NEURON – tutorial A of Gillies & Sterratt

http://web.mit.edu/neuron_v7.4/nrntuthtml/index.html

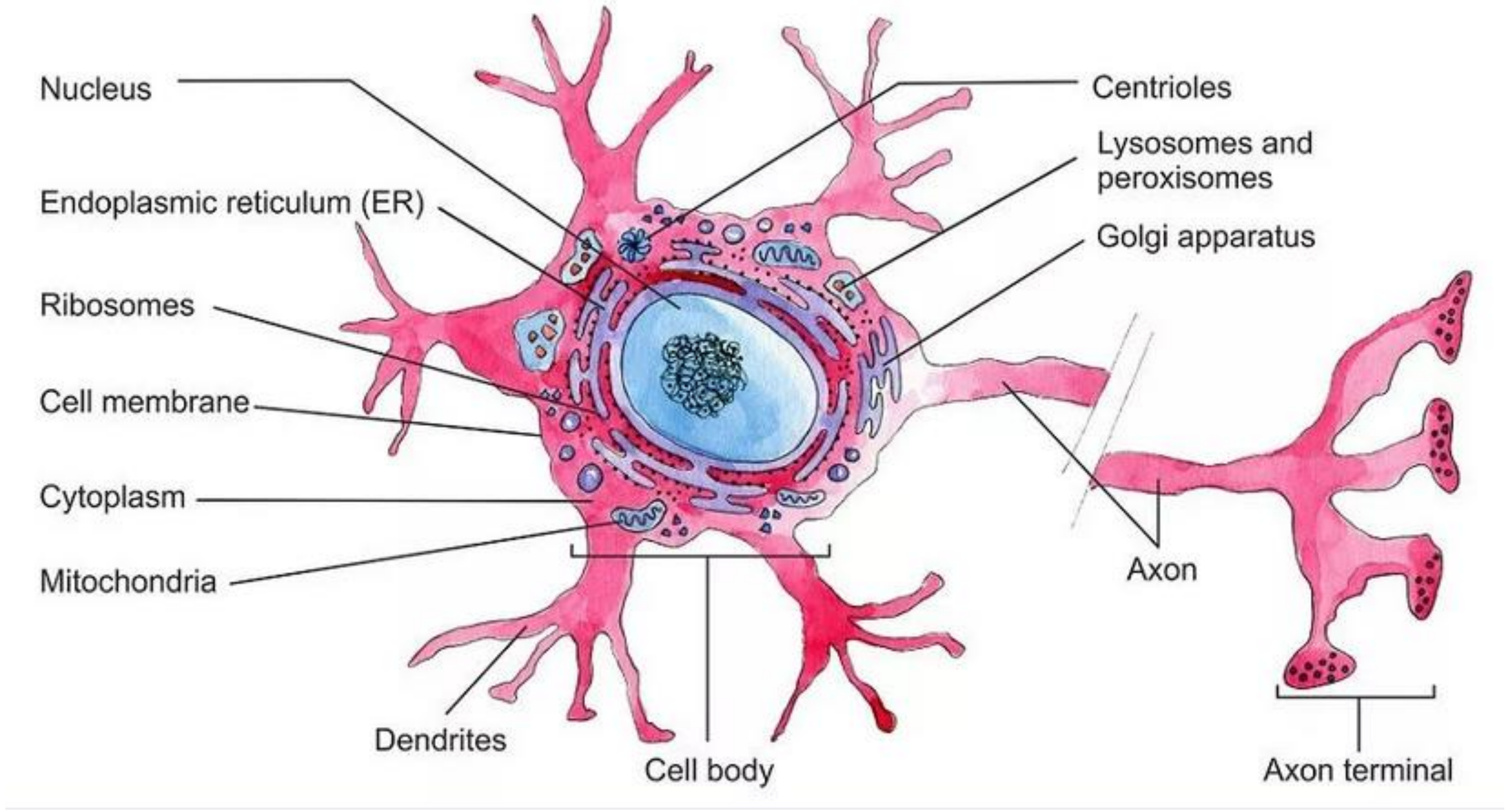
Lubica Benuskova

Lecture 2

Centre for Cognitive Science

Neurons

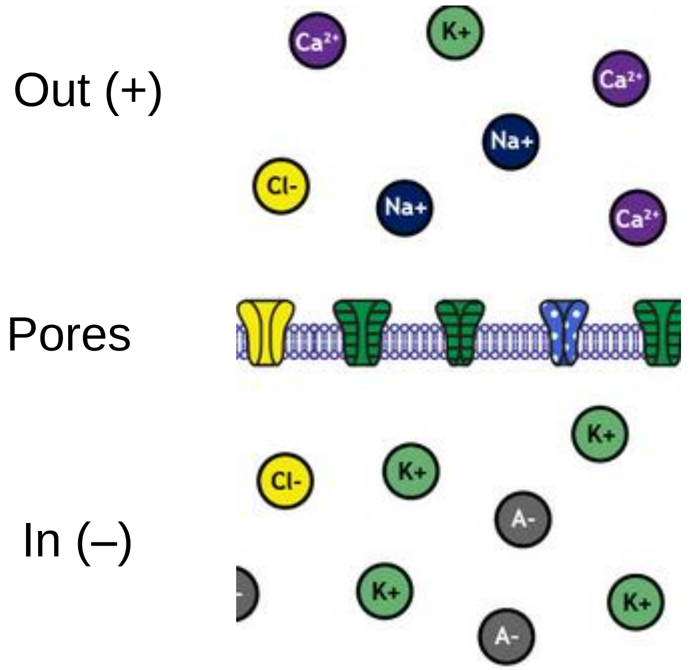
- The main information processing and communicating elements of the brain are called **neurons** (nerve cells).

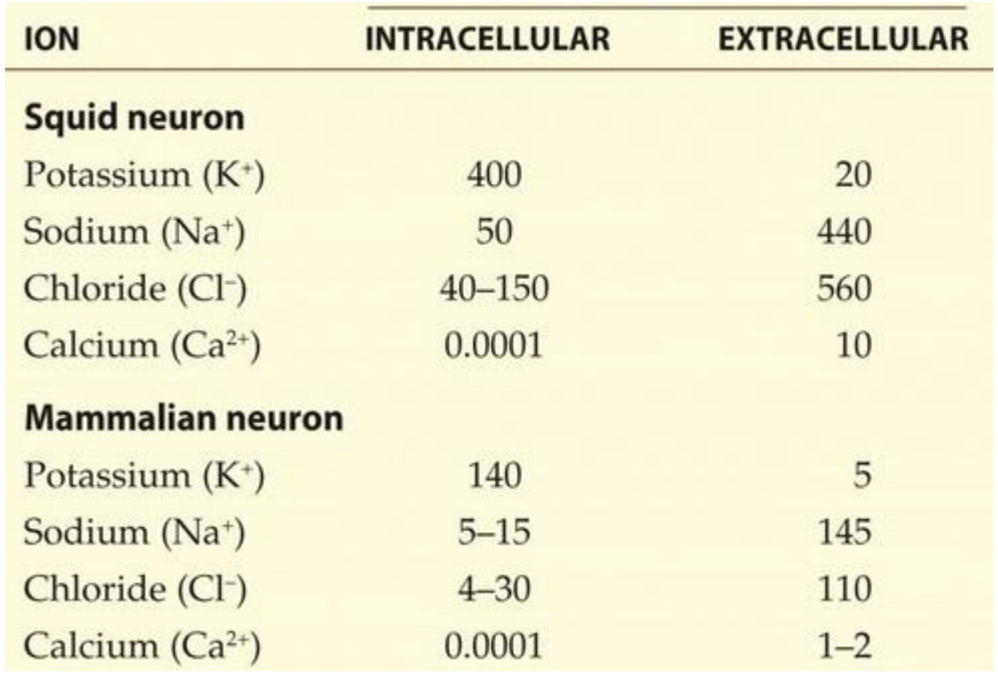


source: https://biologydictionary.net/neuron/

Outside and inside neurons are charged **ions**

- There are different concentration of several types of ions present in the liquid inside and outside of a neuron





ION	INTRACELLULAR	EXTRACELLULAR
Squid neuron		
Potassium (K^+)	400	20
Sodium (Na^+)	50	440
Chloride (Cl^-)	40–150	560
Calcium (Ca^{2+})	0.0001	10
Mammalian neuron		
Potassium (K^+)	140	5
Sodium (Na^+)	5–15	145
Chloride (Cl^-)	4–30	110
Calcium (Ca^{2+})	0.0001	1–2

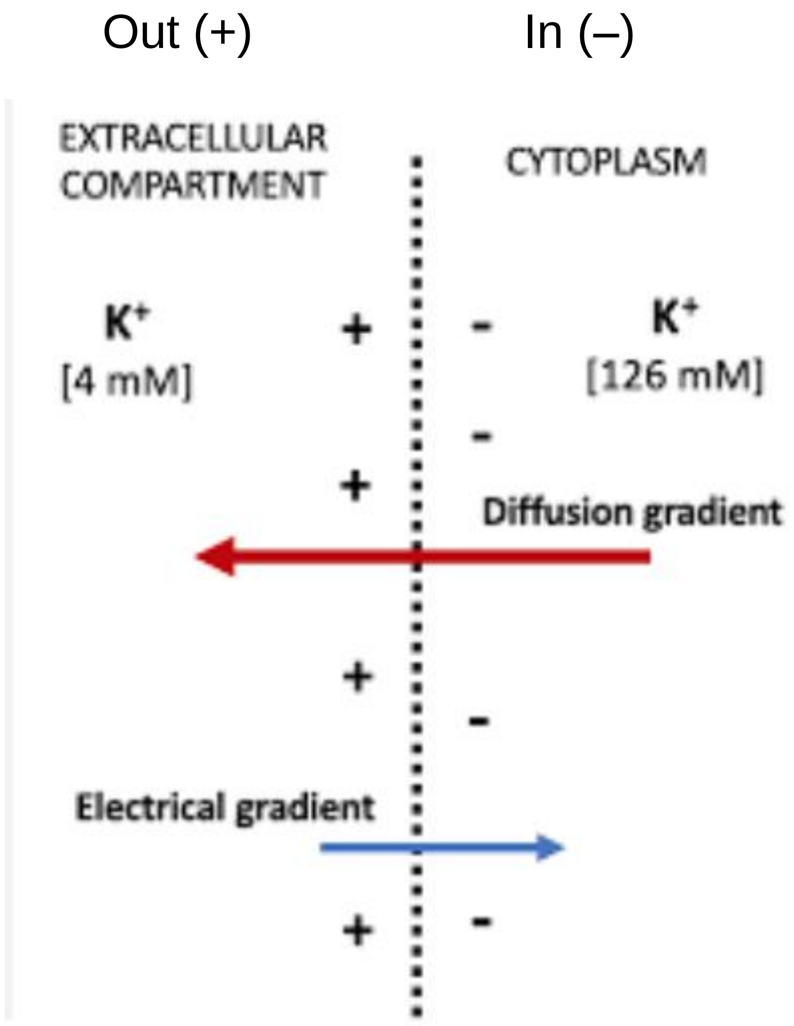
- As a result, there is an electrical voltage difference between the inside and the outside of neuron membrane:

$$V = V_{in} - V_{out} \approx -70 \text{ mV}$$

Ions are subject to two opposing forces

- Diffusion or concentration gradient: ions tend to flow from the side where their concentration is higher to the side with their lower concentration.

