Modelling intracellular cascades and neural plasticity

Simula summer school

27.6.2019

Nicolangelo Iannella (Tuomo Mäki-Marttunen)

Outline

- Brief introduction to LTP/LTD
- Modeling approaches
- Formulation of the mass-action laws
- Simulation in NEURON’s RxD extension
- A relevant detour: Extending cable theory for calcium based plasticity

Background

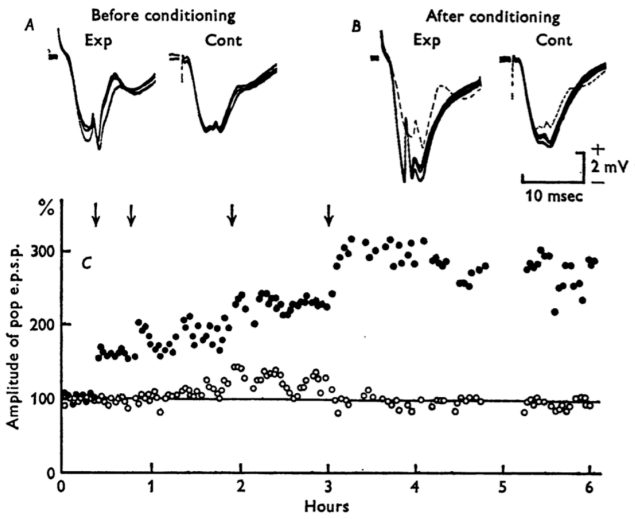
- W. James (1890): a modern perspective for the problem of plasticity
- S.R. y Cajal (1894): formation of new connections needed for learning
- Hebb's rule (1949): "neurons that fire together wire together"
- Bliss & Lomø (1973): LTP discovered
- Ito et al. (1982): LTD discovered
- Bhalla & Yiengar (1999): Unified modeling framework for many underlying pathways

J. Physiol. (1973), **232**, *pp.* 331–356
With 12 text-figures
Printed in Great Britain

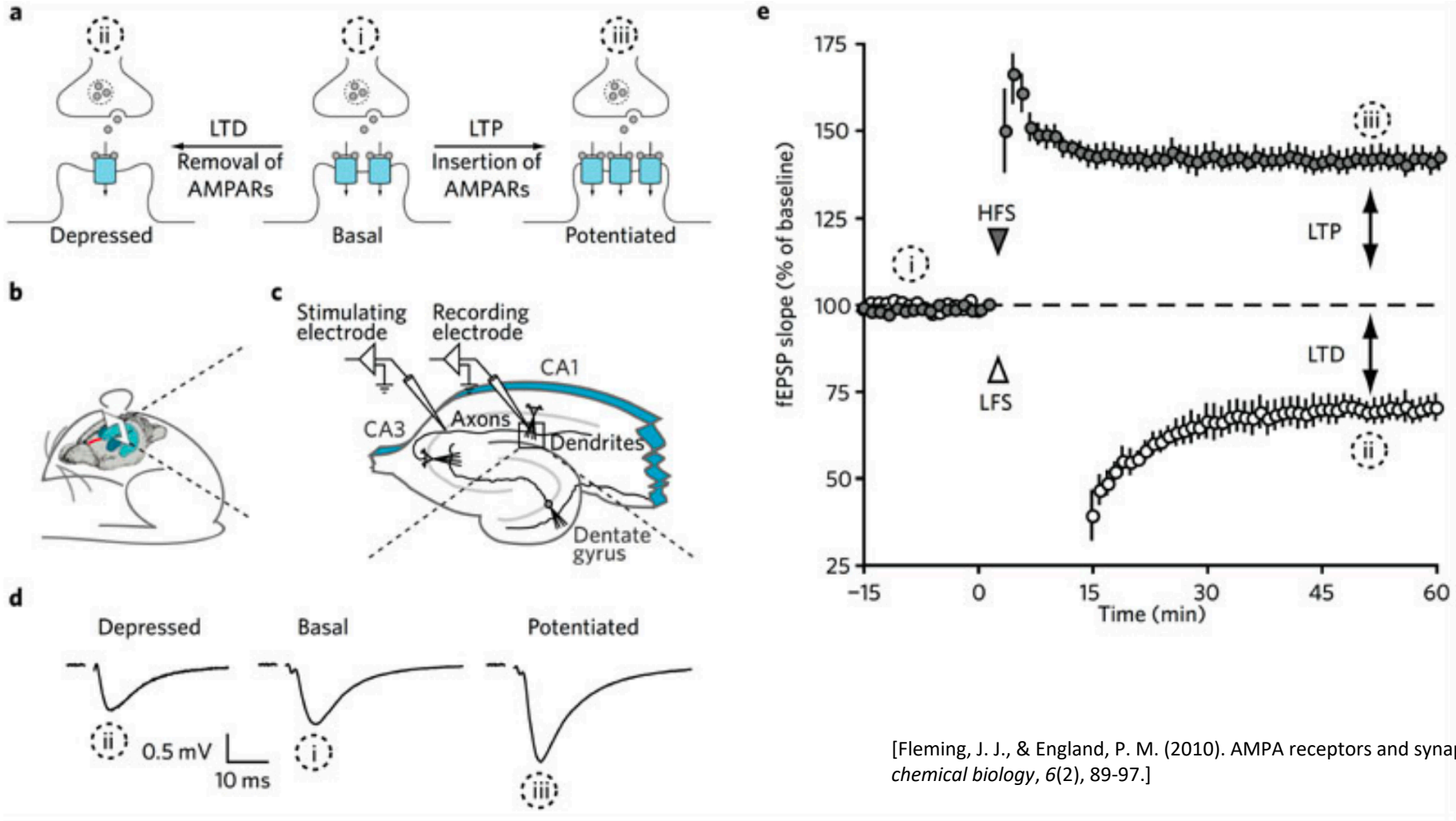
LONG-LASTING POTENTIATION OF SYNAPTIC TRANSMISSION IN THE DENTATE AREA OF THE ANAESTHETIZED RABBIT FOLLOWING STIMULATION OF THE PERFORANT PATH

By T. V. P. BLISS and T. LØMO
From the National Institute for Medical Research, Mill Hill, London NW7 1AA and the Institute of Neurophysiology, University of Oslo, Norway

(*Received 12 February 1973*)

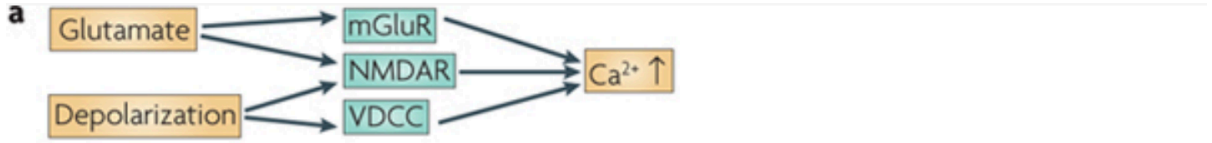


Long-term synaptic plasticity



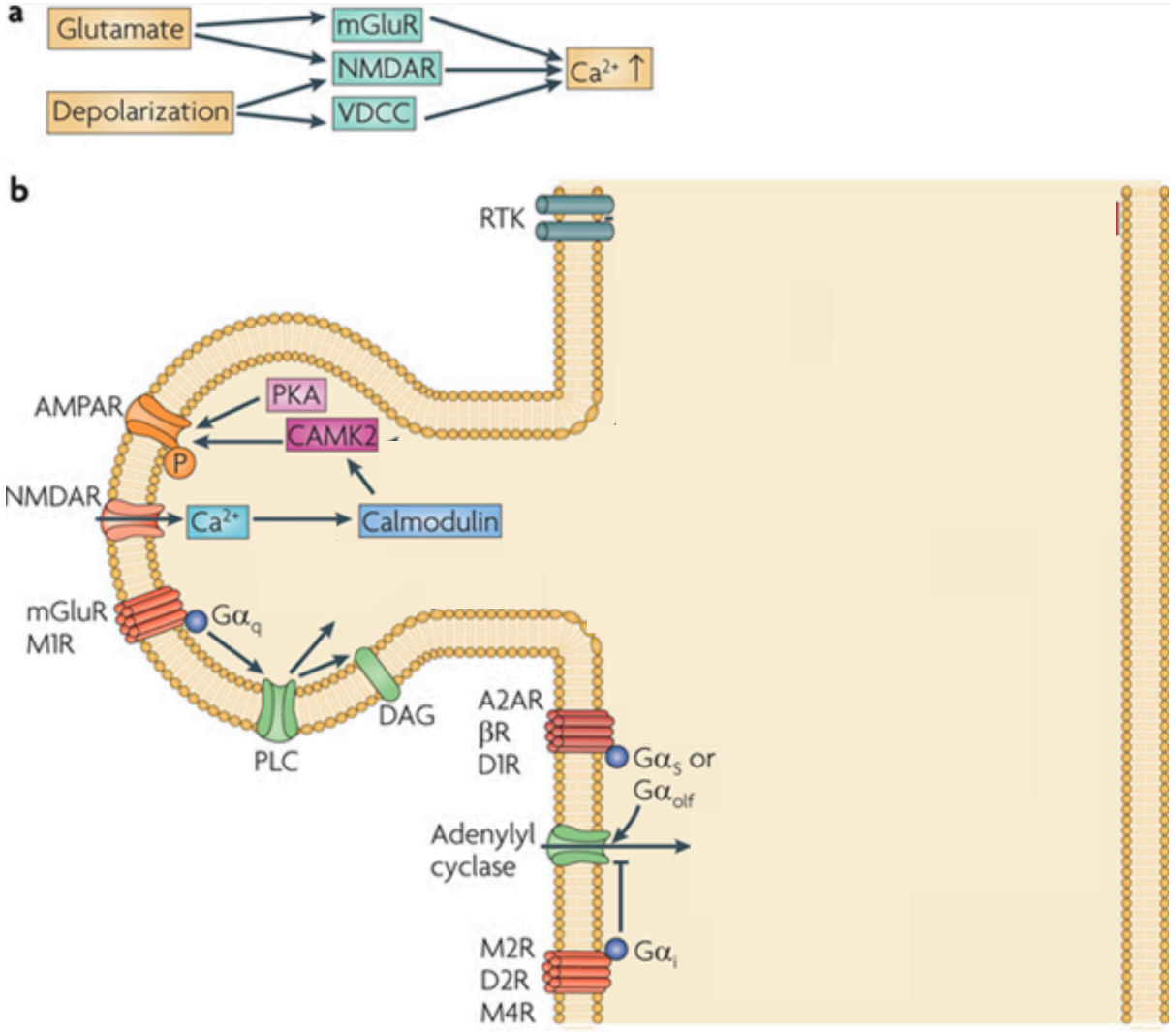
[Fleming, J. J., & England, P. M. (2010). AMPA receptors and synaptic plasticity: a chemist's perspective. *Nature chemical biology*, *6*(2), 89-97.]

Long-term synaptic plasticity: cellular mechanisms



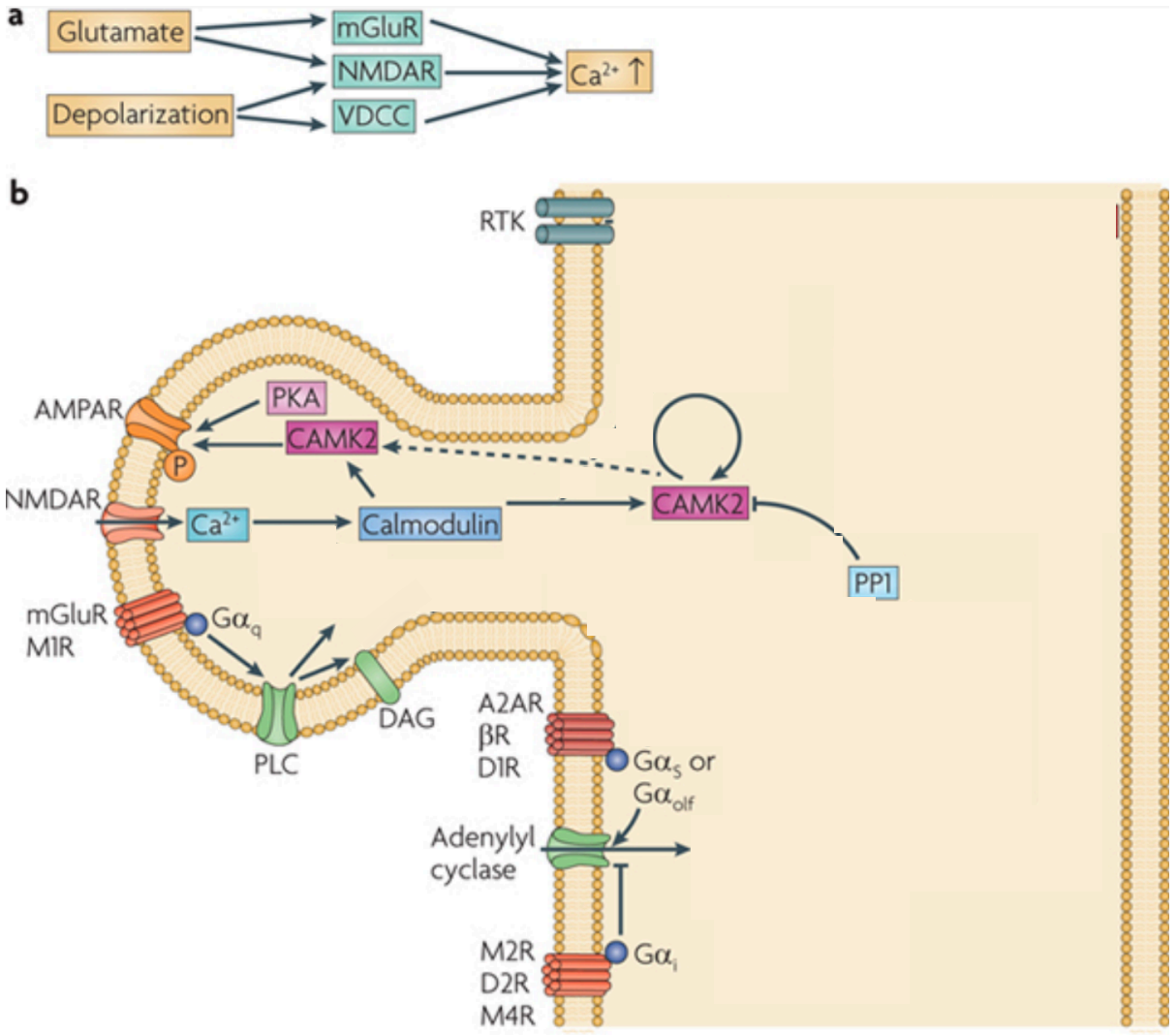
[Kotaleski, J. H., & Blackwell, K. T. (2010). Modelling the molecular mechanisms of synaptic plasticity using systems biology approaches. *Nature Reviews Neuroscience*, *11*(4), 239-251.]

Long-term synaptic plasticity: cellular mechanisms



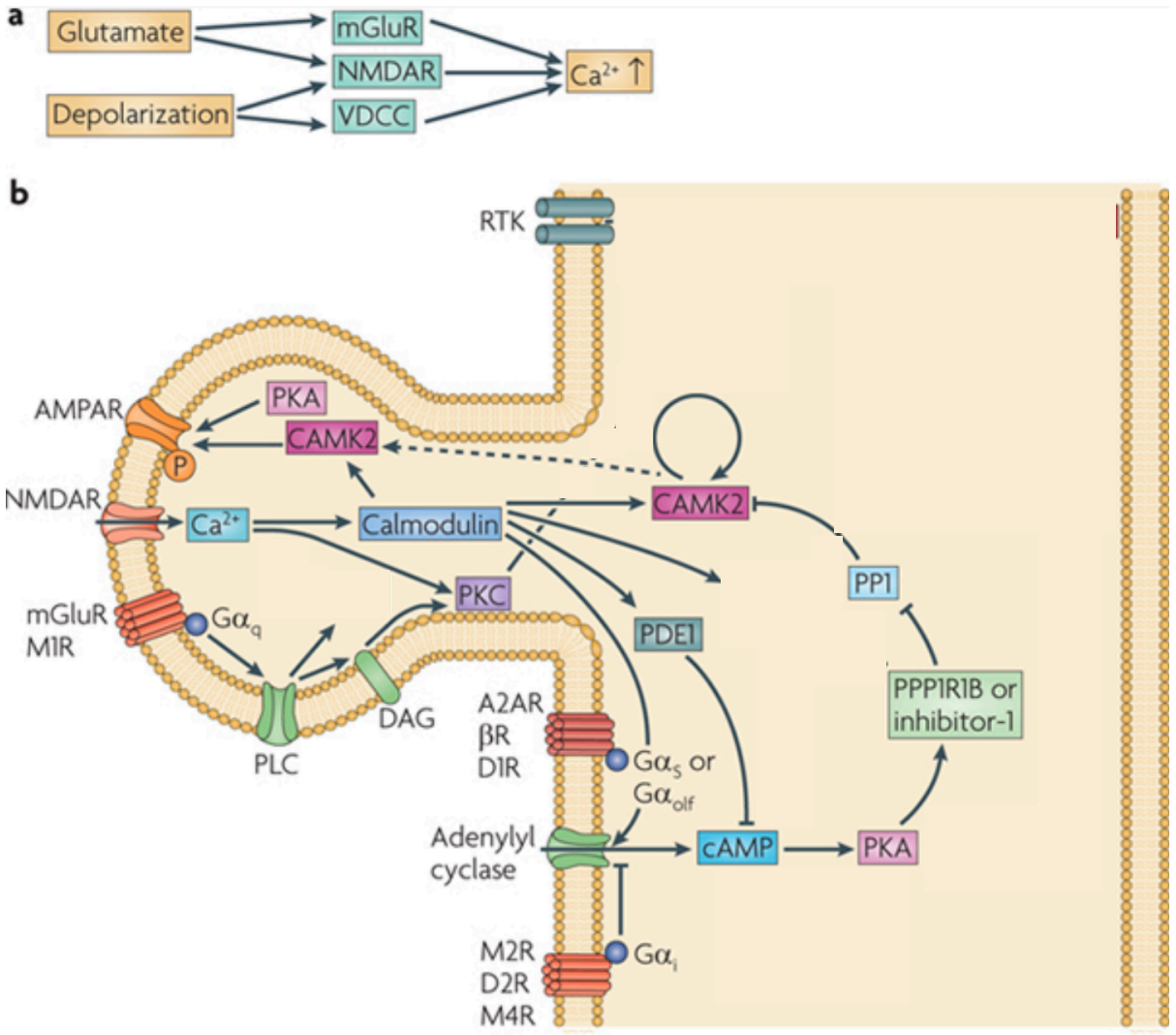
[Kotaleski, J. H., & Blackwell, K. T. (2010). Modelling the molecular mechanisms of synaptic plasticity using systems biology approaches. *Nature Reviews Neuroscience*, *11*(4), 239-251.]

