

# NEURON - tutorial D of Gillies & Sterratt (part 1)

[http://web.mit.edu/neuron\\_v7.4/nrntuthtml/index.html](http://web.mit.edu/neuron_v7.4/nrntuthtml/index.html)

Lubica Benuskova

Lecture 9

How to program ion channels with **NMODL**

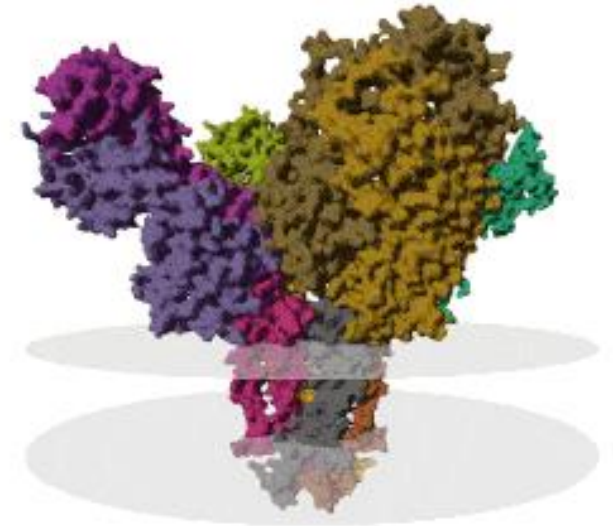
Centre for Cognitive Science

Bratislava

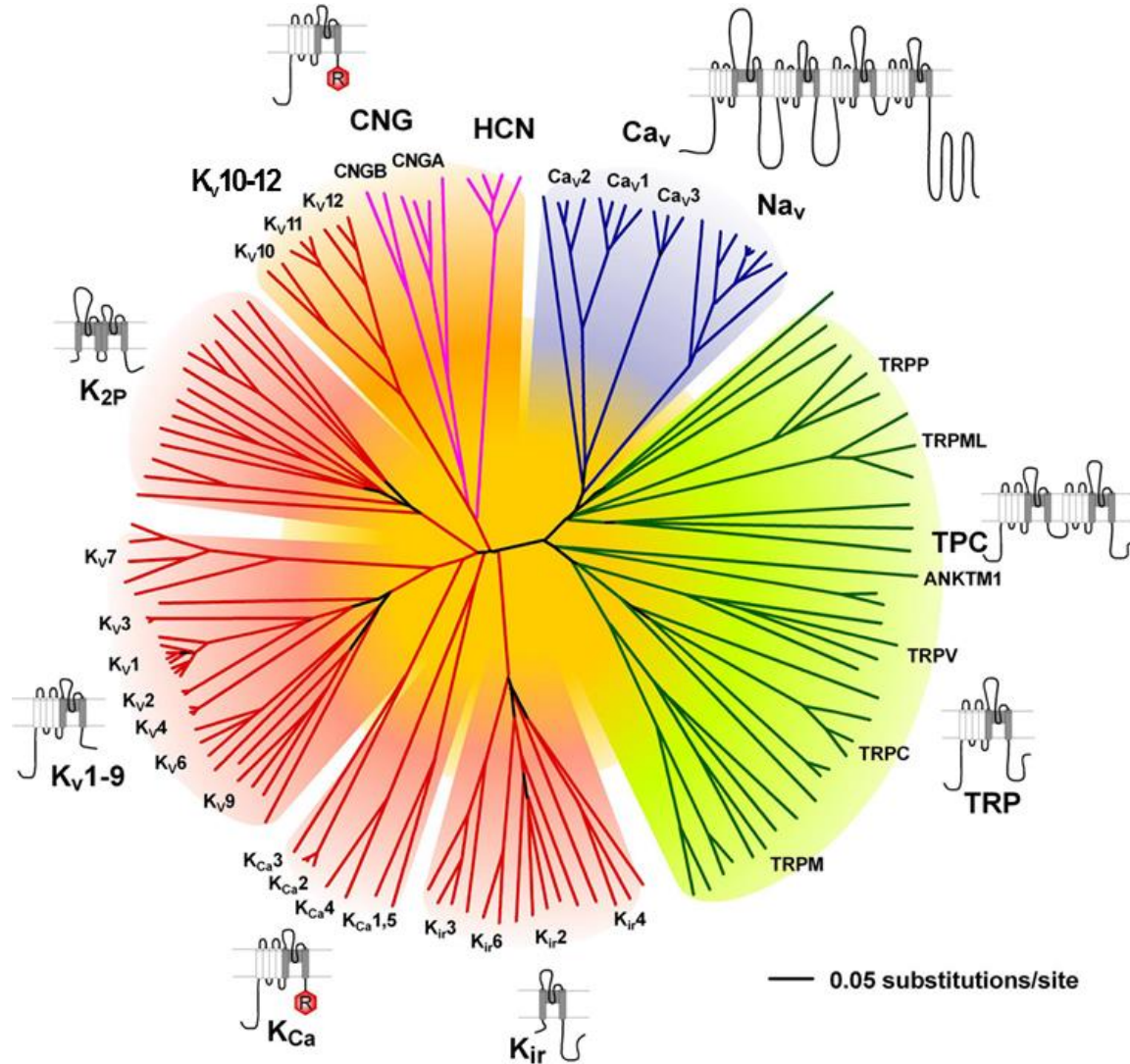


# Voltage-gated ion channels

- Voltage-gated ion channels for  $\text{Na}^+$ ,  $\text{Ca}^{2+}$ ,  $\text{K}^+$ , and  $\text{Cl}^-$  are **responsible for ion currents** flowing through the membrane and changes in membrane voltage.
- Channels differ with respect to particular voltage at which they open/close, how long they remain open/closed and their conductivity.
- Different neuron types possess different subtypes of ion channels allowing them to have a very **specific electrophysiological behaviour**.



