Brainlab:

A toolkit to aid in the design, simulation, and analysis of spiking neural networks with the NCS environment

Rich Drewes
University of Nevada, Reno

April 26, 2005
Thesis defense

Outline

NCS, the NeoCortical Simulator

- Powerful batch processing spiking neural network simulator
- Parallel (MPI)
- Biologically realistic neuron and synapse models
- Accepts model and simulation parameters in text file

.in file: the BRAIN block

```
BRAIN
    TYPE                          AREA-BRAIN
    FSV                           10000.00
    JOB                           E0_332
    SEED                         -333
    DURATION                      80.0
    COLUMN_TYPE                   col
    COLUMN_TYPE                   datain0
    COLUMN_TYPE                   keyin0
    CONNECT                       datain0 datain0_1CELL ER SOMA
                                  col layER ES SOMA E 1.000 10.000
    CONNECT                       keyin0 keyin0_1CELL ER SOMA
                                  col layL1 ES SOMA E 1.000 10.000
    STIMULUS_INJECT               stimdatain0_0-inj-0
    REPORT                        EMRep
    OUTPUT_CELLS                  YES
    OUTPUT_CONNECT_MAP            YES
    DISTANCE                      YES
END_BRAIN
```

.in file: a COLUMN block

```
COLUMN
    TYPE                          col
    COLUMN_SHELL                  col_sh
    LAYER_TYPE                    layL1
    LAYER_TYPE                    layEM
    LAYER_TYPE                    layER
    LAYER_TYPE                    layES
    LAYER_TYPE                    layI1
    CONNECT                       layL1 ES SOMA layEM ES SOMA
                                  E 0.600 10.000
    CONNECT                       layER ES SOMA layEM ES SOMA
                                  E 1.000 10.000
    CONNECT                       layER ES SOMA layES ES SOMA
                                  E 0.600 10.000
    CONNECT                       layI1 I1 SOMA layES ES SOMA
                                  I 0.600 10.000
END_COLUMN
```

Typical MATLAB-NCS usage

```
fprintf(fid3,'\tAMP_START\t\t'); fprintf(fid3,'%g\n',AmpStart);
fprintf(fid3,'\tAMP_END\t\t\t'); fprintf(fid3,'%g\n',AmpEnd);
fprintf(fid3,'\tWIDTH\t\t\t'); fprintf(fid3,'%g\n',PulseWidth);
fprintf(fid3,'\tTIME_START\t\t'); fprintf(fid3,'%s\n',StimStart);
fprintf(fid3,'\tTIME_END\t\t'); fprintf(fid3,'%s\n',StimEnd);
fprintf(fid3,'\tFREQ_START\t\t'); fprintf(fid3,'%g\n',FreqStart);
```

My MATLAB-NCS experience

- "printf" style model building is limited
    - Easy to make errors
    - Small changes to model can require many changes
- Saving models and then executing them is limiting
    - Prevents strategies like GA search for models
    - Prevents encapsulation of an "experiment as a script"
- There is typically very little code reuse
- MATLAB is not a particularly elegant text processing language
- MATLAB support for general purpose libraries is weak

Design goals for a new approach to using NCS

- Allow very complex experiments with maximum clarity and minimum effort
- Allow very simple experiments with minimum overhead
- A model is a script (there is no other reasonable way)
- Object oriented neural model design
- Integrated modeling, experimentation, analysis
- Container for standard, reusable component libraries.
- Free, and open source, from top to bottom
- Based on a language that is clean, good at text processing as well as math, with extensive general purpose libraries
- Remote or local operation; make remote operation transparent
- Possible interactive use

Design goals for a new approach to using NCS

- Allow very complex experiments with maximum clarity and minimum effort
- Allow very simple experiments with minimum overhead
- A model is a script (there is no other reasonable way)
- Object oriented neural model design
- Integrated modeling, experimentation, analysis
- Container for standard, reusable component libraries.
- Free, and open source, from top to bottom
- Based on a language that is clean, good at text processing as well as math, with extensive general purpose libraries
- Remote or local operation; make remote operation transparent
- Possible interactive use

Design goals for a new approach to using NCS

- Allow very complex experiments with maximum clarity and minimum effort
- Allow very simple experiments with minimum overhead
- A model is a script (there is no other reasonable way)
- Object oriented neural model design
- Integrated modeling, experimentation, analysis
- Container for standard, reusable component libraries.
- Free, and open source, from top to bottom
- Based on a language that is clean, good at text processing as well as math, with extensive general purpose libraries
- Remote or local operation; make remote operation transparent
- Possible interactive use

