	See above
Aphalf	3012	Distance from soma at which AP amplitude has decayed to 50%
APhalf_pass	3013	Distance from soma at which AP amplitude has decayed to 50% (g_na=0)
input_resistance	3015	Input resistance at soma
Rfwd_min	3026	minimum somatofugal input resistance:
		Cut morphology in half at a given point, and measure the input resistance at the end with the somatofugal portion of the morphology
Rfwd_max	3027	Maximum somatofugal input resistance
Zfwd_min	3022	minimum somatofugal input impedance (f=200 Hz)
Zfwd_max	3023	Maximum somatofugal input impedance (f=200 Hz)
Rmismatch_peak	3002	Cut morphology in half at a given point
		Measure resting input resistance at both new ends.
		Mismatch is defined as the ratio of somatopetal/somatofugal input resistance.
		=> peak value of this mismatch
Zmismatch_peak	3003	Same as above, but measuring input impedance at 200 Hz
aRmismatch_peak	3004	Same as Rmismatch_peak, but measuring resistance at time, when the peak of the action potential has just reached the point of measurement.
aZmismatch_peak	3005	Analogous
Rmismatch_mean	3006	Same calculations as above, but take the mean over all points instead of peak.
Zmismatch_mean	3007	Analogous
aRmismatch_mean	3008	Analogous
aZmismatch_mean	3009	Analogous
dZfwd_max	3024	Maximum ΔZfwd/ Δr
dZfwd_min	3025	Minimum ΔZfwd/ Δr
dRfwd_max	3028	Maximum ΔZfwd/ Δr
dRfwd_min	3029	Maximum ΔZfwd/ Δr
aZfwd_min	3030	Same as Zfwd_min, but calculations done when action potential has just reached the point at which the cut is made.
aZfwd_max	3031	Analogous
daZfwd_max	3032	Analogous
daZfwd_min	3033	Analogous
aRfwd_min	3034	Analogous
aRfwd_max	3035	Analogous
daRfwd_max	3036	Analogous
daRfwd_min	3037	Analogous
cZfwd_min	3038	Minimum of ΔZfwd/ (Δr · Zfwd) over morphology
cZfwd_max	3039	Maximum of ΔZfwd/ (Δr · Zfwd) over morphology
cRfwd_min	3040	Minimum of ΔRfwd/ (Δr · Rfwd) over morphology
cRfwd_max	3041	Maximum of ΔRfwd/ (Δr · Rfwd) over morphology
caZfwd_min	3042	Minimum of ΔZfwd/ (Δr · Zfwd) over morphology when AP has just reached point
caZfwd_max	3043	Maximum of ΔZfwd/ (Δr · Zfwd) over morphology when AP has just reached point
caRfwd_min	3044	Minimum of ΔRfwd/ (Δr · Rfwd) over morphology when AP has just reached point
caRfwd_max	3045	Maximum of ΔRwd/ (Δr · Rfwd) over morphology when AP has just reached point