# Dendrogram of ion channel families



- TRP channels behave like microscopic thermometers and are used in animals to sense hot or cold.
- TRPs act as sensors of osmotic pressure, volume, stretch, and vibration.
- TRP channels mediate a variety of sensations like the sensations of pain, temperature, different kinds of taste, pressure.

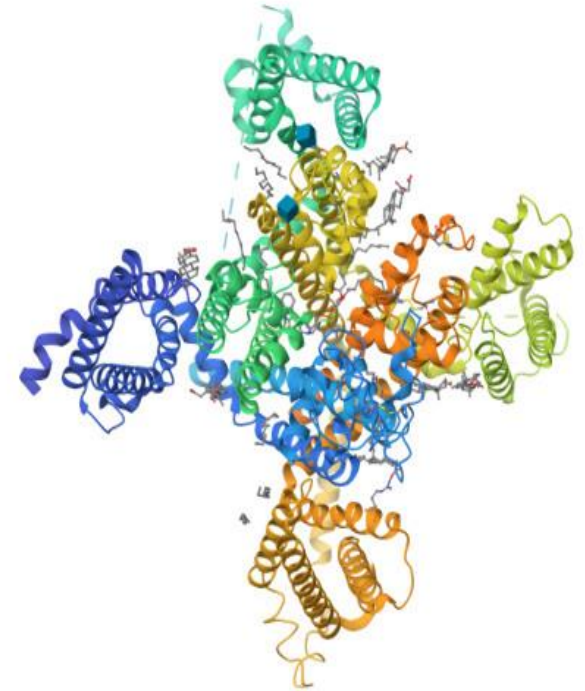
---

## Our current goal

- So far, our simulated neurons were given a characteristic morphology for a subthalamic nucleus neuron,
- BUT they still contain only the default Hodgkin and Huxley types of sodium  $\text{Na}^+$  and potassium  $\text{K}^+$  ion-selective channels.
- We would like to make these neurons more **electrophysiologically similar** to subthalamic nucleus neurons.
- There are several channel types we should add, but for now we will add only one new type, called the **T-type calcium channel**.

