NEURON

for computer simulations of neurons and neural networks

The NEURON simulation environment

A brief introduction

Michael Rudolph

Advanced Course in
Computational Neuroscience
Obidos, Portugal, 2004

References

web:

- `http://www.neuron.yale.edu/`
- `http://neuron.duke.edu/`

papers:

- Hines, M.L. and Carnevale, N.T.
  The NEURON Simulation Environment.
  Neural Computation **9** (1997), 1179-1209.
- Hines, M.L. and Carnevale, N.T.
  Expanding NEURON's repertoire of mechanisms with NMODL.
  Neural Computation **12** (2000), 839-851.
- Hines, M.L. and Carnevale, N.T.
  NEURON: a tool for neuroscientists.
  The Neuroscientist **7** (2001), 123-135.

The NEURON book (in preparation)

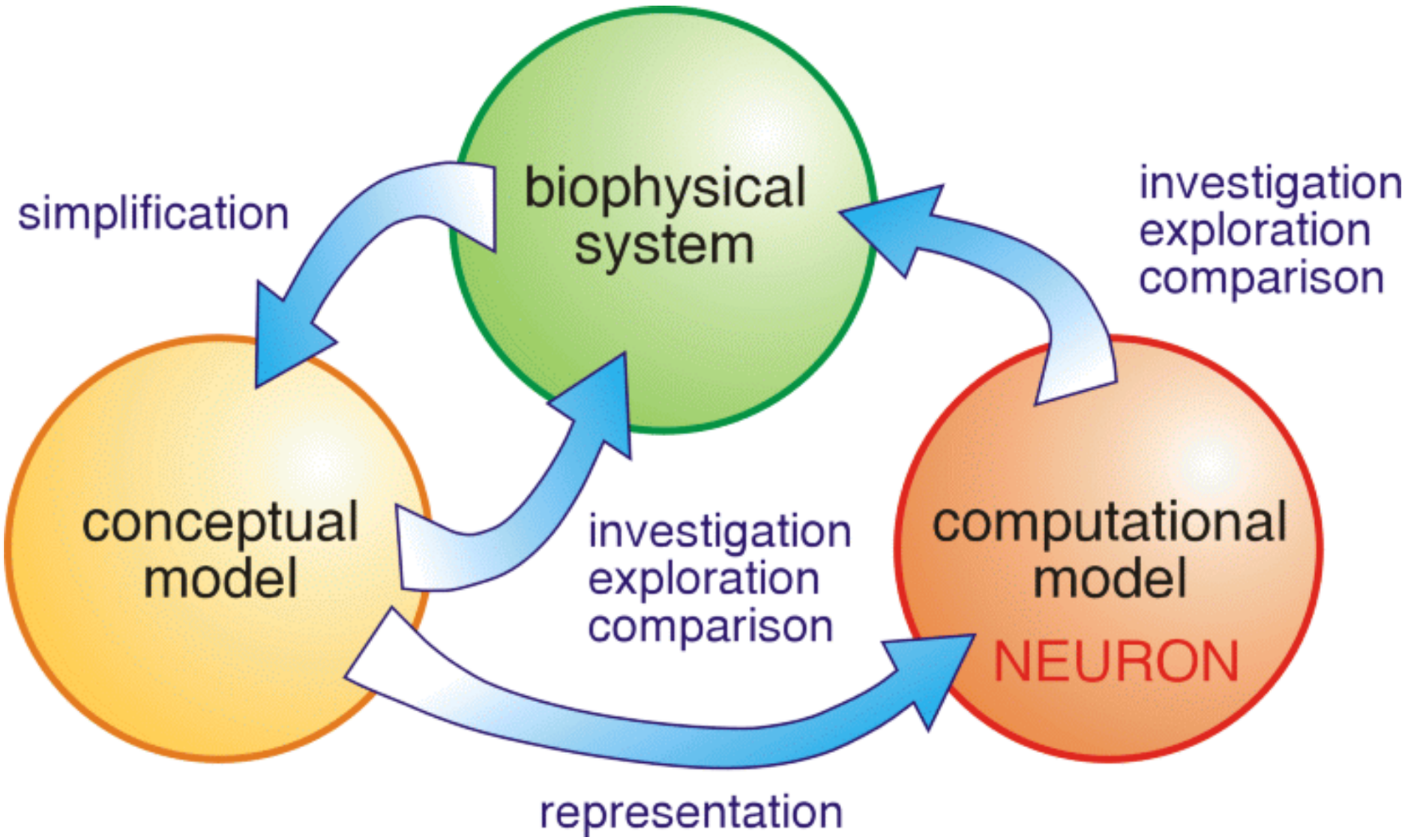
Outline

1. The idea behind NEURON
2. Some mathematics behind NEURON
3. Running NEURON
4. Two programming levels = infinite possibilities
5. A step towards infinity: some examples

1 The idea behind NEURON

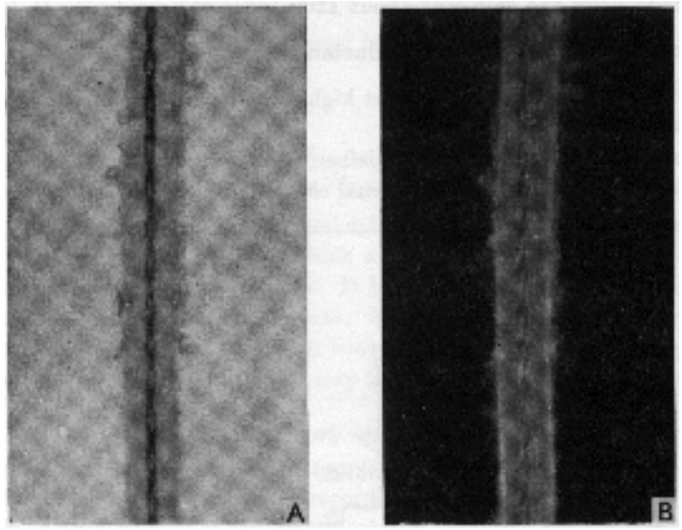
What is NEURON?

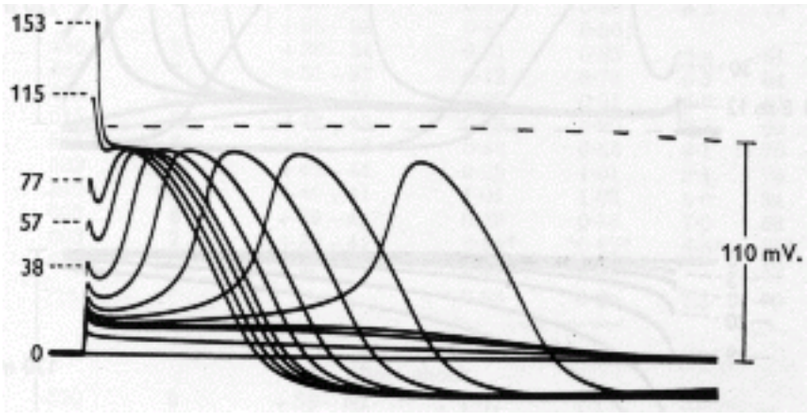
NEURON is a simulation environment for models of biophysical processes, models of individual neurons and networks of neurons.



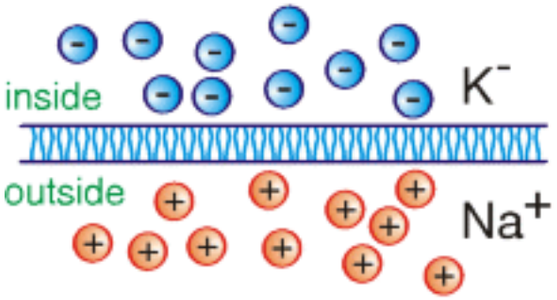
1 The idea behind NEURON

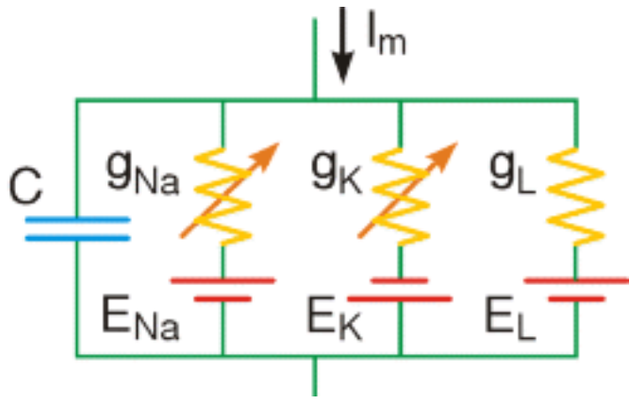
biophysical system



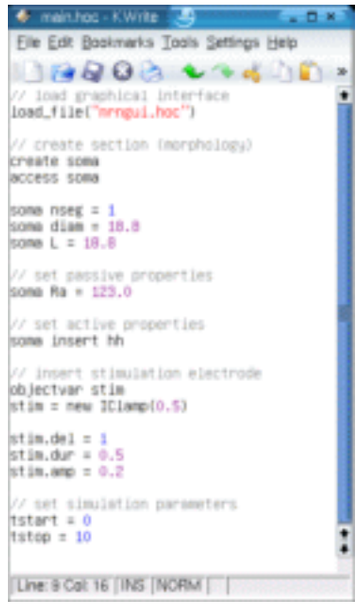


conceptual model



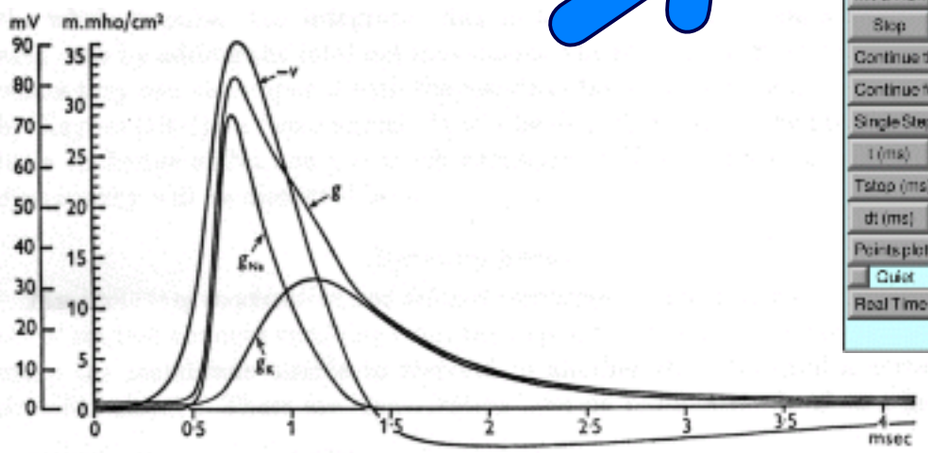


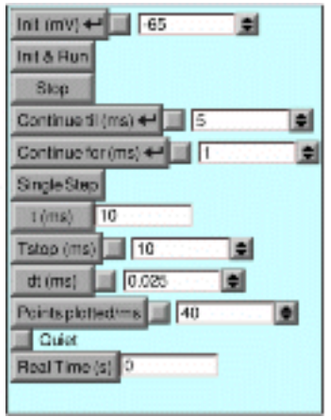
computational model NEURON

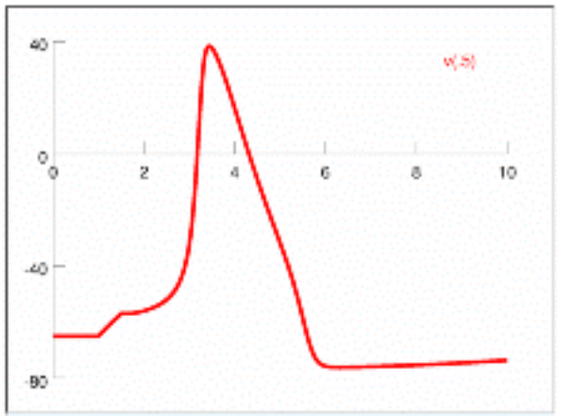


$$C\frac{dV(t)}{dt} = -I_m(t)$$
$$I_m(t) = g_{Na}(t)\left(V(t)-E_{Na}\right) + g_K(t)\left(V(t)-E_K\right) + g_L\left(V(t)-E_L\right)$$
$$g_{Na}(t) = \bar{g}_{Na}\, m^3(t)\, h(t)$$
$$g_K(t) = \bar{g}_K\, n^4(t)$$

$$\frac{dm(t)}{dt} = \alpha_m(V)\left(1-m(t)\right) - \beta_m(V)\, m(t)$$
$$\frac{dh(t)}{dt} = \alpha_h(V)\left(1-h(t)\right) - \beta_h(V)\, h(t)$$
$$\frac{dn(t)}{dt} = \alpha_n(V)\left(1-n(t)\right) - \beta_n(V)\, n(t)$$







1 The idea behind NEURON

What can NEURON do?

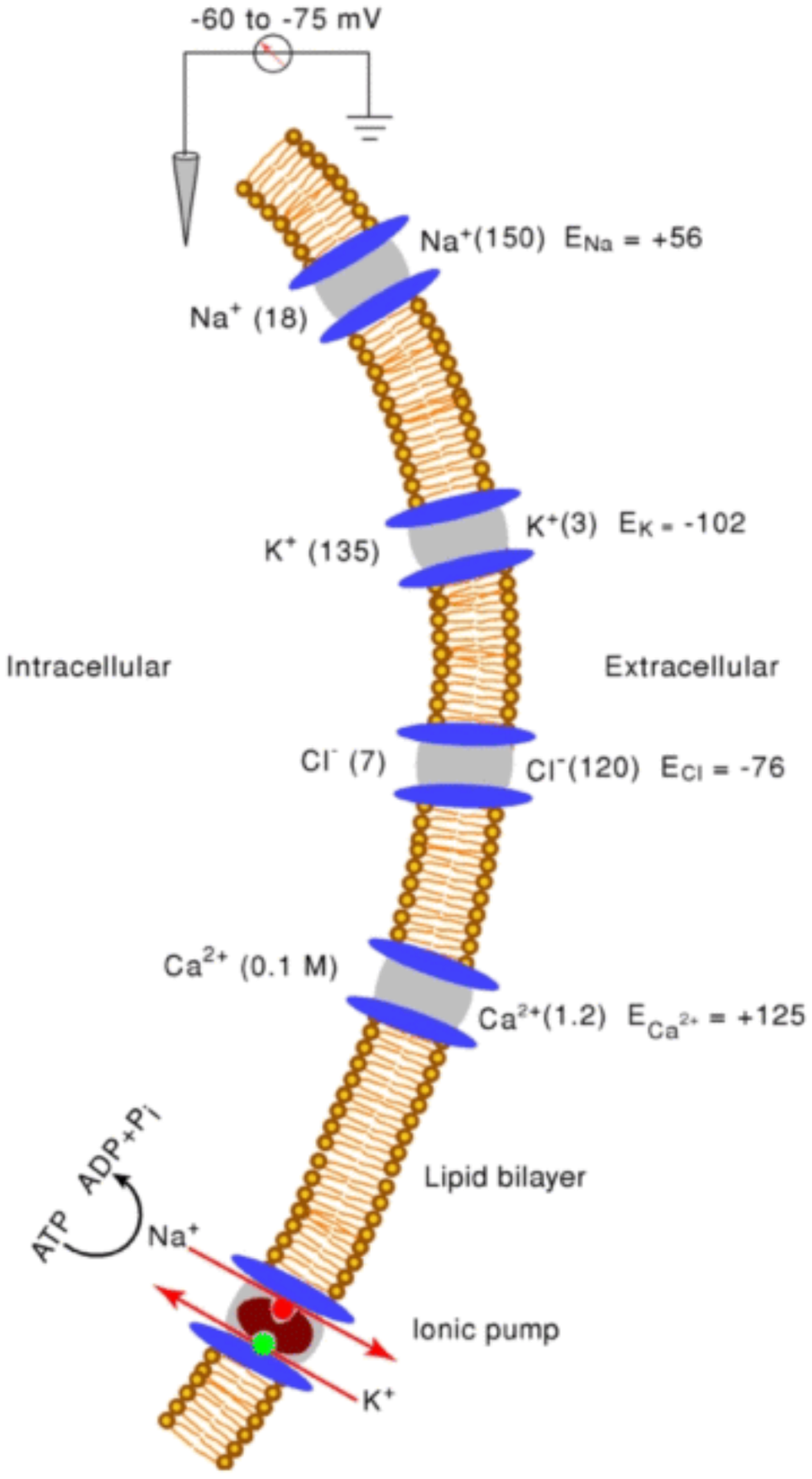
Generally:

NEURON provides tools for constructing, exercising, and managing simplified up to biologically realistic models of electrical and chemical signaling in neurons and networks of neurons.

Specifically:

NEURON can simulate:

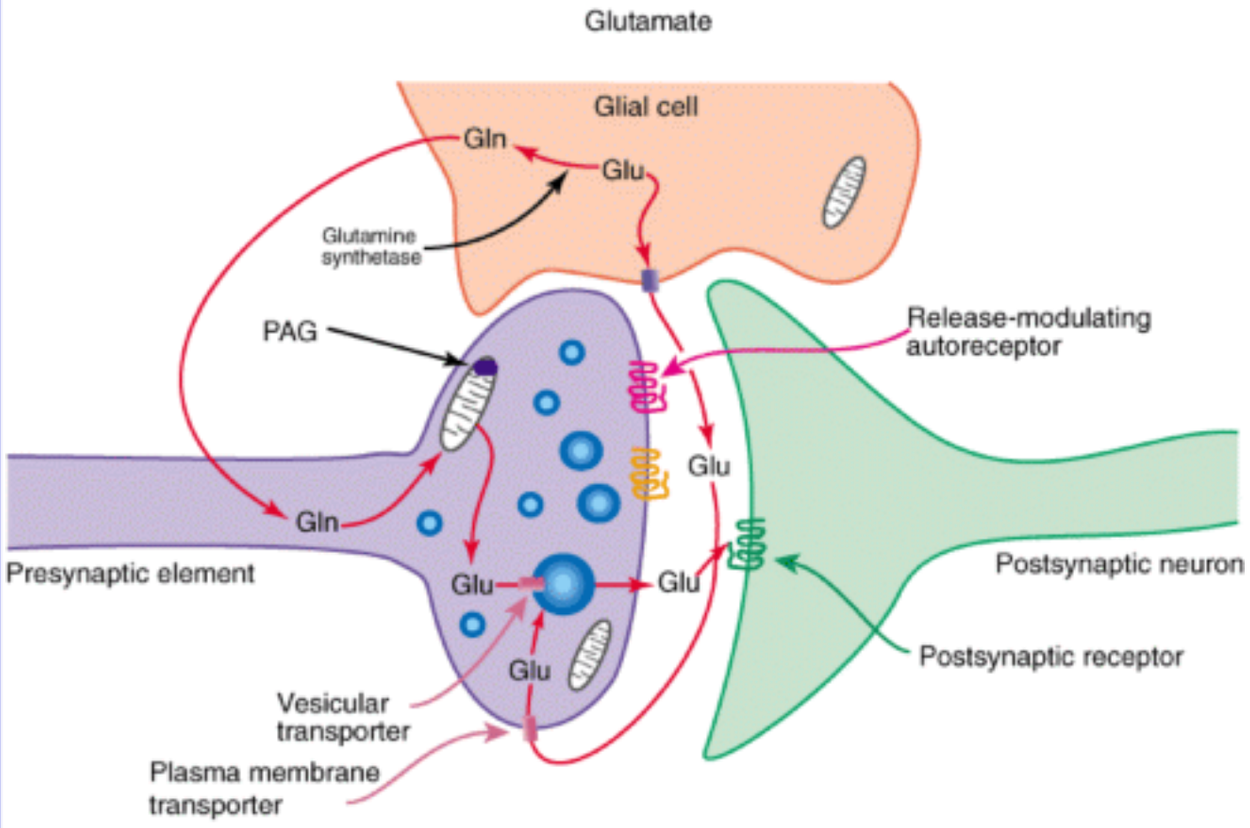
- biophysical and biochemical dynamics of active membrane properties (transmembrane currents, transmembrane channels)

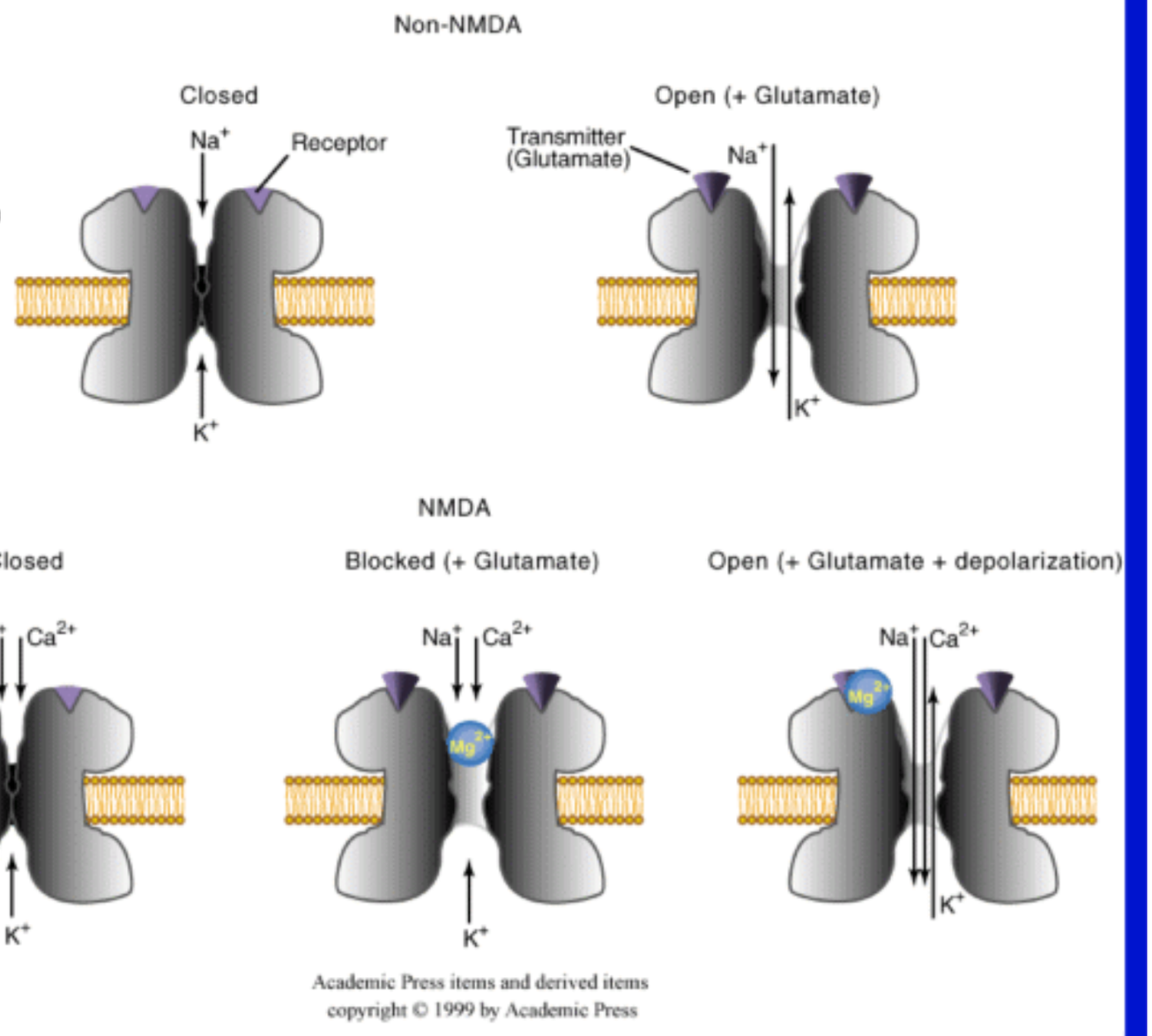


[*Fundamental Neuroscience*. Zigmond *et al.* (eds.)]

1 The idea behind NEURON

- biophysical and biochemical dynamics of synaptic transmission

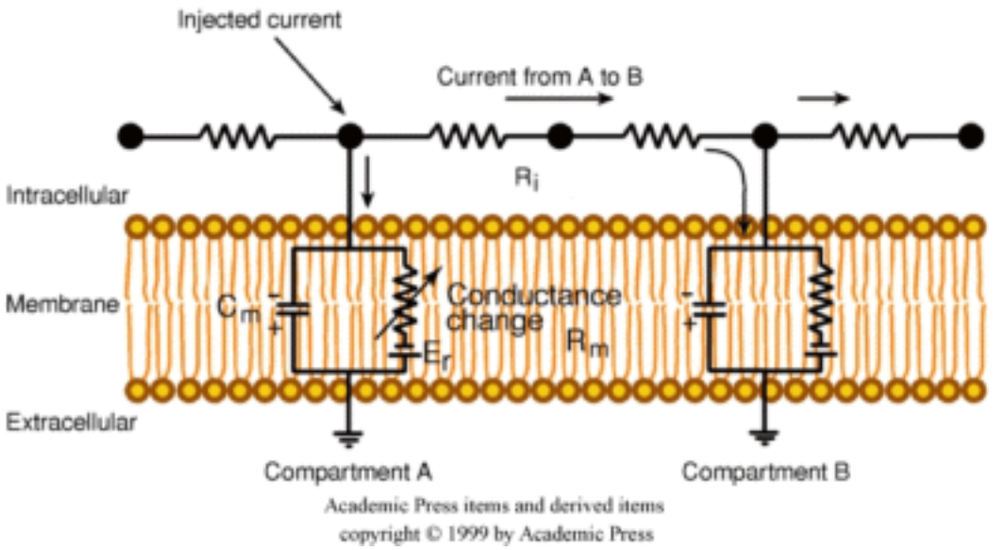


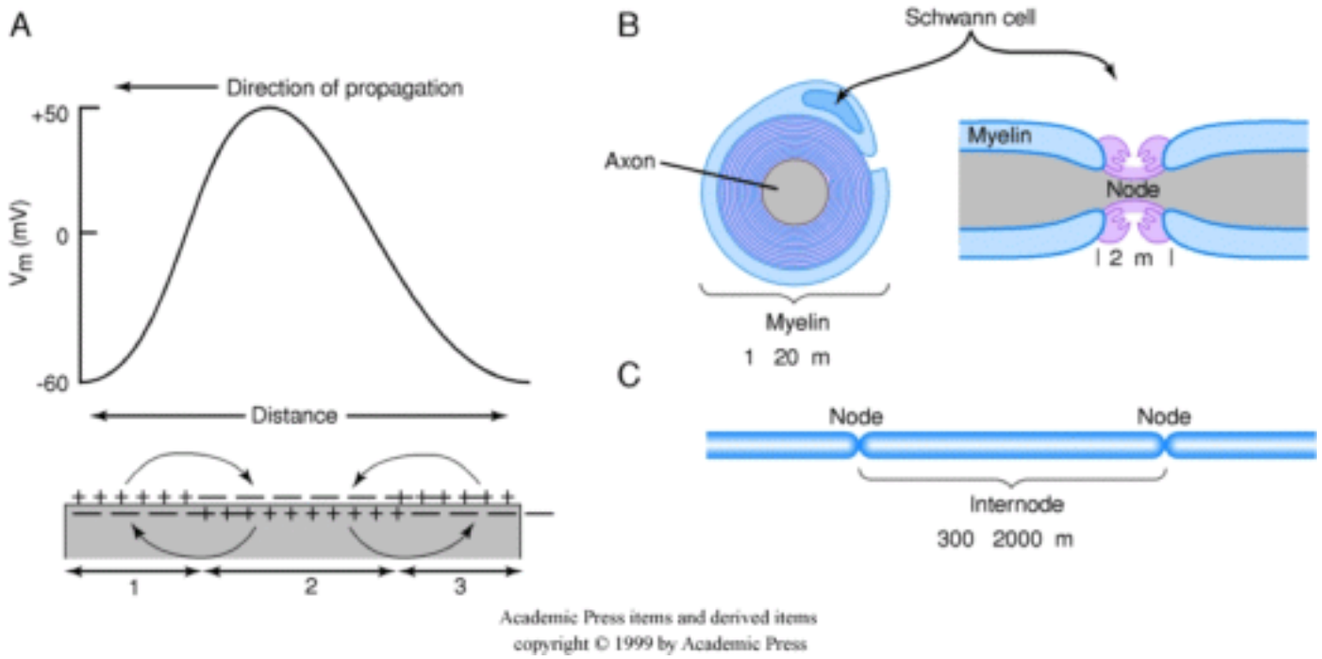


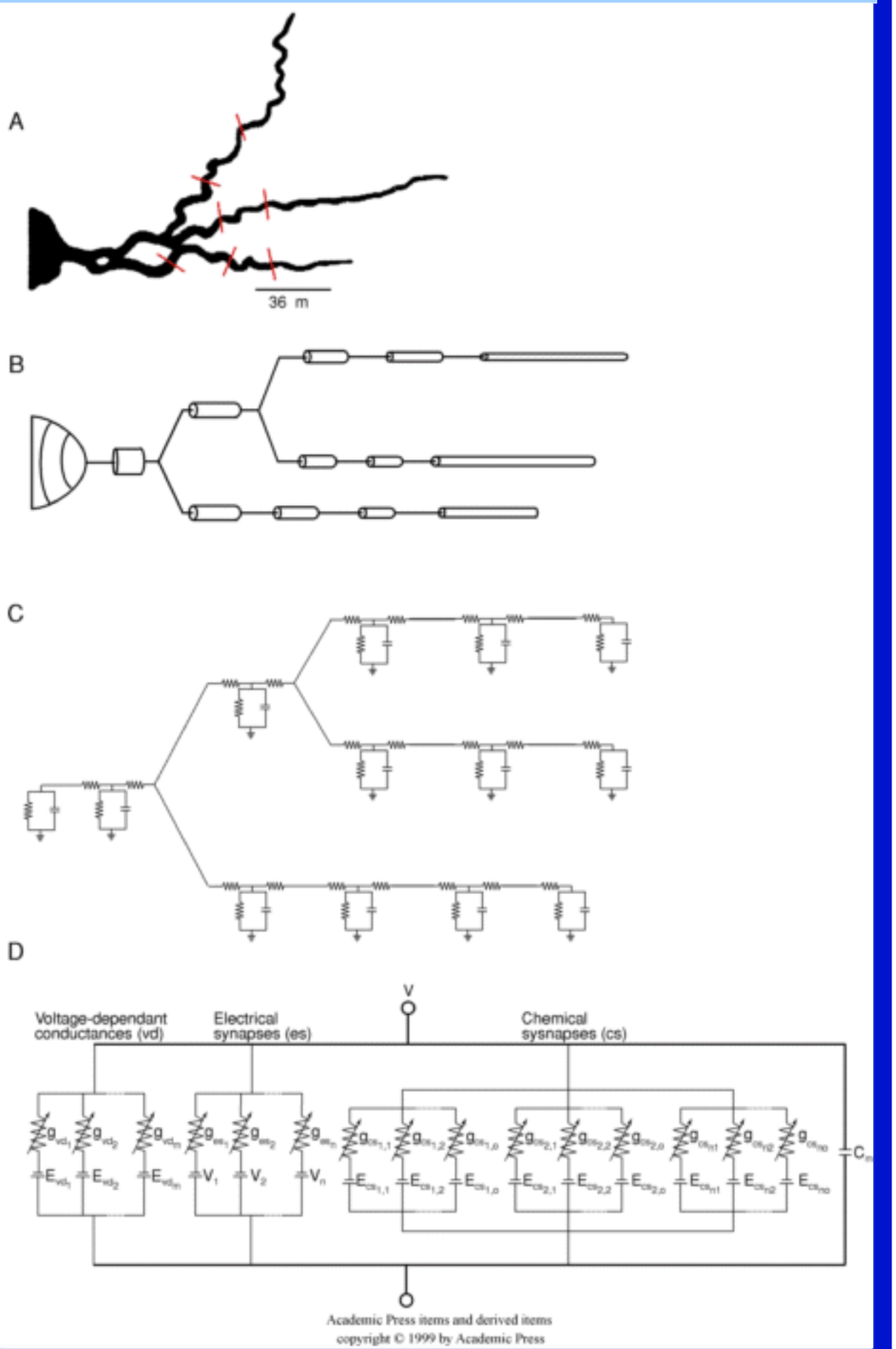
[*Fundamental Neuroscience*. Zigmond *et al.* (eds.)]