Long-term synaptic plasticity: cellular mechanisms



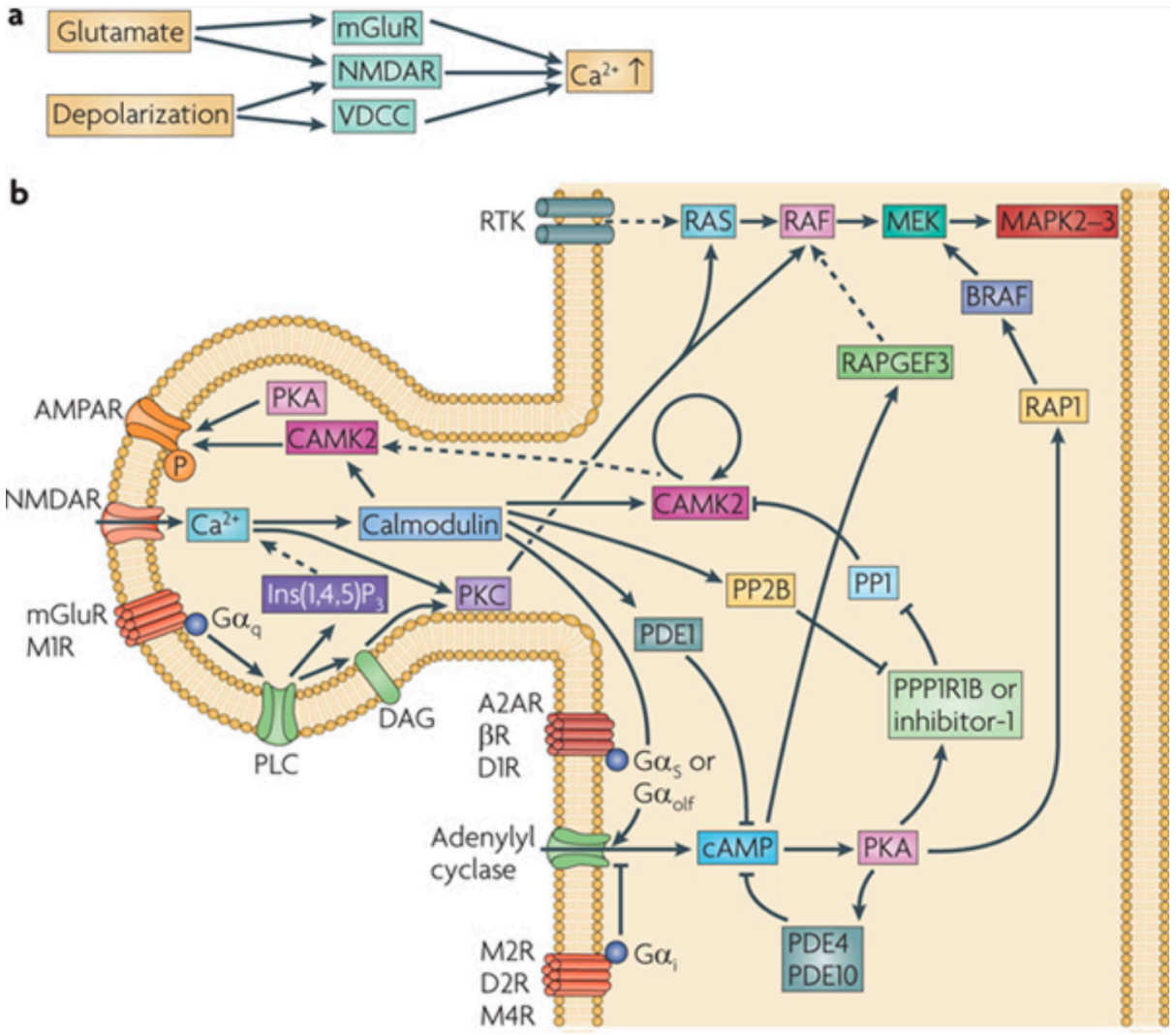
[Kotaleski, J. H., & Blackwell, K. T. (2010). Modelling the molecular mechanisms of synaptic plasticity using systems biology approaches. *Nature Reviews Neuroscience*, *11*(4), 239-251.]

Long-term synaptic plasticity: cellular mechanisms



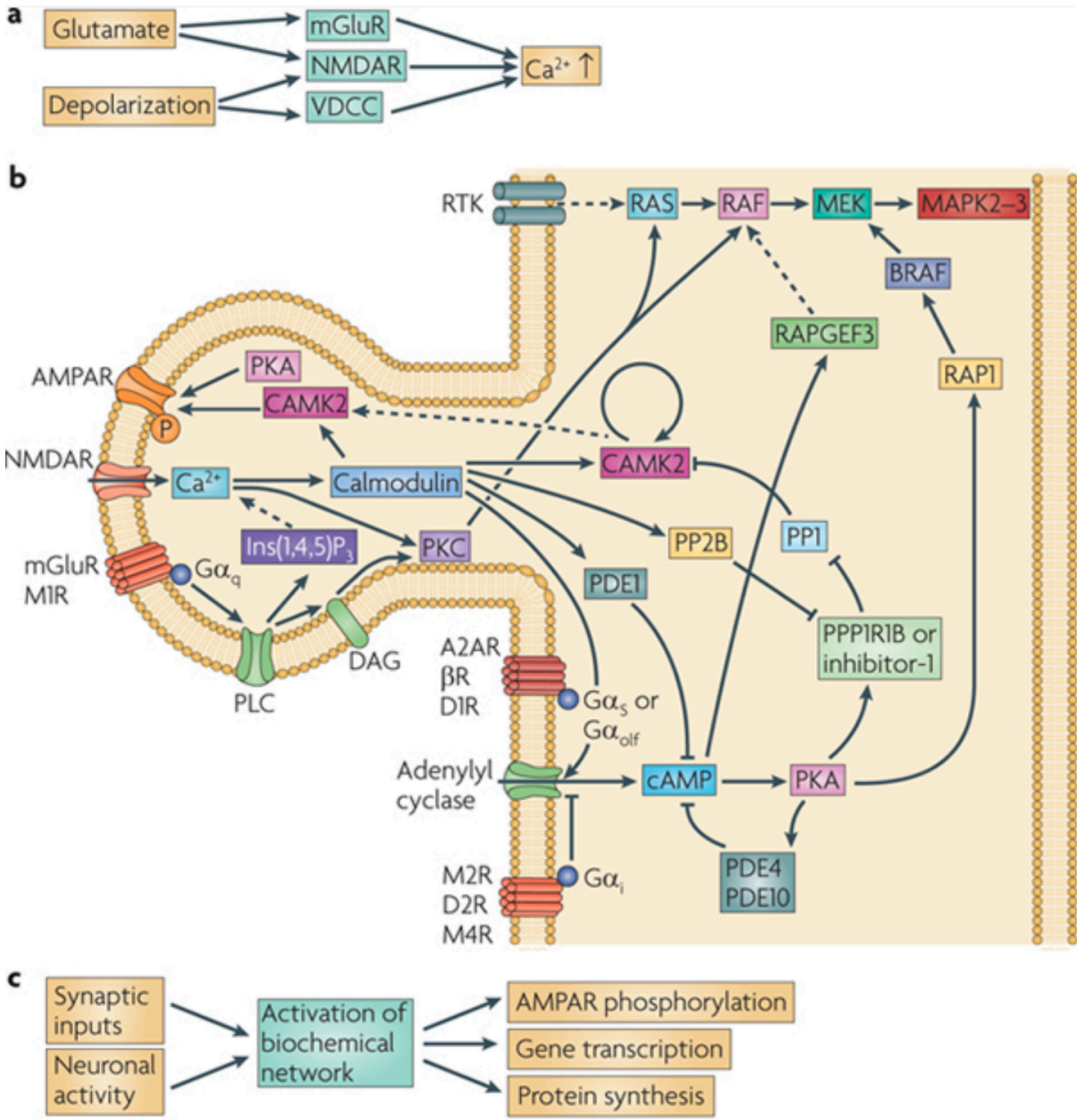
[Kotaleski, J. H., & Blackwell, K. T. (2010). Modelling the molecular mechanisms of synaptic plasticity using systems biology approaches. *Nature Reviews Neuroscience*, *11*(4), 239-251.]

Long-term synaptic plasticity: cellular mechanisms



[Kotaleski, J. H., & Blackwell, K. T. (2010). Modelling the molecular mechanisms of synaptic plasticity using systems biology approaches. *Nature Reviews Neuroscience*, *11*(4), 239-251.]

Long-term synaptic plasticity: cellular mechanisms



[Kotaleski, J. H., & Blackwell, K. T. (2010). Modelling the molecular mechanisms of synaptic plasticity using systems biology approaches. *Nature Reviews Neuroscience*, *11*(4), 239-251.]

Modeling approaches

- Almost all models based on the law of mass action

$$A + B \underset{k_b}{\overset{k_f}{\rightleftharpoons}} C + D$$

- Deterministic models: $d[A]/dt = k_b[\mathrm{C}][\mathrm{D}] - k_f[\mathrm{A}][\mathrm{B}]$
- Stochastic models
  - Gillespie's algorithm and its variants: at each iteration, sample the type and time of the next reaction in each voxel
  - Particle-based models: simulate movement of each particle separately, apply reactions when collisions

Modeling approaches

- Almost all models based on the law of mass action

$$A + B \underset{k_b}{\overset{k_f}{\rightleftharpoons}} C + D$$

- Deterministic models: $d[A]/dt = k_b[\mathrm{C}][\mathrm{D}] - k_f[\mathrm{A}][\mathrm{B}]$ COPASI NEURON
- Stochastic models
  - Gillespie's algorithm and its variants: at each iteration, sample the type and time of the next reaction in each voxel STEPS NeuroRD
  - Particle-based models: simulate movement of each particle separately, apply reactions when collisions MCell

Modeling approaches

- Almost all models based on the law of mass action

$$A + B \underset{k_b}{\overset{k_f}{\rightleftharpoons}} C + D$$

- Deterministic models: $d[A]/dt = k_b[\mathrm{C}][\mathrm{D}] - k_f[\mathrm{A}][\mathrm{B}]$

COPASI NEURON

- Stochastic models
  - Gillespie's algorithm and its variants: at each iteration, sample the type and time of the next reaction in each voxel

STEPS NeuroRD

  - Particle-based models: simulate movement of each particle separately, apply reactions when collisions

MCell

Challenges

- Model construction
  - Reaction rates usually cannot be fitted to data from the modeled system
  - Difficult to determine the concentrations of the species
  - Large variability in the reproduced phenomena

- Model simulation
  - Multiple spatial and temporal scales
  - Model interaction with HH-type models often not considered

Differential equations from reactions

- Which of these diff. equations describes the following reaction?

$$A \underset{k_b}{\overset{k_f}{\longleftrightarrow}} B$$

1) $\frac{d[A]}{dt} = k_b[B], \frac{d[B]}{dt} = k_f[A]$

2) $\frac{d[A]}{dt} = \frac{k_f}{k_b}\frac{[A]}{[B]}, \frac{d[B]}{dt} = \frac{k_b}{k_f}\frac{[B]}{[A]}$

3) $\frac{d[A]}{dt} = k_f[A] - k_b[B], \frac{d[B]}{dt} = k_b[B] - k_f[A]$

4) $\frac{d[A]}{dt} = -k_f[A] + k_b[B], \frac{d[B]}{dt} = k_f[A] - k_b[B]$

Differential equations from reactions

- Which of these diff. equations describes the following reaction?

$$A + B \underset{k_b}{\overset{k_f}{\longleftrightarrow}} 2A$$

1) $\frac{d[A]}{dt} = k_f[A][B] - 2k_b[A]^2, \frac{d[B]}{dt} = -k_f[A][B]$

2) $\frac{d[A]}{dt} = \frac{1}{2}k_f[A][B] - k_b[A]^2, \frac{d[B]}{dt} = -k_f[A][B]$

3) $\frac{d[A]}{dt} = k_f[A][B] - k_b[A]^2, \frac{d[B]}{dt} = -k_f[A][B]$

4) $\frac{d[A]}{dt} = 2k_f[A][B] - k_b[A]^2, \frac{d[B]}{dt} = -k_f[A][B]$

Differential equations from reactions

- Which of these diff. equations describes the following reaction?

$$A + B \underset{k_b}{\overset{k_f}{\longleftrightarrow}} AB \xrightarrow{k_{cat}} A + C$$

1) $\frac{d[A]}{dt} = -k_f[A][B] + k_b[AB] + k_{cat}[AB], \frac{d[B]}{dt} = -k_f[A][B], \frac{d[AB]}{dt} = k_f[A][B] - k_{cat}[AB], \frac{d[C]}{dt} = k_{cat}[AB]$

2) $\frac{d[A]}{dt} = -k_f[A][B] + k_b[AB] + k_{cat}[AB], \frac{d[B]}{dt} = -k_f[A][B] + k_b[AB], \frac{d[AB]}{dt} = k_f[A][B] - k_b[AB] - k_{cat}[AB], \frac{d[C]}{dt} = k_{cat}[AB]$

3) $\frac{d[A]}{dt} = k_b[AB] + k_{cat}[AB], \frac{d[B]}{dt} = k_b[AB], \frac{d[AB]}{dt} = k_f[A][B] - k_{cat}[AB], \frac{d[C]}{dt} = k_{cat}[AB]$

4) $\frac{d[A]}{dt} = -k_f[A][B] + k_b[AB] + k_{cat}[AB], \frac{d[B]}{dt} = -k_f[A][B] + k_b[AB], \frac{d[AB]}{dt} = -k_b[AB] + k_{cat}[A][C], \frac{d[C]}{dt} = k_{cat}[A][C]$

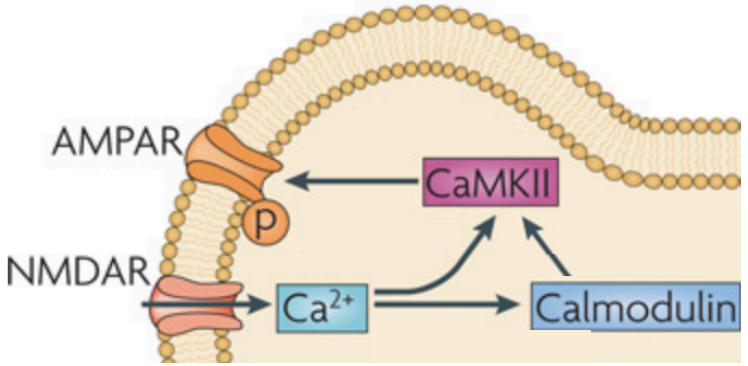
Differential equations from reactions

- Which of these diff. equations describes the following reaction?

$$A + B \underset{k_b}{\overset{k_f}{\longleftrightarrow}} AB \xrightarrow{k_{cat}} A + C$$

2) $\frac{d[A]}{dt} = -k_f[A][B] + k_b[AB] + k_{cat}[AB]$, $\frac{d[B]}{dt} = -k_f[A][B] + k_b[AB]$, $\frac{d[AB]}{dt} = k_f[A][B] - k_b[AB] - k_{cat}[AB]$, $\frac{d[C]}{dt} = k_{cat}[AB]$

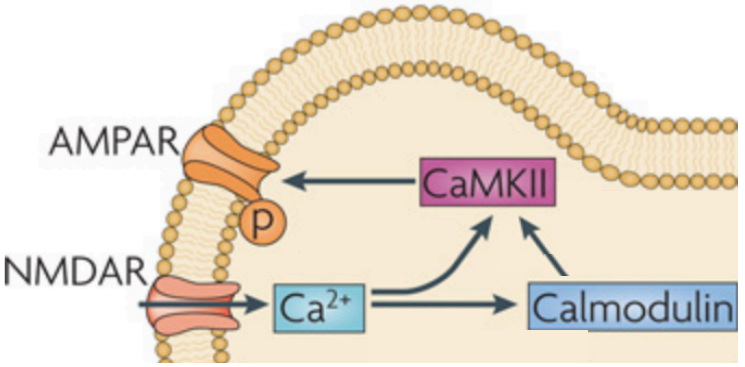
Mass action law in LTP – CaMKII model



- Ca^{2+} enters through NMDA channels

$$\mathrm{Ca}_{(\mathrm{ext})} \longrightarrow \mathrm{Ca}$$

Mass action law in LTP – CaMKII model



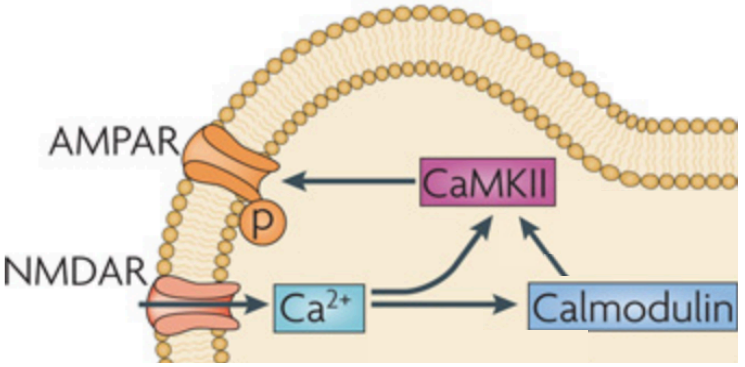
- Ca^{2+} enters through NMDA channels

$$\mathrm{Ca}_{(\mathrm{ext})} \longrightarrow \mathrm{Ca}$$

- Ca^{2+} binds with calmodulin

$$2\mathrm{Ca} + \mathrm{CaM} \rightleftharpoons \mathrm{CaMCa}_2$$
$$2\mathrm{Ca} + \mathrm{CaMCa}_2 \rightleftharpoons \mathrm{CaMCa}_4$$

Mass action law in LTP – CaMKII model



- Ca^{2+} enters through NMDA channels

$$\mathrm{Ca}_{(\mathrm{ext})} \longrightarrow \mathrm{Ca}$$

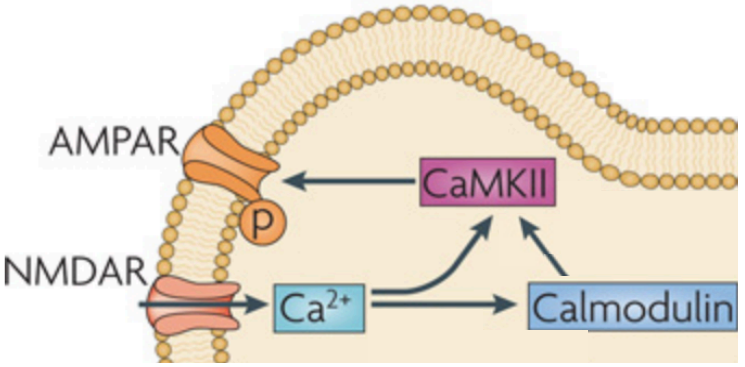
- Ca^{2+} binds with calmodulin

$$2\mathrm{Ca} + \mathrm{CaM} \rightleftharpoons \mathrm{CaMCa}_2$$
$$2\mathrm{Ca} + \mathrm{CaMCa}_2 \rightleftharpoons \mathrm{CaMCa}_4$$

