DynaSim – a Matlab toolbox for neural modeling

Jason Sherfey[1,2], Nancy Kopell[2,3]

[1]Graduate Program for Neuroscience, Boston University; [2]Center for Computational Neuroscience and Neural Technology; [3]Dept. of Mathematics and Statistics, BU

INTRODUCTION

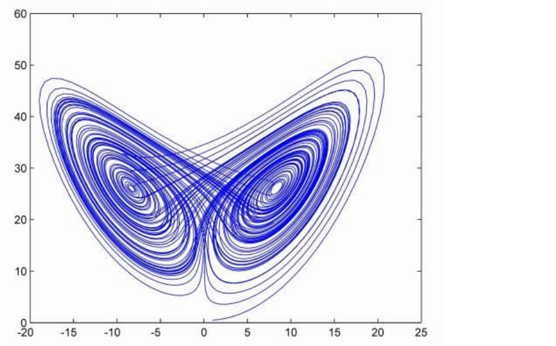
DynaSim is an open-source Matlab/Octave toolbox for rapid prototyping of large ODE systems (e.g., neural models), batch simulation management, and efficient model sharing. DynaSim is available at github.com/DynaSim. Major features include:

- **Equation-based modular model specification**
- **Parameter sweeping and cluster computing**
- **Graphical interface for building and exploring models**
- **Integrated online model repository: www.InfiniteBrain.org**

DynaSim Toolbox

Example: Lorenz equations

```
eqns={
  's=10; r=27; b=2.666';
  'dx/dt=s*(y-x)';
  'dy/dt=r*x-y-x*z';
  'dz/dt=-b*z+x*y';
};
data=SimulateModel(eqns,'tspan',[0 100],'ic',[1 2 .5]);
figure; plot(data.pop1_x,data.pop1_z)
```



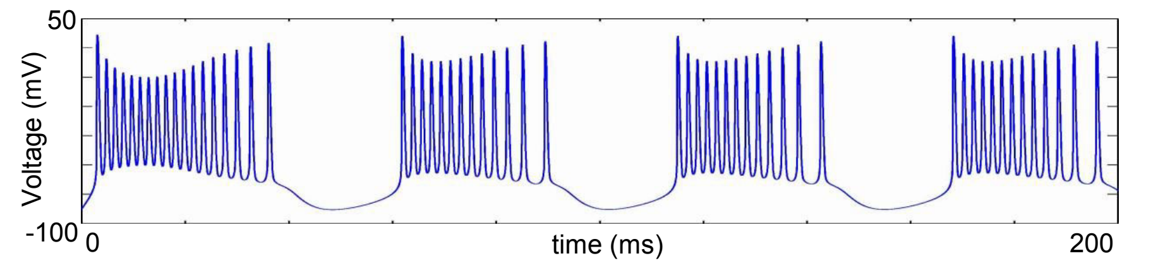
Pass equations directly to SimulateModel.

Modular models with reusable mechanisms

Larger models can easily build on existing sub-models (“mechanisms”) with terms linked to @ identifiers. (Design inspired by NEURON.)

Hodgkin-Huxley-type bursting neuron

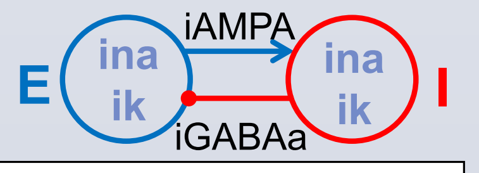
```
eqns='dv/dt=5+@current; {iNaF, iKDR, iM}; v(0)=-70';
data=SimulateModel(eqns,'tspan',[0 200]);
figure; plot(data.time,data.pop1_v))
```