1 The idea behind NEURON

- electrotonic and active signalling along dendritic and axonal cables, as well as in cable structures with complex branching morphology



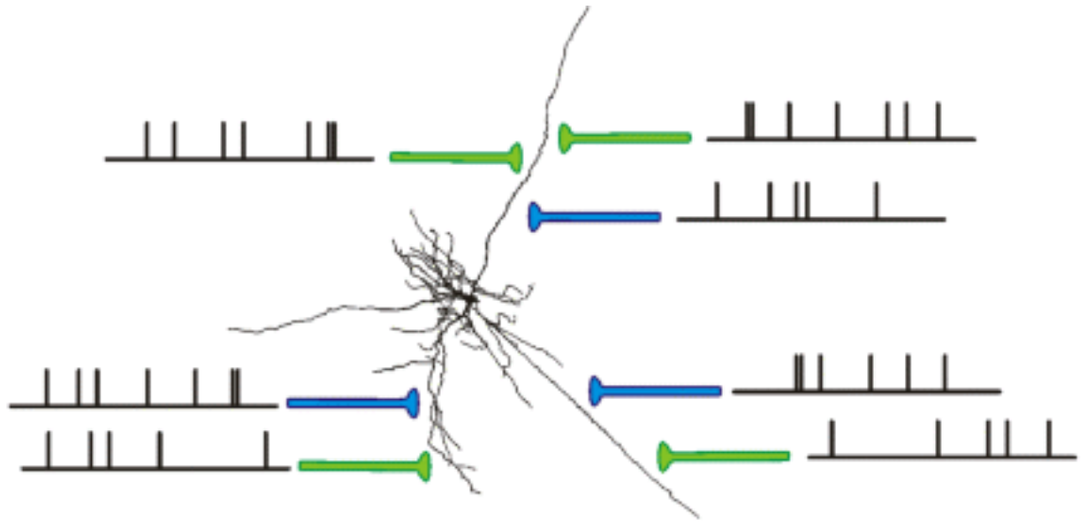


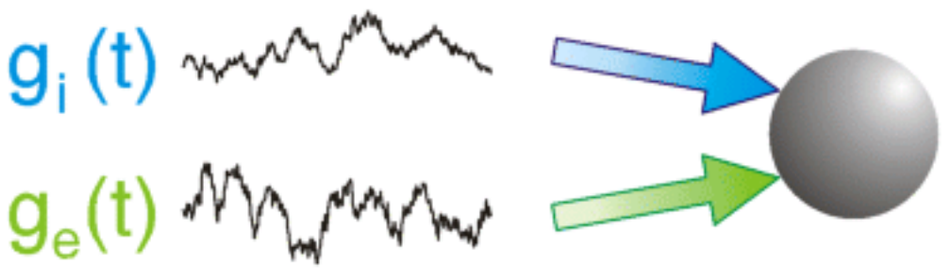


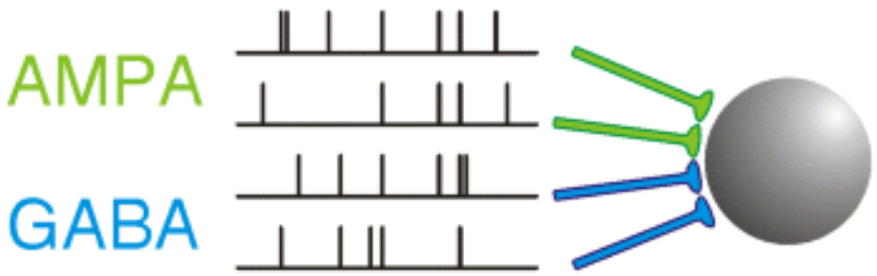
[*Fundamental Neuroscience*. Zigmond *et al.* (eds.)]

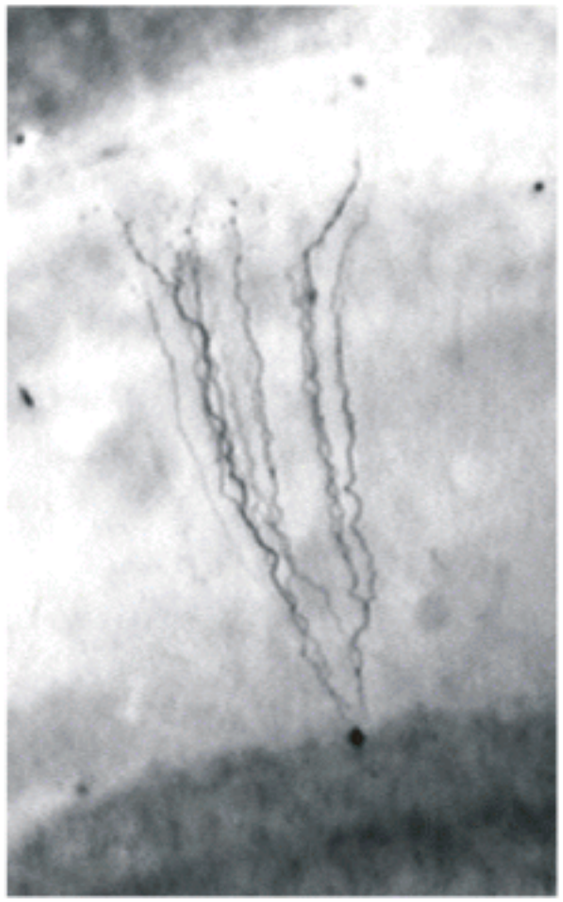
1 The idea behind NEURON

- information processing in simplified up to morphologically and biophysically realistic neuronal models

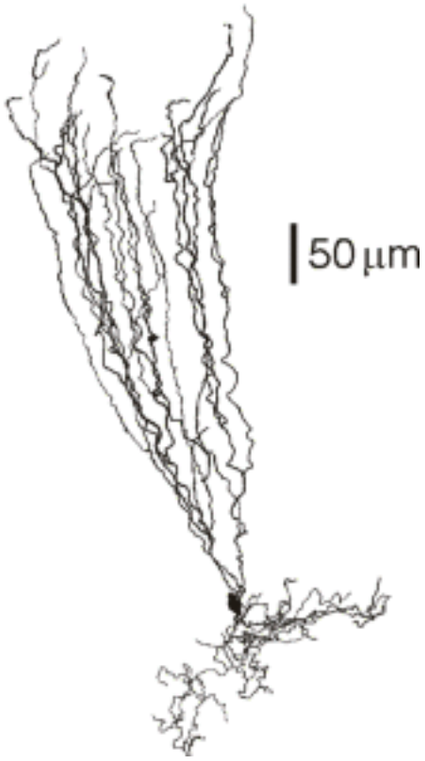






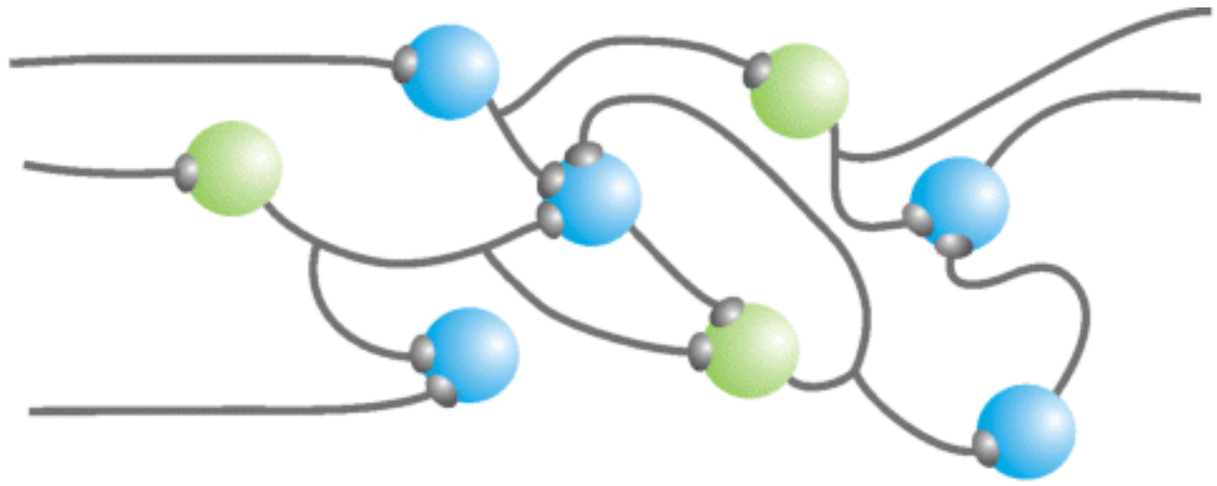


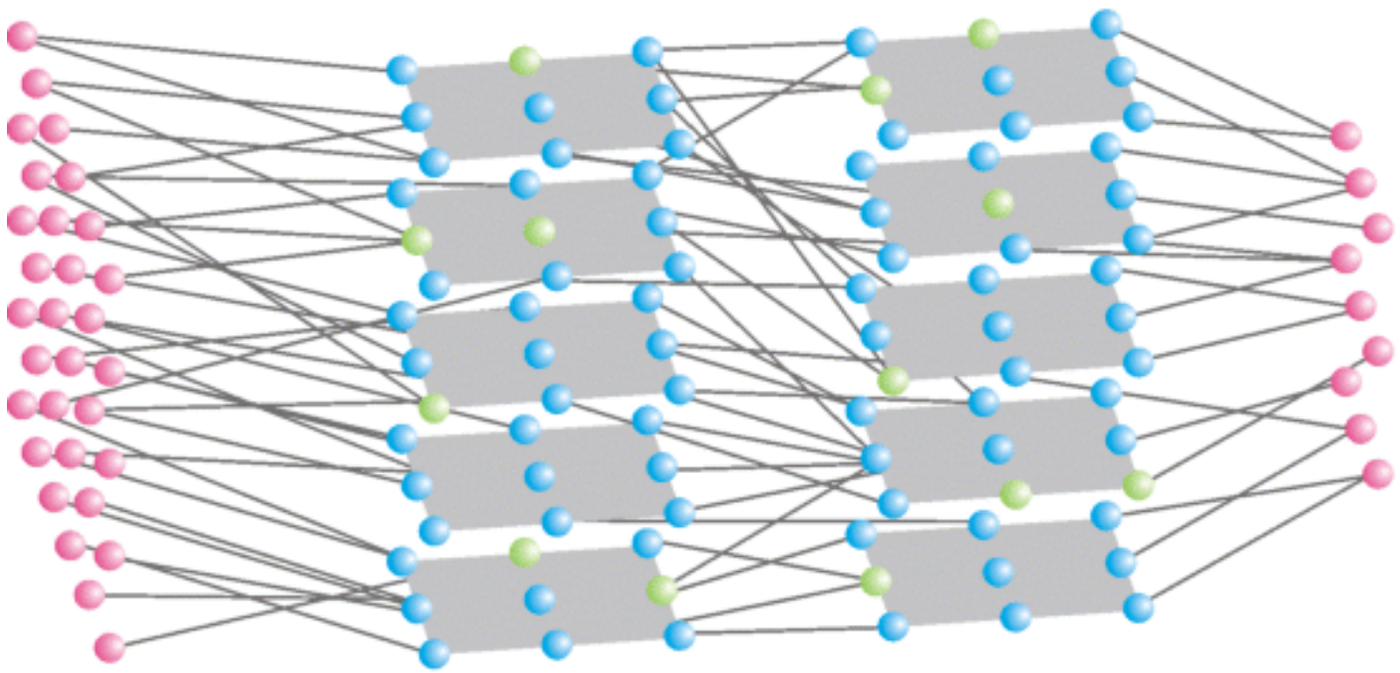
(ganglion cell of mormyrid ELL)

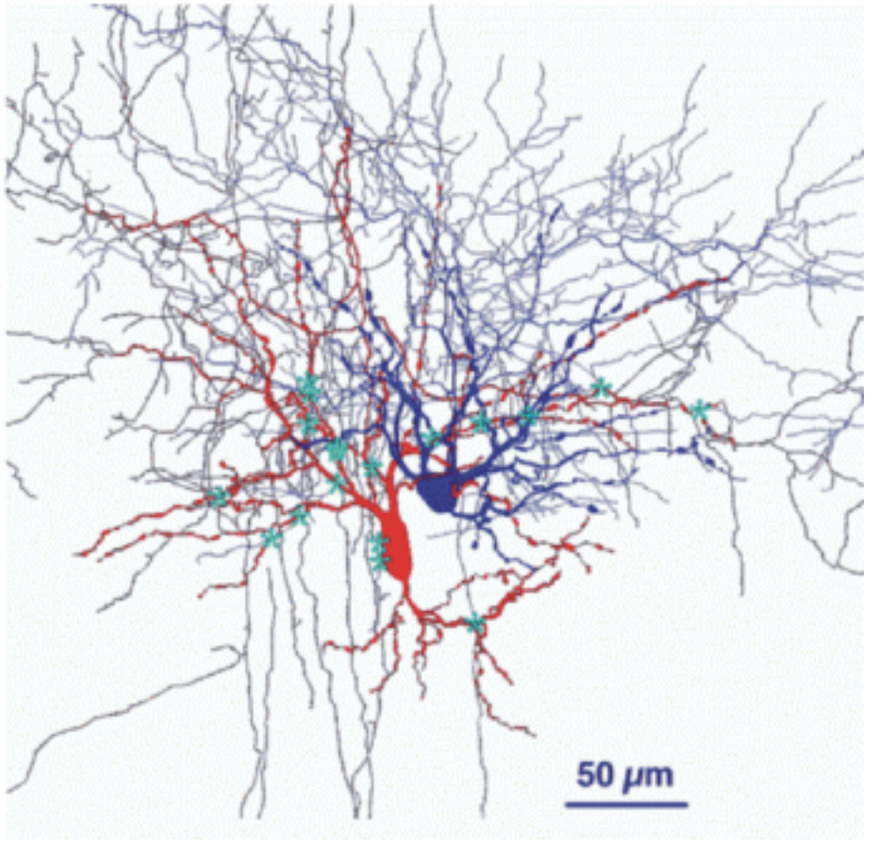


1 The idea behind NEURON

- information processing in neuronal microcircuits and large networks of inter-connected neurons (ranging from simplfied models, such as IAF, up to more realistic neuronal models)





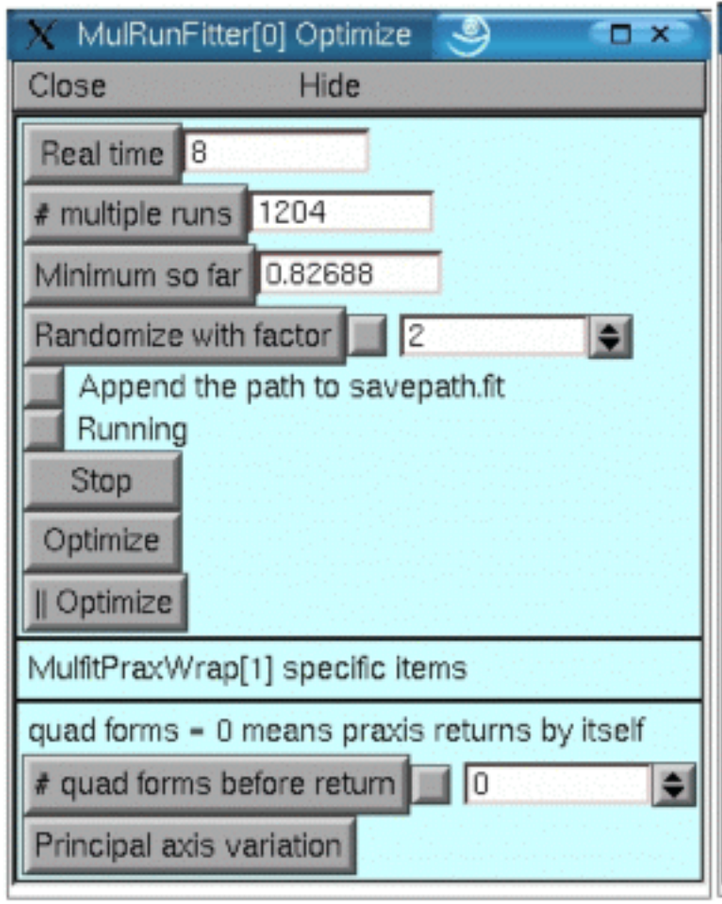


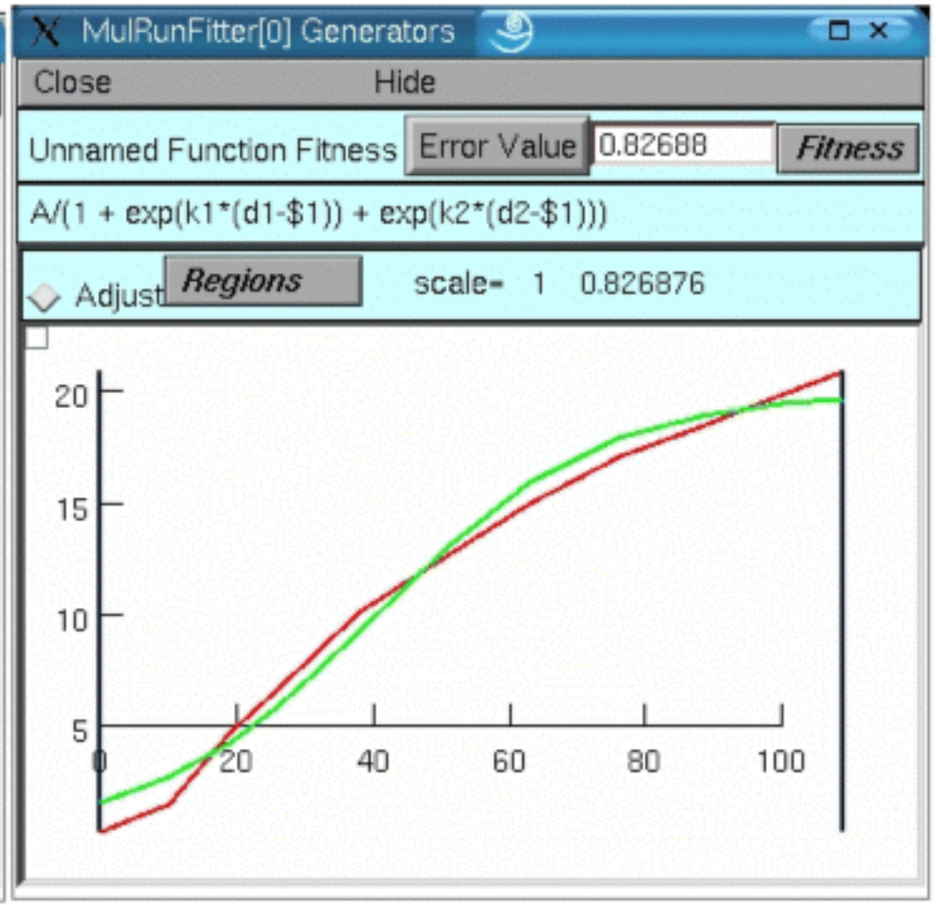
1 The idea behind NEURON

Moreover:

NEURON provides:

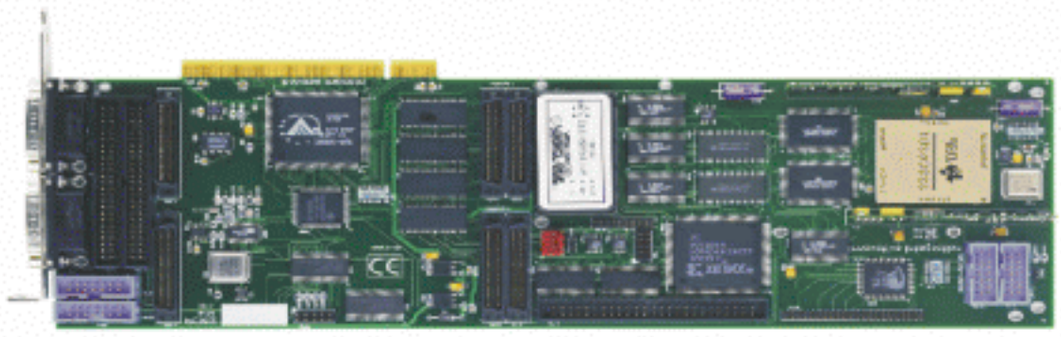
- tools for analyzing experimental data



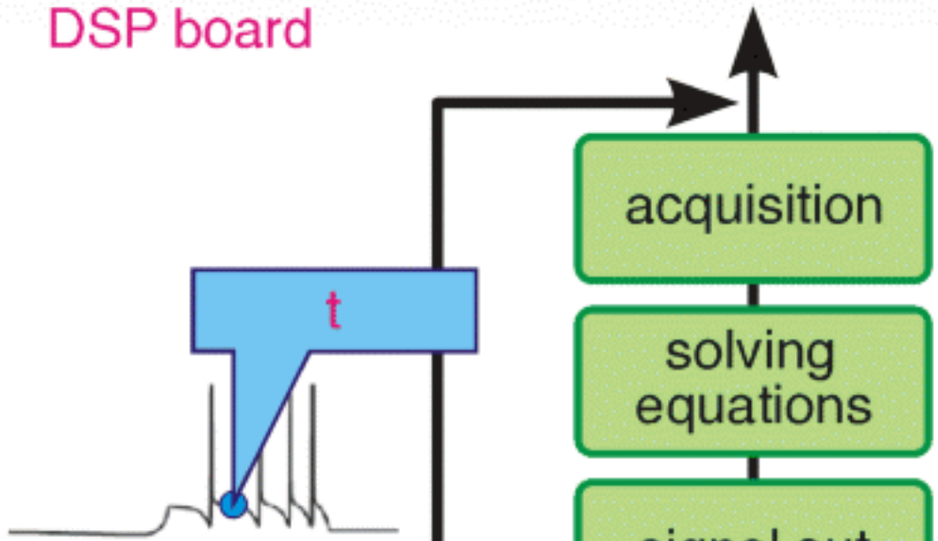


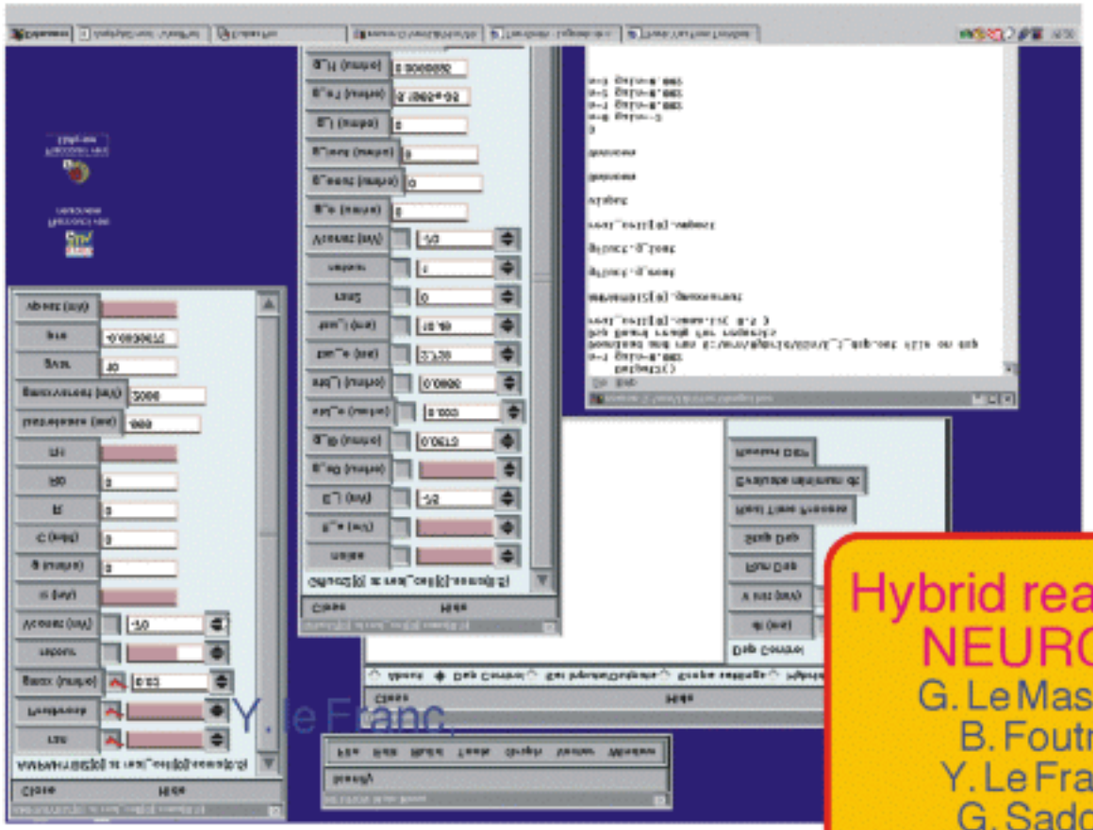
1 The idea behind NEURON

- tools for interfacing models in computo with biological neurons in vitro and in vivo (e.g. dynamic clamp)

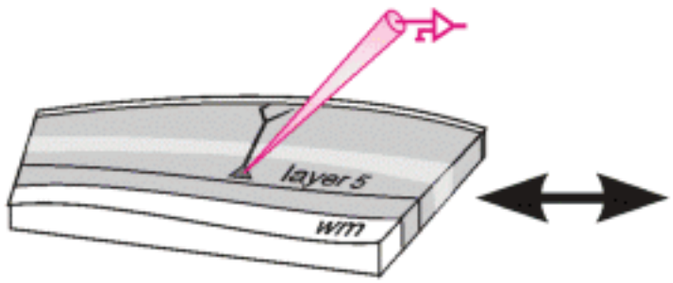


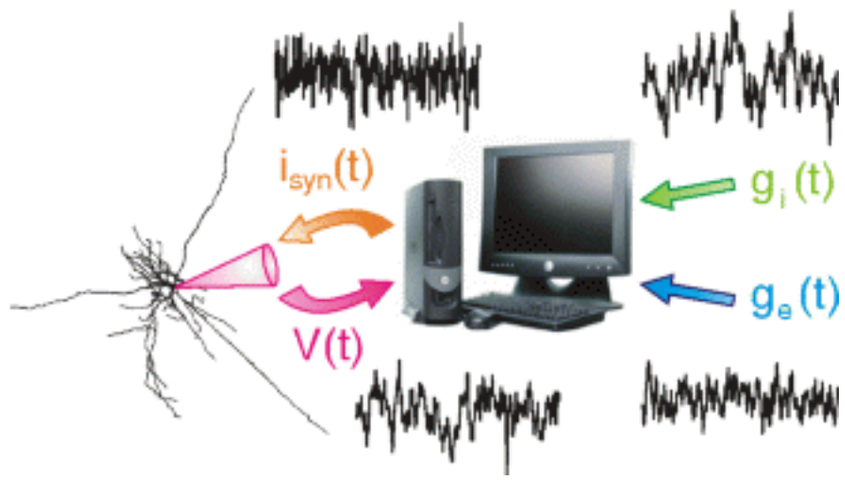
DSP board



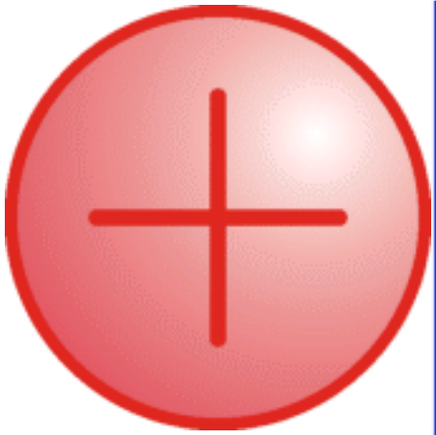


NEURON interface





1 The idea behind NEURON

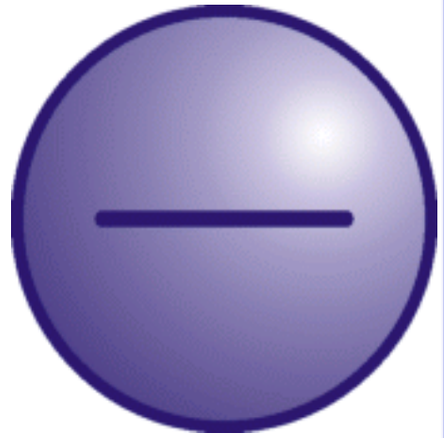


Why using NEURON?

- (very) easy to use
- suited for computational models on many levels (subcellular, cellular, network)
- well suited for computational models that are closely linked to experimental data
- computationally efficient and accurate while at the same time minimizing the required user effort
- optimized for handling tree-shaped cable structures
- optimized network simulations utilizing the event-based approach
- user is not required to translate the problem into another domain, instead user is able to deal directly with concepts that are familiar at the neuroscience level

1 The idea behind NEURON

Why **not** using NEURON?



Hmmmmmm... ???

And in the case of questions?