- Ca^{2+}-bound calmodulin binds to CaMKII

$$\mathrm{CaMKII} + \mathrm{CaMCa}_4 \rightleftharpoons \mathrm{CaMKIICaMCa}_4$$

Mass action law in LTP – CaMKII model



- Ca^{2+} enters through NMDA channels

$$\mathrm{Ca}_{(\mathrm{ext})} \longrightarrow \mathrm{Ca}$$

- Ca^{2+} binds with calmodulin

$$2\mathrm{Ca} + \mathrm{CaM} \rightleftharpoons \mathrm{CaMCa}_2$$
$$2\mathrm{Ca} + \mathrm{CaMCa}_2 \rightleftharpoons \mathrm{CaMCa}_4$$

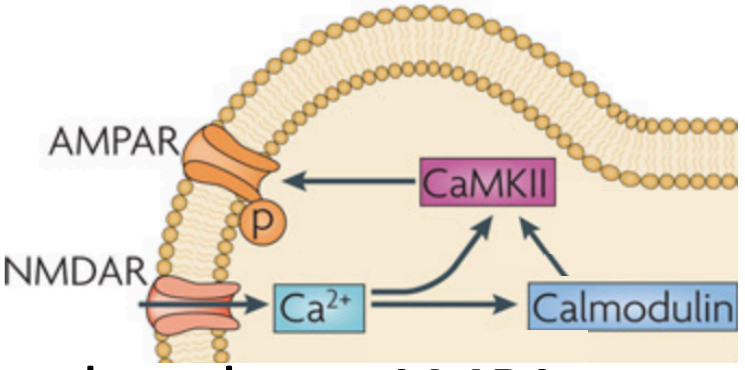
- Ca^{2+}-bound calmodulin binds to CaMKII

$$\mathrm{CaMKII} + \mathrm{CaMCa}_4 \rightleftharpoons \mathrm{CaMKIICaMCa}_4$$

- Calmodulin-bound CaMKII is autophosphorylated

$$\mathrm{CaMKIICaMCa}_4 + \mathrm{CaMKIICaMCa}_4 \rightleftharpoons \mathrm{Complex}$$
$$\mathrm{Complex} + \mathrm{Complex} \rightleftharpoons \mathrm{Complex} + \mathrm{pComplex}$$
$$\mathrm{pCaMKIICaMCa}_4 + \mathrm{CaMKIICaMCa}_4 \rightleftharpoons \mathrm{pComplex}$$

Mass action law in LTP – CaMKII model



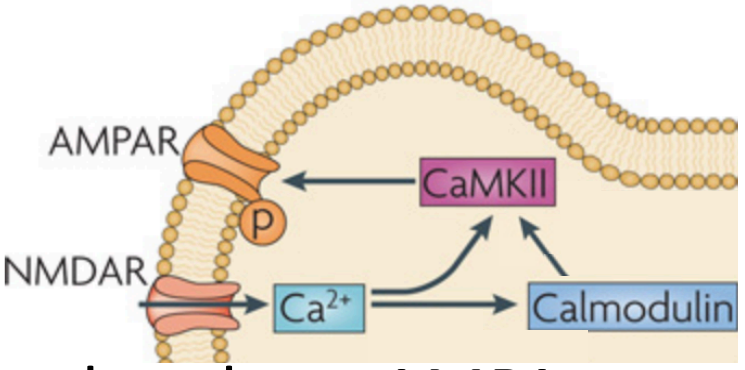
- Phosphorylated calmodulin-bound CaMKII phosphorylates AMPA receptors

$$\mathrm{GluR1} + \mathrm{pCaMKIICaMCa_4} \rightleftharpoons \mathrm{GluR1\_pCaMKIICaMCa_4} \longrightarrow \mathrm{GluR1\_pS831} + \mathrm{pCaMKIICaMCa_4}$$

Jędrzejewska-Szmek, Joanna, et al. "β-adrenergic signaling broadly contributes to LTP induction." *PLoS computational biology* 13.7 (2017): e1005657.

Carroll, Reed C., et al. "Role of AMPA receptor endocytosis in synaptic plasticity." *Nature Reviews Neuroscience* 2.5 (2001): 315.

Mass action law in LTP – CaMKII model



- Phosphorylated calmodulin-bound CaMKII phosphorylates AMPA receptors

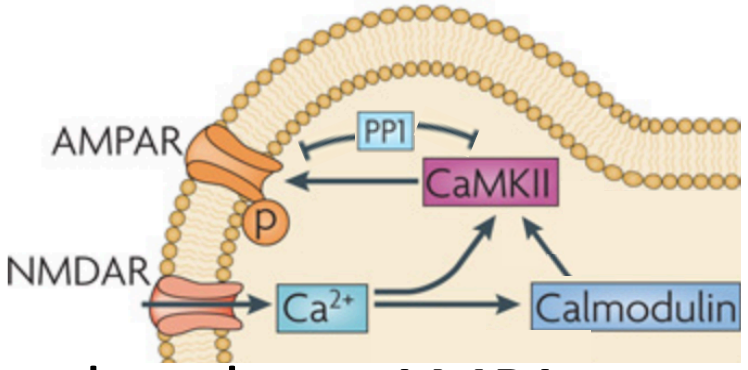
$$\mathrm{GluR1} + \mathrm{pCaMKIICaMCa_4} \rightleftharpoons \mathrm{GluR1\_pCaMKIICaMCa_4} \longrightarrow \mathrm{GluR1\_pS831} + \mathrm{pCaMKIICaMCa_4}$$

- Phosphorylated AMPA receptors have higher ion-channel conductance and are more easily inserted to the membrane

Jędrzejewska-Szmek, Joanna, et al. "β-adrenergic signaling broadly contributes to LTP induction." *PLoS computational biology* 13.7 (2017): e1005657.

Carroll, Reed C., et al. "Role of AMPA receptor endocytosis in synaptic plasticity." *Nature Reviews Neuroscience* 2.5 (2001): 315.

Mass action law in LTP – CaMKII model



- Phosphorylated calmodulin-bound CaMKII phosphorylates AMPA receptors

$$\mathrm{GluR1} + \mathrm{pCaMKIICaMCa_4} \rightleftharpoons \mathrm{GluR1\_pCaMKIICaMCa_4} \longrightarrow \mathrm{GluR1\_pS831} + \mathrm{pCaMKIICaMCa_4}$$

- Phosphorylated AMPA receptors have higher ion-channel conductance and are more easily inserted to the membrane
- AMPA receptors are dephosphorylated by phosphatases such as PP1

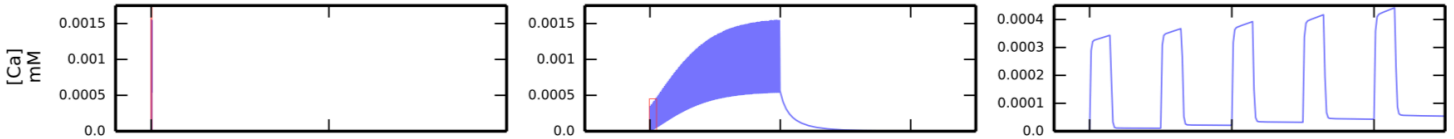
$$\mathrm{GluR1\_pS831} + \mathrm{PP1} \rightleftharpoons \mathrm{GluR1\_pS831\_PP1} \longrightarrow \mathrm{GluR1} + \mathrm{PP1}$$

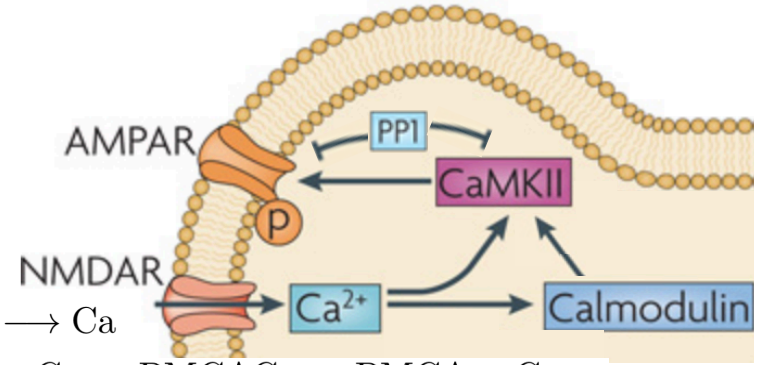
$$\mathrm{pCaMKIICaMCa} + \mathrm{PP1} \rightleftharpoons \mathrm{pCaMKIICaMCa_4PP1} \longrightarrow \mathrm{CaMKIICaMCa_4} + \mathrm{PP1}$$

Jędrzejewska-Szmek, Joanna, et al. "β-adrenergic signaling broadly contributes to LTP induction." *PLoS computational biology* 13.7 (2017): e1005657.

Carroll, Reed C., et al. "Role of AMPA receptor endocytosis in synaptic plasticity." *Nature Reviews Neuroscience* 2.5 (2001): 315.

Mass action law in LTP – CaMKII model





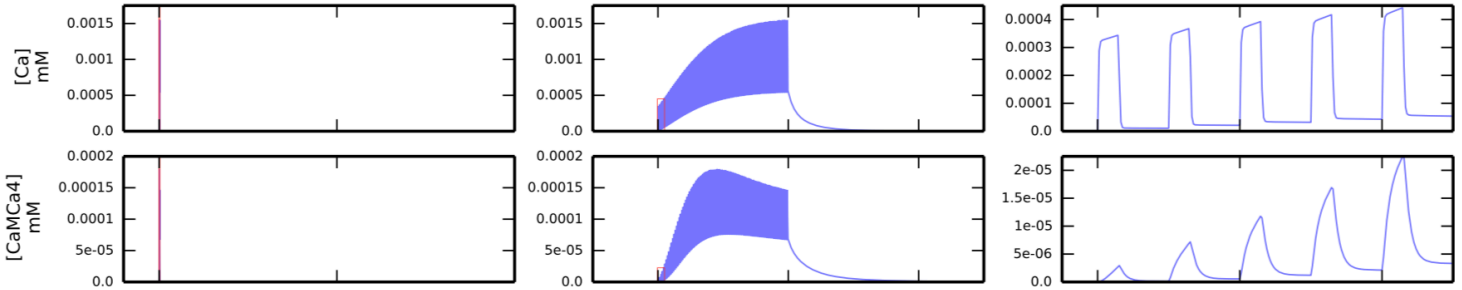
$$\mathrm{Ca}_{(\mathrm{ext})} \longrightarrow \mathrm{Ca}$$

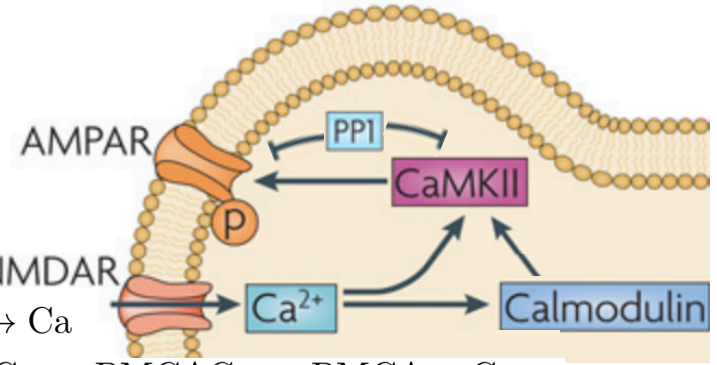
$$\mathrm{PMCA} + \mathrm{Ca} \rightleftharpoons \mathrm{PMCACa} \rightarrow \mathrm{PMCA} + \mathrm{Ca}_{\mathrm{out}}$$

$$\mathrm{ncx} + \mathrm{Ca} \rightleftharpoons \mathrm{ncxCa} \rightarrow \mathrm{ncx} + \mathrm{Ca}_{\mathrm{out}}$$

$$\mathrm{Ca}_{\mathrm{Out}} + \mathrm{Leak} \rightleftharpoons \mathrm{Ca}_{\mathrm{Out}}\mathrm{Leak} \rightarrow \mathrm{Ca} + \mathrm{Leak}$$

Mass action law in LTP – CaMKII model





$$\mathrm{Ca}_{(\mathrm{ext})} \longrightarrow \mathrm{Ca}$$

$$\mathrm{PMCA} + \mathrm{Ca} \rightleftharpoons \mathrm{PMCACa} \rightarrow \mathrm{PMCA} + \mathrm{Ca}_{\mathrm{out}}$$

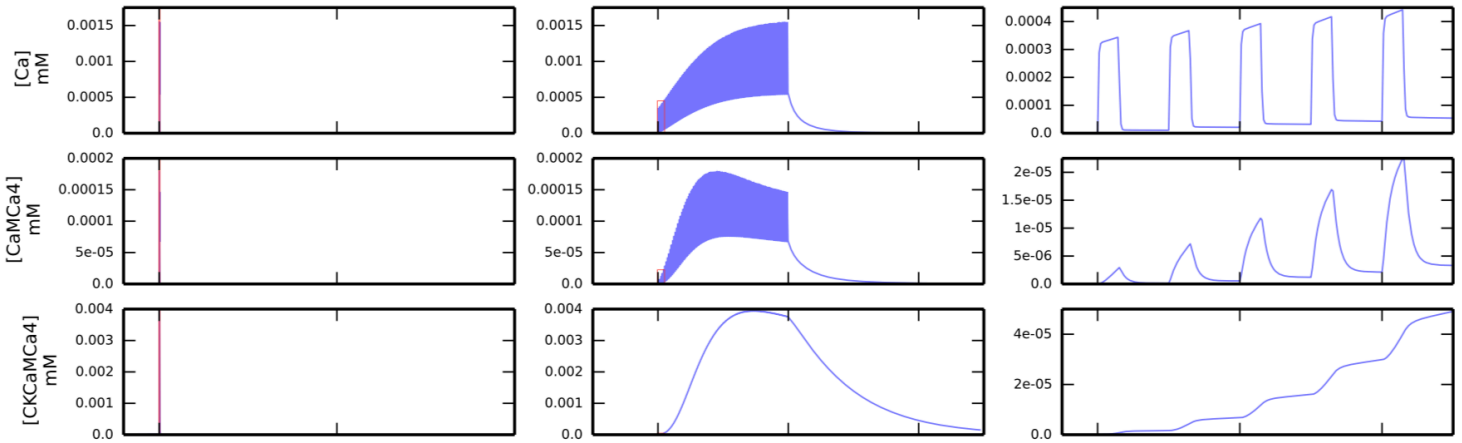
$$\mathrm{ncx} + \mathrm{Ca} \rightleftharpoons \mathrm{ncxCa} \rightarrow \mathrm{ncx} + \mathrm{Ca}_{\mathrm{out}}$$

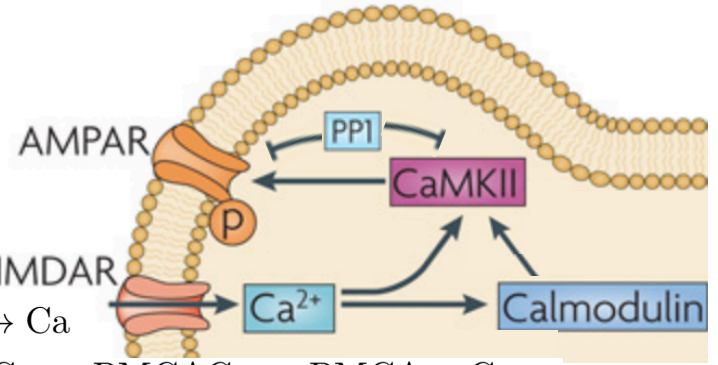
$$\mathrm{Ca}_{\mathrm{Out}} + \mathrm{Leak} \rightleftharpoons \mathrm{Ca}_{\mathrm{Out}}\mathrm{Leak} \rightarrow \mathrm{Ca} + \mathrm{Leak}$$

$$2\mathrm{Ca} + \mathrm{CaM} \rightleftharpoons \mathrm{CaMCa}_2$$

$$2\mathrm{Ca} + \mathrm{CaMCa}_2 \rightleftharpoons \mathrm{CaMCa}_4$$

Mass action law in LTP – CaMKII model





$$\mathrm{Ca}_{(\mathrm{ext})} \longrightarrow \mathrm{Ca}$$

$$\mathrm{PMCA} + \mathrm{Ca} \rightleftharpoons \mathrm{PMCACa} \rightarrow \mathrm{PMCA} + \mathrm{Ca}_{\mathrm{out}}$$

$$\mathrm{ncx} + \mathrm{Ca} \rightleftharpoons \mathrm{ncxCa} \rightarrow \mathrm{ncx} + \mathrm{Ca}_{\mathrm{out}}$$

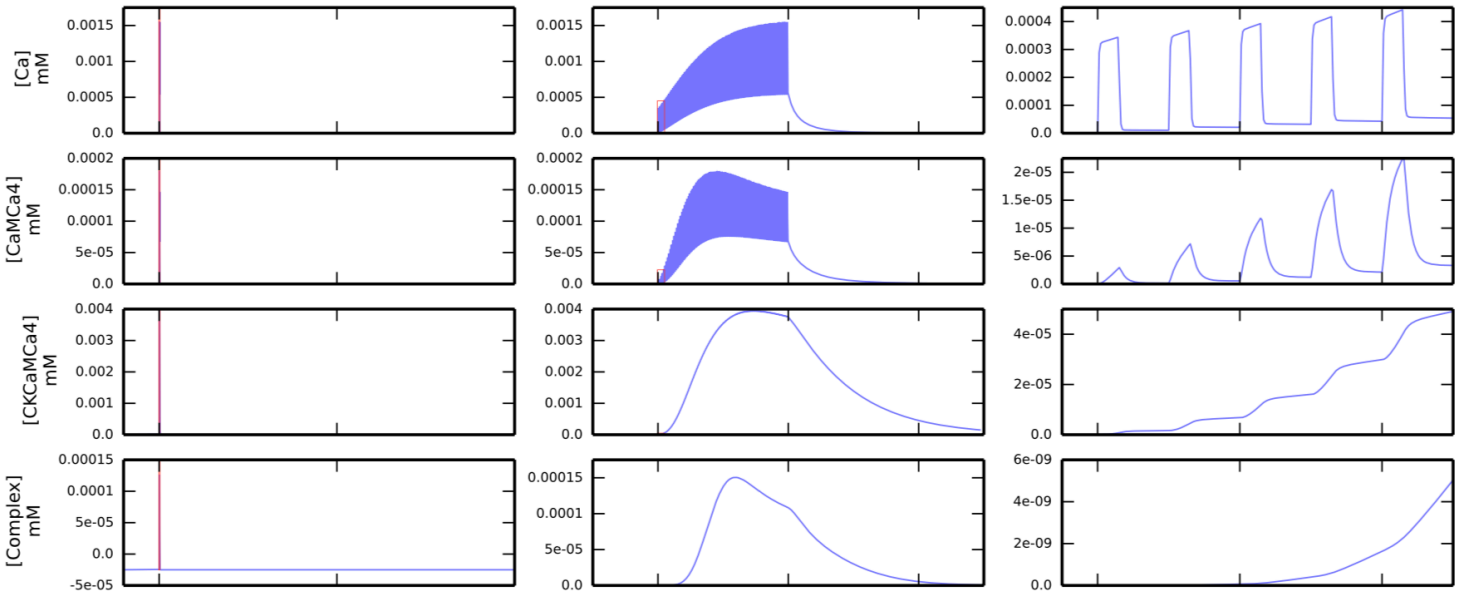
$$\mathrm{Ca}_{\mathrm{Out}} + \mathrm{Leak} \rightleftharpoons \mathrm{Ca}_{\mathrm{Out}}\mathrm{Leak} \rightarrow \mathrm{Ca} + \mathrm{Leak}$$