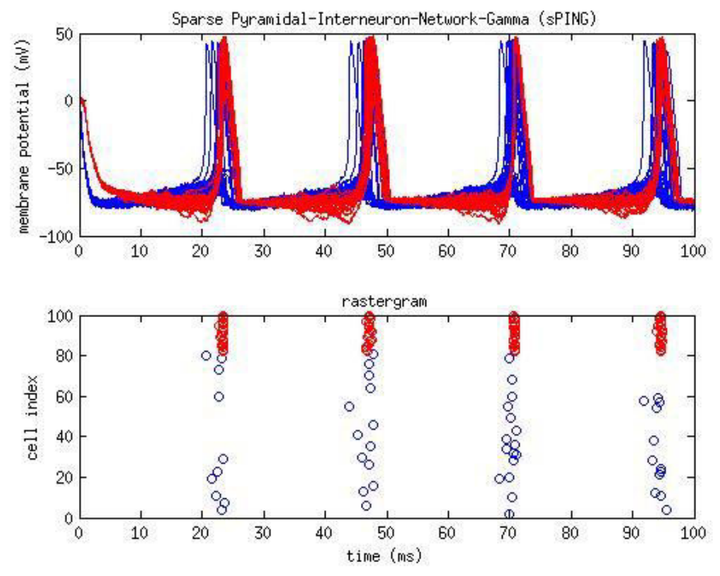
Existing sub-models of iNaF, iKDR, and iM currents simplify the specification of a detailed neuron model. Mechanisms can be stored in text files, infinitebrain.org, or defined in script.

Pyramidal-Interneuron Network rhythm

Define/reuse two mechanisms (ina, ik) with currents (INa, IK) linked to dynamics for two populations (E, I).



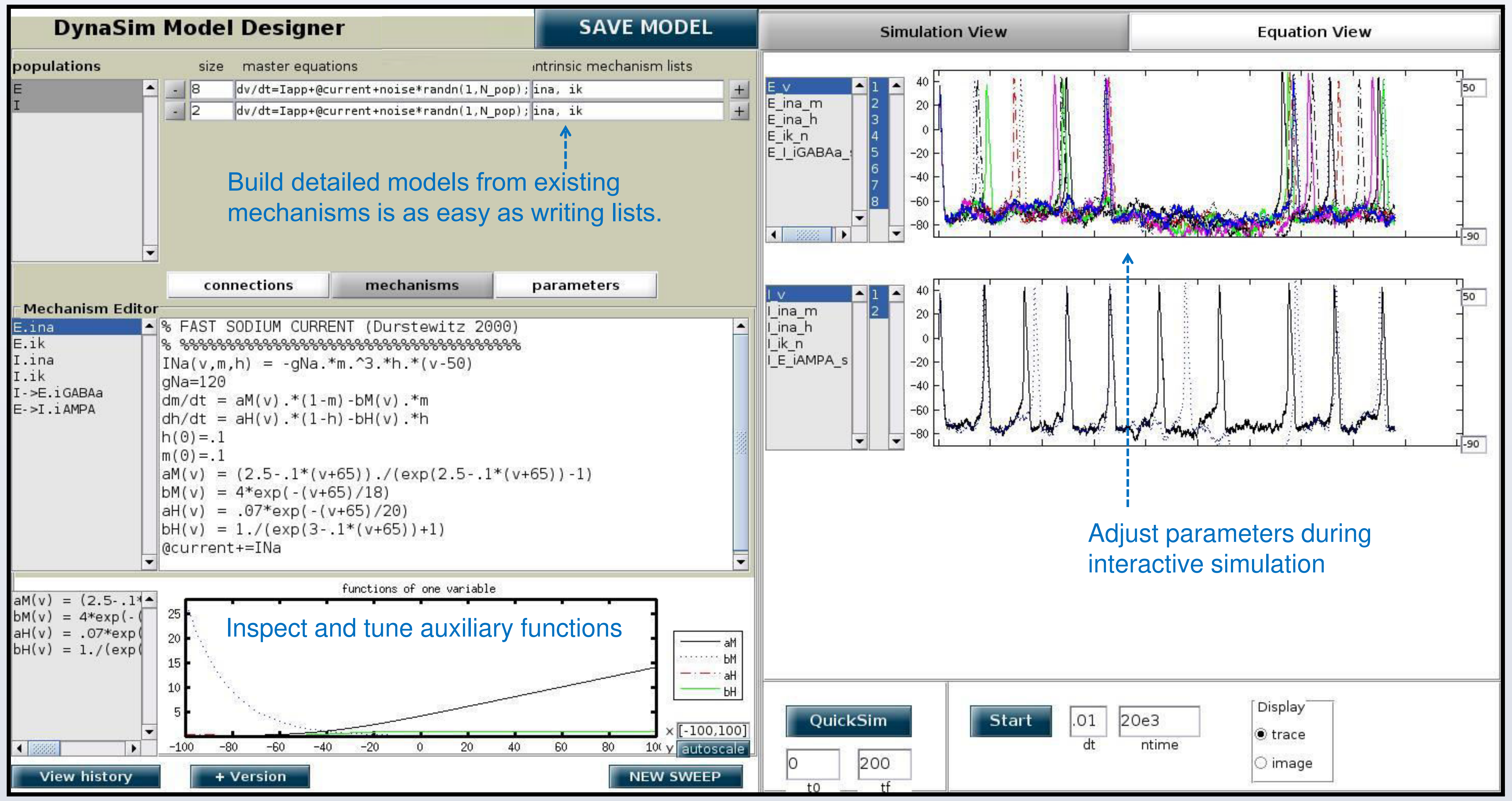
```
ina={
   'INa(v,m,h) = -gNa.*m.^3.*h.*(v-50); gNa=120';
   'dm/dt = aM(v).*(1-m)-bM(v).*m; m(0)=.1';
   'dh/dt = aH(v).*(1-h)-bH(v).*h; h(0)=.1';
   'aM(v) = (2.5-.1*(v+65))./(exp(2.5-.1*(v+65))-1)';
   'bM(v) = 4*exp(-(v+65)/18)';
   'aH(v) = .07*exp(-(v+65)/20)';
   'bH(v) = 1./(exp(3-.1*(v+65))+1)';
    '@current+=INa';
 };

ik={
   'IK(v,n) = -gK.*n.^4.*(v+77); gK=36';
   'dn/dt = aN(v).*(1-n)-bN(v).*n; n(0)=0';
   'aN(v) = (.1-.01*(v+65))./(exp(1-.1*(v+65))-1)';
   'bN(v) = .125*exp(-(v+65)/80)';
    '@current+=IK';
 };

master_equations='dv/dt=Iapp+@current+randn(1,N_pop); {ina,ik}';
s.populations(1).name='E';
s.populations(1).size=80;
s.populations(1).equations=master_equations;
s.populations(2).name='I';
s.populations(2).size=20;
s.populations(2).equations=master_equations;
s.connections(1).direction='I->E';
s.connections(1).mechanism_list={'iGABAa'};
s.connections(2).direction='E->I';
s.connections(2).mechanism_list={'iAMPA'};
s.mechanisms(1).name='ina';
s.mechanisms(1).equations=ina;
s.mechanisms(2).name='ik';
s.mechanisms(2).equations=ik;
data=SimulateModel(s);
```