- Michael Hines `michael.hines@yale.edu`
- John W. Moore `jwm@neuro.duke.edu`
- Ted Carnevale `ted.carnevale@yale.edu`
- **user group** `listserv@biomed.med.yale.edu`

Outline

1. The idea behind NEURON
2. Some mathematics behind NEURON
3. Running NEURON
4. Two programming levels = infinite possibilities
5. A step towards infinity: some examples

❷ Some mathematics behind NEURON

NEURON's basic computational task:

to numerically solve the cable equation which describes the relationship between current and voltage in an one-dimensional cable

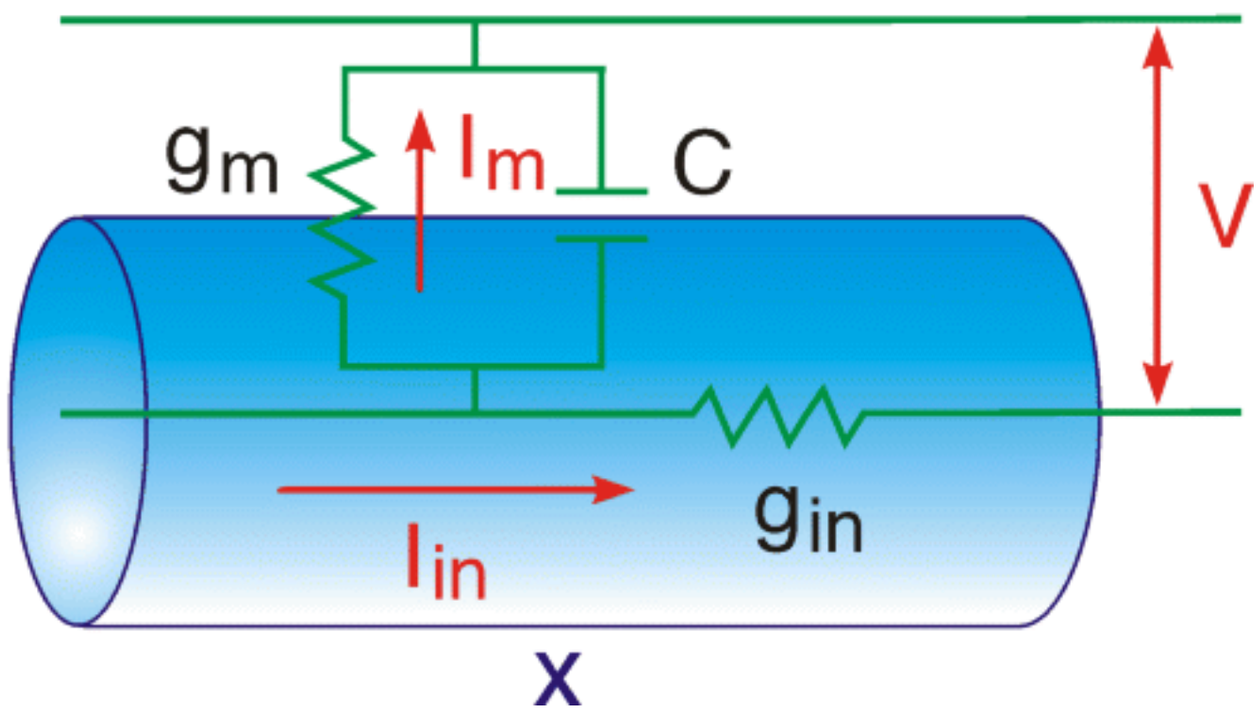
$$g_{in}\frac{\partial^2 V(t,x)}{\partial x^2} = I_m(t,x) = C\frac{\partial V(t,x)}{\partial t} + g_m V(t,x)$$

V membrane potential

$V = V_{in} - V_{out}$

C membrane capacity
g_m membrane conductance
g_{in} intracellular (axial) conductance

I_m membrane current
I_{in} intracellular (axial) current

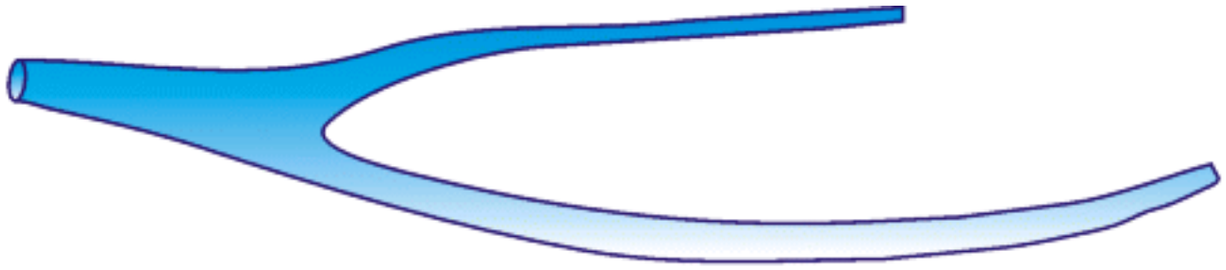


❷ Some mathematics behind NEURON

Principal problem of computational modeling:

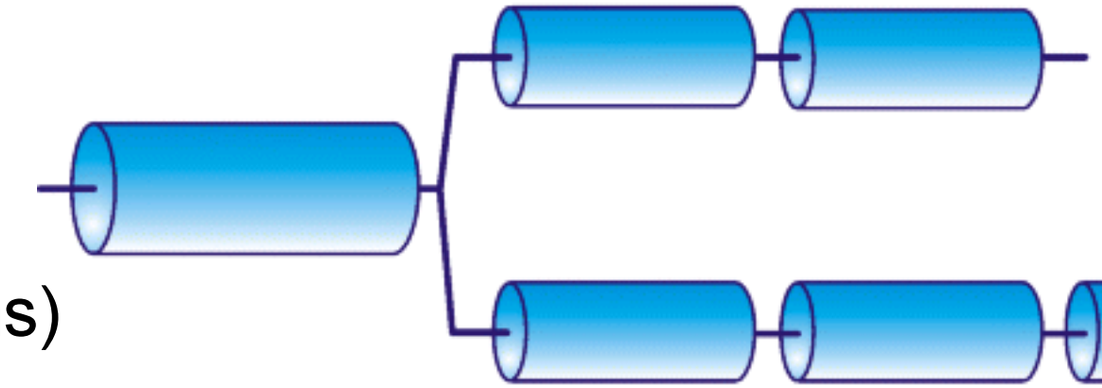
in reality: space and time are continuous variables and dynamics is described by differential equations

in computo: explicit solution of describing differential equations is often not available

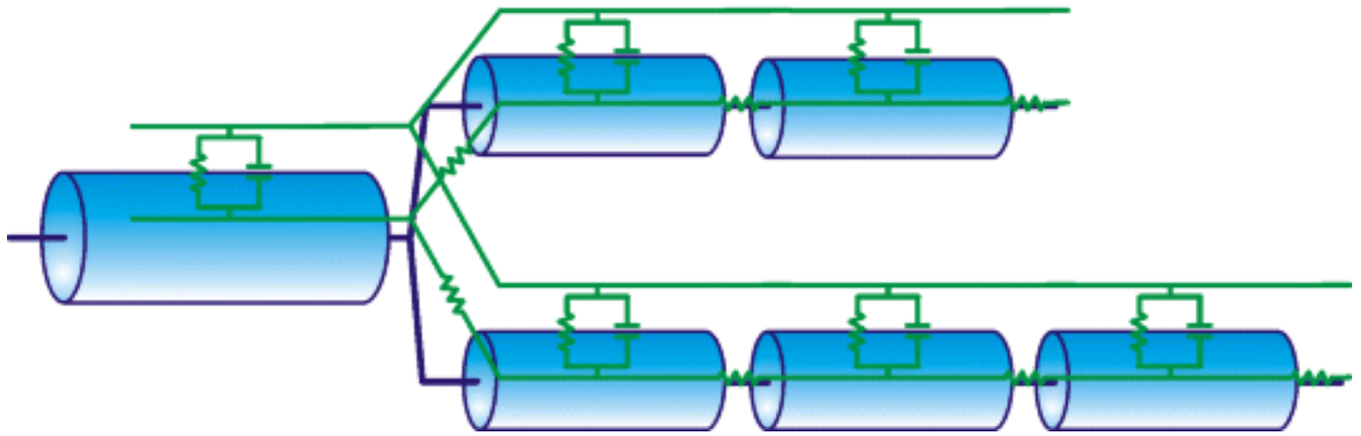


Solution:

approximate continuous system by one which is discontinuous in space and time (dynamics described by difference equations)



compartmentalization
(spatial discretization)
temporal discretization

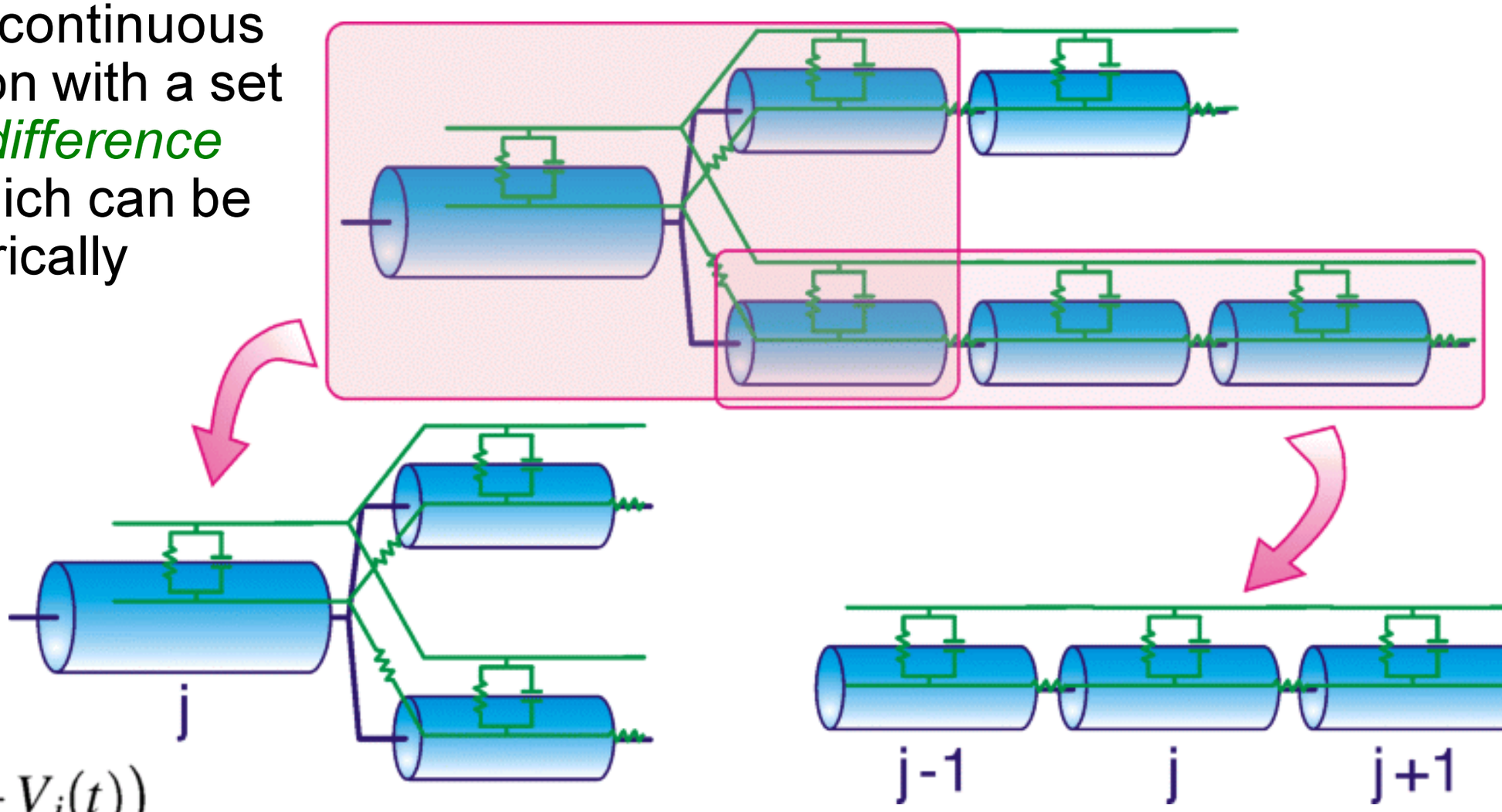


compartmental modeling

② Some mathematics behind NEURON

Compartmental modeling:

approximate continuous cable equation with a set of *algebraic difference equations* which can be solved numerically



$$\sum_k g_{in\,jk}\left(V_k(t)-V_j(t)\right) = C_j\frac{dV_j(t)}{dt}+g_j(t)V_j(t)$$

$$g_{in\,j-1j}\left(V_{j-1}(t)-V_j(t)\right)+g_{in\,jj+1}\left(V_{j+1}(t)-V_j(t)\right) = C_j\frac{dV_j(t)}{dt}+g_j(t)V_j(t)$$

❷ Some mathematics behind NEURON

compartmentalization (spatial discretization) ⇒ reduction of cable equation to a set of ordinary differential equations with first order derivatives in time

temporal discretization ⇒ algebraic difference equations which can be solved numerically

Methods of numerical integration:

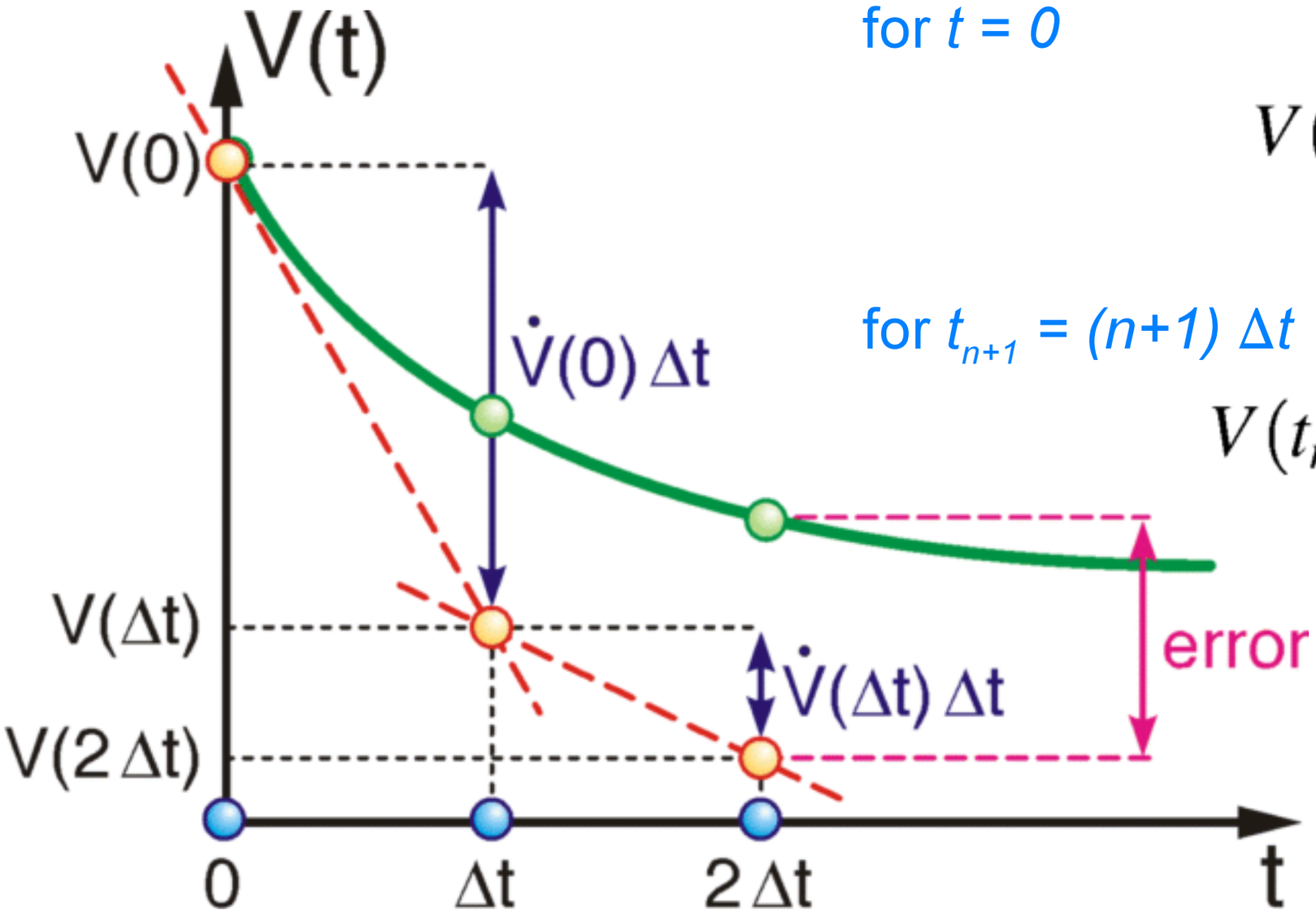
- forward Euler method
  (simple, unstable, inaccurate)
- backward Euler method
  (inaccurate, stable)
- Crank-Nicholson method
  (stable, more accurate)
- ...

} used in NEURON

⇒ selection of a method of numerical integration is guided by concerns of stability, accuracy and efficiency

An example for numerical integration: the forward Euler method

basic idea: extrapolation of function *V(t)* from the value at time *t* to a new value a brief interval Δt into the future



for $t = 0$

$$V(0), \dot{V}(0)$$

$$V(\Delta t) = V(0) + \dot{V}(0)\,\Delta t$$

for $t_{n+1} = (n+1)\,\Delta t$

$$V(t_n), \dot{V}(t_n)$$

$$V(t_{n+1}) = V(t_n) + \dot{V}(t_n)\,\Delta t$$

Outline

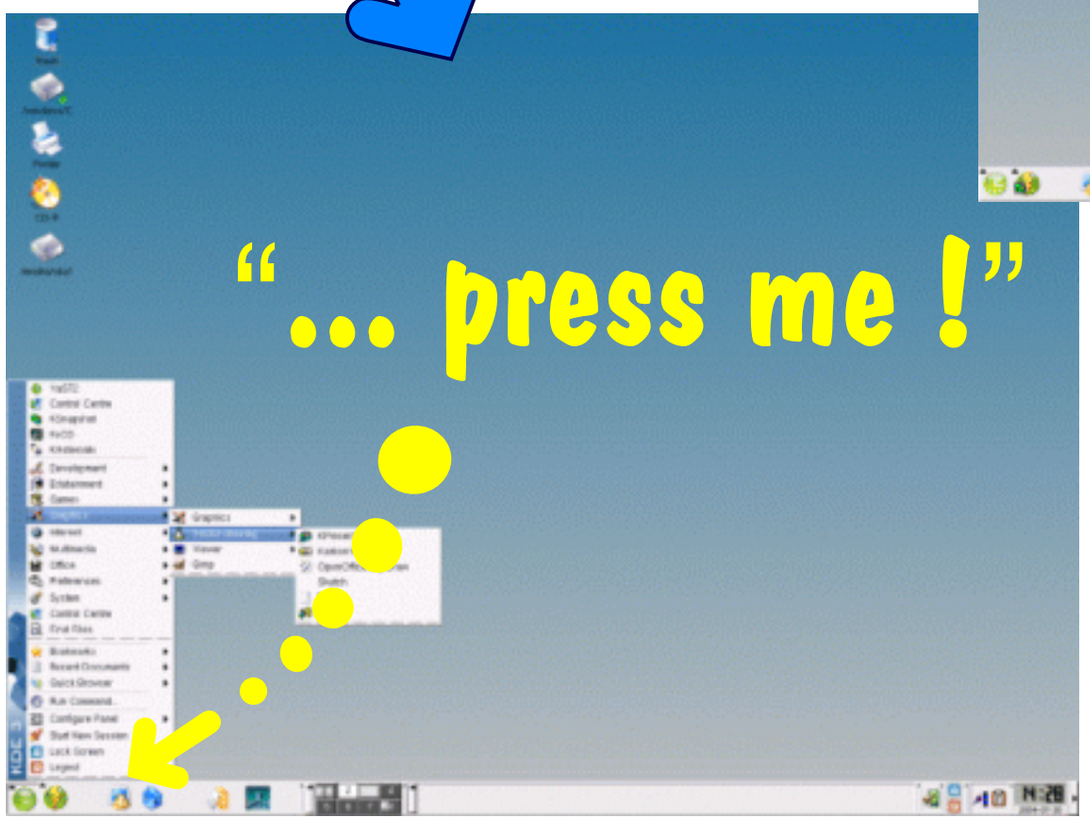
1. The idea behind NEURON
2. Some mathematics behind NEURON
3. Running NEURON
4. Two programming levels = infinite possibilities
5. A step towards infinity: some examples

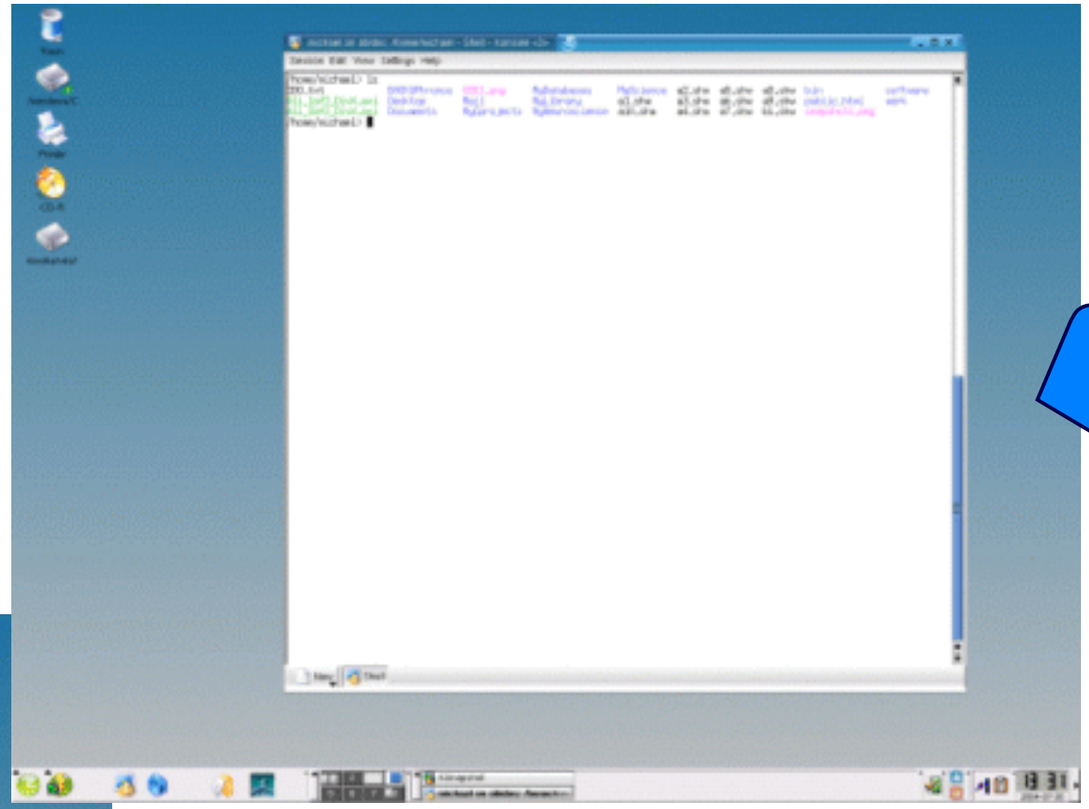
3 Running NEURON

A LINUX environment is the natural habitat of the NEURON simulation environment

- UNIX command shells

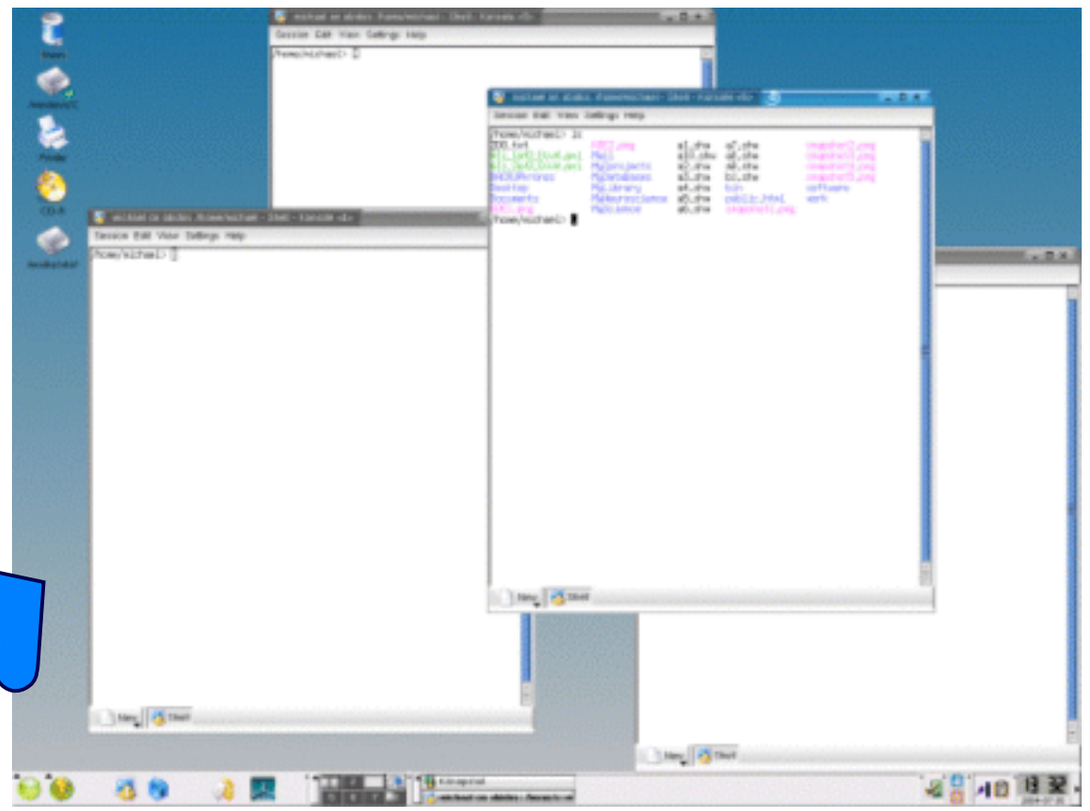
UNIX windows manager (e.g. KDE)





KDE with one shell

KDE with many shells



listing contents of directories / folders

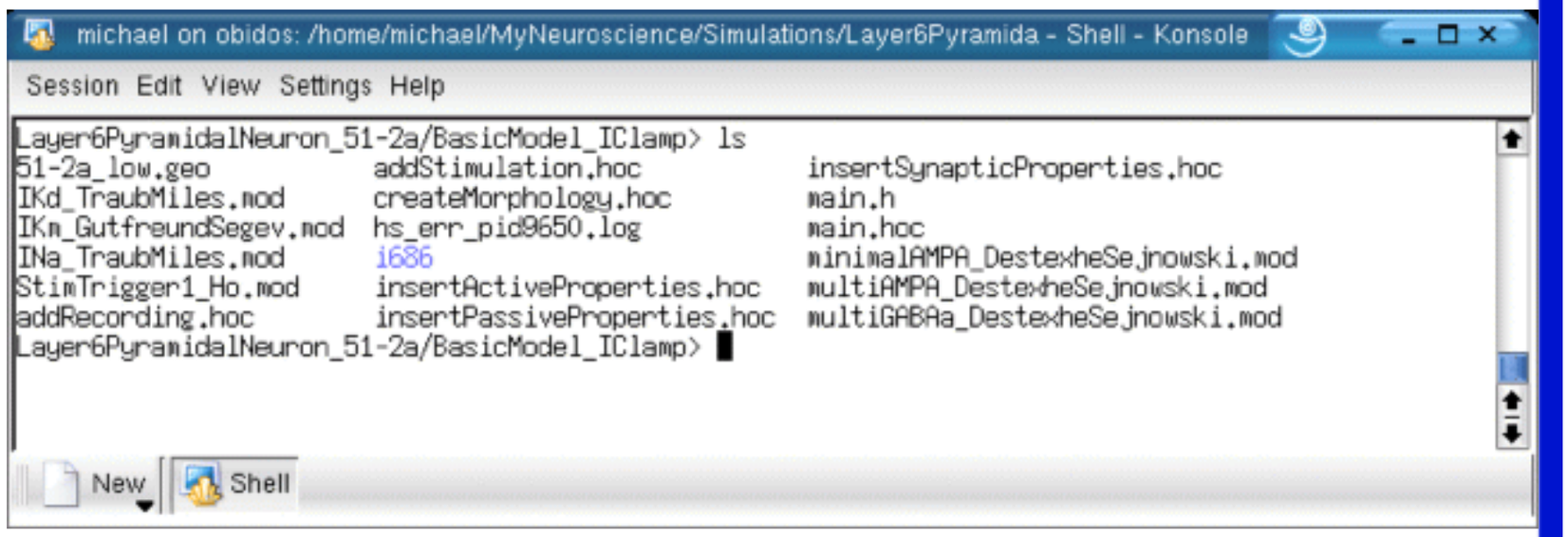
- list names of files / directories:

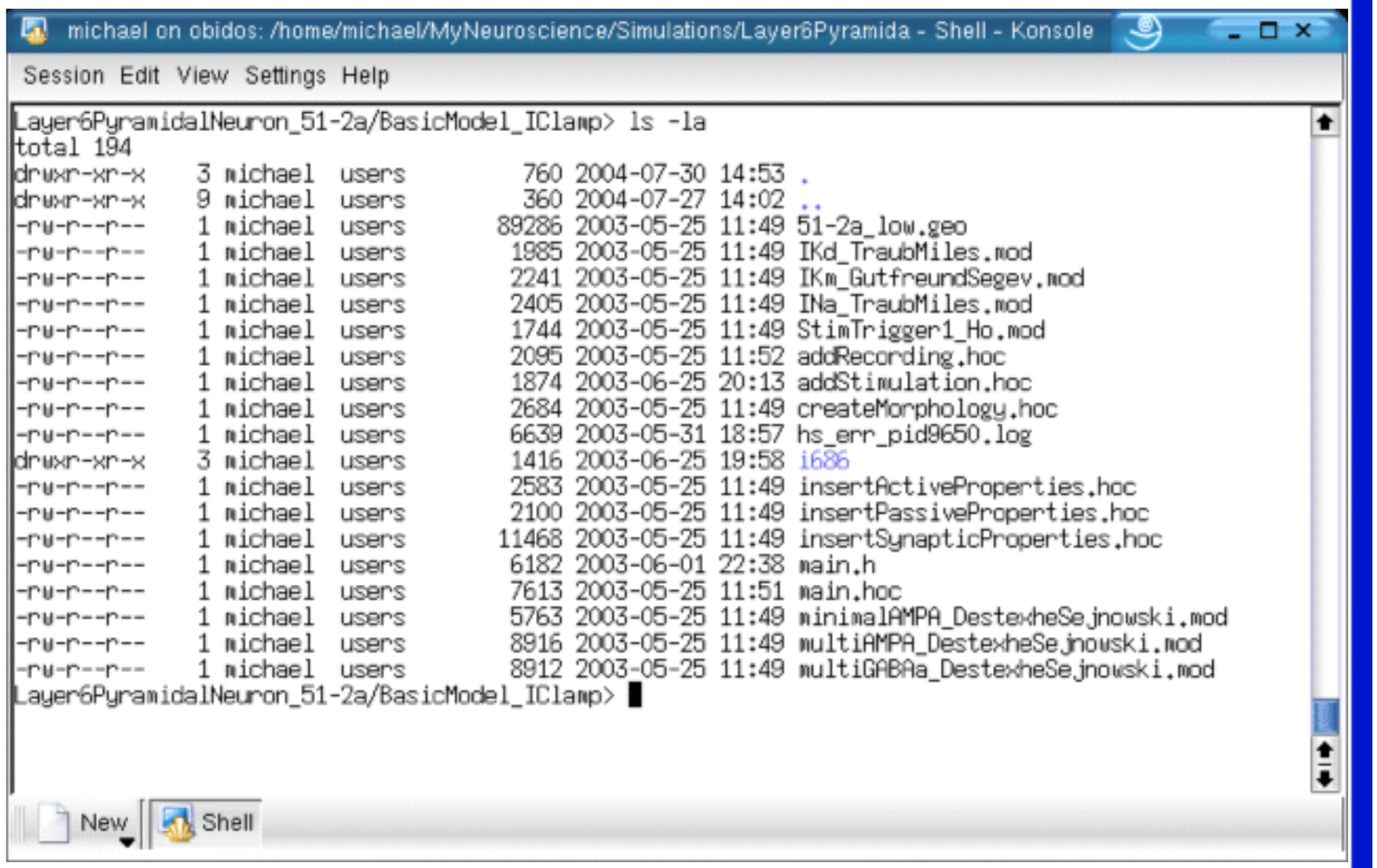
```
ls
```

- detailed list of files / directories:

```
ls -la
```

{ `ls` comes from "`list`" }





permission attributes

```
drwxr-xr-x 3    michael users   1416   2003-06-25 19:58   i686
-rw-r--r-- 1    michael users   7613   2003-05-25 11:51   main.hoc
```

entry type
- `d` directory
- `l` symbolic link
- `-` regular file

permission attributes of file / directory

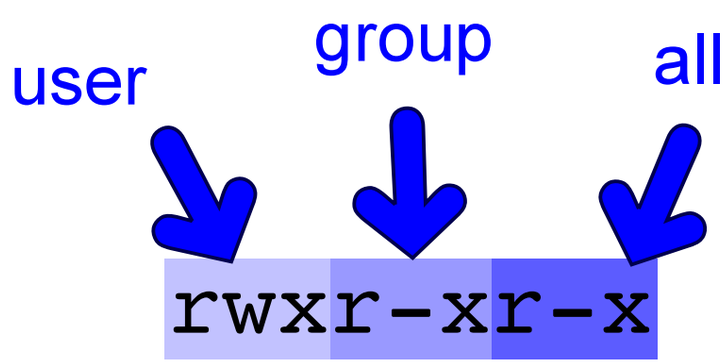
user / owner of file

group of file user / owner

file size (in Byte)

date and time of last modification

file name



- `r` read permission
- `w` write permission
- `x` permission to execute

- changing directories / folders
  - change to home directory: `cd`
  - change one directory up: `cd ..`
  - change to specific directory: `cd <name>`