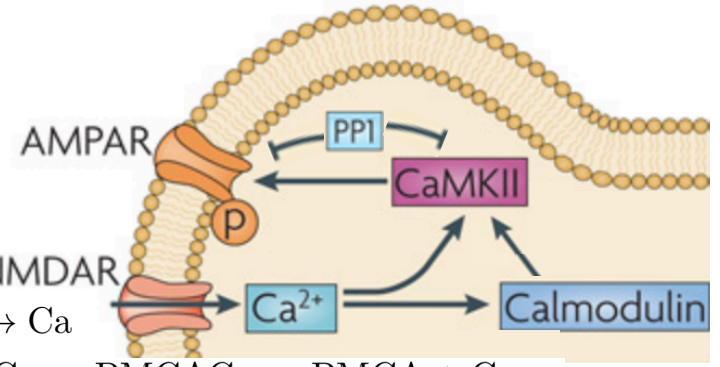
$$2\mathrm{Ca} + \mathrm{CaM} \rightleftharpoons \mathrm{CaMCa}_2$$

$$2\mathrm{Ca} + \mathrm{CaMCa}_2 \rightleftharpoons \mathrm{CaMCa}_4$$

$$\mathrm{CaMKII} + \mathrm{CaMCa}_4 \rightleftharpoons \mathrm{CaMKIICaMCa}_4$$

Mass action law in LTP – CaMKII model





$$\mathrm{Ca}_{(\mathrm{ext})} \longrightarrow \mathrm{Ca}$$

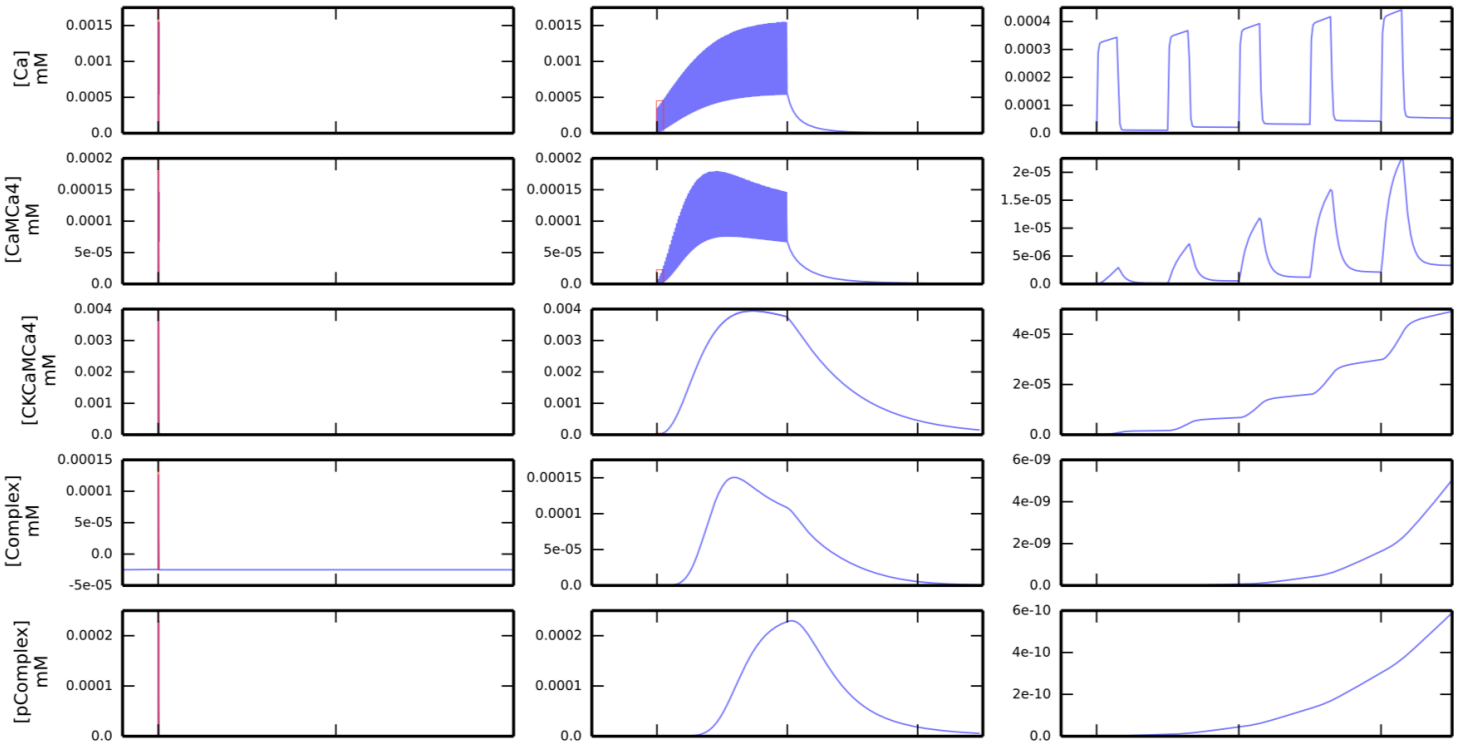
$$\mathrm{PMCA} + \mathrm{Ca} \rightleftharpoons \mathrm{PMCACa} \rightarrow \mathrm{PMCA} + \mathrm{Ca}_{\mathrm{out}}$$
$$\mathrm{ncx} + \mathrm{Ca} \rightleftharpoons \mathrm{ncxCa} \rightarrow \mathrm{ncx} + \mathrm{Ca}_{\mathrm{out}}$$
$$\mathrm{Ca}_{\mathrm{Out}} + \mathrm{Leak} \rightleftharpoons \mathrm{Ca}_{\mathrm{Out}}\mathrm{Leak} \rightarrow \mathrm{Ca} + \mathrm{Leak}$$

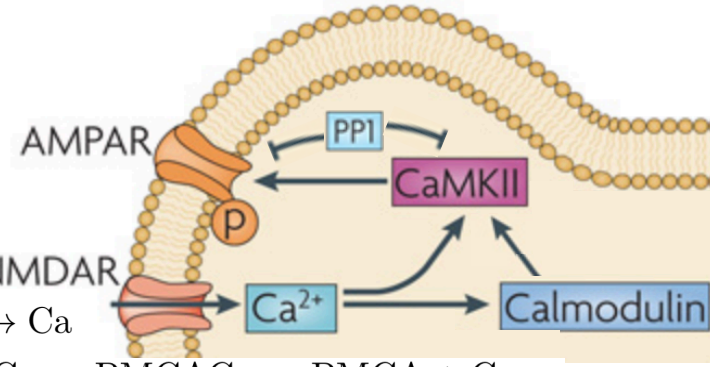
$$2\mathrm{Ca} + \mathrm{CaM} \rightleftharpoons \mathrm{CaMCa}_2$$
$$2\mathrm{Ca} + \mathrm{CaMCa}_2 \rightleftharpoons \mathrm{CaMCa}_4$$

$$\mathrm{CaMKII} + \mathrm{CaMCa}_4 \rightleftharpoons \mathrm{CaMKIICaMCa}_4$$

$$\mathrm{CaMKIICaMCa}_4 + \mathrm{CaMKIICaMCa}_4 \rightleftharpoons \mathrm{Complex}$$

Mass action law in LTP – CaMKII model





$$Ca_{(ext)} \longrightarrow Ca$$

$$PMCA + Ca \rightleftharpoons PMCACa \rightarrow PMCA + Ca_{out}$$
$$ncx + Ca \rightleftharpoons ncxCa \rightarrow ncx + Ca_{out}$$
$$Ca_{Out} + Leak \rightleftharpoons Ca_{Out}Leak \rightarrow Ca + Leak$$

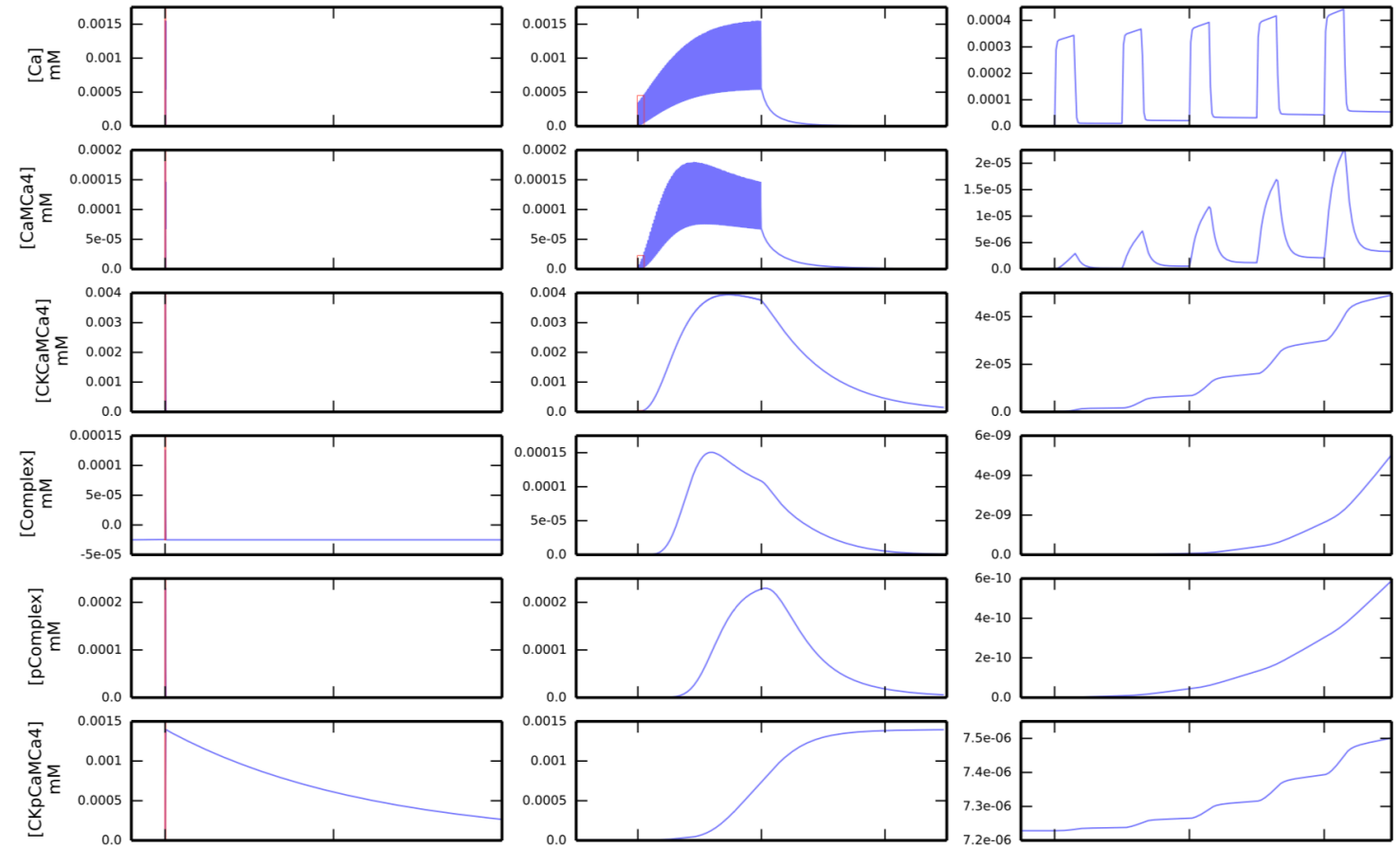
$$2Ca + CaM \rightleftharpoons CaMCa_2$$
$$2Ca + CaMCa_2 \rightleftharpoons CaMCa_4$$

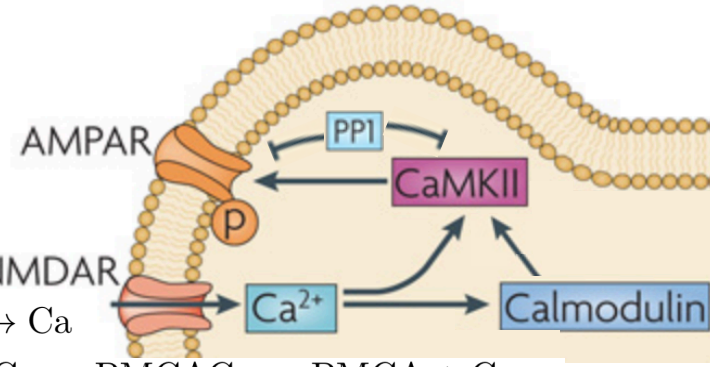
$$CaMKII + CaMCa_4 \rightleftharpoons CaMKIICaMCa_4$$

$$CaMKIICaMCa_4 + CaMKIICaMCa_4 \rightleftharpoons Complex$$

$$Complex + Complex \rightleftharpoons Complex + pComplex$$

Mass action law in LTP – CaMKII model





$$Ca_{(ext)} \longrightarrow Ca$$

$$PMCA + Ca \rightleftharpoons PMCACa \rightarrow PMCA + Ca_{out}$$

$$ncx + Ca \rightleftharpoons ncxCa \rightarrow ncx + Ca_{out}$$

$$Ca_{Out} + Leak \rightleftharpoons Ca_{Out}Leak \rightarrow Ca + Leak$$

$$2Ca + CaM \rightleftharpoons CaMCa_2$$

$$2Ca + CaMCa_2 \rightleftharpoons CaMCa_4$$

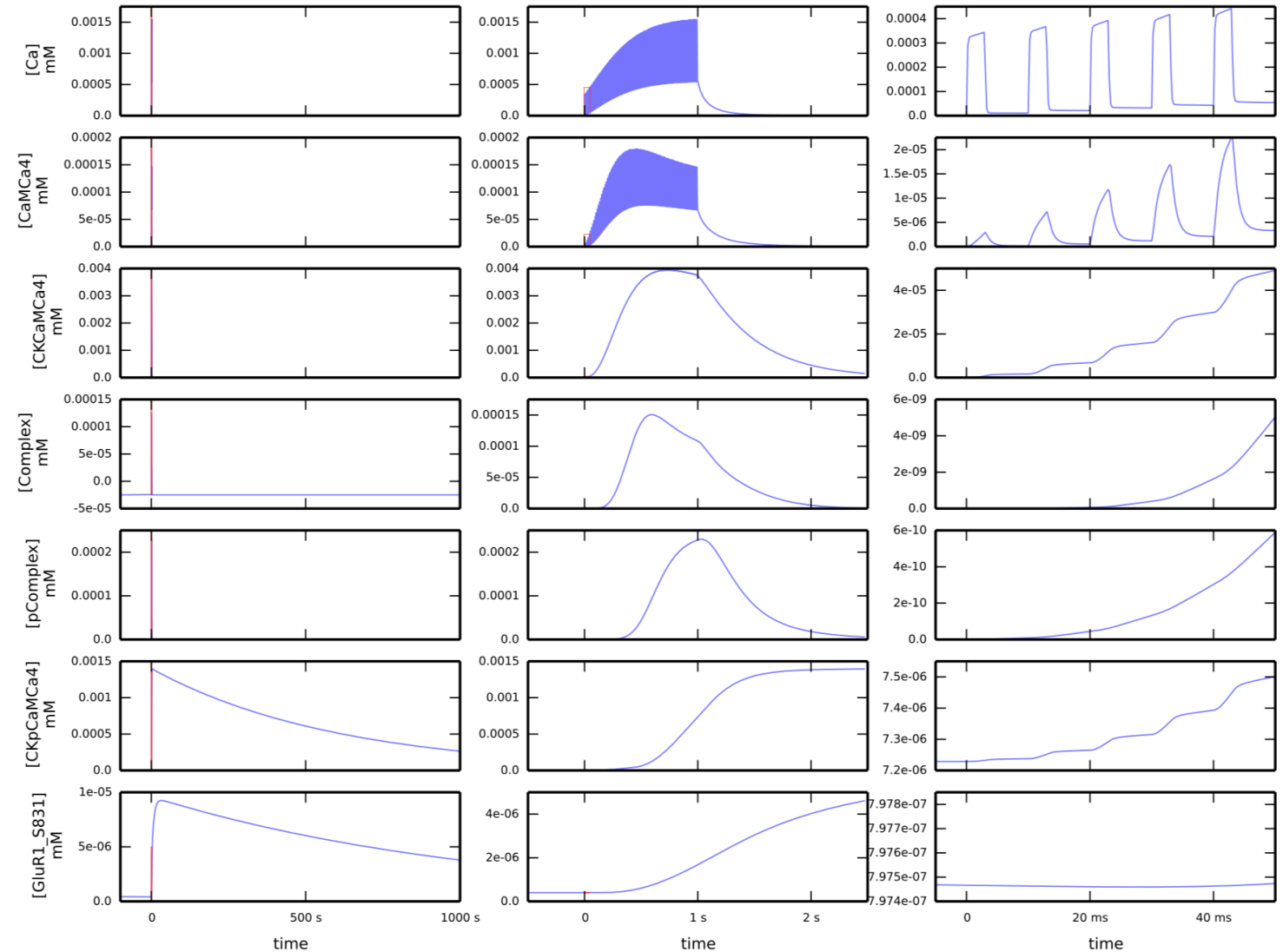
$$CaMKII + CaMCa_4 \rightleftharpoons CaMKIICaMCa_4$$