GRAPHICAL USER INTERFACE

Open a model in the DynaSim GUI: **dynasim(model)**

View full model equations and dynamics during interactive model building



populations	size	master equations	intrinsic mechanism lists
E	8	dv/dt=Iapp+@current+noise*randn(1,N_pop);	ina, ik
I	2	dv/dt=Iapp+@current+noise*randn(1,N_pop);	ina, ik

Build detailed models from existing mechanisms is as easy as writing lists.

Mechanism Editor: E.ina, E.ik, I.ina, I.ik, I->E.iGABAa, E->I.iAMPA

```
% FAST SODIUM CURRENT (Durstewitz 2000)
% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
INa(v,m,h) = -gNa.*m.^3.*h.*(v-50)
gNa=120
dm/dt = aM(v).*(1-m)-bM(v).*m
dh/dt = aH(v).*(1-h)-bH(v).*h
h(0)=.1
m(0)=.1
aM(v) = (2.5-.1*(v+65))./(exp(2.5-.1*(v+65))-1)
bM(v) = 4*exp(-(v+65)/18)
aH(v) = .07*exp(-(v+65)/20)
bH(v) = 1./(exp(3-.1*(v+65))+1)
@current+=INa
```

Inspect and tune auxiliary functions

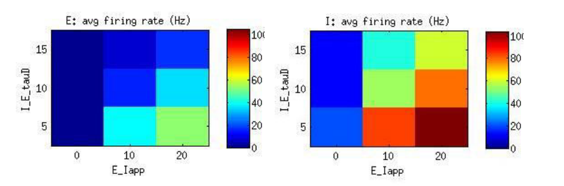
x [-100,100]; t0 0; tf 200; dt .01; ntime 20e3; Display: ● trace ○ image