{ `cd` comes from "`change directory`" }

- creating new directories / folders
  - create directory with specific name: `mkdir <name>`

{ `mkdir` comes from "`make directory`" }

running programs

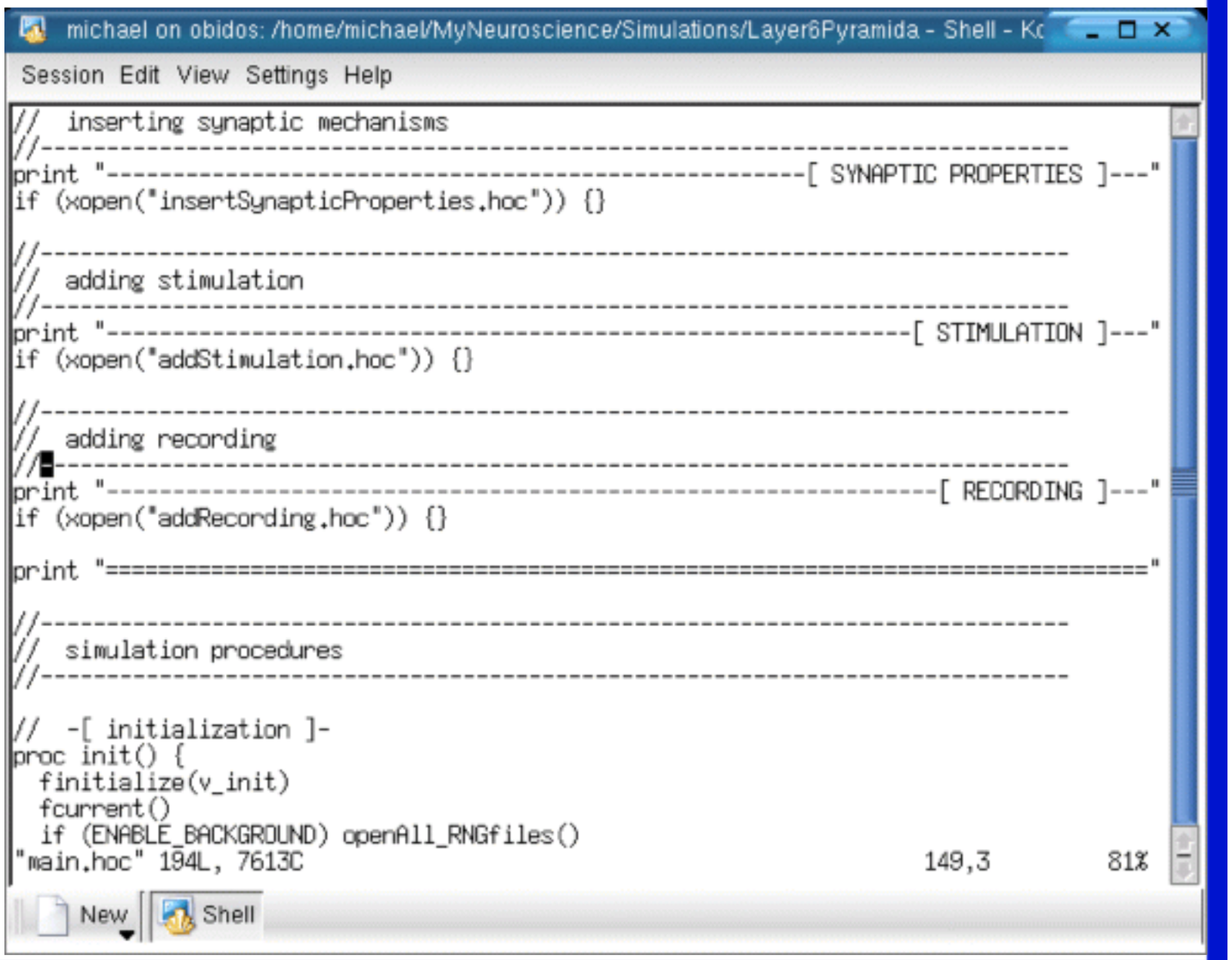
run programs by typing name into command shell, e.g.

```
/home/michael> mathematica
/home/michael> nrnmodl
```

editors

- for hardcore users:

```
vi
```



❸ Running NEURON

michael on obidos: /home/michael/MyNeuroscience/Simulations/Layer6Pyramida - Shell - Kc

Session Edit View Settings Help

```
    IW   main.hoc                          Row 177  Col 1    8:37  Ctrl-K H for help
print "-------------------------------------------------------------[ RECORDING ]---
if (xopen("addRecording.hoc")) {}

print "================================================================================

//--------------------------------------------------------------------------------
//  simulation procedures
//--------------------------------------------------------------------------------

//  -[ initialization ]-
proc init() {
  finitialize(v_init)
  fcurrent()
  if (ENABLE_BACKGROUND) openAll_RNGfiles()
  initAll_RNGdata()
  rec_recordVm()
}

//  -[ single step ]-
proc step() { local i
  if (ENABLE_GRAPHICAL_INTERFACE) Plot()
  if (ENABLE_BACKGROUND) updateAll_RNGdata()
  for i=1,nstep_steprun { advance() }
  rec_recordVm()
  if (ENABLE_BACKGROUND) resetAll_RNGdata()
}

//--------------------------------------------------
```

New | Shell

- for efficient users:

  `joe`

- for Windows©ers:

  `kwrite`

main.hoc - KWrite

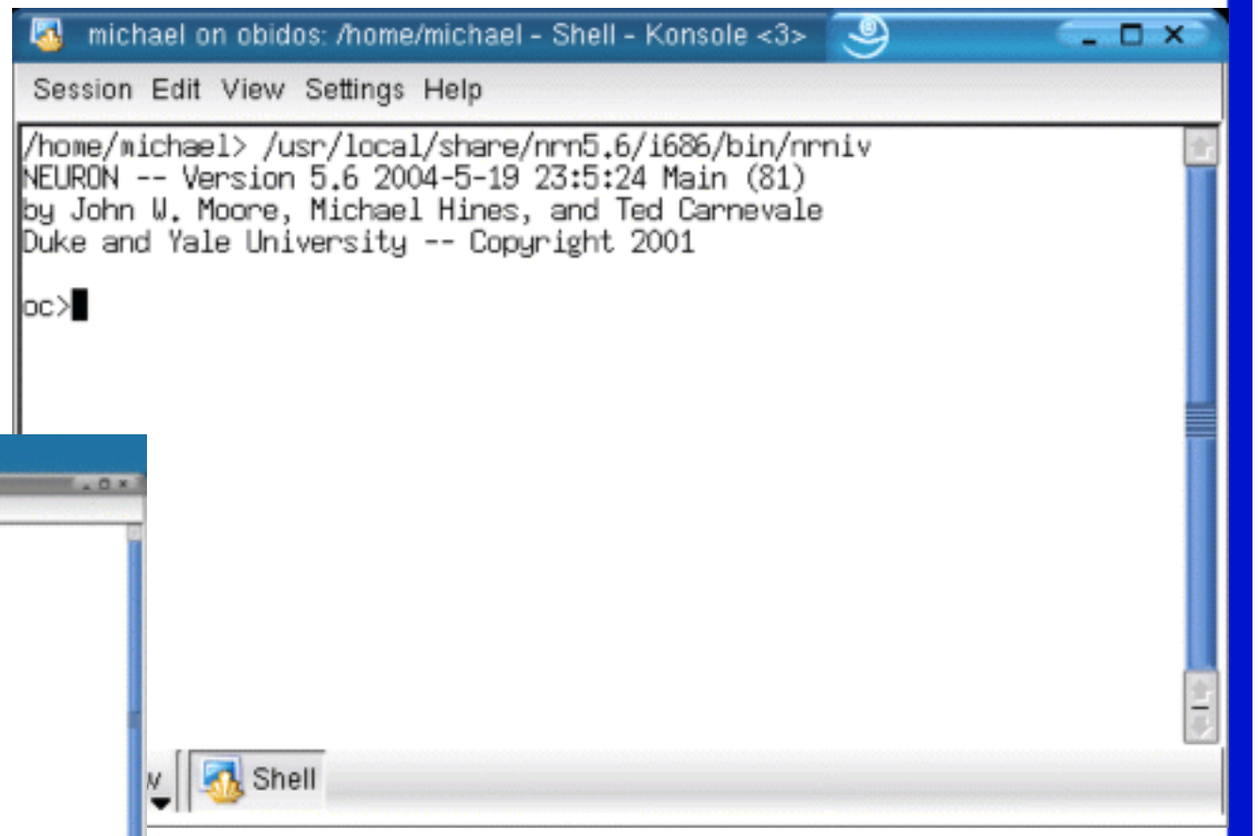
File Edit Bookmarks Tools Settings Help

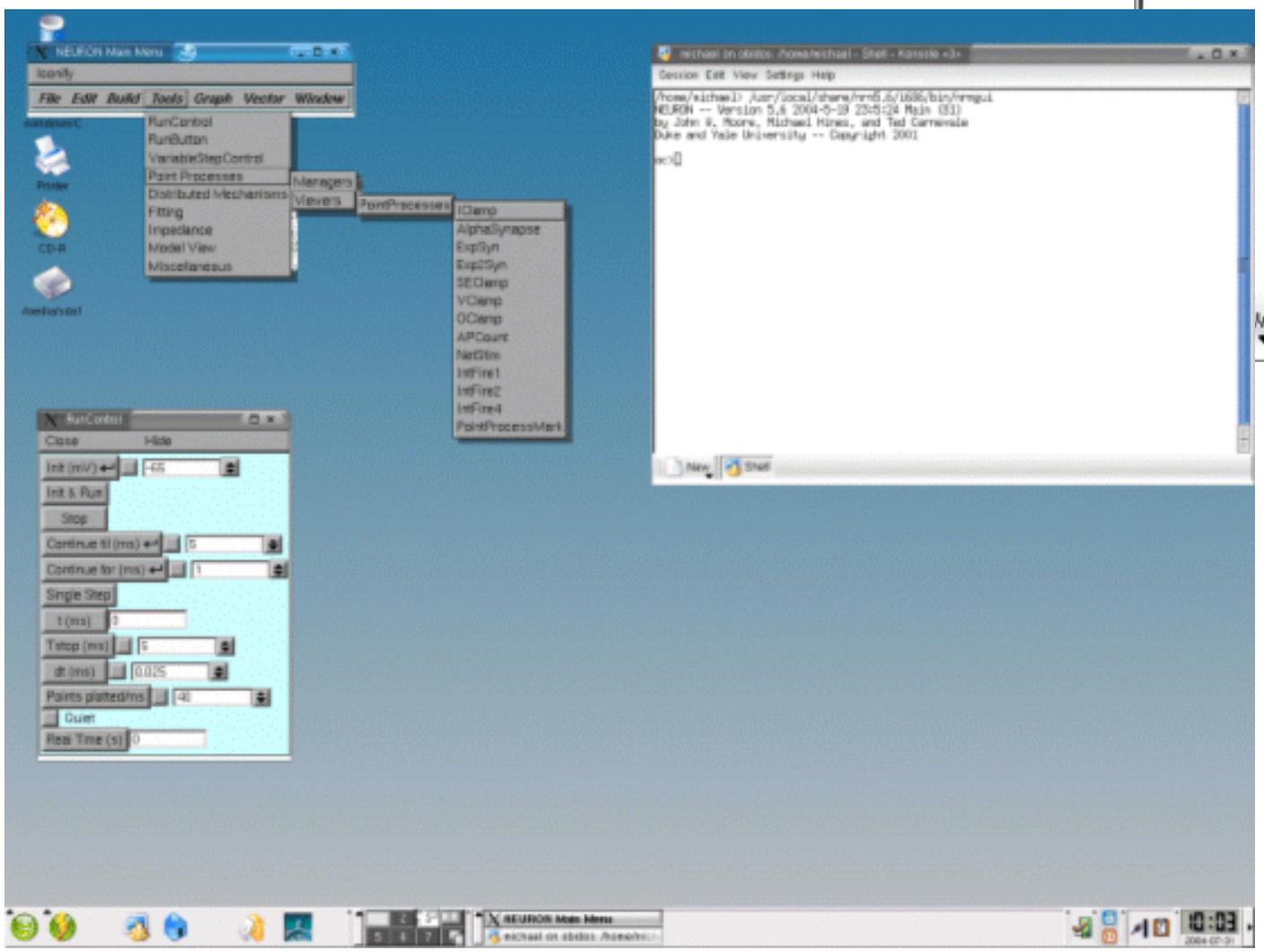
```
//  adding recording
//--------------------------------------------------------------------------------
print "-------------------------------------------------------------[ RECORDING ]---"
if (xopen("addRecording.hoc")) {}

print "================================================================================"

//--------------------------------------------------------------------------------
//  simulation procedures
//--------------------------------------------------------------------------------

//  -[ initialization ]-
proc init() {
  finitialize(v_init)
  fcurrent()
  if (ENABLE_BACKGROUND) openAll_RNGfiles()
  initAll_RNGdata()
  rec_recordVm()
}

//  -[ single step ]-
proc step() { local i
  if (ENABLE_GRAPHICAL_INTERFACE) Plot()
  if (ENABLE_BACKGROUND) updateAll_RNGdata()
  for i=1,nstep_steprun { advance() }
  rec_recordVm()
```

Line: 160 Col: 13 | INS | NORM

NEURON command shell

`$NEURONHOME/i686/bin/nrniv`





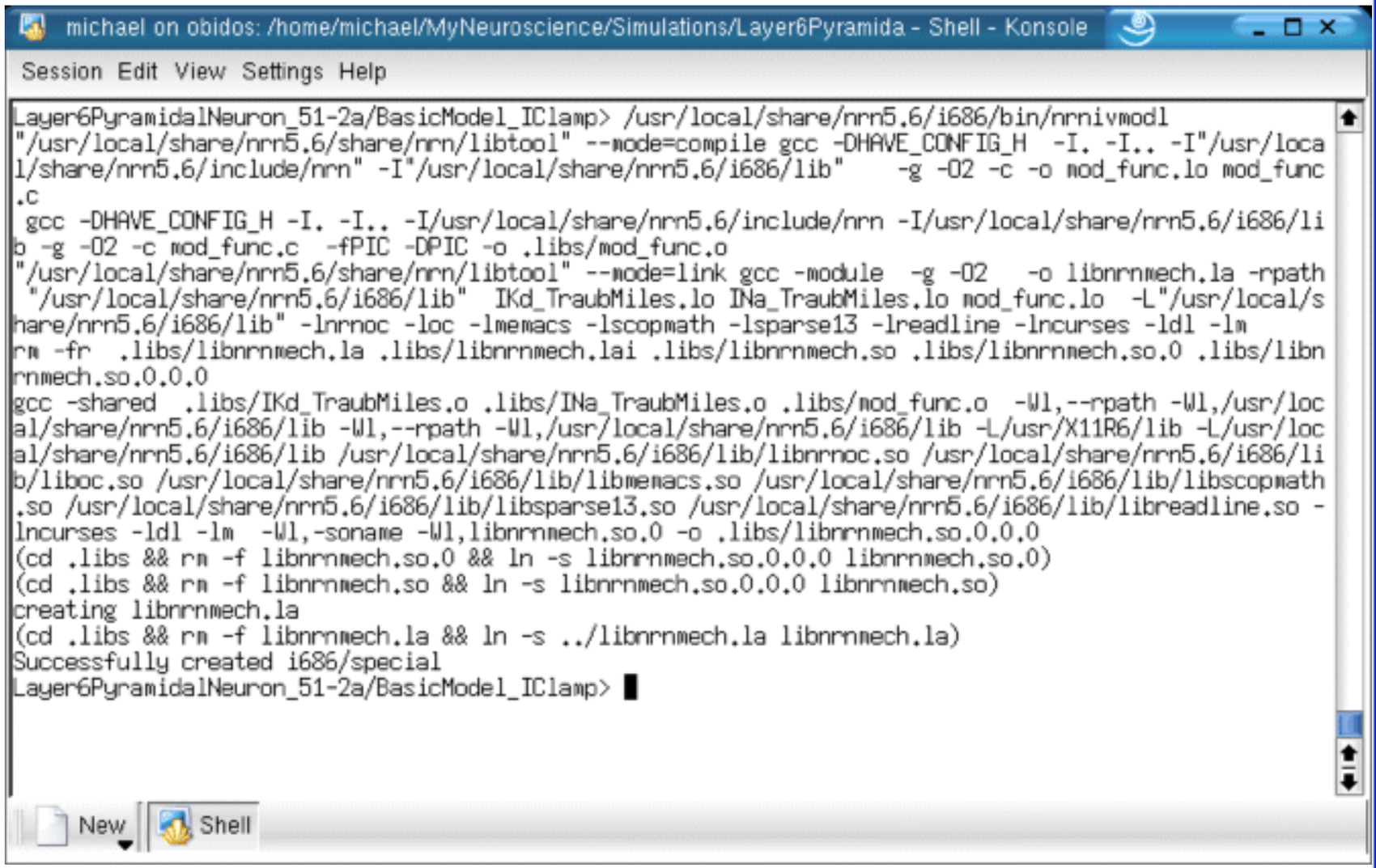
NEURON GUI

`$NEURONHOME/i686/bin/nrngui`

- “customized” versions of NEURON
  - create / edit specific (neuronal) mechanisms (using `modl` programing language) and NEURON script files (using `hoc` programing language)
  - compile `modl` files using

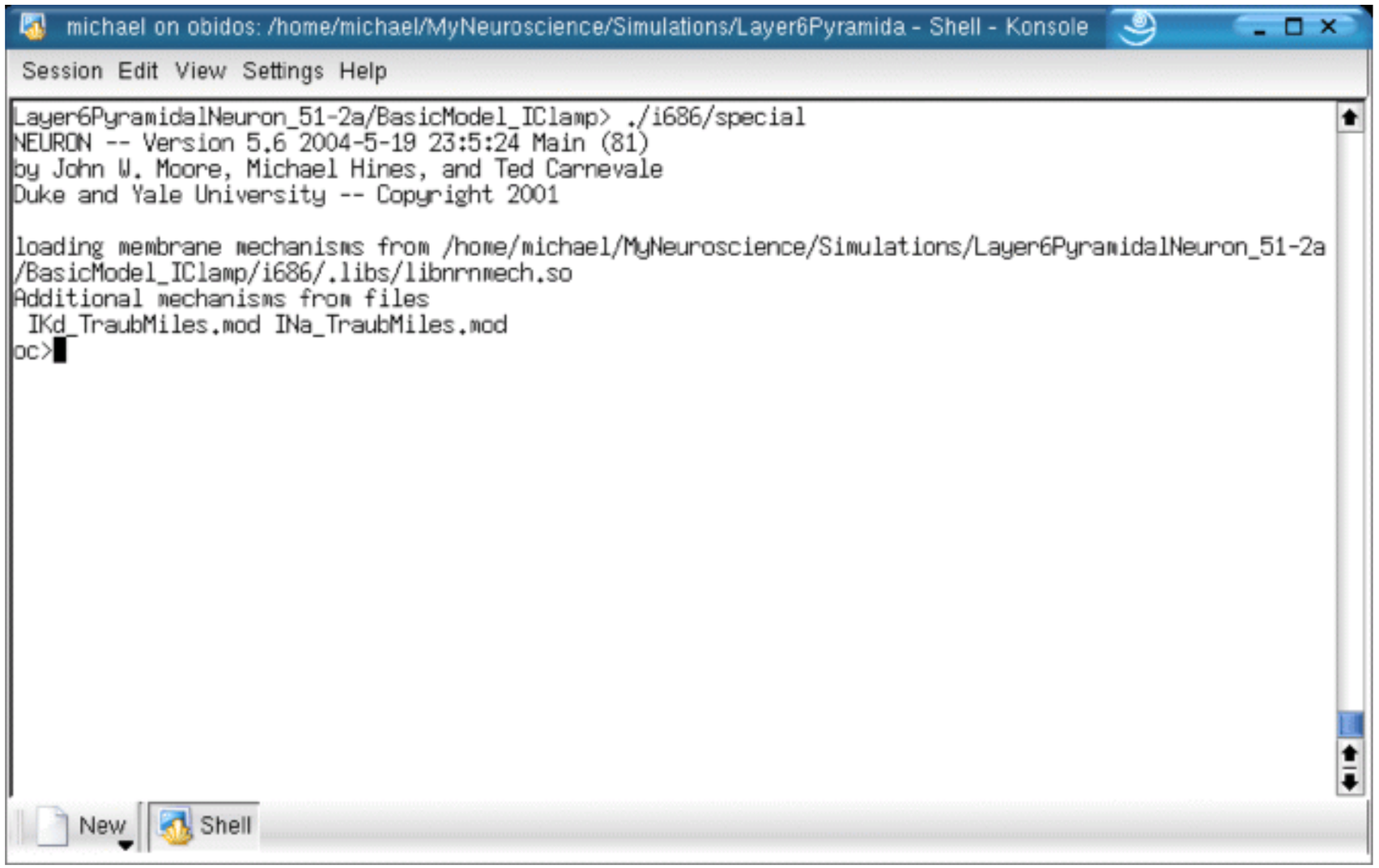
    `$NEURONHOME/i686/bin/nrnivmodl`

- compiles all `modl` files in current directory
- creates directory `./i686`
- creates executable `./i686/special` (“customized” NEURON version)



- run customized version of NEURON

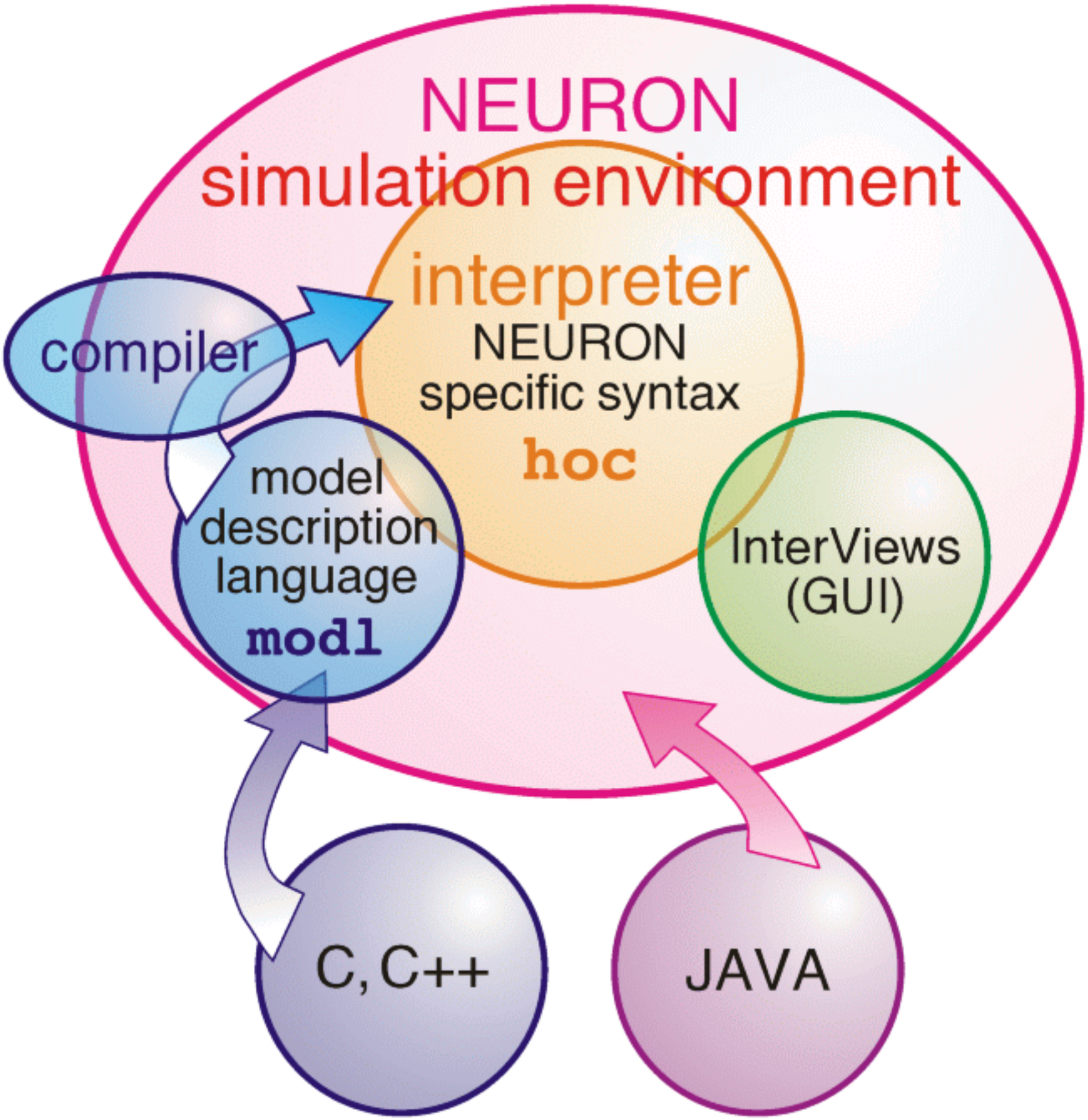
```
./i686/special -
```



Outline

1. The idea behind NEURON
2. Some mathematics behind NEURON
3. Running NEURON
4. Two programming levels = infinite possibilities
5. A step towards infinity: some examples

4 Two programming levels = infinite possibilities



❹ Two programming levels = infinite possibilities

NEURON Interpreter:

- "calculator" optimized for integrating sets of coupled differential equations
- object oriented syntax

① first programming level: `hoc`

- definition of neuronal models (morphology, membrane properties, stimulation and recording protocols)
- basic I/O operations
- control of simulation
- data analysis

② second programming level: `nmodl`

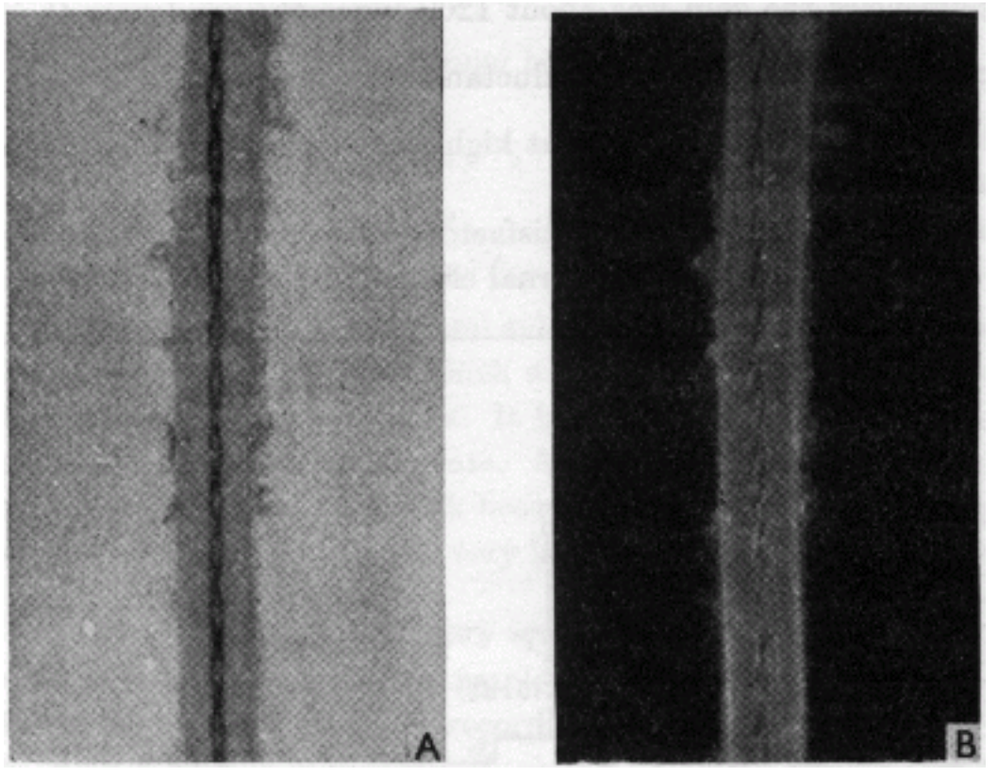
- description of biochmical / biophysical processes in terms of kinetic schemes or sets of differential and algebraic equations
- C-like syntax; allows incorporation of routines written in C

Outline

1. The idea behind NEURON
2. Some mathematics behind NEURON
3. Running NEURON
4. Two programming levels = infinite possibilities
5. A step towards infinity: some examples

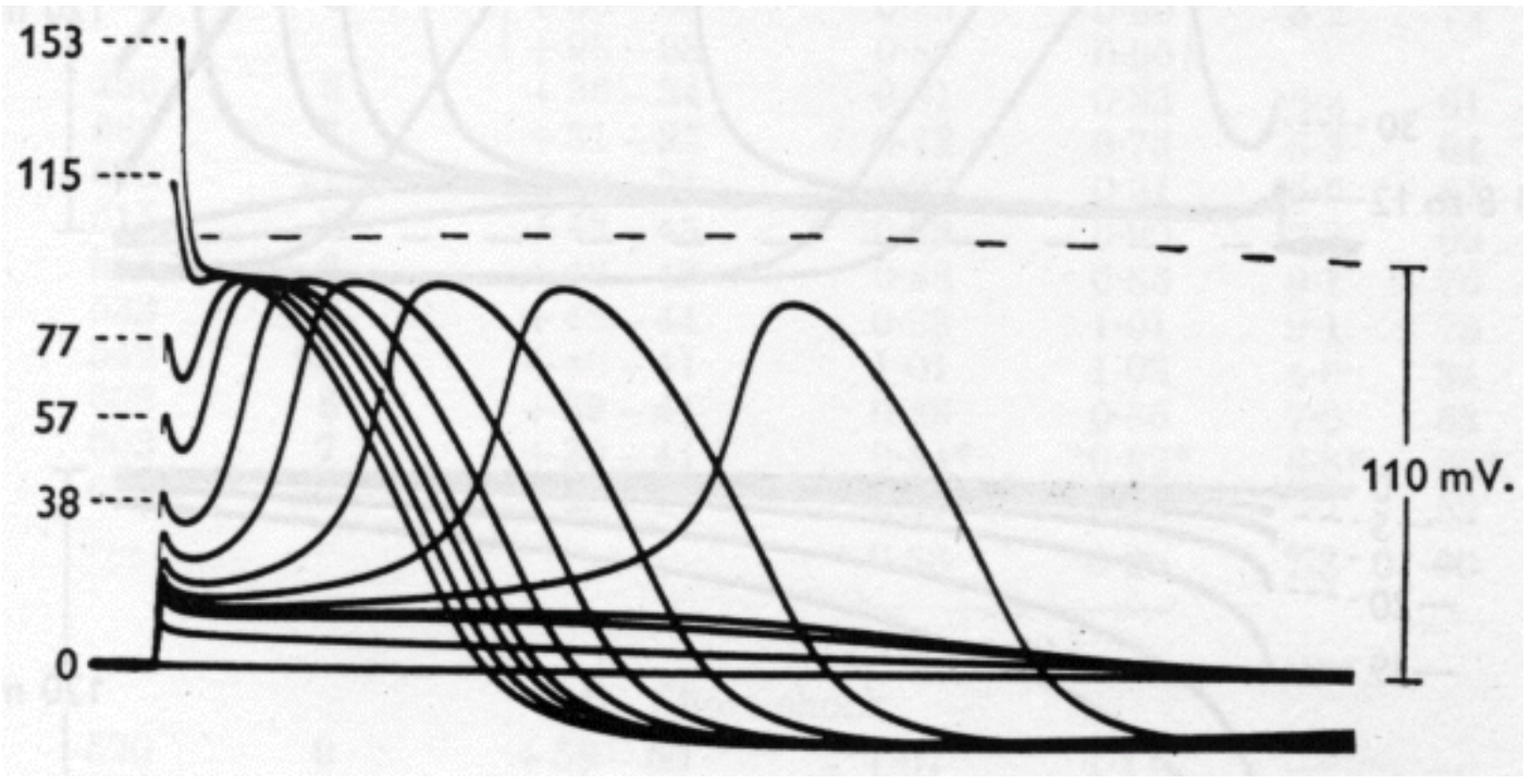
❶ Hodgkin and Huxley's squid axon

biophysical system



loligo pealei
[www.mbl.edu]

the giant axon
[Hodgkin *et al.*, *J. Physiol.* **116** (1952), 424-448]