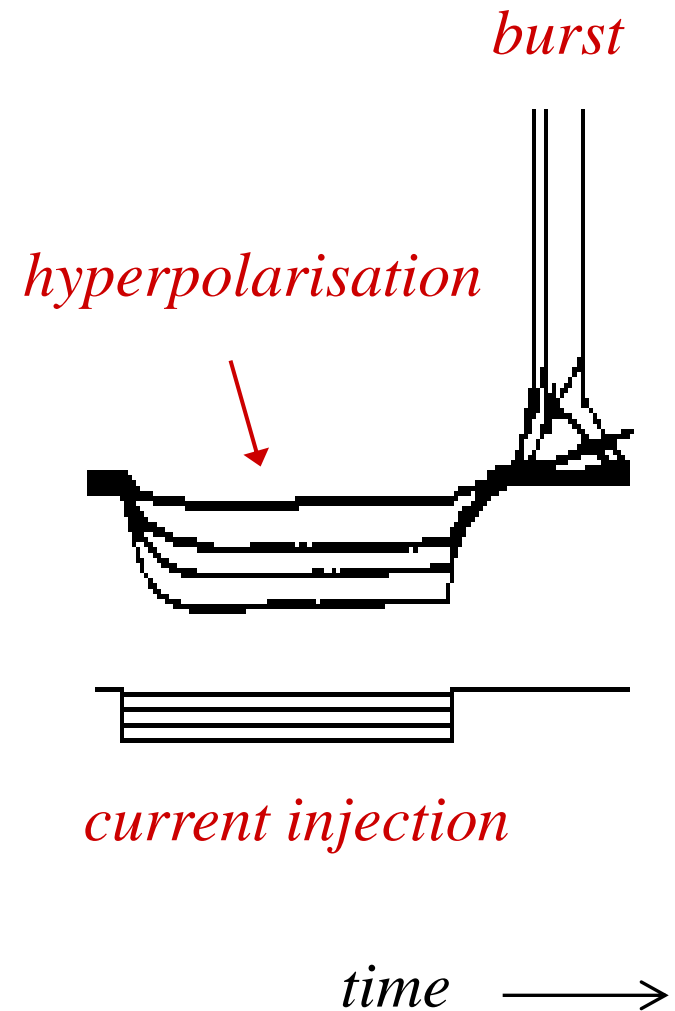
# T-type calcium channel

- We distinguish **T-type** (transient opening) calcium channels and the L-type (Long-lasting) calcium channels.
- The **T-type** channels are much different from the L-type calcium channels due to their ability to be activated by more **negative membrane potentials**.
- They are located within the brain, peripheral nervous system, heart, smooth muscle, bone, and endocrine system.



# Post-hyperpolarising response

- A typical electrophysiological feature of subthalamic neurons is the **post-hyperpolarising** firing.
- When a neuron is hyperpolarised (e.g., by negative current injection or inhibitory synaptic input), at the end of the hyperpolarisation, a burst of spikes is observed.
- This response is mediated by a low threshold calcium selective ion channel, called the **T-type Ca-channel**.



## T-type Ca electric current

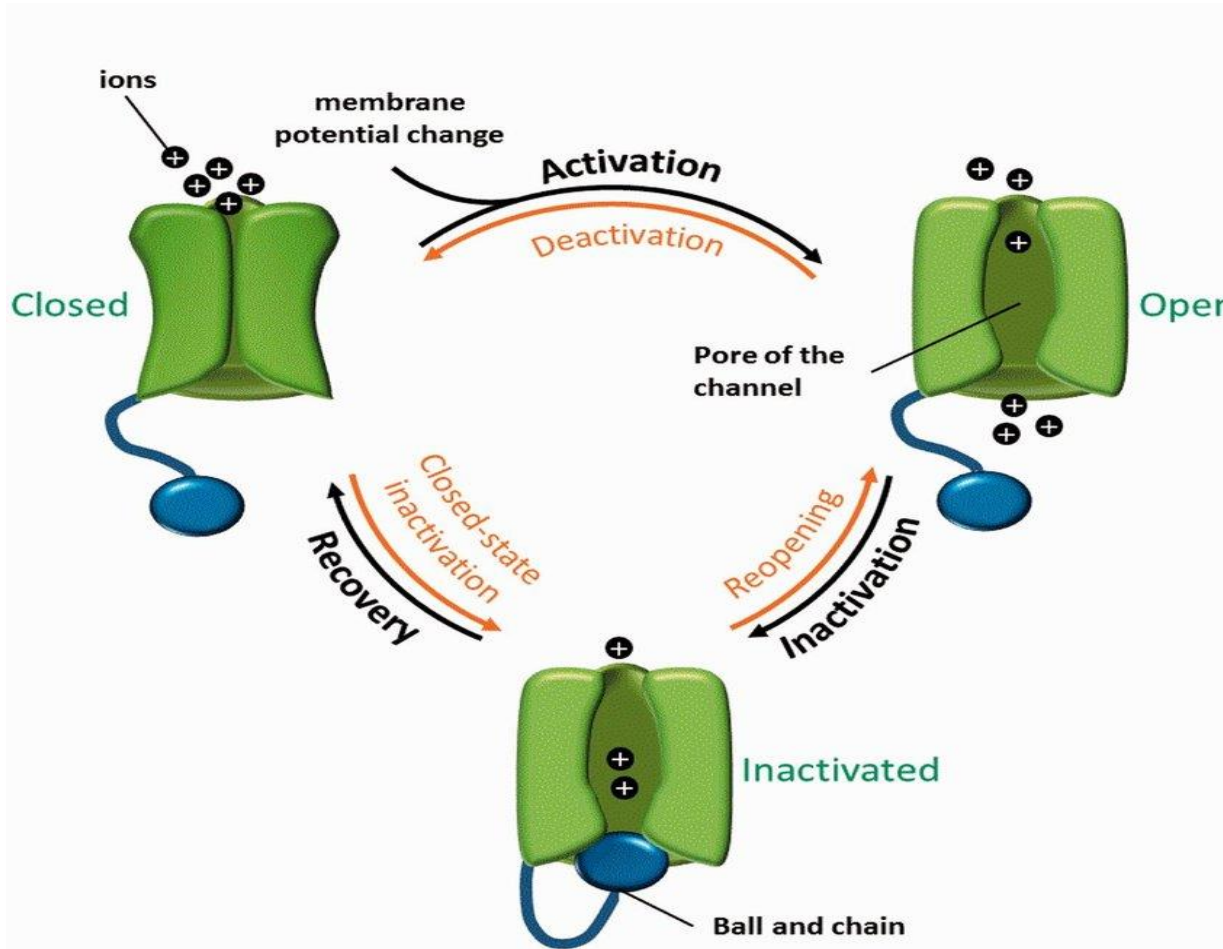
- First, we must know the equation that describes the electric current that flows through the T-type Ca-channel.
- The current  $I_T$  produced from the T-type Ca-channels was characterized within the Hodgkin-Huxley framework by Wang et al. (J. Neurophys. 66: 839-850, 1992):