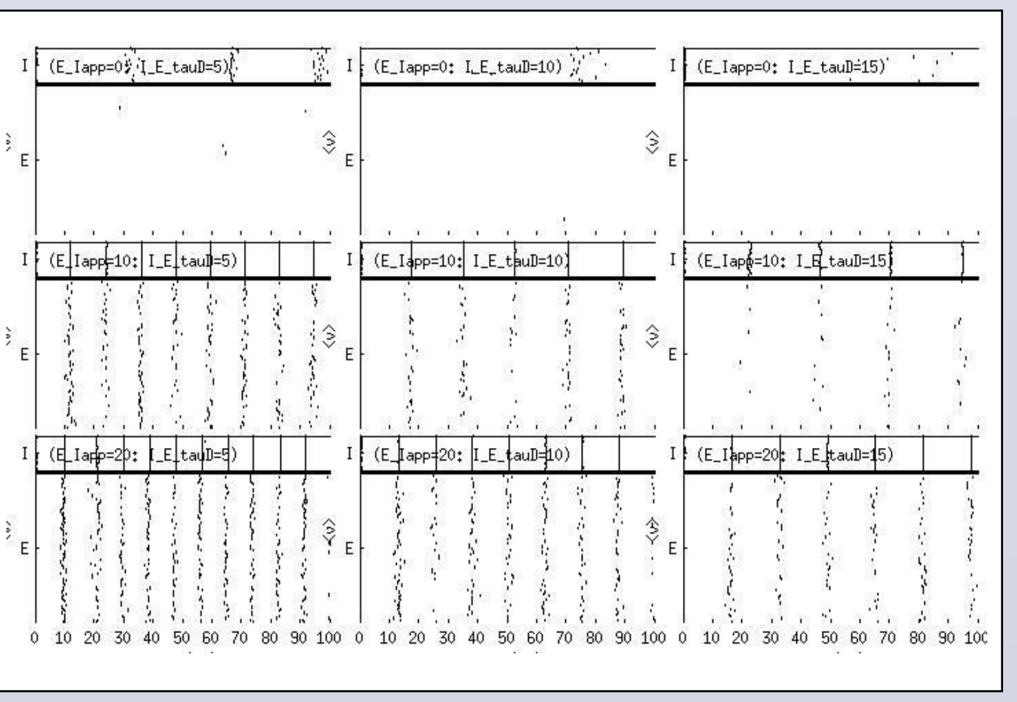
Adjust parameters during interactive simulation

Parameter sweeps and cluster computing

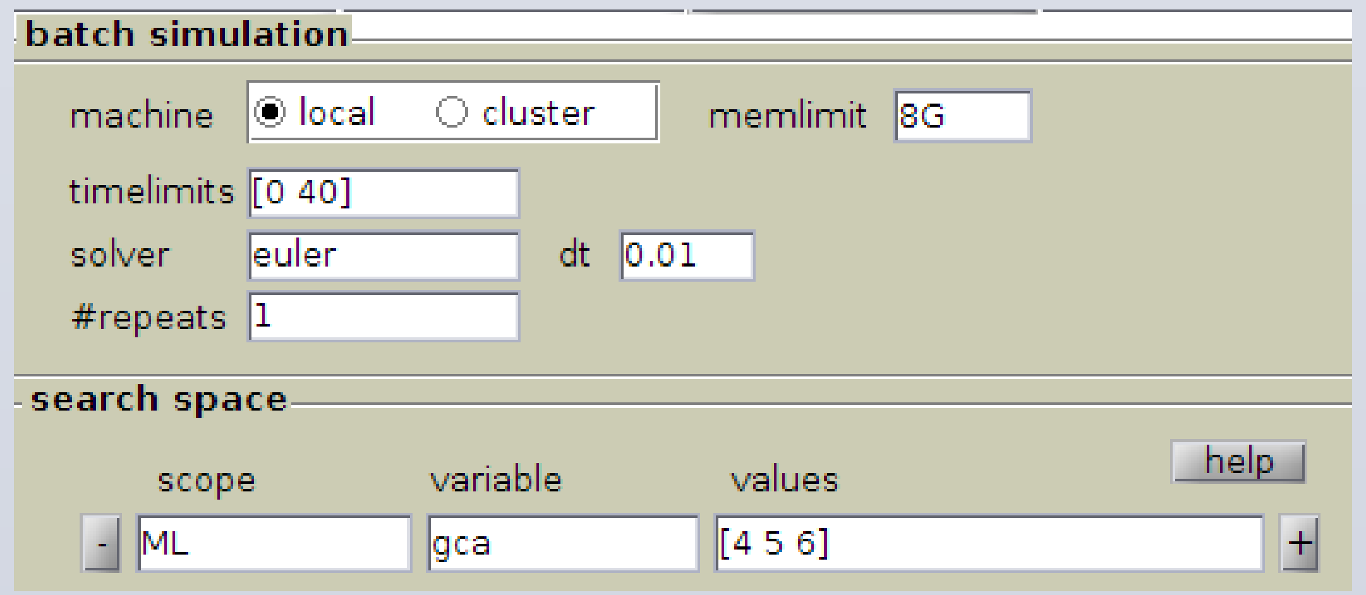
Using DynaSim toolbox in Matlab script

```
vary={
  'E'   ,'Iapp',[0 10 20];
  'I->E','tau',[5 10 15]
  };
data=SimulateModel(s,'vary',vary)
PlotData(data,'plot_type','raster');
PlotFR(data);
```