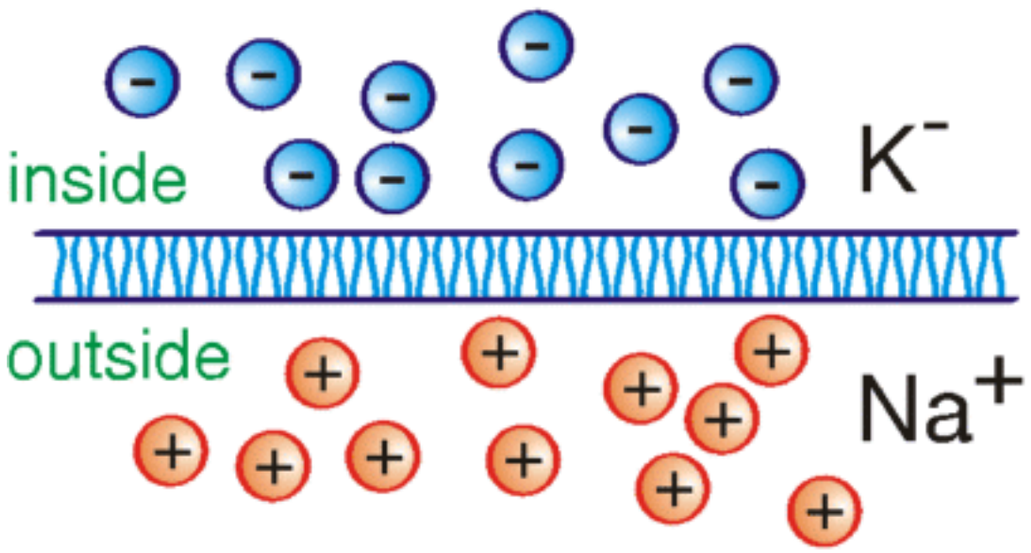
membrane potential time course following a brief current injection
[Hodgkin *et al.*, *J. Physiol.* **116** (1952), 424-448]

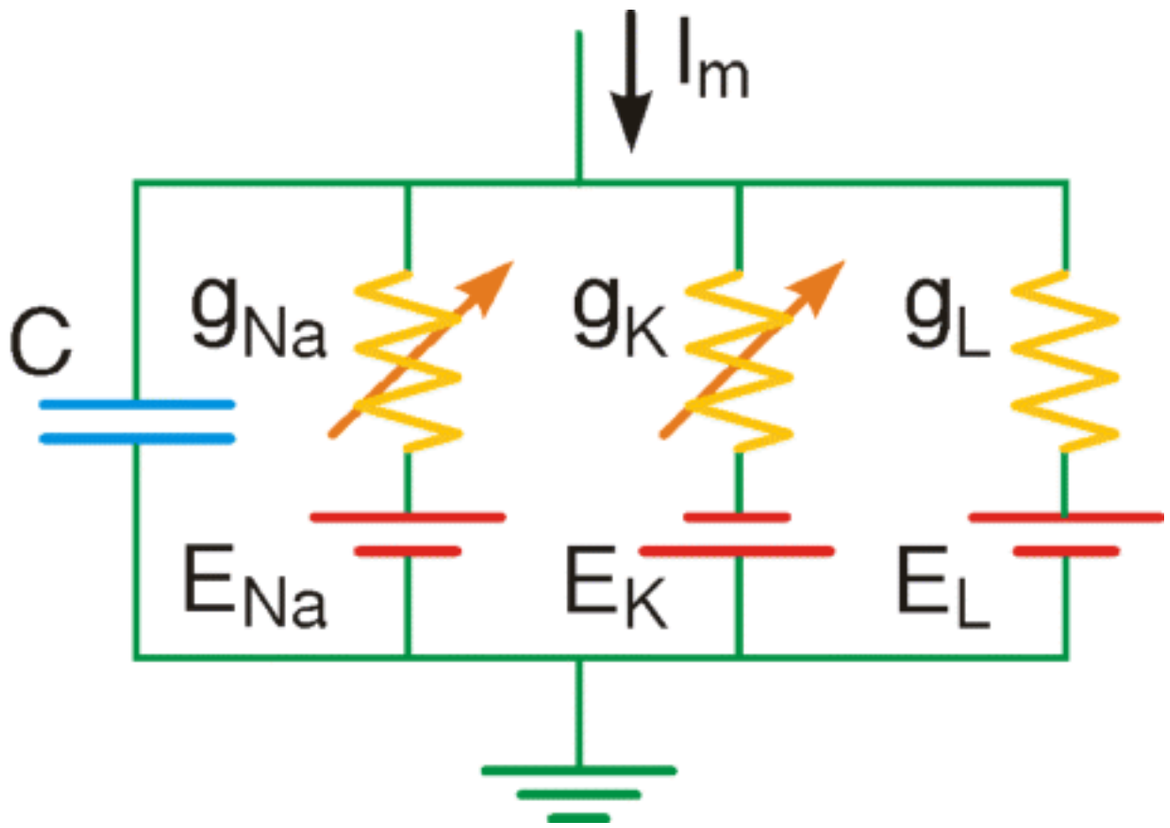
$$C\frac{dV(t)}{dt} = -I_m(t)$$
$$I_m(t) = g_{Na}(t)\left(V(t)-E_{Na}\right) + g_K(t)\left(V(t)-E_K\right) + g_L\left(V(t)-E_L\right)$$
$$g_{Na}(t) = \bar{g}_{Na}\, m^3(t)\, h(t)$$
$$g_K(t) = \bar{g}_K\, n^4(t)$$

conceptual model

Hodgkin-Huxley model
[Hodgkin, Huxley, Katz (1952)]

x	E_x [mV]	g_x [mS cm^2]
Na	115	120
K	-12	36
L	10.6	0.3



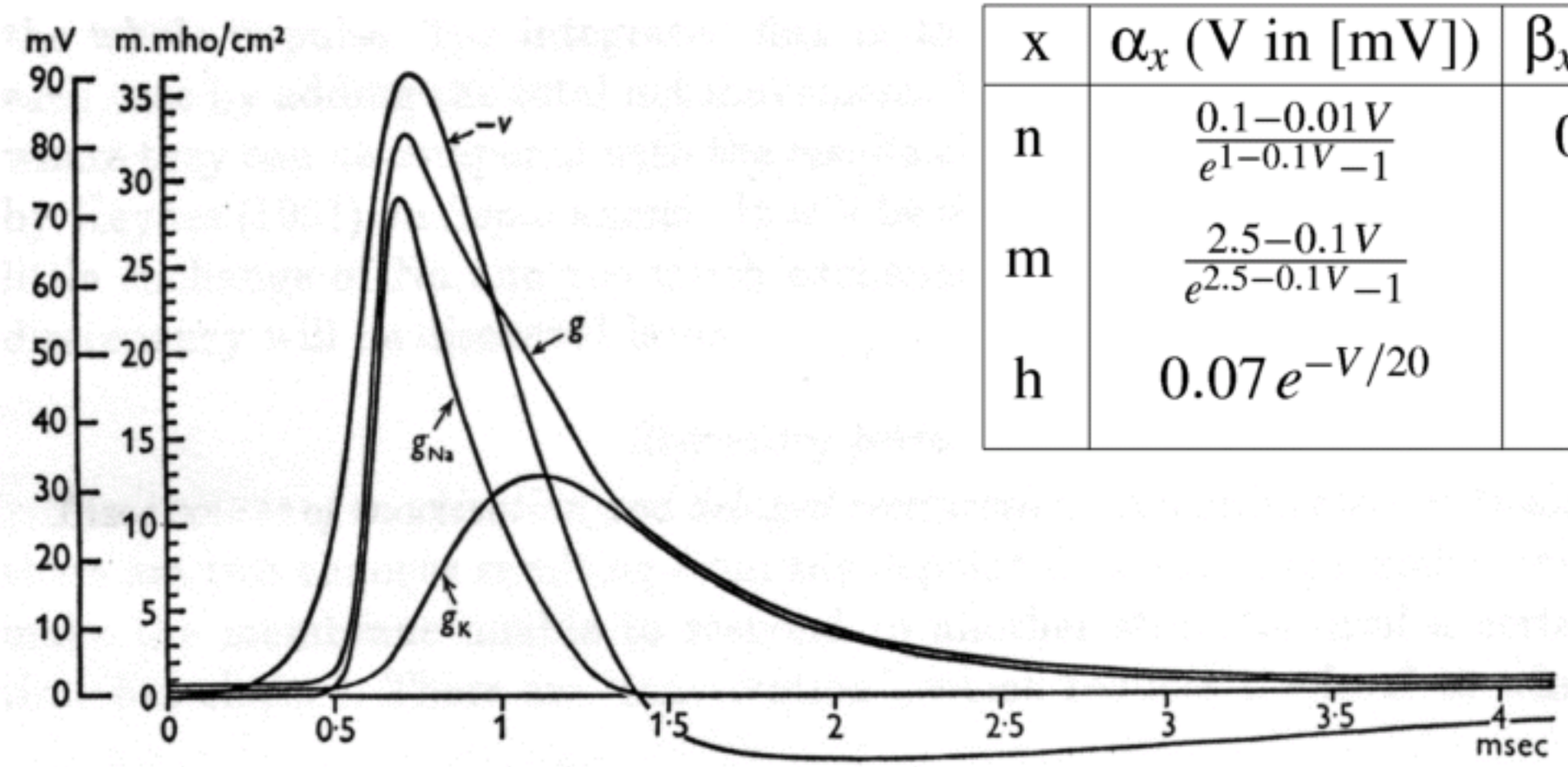


$$\frac{dm(t)}{dt} = \alpha_m(V)\left(1-m(t)\right) - \beta_m(V)\,m(t)$$
$$\frac{dh(t)}{dt} = \alpha_h(V)\left(1-h(t)\right) - \beta_h(V)\,h(t)$$
$$\frac{dn(t)}{dt} = \alpha_n(V)\left(1-n(t)\right) - \beta_n(V)\,n(t)$$

conceptual model

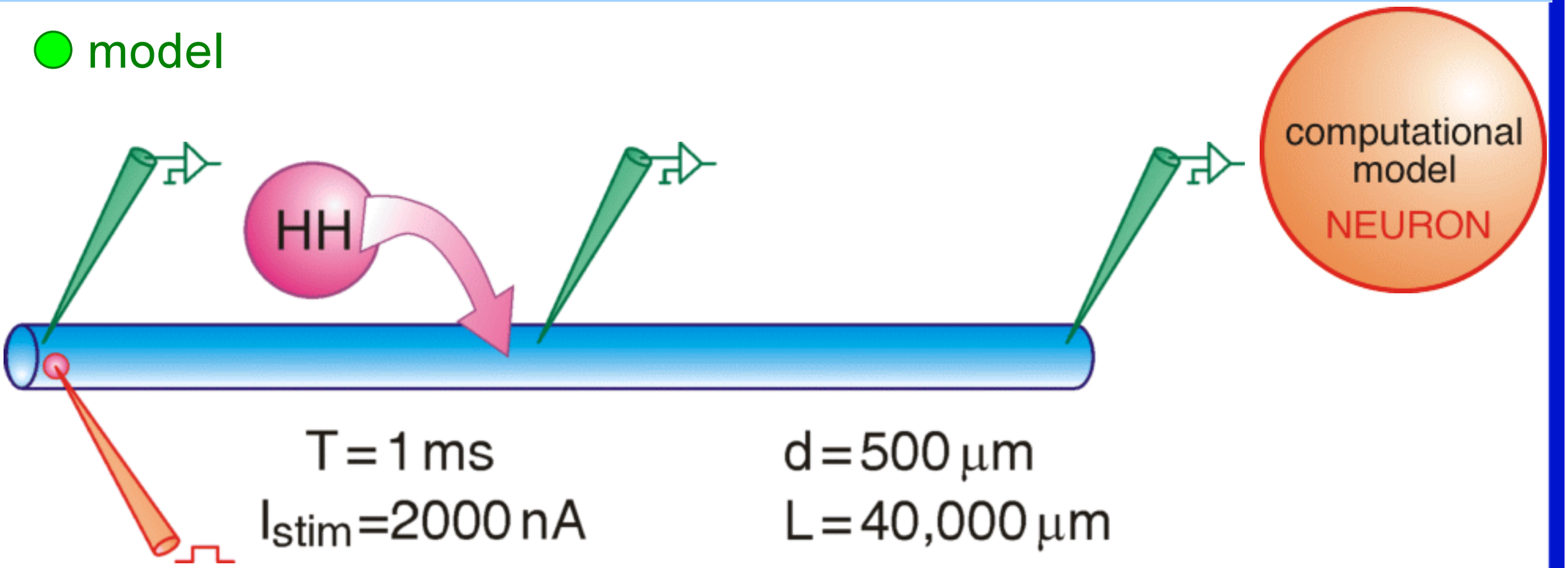
Hodgkin-Huxley model
[Hodgkin, Huxley, Katz (1952)]

x	α_x (V in [mV])	β_x (V in [mV])
n	$\frac{0.1-0.01V}{e^{1-0.1V}-1}$	$0.125\,e^{-V/80}$
m	$\frac{2.5-0.1V}{e^{2.5-0.1V}-1}$	$4\,e^{-V/18}$
h	$0.07\,e^{-V/20}$	$\frac{1}{e^{3-0.1V}+1}$



[Hodgkin & Huxley, *J. Physiol.* **117** (1952), 500-544]

model



morphology

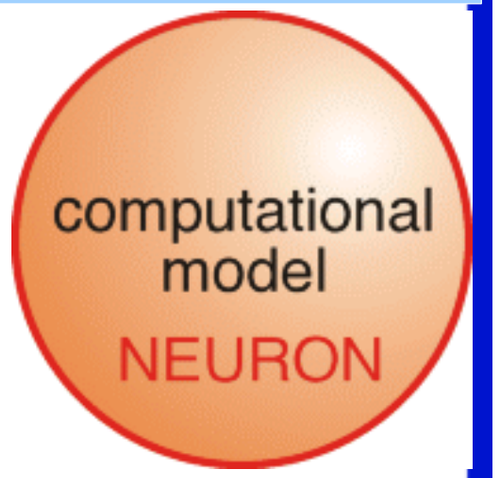
```
create axon
access axon
```

create and access named section (or compartment)

```
axon nseg = 100
axon diam = 500
axon L = 40000
```

define morphological properties and spatial discretization

5 A step towards infinity: some examples



passive properties

```
axon insert pas
axon Ra = 123.0
```

insert and set passive membrane properties

active properties

```
axon insert hh
```

insert and set active membrane properties

stimulation

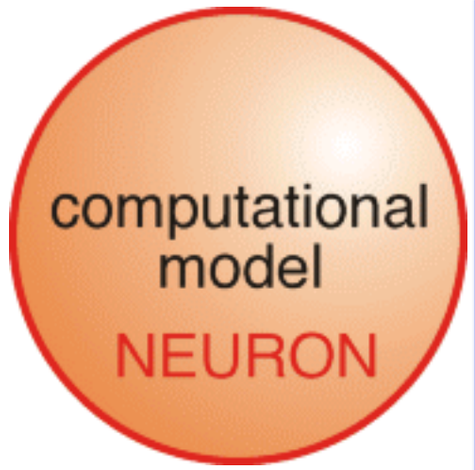
```
objectvar stim
stim = new IClamp(0.5)
```

insert point process (current clamp electrode) at specific site

```
stim.del = 1
stim.dur = 1
stim.amp = 2000
```

define properties of stimulation

merge everything together ...




```
load_file("nrngui.hoc")
```
load graphical user interface

```
create axon
access axon

axon nseg = 100
axon diam = 500
axon L = 40000
```
morphology

```
axon insert pas
axon Ra = 123.0
```
passive properties

```
axon insert hh
```
active properties

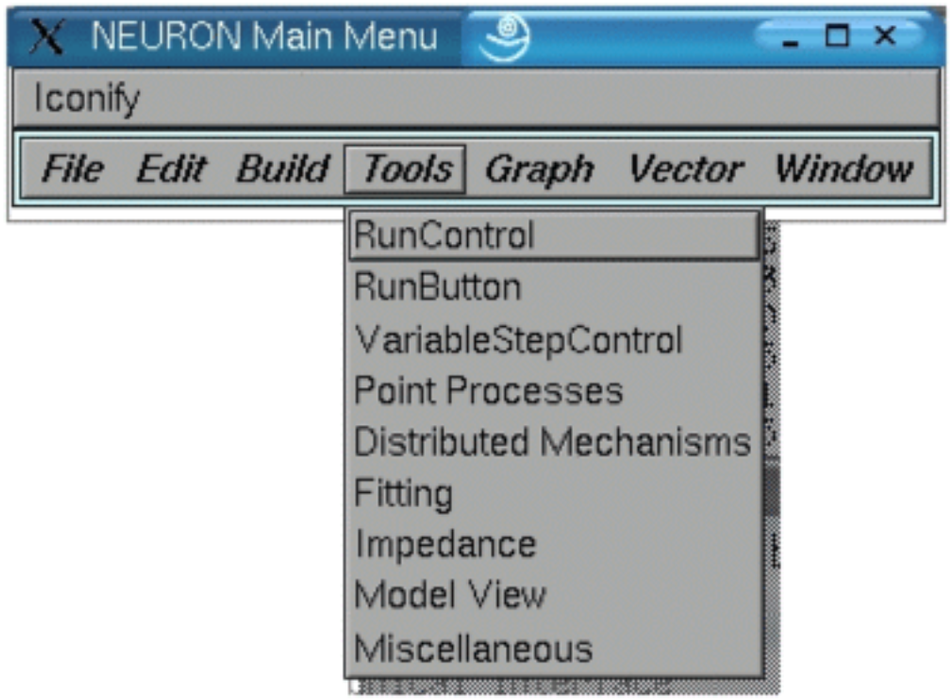
```
objectvar stim
stim = new Iclamp(0.5)

stim.del = 1
stim.dur = 1
stim.amp = 2000
```
stimulation

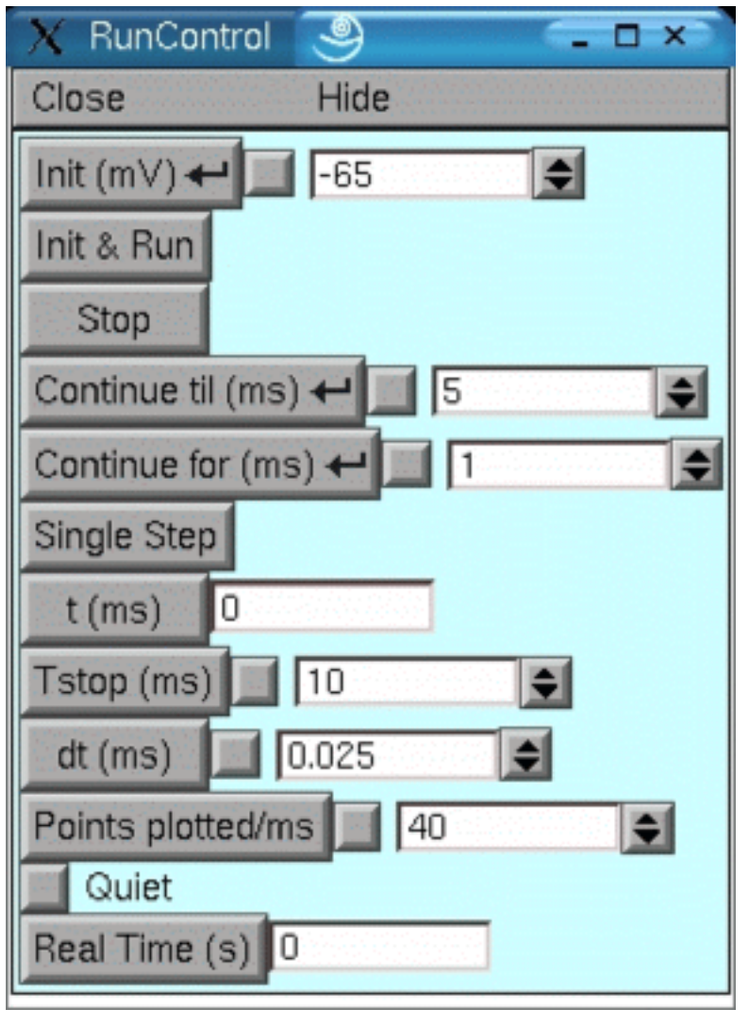
```
tstart = 0
tstop = 10
```
global simulation settings

- ... run it ...

```
oc> load_file("HHexample.hoc")
```

- ... change some GUI settings ...
  - simulation control menu





computational model
NEURON

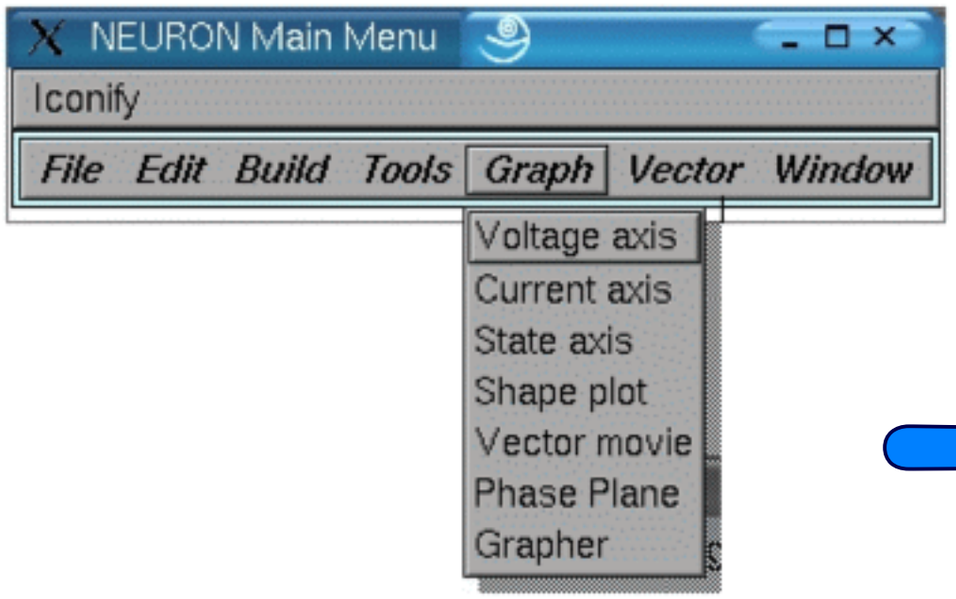
5 A step towards infinity: some examples