$$I_T = g_{T(max)} r^3 s (V - E_{Ca})$$

- where  $g_{T(max)}$  is the maximum T-type Ca conductance;  $r$  is the **activation** state variable;  $s$  is the **inactivation** state variable,  $E_{Ca}$  is the reversal potential for  $Ca^{2+}$ ; and  $V$  is the neuron membrane potential.

# Three-state model of voltage-gated ion channels

- Voltage-gated ion channels go through three stages or states.



Source: Hinard et al, 2016, DOI:[10.1093/database/baw017](https://doi.org/10.1093/database/baw017)



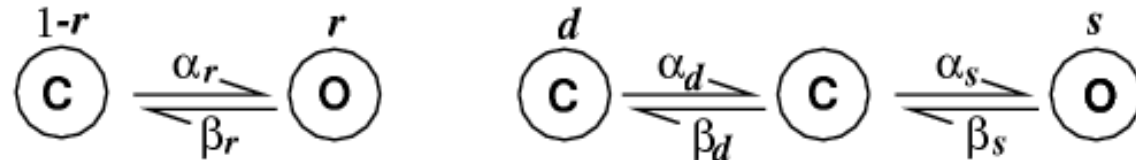
---

## Three-state model of voltage-gated ion channels

- **Activation:** the channel voltage-sensor, a subunit consisting of a collection of charged amino acids, moves under the influence of the membrane electrical field, thus opening the pore.
- **Inactivation:** voltage-gated ion channels go through an inactivated state during which the channel is non-conducting and refractory to open, so-called inactivation.
- **Recovery:** the inactivated state is followed by the return to the closed state via a transition named recovery from inactivation.

## T-type Ca channel: kinetic scheme

- "C" = closed state and "O" = open state of the channel. Letter  $r$  denotes an activation state variable. Channel can close and open spontaneously – left hand side schematic.



- Right side: Letters  $s$  denotes fast process of opening from the closed state, letter  $d$  means slow process of recovery from inactivated to closed state.
- The  $\alpha$  and  $\beta$  denote the forward and backward rate constants from one state to another, respectively. They are voltage dependent functions specified by Wang et al. (1991).

