NMODL

The NEURON Model Description Language

Add new membrane mechanisms to NEURON

Density mechanisms

- distributed channels
- ion accumulation

Point Processes

- electrodes
- synapses

Described by

- differential equations
- kinetic schemes
- algebraic equations

Advantages

- Specification only--independent of solution method
- Efficient--translated into C
- Compact
  - One NMODL statement $\rightarrow$ many C statements
  - Interface code automatically generated
- Consistent ion current / concentration interactions
- Consistent units

NMODL general block structure

What the model looks like from outside

```
NEURON {
   SUFFIX kchan
   USEION k READ ek WRITE ik
   RANGE gbar, . . .
}
```

What names are manipulated by this model

```
UNITS { (mv) = (millivolt) . . . }
PARAMETER { gbar = 0.036 (S/cm2) <0, 1e9> . . . }
STATE { n . . . }
ASSIGNED { ik (mA/cm2) . . . }
```

Initial default values for states

```
INITIAL {
   rates(v)
   n = ninf
}
```

Calculate currents (if any) as function of v, t, states

(and specify how states are integrated)

```
BREAKPOINT {
   SOLVE deriv METHOD cnexp
   ik = gbar * n^4 * (v - ek)
}
```

State equations

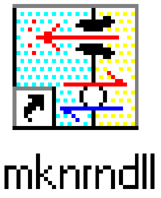
```
DERIVATIVE deriv {
   rates(v)
   n' = (ninf - n)/ntau
}
```

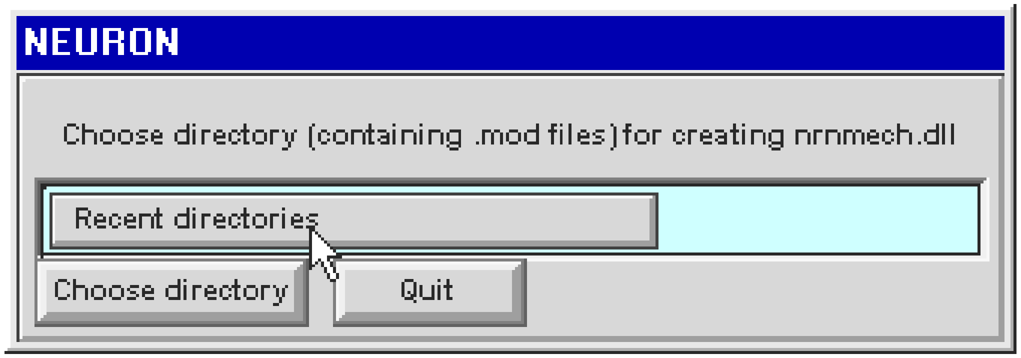
Functions and procedures

```
PROCEDURE rates(v(mV)) {
   . . .
}
```

UNIX `nrnivmodl`

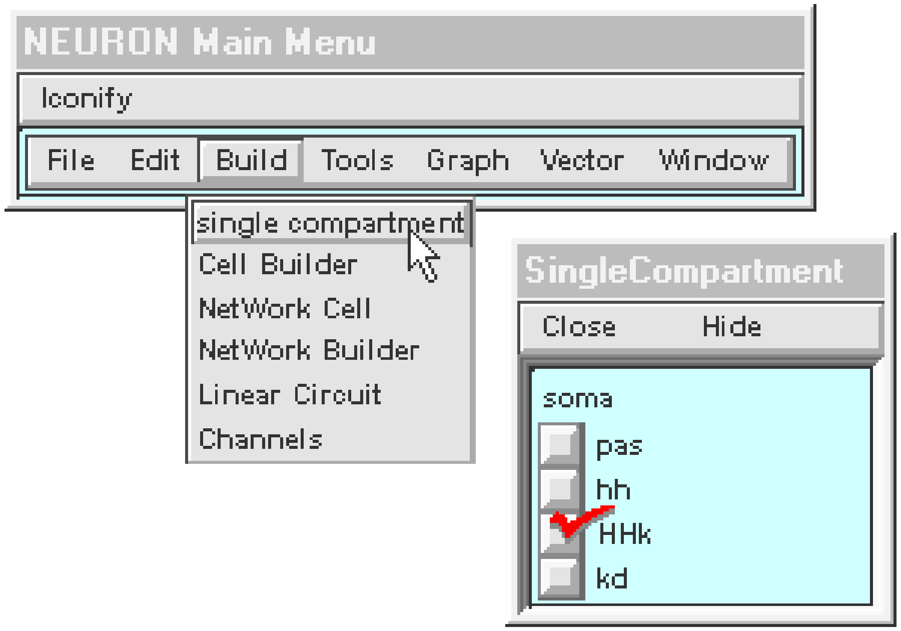
MSWin





Choose directory (containing .mod files)for creating nrnmech.dll

Result: NEURON has a new mechanism



Density mechanism

```
NEURON {
   SUFFIX leak
   NONSPECIFIC_CURRENT i
   RANGE i, e, g
}

PARAMETER {
   g = 0.001 (mho/cm2) <0, 1e9>
   e = −65 (millivolt)
}

ASSIGNED {
   i (milliamp/cm2)
   v (millivolt)
}

BREAKPOINT {
   i = g*(v - e)
}
```