Using DynaSim GUI



machine ● local ○ cluster; memlimit 8G
timelimits [0 40]
solver euler; dt 0.01
#repeats 1

scope	variable	values
ML	gca	[4 5 6]

Cluster computing is as simple as setting an optional flag:

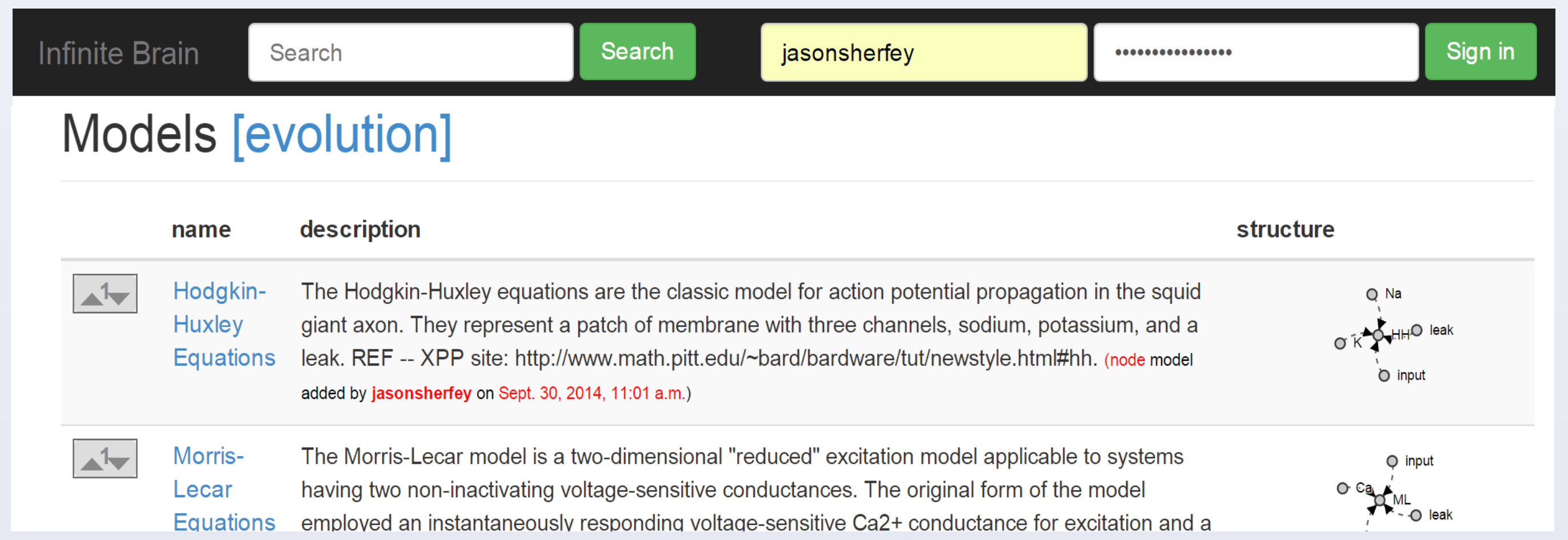
```
[~, info]=SimulateModel(s,'vary',vary,'cluster_flag',1);
data=ImportData(info);
PlotData(data,'plot_type','raster');
```

- Batch management - DynaSim can run many simulations varying some aspect of the model specified using a simple syntax.
- Cluster computing – DynaSim handles job creation and submission using qsub from a login node.
- Results can be saved to disk with a model file for repeating simulations.