- Electrical gradient caused by electrical voltage difference between the inside and the outside of neuron membrane: positive ions are repulsed by the overall charge on that side.



Goldman-Hodgkin-Katz equation for resting potential

$$E_0 = \frac{RT}{F}\ln\left(\frac{\sum_i^n P_{M_i^+}[M_i^+]_{\mathrm{out}} + \sum_j^m P_{A_j^-}[A_j^-]_{\mathrm{in}}}{\sum_i^n P_{M_i^+}[M_i^+]_{\mathrm{in}} + \sum_j^m P_{A_j^-}[A_j^-]_{\mathrm{out}}}\right)$$

- E_0 = the resting membrane potential (in volts)
- M = positive ion species
- A = negative ion species
- P_{ion} = permeability for that ion (in meters per second)
- $[\mathrm{ion}]_{\mathrm{out}}$ = extracellular concentration of that ion (moles per m3)
- $[\mathrm{ion}]_{\mathrm{in}}$ = intracellular concentration of that ion (moles per m3)
- R = the ideal gas constant = 8.3 (joules per kelvin per mole)
- T = temperature in kelvins
- F = Faraday's constant = 9.65×10^4 (coulombs per mole)
- ln = the natural logarithm, the logarithm with base 10

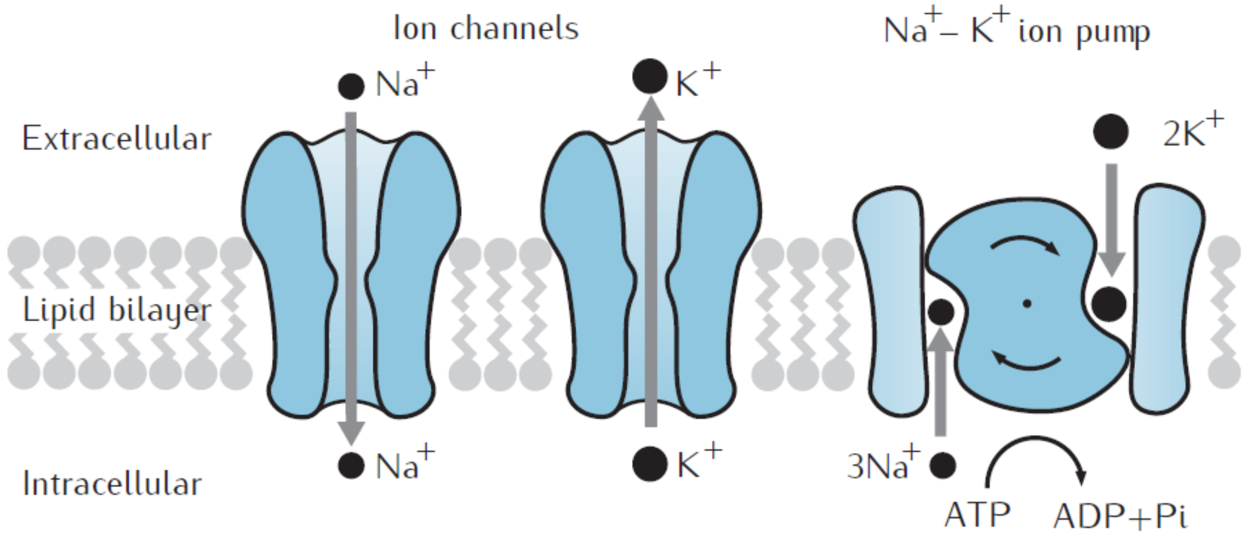
GHK equation for resting potential for Na, K and Cl

$$E_0 = \frac{RT}{F} \ln \left(\frac{P_{\mathrm{Na}}[\mathrm{Na}^+]_{\mathrm{out}} + P_{\mathrm{K}}[\mathrm{K}^+]_{\mathrm{out}} + P_{\mathrm{Cl}}[\mathrm{Cl}^-]_{\mathrm{in}}}{P_{\mathrm{Na}}[\mathrm{Na}^+]_{\mathrm{in}} + P_{\mathrm{K}}[\mathrm{K}^+]_{\mathrm{in}} + P_{\mathrm{Cl}}[\mathrm{Cl}^-]_{\mathrm{out}}} \right)$$

- The Nernst equation for a single ion is the so-called a reversal or equilibrium potential, meaning that at this value of the membrane potential, there is no flow of this type of ion through the membrane.

$$E_{0,\,\mathrm{Na}} = \frac{RT}{F} \ln \left(\frac{P_{\mathrm{Na}}[\mathrm{Na}^+]_{\mathrm{out}}}{P_{\mathrm{Na}}[\mathrm{Na}^+]_{\mathrm{in}}} \right) = \frac{RT}{F} \ln \left(\frac{[\mathrm{Na}^+]_{\mathrm{out}}}{[\mathrm{Na}^+]_{\mathrm{in}}} \right)$$

Neuronal membrane



- The membrane contains proteins that are:
  - Ion pumps: pump ions against their electric/concentration gradients
  - Ion channels: pass ions along their electric/concentration gradients
    - Always open
    - Voltage-gated: only open for certain value of membrane voltage
    - Ligand-gated (receptors): open when a ligand binds to them

Parts of a neuron

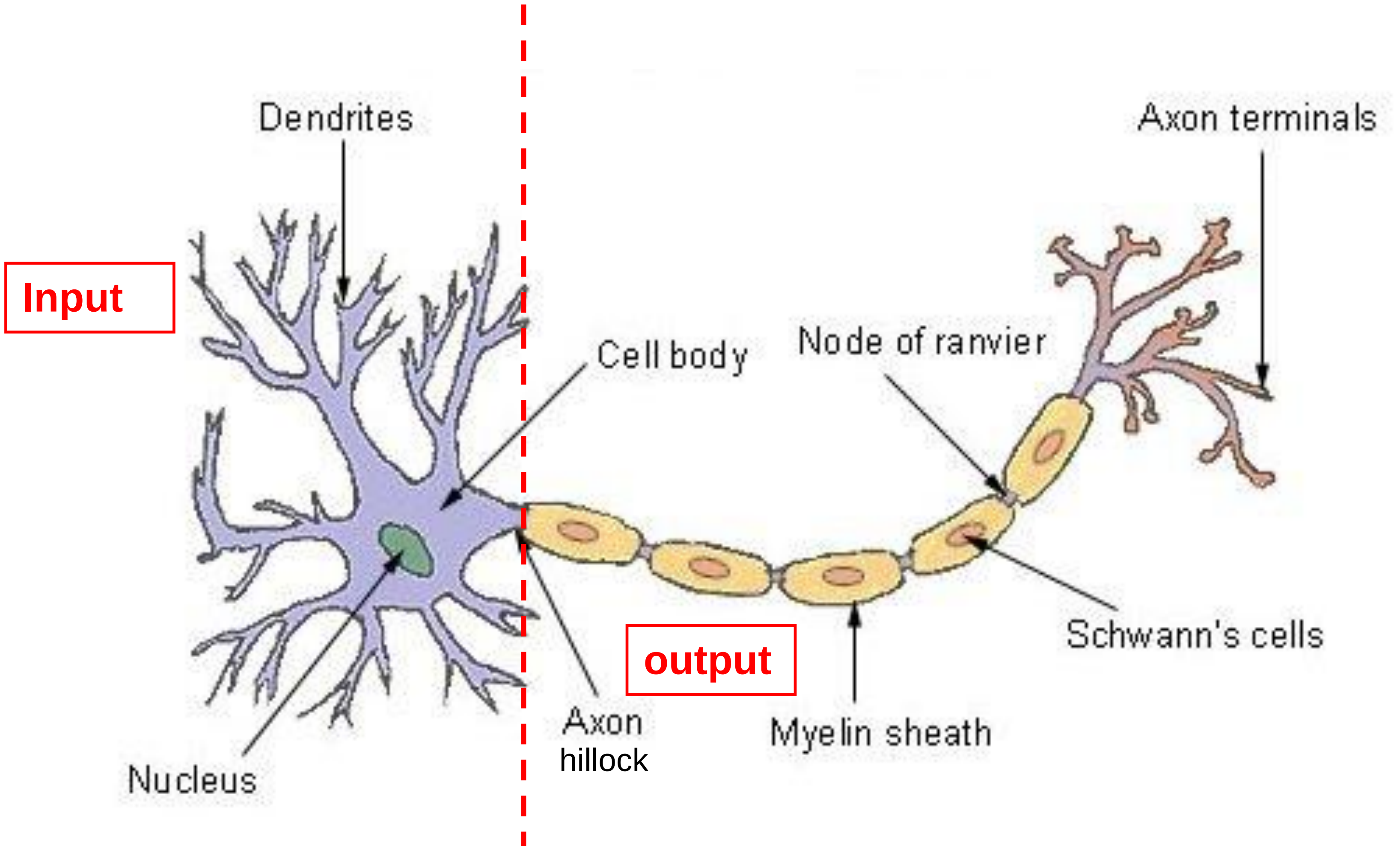
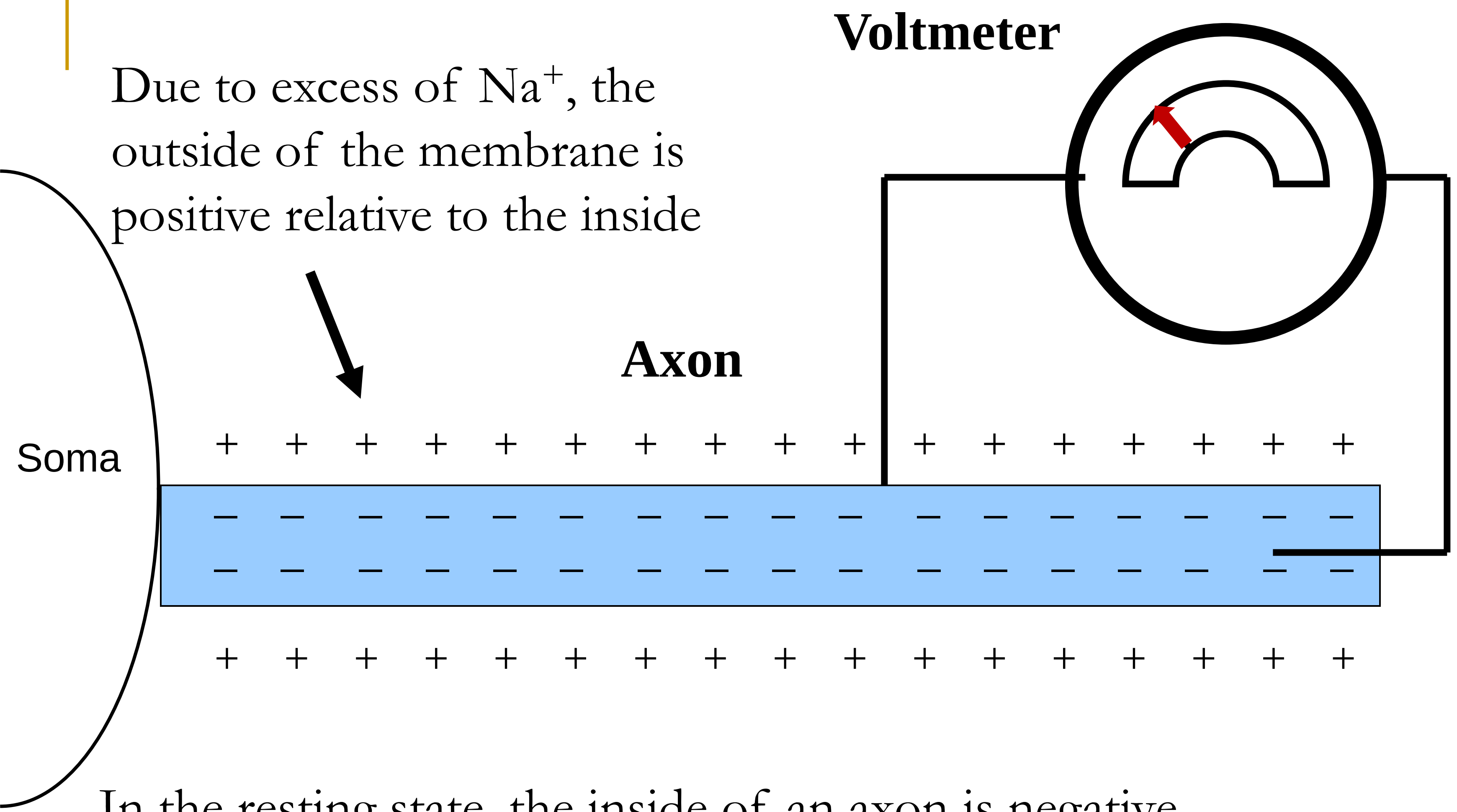