Design goals for a new approach to using NCS

- Allow very complex experiments with maximum clarity and minimum effort
- Allow very simple experiments with minimum overhead
- A model is a script (there is no other reasonable way)
- Object oriented neural model design
- Integrated modeling, experimentation, analysis
- Container for standard, reusable component libraries.
- Free, and open source, from top to bottom
- Based on a language that is clean, good at text processing as well as math, with extensive general purpose libraries
- Remote or local operation; make remote operation transparent
- Possible interactive use

Design goals for a new approach to using NCS

- Allow very complex experiments with maximum clarity and minimum effort
- Allow very simple experiments with minimum overhead
- A model is a script (there is no other reasonable way)
- Object oriented neural model design
- Integrated modeling, experimentation, analysis
- Container for standard, reusable component libraries.
- Free, and open source, from top to bottom
- Based on a language that is clean, good at text processing as well as math, with extensive general purpose libraries
- Remote or local operation; make remote operation transparent
- Possible interactive use

Design goals for a new approach to using NCS

- Allow very complex experiments with maximum clarity and minimum effort
- Allow very simple experiments with minimum overhead
- A model is a script (there is no other reasonable way)
- Object oriented neural model design
- Integrated modeling, experimentation, analysis
- Container for standard, reusable component libraries.
- Free, and open source, from top to bottom
- Based on a language that is clean, good at text processing as well as math, with extensive general purpose libraries
- Remote or local operation; make remote operation transparent
- Possible interactive use

Design goals for a new approach to using NCS

- Allow very complex experiments with maximum clarity and minimum effort
- Allow very simple experiments with minimum overhead
- A model is a script (there is no other reasonable way)
- Object oriented neural model design
- Integrated modeling, experimentation, analysis
- Container for standard, reusable component libraries.
- Free, and open source, from top to bottom
- Based on a language that is clean, good at text processing as well as math, with extensive general purpose libraries
- Remote or local operation; make remote operation transparent
- Possible interactive use

Design goals for a new approach to using NCS

- Allow very complex experiments with maximum clarity and minimum effort
- Allow very simple experiments with minimum overhead
- A model is a script (there is no other reasonable way)
- Object oriented neural model design
- Integrated modeling, experimentation, analysis
- Container for standard, reusable component libraries.
- Free, and open source, from top to bottom
- Based on a language that is clean, good at text processing as well as math, with extensive general purpose libraries
- Remote or local operation; make remote operation transparent
- Possible interactive use

Design goals for a new approach to using NCS

- Allow very complex experiments with maximum clarity and minimum effort
- Allow very simple experiments with minimum overhead
- A model is a script (there is no other reasonable way)
- Object oriented neural model design
- Integrated modeling, experimentation, analysis
- Container for standard, reusable component libraries.
- Free, and open source, from top to bottom
- Based on a language that is clean, good at text processing as well as math, with extensive general purpose libraries
- Remote or local operation; make remote operation transparent
- Possible interactive use

Design goals for a new approach to using NCS

- Allow very complex experiments with maximum clarity and minimum effort
- Allow very simple experiments with minimum overhead
- A model is a script (there is no other reasonable way)
- Object oriented neural model design
- Integrated modeling, experimentation, analysis
- Container for standard, reusable component libraries.
- Free, and open source, from top to bottom
- Based on a language that is clean, good at text processing as well as math, with extensive general purpose libraries
- Remote or local operation; make remote operation transparent
- Possible interactive use

Hello, world

```
from brainlab import *

brain=BRAIN()
c1=brain.Standard1CellColumn()
c2=brain.Standard1CellColumn()
brain.AddConnect(c1, c2, prob=1.0)
brain.Run()
```

Paramaterized experiments

```
brain=BRAIN({'DURATION': .3})
c1=brain.Standard1CellColumn()
c2=brain.Standard1CellColumn()
syn=brain.syntypes['C.strong']
brain.AddConnect(c1, c2, syn, prob=1.0)
brain.AddSpikeStim(c1, [0.0, .01, .02, .03, .04, .05])
rep=brain.AddSimpleReport('test', c2, 'v', dur=(0, .3))

for maxcond in [0, .1, .2, .3, .4, .5, .6, .7, .8, .9]:
    syn.parms['MAX_CONDUCT']=maxcond

    job='tj%f' % maxcond
    brain.Run({'JOB': job})

    # analyze the results
    res=LoadSpikeData(brain, rep)
    print "G:", maxcond, "spikes:" len(res[0])
```

Algorithmic model building

```
b=BRAIN()

cols={}
for i in range(0, 10):
    for j in range(0, 10):
        cols[i,j]=b.Standard1CellColumn()

# make 1000 random connections, with probability
# proportional to distance
for i in range(0, 1000):
    x1,y1=randrange(0,10), randrange(0,10)
    x2,y2=randrange(0,10), randrange(0,10)
    p=1.0 - ((x1-x2)**2 + (y1-y2)**2) / 200.0
    b.AddConnect(cols[x1,y1], cols[x2,y2], prob=p)

print b     # a 5000 line .in file
b.Run()
```

Why Python?