MODEL REPOSITORY

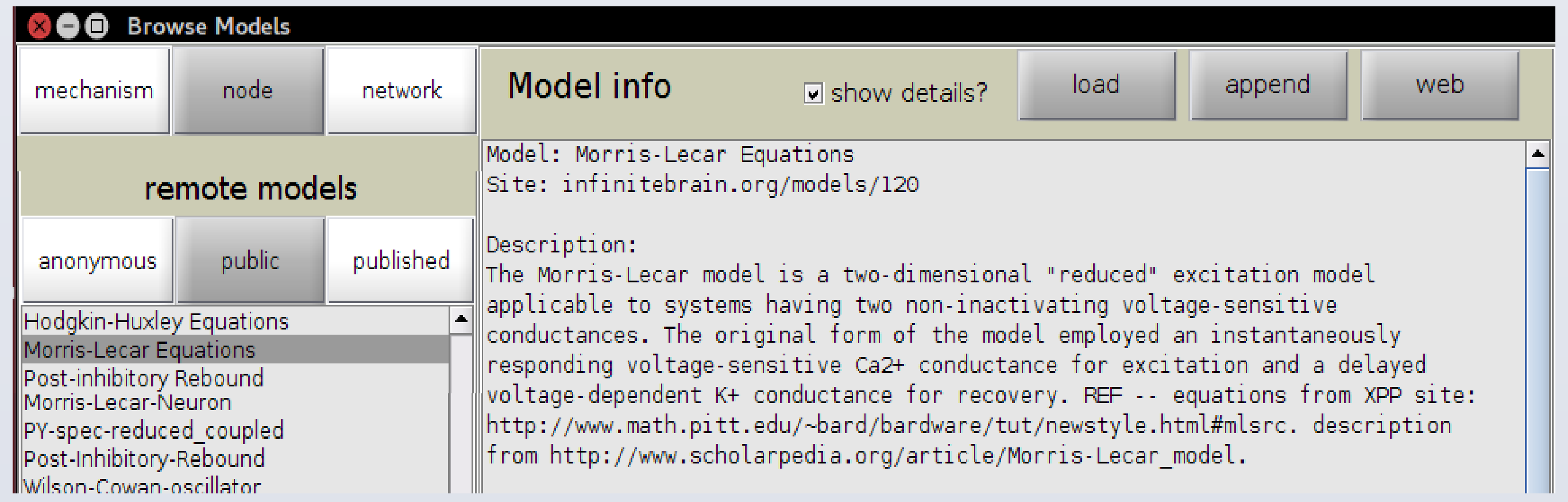
InfiniteBrain.org is an online repository for models under development. Models can be uploaded and downloaded directly through the DynaSim toolbox or through a web browser. Every online model has its own detail page to aid sharing models and having public discussions. Privacy settings enable users to control who can see their models. Citations can be added to models, and all models with citations appear in a “published” model list.

InfiniteBrain.org



name	description	structure
Hodgkin-Huxley Equations	The Hodgkin-Huxley equations are the classic model for action potential propagation in the squid giant axon. They represent a patch of membrane with three channels, sodium, potassium, and a leak. REF -- XPP site: http://www.math.pitt.edu/~bard/bardware/tut/newstyle.html#hh. (node model added by jasonsherfey on Sept. 30, 2014, 11:01 a.m.)	
Morris-Lecar Equations	The Morris-Lecar model is a two-dimensional "reduced" excitation model applicable to systems having two non-inactivating voltage-sensitive conductances. The original form of the model employed an instantaneously responding voltage-sensitive Ca2+ conductance for excitation and a	