## T-type Ca channel: rate functions $\alpha$ & $\beta$

- Rate functions obey these equations (J. Neurophys. 66: 839-850):

$$\alpha_r = 1 / \left( 1.7 + e^{-\frac{V+28.2}{13.5}} \right)$$

$$\beta_r = e^{-\frac{V+63.0}{7.8}} / \left( 1.7 + e^{-\frac{V+28.2}{13.1}} \right)$$

$$\alpha_s = e^{-\frac{V+160.3}{17.8}}$$

$$\beta_s = e^{-\frac{V+160.3}{17.8}} \left( \sqrt{0.25 + e^{-\frac{V+83.5}{6.3}}} - 0.5 \right)$$

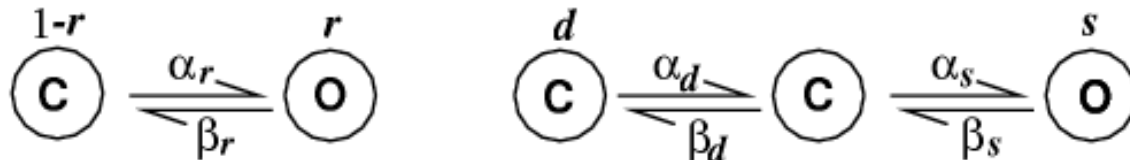
$$\alpha_d = \left( 1.0 + e^{-\frac{V+37.4}{30.0}} \right) / (240.0 (0.5 + d))$$

$$\beta_d = \alpha_d (d - 0.5)$$

$$d = \sqrt{0.25 + e^{-\frac{V+83.5}{6.3}}}$$

## T-type Ca channel: from scheme to equations

- A kinetic scheme is a network of states and connections between them representing the scheme of a dynamical process.



- These kinetic schemes translate to three differential equations:

$$\begin{aligned}\dot{r} &= \alpha_r(1-r) - \beta_r r \\ \dot{s} &= \alpha_s(1-s-d) - \beta_s s \\ \dot{d} &= \alpha_d(1-s-d) - \beta_d d\end{aligned}$$

---

# NEURON model description language

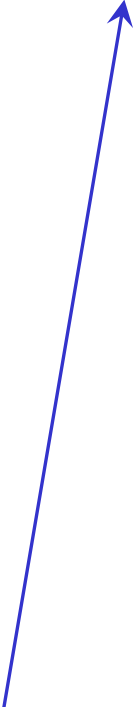
- We would like to add this channel to our model. This cannot be done with the programming language **hoc** that we have used in the previous tutorials.
- Instead, we use the **NEURON Model Description Language (NMODL)** provided for defining additional membrane mechanisms such as ion channels or ion pumps or point processes such as synapses.
- There is more at:  
[https://neuron.yale.edu/neuron/static/py\\_doc/modelspec/programmatic/mechanisms/nmodl.html](https://neuron.yale.edu/neuron/static/py_doc/modelspec/programmatic/mechanisms/nmodl.html)

# CaT.mod

- A membrane mechanism description using NMODL is laid out in an ordinary text file. The text file **CaT.mod** containing a specification of the T-type Ca channel in NMODL looks like this:

```
TITLE Calcium T channel for
Subthalamic Nucleus
UNITS {
    (mV) = (millivolt)
    (mA) = (milliamp)
}
NEURON {
    SUFFIX CaT
    USEION ca READ eca WRITE ica
    RANGE gmax
}
PARAMETER {
    gmax = 0.002 (mho/cm2)
}
```

```
ASSIGNED {
    v (mV)
    eca (mV)
    ica (mA/cm2)
    ralpha (/ms)
    rbeta (/ms)
    salpha (/ms)
    sbeta (/ms)
    dalpha (/ms)
    dbeta (/ms)
}
STATE {
    r s d
}
```



# CaT.mod — contd.

```
BREAKPOINT {  
    SOLVE states METHOD cnexp  
    ica = gmax*r*r*r*s*(v-eca)  
}  
  
INITIAL {  
    settables(v)  
    r = ralpha/(ralpha+rbeta)  
    s = (salpha*(dbeta+dalpha) - (salpha*dbeta)) /  
        ((salpha+sbeta)*(dalpha+dbeta) - (salpha*dbeta))  
    d = (dbeta*(salpha+sbeta) - (salpha*dbeta)) /  
        ((salpha+sbeta)*(dalpha+dbeta) - (salpha*dbeta))  
}  
  
DERIVATIVE states {  
    settables(v)  
    r' = ((ralpha*(1-r)) - (rbeta*r))  
    d' = ((dbeta*(1-s-d)) - (dalpha*d))  
    s' = ((salpha*(1-s-d)) - (sbeta*s))  
}
```