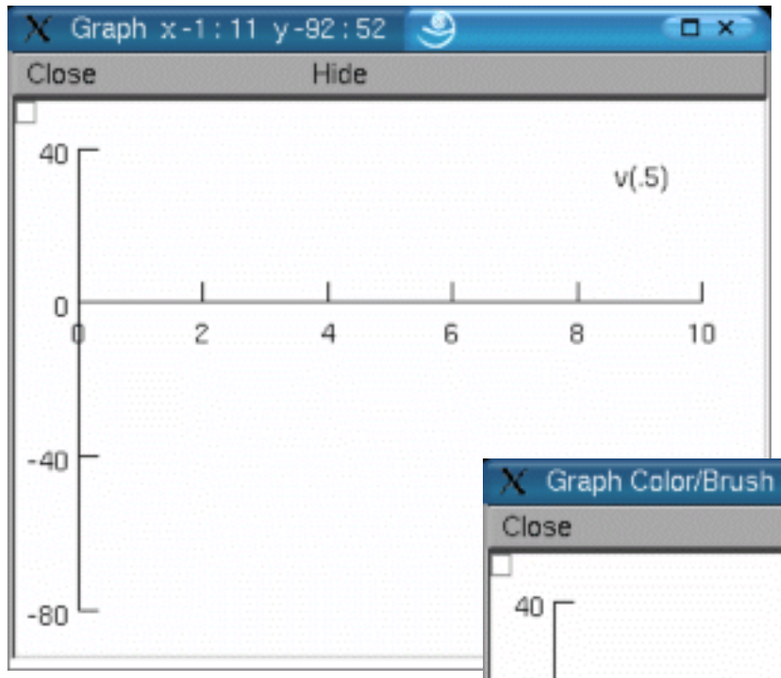
1 Hodgkin & Huxley

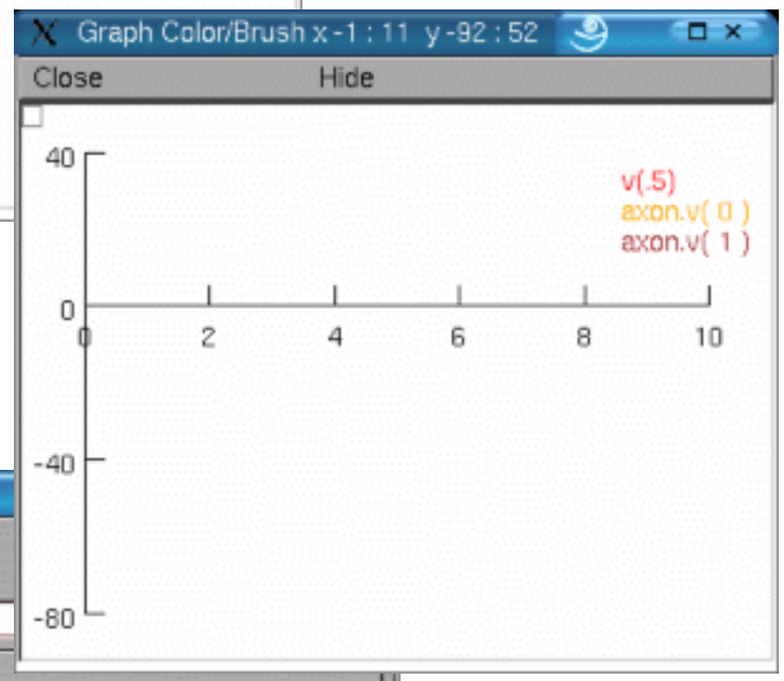
- membrane potential time course

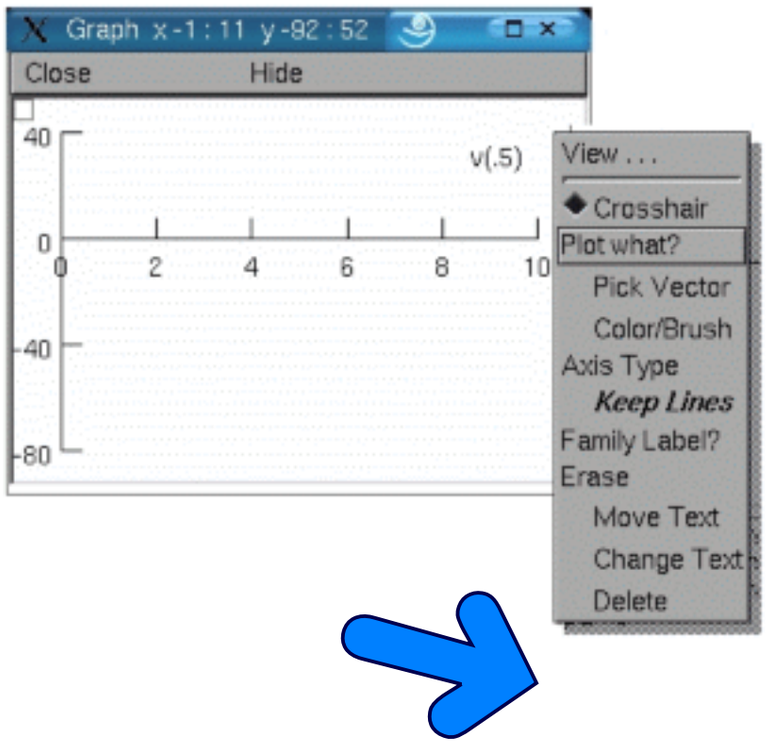
computational model

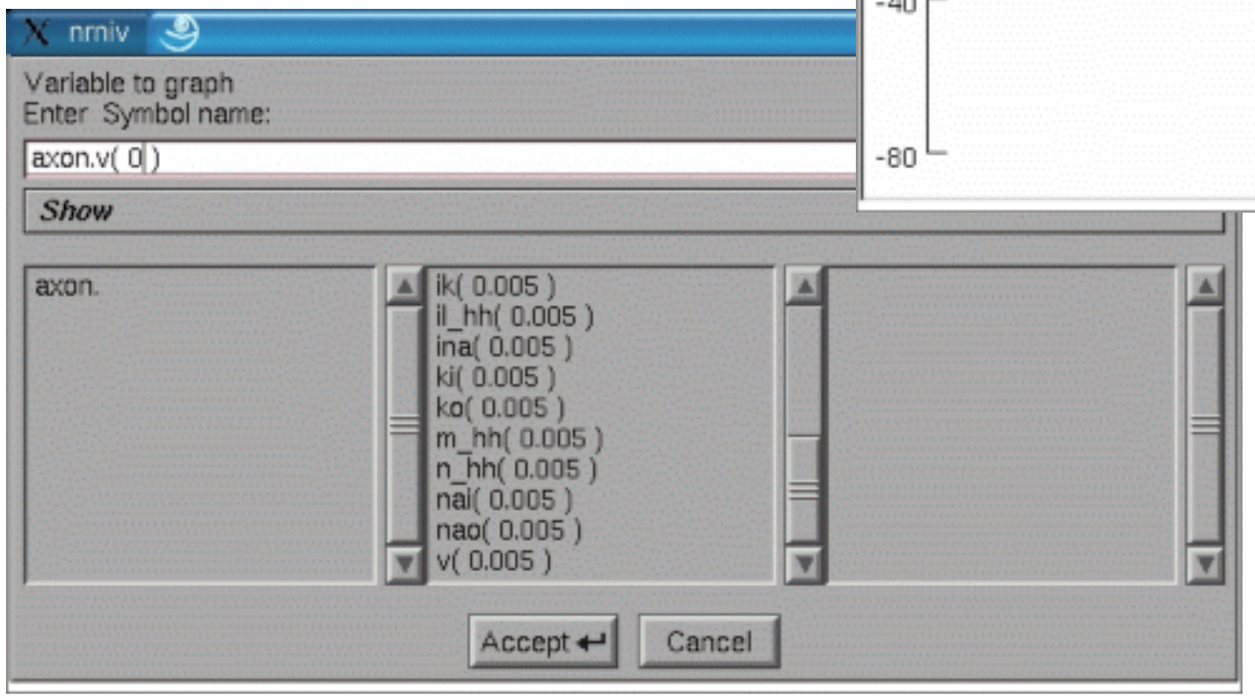
NEURON









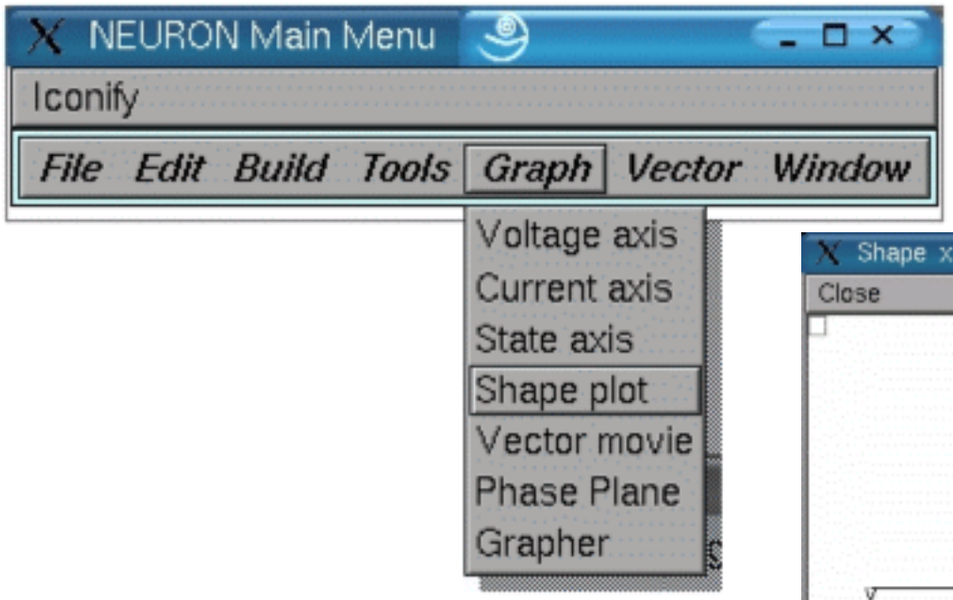


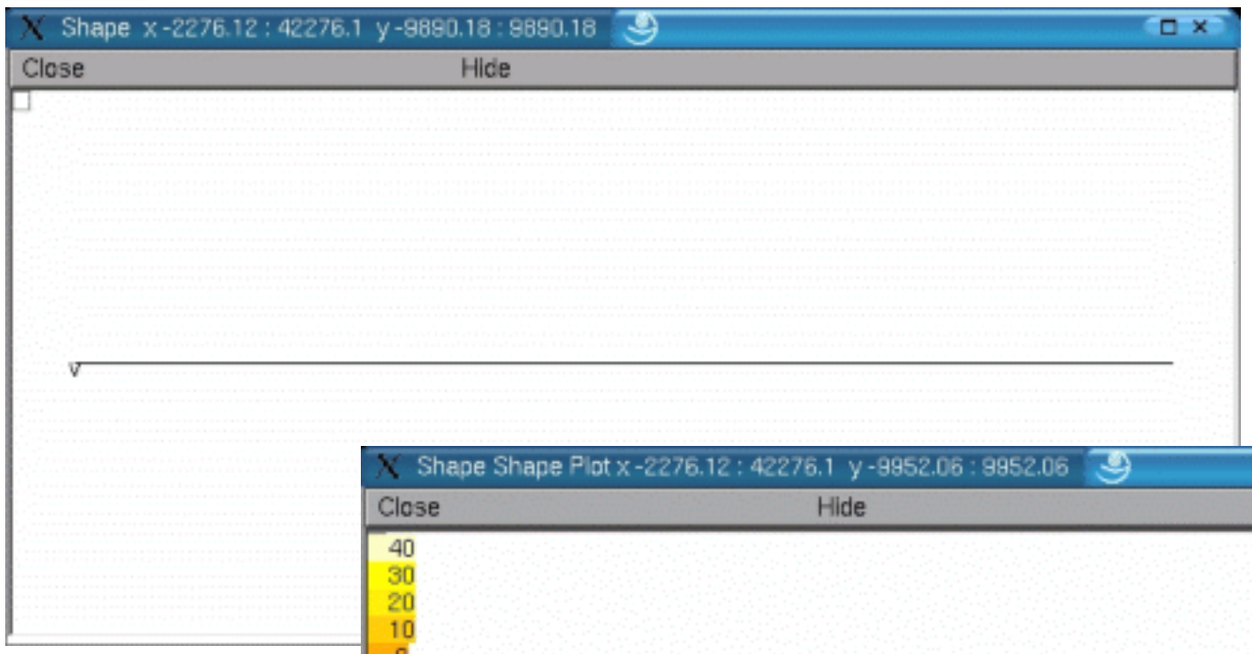
5 A step towards infinity: some examples

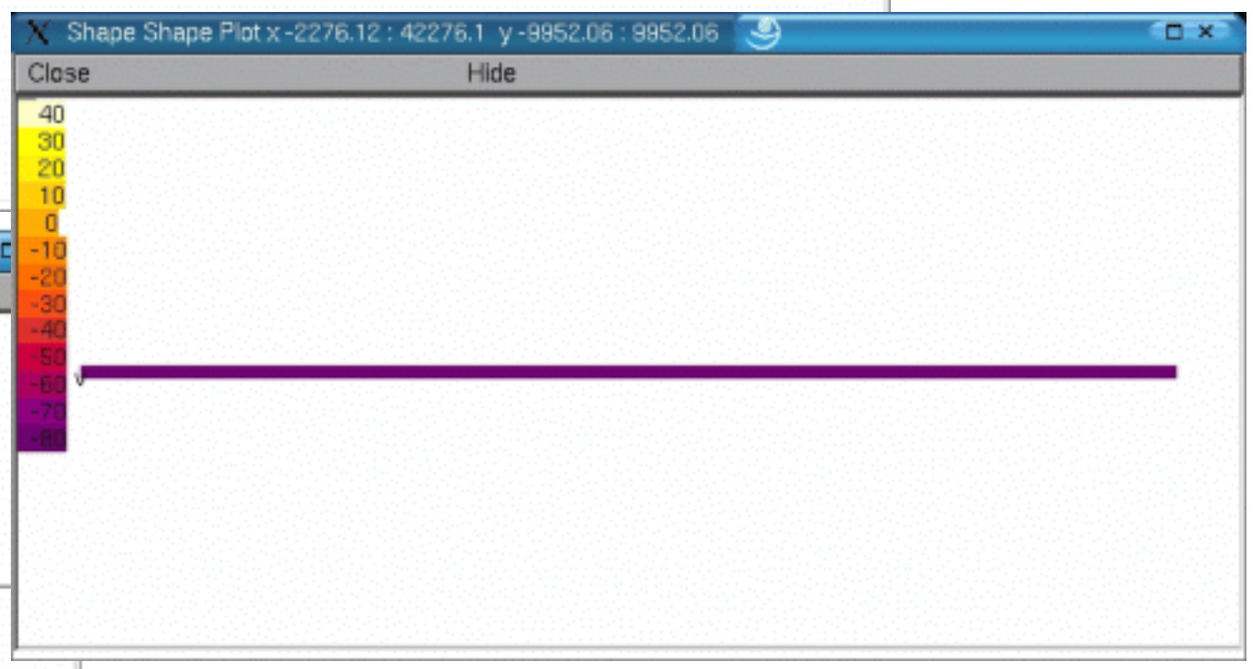
- "shape plot" to display morphology

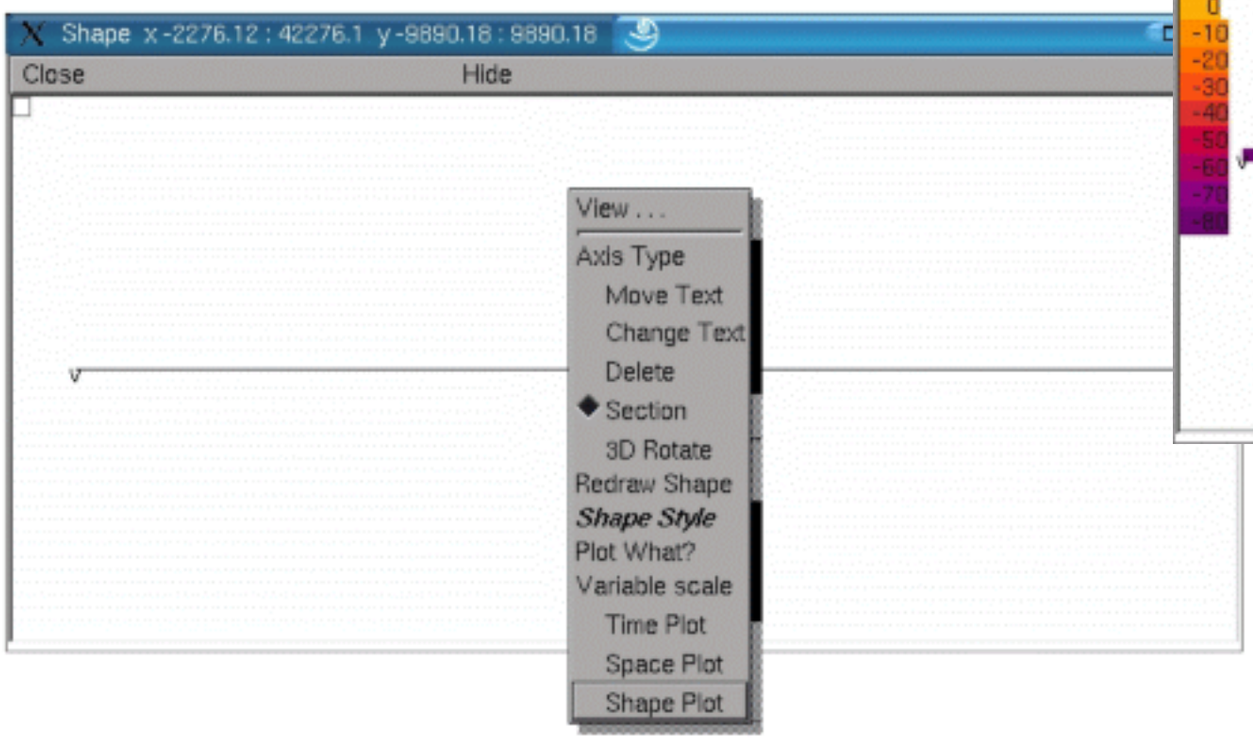
computational model

NEURON







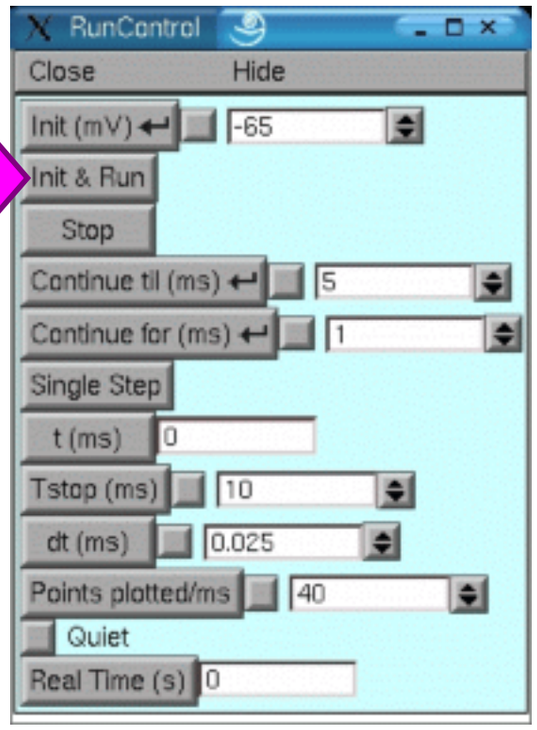


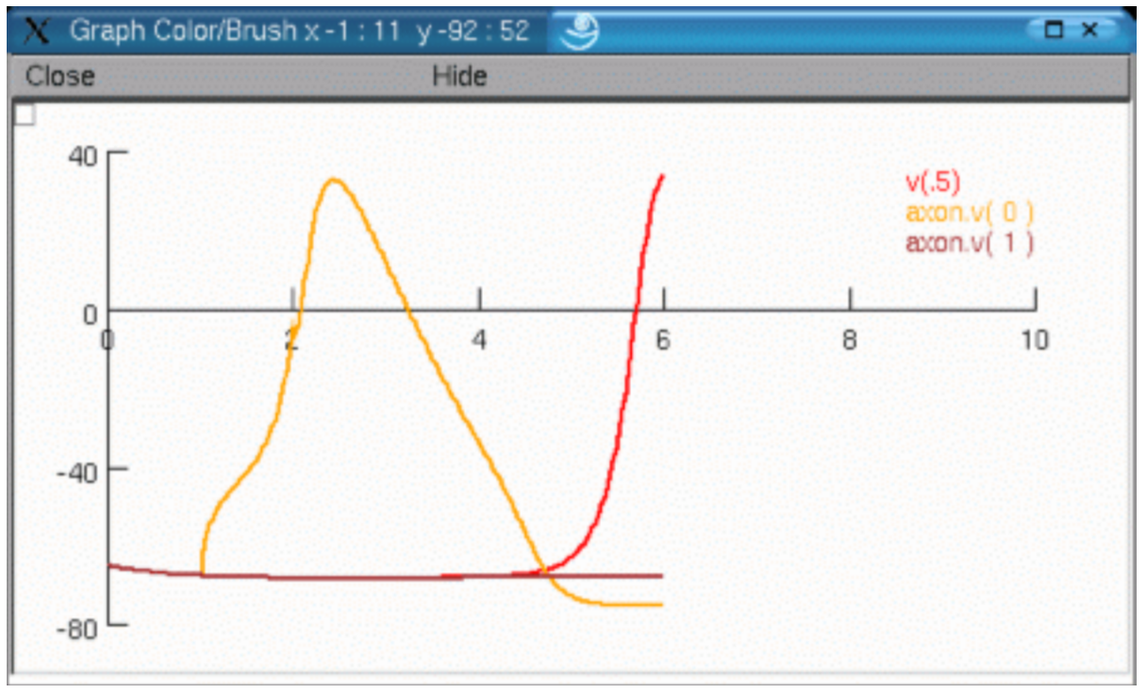
5 A step towards infinity: some examples

1 Hodgkin & Huxley

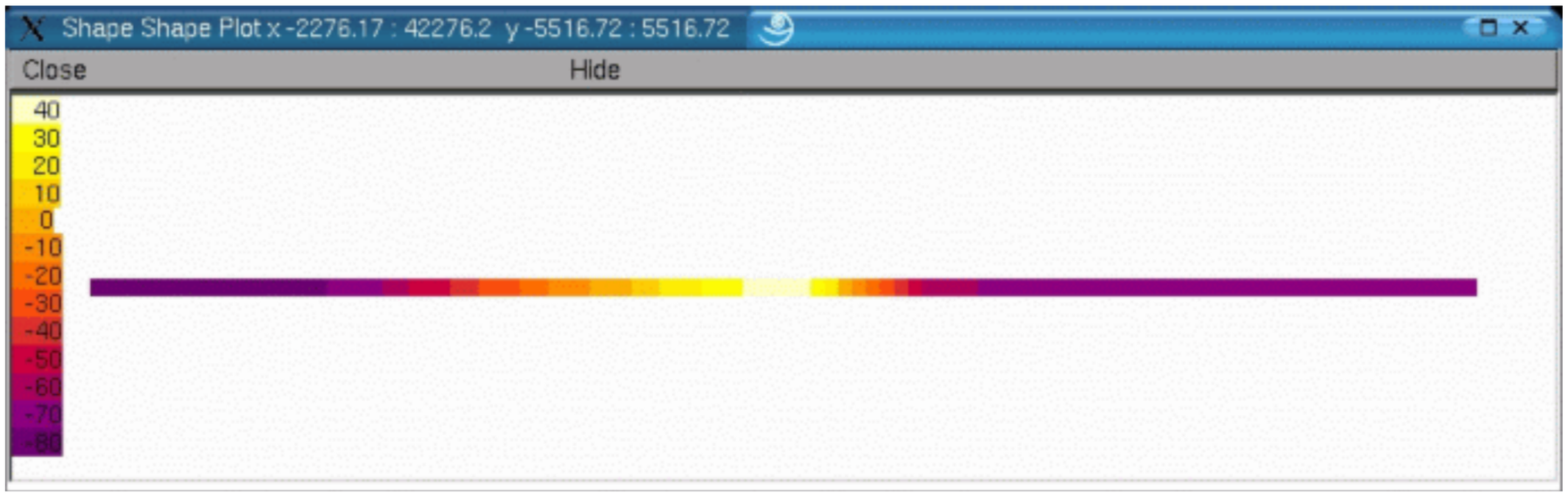
computational model NEURON

- ... start and watch the simulation running ...





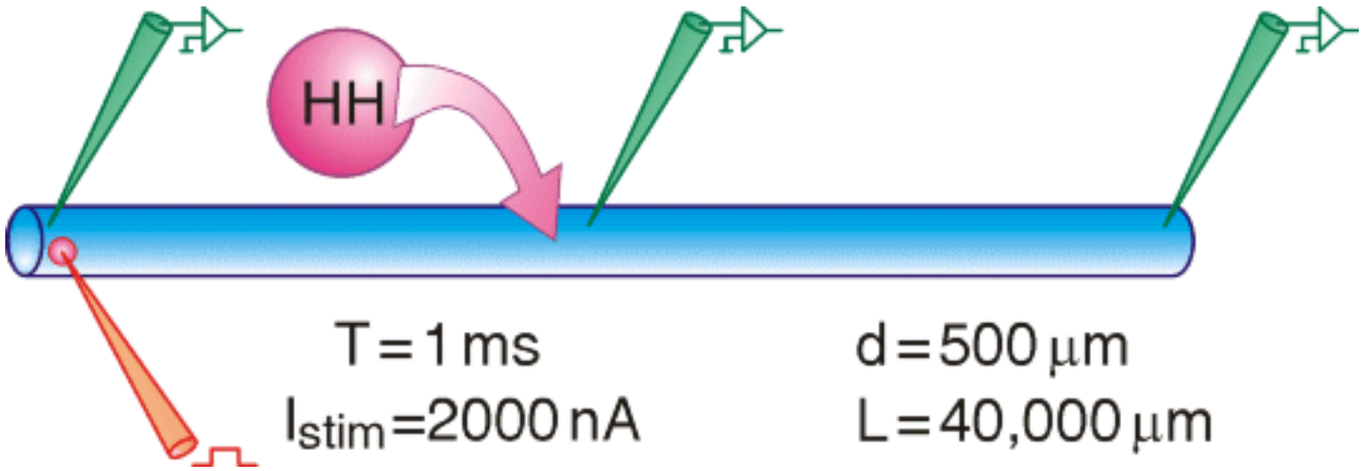
membrane potential as function of time at various sites along the axon