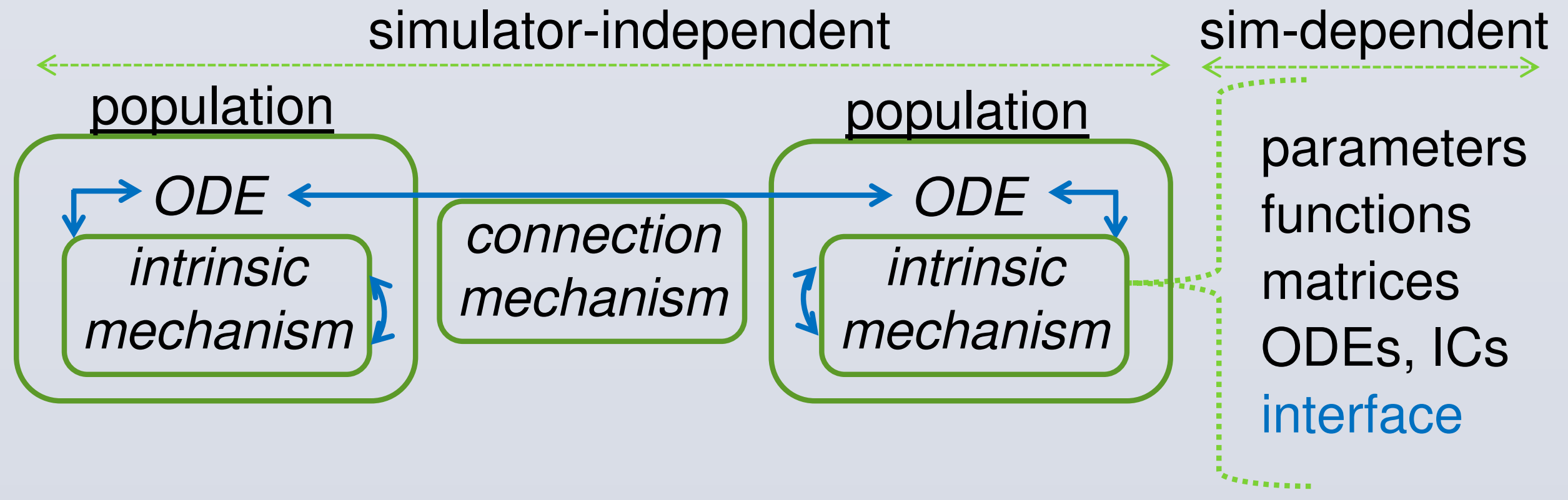
DynaSim/InfiniteBrain Interface



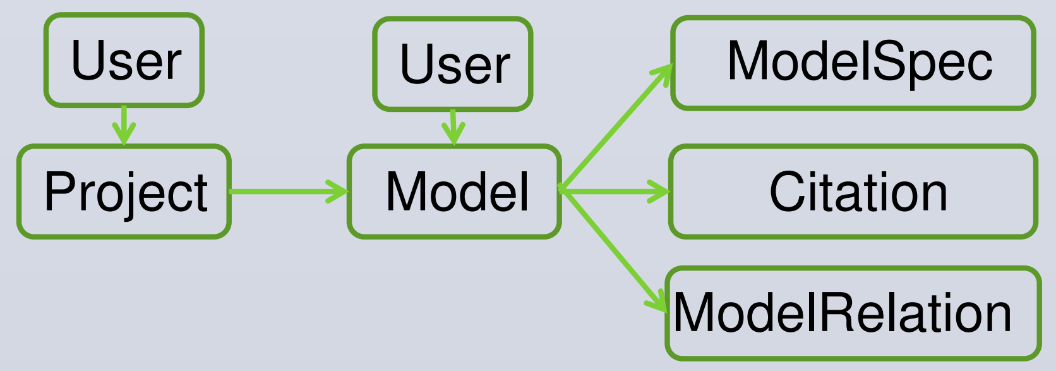
Hodgkin-Huxley Equations
Morris-Lecar Equations
Post-inhibitory Rebound
Morris-Lecar-Neuron
PY-spec-reduced_coupled
Post-Inhibitory-Rebound
Wilson-Cowan-oscillator

```
Model: Morris-Lecar Equations
Site: infinitebrain.org/models/120

Description:
The Morris-Lecar model is a two-dimensional "reduced" excitation model
applicable to systems having two non-inactivating voltage-sensitive
conductances. The original form of the model employed an instantaneously
responding voltage-sensitive Ca2+ conductance for excitation and a delayed
voltage-dependent K+ conductance for recovery. REF -- equations from XPP site:
http://www.math.pitt.edu/~bard/bardware/tut/newstyle.html#mlsrc. description
from http://www.scholarpedia.org/article/Morris-Lecar_model.
```

Components of DynaSim model



InfiniteBrain.org database entities



Limitations of DNSim: (1) lack of explicit spatial representation (limits the spatial complexity of models that can be easily implemented); (2) lack of unit handling (requires users ensure consistency).

Acknowledgments

This work was partially supported by grants NIH-T90DA32484 to JS and NSF DMS-1042134 to NK.