4.6 List of geometric parameters

PARAMETER	#	DESCRIPTION
branchpoints_num	1003	Number of branchpoints
distance_max	1006	Maximum r
area_max	1002	Total membrane area (spine corrected)
taper_mean	1010	Mean taper Δ(diameter)/ Δr in the somatofugal direction
darea_max	1000	Maximum Δ(membrane area)/ Δr
darea_maxdist	1001	r at which maximum Δ(membrane area)/ Δr is reached
dAdr_relmax	1012	First relative maximum in the change of membrane area after the first minimum change of membrane area as a fxn of distance from soma; these values are calculated semi-automatically
dAdr_ratio	1011	dAdr_relmax / preceding minimum change in membrane area as a fxn of distance from soma; these values are calculated semi-automatically
d2area_max	1013	Maximum rate of change in Δ(membrane area)/ Δr as a function of distance from the soma
d2area_maxdist	1014	Distance from the soma at which d2area_max is reached
d2area_maxAr_ratio	1015	At d2area_maxdist: membrane area distal to soma/membrane area proximal to soma
d2area_maxAr_percent	1016	At d2area_maxdist: 100*membrane area distal to soma/total membrane area
rallratio_mean	1004	Mean of the distribution of Rall ratios obtained from the branchpoints in the morphology
rallratio_peak	1005	Peak in the distribution of Rall-ratios obtained from the branchpoints in the morphology
sections_max	1007	Maximum number of sections at a given distance from the soma
sections_maxdist	1008	r at which sections_max
sections_mean	1009	Mean number of sections at all r
diam_mean	1017	Mean dendritic diameter
branchdensity	1018	Mean distance between branchpoints
branchdensityII	1019	Number of branchpoints / total length of dendritic sections
branchdensityII_noend	1023	Number of branchpoints / total length of non-terminal dendritic sections
diamratio_peak	1020	Peak of the distribution of diamter ratios at branchpoints given by Σdaughter branches / parent branch
diamratio_mean	1021	Mean of the above distribution
diamratio_noend_peak	1024	Same as diamratio_peak but leaving out terminal branchpoints
diamratio_noend_mean	1025	Same as diamratio_mean but leaving out terminal branchpoints
mean_stem_dendrite_diam	1026	Mean diameter of dendrites branching off from soma
rallratio_noend_peak	1027	Same as rallratio_peak, but omitting terminal branchpoints
rallratio_noend_mean	1028	Same as rallratio_mean, but omitting terminal branchpoints
deq_relmax	2014	Equivalent to dAdr_relmax in the equivalent cable representation
deq_ratio	2015	Equivalent to dAdr_ratio in the equivalent cable representation
ddeq_max	2016	Equivalent to d2_area_max in the equivalent cable representation
ddeq_maxdist	2017	Equivalent to d2_area_maxdist in the equivalent cable representation
ddeq_maxAr_ratio	2018	Equivalent to d2_area_maxAr_ratio in the equivalent cable representation
adarea_max	2000	darea_max in electrotonic space
adarea_maxdist	2001	darea_maxdist in electrotonic space
adistance_max	2002	distance_max in electrotonic space
asections_max	2003	sections_max in electrotonic space
asections_maxdist	2004	sections_maxdist in electrotonic space
asections_mean	2005	sections_mean in electrotonic space
ataper_mean	2006	taper_mean in electrotonic space
adiam_mean	2011	diam_mean in electrotonic space
abranchdensity	2007	branchdensity in electrotonic space
abranchdensityII	2008	branchdensityII in electrotonic space
abranchdensityII_noend	2010	branchdensityII_noend in electrotonic space
adeq_max	2012	deq_max in electrotonic space
adeq_maxdist	2013	deq_maxdist in electrotonic space

5. Appendix 1: Modifications to NEURON 4.1.1

To run all parts of *Dendritica* successfully, the following modifications to the NEURON source code are required.

```
diff -r nrn.new/src/ivoc/vector.c nrn.old/src/ivoc/vector.c
65,66c65,66
< // #define BYTEHEADER  int BYTESWAP_FLAG=0;
< // #define BYTESWAP(_X__,_TYPE__)
•  #define BYTEHEADER  int BYTESWAP_FLAG=0;
•  #define BYTESWAP(_X__,_TYPE__)
68,69c68,69
< #if 1
< // #include <sys/isa_defs.h>
•  #if 0
•  #include <sys/isa_defs.h>
Only in nrn.new/src/ivoc: vector.c.byteswap
Only in nrn.new/src/ivoc: vector.c.orig
diff -r nrn.new/src/nrnoc/cabcode.c nrn.old/src/nrnoc/cabcode.c
48,49c48
< #define NSECSTACK 10000
< /* A.R.   28.12.1998 */
•  #define NSECSTACK 20
Only in nrn.new/src/nrnoc: cabcode.c.orig
diff -r nrn.new/src/oc/hoc_oop.c nrn.old/src/oc/hoc_oop.c
184,185c184
< #define NTYPESTACK  10000
< /* A.R.     28.12.1998 */
•  #define NTYPESTACK  30
218,219c217
< #define NTEMPLATESTACK  10000
< /* A.R.         28.12.1998 */
•  #define NTEMPLATESTACK  20
Only in nrn.new/src/oc: hoc_oop.c.orig
```

NEURON must be recompiled for the changes to take effect.