membrane potential along the axon

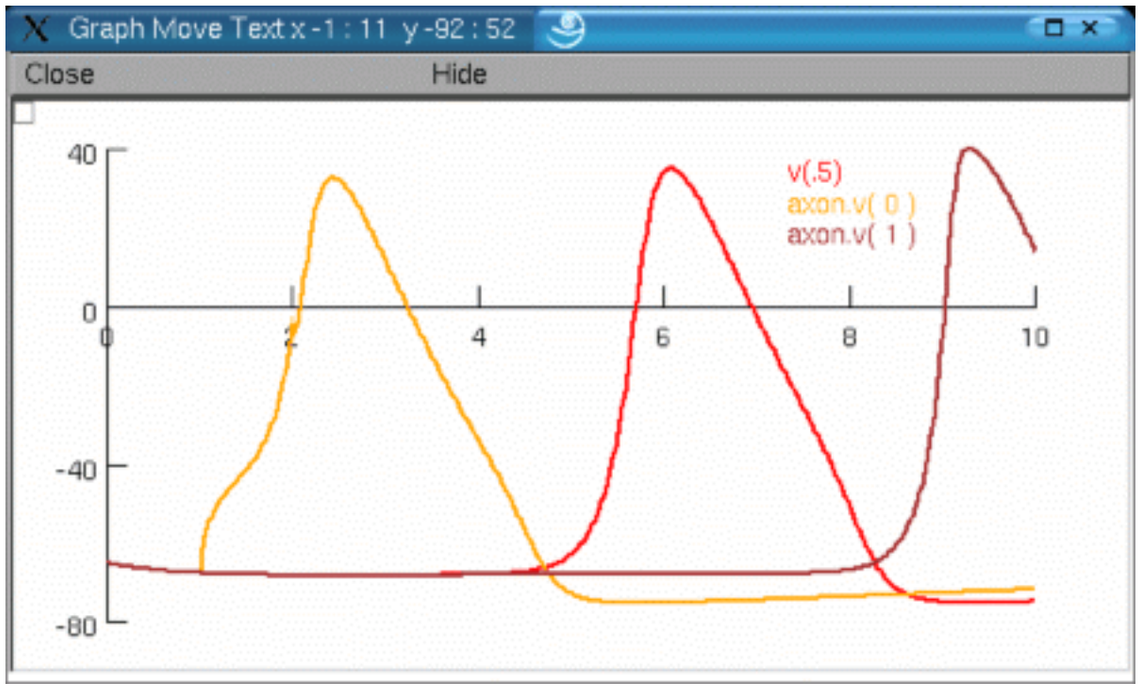
- ... and, finally, enjoy the result

computational model
NEURON

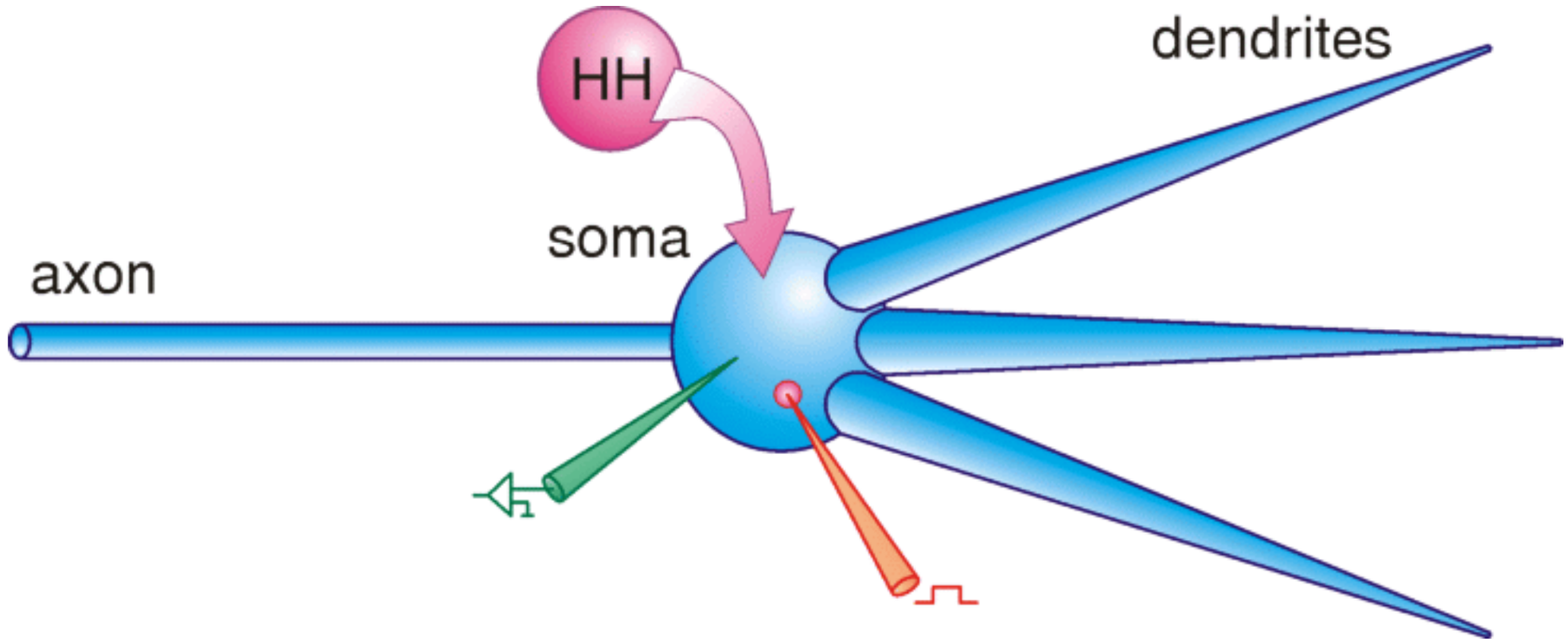


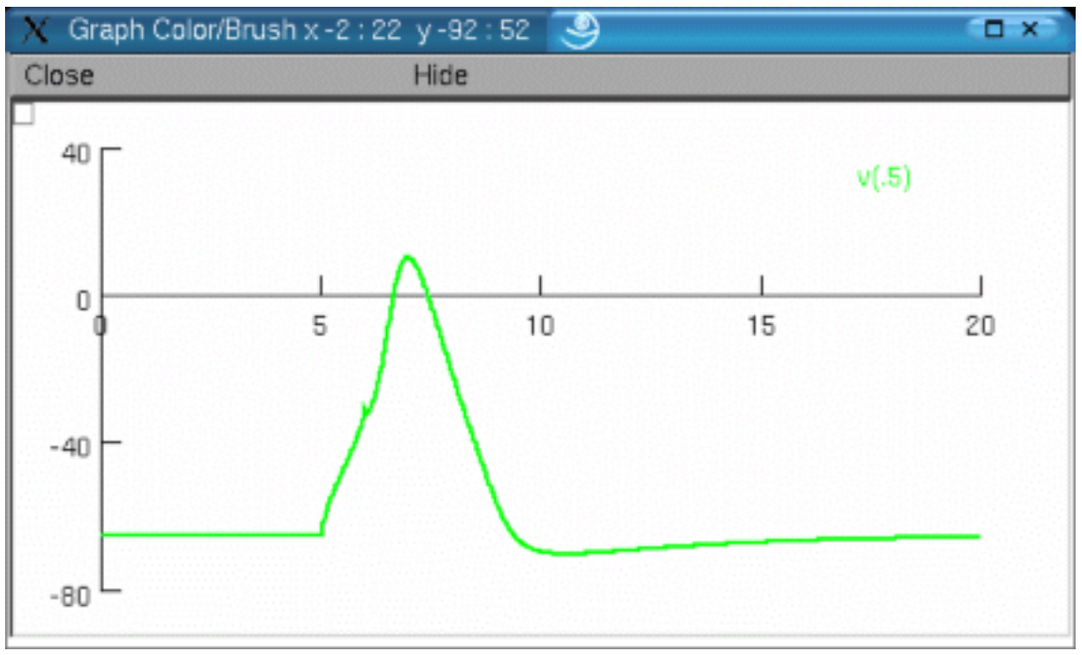
propagating axonal action potentials

membrane potential as function of time

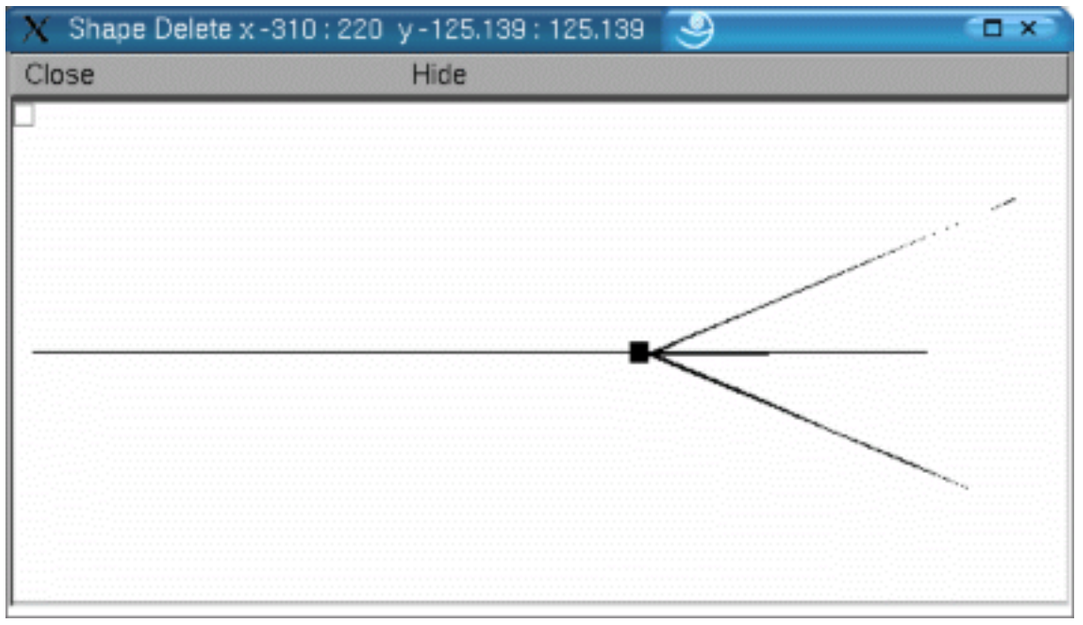


❷ 5-compartment model with Hodgkin-Huxley channels in soma



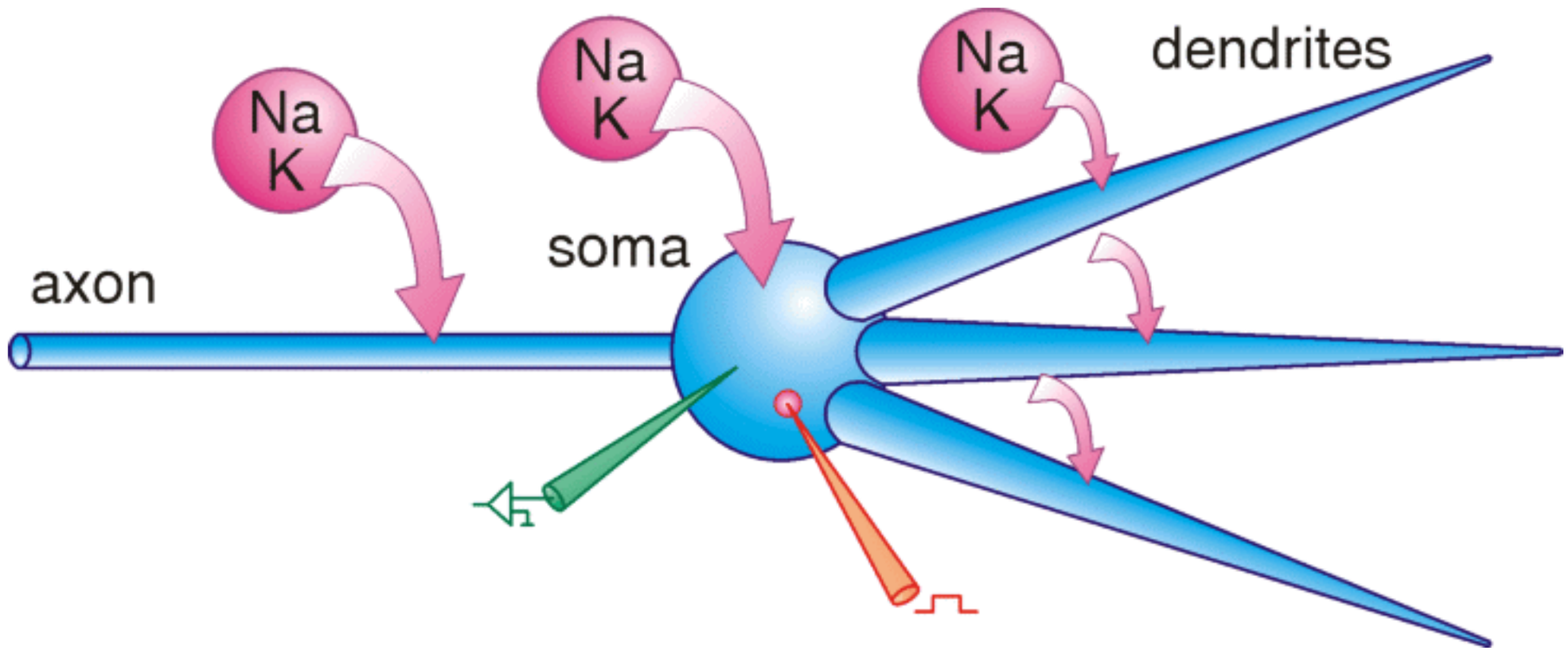


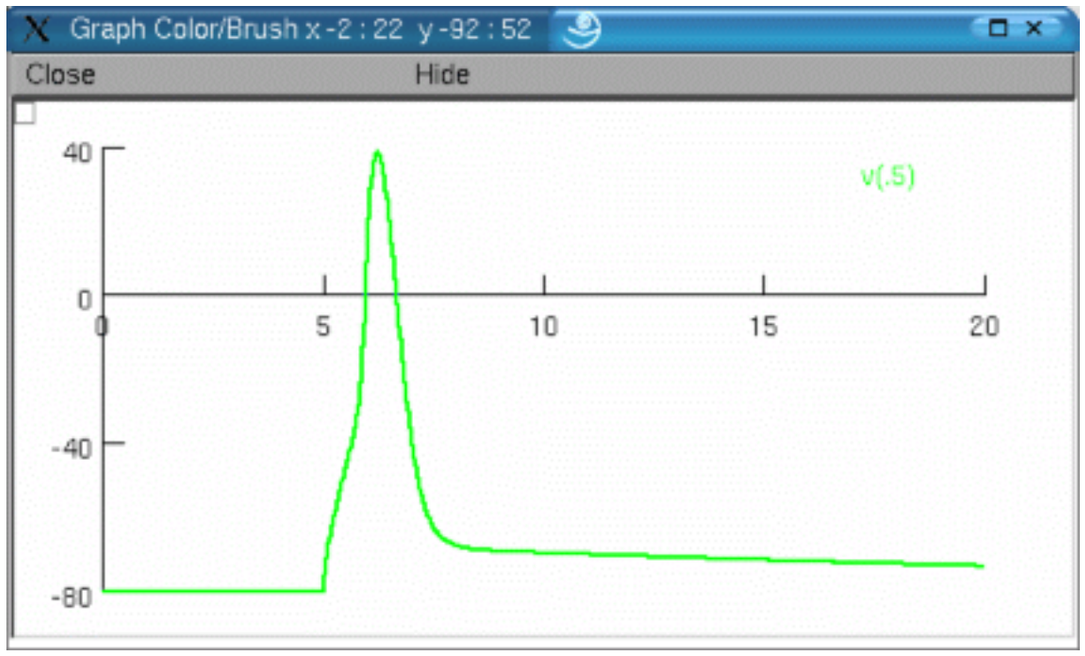
membrane potential



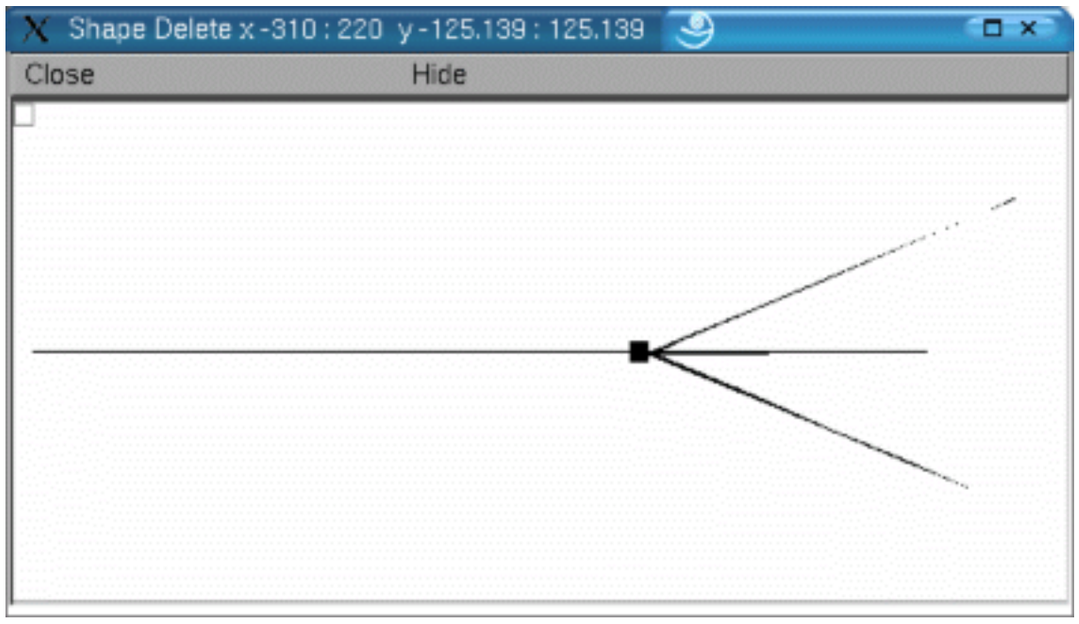
shape plot

❸ 5-compartment model with custom ion channels



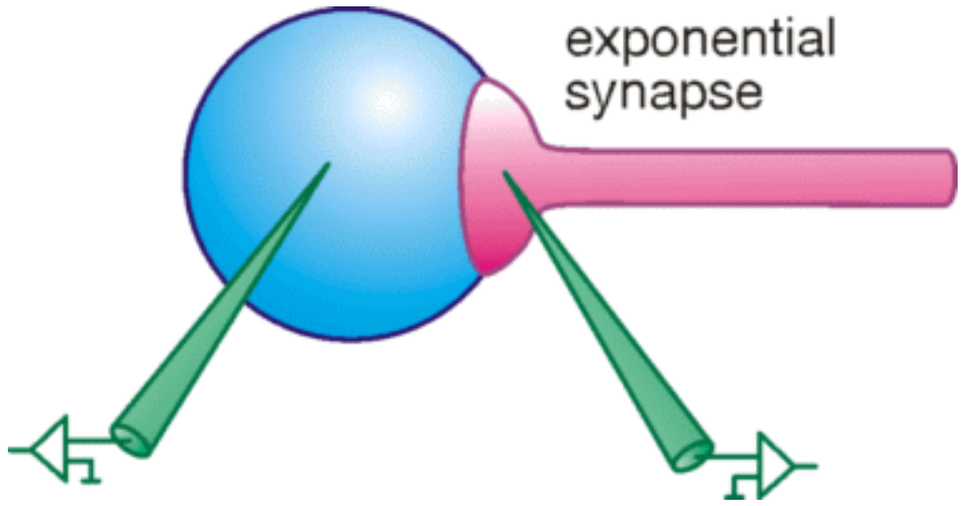


membrane potential

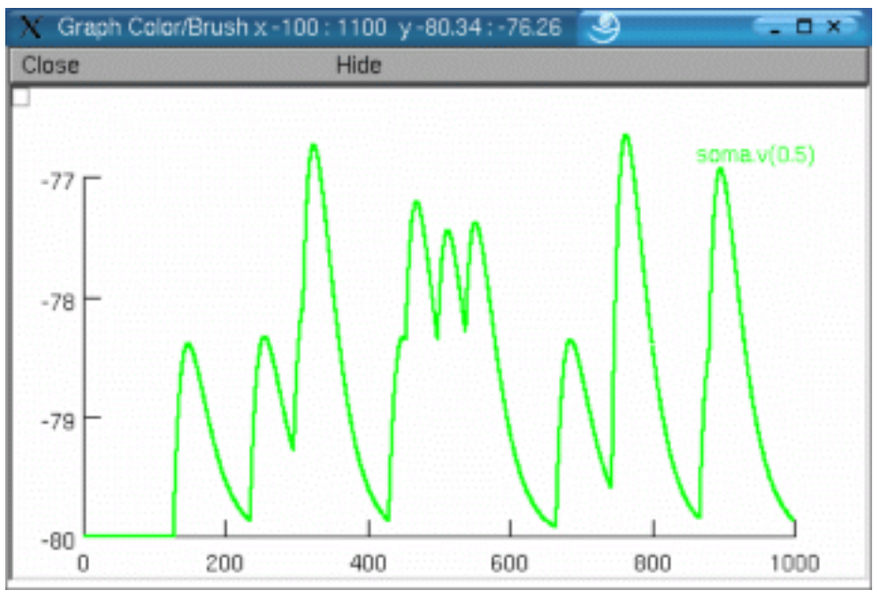


shape plot

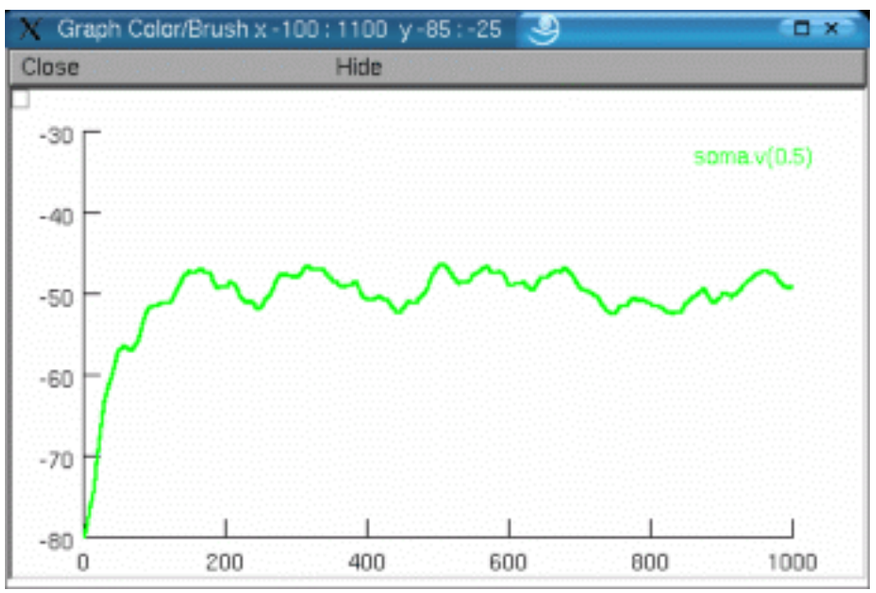
4 passive single-compartment model with exponential synapse



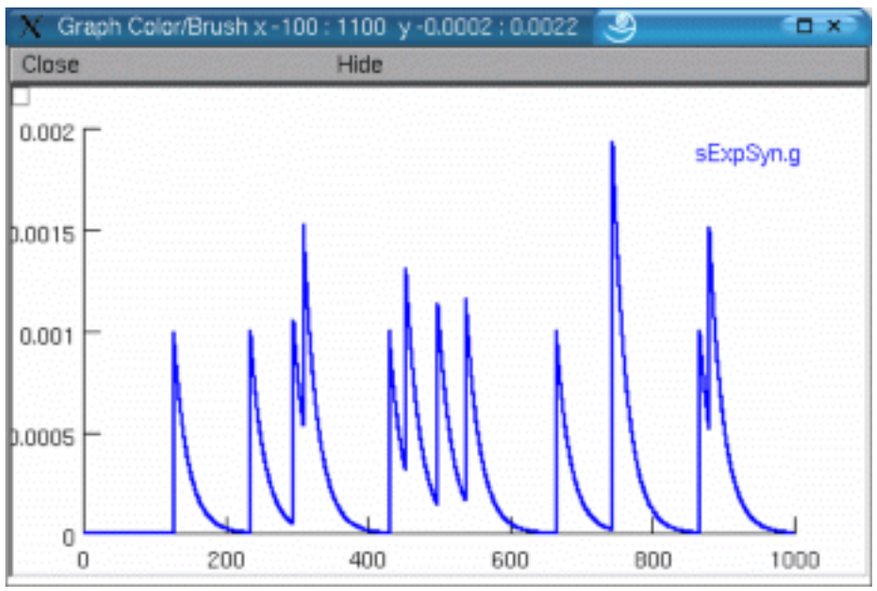
10 Hz

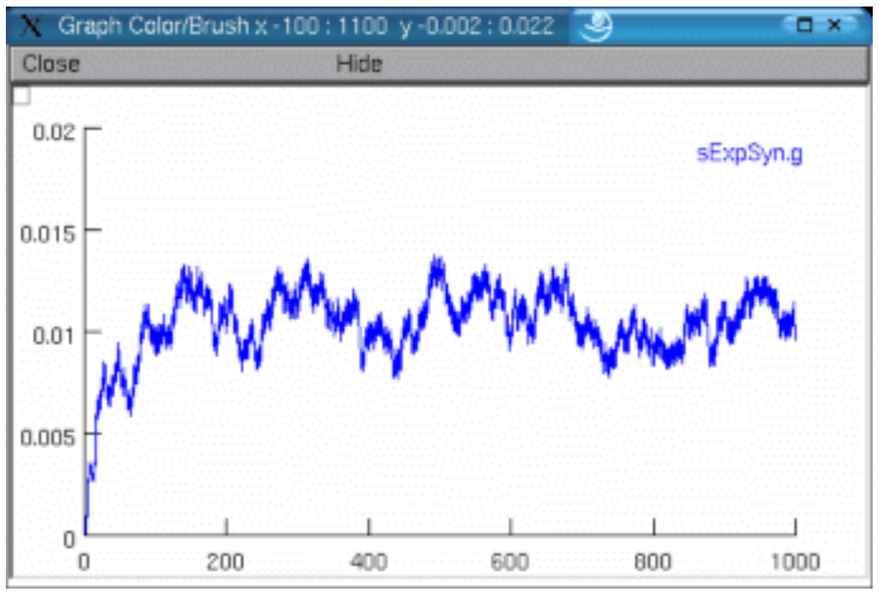


1000 Hz



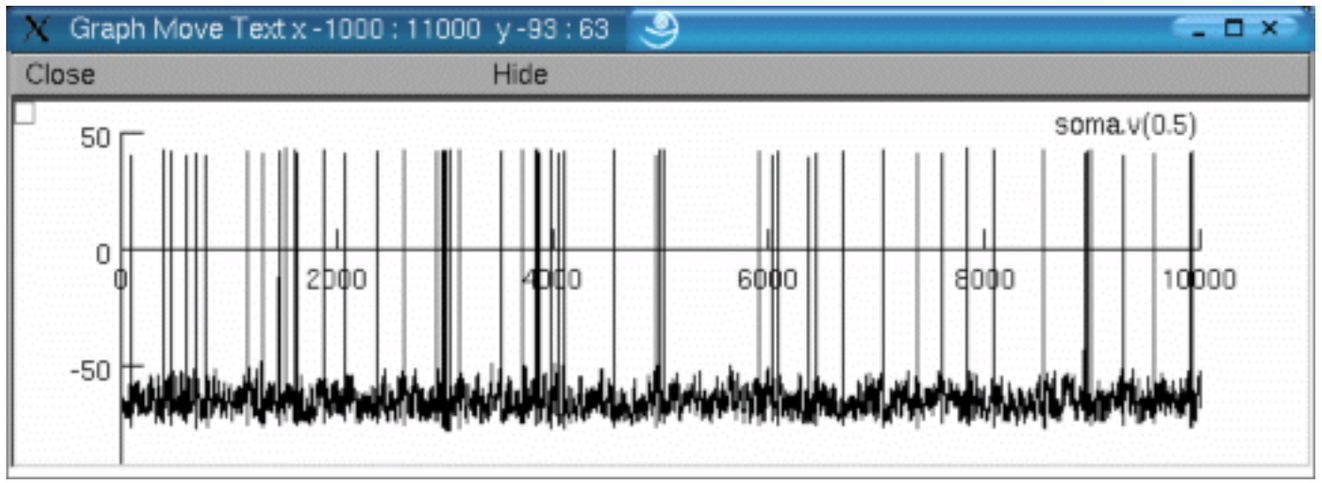
membrane potential



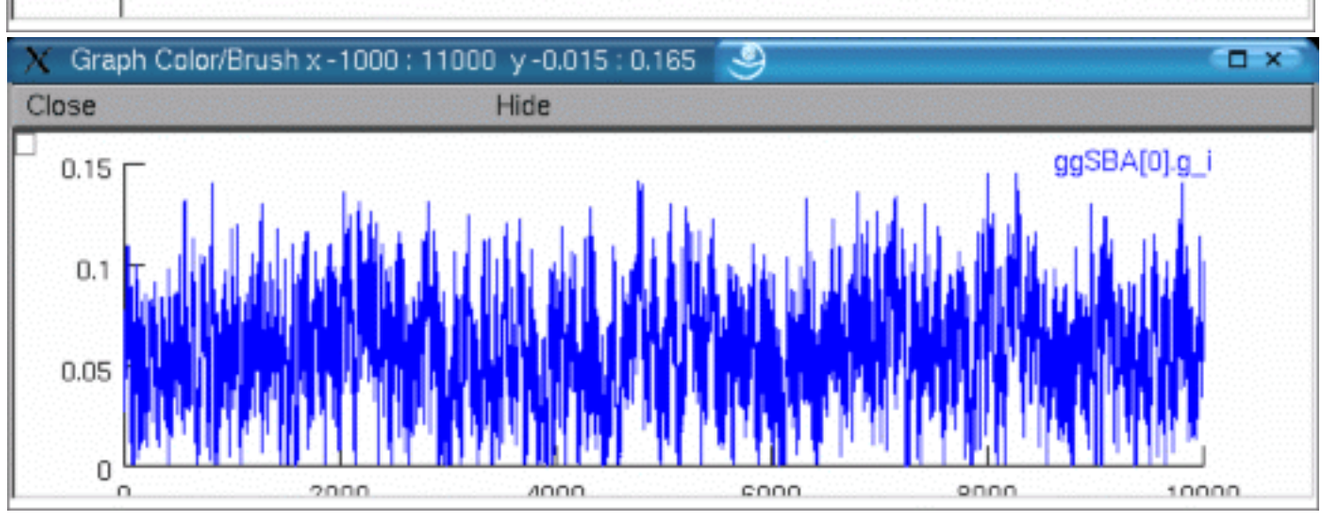


synaptic conductance

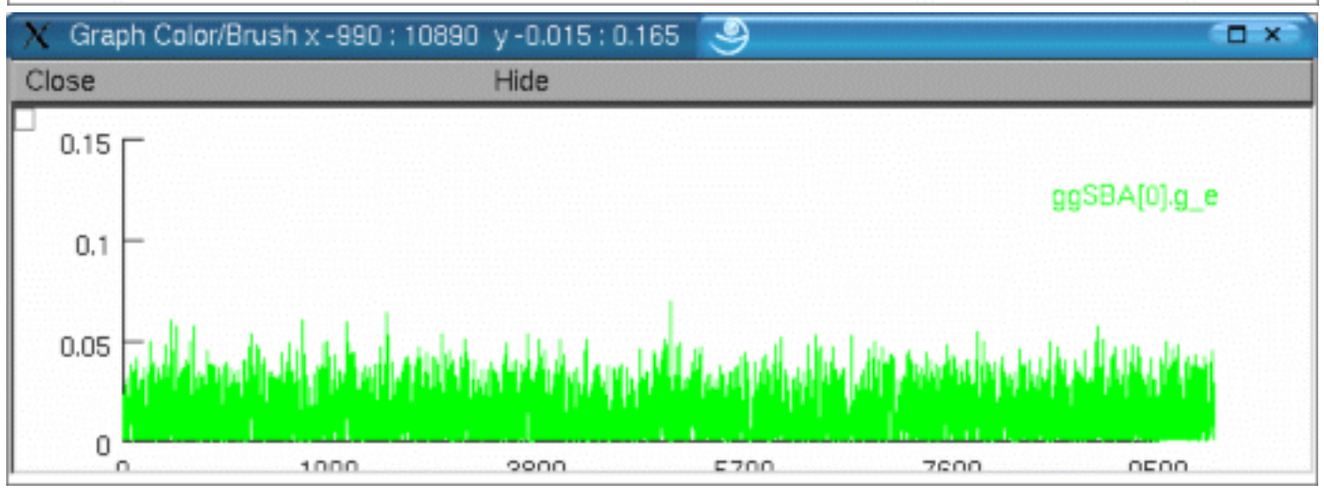
❺ point-conductance model of synaptic activity