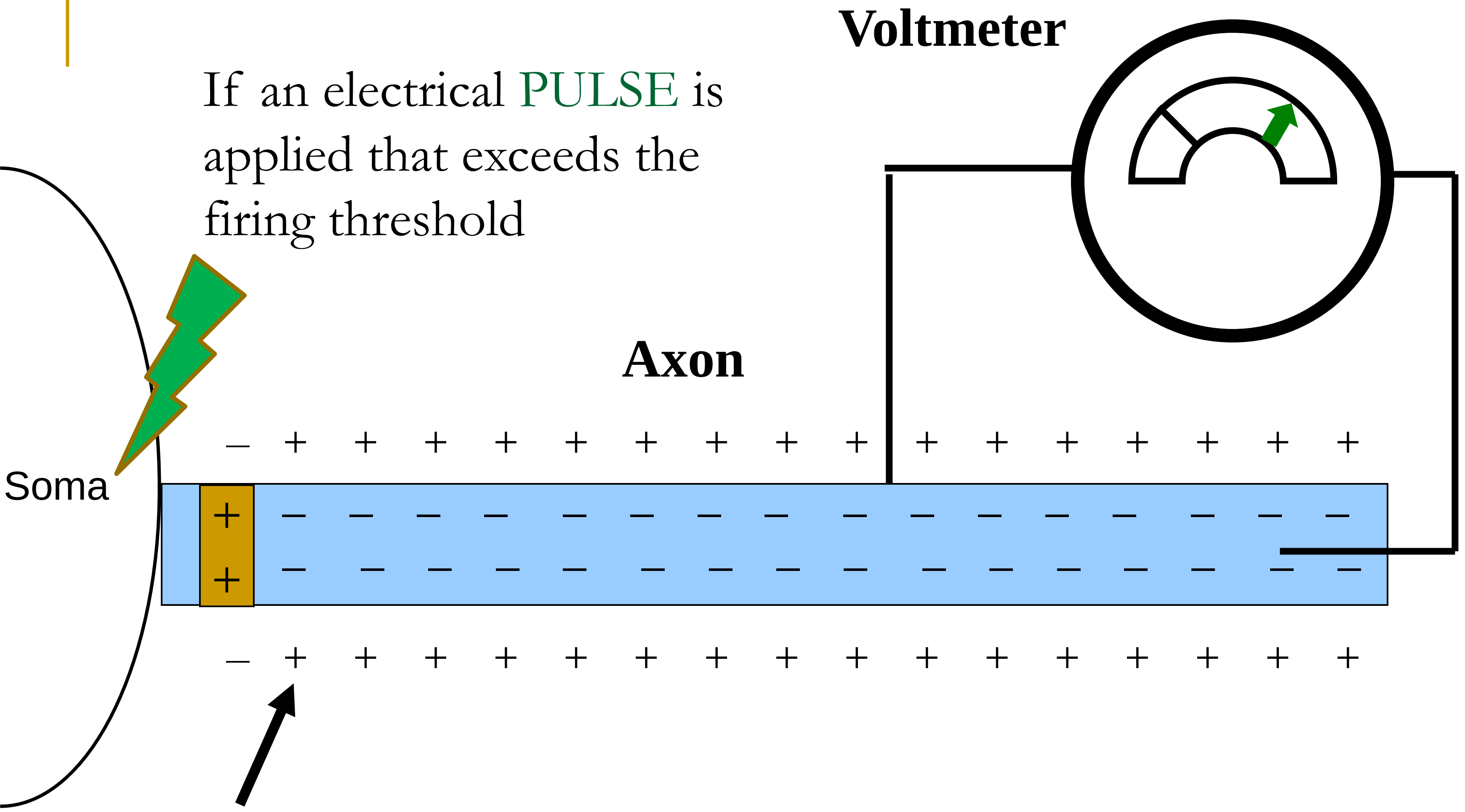
Image source: http://www.daviddarling.info/encyclopedia/N/neuron.html

Due to excess of Na^+, the outside of the membrane is positive relative to the inside

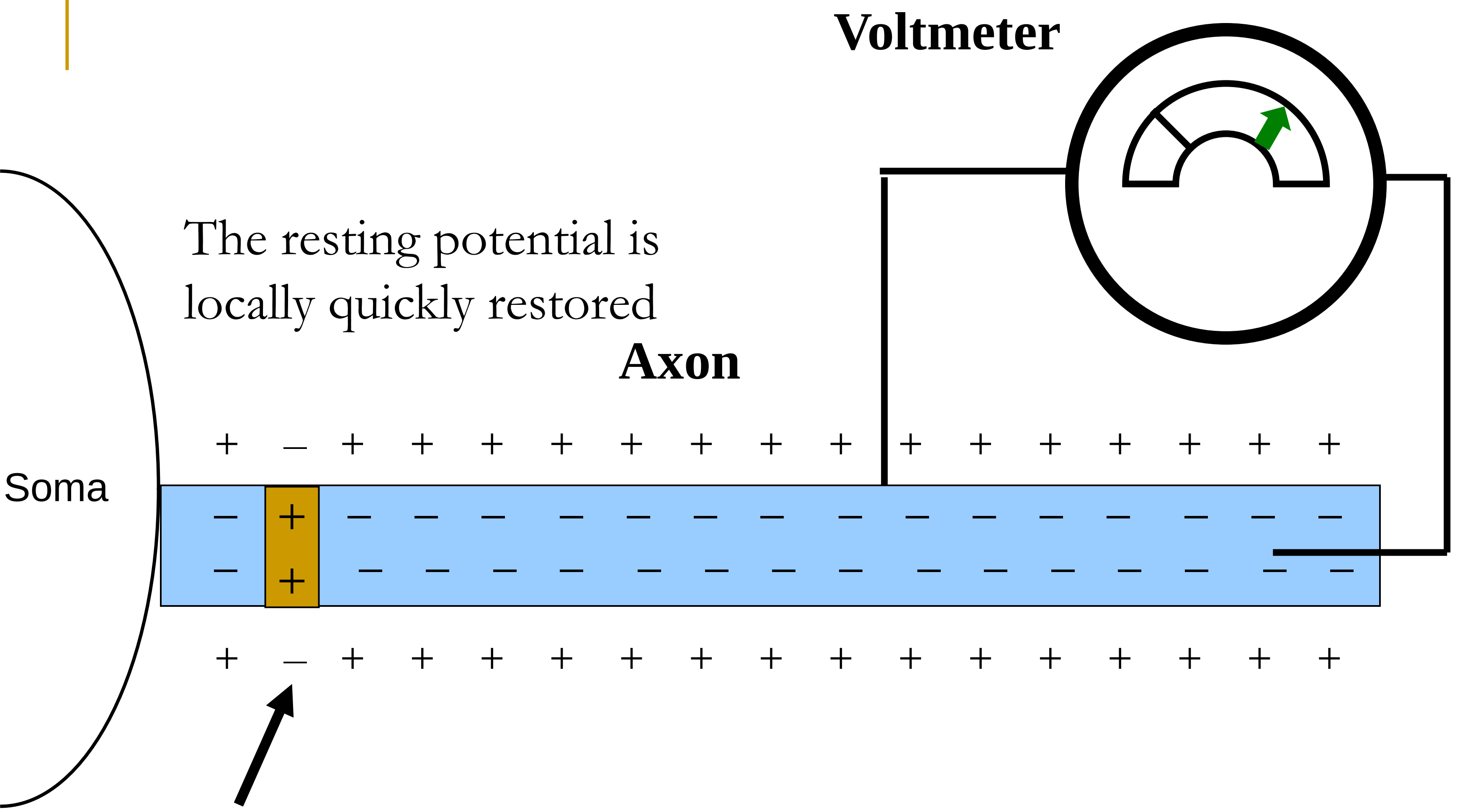


In the resting state, the inside of an axon is negative with respect to the outside, by about -70 millivolts (mV). This is called the RESTING POTENTIAL V_0