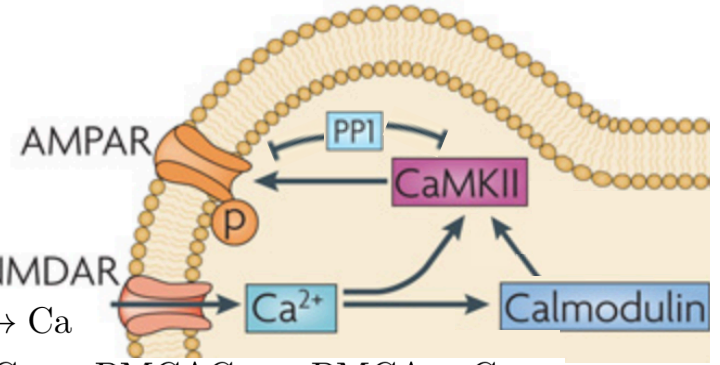
$$CaMKIICaMCa_4 + CaMKIICaMCa_4 \rightleftharpoons Complex$$

$$Complex + Complex \rightleftharpoons Complex + pComplex$$

$$pCaMKIICaMCa_4 + CaMKIICaMCa_4 \rightleftharpoons pComplex$$

Mass action law in LTP – CaMKII model





$$Ca_{(ext)} \longrightarrow Ca$$

$$PMCA + Ca \rightleftharpoons PMCACa \rightarrow PMCA + Ca_{out}$$
$$ncx + Ca \rightleftharpoons ncxCa \rightarrow ncx + Ca_{out}$$
$$Ca_{Out} + Leak \rightleftharpoons Ca_{Out}Leak \rightarrow Ca + Leak$$

$$2Ca + CaM \rightleftharpoons CaMCa_2$$
$$2Ca + CaMCa_2 \rightleftharpoons CaMCa_4$$

$$CaMKII + CaMCa_4 \rightleftharpoons CaMKIICaMCa_4$$

$$CaMKIICaMCa_4 + CaMKIICaMCa_4 \rightleftharpoons Complex$$

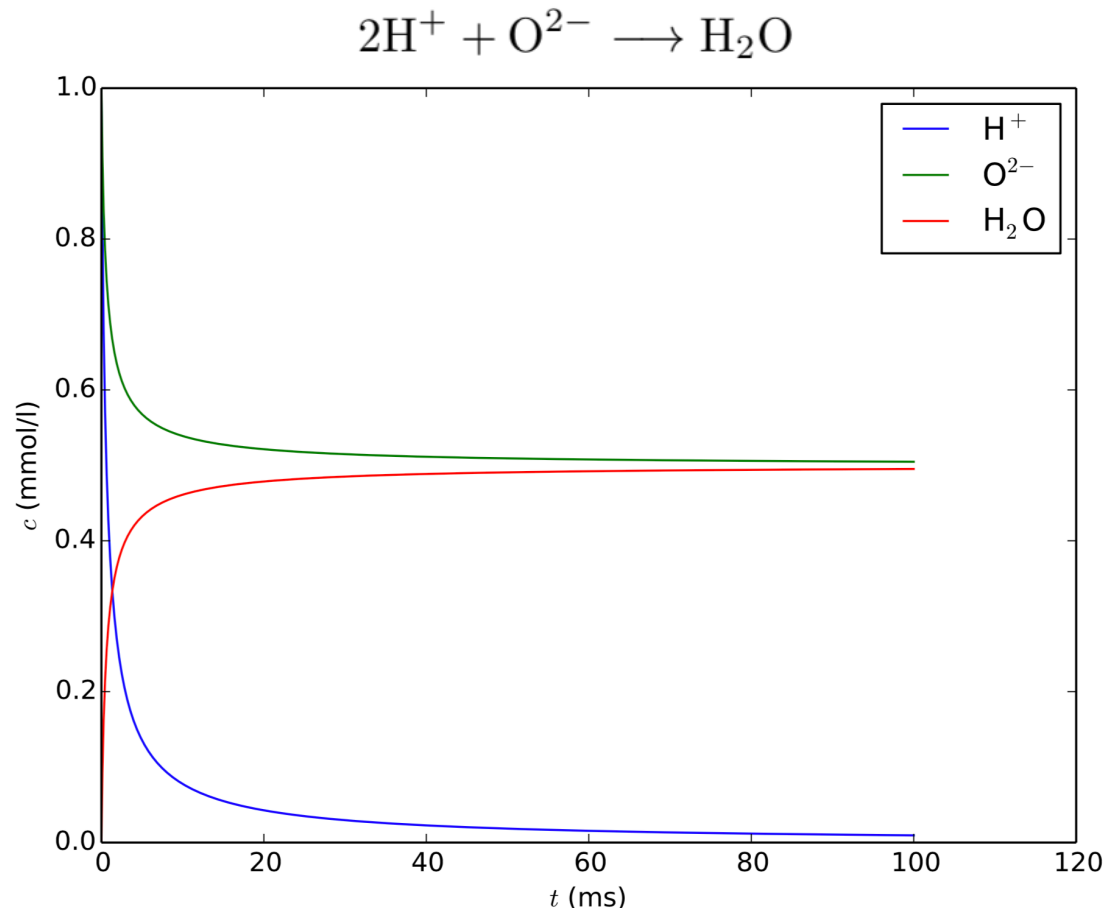
$$Complex + Complex \rightleftharpoons Complex + pComplex$$

$$pCaMKIICaMCa_4 + CaMKIICaMCa_4 \rightleftharpoons pComplex$$

$$GluR1 + pCaMKIICaMCa_4 \rightleftharpoons GluR1\_pCaMKIICaMCa_4 \longrightarrow GluR1\_pS831 + pCaMKIICaMCa_4$$
$$GluR1\_pS831 + PP1 \rightleftharpoons GluR1\_pS831\_PP1 \longrightarrow GluR1 + PP1$$

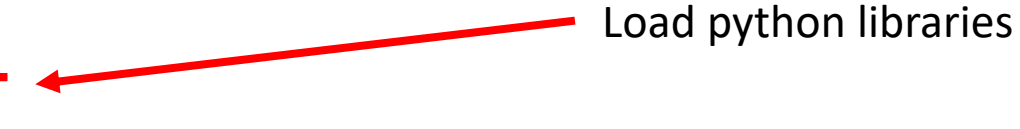
Examples for NEURON's RxD extension

```
from neuron import h, rxd
import time
from pylab import *

h.load_file('stdrun.hoc')

dend = h.Section()
r = rxd.Region(h.allsec())

hydrogen = rxd.Species(r, name='hydrogen',charge=1,initial=1)
oxygen = rxd.Species(r, name='oxygen',charge=-2,initial=1)
water = rxd.Species(r, name='water',charge=0,initial=0)

reaction = rxd.Reaction(2 * hydrogen + oxygen > water, 1)

vec_t = h.Vector()
vec_H = h.Vector()
vec_O = h.Vector()
vec_H2O = h.Vector()
vec_t.record(h._ref_t)
vec_H.record(hydrogen.nodes(dend)(0.5)[0].._ref_concentration)
vec_O.record(oxygen.nodes(dend)(0.5)[0].._ref_concentration)
vec_H2O.record(water.nodes(dend)(0.5)[0].._ref_concentration)

h.finitialize()
timenow = time.time()
h.continuerun(100)
print('Simulation done in '+str(time.time()-timenow)+' seconds')

f,ax = subplots(1,1)
ax.plot(array(vec_t),array(vec_H),label='H$^+$')
ax.plot(array(vec_t),array(vec_O),label='O$^{2-}$')
ax.plot(array(vec_t),array(vec_H2O),label='H$_2$O')
ax.legend()
ax.set_xlabel('$t$ (ms)')
ax.set_ylabel('$c$ (nmol/l)')

f.savefig('H2O.eps')
```



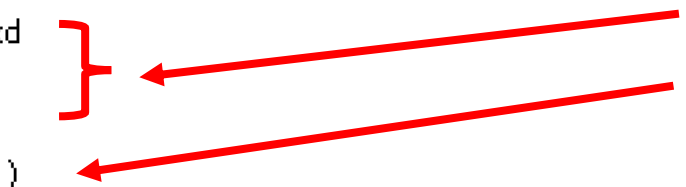
Examples for NEURON's RxD extension

```
from neuron import h, rxd
import time
from pylab import *

h.load_file('stdrun.hoc')

dend = h.Section()
r = rxd.Region(h.allsec())

hydrogen = rxd.Species(r, name='hydrogen',charge=1,initial=1)
oxygen = rxd.Species(r, name='oxygen',charge=-2,initial=1)
water = rxd.Species(r, name='water',charge=0,initial=0)

reaction = rxd.Reaction(2 * hydrogen + oxygen > water, 1)
```



Examples for NEURON's RxD extension

```
from neuron import h, rxd
import time
from pylab import *

h.load_file('stdrun.hoc')

dend = h.Section()
r = rxd.Region(h.allsec())

hydrogen = rxd.Species(r, name='hydrogen',charge=1,initial=1)
oxygen = rxd.Species(r, name='oxygen',charge=-2,initial=1)
water = rxd.Species(r, name='water',charge=0,initial=0)

reaction = rxd.Reaction(2 * hydrogen + oxygen > water, 1)
```

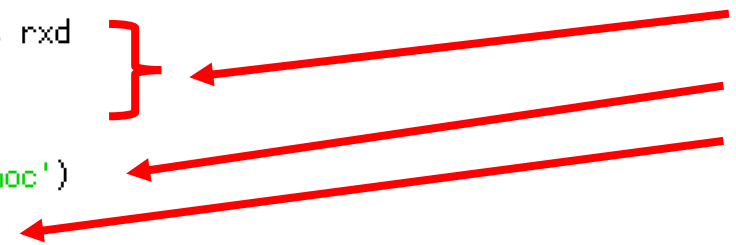


Load python libraries

Load standard hoc functions that may be useful

Examples for NEURON's RxD extension

```
from neuron import h, rxd
import time
from pylab import *

h.load_file('stdrun.hoc')

dend = h.Section()
r = rxd.Region(h.allsec())

hydrogen = rxd.Species(r, name='hydrogen',charge=1,initial=1)
oxygen = rxd.Species(r, name='oxygen',charge=-2,initial=1)
water = rxd.Species(r, name='water',charge=0,initial=0)

reaction = rxd.Reaction(2 * hydrogen + oxygen > water, 1)
```



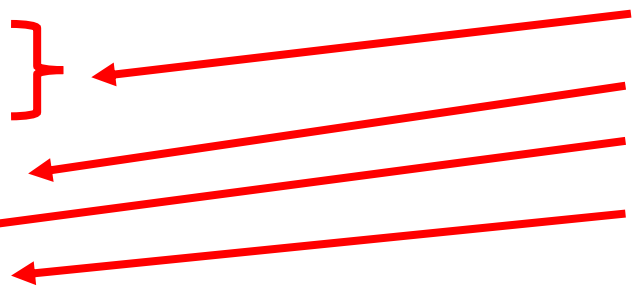
Load python libraries

Load standard hoc functions that may be useful

Create a compartment called 'dend'

Examples for NEURON's RxD extension

```
from neuron import h, rxd
import time
from pylab import *

h.load_file('stdrun.hoc')

dend = h.Section()
r = rxd.Region(h.allsec())

hydrogen = rxd.Species(r, name='hydrogen',charge=1,initial=1)
oxygen = rxd.Species(r, name='oxygen',charge=-2,initial=1)
water = rxd.Species(r, name='water',charge=0,initial=0)

reaction = rxd.Reaction(2 * hydrogen + oxygen > water, 1)
```



Load python libraries

Load standard hoc functions that may be useful

Create a compartment called 'dend'

Create a region where reactions occur (include all sections, i.e. dend)

Examples for NEURON's RxD extension

```
from neuron import h, rxd
import time
from pylab import *

h.load_file('stdrun.hoc')

dend = h.Section()
r = rxd.Region(h.allsec())

hydrogen = rxd.Species(r, name='hydrogen',charge=1,initial=1)
oxygen = rxd.Species(r, name='oxygen',charge=-2,initial=1)
water = rxd.Species(r, name='water',charge=0,initial=0)

reaction = rxd.Reaction(2 * hydrogen + oxygen > water, 1)
```

Load python libraries

Load standard hoc functions that may be useful

Create a compartment called 'dend'

Create a region where reactions occur (include all sections, i.e. dend)

Create species for region r. Set initial concentrations 1 mM (H^+), 1 mM (O^{2-}), and 0 mM (H_2O)

Examples for NEURON's RxD extension

```
from neuron import h, rxd
import time
from pylab import *

h.load_file('stdrun.hoc')

dend = h.Section()
r = rxd.Region(h.allsec())

hydrogen = rxd.Species(r, name='hydrogen',charge=1,initial=1)
oxygen = rxd.Species(r, name='oxygen',charge=-2,initial=1)
water = rxd.Species(r, name='water',charge=0,initial=0)

reaction = rxd.Reaction(2 * hydrogen + oxygen > water, 1)
```

Load python libraries

Load standard hoc functions that may be useful

Create a compartment called 'dend'

Create a region where reactions occur (include all sections, i.e. dend)

Create species for region r. Set initial concentrations 1 mM (H^+), 1 mM (O^{2-}), and 0 mM (H_2O)

Create reaction $2H^+ + O^{2-} \longrightarrow H_2O$ with rate constant 1 $(mM)^2/ms$

Examples for NEURON's RxD extension

```
vec_t = h.Vector()
vec_H = h.Vector()
vec_O = h.Vector()
vec_H2O = h.Vector()
vec_t.record(h._ref_t)
vec_H.record(hydrogen.nodes(dend)(0.5)[0]._ref_concentration)
vec_O.record(oxygen.nodes(dend)(0.5)[0]._ref_concentration)
vec_H2O.record(water.nodes(dend)(0.5)[0]._ref_concentration)

h.finitialize()
timenow = time.time()
h.continuerun(100)
print('Simulation done in '+str(time.time()-timenow)+' seconds')

f,ax = subplots(1,1)
ax.plot(array(vec_t),array(vec_H),label='H$^+$')
ax.plot(array(vec_t),array(vec_O),label='O$^{2-}$')
ax.plot(array(vec_t),array(vec_H2O),label='H$_2$O')
ax.legend()
ax.set_xlabel('$t$ (ms)')
ax.set_ylabel('$c$ (mmol/l)')

f.savefig('H2O.eps')
```

Create NEURON vectors

Examples for NEURON's RxD extension

```
vec_t = h.Vector()
vec_H = h.Vector()
vec_O = h.Vector()
vec_H2O = h.Vector()
vec_t.record(h._ref_t)
vec_H.record(hydrogen.nodes(dend)(0.5)[0]._ref_concentration)
vec_O.record(oxygen.nodes(dend)(0.5)[0]._ref_concentration)
vec_H2O.record(water.nodes(dend)(0.5)[0]._ref_concentration)

h.finitialize()
timenow = time.time()
h.continuerun(100)
print('Simulation done in '+str(time.time()-timenow)+' seconds')

f,ax = subplots(1,1)
ax.plot(array(vec_t),array(vec_H),label='H$^+$')
ax.plot(array(vec_t),array(vec_O),label='O$^{2-}$')
ax.plot(array(vec_t),array(vec_H2O),label='H$_2$O')
ax.legend()
ax.set_xlabel('$t$ (ms)')
ax.set_ylabel('$c$ (mmol/l)')

f.savefig('H2O.eps')
```

Create NEURON vectors

Record to the time to vector vec_t

Examples for NEURON's RxD extension

```
vec_t = h.Vector()
vec_H = h.Vector()
vec_O = h.Vector()
vec_H2O = h.Vector()
vec_t.record(h._ref_t)
vec_H.record(hydrogen.nodes(dend)(0.5)[0]._ref_concentration)
vec_O.record(oxygen.nodes(dend)(0.5)[0]._ref_concentration)
vec_H2O.record(water.nodes(dend)(0.5)[0]._ref_concentration)

h.finitialize()
timenow = time.time()
h.continuerun(100)
print('Simulation done in '+str(time.time()-timenow)+' seconds')

f,ax = subplots(1,1)
ax.plot(array(vec_t),array(vec_H),label='H$^+$')
ax.plot(array(vec_t),array(vec_O),label='O$^{2-}$')
ax.plot(array(vec_t),array(vec_H2O),label='H$_2$O')
ax.legend()
ax.set_xlabel('$t$ (ms)')
ax.set_ylabel('$c$ (mmol/l)')

f.savefig('H2O.eps')
```

Create NEURON vectors

Record to the time to vector vec_t

Record the concentrations of $[H^+]$, $[O^{2-}]$, and $[H_2O]$

Examples for NEURON's RxD extension

```
vec_t = h.Vector()
vec_H = h.Vector()
vec_O = h.Vector()
vec_H2O = h.Vector()
vec_t.record(h._ref_t)
vec_H.record(hydrogen.nodes(dend)(0.5)[0]._ref_concentration)
vec_O.record(oxygen.nodes(dend)(0.5)[0]._ref_concentration)
vec_H2O.record(water.nodes(dend)(0.5)[0]._ref_concentration)

h.finitialize()
timenow = time.time()
h.continuerun(100)
print('Simulation done in '+str(time.time()-timenow)+' seconds')

f,ax = subplots(1,1)
ax.plot(array(vec_t),array(vec_H),label='H$^+$')
ax.plot(array(vec_t),array(vec_O),label='O$^{2-}$')
ax.plot(array(vec_t),array(vec_H2O),label='H$_2$O')
ax.legend()
ax.set_xlabel('$t$ (ms)')
ax.set_ylabel('$c$ (mmol/l)')

f.savefig('H2O.eps')
```

Create NEURON vectors

Record to the time to vector vec_t

Record the concentrations of $[H^+]$, $[O^{2-}]$, and $[H_2O]$

Initialize, set the timer and run for 100 ms

Examples for NEURON's RxD extension

```
vec_t = h.Vector()
vec_H = h.Vector()
vec_O = h.Vector()
vec_H2O = h.Vector()
vec_t.record(h._ref_t)
vec_H.record(hydrogen.nodes(dend)(0.5)[0]._ref_concentration)
vec_O.record(oxygen.nodes(dend)(0.5)[0]._ref_concentration)
vec_H2O.record(water.nodes(dend)(0.5)[0]._ref_concentration)

h.finitialize()
timenow = time.time()
h.continuerun(100)
print('Simulation done in '+str(time.time()-timenow)+' seconds')

f,ax = subplots(1,1)
ax.plot(array(vec_t),array(vec_H),label='H$^+$')
ax.plot(array(vec_t),array(vec_O),label='O$^{2-}$')
ax.plot(array(vec_t),array(vec_H2O),label='H$_2$O')
ax.legend()
ax.set_xlabel('$t$ (ms)')
ax.set_ylabel('$c$ (mmol/l)')

f.savefig('H2O.eps')
```