6. Appendix 2: List of functions

<u>parse.hoc</u>
```
get()
get_somadist()
connect_axon()
add_axon()
remove_axon()
insert_channels()
make_sectionlists()
isterminal()
make_distvectors()
switch()
dist_switch()
L_switch()
make_vectors()
single_vectors()
set_origin()
```

<u>help.hoc</u>
```
hlp()
hlpscan()
hlpfound()
fxnscan()
check()
consistency()
get_parents()
find_section()
traces()
fxarea()
sectest()
which()
```

<u>electrophysiology.hoc</u>
```
rest()
simcore()
sim()
initsimvclamp()
dvdr_calc()
sim_err()
sim_fit()
rinput_calc()
sim_calc()
threshold_calc()
forwardthreshold_calc()
threshold_find()
threshold()
thresh()
APdecay()
APdecay_sensitivity()
sigmoidal()
sigmoidal_calc()
scrappy()
```

<u>impedance.hoc</u>
```
impedance_calc()
impedance_mismatch()
switch_off_intra()
switch_on_intra()
get_children()
switch_on()
imp_calc()
impedance_check()
get_Zfwdvalues()
get_cZ()
get_APfrequencies()
cosine()
cosinefxn()
cosinefit()
```

<u>forward.hoc</u>
```
name_somadist()
name_halfdecay()
resize_cell()
```

<u>output.hoc</u>
```
batch()
calculation()
manual()
save_geometry()
save_active()
save_cable()
save_forwardmini()
save_all()
save_back()
save_fI()
save_fII()
helpme()
write_numbers()
write_nathreshold()
geometry_read()
active_read()
normforward()
equivforward()
equivforwardII()
printvectors()
printvectors_back()
printvectors_forward()
printvectors_forward2()
make_figures()
equivZfwdwrite()
```

<u>statistics.hoc</u>
```
get_neurondata()
add_vals()
lg()
get_data()
compare_activemodels()
c2()
c3()
c()
dissect()
antidissect()
clean()
correl_raw()
correl_func1()
correl_func2()
correl_func3()
Rcorrelation()
single_corr()
single_corrf()
double_corr()
triple_corr()
clegend()
averages()
label_list()
clabel()
Cplot()
powerplot()
cplot()
get_geomorder()
multiplot()
datalegend()
```

good_corr_func()
good_doublecorr()
checkit()
multi_correlation()
write_singlecorr()

neuronprefs.hoc
add_cell()
cell_name()
set_suffix()
set_spinedensity()
dendII()
dendIII()
swc_format()
make_sectionrefs()

geometry.hoc
fdistance()
fL()
segL()
mindist()
maxdist()
farea()
sectionarea()
fseg()
fbranch()
get_parent()
pbranchpoint()
nextparent()
branchpoint()
get_root()
ubranch()
get_rall()
rall_calc()
ename()
gstep()
make_dAr()
get_gdist()
geometry_calc()
mean()
div()
equivalent_calc()
spinetransform()
make_equivalent_cable()
slope_darea()
slope_deq()
dAdr_calc()
dAdr_write()
deq_calc()
deq_write()
estcore()
tap()
lintaper()
get_link()
set_electrotonic()

graphics.hoc
flip()
pt()
P()
ar()
mx()
mn()
mod()
ceil()
fig()
figlab()
clf()
hist()
gauss()
bar()
sort()
filter()
rolling()
rolling2()
roll()
writevec_el()
writeveca()
readveca()
writevec()
readvec()
writevecs()
nvectors()
nasens()
spaceplot()
show()