# CaT.mod – finish

```
UNITSOFF
```

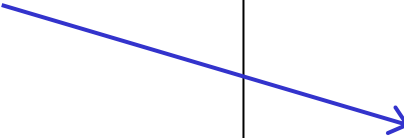
```
PROCEDURE settables(v (mV)) {  
    LOCAL bd  
    TABLE ralpha, rbeta, salpha, sbeta, dalpha, dbeta  
        FROM -100 TO 100 WITH 200  
  
    ralpha = 1.0/(1.7+exp(-(v+28.2)/13.5))  
    rbeta  = exp(-(v+63.0)/7.8)/(exp(-(v+28.8)/13.1)+1.7)  
  
    salpha = exp(-(v+160.3)/17.8)  
    sbeta  = (sqrt(0.25+exp((v+83.5)/6.3))-0.5)*(exp(-(v+160.3)/17.8))  
  
    bd      = sqrt(0.25+exp((v+83.5)/6.3))  
    dalpha  = (1.0+exp((v+37.4)/30.0))/(240.0*(0.5+bd))  
    dbeta   = (bd-0.5)*dalpha  
}  
UNITSON
```



# NEURON and NMODL

- Suppose we have created a text file **CaT.mod** containing our description of the T-type Ca channel in NMODL and we want to insert it into the soma.
- We simply do it like this (see **sthD.hoc**):

```
soma {  
    nseg = 1  
    diam = 18.8  
    L = 18.8  
    Ra = 123.0  
    insert hh  
    gnabar_hh = 0.25  
    gl_hh = .0001667  
    el_hh = -60.0  
    insert CaT  
}
```



## Equilibrium potentials

- Unfortunately, NEURON's default values for **eca**, **ena**, and **ek** are not appropriate for our mammalian subthalamic nucleus.
- We will reset these values according to the measured ion concentrations and calculated equilibrium potentials at 37° Celsius using the Nernst equation (Johnston & Wu, MIT Press, 1995):

<b>Na</b>		<b>K</b>		<b>Ca</b>	
<i>inside</i>	<i>outside</i>	<i>inside</i>	<i>outside</i>	<i>inside</i>	<i>outside</i>
10 mM	145 mM	140 mM	5 mM	2e-4 mM	2.5 mM
ena = 71.5 mV		ek = -89.1 mV		eca = 126.1 mV	

## New SThcell template

- We will reset all these equilibrium potentials to the new values.
- These should be set **after** the channel mechanisms are inserted into a section and *must be set for each section* (i.e., for soma and each dendrite).
- It is done within our SThcell template like this:

```
soma {  
    nseg = 1  
    diam = 18.8  
    L = 18.8  
    Ra = 123.0  
    insert hh  
    ena = 71.5  
    ek = -89.1  
    gnabar_hh = 0.25  
    gl_hh = .0001667  
    el_hh = -60.0  
    insert CaT  
    eca = 126.1  
}
```

## Note on how NEURON deals with ions

- Since we have just introduced  $i_{ca}$ , a calcium current, we may expect NEURON will automatically adjust the intra- and extracellular calcium concentrations. **It doesn't !!!**
- NEURON does not change the ionic concentrations automatically. To do this, we would need another mechanism defined in NMODL that would implement  $ca_i$  and/or  $ca_o$ , the intra- and extracellular calcium concentrations. However, this mechanism would need to know the total calcium current  $i_{ca}$  originating from our CaT mechanism and any other mechanisms affecting calcium current. NEURON provides a means of doing this – see the NMODL webpage.
- Let's continue without modelling calcium accumulation adjacent to the membrane, either intracellularly or extracellularly.

## NEURON and NMODL: compilation

- Assume a membrane mechanism description using NMODL is written in the text file (e.g., the text file **CaT.mod** containing our description of the T-type calcium channel in NMODL).
- The NEURON interpreter cannot read this file directly as it can with **hoc** files. Instead, the NMODL file must be **compiled** into a form that NEURON can use.
- How to compile & incorporate this new mechanism into NEURON depends on what operating system you are using (go to [http://web.mit.edu/neuron\\_v7.4/nrntuthtml/tutorial/tutD.html](http://web.mit.edu/neuron_v7.4/nrntuthtml/tutorial/tutD.html) )

---

## MS Windows

- You just launch
- > **Start/Programs/Neuron/mknrndll**
- This brings up a directory browser that can be used to navigate to your working directory that contains the **CaT.mod** file.
- When you get to the proper directory, click on the button labelled "**Make nrnmech.dll**". This compiles all the **.mod** files in this directory and creates a file called **nrnmech.dll** that contains the new compiled mechanisms.
- **nrnmech.dll** will be automatically loaded when you double click on a **.hoc** file in this directory.