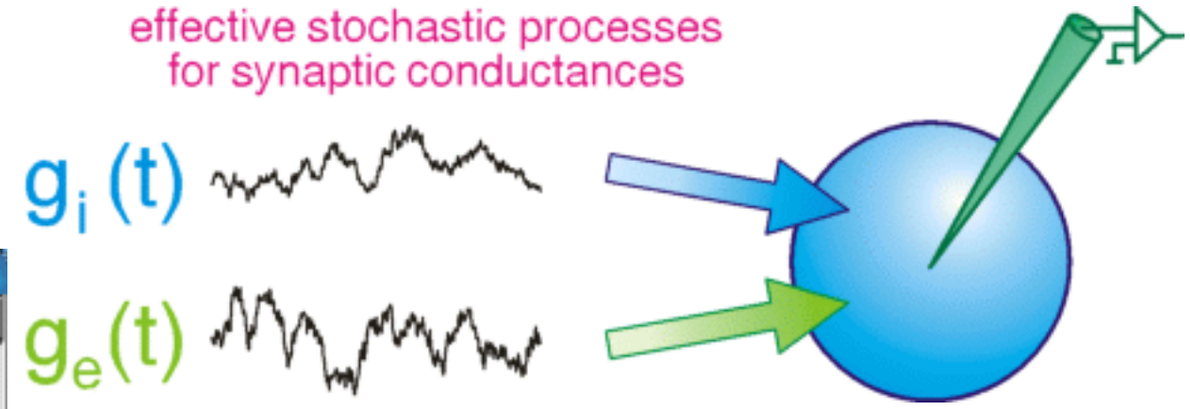
membrane potential



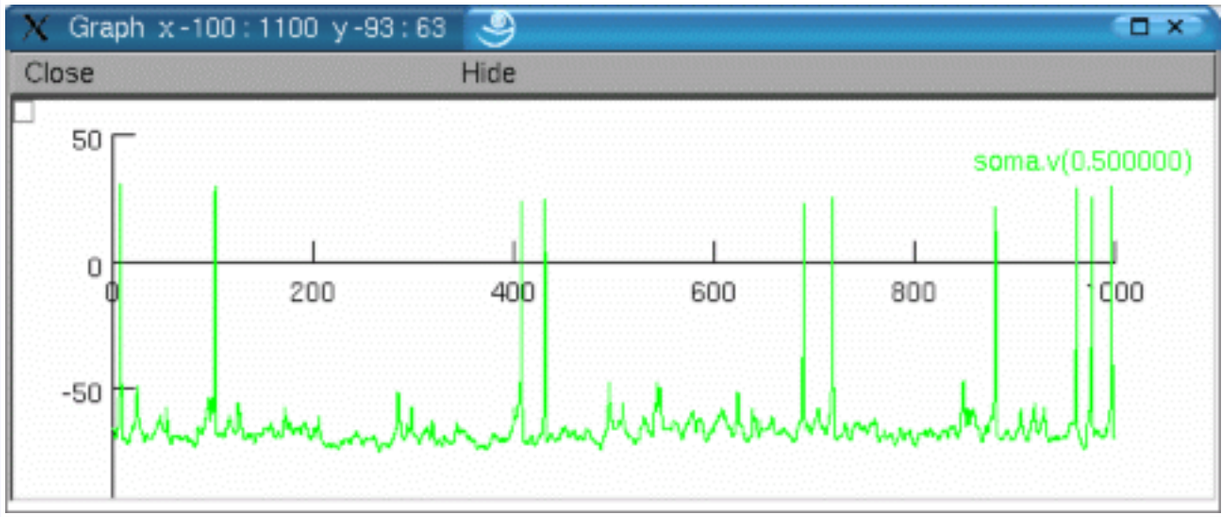
inhibitory synaptic conductance

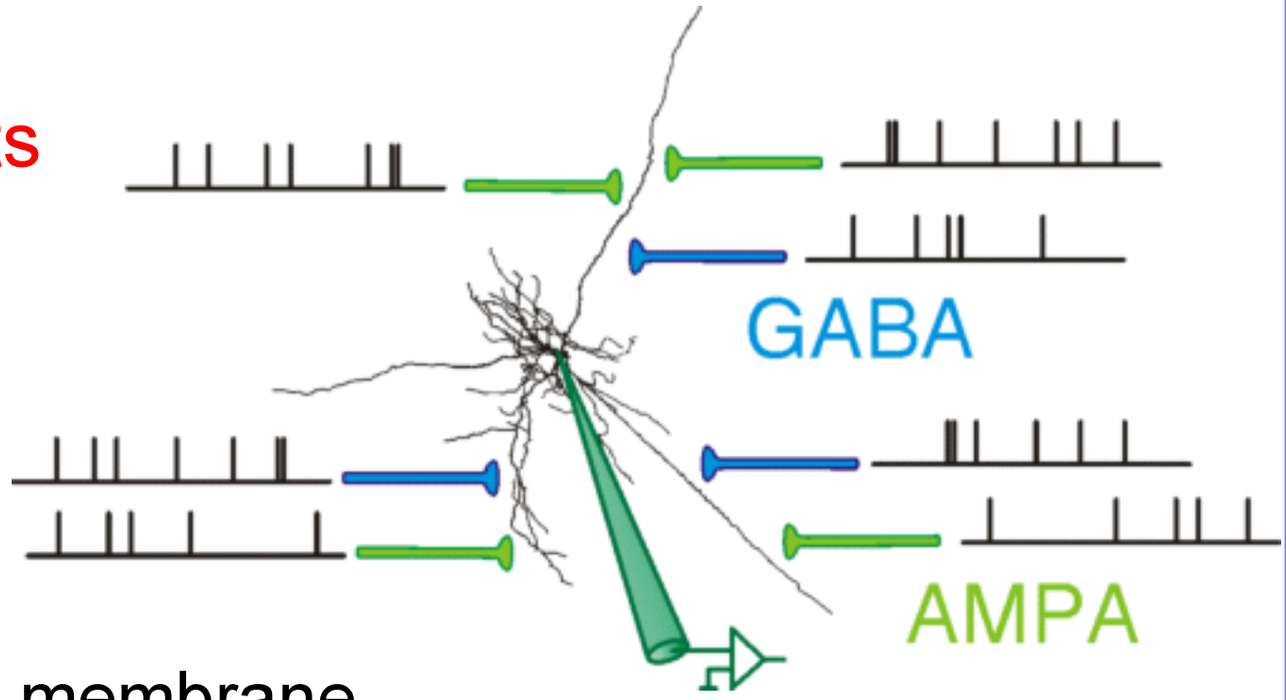


excitatory synaptic conductance

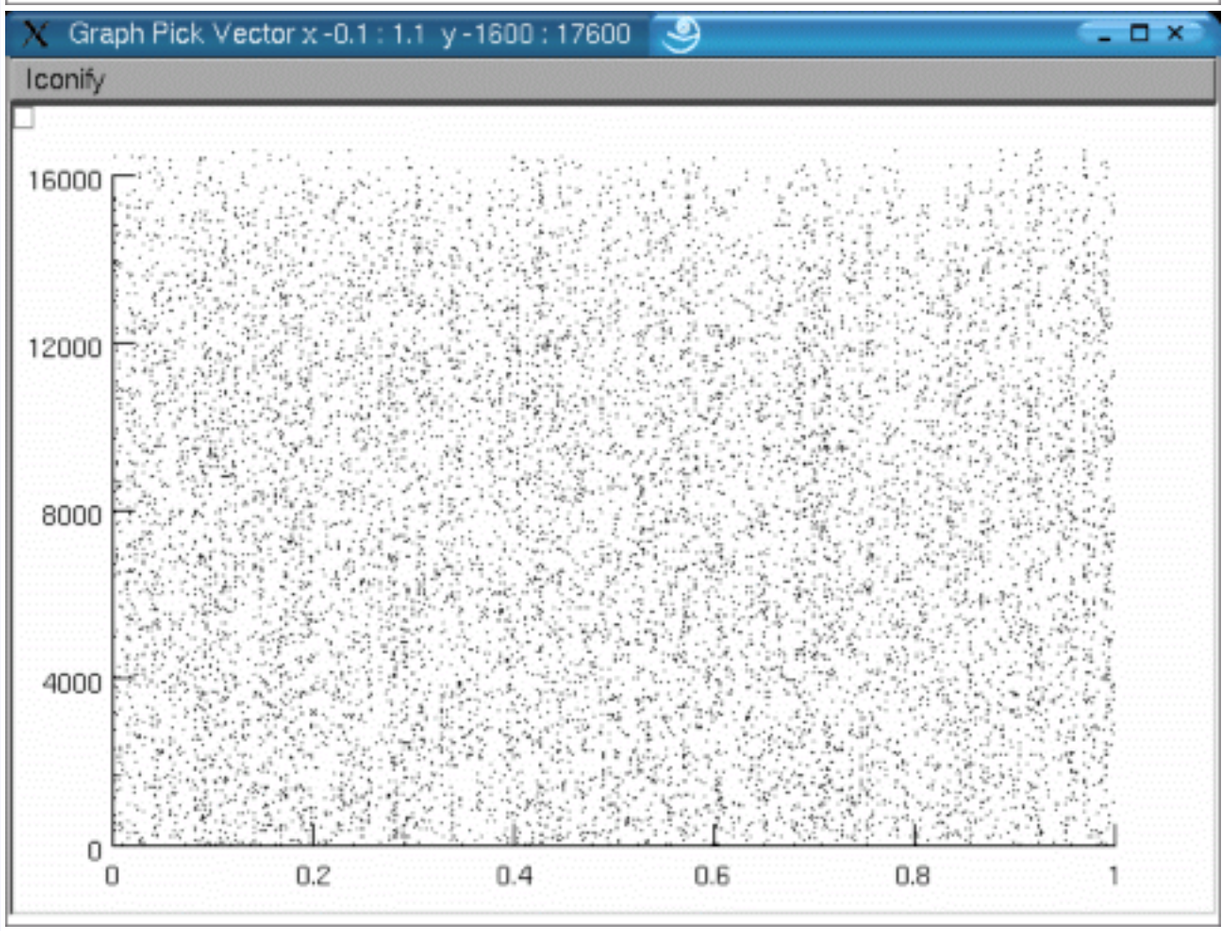


❻ detailed biophysical model with distributed synaptic inputs

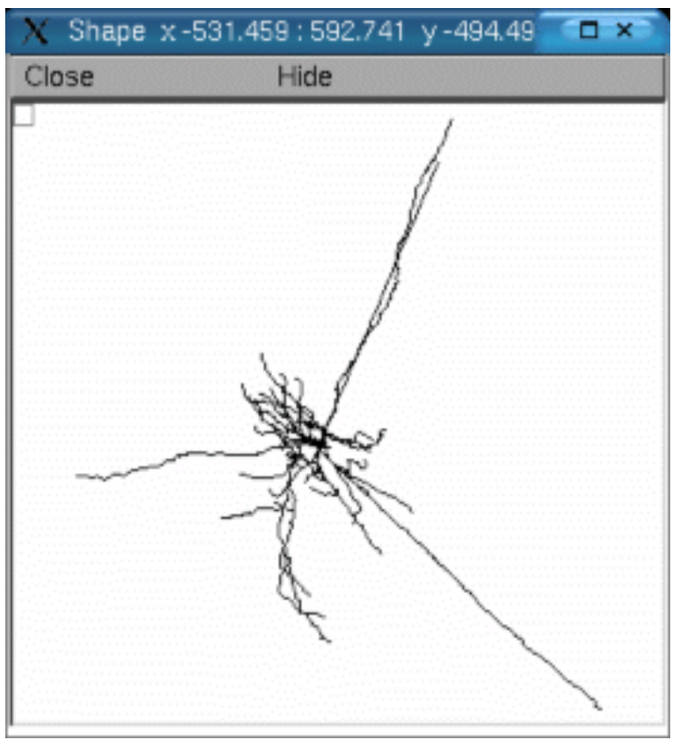




membrane potential

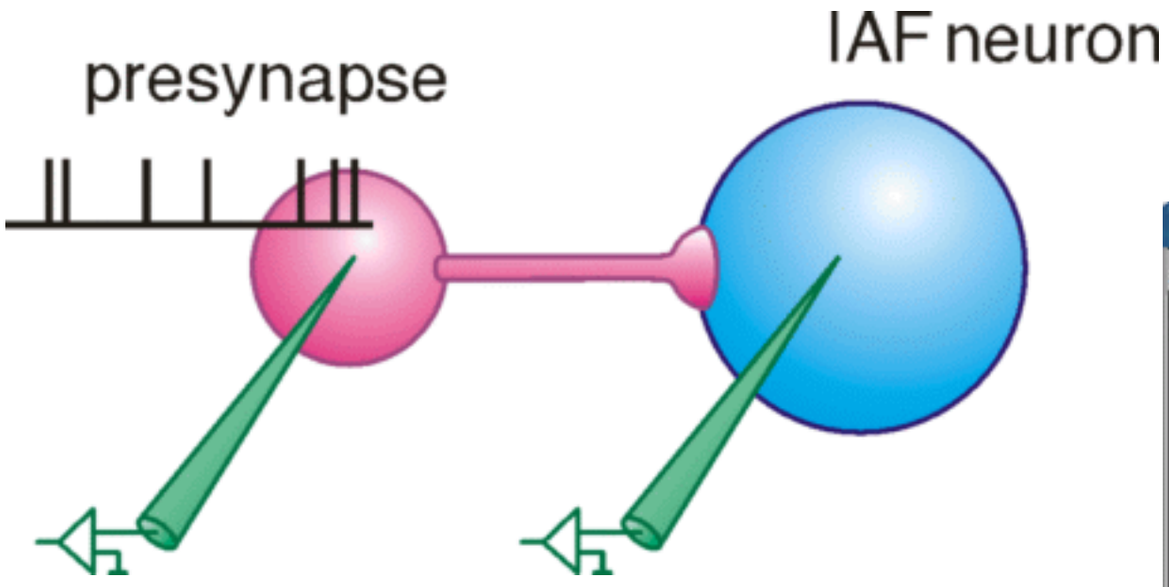


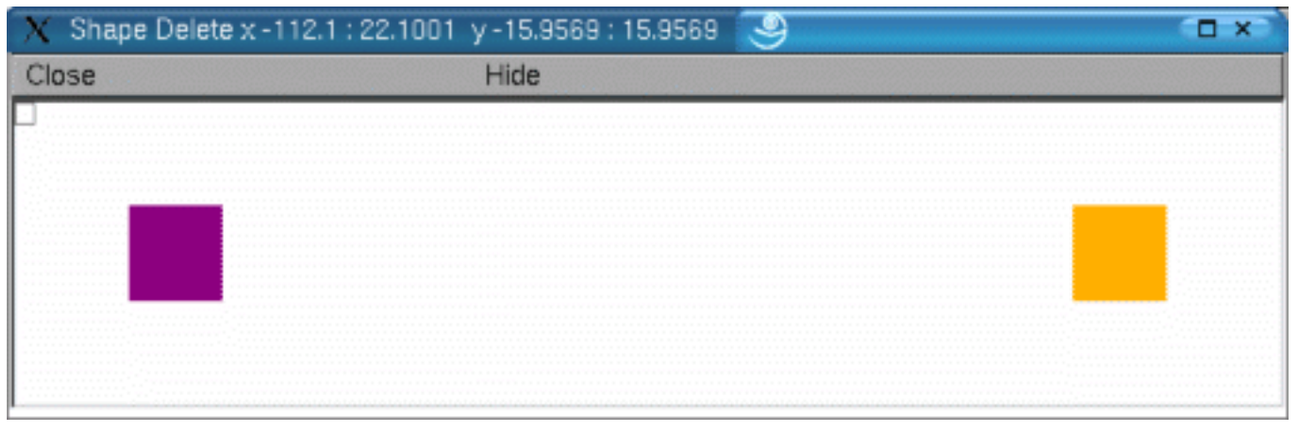
shape plot



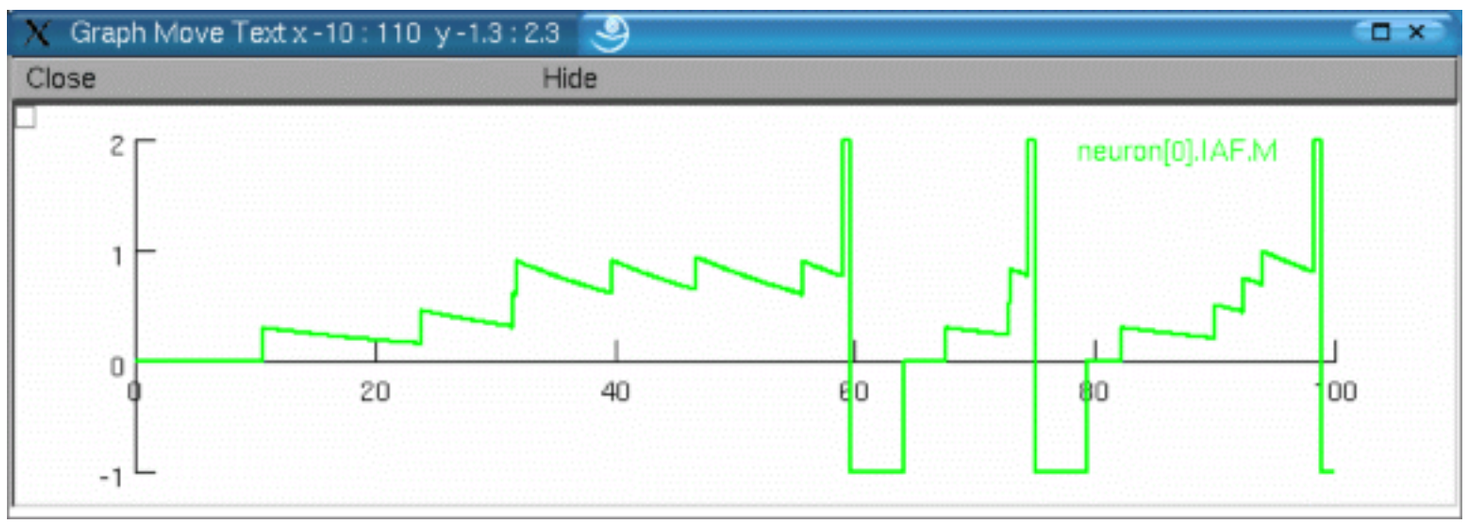
rasterplot of excitatory synaptic inputs

7 simple IAF neuron model with Poisson input

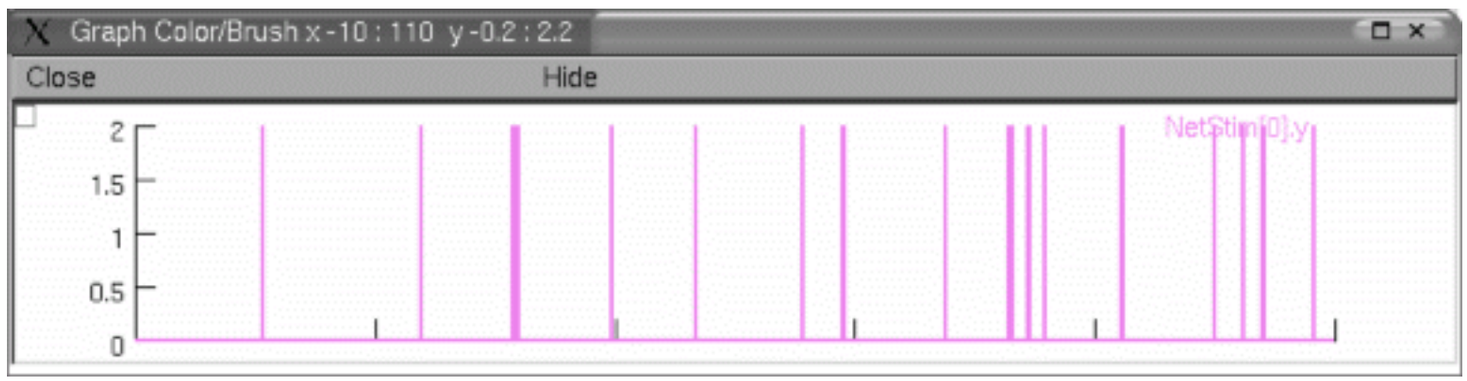




shape plot

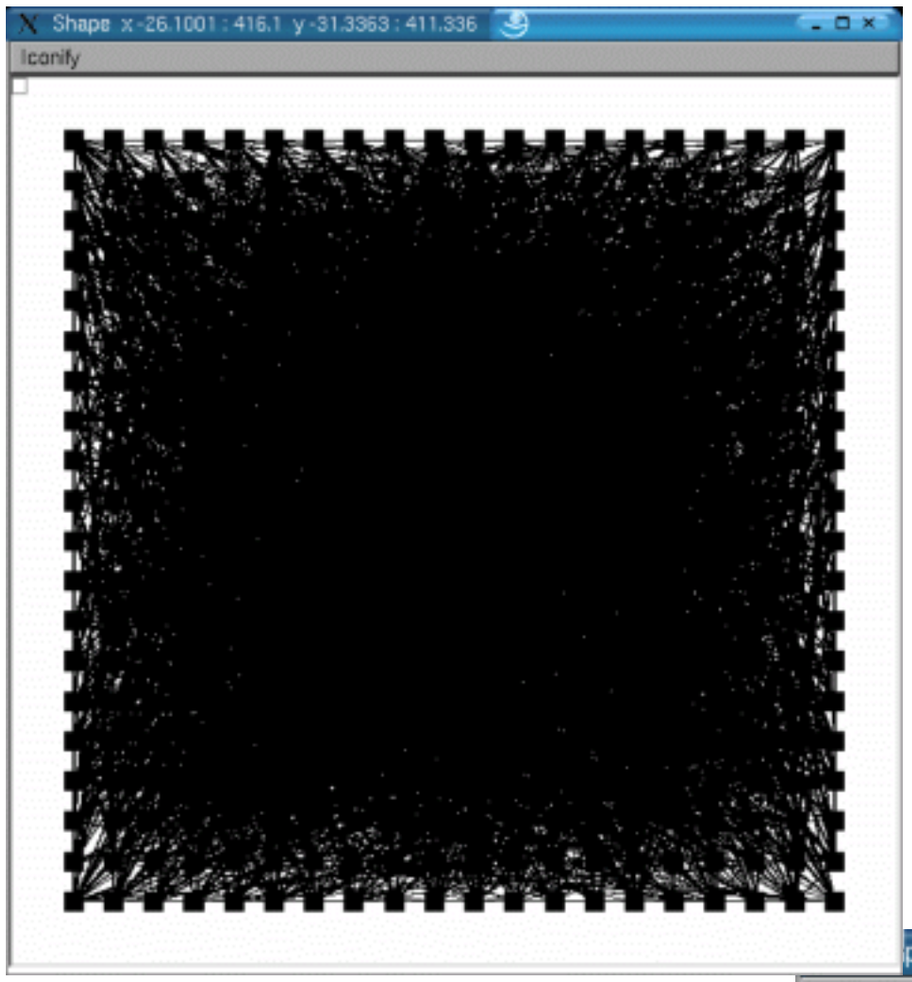


state variable
of IAF neuron



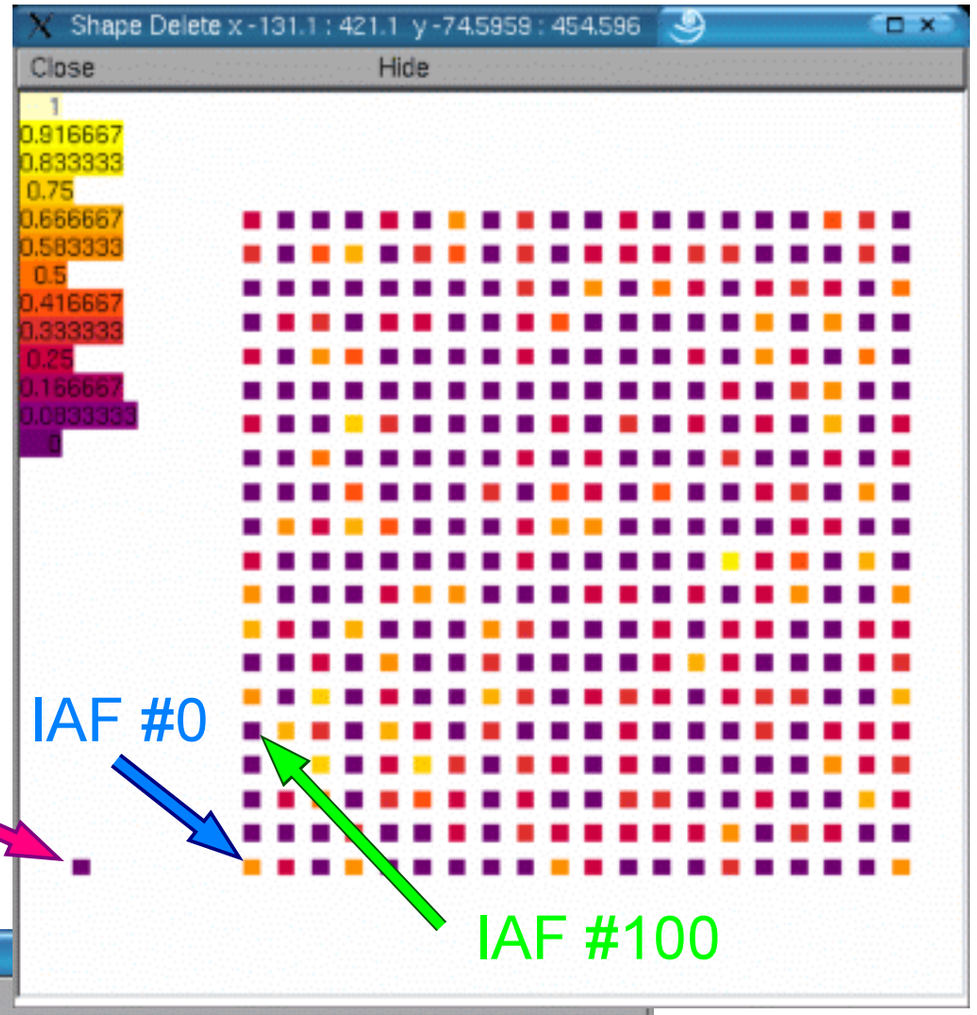
presynaptic
activity

❽ network of randomly connected IAF neurons



shape plot

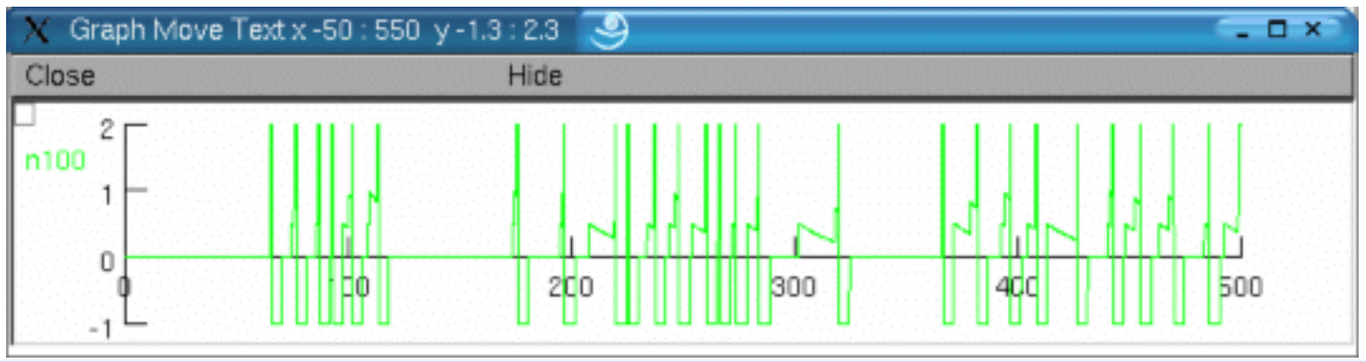
state of all IAF neurons



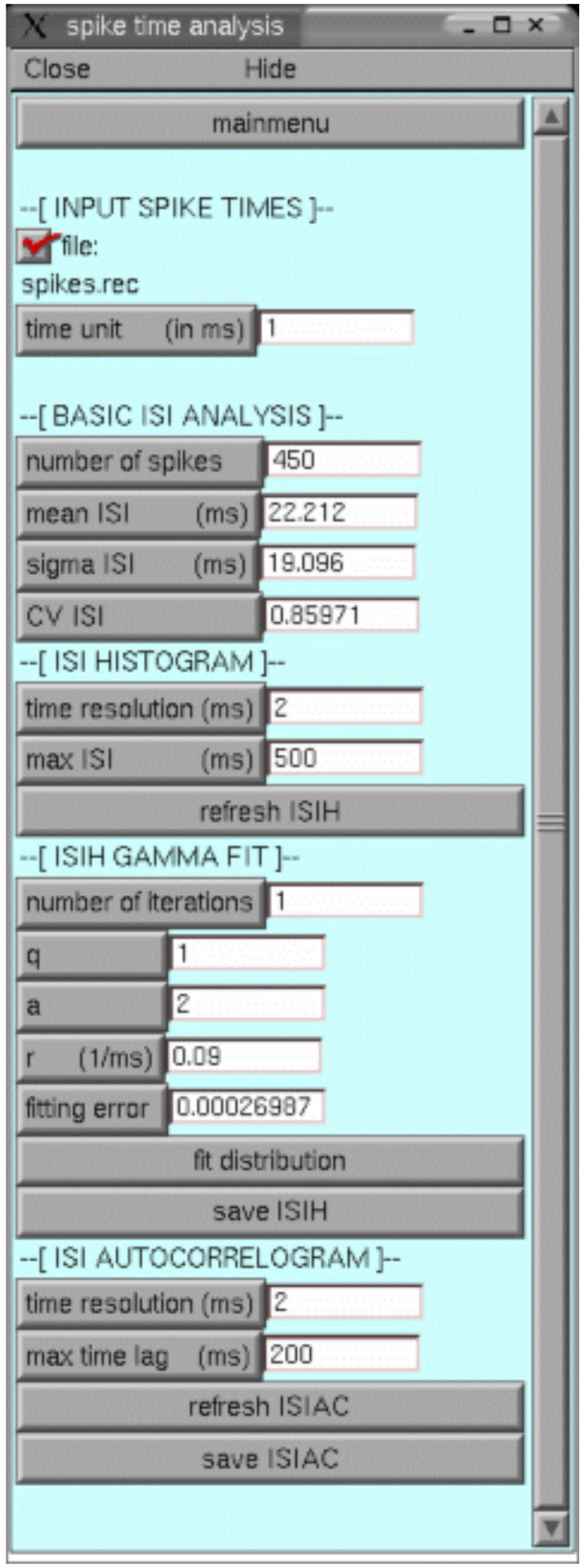
“presynapse”

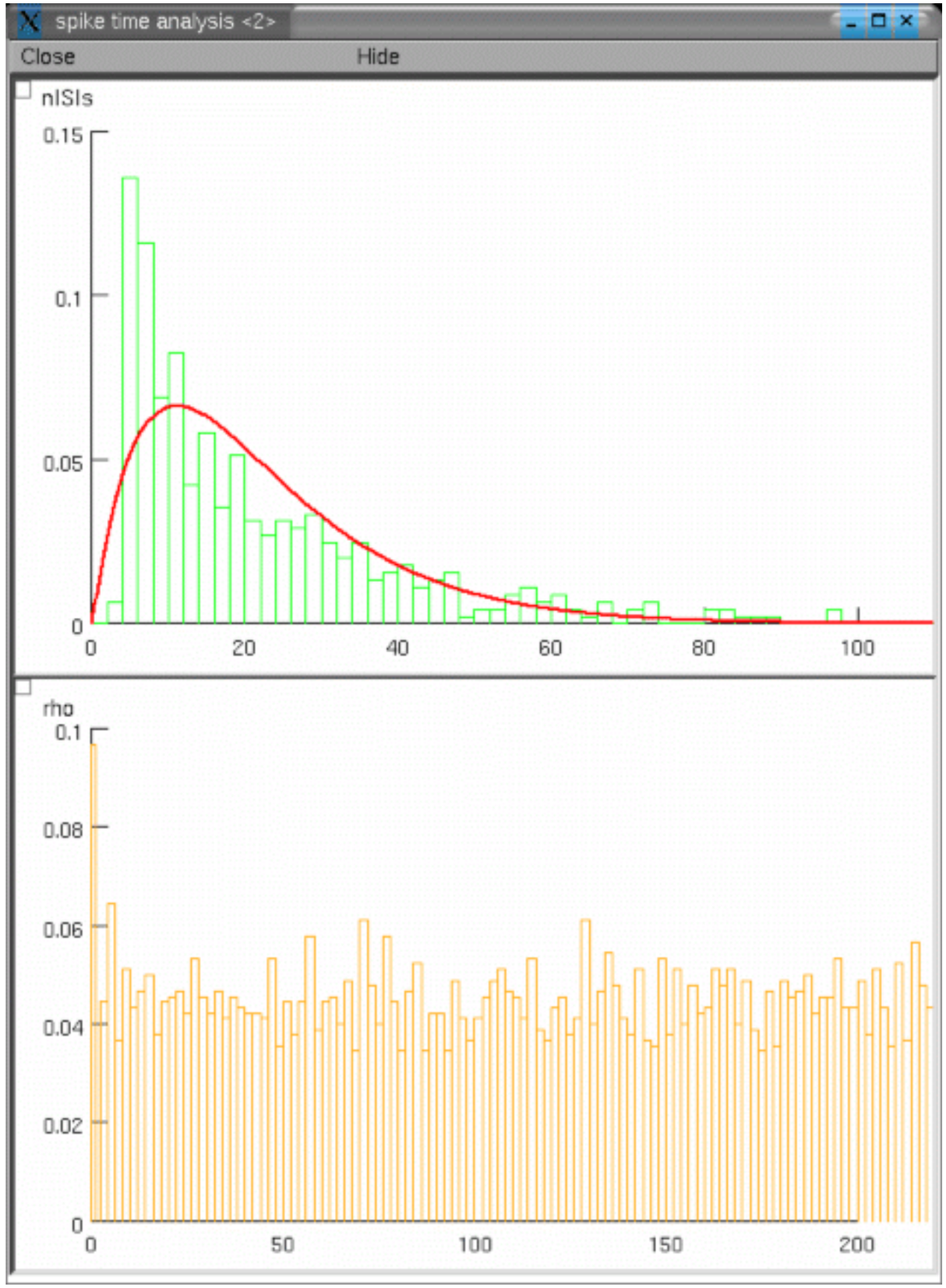
state of IAF neuron #1

state of IAF neuron #100



❾ statistical analysis and fitting of spike trains

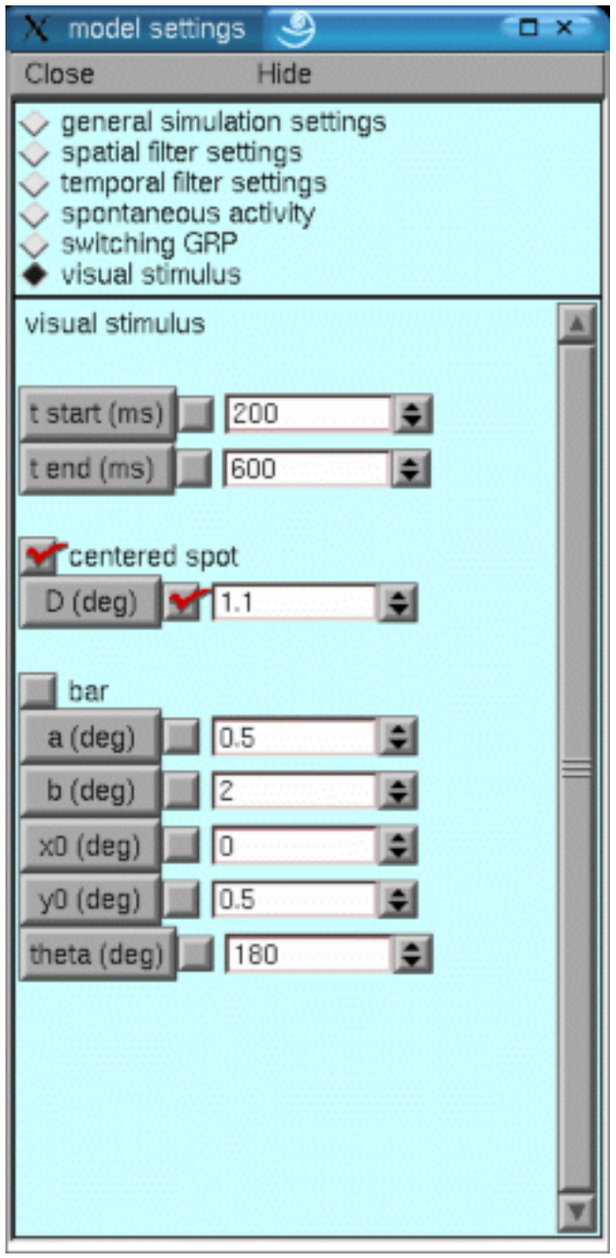




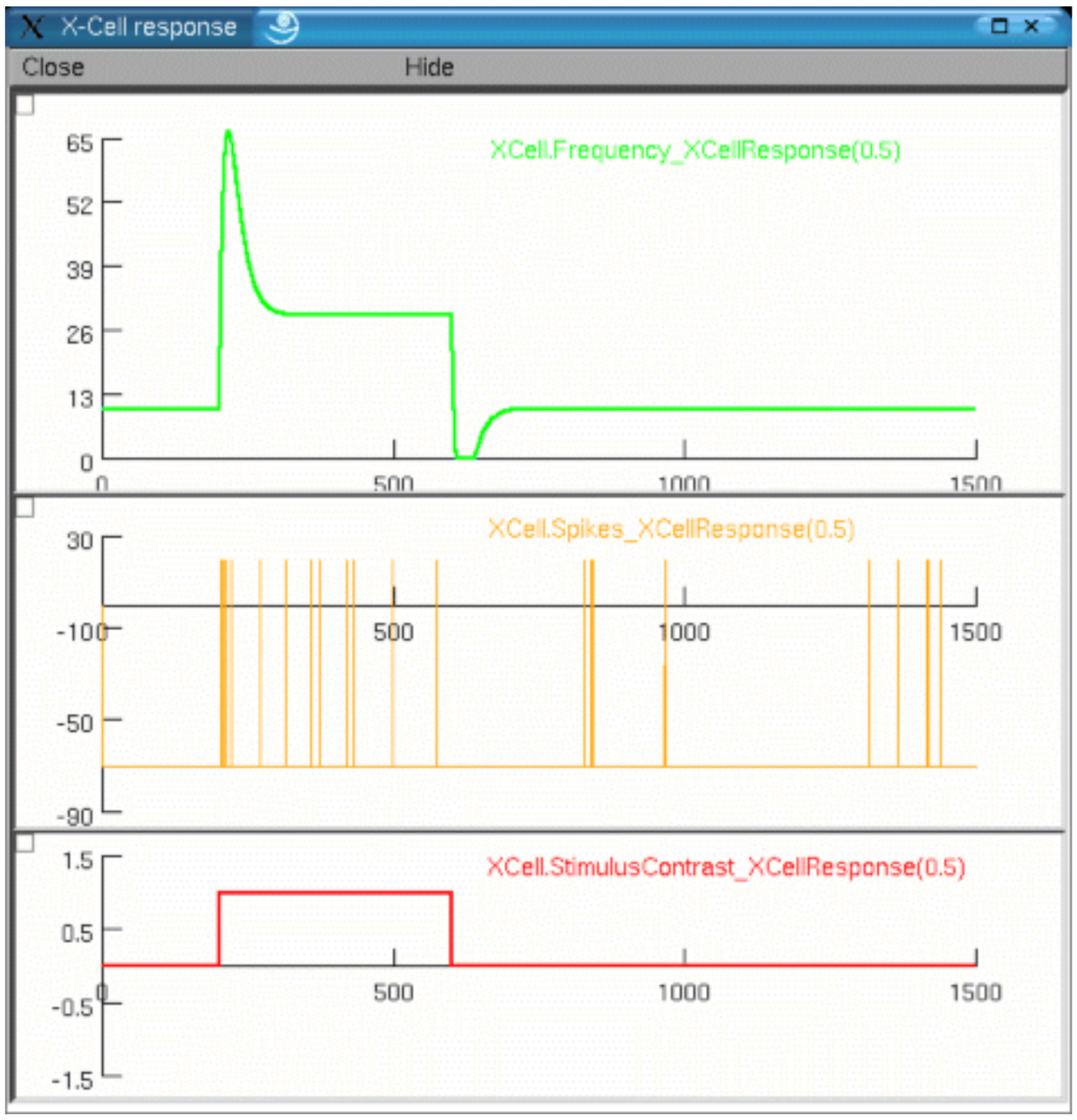
custom graphical interface

⑩ phenomeological model of cat geniculate nonlagged X-cells

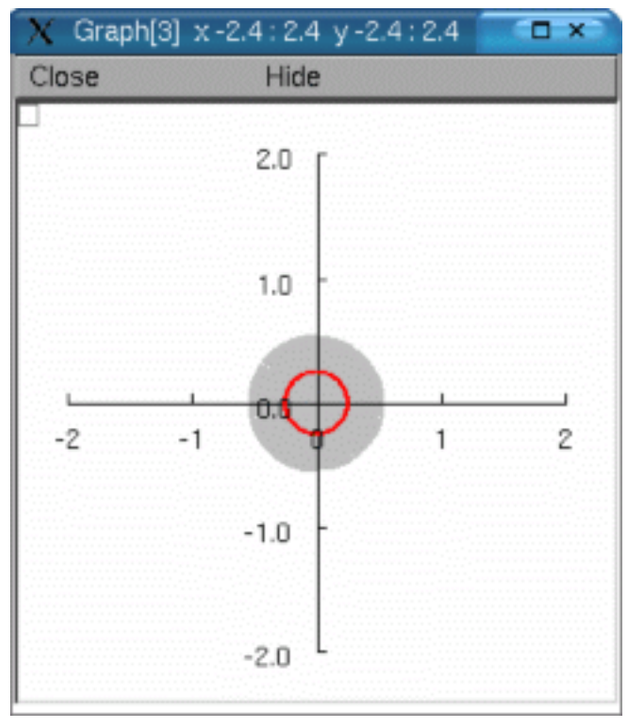
[Gazeres, Borg-Graham, Fregnac. *Visual Neuroscience* **15** (1998), 1157-1174]



parameter menu



X-cell response



visual input