Create NEURON vectors

Record to the time to vector vec_t

Record the concentrations of $[H^+]$, $[O^{2-}]$, and $[H_2O]$

Initialize, set the timer and run for 100 ms

Create an axis, plot the figure and print it into an eps file

Examples for NEURON's RxD extension

```
vec_t = h.Vector()
vec_H = h.Vector()
vec_O = h.Vector()
vec_H2O = h.Vector()
vec_t.record(h._ref_t)
vec_H.record(hydrogen.nodes(dend)(0.5)[0]._ref_concentration)
vec_O.record(oxygen.nodes(dend)(0.5)[0]._ref_concentration)
vec_H2O.record(water.nodes(dend)(0.5)[0]._ref_concentration)

h.finitialize()
timenow = time.time()
h.continuerun(100)
print('Simulation done in '+str(time.time()-timenow)+' seconds')

f,ax = subplots(1,1)
ax.plot(array(vec_t),array(vec_H),label='H$^+$')
ax.plot(array(vec_t),array(vec_O),label='O$^{2-}$')
ax.plot(array(vec_t),array(vec_H2O),label='H$_2$O')
ax.legend()
ax.set_xlabel('$t$ (ms)')
ax.set_ylabel('$c$ (mmol/l)')

f.savefig('H2O.eps')
```

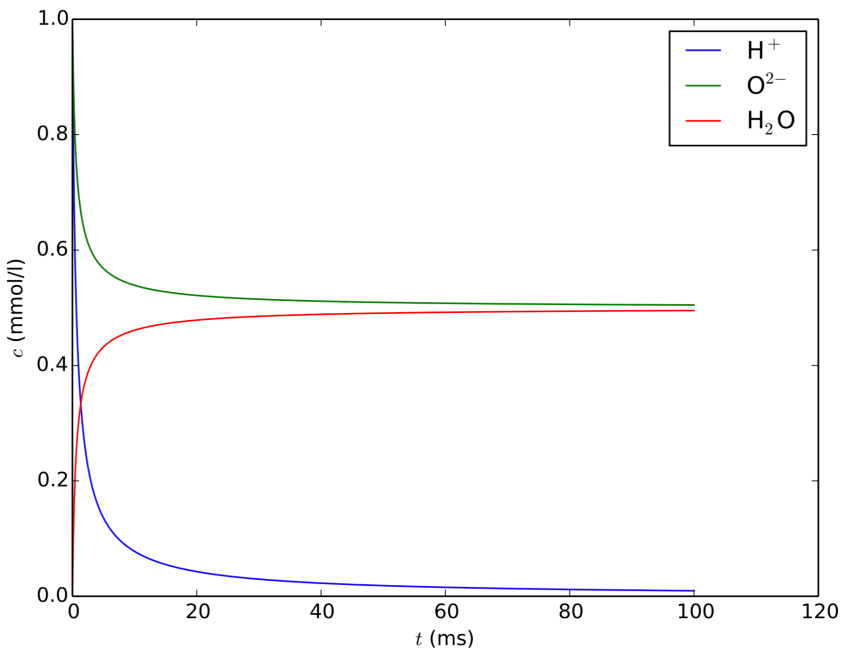
Create NEURON vectors

Record to the time to vector vec_t

Record the concentrations of $[H^+]$, $[O^{2-}]$, and $[H_2O]$

Initialize, set the timer and run for 100 ms

Create an axis, plot the figure and print it into an eps file



Examples for NEURON's RxD extension

```
Ca_flux_rate = rxd.Parameter(cyt, initial=0)
reaction_Ca_flux = rxd.Rate(spec_Ca, Ca_flux_rate)
```

$$\mathrm{Ca}_{(\mathrm{ext})} \longrightarrow \mathrm{Ca}$$

```
T = 1000./Ca_input_freq
tnow = 0
for itrain in range(0,Ntrains):
    for istim in range(0,Ca_input_N):
        tnew = Ca_input_onset + istim*T + trainT*itrain
        h.cvode.event(tnew, lambda: set_param(Ca_flux_rate, Ca_input_flux/6.022e23/my_volume*1e3))
        h.cvode.event(tnew+Ca_input_dur, lambda: set_param(Ca_flux_rate, 0))
```

```
reaction000 = rxd.Reaction(spec_Ca + spec_pmca <> spec_pmcaCa, 50.0, 0.007)
reaction001 = rxd.Reaction(spec_pmcaCa > spec_pmca + spec_CaOut, 0.0035)
reaction002 = rxd.Reaction(spec_Ca + spec_ncx <> spec_ncxCa, 16.8, 0.0112)
reaction003 = rxd.Reaction(spec_ncxCa > spec_ncx + spec_CaOut, 0.0056)
reaction004 = rxd.Reaction(spec_CaOut + spec_Leak <> spec_CaOutLeak, 1.5, 0.0011)
reaction005 = rxd.Reaction(spec_CaOutLeak > spec_Ca + spec_Leak, 0.0011)

reaction067 = rxd.Reaction(spec_CaM + spec_Ca**2 <> spec_CaMCa2, 6.0*spec_CaM*spec_Ca, 0.0091*spec_CaMCa2, custom_dynamics=True)
reaction068 = rxd.Reaction(spec_CaMCa2 + spec_Ca**2 <> spec_CaMCa4, 100.0*spec_CaMCa2*spec_Ca, 1.0*spec_CaMCa4, custom_dynamics=True)
reaction075 = rxd.Reaction(spec_CaMCa4 + spec_CK <> spec_CKCaMCa4, 10.0, 0.003)
```

Examples for NEURON's RxD extension

```
Ca_flux_rate = rxd.Parameter(cyt, initial=0)
reaction_Ca_flux = rxd.Rate(spec_Ca, Ca_flux_rate)
```

$$\mathrm{Ca}_{(\mathrm{ext})} \longrightarrow \mathrm{Ca}$$

```
T = 1000./Ca_input_freq
tnow = 0
for itrain in range(0,Ntrains):
    for istim in range(0,Ca_input_N):
        tnew = Ca_input_onset + istim*T + trainT*itrain
        h.cvode.event(tnew, lambda: set_param(Ca_flux_rate, Ca_input_flux/6.022e23/my_volume*1e3))
        h.cvode.event(tnew+Ca_input_dur, lambda: set_param(Ca_flux_rate, 0))
```

```
reaction000 = rxd.Reaction(spec_Ca + spec_pmca <> spec_pmcaCa, 50.0, 0.007)
reaction001 = rxd.Reaction(spec_pmcaCa > spec_pmca + spec_CaOut, 0.0035)
reaction002 = rxd.Reaction(spec_Ca + spec_ncx <> spec_ncxCa, 16.8, 0.0112)
reaction003 = rxd.Reaction(spec_ncxCa > spec_ncx + spec_CaOut, 0.0056)
reaction004 = rxd.Reaction(spec_CaOut + spec_Leak <> spec_CaOutLeak, 1.5, 0.0011)
reaction005 = rxd.Reaction(spec_CaOutLeak > spec_Ca + spec_Leak, 0.0011)

reaction067 = rxd.Reaction(spec_CaM + spec_Ca**2 <> spec_CaMCa2, 6.0*spec_CaM*spec_Ca, 0.0091*spec_CaMCa2, custom_dynamics=True)
reaction068 = rxd.Reaction(spec_CaMCa2 + spec_Ca**2 <> spec_CaMCa4, 100.0*spec_CaMCa2*spec_Ca, 1.0*spec_CaMCa4, custom_dynamics=True)
reaction075 = rxd.Reaction(spec_CaMCa4 + spec_CK <> spec_CKCaMCa4, 10.0, 0.003)
```

Examples for NEURON's RxD extension

```
Ca_flux_rate = rxd.Parameter(cyt, initial=0)
reaction_Ca_flux = rxd.Rate(spec_Ca, Ca_flux_rate)
```

$$\mathrm{Ca}_{(\mathrm{ext})} \longrightarrow \mathrm{Ca}$$

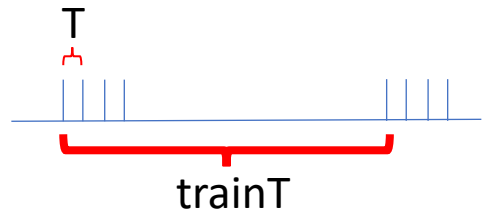
```
T = 1000./Ca_input_freq
tnow = 0
for itrain in range(0,Ntrains):
    for istim in range(0,Ca_input_N):
        tnew = Ca_input_onset + istim*T + trainT*itrain
        h.cvode.event(tnew, lambda: set_param(Ca_flux_rate, Ca_input_flux/6.022e23/my_volume*1e3))
        h.cvode.event(tnew+Ca_input_dur, lambda: set_param(Ca_flux_rate, 0))
```

```
reaction000 = rxd.Reaction(spec_Ca + spec_pmca <> spec_pmcaCa, 50.0, 0.007)
reaction001 = rxd.Reaction(spec_pmcaCa > spec_pmca + spec_CaOut, 0.0035)
reaction002 = rxd.Reaction(spec_Ca + spec_ncx <> spec_ncxCa, 16.8, 0.0112)
reaction003 = rxd.Reaction(spec_ncxCa > spec_ncx + spec_CaOut, 0.0056)
reaction004 = rxd.Reaction(spec_CaOut + spec_Leak <> spec_CaOutLeak, 1.5, 0.0011)
reaction005 = rxd.Reaction(spec_CaOutLeak > spec_Ca + spec_Leak, 0.0011)

reaction067 = rxd.Reaction(spec_CaM + spec_Ca**2 <> spec_CaMCa2, 6.0*spec_CaM*spec_Ca, 0.0091*spec_CaMCa2, custom_dynamics=True)
reaction068 = rxd.Reaction(spec_CaMCa2 + spec_Ca**2 <> spec_CaMCa4, 100.0*spec_CaMCa2*spec_Ca, 1.0*spec_CaMCa4, custom_dynamics=True)
reaction075 = rxd.Reaction(spec_CaMCa4 + spec_CK <> spec_CKCaMCa4, 10.0, 0.003)
```

Examples for NEURON's RxD extension

```
Ca_flux_rate = rxd.Parameter(cyt, initial=0)
reaction_Ca_flux = rxd.Rate(spec_Ca, Ca_flux_rate)
```

$$\mathrm{Ca}_{(\mathrm{ext})} \longrightarrow \mathrm{Ca}$$



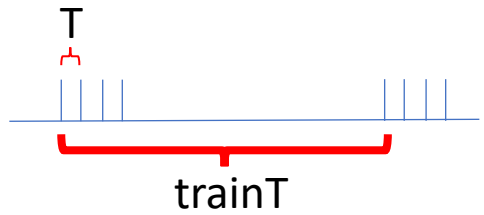

```
T = 1000./Ca_input_freq
tnow = 0
for itrain in range(0,Ntrains):
    for istim in range(0,Ca_input_N):
        tnew = Ca_input_onset + istim*T + trainT*itrain
        h.cvode.event(tnew, lambda: set_param(Ca_flux_rate, Ca_input_flux/6.022e23/my_volume*1e3))
        h.cvode.event(tnew+Ca_input_dur, lambda: set_param(Ca_flux_rate, 0))
```

```
reaction000 = rxd.Reaction(spec_Ca + spec_pmca <> spec_pmcaCa, 50.0, 0.007)
reaction001 = rxd.Reaction(spec_pmcaCa > spec_pmca + spec_CaOut, 0.0035)
reaction002 = rxd.Reaction(spec_Ca + spec_ncx <> spec_ncxCa, 16.8, 0.0112)
reaction003 = rxd.Reaction(spec_ncxCa > spec_ncx + spec_CaOut, 0.0056)
reaction004 = rxd.Reaction(spec_CaOut + spec_Leak <> spec_CaOutLeak, 1.5, 0.0011)
reaction005 = rxd.Reaction(spec_CaOutLeak > spec_Ca + spec_Leak, 0.0011)

reaction067 = rxd.Reaction(spec_CaM + spec_Ca**2 <> spec_CaMCa2, 6.0*spec_CaM*spec_Ca, 0.0091*spec_CaMCa2, custom_dynamics=True)
reaction068 = rxd.Reaction(spec_CaMCa2 + spec_Ca**2 <> spec_CaMCa4, 100.0*spec_CaMCa2*spec_Ca, 1.0*spec_CaMCa4, custom_dynamics=True)
reaction075 = rxd.Reaction(spec_CaMCa4 + spec_CK <> spec_CKCaMCa4, 10.0, 0.003)
```

Examples for NEURON's RxD extension

```
Ca_flux_rate = rxd.Parameter(cyt, initial=0)
reaction_Ca_flux = rxd.Rate(spec_Ca, Ca_flux_rate)
```

$$\mathrm{Ca}_{(\mathrm{ext})} \longrightarrow \mathrm{Ca}$$




```
T = 1000./Ca_input_freq
tnow = 0
for itrain in range(0,Ntrains):
    for istim in range(0,Ca_input_N):
        tnew = Ca_input_onset + istim*T + trainT*itrain
        h.cvode.event(tnew, lambda: set_param(Ca_flux_rate, Ca_input_flux/6.022e23/my_volume*1e3))
        h.cvode.event(tnew+Ca_input_dur, lambda: set_param(Ca_flux_rate, 0))
```

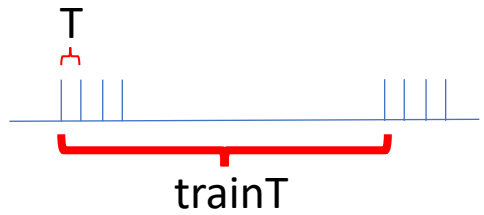
```
reaction000 = rxd.Reaction(spec_Ca + spec_pmca <> spec_pmcaCa, 50.0, 0.007)
reaction001 = rxd.Reaction(spec_pmcaCa > spec_pmca + spec_CaOut, 0.0035)
reaction002 = rxd.Reaction(spec_Ca + spec_ncx <> spec_ncxCa, 16.8, 0.0112)
reaction003 = rxd.Reaction(spec_ncxCa > spec_ncx + spec_CaOut, 0.0056)
reaction004 = rxd.Reaction(spec_CaOut + spec_Leak <> spec_CaOutLeak, 1.5, 0.0011)
reaction005 = rxd.Reaction(spec_CaOutLeak > spec_Ca + spec_Leak, 0.0011)

reaction067 = rxd.Reaction(spec_CaM + spec_Ca**2 <> spec_CaMCa2, 6.0*spec_CaM*spec_Ca, 0.0091*spec_CaMCa2, custom_dynamics=True)
reaction068 = rxd.Reaction(spec_CaMCa2 + spec_Ca**2 <> spec_CaMCa4, 100.0*spec_CaMCa2*spec_Ca, 1.0*spec_CaMCa4, custom_dynamics=True)
reaction075 = rxd.Reaction(spec_CaMCa4 + spec_CK <> spec_CKCaMCa4, 10.0, 0.003)
```

$$\mathrm{PMCA} + \mathrm{Ca} \rightleftharpoons \mathrm{PMCACa} \rightarrow \mathrm{PMCA} + \mathrm{Ca}_{\mathrm{out}}$$

Examples for NEURON's RxD extension

```
Ca_flux_rate = rxd.Parameter(cyt, initial=0)
reaction_Ca_flux = rxd.Rate(spec_Ca, Ca_flux_rate)
```

$$\mathrm{Ca}_{(\mathrm{ext})} \longrightarrow \mathrm{Ca}$$




```
T = 1000./Ca_input_freq
tnow = 0
for itrain in range(0,Ntrains):
    for istim in range(0,Ca_input_N):
        tnew = Ca_input_onset + istim*T + trainT*itrain
        h.cvode.event(tnew, lambda: set_param(Ca_flux_rate, Ca_input_flux/6.022e23/my_volume*1e3))
        h.cvode.event(tnew+Ca_input_dur, lambda: set_param(Ca_flux_rate, 0))
```

```
reaction000 = rxd.Reaction(spec_Ca + spec_pmca <> spec_pmcaCa, 50.0, 0.007)
reaction001 = rxd.Reaction(spec_pmcaCa > spec_pmca + spec_CaOut, 0.0035)
reaction002 = rxd.Reaction(spec_Ca + spec_ncx <> spec_ncxCa, 16.8, 0.0112)
reaction003 = rxd.Reaction(spec_ncxCa > spec_ncx + spec_CaOut, 0.0056)
reaction004 = rxd.Reaction(spec_CaOut + spec_Leak <> spec_CaOutLeak, 1.5, 0.0011)
reaction005 = rxd.Reaction(spec_CaOutLeak > spec_Ca + spec_Leak, 0.0011)

reaction067 = rxd.Reaction(spec_CaM + spec_Ca**2 <> spec_CaMCa2, 6.0*spec_CaM*spec_Ca, 0.0091*spec_CaMCa2, custom_dynamics=True)
reaction068 = rxd.Reaction(spec_CaMCa2 + spec_Ca**2 <> spec_CaMCa4, 100.0*spec_CaMCa2*spec_Ca, 1.0*spec_CaMCa4, custom_dynamics=True)
reaction075 = rxd.Reaction(spec_CaMCa4 + spec_CK <> spec_CKCaMCa4, 10.0, 0.003)
```

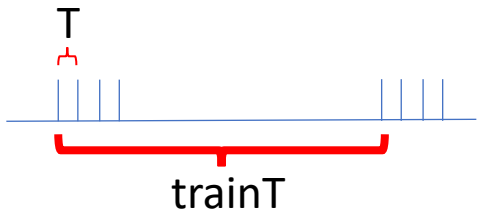
$$\mathrm{PMCA} + \mathrm{Ca} \rightleftharpoons \mathrm{PMCACa} \rightarrow \mathrm{PMCA} + \mathrm{Ca}_{\mathrm{out}}$$

$$\mathrm{ncx} + \mathrm{Ca} \rightleftharpoons \mathrm{ncxCa} \rightarrow \mathrm{ncx} + \mathrm{Ca}_{\mathrm{out}}$$

Examples for NEURON's RxD extension

```
Ca_flux_rate = rxd.Parameter(cyt, initial=0)
reaction_Ca_flux = rxd.Rate(spec_Ca, Ca_flux_rate)
```

$$\mathrm{Ca}_{(\mathrm{ext})} \longrightarrow \mathrm{Ca}$$




```
T = 1000./Ca_input_freq
tnow = 0
for itrain in range(0,Ntrains):
    for istim in range(0,Ca_input_N):
        tnew = Ca_input_onset + istim*T + trainT*itrain
        h.cvode.event(tnew, lambda: set_param(Ca_flux_rate, Ca_input_flux/6.022e23/my_volume*1e3))
        h.cvode.event(tnew+Ca_input_dur, lambda: set_param(Ca_flux_rate, 0))
```

```
reaction000 = rxd.Reaction(spec_Ca + spec_pmca <> spec_pmcaCa, 50.0, 0.007)
reaction001 = rxd.Reaction(spec_pmcaCa > spec_pmca + spec_CaOut, 0.0035)
reaction002 = rxd.Reaction(spec_Ca + spec_ncx <> spec_ncxCa, 16.8, 0.0112)
reaction003 = rxd.Reaction(spec_ncxCa > spec_ncx + spec_CaOut, 0.0056)
reaction004 = rxd.Reaction(spec_CaOut + spec_Leak <> spec_CaOutLeak, 1.5, 0.0011)
reaction005 = rxd.Reaction(spec_CaOutLeak > spec_Ca + spec_Leak, 0.0011)
```

$$\mathrm{PMCA} + \mathrm{Ca} \rightleftharpoons \mathrm{PMCACa} \rightarrow \mathrm{PMCA} + \mathrm{Ca}_{\mathrm{out}}$$

$$\mathrm{ncx} + \mathrm{Ca} \rightleftharpoons \mathrm{ncxCa} \rightarrow \mathrm{ncx} + \mathrm{Ca}_{\mathrm{out}}$$