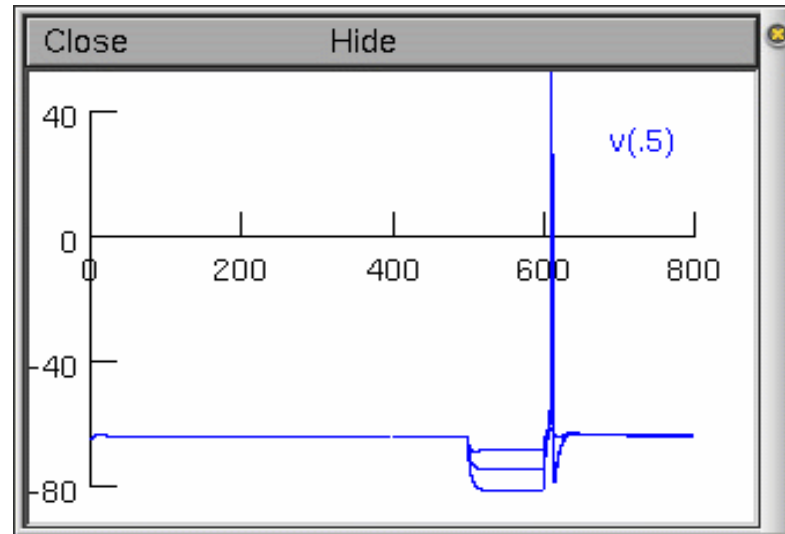
---

## Why the compilation?

- Why do we have to go through this cumbersome compilation procedure every time we want to create or modify a channel?
- The reason is that membrane mechanisms are used in every time step, and therefore need to be efficient.
- Converting the NMODL file to C-code and then compiling this into a new NEURON program or library (which is what **mknrndll** does) leads to more efficient simulation.
- If we do not modify our new channel mechanism, i.e., the **.mod** file, then the compilation is done only once.

## Conclusion

- Now if we launch **sthD.hoc** with a hyperpolarising current injected into any of our neurons, we now observe a **post-hyperpolarising T-type response**:

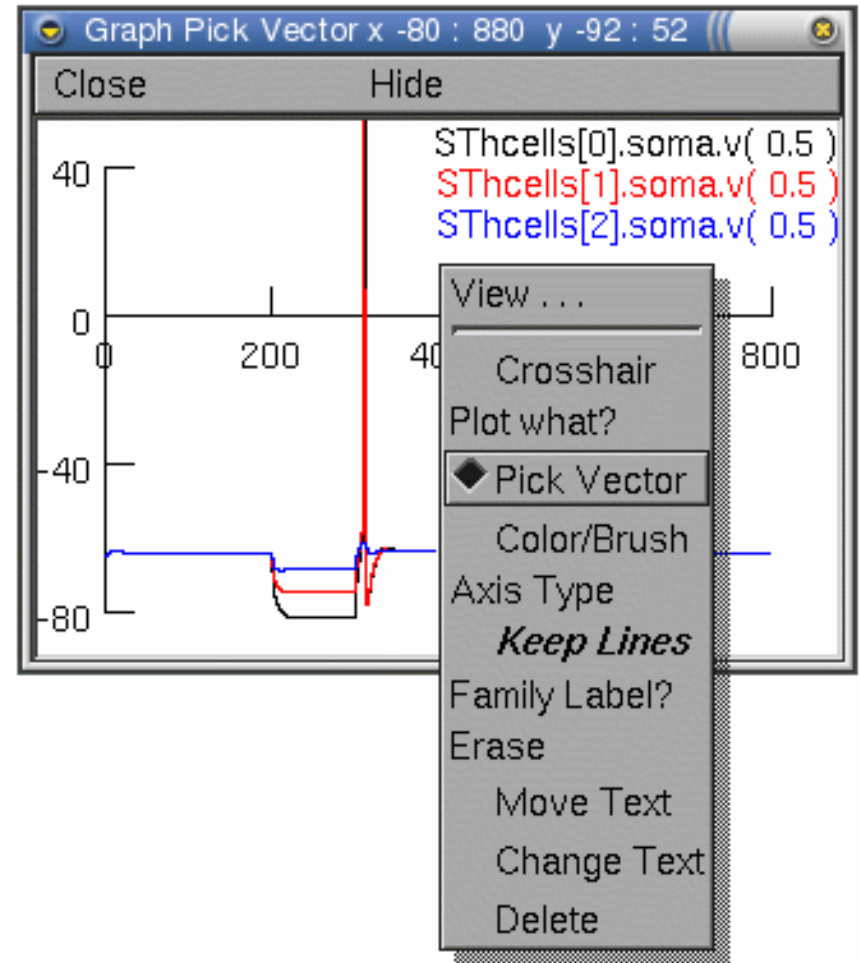


- With the current injections of -0.1, -0.2, and -0.3 nA, respectively.
- Note: The action potential is unrealistic for mammalian subthalamic cells b/c it is still based on HH squid axon channels.



