NEURON – tutorial A of Gillies & Sterratt

http://www.anc.ed.ac.uk/school/neuron/

Lubica Benuskova

COSC422 – lecture 3

Starting with NEURON

- Download: go to http://www.neuron.yale.edu/neuron/download and follow the instructions for download and installation for particular OS (Windows, Mac, 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).
  - Mac OS: double click on the **`nrngui`** icon in the folder where NEURON is installed.
  - Linux/Unix: start NEURON by typing at the command
    > nrngui &

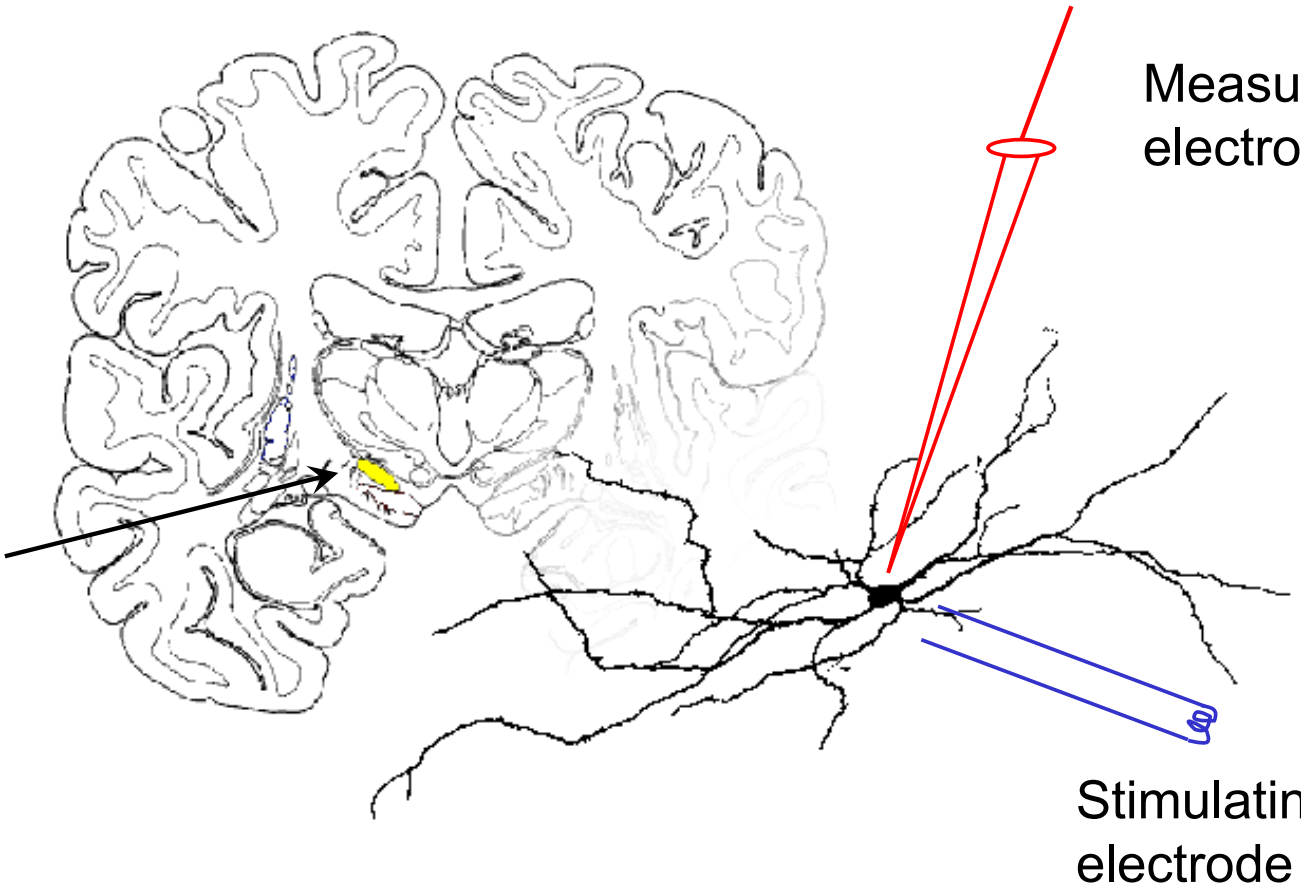
Tutorials by Andrew Gillies and David Sterratt

- The source of my NEURON lectures can be found at: http://www.anc.ed.ac.uk/school/neuron/

- Today we start with the basics: how to create a single soma with Hodgkin-Huxley conductances & how to run the simulator and display the simulation results (tutorial A of Gillies & Sterratt).

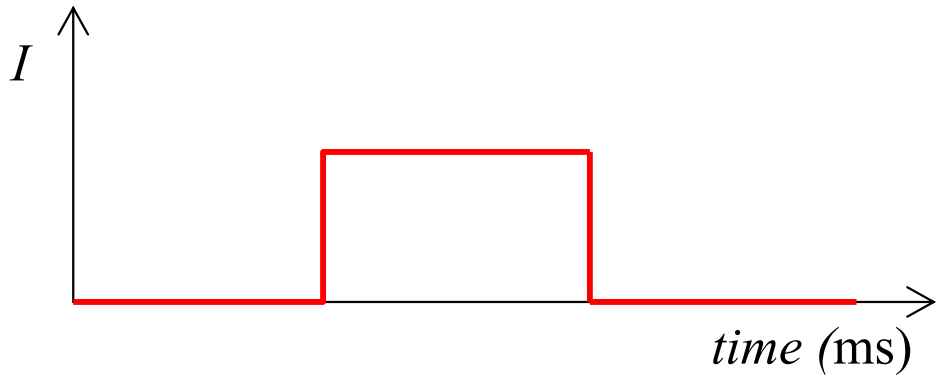
- Next time, this will be extended into a multi-compartmental neuron with dendrites (Tutorial B of Gillies & Sterratt).

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.

Start with soma



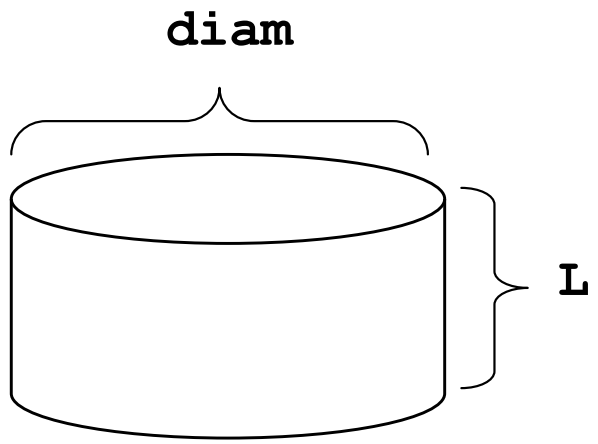
- Let's create a simulation of soma stimulated by a rectangular pulse of electric current.

- Write the code into the text editor and save the file as sthA.hoc in your working directory.

```
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
```

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/cm²**]
  - 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`**.
  - In general, **`nseg`** ≥ 1 when modelling a dendrite, for example.

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.

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

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

- While the former 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., a current clamp).
- The location is specified with a number between 0 and 1 (inclusive) where the number represents the fractional length along the section to place the point process.

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 (NEUR422).
- 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.

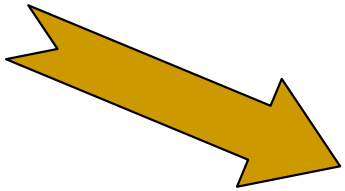
- There is no concept of default parameters of a point process. We must use the dot notation to assign the process parameter values:

```
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. This opens NEURON Main Menu on the screen when you double-click on the file next time you want to use it.

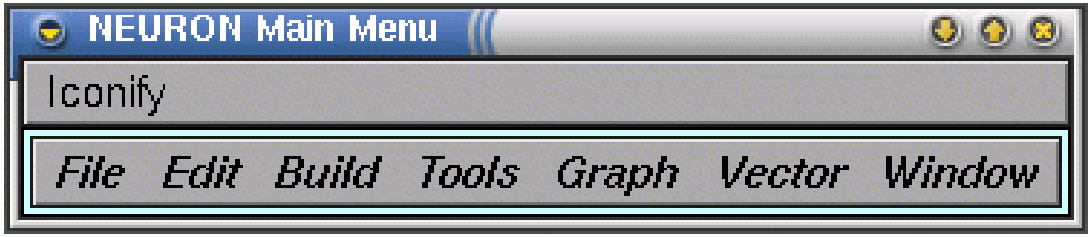
- Store the file with your code as "name.hoc". Next time you want to run it, double click on the "name.hoc" file (e.g., sthA.hoc) and the Main menu toolbar of NEURON appears on the screen.

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

Visualisation of simulation

- We want to observe the voltage changes in the soma due to the square pulse of input current over the period of simulation of 300ms.

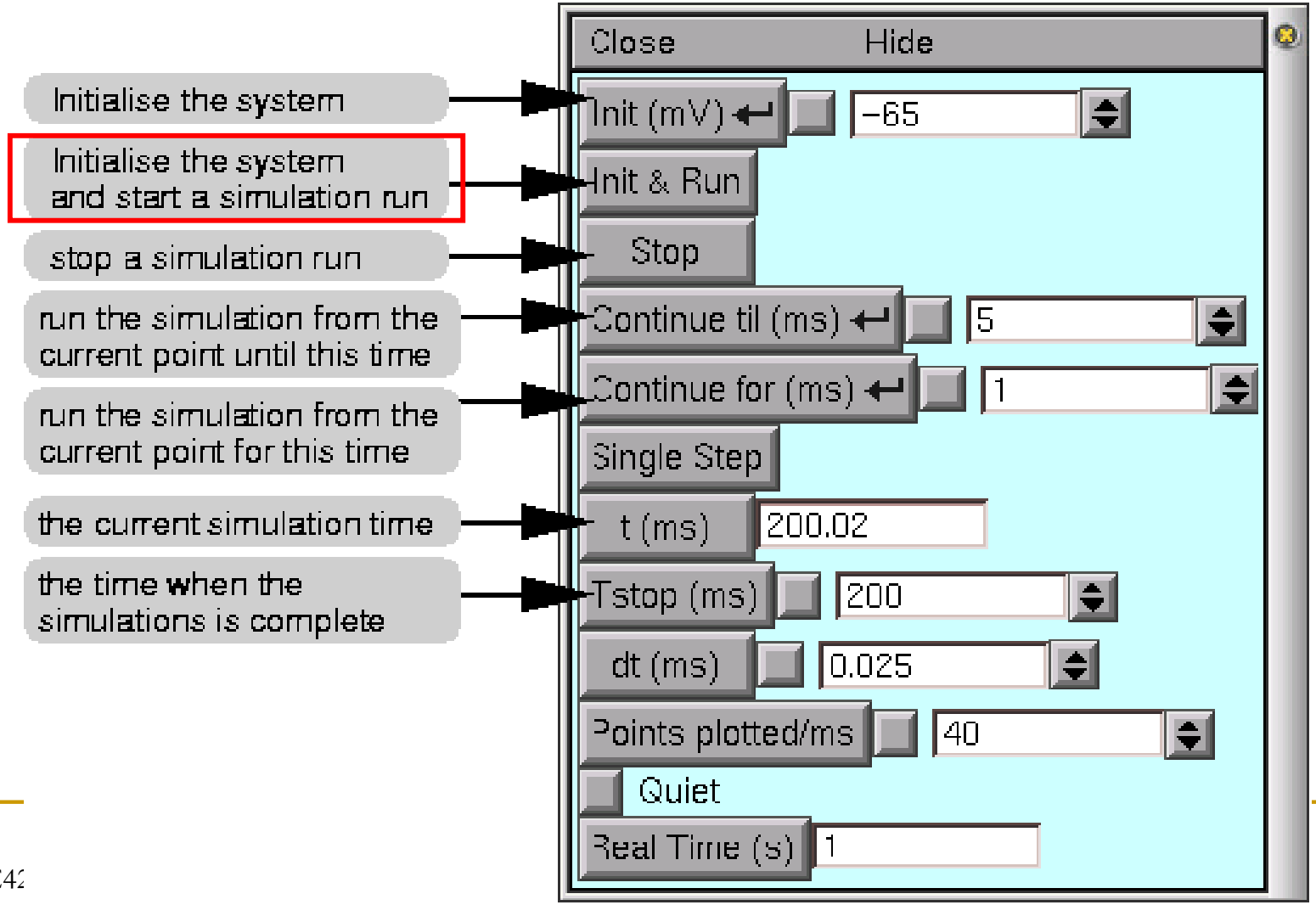
- Under the **Graph menu** of the main window, select **Voltage axis** and the **Current axis**.



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

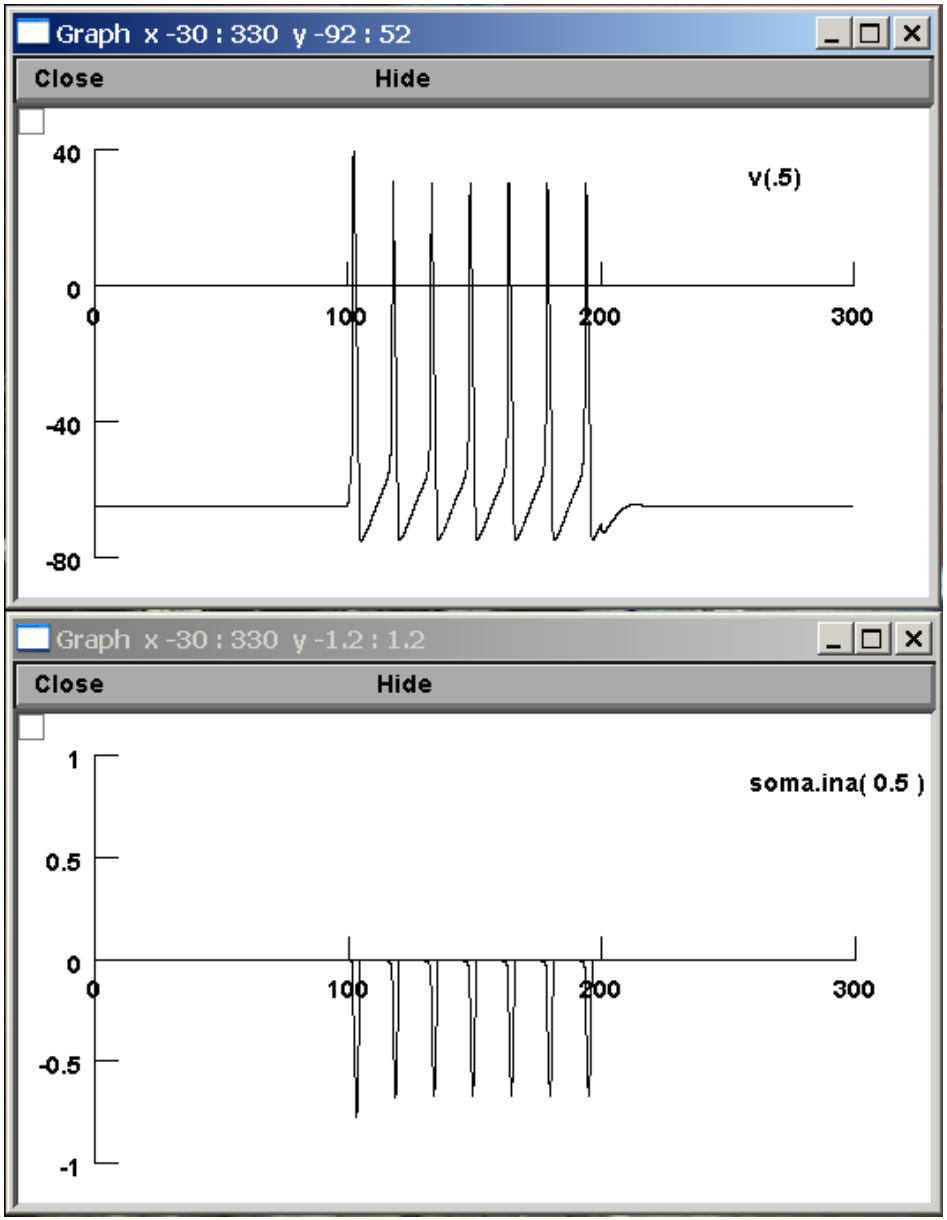
Running a simulation from *RunControl* window

- In the Main Menu toolbar, under the Tools menu select RunControl. This window allows you to control parameters for running the model.



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 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.

```
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
```