If an electrical PULSE is applied that exceeds the firing threshold

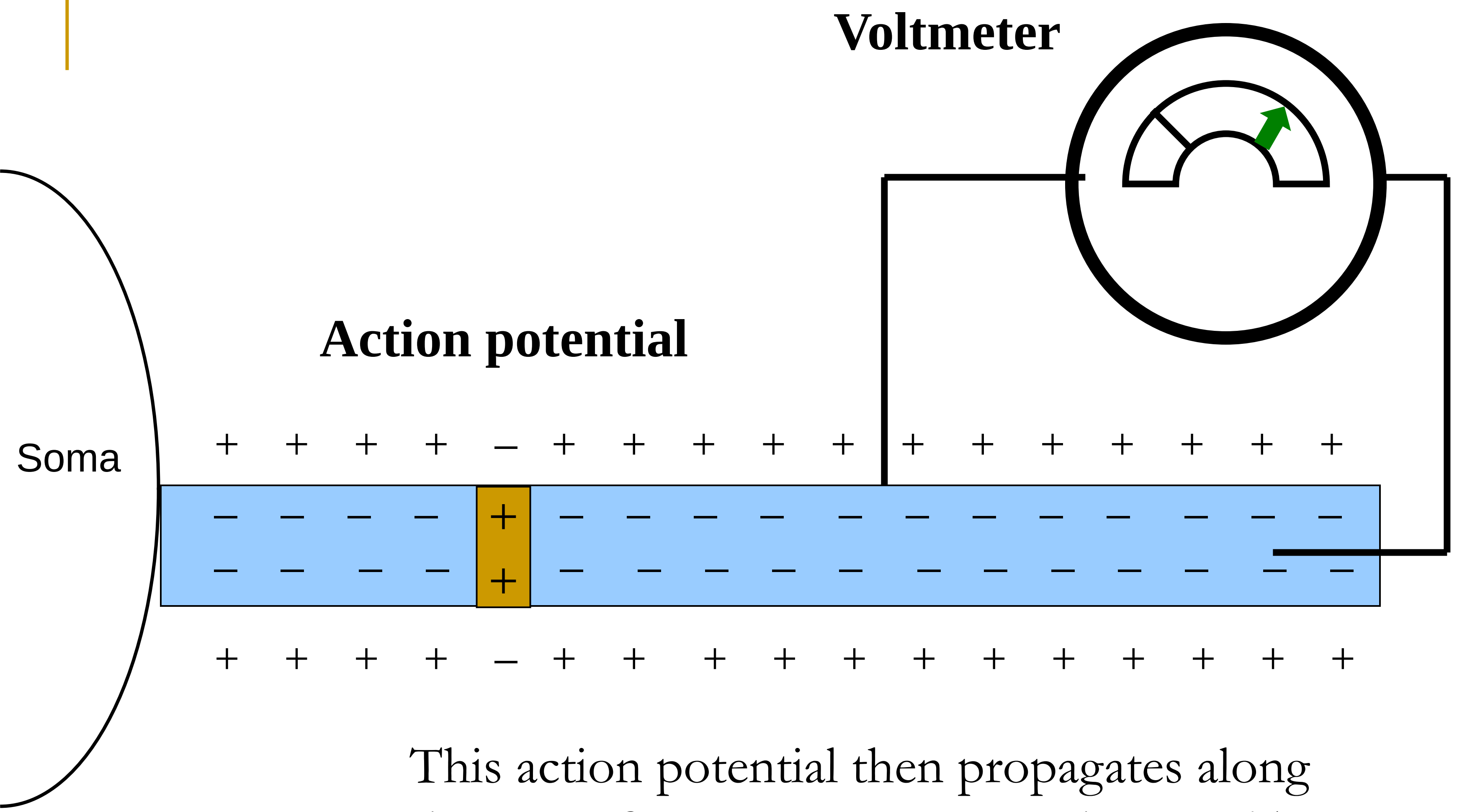


Na$^+$ ions go inside the axon, locally making the inside positive.

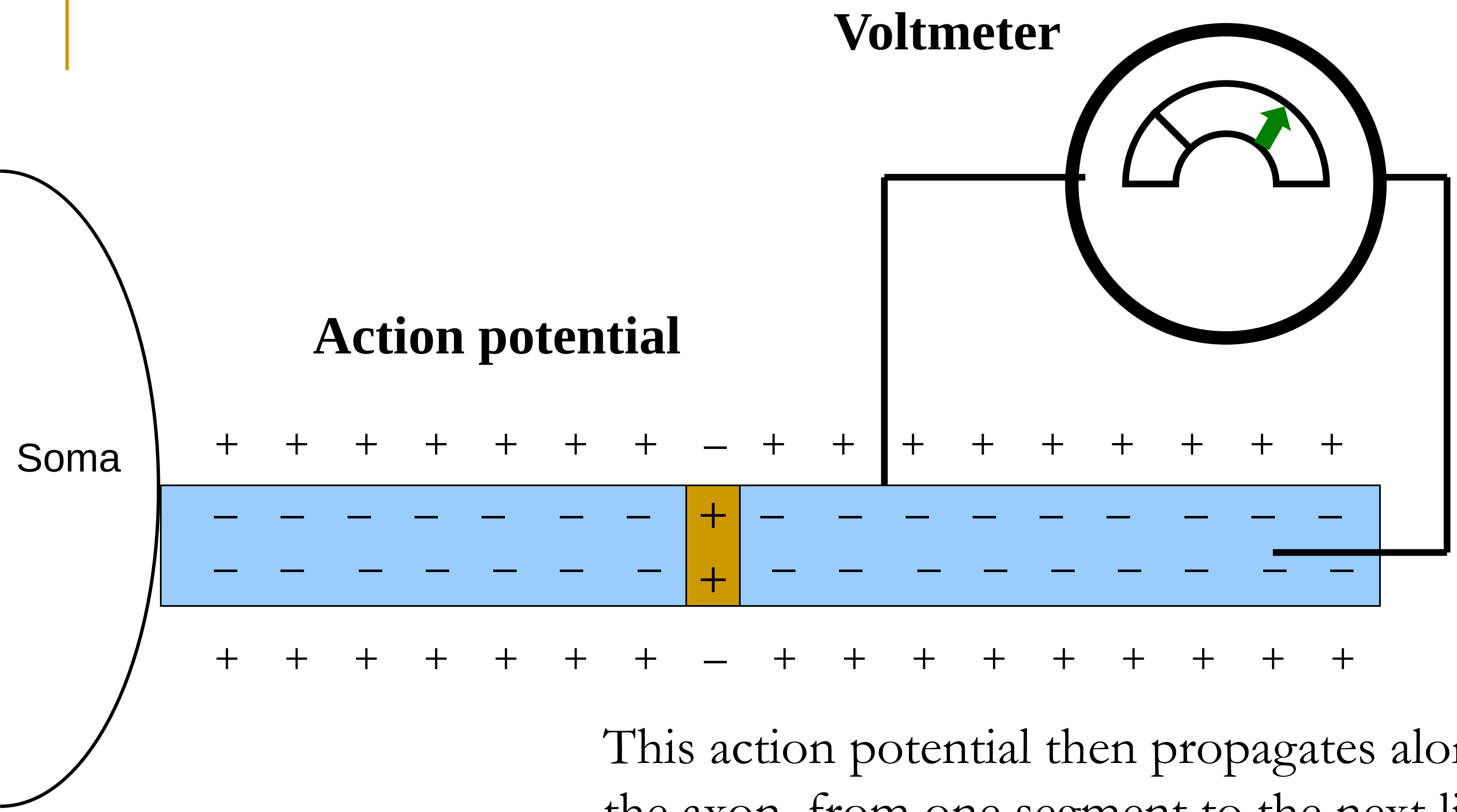


The resting potential is locally quickly restored

However, the local voltage inversion acts as a trigger for voltage inversion in the next segment of an axon.

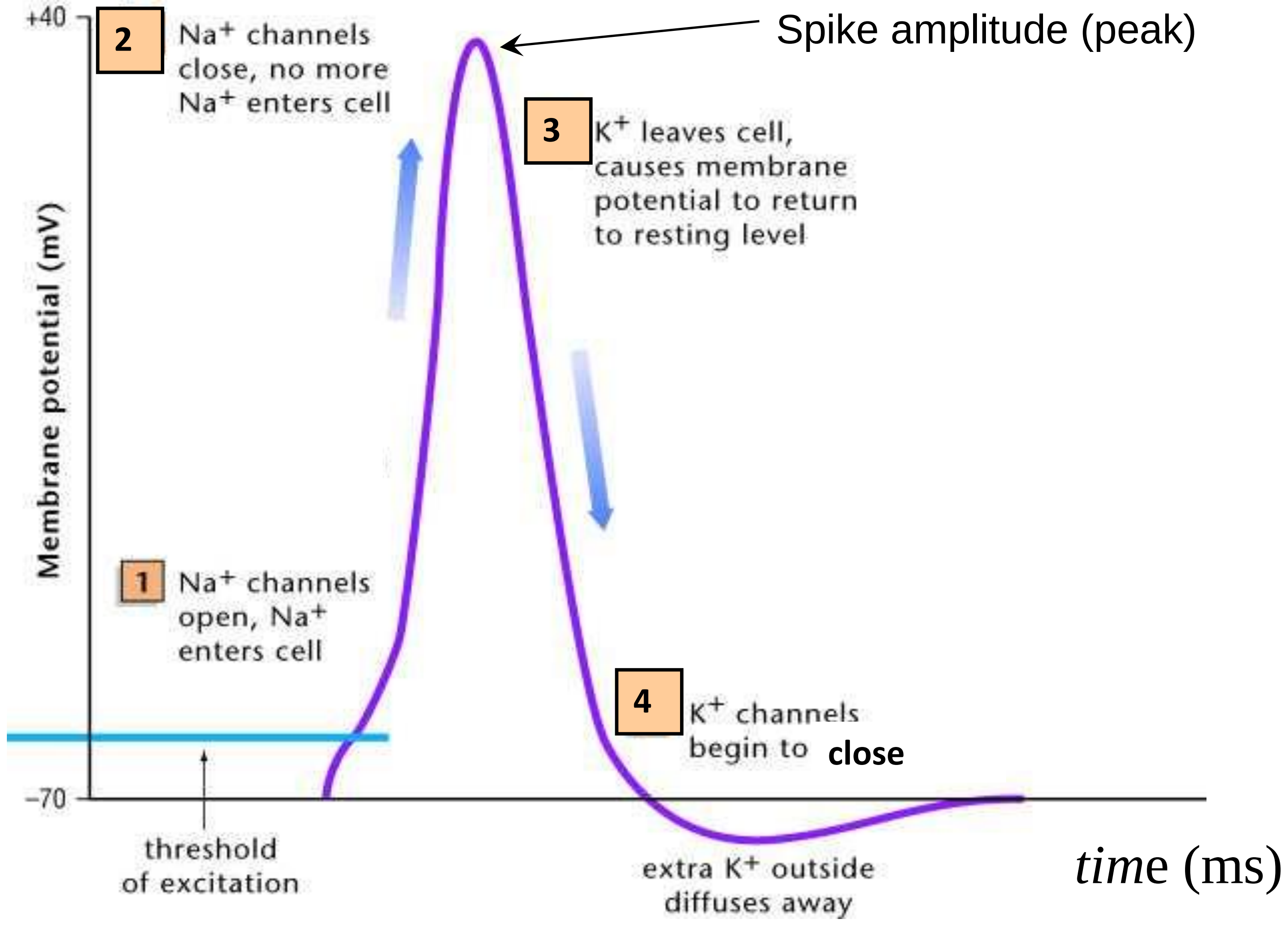


This action potential then propagates along the axon, from one segment to the next like a Mexican wave



This action potential then propagates along the axon, from one segment to the next like a Mexican wave, till it reaches axon terminals.