Point Process

```
NEURON {
   POINT_PROCESS Shunt
   NONSPECIFIC_CURRENT i
   RANGE i, e, r
}

PARAMETER {
   r = 1 (gigaohm) <1e−9,1e9>
   e = 0 (millivolt)
}

ASSIGNED {
   i (nanoamp)
   v (millivolt)
}

BREAKPOINT {
   i = (0.001)*(v − e)/r
}
```

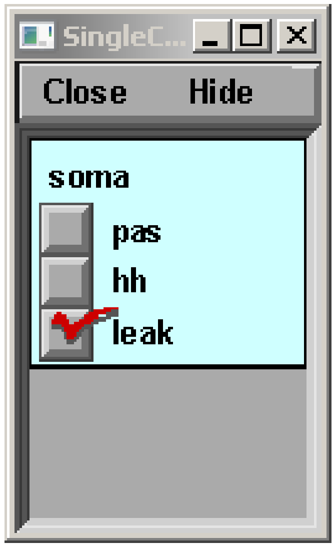
Density mechanism / Point Process

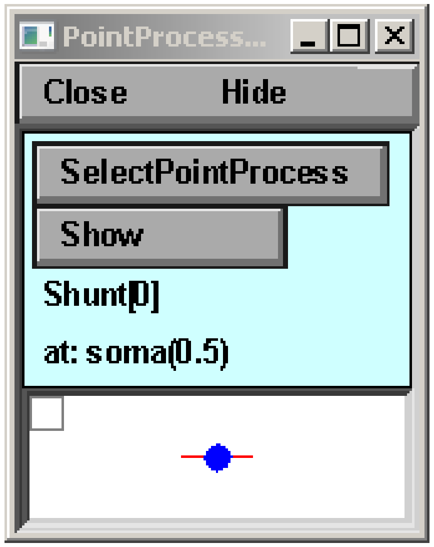
NMODL

```
NEURON {
    SUFFIX leak
    NONSPECIFIC_CURRENT i
    RANGE i, e, g
}
```

```
NEURON {
    POINT_PROCESS Shunt
    NONSPECIFIC_CURRENT i
    RANGE i, e, r
}
```

GUI





Interpreter

hoc:

```
soma {
    insert leak
    g_leak = 1e-4
}
print soma.i_leak(0.5)
```

```
objref s
soma s = new Shunt(0.5)
s.r = 2

print s.i
```

python:

```
soma.insert('leak')
soma.g_leak = 1e-4
print(soma(0.5).leak.i)
```

```
s = h.Shunt(soma(0.5))
s.r = 2.0
print(s.i)
```

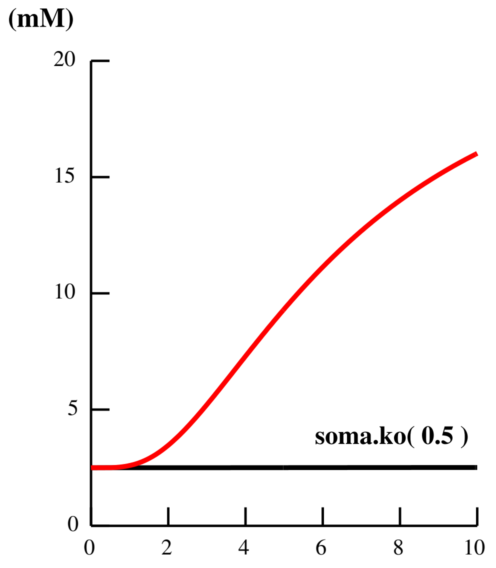
Ion Channel

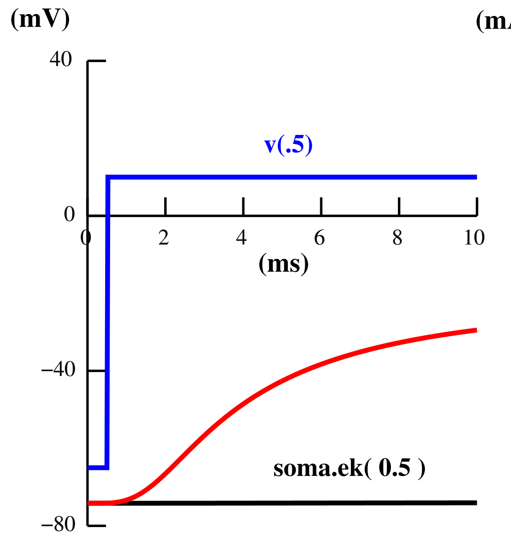
```
NEURON {
  USEION k READ ek WRITE ik
}
BREAKPOINT {
  SOLVE states METHOD cnexp
  ik = gbar*n*n*n*n*(v - ek)
}
DERIVATIVE states {
  rate(v*1(/mV))
  n' = (inf - n)/tau
}
```