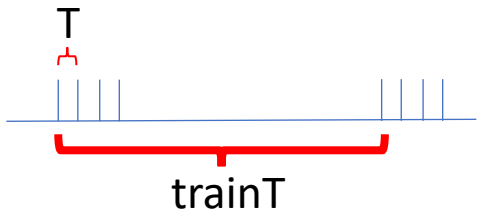
$$\mathrm{Ca}_{\mathrm{Out}} + \mathrm{Leak} \rightleftharpoons \mathrm{Ca}_{\mathrm{Out}}\mathrm{Leak} \rightarrow \mathrm{Ca} + \mathrm{Leak}$$

```
reaction067 = rxd.Reaction(spec_CaM + spec_Ca**2 <> spec_CaMCa2, 6.0*spec_CaM*spec_Ca, 0.0091*spec_CaMCa2, custom_dynamics=True)
reaction068 = rxd.Reaction(spec_CaMCa2 + spec_Ca**2 <> spec_CaMCa4, 100.0*spec_CaMCa2*spec_Ca, 1.0*spec_CaMCa4, custom_dynamics=True)
reaction075 = rxd.Reaction(spec_CaMCa4 + spec_CK <> spec_CKCaMCa4, 10.0, 0.003)
```

Examples for NEURON's RxD extension

```
Ca_flux_rate = rxd.Parameter(cyt, initial=0)
reaction_Ca_flux = rxd.Rate(spec_Ca, Ca_flux_rate)
```

$$\mathrm{Ca}_{(\mathrm{ext})} \longrightarrow \mathrm{Ca}$$




```
T = 1000./Ca_input_freq
tnow = 0
for itrain in range(0,Ntrains):
    for istim in range(0,Ca_input_N):
        tnew = Ca_input_onset + istim*T + trainT*itrain
        h.cvode.event(tnew, lambda: set_param(Ca_flux_rate, Ca_input_flux/6.022e23/my_volume*1e3))
        h.cvode.event(tnew+Ca_input_dur, lambda: set_param(Ca_flux_rate, 0))
```

```
reaction000 = rxd.Reaction(spec_Ca + spec_pmca <> spec_pmcaCa, 50.0, 0.007)
reaction001 = rxd.Reaction(spec_pmcaCa > spec_pmca + spec_CaOut, 0.0035)
reaction002 = rxd.Reaction(spec_Ca + spec_ncx <> spec_ncxCa, 16.8, 0.0112)
reaction003 = rxd.Reaction(spec_ncxCa > spec_ncx + spec_CaOut, 0.0056)
reaction004 = rxd.Reaction(spec_CaOut + spec_Leak <> spec_CaOutLeak, 1.5, 0.0011)
reaction005 = rxd.Reaction(spec_CaOutLeak > spec_Ca + spec_Leak, 0.0011)
```

$$\mathrm{PMCA} + \mathrm{Ca} \rightleftharpoons \mathrm{PMCACa} \rightarrow \mathrm{PMCA} + \mathrm{Ca}_{\mathrm{out}}$$

$$\mathrm{ncx} + \mathrm{Ca} \rightleftharpoons \mathrm{ncxCa} \rightarrow \mathrm{ncx} + \mathrm{Ca}_{\mathrm{out}}$$

$$\mathrm{Ca}_{\mathrm{Out}} + \mathrm{Leak} \rightleftharpoons \mathrm{Ca}_{\mathrm{Out}}\mathrm{Leak} \rightarrow \mathrm{Ca} + \mathrm{Leak}$$

```
reaction067 = rxd.Reaction(spec_CaM + spec_Ca**2 <> spec_CaMCa2, 6.0*spec_CaM*spec_Ca, 0.0091*spec_CaMCa2, custom_dynamics=True)
reaction068 = rxd.Reaction(spec_CaMCa2 + spec_Ca**2 <> spec_CaMCa4, 100.0*spec_CaMCa2*spec_Ca, 1.0*spec_CaMCa4, custom_dynamics=True)
reaction075 = rxd.Reaction(spec_CaMCa4 + spec_CK <> spec_CKCaMCa4, 10.0, 0.003)
```

$$2\mathrm{Ca} + \mathrm{CaM} \rightleftharpoons \mathrm{CaMCa}_2$$

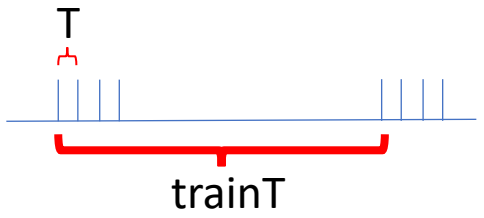
$$2\mathrm{Ca} + \mathrm{CaMCa}_2 \rightleftharpoons \mathrm{CaMCa}_4$$

$$\mathrm{CaMKII} + \mathrm{CaMCa}_4 \rightleftharpoons \mathrm{CaMKIICaMCa}_4$$

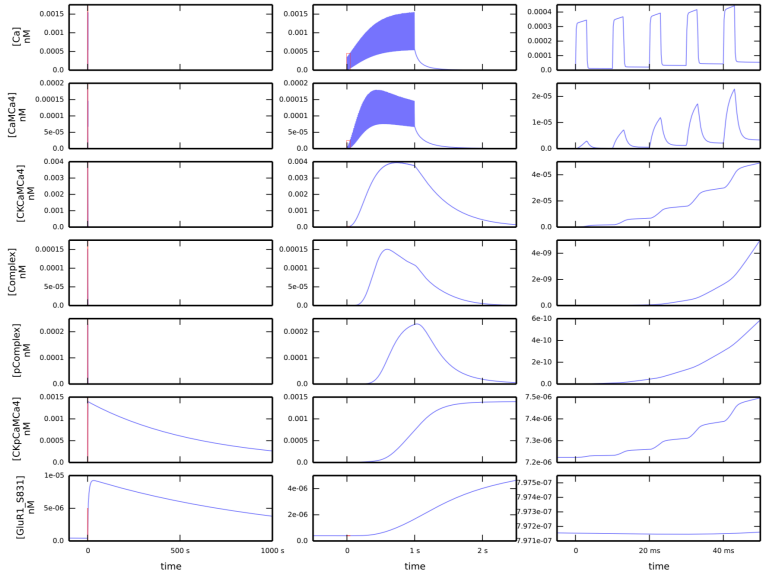
Examples for NEURON's RxD extension

```
Ca_flux_rate = rxd.Parameter(cyt, initial=0)
reaction_Ca_flux = rxd.Rate(spec_Ca, Ca_flux_rate)
```

$$\mathrm{Ca}_{(\mathrm{ext})} \longrightarrow \mathrm{Ca}$$




```
T = 1000./Ca_input_freq
tnow = 0
for itrain in range(0,Ntrains):
    for istim in range(0,Ca_input_N):
        tnew = Ca_input_onset + istim*T + trainT*itrain
        h.cvode.event(tnew, lambda: set_param(Ca_flux_rate, Ca_input_flux/6.022e23/my_volume*1e3))
        h.cvode.event(tnew+Ca_input_dur, lambda: set_param(Ca_flux_rate, 0))
```




```
reaction000 = rxd.Reaction(spec_Ca + spec_pmca <> spec_pmcaCa, 50.0, 0.007)
reaction001 = rxd.Reaction(spec_pmcaCa > spec_pmca + spec_CaOut, 0.0035)
reaction002 = rxd.Reaction(spec_Ca + spec_ncx <> spec_ncxCa, 16.8, 0.0112)
reaction003 = rxd.Reaction(spec_ncxCa > spec_ncx + spec_CaOut, 0.0056)
reaction004 = rxd.Reaction(spec_CaOut + spec_Leak <> spec_CaOutLeak, 1.5, 0.0011)
reaction005 = rxd.Reaction(spec_CaOutLeak > spec_Ca + spec_Leak, 0.0011)
```

$$\mathrm{PMCA} + \mathrm{Ca} \rightleftharpoons \mathrm{PMCACa} \rightarrow \mathrm{PMCA} + \mathrm{Ca}_{\mathrm{out}}$$

$$\mathrm{ncx} + \mathrm{Ca} \rightleftharpoons \mathrm{ncxCa} \rightarrow \mathrm{ncx} + \mathrm{Ca}_{\mathrm{out}}$$

$$\mathrm{Ca}_{\mathrm{Out}} + \mathrm{Leak} \rightleftharpoons \mathrm{Ca}_{\mathrm{Out}}\mathrm{Leak} \rightarrow \mathrm{Ca} + \mathrm{Leak}$$

```
reaction067 = rxd.Reaction(spec_CaM + spec_Ca**2 <> spec_CaMCa2, 6.0*spec_CaM*spec_Ca, 0.0091*spec_CaMCa2, custom_dynamics=True)
reaction068 = rxd.Reaction(spec_CaMCa2 + spec_Ca**2 <> spec_CaMCa4, 100.0*spec_CaMCa2*spec_Ca, 1.0*spec_CaMCa4, custom_dynamics=True)
reaction075 = rxd.Reaction(spec_CaMCa4 + spec_CK <> spec_CKCaMCa4, 10.0, 0.003)
```

$$2\mathrm{Ca} + \mathrm{CaM} \rightleftharpoons \mathrm{CaMCa}_2$$

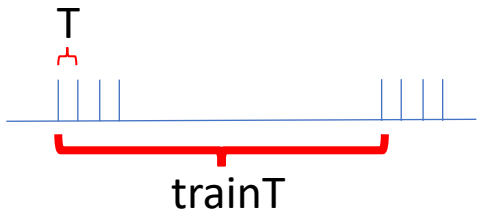
$$2\mathrm{Ca} + \mathrm{CaMCa}_2 \rightleftharpoons \mathrm{CaMCa}_4$$

$$\mathrm{CaMKII} + \mathrm{CaMCa}_4 \rightleftharpoons \mathrm{CaMKIICaMCa}_4$$

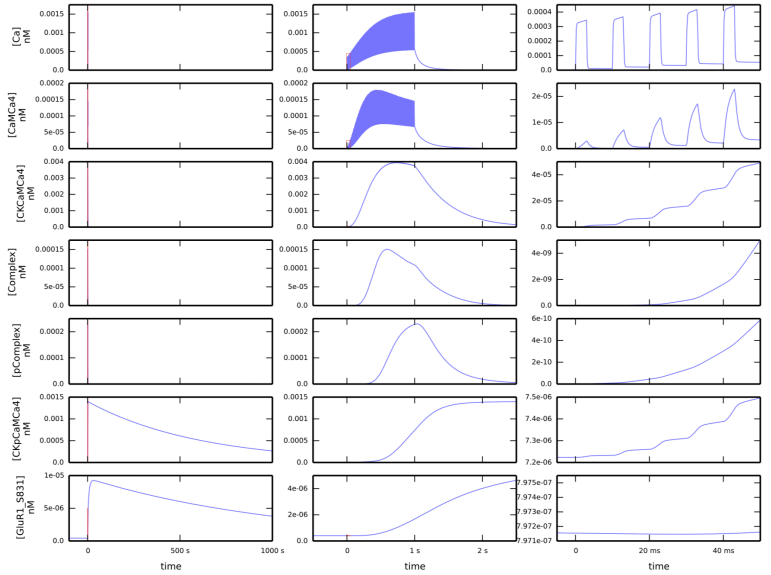
Examples for NEURON's RxD extension

```
Ca_flux_rate = rxd.Parameter(cyt, initial=0)
reaction_Ca_flux = rxd.Rate(spec_Ca, Ca_flux_rate)
```

$$\mathrm{Ca}_{(\mathrm{ext})} \longrightarrow \mathrm{Ca}$$




```
T = 1000./Ca_input_freq
tnow = 0
for itrain in range(0,Ntrains):
    for istim in range(0,Ca_input_N):
        tnew = Ca_input_onset + istim*T + trainT*itrain
        h.cvode.event(tnew, lambda: set_param(Ca_flux_rate, Ca_input_flux/6.022e23/my_volume*1e3))
        h.cvode.event(tnew+Ca_input_dur, lambda: set_param(Ca_flux_rate, 0))
```




```
reaction000 = rxd.Reaction(spec_Ca + spec_pmca <> spec_pmcaCa, 50.0, 0.007)
reaction001 = rxd.Reaction(spec_pmcaCa > spec_pmca + spec_CaOut, 0.0035)
reaction002 = rxd.Reaction(spec_Ca + spec_ncx <> spec_ncxCa, 16.8, 0.0112)
reaction003 = rxd.Reaction(spec_ncxCa > spec_ncx + spec_CaOut, 0.0056)
reaction004 = rxd.Reaction(spec_CaOut + spec_Leak <> spec_CaOutLeak, 1.5, 0.0011)
reaction005 = rxd.Reaction(spec_CaOutLeak > spec_Ca + spec_Leak, 0.0011)
```

$$\mathrm{PMCA} + \mathrm{Ca} \rightleftharpoons \mathrm{PMCACa} \rightarrow \mathrm{PMCA} + \mathrm{Ca}_{\mathrm{out}}$$

$$\mathrm{ncx} + \mathrm{Ca} \rightleftharpoons \mathrm{ncxCa} \rightarrow \mathrm{ncx} + \mathrm{Ca}_{\mathrm{out}}$$

$$\mathrm{Ca}_{\mathrm{Out}} + \mathrm{Leak} \rightleftharpoons \mathrm{Ca}_{\mathrm{Out}}\mathrm{Leak} \rightarrow \mathrm{Ca} + \mathrm{Leak}$$

```
reaction067 = rxd.Reaction(spec_CaM + spec_Ca**2 <> spec_CaMCa2, 6.0*spec_CaM*spec_Ca, 0.0091*spec_CaMCa2, custom_dynamics=True)
reaction068 = rxd.Reaction(spec_CaMCa2 + spec_Ca**2 <> spec_CaMCa4, 100.0*spec_CaMCa2*spec_Ca, 1.0*spec_CaMCa4, custom_dynamics=True)
reaction075 = rxd.Reaction(spec_CaMCa4 + spec_CK <> spec_CKCaMCa4, 10.0, 0.003)
```

$$2\mathrm{Ca} + \mathrm{CaM} \rightleftharpoons \mathrm{CaMCa}_2$$

$$2\mathrm{Ca} + \mathrm{CaMCa}_2 \rightleftharpoons \mathrm{CaMCa}_4$$

$$\mathrm{CaMKII} + \mathrm{CaMCa}_4 \rightleftharpoons \mathrm{CaMKIICaMCa}_4$$

Python3: <> replaced by !=

Extending Cable Theory for calcium based plasticity

We have seen numerical methods focusing on plasticity (LTP/LTD) occurring in small spatial compartments such as dendritic spines, and some numerical methods that incorporates “space”.

Neuron’s RxD extension

What about theoretical frameworks that embraces both the nonlinear nature of the dendrite and plasticity?

Can this be developed?

Yes

One can show that for calcium diffusion in the presence of buffers this system can be described by a cable equation as long as certain conditions are met.

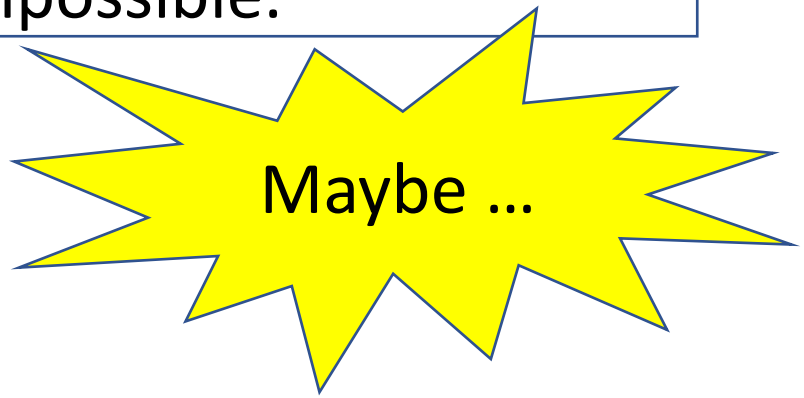
Extending Cable Theory for calcium based plasticity

But what about the nonlinear nature introduced by the presence of voltage-gated ion channels?

We know that no-one has found analytical solutions to the Hodgkin-Huxley (HH) equations (but there have been some attempts to derive analytical approximations).

For a dendrite with voltage-gated ion channels (described using the HH formalism), deriving a theoretical framework seems impossible.

Let's look at ion channels more closely .. Actually how they are distributed.



Experiments have shown that they are small nano-scale pores in the membrane.

Extending Cable Theory for calcium based plasticity

However the typical equation for a nonlinear dendrite there is no mention of how channels are distributed.

$$C_m \frac{\partial V}{\partial t} = \frac{d}{4\rho_i} \frac{\partial^2 V}{\partial x^2} + g_l(V_l - V) + g_{\mathrm{na}} m^3 h (V_{\mathrm{na}} - V) + g_{\mathrm{k}} n^4 (V_{\mathrm{k}} - V)$$

here here

What this is saying is that the ion channels are uniformly and continuously distributed across the membrane ie $F_{distribution}$=1.

But there is a small (descriptive) problem – physically since they are pores in the membrane and they are uniformly & continuously distributed across the membrane … the membrane should not exist.

However

If we embrace their discrete nature then this allows an opportunity to develop formal analytical methods.

Extending Cable Theory for calcium based plasticity

Since ion channels are discrete and small in the spatial sense one can describe their spatial distribution using the Dirac delta function.