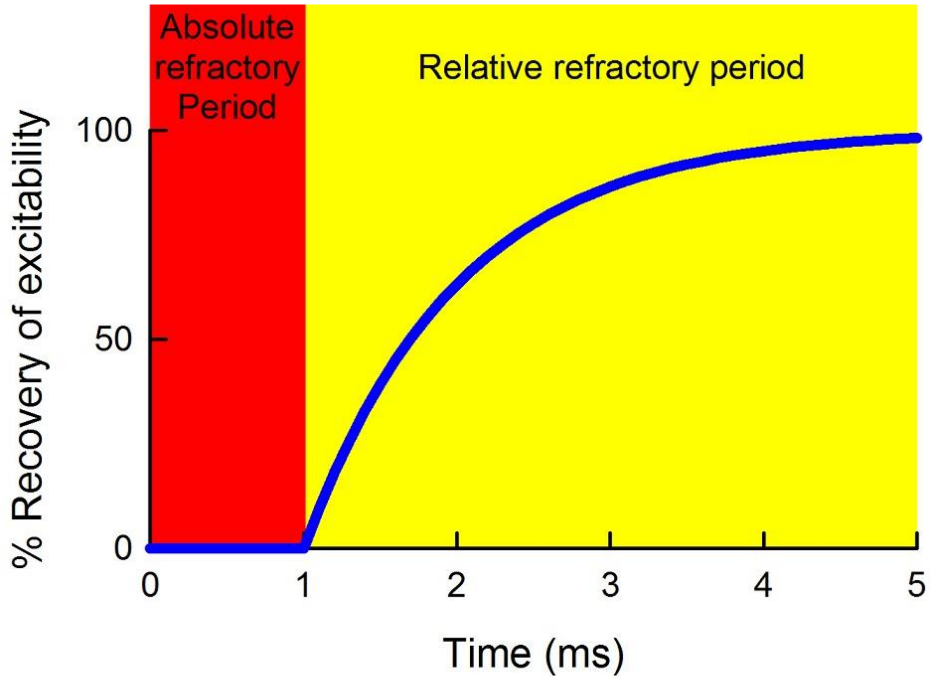
Change of the local voltage during an action potential


+40
2
Na⁺ channels close, no more Na⁺ enters cell
Spike amplitude (peak)
3
K⁺ leaves cell, causes membrane potential to return to resting level
Membrane potential (mV)
1
Na⁺ channels open, Na⁺ enters cell
4
K⁺ channels begin to close
–70
threshold of excitation
extra K⁺ outside diffuses away
time (ms)

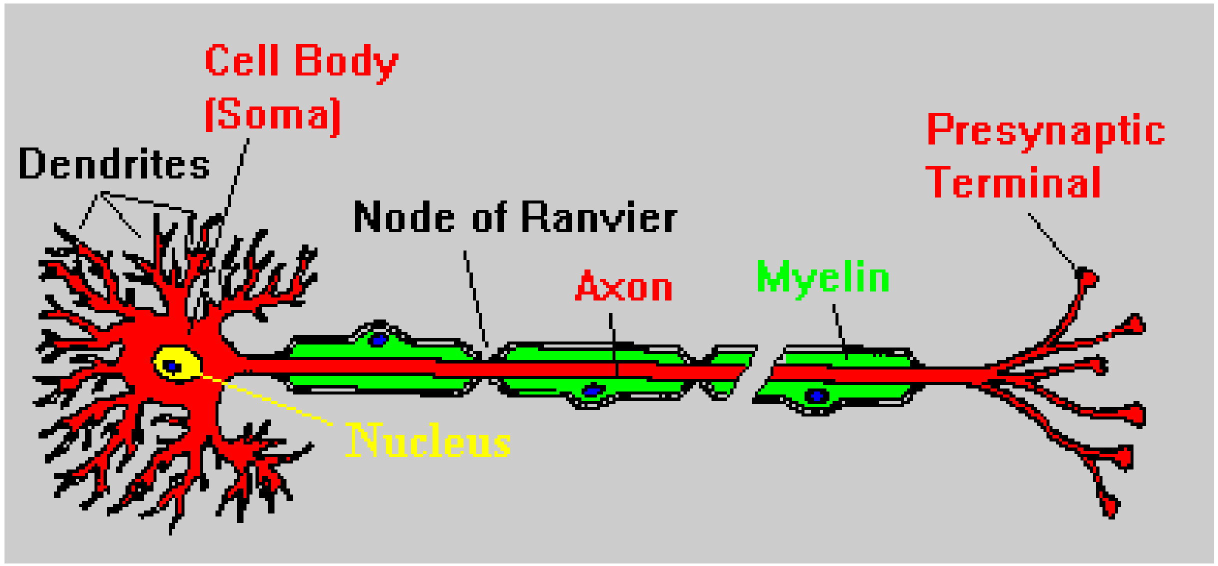
Refractory period

- The period from the initiation of an action potential to immediately after it is referred to as the absolute refractory period 1-2 ms. This is the time during which another stimulus given to the neuron (no matter how strong) will not evoke an action potential.

- The period immediately following the absolute refractory period is the relative refractory period 5-15 ms. The neuron can be excited if a stronger than normal stimulus is applied.



Spikes “jump” along axons from one node to another



- Axons have a myelin sheath surrounding the axons, that makes up the “white matter” of the brain (“grey matter” are somas and dendrites).
- This speeds transmission, because the spike jumps between the gaps (nodes of Ranvier) and the sheath provides electrical insulation.
- In each gap, the original hight of spike amplitude is restored.

Hodgkin and Huxley

- In 1963, Brits Allan Hodgkin & Andrew Huxley received the Nobel prize in Physiology and Medicine for their work on axon potentials. (The 3rd laureate was Sir John Eccles for work on synapses.)

- H&H developed an action potential theory and model using one of the earliest applications of a technique of electrophysiology, known as the "voltage clamp", which enabled to measure ionic currents flowing through the membrane.

- This is historically the first computational model of biological neurons and applies to all axons (albeit with different values of parameters).

Hodgkin-Huxley model

- The membrane current I_{m} flowing through the ion channels:

$$I_m = g_{Na} m^3 h(V - E_{Na}) + g_K n^4 (V - E_K) + g_L (V - E_L)$$

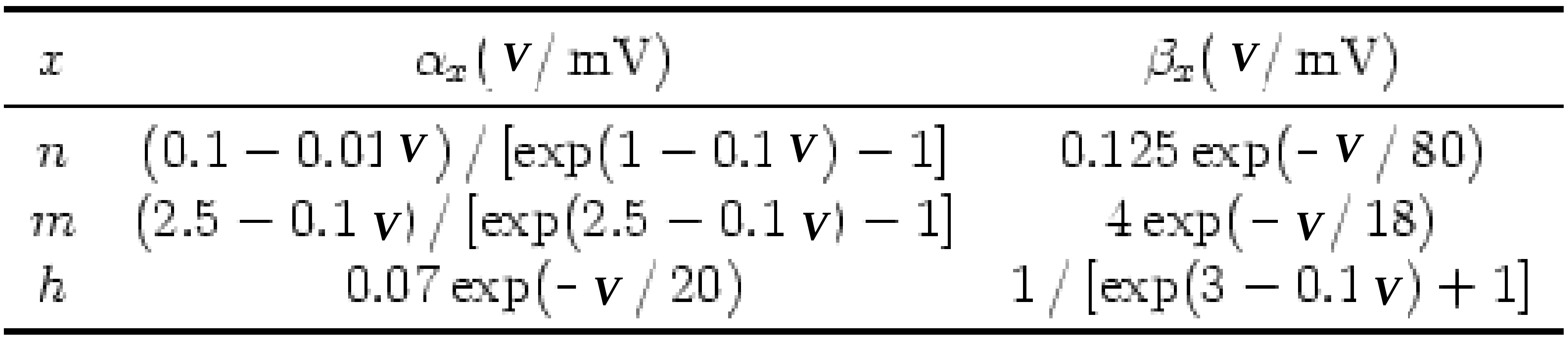
- Here: *g* is the electric conductance, *Na* = sodium, *K* = potassium, *L* is all the other ions (the so-called leakage current). *E* denotes the equilibrium potential for that ion, *V* is the membrane voltage.

- Symbols *m, n, h* denote empirically derived parametric functions of the model that correspond to kinetics of individual subunits/gates of Na and K ion channels.

H-H model: parametric functions

- The three variables *m*, *n*, and *h* are called gating variables. They evolve according to the differential equations (*V* is voltage):

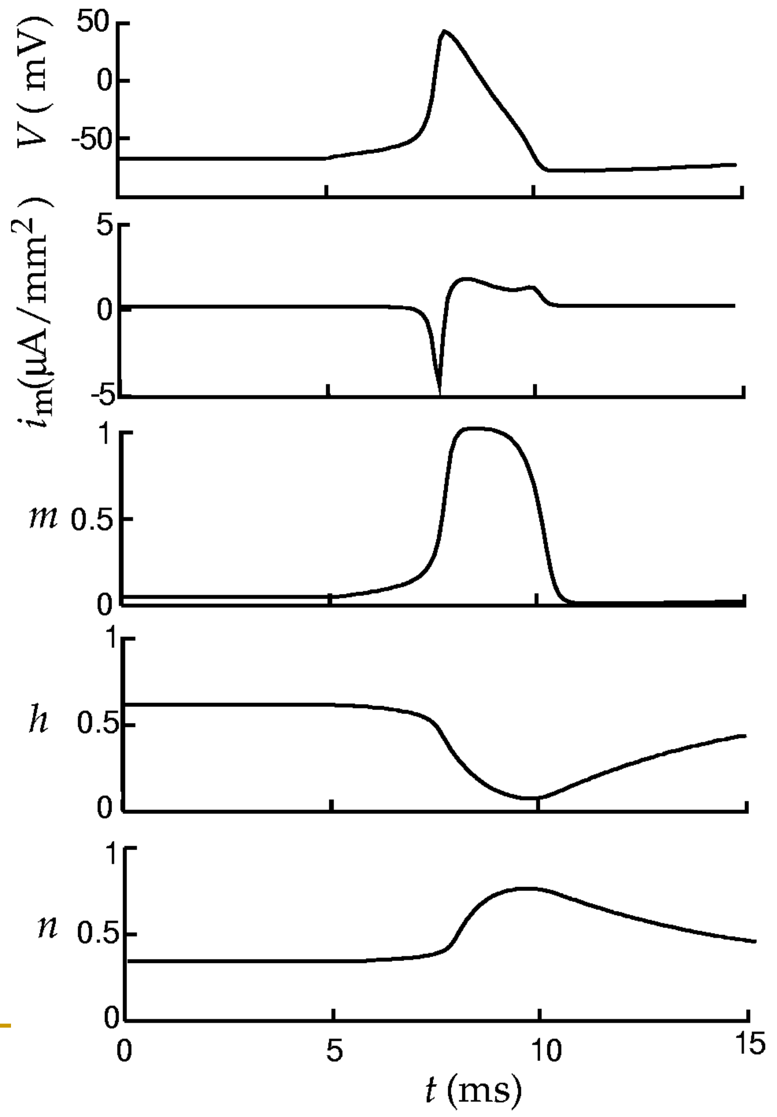
$$\dot{m} = \alpha_m(V)(1-m) - \beta_m(V)m$$
$$\dot{n} = \alpha_n(V)(1-n) - \beta_n(V)n$$
$$\dot{h} = \alpha_h(V)(1-h) - \beta_h(V)h$$



x	$\alpha_x(V/\mathrm{mV})$	$\beta_x(V/\mathrm{mV})$
n	$(0.1 - 0.01\,V) / [\exp(1 - 0.1\,V) - 1]$	$0.125 \exp(-V/80)$
m	$(2.5 - 0.1\,V) / [\exp(2.5 - 0.1\,V) - 1]$	$4 \exp(-V/18)$
h	$0.07 \exp(-V/20)$	$1 / [\exp(3 - 0.1\,V) + 1]$