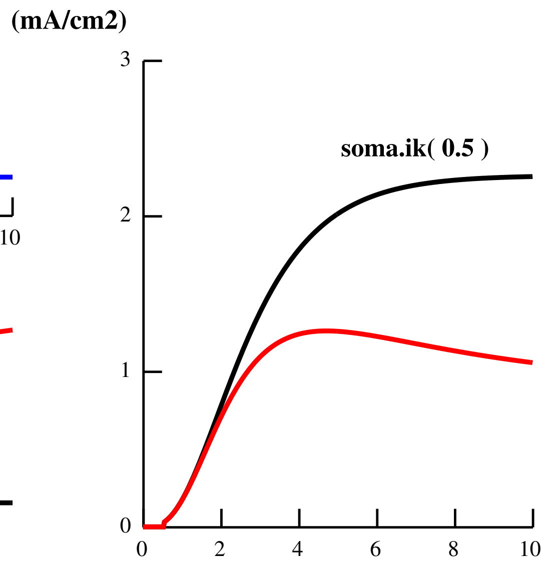
Ion Accumulation

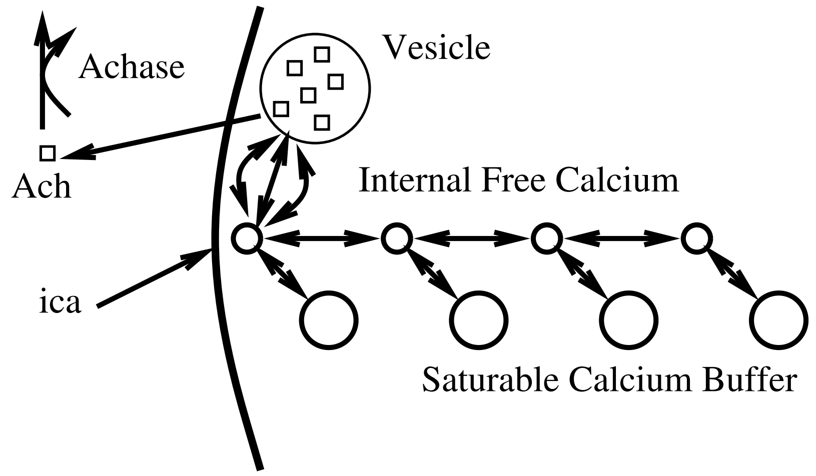
```
NEURON {
  USEION k READ ik WRITE ko
}
BREAKPOINT {
  SOLVE state METHOD cnexp
}

DERIVATIVE state {
  ko' = ik/fhspace/F*(1e8)
         + k*(kbath - ko)
}
```









```
STATE {
   Vesicle Ach Achase Ach2ase X Buffer[N] CaBuffer[N] Ca[N]
}
KINETIC calcium_evoked_release  {
   : release
 ~ Vesicle + 3Ca[0] <-> Ach  (Agen, Arev)
 ~ Ach + Achase <-> Ach2ase  (Aase2, 0)  : idiom for enzyme reaction
 ~ Ach2ase <-> X + Achase    (Aase2, 0)  : requires two reactions
   : Buffering
   FROM i = 0 TO N-1 {
     ~ Ca[i] + Buffer[i] <-> CaBuffer[i]  (kCaBuffer, kmCaBuffer)
   }
   : Diffusion
   FROM i = 1 TO N-1 {
     ~ Ca[i-1] <-> Ca[i]     (Dca*a[i-1], Dca*b[i])
   }
   : inward flux
 ~ Ca[0] <<      (ica)
}
```

UNITS Checking

```
NEURON { POINT_PROCESS Shunt ... }
PARAMETER {
    e = 0 (millivolt)
    r = 1 (gigaohm) <1e-9,1e9>
}
ASSIGNED {
    i (nanoamp)
    v (millivolt)
}
BREAKPOINT {
    i = (v - e)/r
}
```

Units are incorrect in the "i = ..." current assignment.

```
BREAKPOINT {
    i = (v - e)/r
}
```

The output from

modlunit shunt

is:

```
    Checking units of shunt.mod
    The previous primary expression with units: 1-12 coul/sec
    is missing a conversion factor and should read:
      (0.001)*()
    at line 14 in file shunt.mod
           i = (v - e)/r<>
```

To fix the problem replace the line with:

```
           i = (0.001)*(v - e)/r
```

What conversion factor will make the following consistent?

```
     nai'    =   ina      /     FARADAY     *     (c/radius)
   (uM/ms)   (mA/cm2)   /   (coulomb/mole)           /  (um)
```