One approximation that we will make for pragmatic purposes is that we will consider hotspots of ions along the dendrite. Now we will present the entire starting system of equations.

$$
\begin{aligned}
C_m \frac{\partial V}{\partial t} &= \frac{d}{4\rho_i}\frac{\partial^2 V}{\partial x^2} + g_\ell(V_\ell - V) + \sum_{\mu=1}^{k}\sum_{i=1}^{N_\mu} I^{\mu}_{\text{ion}}(x,t;V;[\text{Ca}])\delta(x-x_i^\mu) + \overline{I}_A(x,t), \\
\frac{\partial [\text{Ca}]_\text{i}}{\partial t} &= D_{\text{Ca}}\frac{\partial^2 [\text{Ca}]_\text{i}}{\partial x^2} + b_M\,[M] - f_M[\text{Ca}]_\text{i}\mathcal{B}^{\text{T}}_{\text{M}} + \text{f}_{\text{M}}[\text{Ca}]_\text{i}\,[\text{M}] \\
&\qquad -\frac{4P_m}{d}K_p\frac{[\text{Ca}]_\text{i}}{[\text{Ca}]_\text{i}+\text{K}_\text{p}} + \sum_{\nu=1}^{s}\sum_{i=1}^{N_\nu}\frac{2\mathcal{I}^{\nu}_{\text{Ca}}(x,t;V;[\text{Ca}])}{\mathcal{F}d}\delta(x-x_i^\nu), \\
\frac{\partial\,[M]}{\partial t} &= D_M\frac{\partial^2\,[M]}{\partial x^2} - b_M\,[M] + f_M[\text{Ca}]_\text{i}\mathcal{B}^{\text{T}}_{\text{M}} - \text{f}_{\text{M}}[\text{Ca}]_\text{i}\,[\text{M}]\,, \\
\frac{\partial m^\mu}{\partial t} &= \alpha^\mu_m(V) - \Big[\alpha^\mu_m(V)+\beta^\mu_m(V)\Big]m^\mu(V) \;=\; \frac{m^\mu_\infty(V)-m^\mu}{\tau^\mu_m(V)}, \\
\frac{\partial h^\mu}{\partial t} &= \alpha^\mu_h(V) - \Big[\alpha^\mu_h(V)+\beta^\mu_h(V)\Big]h^\mu(V) \;=\; \frac{h^\mu_\infty(V)-h^\mu}{\tau^\mu_h(V)},
\end{aligned}
$$

Extending Cable Theory for calcium based plasticity

The calcium system can be linearized using the rapid buffer approximation. This allows the system of reaction-diffusion equations to be reduced to the following set of equations:

Explicitly introduced NMDA current since it will be needed for synaptic plasticity

$$C_m \frac{\partial V}{\partial t} = \frac{d}{4\rho_i}\frac{\partial^2 V}{\partial x^2} + g_\ell(V_\ell - V) + \sum_{\mu=1}^{k}\sum_{i=1}^{N_\mu} I^{\mu}_{\text{ion}}(x,t;V;[\text{Ca}])\delta(x - x_i^{\mu}) + \sum_{i=1}^{N_{\text{NMDA}}} I^{\text{NMDA}}(x,t;V)\delta(x - x_i^{\text{NMDA}}) + I_A(x,t)$$

$$\tau_{\text{Ca}}\frac{\partial [\text{Ca}]_{\text{i}}}{\partial t} = \lambda^2_{\text{Ca}}\frac{\partial^2 [\text{Ca}]_{\text{i}}}{\partial x^2} - [\text{Ca}]_{\text{i}} + \sum_{\nu=1}^{\text{s}}\sum_{\text{i}=1}^{\text{N}_\nu}\frac{\mathcal{I}^{\nu}_{\text{Ca}}(\text{x},\text{t};\text{V};[\text{Ca}])}{2\mathcal{F}\text{P}_{\text{m}}}\delta(\text{x} - \text{x}_{\text{i}}^{\nu}), + \frac{1}{2\mathcal{F}P_m}\sum_{i=1}^{N_{\text{NMDA}}}\mathcal{I}^{\text{NMDA}}_{\text{Ca}}(x,t;V)\delta(x - x_i^{\text{NMDA}}),$$

where the applied current density per unit area I_A represents an externally applied input, $\tau_{Ca} = d(1+\beta)/(4P_m)$, $\lambda_{\text{Ca}} = \sqrt{d(D_{\text{Ca}} + \beta D_M)/4P_m}$ and $\beta = \mathcal{B}^T/K_p$.

Extending Cable Theory for calcium based plasticity

Rewriting in terms of dimensionless quantities leads to:

$$\begin{aligned}
\frac{\partial \Phi}{\partial T} &= \frac{\partial^2 \Phi}{\partial X^2} - \Phi + \frac{R_m}{\lambda} \sum_{\mu=1}^{k} \sum_{i=1}^{N_\mu} I_{\text{ion}}^{\mu}(X, T; \Phi; \Phi^{\text{Ca}}) \delta(X - X_i^{\mu}) \\
&\quad + \frac{R_m}{\lambda} \sum_{i=1}^{N_{\text{NMDA}}} I^{\text{NMDA}}(X, T; \Phi) \delta(X - X_i^{\text{NMDA}}) + \frac{\mathcal{I}_A(X, T)}{U_{\text{peak}}} \\
\frac{\partial \Phi^{\text{Ca}}}{\partial T_{\text{Ca}}} &= \frac{\partial^2 \Phi^{\text{Ca}}}{\partial X_{\text{Ca}}^2} - \Phi^{\text{Ca}} + \mathcal{N}_1 \sum_{\nu=1}^{s} \sum_{i=1}^{N_\nu} \mathcal{I}_{\text{Ca}}^{\nu}(X_{\text{Ca}}, T_{\text{Ca}}; \Phi; \Phi^{\text{Ca}}) \delta(X_{\text{Ca}} - X_{\text{Ca},i}^{\nu}) \\
&\quad + \mathcal{N}_1 \sum_{i=1}^{N_{\text{NMDA}}} \mathcal{I}_{\text{Ca}}^{\text{NMDA}}(X_{\text{Ca}}, T_{\text{Ca}}; \Phi) \delta(X_{\text{Ca}} - X_{\text{Ca},i}^{\text{NMDA}}),
\end{aligned}$$

where $\mathcal{N}_1 = U_{\text{peak}}/(2\lambda_{\text{Ca}} \mathcal{F} P_m [\text{Ca}]_{\text{ref}})$, the applied current is $\mathcal{I}_A = R_m I_A$ and $X_i^{\mu} = x_i^{\mu}/\lambda$ and $X_{\text{Ca},i}^{\mu} = x_i^{\mu}/\lambda_{\text{Ca}}$ denotes the corresponding location of the i^{th} hotspot for channel μ in the cable and chemical cable, respectively.

Extending Cable Theory for calcium based plasticity

and with

$$I^{\mu}_{\text{ion}}(X_i, T; \Phi, \Phi_{\text{Ca}}) \quad = \quad \varepsilon g^{\mu *} \mathcal{N}^{\mu *}(X_i) \, (m^{\mu}[\Phi_i])^p \, (h^{\mu}[\Phi_i])^q \, \mathcal{F}^{\mu}(\Phi_i, \Phi_i^{\text{Ca}}),$$

- where μ denotes the ion channel type of the i^{th} hotspot,
- ε is a parameter scaling the "whole-cell" macroscopic transmembrane current density into spatially discrete clusters of ion channels
- $g^{\mu *}$ is the single channel conductance
- $\mathcal{N}^{\mu *}(X_i)$ denotes the number of channels per unit length, with units of $(\text{cm})^{-1}$
- $\theta^{\mu}(X_i)$ represents the number of channels in the i^{th} hotspot of channel μ
- $m^{\mu}[\Phi_i]$ and $h^{\mu}[\Phi_i]$ represent the activation and inactivation variables of channel μ, respectively
- p and q are exponents and
- $\mathcal{F}^{\mu}(\Phi_i, \Phi_i^{\text{Ca}})$ represents the voltage and/or calcium current dependence of channel μ.

and

$$I^{\text{NMDA}}(X, T; \Phi) \quad = \quad \varepsilon g^{\text{NMDA}} \mathcal{N}^{\text{NMDA}}(X_i) \bar{\mathfrak{g}}_{\text{NMDA}}(T - T_j) H(T - T_j) B(\Phi)(\Phi^{rev}_{\text{NMDA}} - \Phi)$$

$$I^{\text{NMDA}}_{\text{Ca}}(X_{\text{Ca}}, T_{\text{Ca}}; \Phi) \quad = \quad \varepsilon g^{\text{NMDA}} \mathcal{N}^{\text{NMDA}}(X_{\text{Ca},i}) \bar{\mathfrak{g}}_{\text{NMDA}}(T_{\text{Ca}} - T_{\text{Ca},j}) H(T_{\text{Ca}} - T_{\text{Ca},j}) B(\Phi)(\Phi^{rev}_{\text{Ca}} - \Phi),$$

Extending Cable Theory for calcium based plasticity

The presence of delta functions permits us to consider ion channel hotspots act as current sources and thus allows us to recast the our system into a system of nonlinear Volterra integral equations.

$$\Phi(X,T) = \Psi(X,T) + \Psi^{\mathrm{NMDA}}(X,T) + \varepsilon\frac{R_m}{\lambda}\sum_{\mu=1}^{k}\int_0^T\sum_{i=1}^{N_\mu} f^\mu\left[\Phi_i(s),\Phi_i^{\mathrm{Ca}}(s)\right]G(X,X_i^\mu;T-s)ds,$$

$$\begin{aligned}\Phi^{\mathrm{Ca}}(X_{\mathrm{Ca}},T_{\mathrm{Ca}}) &= \Psi^{\mathrm{Ca}}(X_{\mathrm{Ca}},T_{\mathrm{Ca}}) + \Psi^{\mathrm{NMDA}}_{\mathrm{Ca}}(X_{\mathrm{Ca}},T_{\mathrm{Ca}}) \\ &\quad + \varepsilon\mathcal{N}_1\sum_{\nu=1}^{s}\int_0^{T_{\mathrm{Ca}}}\sum_{i=1}^{N_\nu}\mathfrak{F}^\nu_{\mathrm{Ca}}\left[\Phi_i(s),\Phi_i^{\mathrm{Ca}}(s)\right]G\left(X_{\mathrm{Ca}},X_{\mathrm{Ca},i};T_{\mathrm{Ca}}-s\right)ds,\end{aligned}$$

Extending Cable Theory for calcium based plasticity

where

$$\Psi(X,T) = \int_0^T \int_0^L \frac{I_A(Y,s)}{U_{\text{peak}}} G(X,Y;T-s)dY\,ds,$$

$$\Psi^{\text{NMDA}}(X,T) = \varepsilon \frac{R_m}{\lambda} \int_0^T \sum_{i=1}^{N_{\text{NMDA}}} f^{\text{NMDA}}[\Phi_i(s)]\, G(X, X_i^{\text{NMDA}}; T-s)ds$$

$$f^{\mu}_{\text{ion}}(X_i,T;\Phi,\Phi_{\text{Ca}}) = g^{\mu *}\mathcal{N}^{\mu *}(X_i)\,(m^{\mu}[\Phi_i])^p\,(h^{\mu}[\Phi_i])^q\,\mathcal{F}^{\mu}(\Phi_i,\Phi_i^{\text{Ca}}),$$

$$f^{\text{NMDA}}(X_i,T;\Phi) = g^{\text{NMDA}}\mathcal{N}^{\text{NMDA}}(X_i)\overline{\mathfrak{g}}(T-T_j)H(T-T_j)B(\Phi)(\Phi^{\text{rev}}_{\text{NMDA}}-\Phi),$$

$$\Psi^{\text{Ca}}(X_{\text{Ca}},T_{\text{Ca}}) = \int_0^L G(X_{\text{Ca}},Y_{\text{Ca}};T_{\text{Ca}})\Phi^{\text{Ca}}(Y_{\text{Ca}},0)\,dY_{\text{Ca}}$$

$$\Psi^{\text{NMDA}}_{\text{Ca}}(X_{\text{Ca}},T_{\text{Ca}}) = \varepsilon\mathcal{N}_1 \int_0^{T_{\text{Ca}}} \sum_{i=1}^{N_{\text{NMDA}}} f^{\text{NMDA}}_{\text{Ca}}\left[\Phi_i(s),\Phi_i^{\text{Ca}}(s)\right] G\left(X_{\text{Ca}},X_{\text{Ca},i};T_{\text{Ca}}-s\right)ds$$

Extending Cable Theory for calcium based plasticity

In order to explicitly find analytical express one can apply a regular perturbation expansion of Φ and Φ_{Ca} but this leads to messy mathematics. For simplicity we will only consider calcium channels to be voltage dependent. Then we only expand in the voltage component.

$$\Phi(X,T) = \Phi_0(X,T) + \varepsilon\Phi_1(X,T) + \varepsilon^2\Phi_2(X,T) + \varepsilon^3\Phi_3(X,T) + O(\varepsilon^4),$$

where

$$\Phi_1(X,T) = \frac{R_m}{\lambda}\sum_{\mu=1}^{k}\sum_{i=1}^{N_\mu}\int_0^T G(X,X_i^\mu;T-s)f^\mu[\Phi_{0i}(s)]\,ds + \frac{R_m}{\lambda}\sum_{i=1}^{N_{\mathrm{NMDA}}}\int_0^T G(X,X_i^{\mathrm{NMDA}};T-s)f^{\mathrm{NMDA}}[\Phi_{0i}(s)]\,ds,$$

Other higher order terms are similar but involve products of Φ_i and derivatives of f^μ.

Extending Cable Theory for calcium based plasticity

and

$$\Phi^{\mathrm{Ca}}\left(X_{\mathrm{Ca}}, T_{\mathrm{Ca}}\right) = \int_0^L G(X_{\mathrm{Ca}}, Y_{\mathrm{Ca}}; T_{\mathrm{Ca}})\Phi^{\mathrm{Ca}}(Y_{\mathrm{Ca}}, 0)\, dY_{\mathrm{Ca}}$$
$$+ \frac{U_{\mathrm{peak}}}{2\lambda_{\mathrm{Ca}}\mathcal{F}P_m[\mathrm{Ca}]_{\mathrm{ref}}} \sum_{\nu=1}^{s}\sum_{i=1}^{N_\nu} \int_0^{T_{\mathrm{Ca}}} G\left(X_{\mathrm{Ca}}, X^{\nu}_{\mathrm{Ca},i}; T_{\mathrm{Ca}} - s\right) \mathcal{I}^{\nu}_{\mathrm{Ca}}\left(X^{\nu}_{\mathrm{Ca},i}, s\right) ds$$
$$+ \frac{U_{\mathrm{peak}}}{2\lambda_{\mathrm{Ca}}\mathcal{F}P_m[\mathrm{Ca}]_{\mathrm{ref}}} \sum_{i=1}^{N_{\mathrm{NMDA}}} \int_0^{T_{\mathrm{Ca}}} G\left(X_{\mathrm{Ca}}, X^{\mathrm{NMDA}}_{\mathrm{Ca},i}; T_{\mathrm{Ca}} - s\right) \mathcal{I}^{\mathrm{NMDA}}_{\mathrm{Ca}}\left(X^{\mathrm{NMDA}}_{\mathrm{Ca},i}, s\right) ds.$$

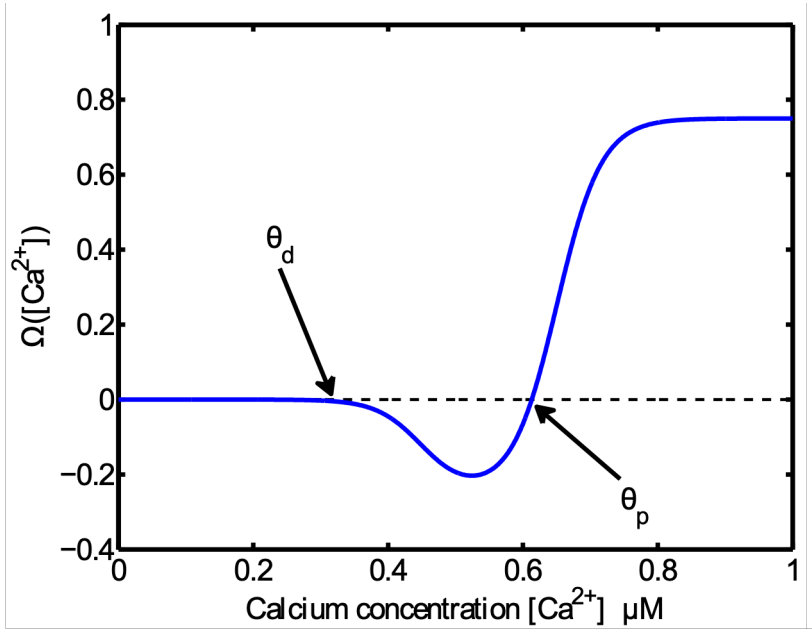
Thus arriving at our expression for calcium

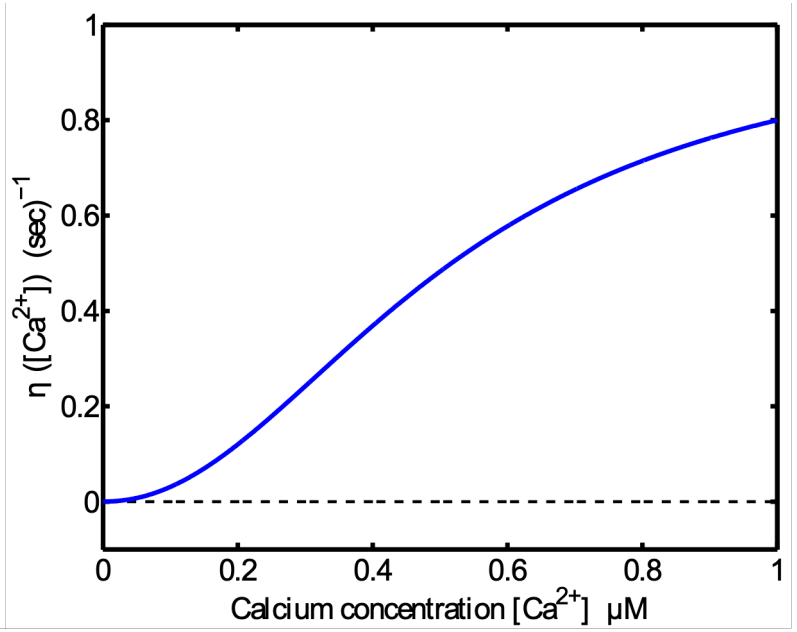
We can now apply Shouval's Calcium-dependent plasticity model

Extending Cable Theory for calcium based plasticity

Calcium dependent plasticity

$$\begin{aligned}
\frac{dW_j}{dt} &= \eta\left([\mathrm{Ca}^{2+}]_j\right)\left(\Omega\left([\mathrm{Ca}^{2+}]_j\right) - W_j\right), \\
\Omega(x) &= \sigma(x - a_2, b_2) - A\sigma(x - a_1, b_1), \\
\eta(x) &= \frac{p_2 + x^{p_3}}{p_1 + p_4(p_2 + x^{p_3})}, \\
\sigma(x, a) &= \frac{e^{ax}}{1 + e^{ax}},
\end{aligned}$$





Extending Cable Theory for calcium based plasticity

Calcium dependent plasticity