H-H model: dynamics of variables

- Resulting time course of the membrane voltage V during spike:
- Underlying time course of the membrane current I_m :
- Time course of the variable m:
- Time course of the variable h:
- Time course of the variable n:



Hodgkin-Huxley model (continued)

- There is also a current related to the membrane capacitance C_m :

$$I_C = C_m \frac{dV}{dt}$$

- Total membrane current is the sum: $I = I_m + I_C$

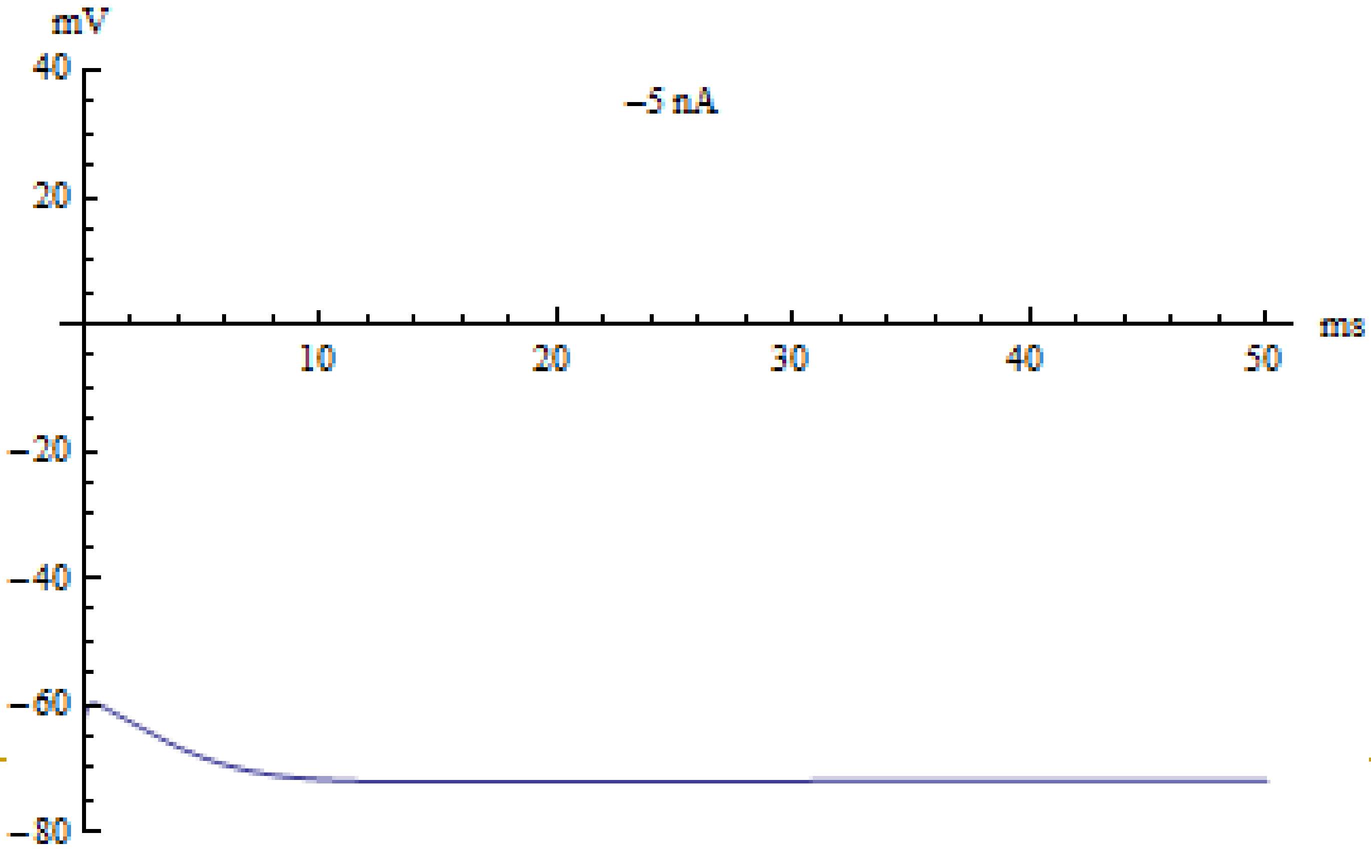
- From the so-called cable theory, we know that $I = \frac{a}{2R}\frac{\partial^2 V}{\partial x^2}$

- where a is the radius of the axon, R is the specific resistance of the axoplasm, and x is the position along the axon. Substitution of this expression for I transforms the original set of equations into a partial differential equation, because the voltage V becomes a function of both x and t. Thus, the final equation for membrane voltage reads

$$(a/2R)\frac{\partial^2 V}{\partial x^2} = C_m \frac{\partial V}{\partial t} + g_{Na}\, m^3 h(V - E_{Na}) + g_K n^4 (V - E_K) + g_L (V - E_L)$$

Animation of an action potential (spike)

- The voltage V(t) (in mV) of the Hodgkin–Huxley model, graphed over 50 milliseconds. The injected current varies from −5 nA to 12 nA. The graph passes through three stages: an equilibrium stage, a single-spike stage, and a limit cycle stage (Wikipedia).



Model of soma and Hodgkin–Huxley model

- Today we start with the basics: how to create a single soma with Hodgkin-Huxley conductances and how to run the simulator and display the simulation result.

- This is contained in tutorial A of Gillies & Sterratt: http://web.mit.edu/neuron_v7.4/nrntuthtml/tutorial/tutA.html

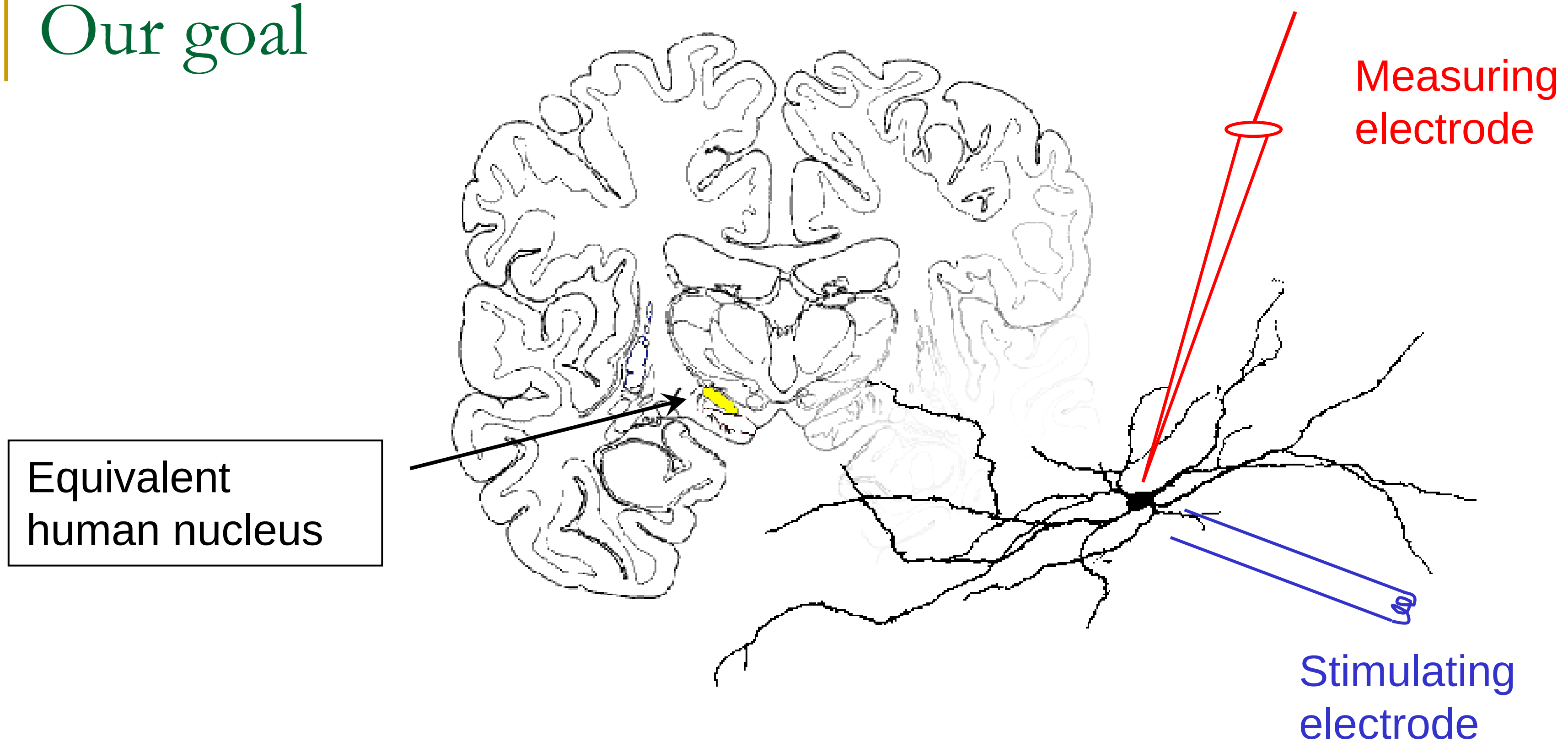
- Next time, this will be extended into a multi-compartmental neuron with dendrites (Tutorial B of Gillies & Sterratt).

Tutorial by Andrew Gillies and David Sterratt

- **Tutorial** is divided into **5 parts (A - E)** and will take you, step by step, through the process of creating complex simulations: http://web.mit.edu/neuron_v7.4/nrntuthtml/index.html

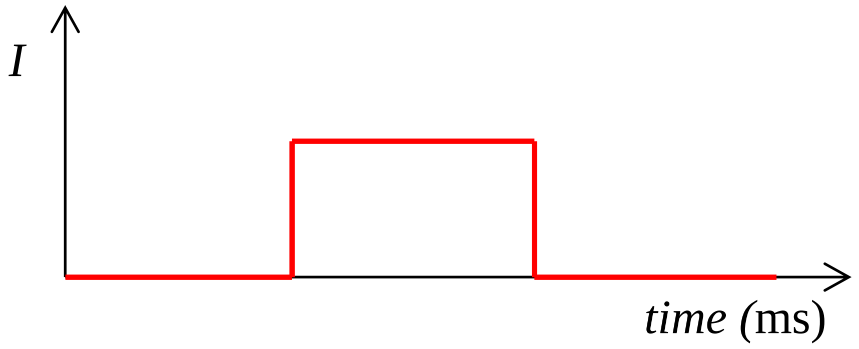
- **Part A**: creating model of soma with Hodgkin-Huxley model.
- **Part B**: building a multi-compartmental neuron with dendrites and using different types of graphs.
- **Part C:** replicating neurons using templates and connecting them into a network.
- **Part D**: adding new membrane mechanisms to the model.
- **Part E:** ways of saving and working with the simulation data.

Our goal



- It is always good to have a final product in mind when we start the modelling task, so here is ours:
- We want to model and study neurons in the rat subthalamic nucleus.

Soma

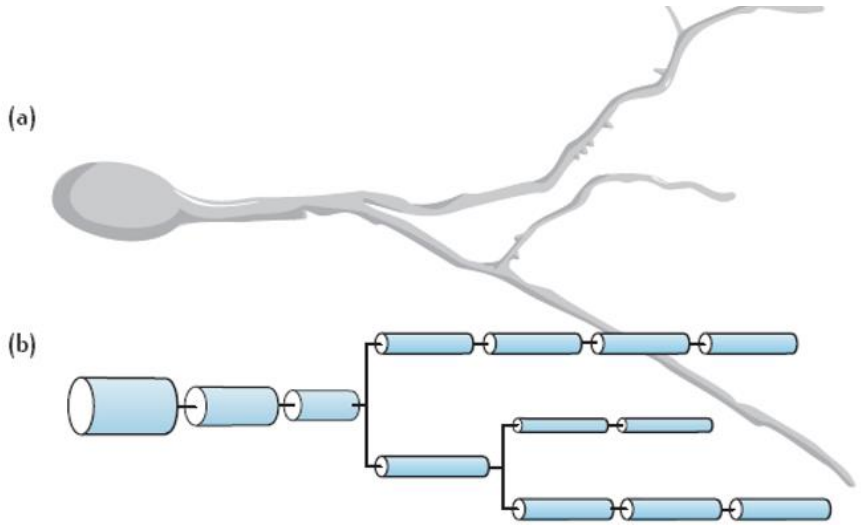


- This piece of code encodes a soma, which is stimulated by a rectangular pulse of electric current and generates spikes.
- Write the code into the text editor (e.g., Notepad) and save the file as **`sthA.hoc`** in your working directory.

```
load_file("nrngui.hoc")

create soma
access soma

soma nseg = 1
soma diam = 18.8
soma L = 18.8
soma Ra = 123.0

soma insert hh

objectvar stim

soma stim = new
IClamp(0.5)

stim.del = 100
stim.dur = 100
stim.amp = 0.1

tstop = 300
```

Morphology

- In NEURON, the neuron's geometry is described in terms of cylindrical sections.

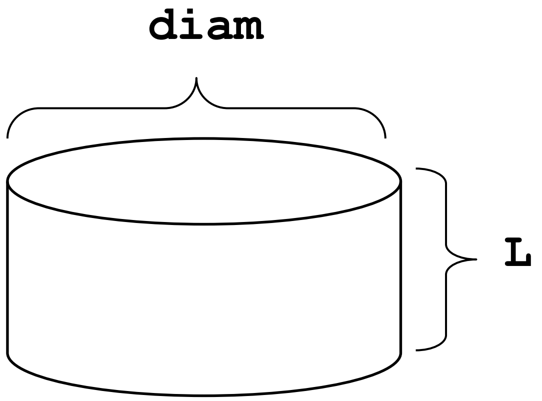


- The simplest reduction is to remove the neuron's axon and dendrites leaving only one section - the cell body, soma.

- We create a soma with the command: `create soma`

Command: `create soma`

- Creates one section of a neuron, i.e. soma, with the default values of these parameters:
  - number of segments [**`nseg = 10`**]
  - the diameter [**`diam = 500 µm`**]
  - the length [**`L = 100 µm`**]
  - the specific membrane capacitance [**C = 1 µF/cm^2**]
  - and the cytoplasmic (axial) resistivity [**`Ra = 35.4 ohm cm`**]).



- The values of these parameters must be taken from experimental data of our modelled neuron. Let's explain each parameter now.

The parameter `nseg`

- The parameter **`nseg`** specifies the number of internal points at which NEURON computes solutions (i.e. the time course of membrane potential, ionic currents, etc.).

- A section can be broken into **`nseg`** segments of equal length (`=L/nseg`), then NEURON will compute the time course of these variables at the centre of each segment.

- For soma: If we assume the density of transmembrane current is reasonably uniform over the surface of the soma, a single point will be sufficient, and we can assign a value of 1 to **`nseg`**.

Changing parameter values

- Since NEURON deals with many different sections each with their own unique name, we must tell NEURON *which section we are working on* when we want to refer to the value of a section parameter. There are three ways to do this in NEURON:

- The dot notation. Here, we refer to the section parameters with the section name followed by a "dot" or period (".") followed by the parameter name and assignment of its value:

```
soma.nseg = 1
soma.diam = 18.8
soma.L = 18.8
soma.Ra = 123.0
```

Other notation for parameter values

- “Space” notation:

```
soma nseg = 1
soma diam = 18.8
soma L = 18.8
soma Ra = 123.0
```

- “Brace” notation, we can group multiple properties within braces:

```
soma {
        nseg = 1
        diam = 18.8
        L = 18.8
        Ra = 123.0
}
```

`Access` statement: default section

- An `access` statement declares a default section:

```
access soma
```

- We need the **`access`** statement because NEURON's graphical tools don't work unless a default section has been declared.
- We can specify any section we like as the default section, but in general it should be a section that is of special interest.
- We shouldn't have more than one **`access`** statement.

Our program code so far

- Thus, the program so far is:

```
create soma
access soma

soma nseg = 1
soma diam = 18.8
soma L = 18.8
soma Ra = 123.0
```

- Notice that the **`access`** statement is straight after the **`create`** statement.

Inserting membrane properties

- Each section in NEURON has the default properties (like `L`, `diam`, `C` and `Ra`) automatically inserted, but other mechanisms (e.g., channels) with their own parameters must be explicitly inserted into each section.

- NEURON includes two built-in channel membrane mechanisms: Hodgkin-Huxley channels (**`hh`**) and passive channels (**`pas`**). Each of these mechanisms can be inserted using the **`insert`** command.

- You can insert a buil-in ion channel mechanism to soma like this:

```
soma insert hh
```

Inserting membrane properties

- When you add a new membrane mechanism (like **`hh`** or **`pas`**) to a section, you add new parameters and their default values to the section as well.
  - For example, if you add passive channels to the section, you will introduce two new properties to the section: **`g_pas`** (passive membrane conductance [S/cm^2]) and **`e_pas`** (reversal potential [mV] for each ion).

- For our simulation, neither the default Hodgkin-Huxley channels nor the passive properties will accurately model our neuron type. However, for the moment we will insert these properties to explore a working system. (It's possible to develop our own channel properties – we'll see how later).

Parameters of Hodgkin-Huxley channels

- The Hodgkin-Huxley channels add the following new parameters to the section:
    - **`gnabar_hh`**: The maximum specific sodium channel conductance [Default value = 0.120 S/cm^2]
    - **`gkbar_hh`**: The maximum specific potassium channel conductance [Default value = 0.036 S/cm^2]
    - **`gl_hh`**: The maximum specific leakage conductance [Default = 0.0003 S/cm^2]
    - **`ena`**: The reversal potential for the sodium channel [Default value = 50 mV]
    - **`ek`**: The reversal potential for the potassium channel [Default value = –77 mV]
    - **`el_hh`**: The reversal potential for the leakage channel [Default = –54.3 mV]

State variables in Hodgkin-Huxley model

- The command **`insert hh`** also adds to the model the following *state variables*:

- **`m_hh`**: The sodium activation state variable
- **`h_hh`**: The sodium inactivation state variable
- **`n_hh`**: The potassium activation state variable
- **`ina`**: The sodium current
- **`ik`**: The potassium current

Adding stimulation – point processes

- NEURON makes the distinction between mechanisms that are attributed to the entire section (like HH channels) and mechanisms that are associated with a particular **point** in the section (e.g., voltage, synapse or stimulation).

- While the mechanisms are expressed in terms of per unit area, the point processes are more conveniently expressed in absolute terms (e.g., current injection is usually expressed in terms of nA instead of nA/cm^2).

- Point processes also differ in that you can insert several different processes within the same section, but you have to specify the location of that process within a section.

Point processes as objects

- In NEURON, point processes are handled as objects. To create an object, you need to first create an **object variable** to be associated with the object. To declare an object variable, you enter the following:

```
objectvar stim
```

- This creates an object variable named **`stim`**, and now you need to create the actual object with a section to which it applies:

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

- **`New`** means it's a new instance of a particular object (e.g., Iclamp).
- The location is specified with a number between 0 and 1 (inclusive) where the number represents the fractional length along the section.

Built-in point processes

- NEURON has inbuilt: **`IClamp`**, **`VClamp`** and **`ExpSyn`**.
- Additional point processes can be built into the simulator with the NEURON model description language (tutorial part D).
- As with channels, each point process has its own set of parameters. Below is a list of **`IClamp`** parameters:
  - **`del`**: The delay until the onset of the stimulus (in ms)
  - **`dur`**: The duration of the stimulus (in ms)
  - **`amp`**: The amplitude of the stimulus (in nA)

Assigning point process parameter values

- In our model, we want to stimulate the soma by giving it a current pulse. We can do this by adding a **`IClamp`** as above and then setting its parameters.

- There is no concept of default parameters of a point process. We must use the dot notation to assign the process parameter values:

```
stim.del = 100
stim.dur = 100
stim.amp = 0.1
```

- This creates an electrode current clamp in the centre of the soma (IClamp(0.5)) that will start injecting a 0.1nA current at a time instant 100ms from the beginning of simulation for a duration of 100ms.

The program `sthA.hoc` so far

```
create soma
access soma

soma nseg = 1
soma diam = 18.8
soma L = 18.8
soma Ra = 123.0

soma insert hh

objectvar stim
stim = new IClamp(0.5)

stim.del = 100
stim.dur = 100
stim.amp = 0.1

tstop = 300
```

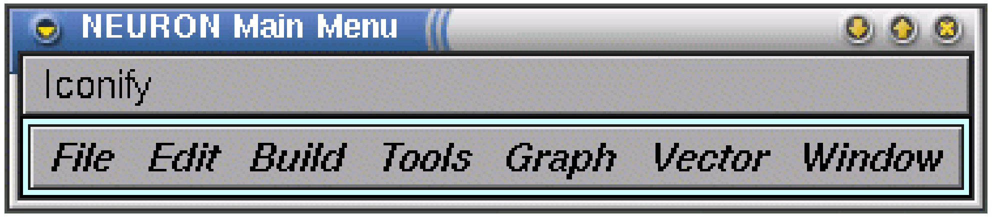
- The command **`tstop`** specifies the length of simulation in ms.

Running the stored model

- IMPORTANT: Don't forget to type the command **`load_file("nrngui.hoc")`** at the first line of your code.

- Store the file with your code as "name.hoc". Each time you want to run it, double click on the "name.hoc" file (e.g., **sthA.hoc**).

- This opens NEURON Main Menu on the screen.

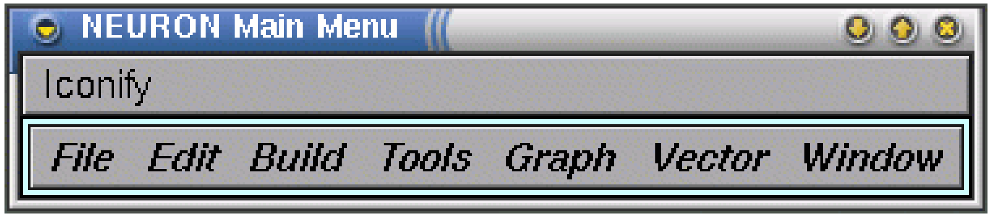


- Equivalent action in Linux/Unix: > **`nrngui sthA.hoc`**

Visualisation of simulation

- We will observe the voltage and current changes in the soma resulting from the square pulse of input current (Iclamp) over the period of simulation time = 300ms.

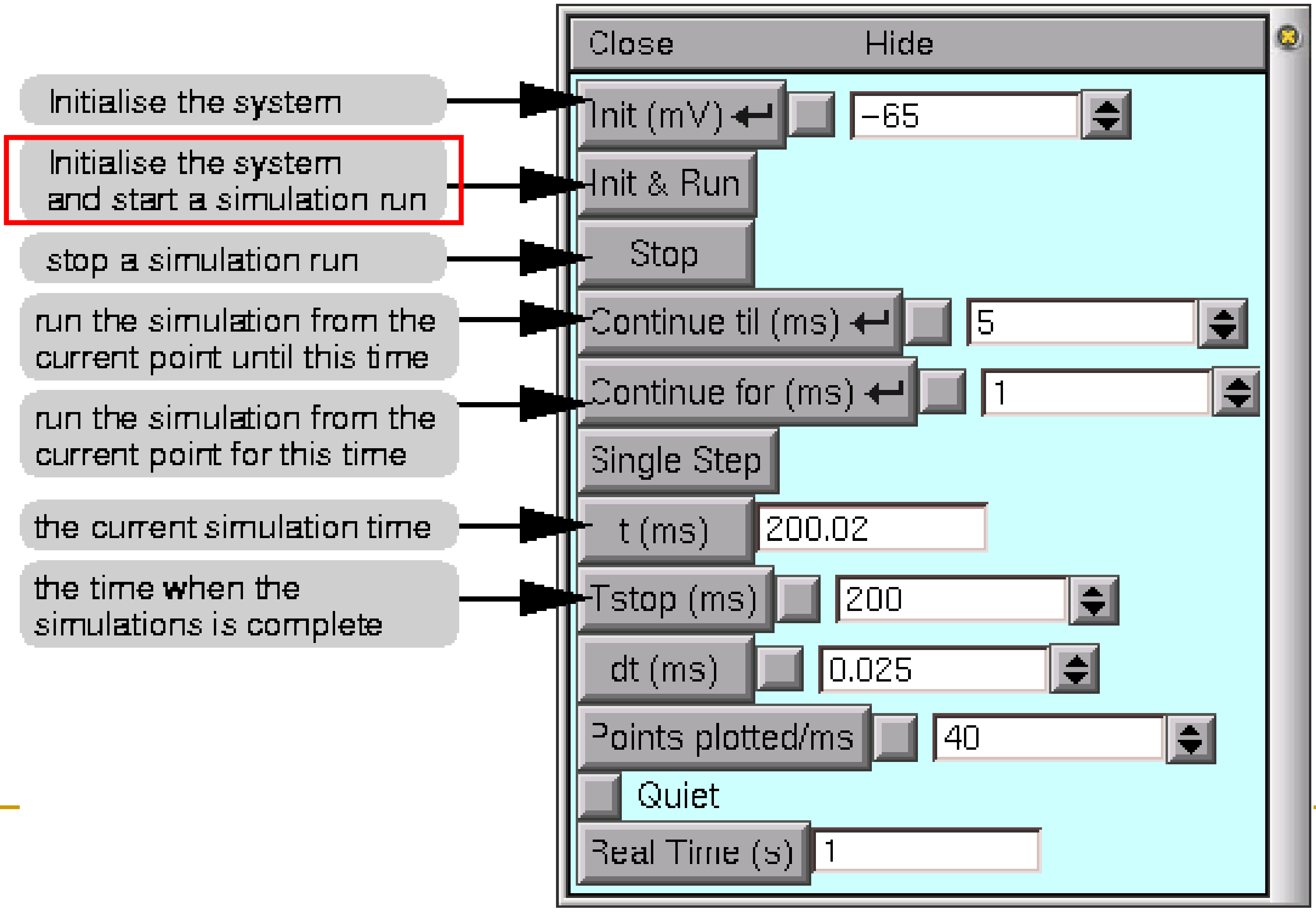
- Under the **Graph menu** of the main window, select **Voltage axis** and the **Current axis**.



- Right click on the Current axis window and choose **`Plot what?`** Select **`soma`** and click accept. Then choose **`ina(0.5)`** and click accept.

Running a simulation using *RunControl*

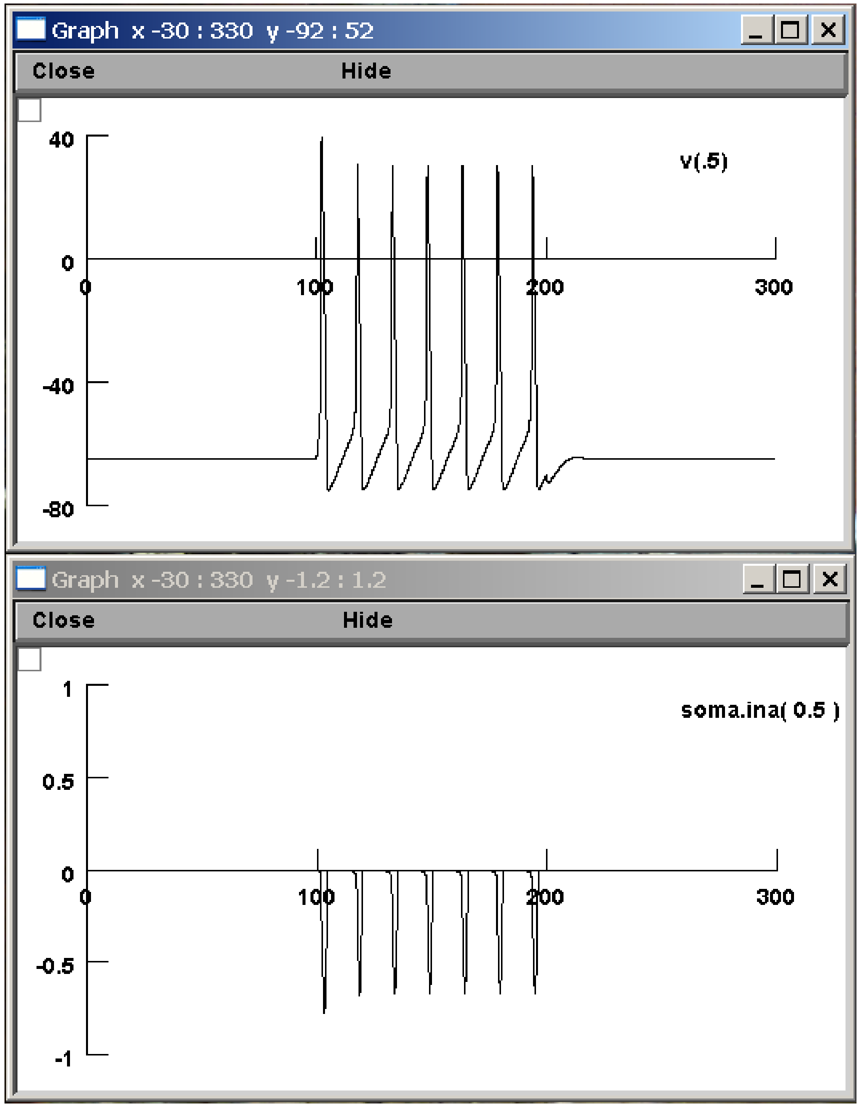
- This window allows you to control parameters for running the model. In the Main Menu toolbar, under the Tools menu **click Init & Run.**


Initialise the system
Initialise the system and start a simulation run
stop a simulation run
run the simulation from the current point until this time
run the simulation from the current point for this time
the current simulation time
the time when the simulations is complete
Close
Hide
Init (mV)
Init & Run
Stop
Continue til (ms)
Continue for (ms)
Single Step
t (ms)
Tstop (ms)
dt (ms)
Points plotted/ms
Quiet
Real Time (s)

Init (mV)	-65
Continue til (ms)	5
Continue for (ms)	1
t (ms)	200.02
Tstop (ms)	200
dt (ms)	0.025
Points plotted/ms	40
Real Time (s)	1

Inspecting the result

- Injected current causes soma to fire series of 7 consecutive spikes at regular intervals.
- Frequency = 7 spikes/ 0.1 sec = 70 Hz.
- Each spike is caused by an inward current of sodium Na^+ through the voltage-gated sodium channels.



Exercises

- Change parameters of stimulation, i.e. change the values **`stim.dur & stim.amp`** and run the simulation again to see how the output of soma changes.

- Return to the default values of stimulation and change the soma parameters. Observe the effect of **`diam`**, **`L`** and **`Ra`** upon the output.

```
load_file("nrngui.hoc")

access soma

soma nseg = 1
soma diam = 18.8
soma L = 18.8
soma Ra = 123.0

soma insert hh

objectvar stim

soma stim = new
IClamp(0.5)

stim.del = 100
stim.dur = 100
stim.amp = 0.1

tstop = 300
```

Running NEURON

- Download: go to http://www.neuron.yale.edu/neuron/download and **follow the instructions** for download and installation for particular OS (Windows, Linux/Unix).

- How do you start the simulator: Windows: go to the Start menu, choose Programs, find the NEURON program group, and select **`nrngui`** (or double-click the shortcut on the desktop). Linux/Unix: start NEURON by typing **> nrngui &**

- Then you have to download the `.hoc` file through the File menu.

- Alternatively, in MS Win double click on **`sthA.hoc`** file or in Linux/Unix type: > **`nrngui sthA.hoc`**