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 https://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

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.

Component of DynaSim model
- ODEs, ICs
- parameters
- matrices
- functions

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.

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 CMS-1042134 to NK.

DynaSim Toolbox

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

Example: Lorenz equations
\[
\begin{align*}
\frac{dx}{dt} &= r \cdot y - x \cdot z \\
\frac{dy}{dt} &= x \cdot y - b \cdot z \\
\frac{dz}{dt} &= -a \cdot x + y \cdot z
\end{align*}
\]

Example: Hodgkin-Huxley-type bursting neuron
\[
\begin{align*}
\frac{dv}{dt} &= I_{\text{app}} + g_Na \cdot m^3 \cdot h \cdot (v-50) \\
\frac{dn}{dt} &= a_N(v) \cdot (1-n) - b_N(v) \cdot n \\
\frac{dh}{dt} &= a_H(v) \cdot (1-h) - b_H(v) \cdot h
\end{align*}
\]

\[
aM(v) = \frac{(2.5 - 0.1 \cdot (v+65))}{(exp(2.5 - 0.1 \cdot (v+65)) - 1)}
\]
\[
aH(v) = 0.07 \cdot exp(-\frac{(v+65)}{20})
\]
\[
bH(v) = \frac{1}{(exp(3 - 0.1 \cdot (v+65))) + 1}
\]

Pyramidal-Interneuron Network rhythm
\[
\begin{align*}
\frac{dv}{dt} &= v \cdot (1 - v) \cdot (v-77) \\
\frac{dm}{dt} &= a_N(v) \cdot (1-m) - b_N(v) \cdot m; m(0)=0.1 \\
\frac{dn}{dt} &= a_M(v) \cdot (1-n) - b_M(v) \cdot n; n(0)=0.1 \\
\frac{dh}{dt} &= a_H(v) \cdot (1-h) - b_H(v) \cdot h; h(0)=0.1
\end{align*}
\]

\[
aN(v) = \frac{0.125 \cdot exp(-\frac{(v+65)}{80})}{exp(3 - 0.1 \cdot (v+65)) + 1}
\]
\[
bN(v) = \frac{0.125 \cdot exp(-\frac{(v+65)}{80})}{exp(3 - 0.1 \cdot (v+65)) + 1}
\]

\[
aM(v) = \frac{(2.5 - 0.1 \cdot (v+65))}{(exp(2.5 - 0.1 \cdot (v+65)) - 1)}
\]

\[
aH(v) = 0.07 \cdot exp(-\frac{(v+65)}{20})
\]
\[
bH(v) = \frac{1}{(exp(3 - 0.1 \cdot (v+65))) + 1}
\]

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 https://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

Using DynaSim GUI

Define/reuse two mechanisms (@current+=INa)

Using DynaSim in Matlab script

PlotData:

Graphical User Interface

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

View full model equations and dynamics during interactive model building

Parameter sweeps and cluster computing

DynaSim/InfiniteBrain Interface

Components of DynaSim model
- simulator-independent
- sim-dependent

InfiniteBrain.org database entities

User
Model
ModelSpec
Project
ModelRelation

infinitiebrain.org

Examples:

\[
\begin{align*}
\text{eqns} &= \{ \text{ode1: } x'=y+z, \text{ode2: } y'=x+z \} \\
\text{data} &= \text{SimulateModel(eqns, } [0 100], \text{ic}=[1 2 .5]);
\end{align*}
\]

Using DynaSim toolbox in Matlab script

Using DynaSim GUI

Cluster computing is as simple as setting an optional flag:

\[
\begin{align*}
&\text{[-, info]=SimulateModel(s', vary, vary, 'cluster_flag', 1); } \\
&\text{PlotData(data(1,:), plot_type, 'raster'); }
\end{align*}
\]

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.

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

GRAPHICAL USER INTERFACE

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

View full model equations and dynamics during interactive model building

Parameter sweeps and cluster computing

DynaSim/InfiniteBrain Interface

Components of DynaSim model
- simulator-independent
- sim-dependent

InfiniteBrain.org database entities

User
Model
ModelSpec
Project
ModelRelation

infinitiebrain.org

Examples:

\[
\begin{align*}
\text{eqns} &= \{ \text{ode1: } x'=y+z, \text{ode2: } y'=x+z \} \\
\text{data} &= \text{SimulateModel(eqns, } [0 100], \text{ic}=[1 2 .5]);
\end{align*}
\]

Using DynaSim toolbox in Matlab script

Using DynaSim GUI

Cluster computing is as simple as setting an optional flag:

\[
\begin{align*}
&\text{[-, info]=SimulateModel(s', vary, vary, 'cluster_flag', 1); } \\
&\text{PlotData(data(1,:), plot_type, 'raster'); }
\end{align*}
\]

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.

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