- Open source, free
- Widely used and growing, active scientific community
- Competitive array math package and plotting packages
- Clean language design
- Object oriented, dynamically typed, garbage collected, bytecode compiled
- Efficient
- Enforced indentation!

Brainlab implementation

- `COLUMN`, `LAYER`, `CELL`, `SYNAPSE` are Python object classes
- These are *nested* classes of `BRAIN` class
- `__repr__()` is overridden to output the `.in` file representation
- NCS parameter names are dictionary keys
- Three modules:
  - brainlab.py: simulation environment, data analysis, graphing
  - brain.py: model construction
  - netplot.py: three dimensional display

Brainlab implementation

- `COLUMN`, `LAYER`, `CELL`, `SYNAPSE` are Python object classes
- These are *nested* classes of `BRAIN` class
- `__repr__()` is overridden to output the `.in` file representation
- NCS parameter names are dictionary keys
- Three modules:
    - brainlab.py: simulation environment, data analysis, graphing
    - brain.py: model construction
    - netplot.py: three dimensional display

Brainlab implementation

- `COLUMN`, `LAYER`, `CELL`, `SYNAPSE` are Python object classes
- These are *nested* classes of `BRAIN` class
- `__repr__()` is overridden to output the `.in` file representation
- NCS parameter names are dictionary keys
- Three modules:
  - brainlab.py: simulation environment, data analysis, graphing
  - brain.py: model construction
  - netplot.py: three dimensional display

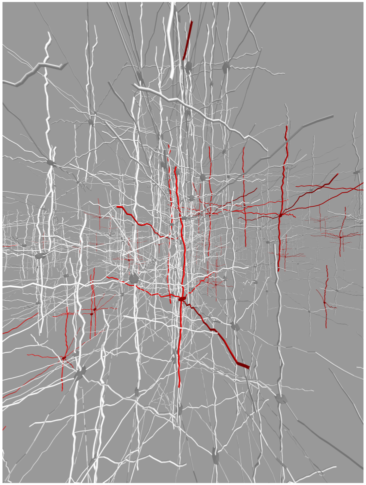
Brainlab implementation

- `COLUMN`, `LAYER`, `CELL`, `SYNAPSE` are Python object classes
- These are *nested* classes of `BRAIN` class
- `__repr__()` is overridden to output the `.in` file representation
- NCS parameter names are dictionary keys
- Three modules:
  - brainlab.py: simulation environment, data analysis, graphing
  - brain.py: model construction
  - netplot.py: three dimensional display

Brainlab implementation

- `COLUMN`, `LAYER`, `CELL`, `SYNAPSE` are Python object classes
- These are *nested* classes of `BRAIN` class
- `__repr__()` is overridden to output the `.in` file representation
- NCS parameter names are dictionary keys
- Three modules:
  - brainlab.py: simulation environment, data analysis, graphing
  - brain.py: model construction
  - netplot.py: three dimensional display

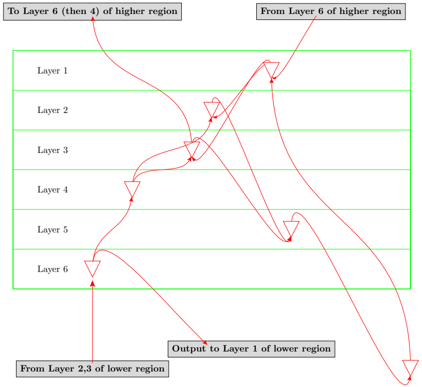
netplot 3D module



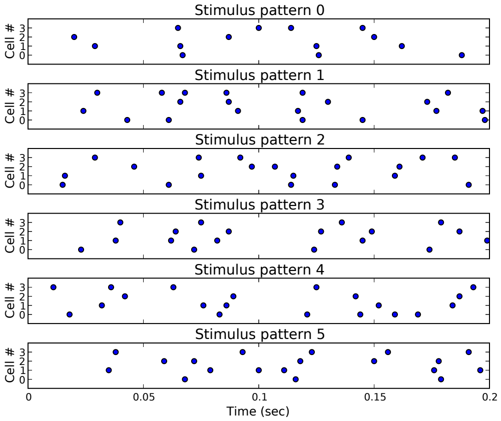
My Ph.D. research

Could a cortical microcircuit act as a functional unit for the memorization of correlations of spatio-temporal spike trains?

- