NEURON - tutorial D of Gillies & Sterratt (part 2)

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

COSC422 – lecture 11

How to program ion channels with **NMODL**

NMODL: creating the .mod file

- A description of a membrane mechanism in **NMODL** in a text file is divided into a number of blocks.

- Each block begins with a keyword defining the type of block, then an open brace "{", followed by block specific definitions, and finally, the block is ended with a closing brace "}".

- We will construct a NMODL file, called **CaT.mod** to calculate the low threshold calcium channel kinetics. As with hoc files we can use any text editor to create this file.

- (Note: there are a number of NMODL examples provided with the NEURON package.)

The TITLE and UNITS blocks

- We start the NMODL file with some standard definitions, i.e.:

```
TITLE Calcium T channel for Subthalamic Nucleus
UNITS {
        (mV) = (millivolt)
        (mA) = (milliamp)
}
```

- The TITLE keyword identifies what this file is describing.
- The UNITS block defines conventions for the units that will be used in this file. It uses these units to check that equations are consistent.
- By default, NMODL understands units in the UNIX units database (see file /usr/share/units.dat or look at the units command).

The NEURON block

- The NEURON block is the public interface of the mechanism. It tells the **hoc** interpreter how to refer to the mechanism and what variables it can see or change. The structure of the block is as follows:

```
NEURON {
    SUFFIX suffix
    USEION ions... READ vars... WRITE vars...
    RANGE var,var,...
    GLOBAL var,var,...
}
```

- Now we will explain each line in turn.

The NEURON block: **`SUFFIX suffix`**

- The first step is to identify this particular mechanism from all other membrane mechanisms when referencing it from the **`hoc`** file.
- This is done through the SUFFIX statement of the block.
- Access to all variables in this mechanism from the **`hoc`** file is then done using the **`suffix`**.
- For example, we will call this channel mechanism "**`CaT`**", so to access variables in the mechanism from hoc we use **`var_CaT`** (where **`var`** is a variable in this mechanism).

The NEURON block: **`USEION`**

- The **`USEION`** specifies what ions this channel mechanism uses.
- There are three implicitly defined ions NEURON knows about **`(na, k, ca)`** however, others may also be defined via this statement.
- NEURON can keep track of the intracellular and extracellular concentrations of each ion.
- Dealing with ions is difficult, because more than one mechanism may affect a particular ion. For example, we may have more than one calcium channel mechanism. Therefore, when dealing with ions use exactly the same name used in all other mechanisms.

USEION *ions* READ *vars* WRITE *vars*

- The READ modifier lists ion variables needed in calculating the ion channel current (usually the equilibrium potential, or concentration).
- The WRITE modifier lists what ion variables are calculated in this mechanism (usually the current). In our example we use Ca ion:
  - **`USEION ca READ eca WRITE ica`**
- where **`eca`** is the equilibrium potential for ion ca (calcium),
- **`ica`** is the calcium current

Note on how NEURON deals with ions

- Since we have just introduced ica, 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 WRITE cai and/or cao, the intra- and extracellular calcium concentrations. However this mechanism would need to know the total calcium current ica 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.

The NEURON block: `Range and Global`

- The RANGE statement makes the following variables visible to the NEURON interpreter and that they are to be functions of a position.

- The GLOBAL statement specifies variables that are always the same for the mechanism. CaT mechanism does not have any GLOBAL variables. Our final NEURON block now has the form:

```
NEURON {
        SUFFIX CaT
        USEION ca READ eca WRITE ica
        RANGE gmax
}
```

The PARAMETER block

- The PARAMETER block in **`CaT.mod`** is this:

```
PARAMETER {
        gmax   = 0.002 (mho/cm2)
}
```

- For each parameter, we specify the name of the parameter, its default value and its units (in parentheses).

- The PARAMETER block specifies variables that:
  - are not changed as a result of the calculations in the mechanism;
  - are (generally) constant throughout time; and
  - can be changed in the **`hoc`** file, e.g. `soma gmax_CaT = 0.001`

The ASSIGNED block

- The ASSIGNED block declares variables that are either:

- calculated by the mechanism itself or
- computed by NEURON.

- Variables that this mechanism will compute are the calcium current ica, and variables for the rate equations **`ralpha, rbeta, salpha,`** etc.

- The variables that the mechanism uses that are computed by NEURON are the membrane potential *V* and the calcium equilibrium potential **`eca`**.

The ASSIGNED block

- For **CaT**, the ASSIGNED block looks like this:

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

The heart of CaT mechanism

- Recall we want to calculate:

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

- We wish to calculate the values of the state variables *r*, s and *d* in order to calculate the calcium current from the above equation.
- The state variables are given by the three kinetic differential equations:

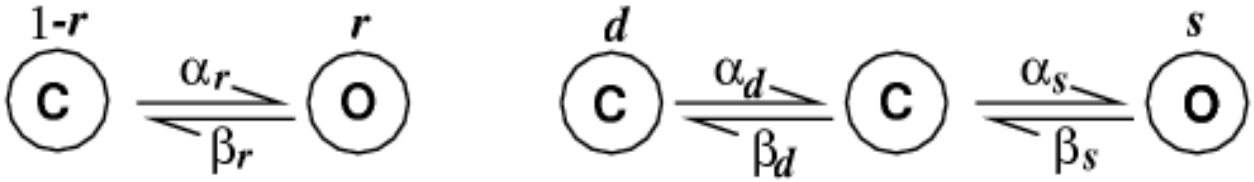
$$\dot{r} = \alpha_r (1 - r) - \beta r$$

$$\dot{s} = \alpha_s (1 - s - d) - \beta_s s$$

$$\dot{d} = \alpha_d (1 - s - d) - \beta_d d$$

The STATE block

- The STATE block declares state variables.
- There are 3 state variables in our kinetic channel model, *r*, *s* and *d*.



- For **CaT**, the STATE block looks like this:

```
STATE {
    r s d
}
```

PROCEDURE

- However, we must first calculate each of the rate functions ralpha, rbeta, salpha, sbeta, dalpha and dbeta.

- We can create a PROCEDURE to do this.

- A procedure is defined using the following format:

```
PROCEDURE name(vars) {
    calculations...
}
```

```
PROCEDURE settables(v (mV)) {
    LOCAL  bd

    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
}
```

Why we call procedure **`settables`**

- The above procedure takes the current voltage v as an argument **`(vars)`** and calculates values of the rate functions ralpha, rbeta, etc.
- The rate functions will need to be reevaluated at each time step.
- However, as the voltage is changing, it is more computationally efficient to create a table of values calculated at closely spaced voltages at the start of a simulation, and use the table lookup with linear interpolation based on the current voltage (memory is cheaper than computation).
- This can be done by adding a TABLE line to the procedure.

Procedure **settables**

- The TABLE command has the form:

```
TABLE funcs DEPEND vars FROM lowest TO highest WITH steps
```

- Where **funcs**, are the variables representing the functions to create tables for (e.g. the alpha and beta function variables)

- **vars** are those variables, which if they change value then all tables must be recalculated.

- **lowest** and **highest** are the lowest and highest values of the voltage we make the tables over, with steps **steps** between them.

```
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
}
```

The DERIVATIVE block

$$\dot{r} = \alpha_r(1-r) - \beta r$$
$$\dot{s} = \alpha_s(1-s-d) - \beta_s s$$
$$\dot{d} = \alpha_d(1-s-d) - \beta_d d$$

- The α and β rate functions are used in equations:
- These are specified in the DERIVATIVE block, which we will call **`states`**.

```
DERIVATIVE states {
    settables(v)
    r' = ((ralpha*(1-r)) - (rbeta*r))
    d' = ((dbeta*(1-s-d)) - (dalpha*d))
    s' = ((salpha*(1-s-d)) - (sbeta*s))
}
```

- Each time NEURON calculates the differential equations, the a and b must be updated, so the first line calls the procedure **`settables`** with the current voltage v.

The BREAKPOINT block

- The BREAKPOINT is the top level mechanism calculation block that calculates the calcium current:

```
BREAKPOINT {
        SOLVE states METHOD cnexp
        ica = gmax*r*r*r*s*(v-eca)
}
```

- The Ca current is calculated according to the equation:

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

- The SOLVE statement refers to the states defined in the DERIVATIVE block. The METHOD cnexp part of the line tells NEURON to use the "**cnexp**" method of integration, which is suitable for mechanisms of the form: $\mathrm{d}x/\mathrm{d}t = f(V, x)$, where f is linear in x and involves no other states.

The INITIAL block

- This block is the last one. It is used to set the state variables *r*, *d* and *s* to their initial values. The INITIAL block first calls the procedure **`settables`** with the present voltage to calculate the values of the α's and β's, which are used to calculate the initial values of *r*, *d* and *s* .

```
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))
}
```

Putting it all together

```
TITLE Calcium T channel for Subthalamic Nucleus
UNITS {…}
NEURON {…}
PARAMETER {…}
ASSIGNED {…}
STATE {…}
BREAKPOINT {…}
INITIAL {…}
DERIVATIVE states {…}

UNITSOFF
PROCEDURE settables(v (mV)) {…}
UNITSON
```

- the commands UNITSON and UNITSOFF in the file activate the units checking (e.g. mV, mA etc.) and deactivate it respectively.

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)
```