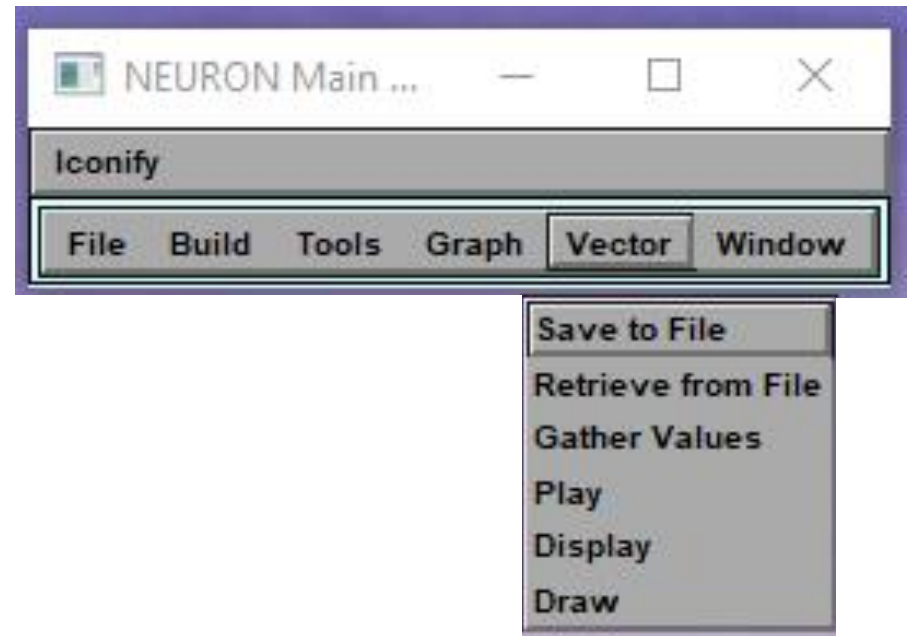
## Picking vectors from the plots

- Everything that is plotted can be saved to a specified data file as numbers for processing by other software.
- The *Graph Properties* menu appears after a right click on the graph of interest.
- From this menu, select the *Pick Vector* option. **We can only select one plot at a time to save!**



## Saving the vectors in a file **name . dat**

- The *Main Menu* toolbar contains a *Vector* menu allowing vectors to be saved or retrieved from data files.



- Select *Save File* from the *Vector* menu.

- This also pops up a file dialogue window allowing you to enter a filename for saving the selected vector data.

## Structure of the vector data file

- The beginning of the data file will look something like this:

```
label:SThcells[2].soma.v( 0.5 )
32001
0      -65
0.025 -65.0488
0.05   -65.0844
0.075  -65.1125
0.1    -65.136
```

- The first line of the file is a text string identifying the data being plotted, the second line is the number of data points in the file (here 32001), and finally the third line onward contains the data. First column is the time and, second column, soma voltage at that time.

---

## Speed of simulation

- To speed up a simulation we can reduce the number of segments **nseg** or increase **dt**. However, this will decrease the accuracy of results.
- Another strategy is a variable time step method. The principle is that the **dt** is longer when quantities are not changing much (such as between spikes) and shorter when quantities are changing quickly (such as during a spike).
- By default, NEURON uses fixed time step integration. The command  
**cvode\_active()**
- returns 0, indicating that variable time step is turned off. To turn on the variable time step integration we type: **cvode\_active(1)**

---

## Working with NEURON: practical hints (tut E)

- Quite often we want to analyze or record certain simulation results and perform large numbers of simulations as fast as possible (for example in parameter searching).
- Methods of getting data out of NEURON to be stored, visualized or analyzed by other software tools are the content of tutorial E.
- Tutorial E also introduces the ways of speeding up the simulation and the consequences and decisions we take in doing this.
- Much more can be found at: <https://neuron.yale.edu/neuron/docs> .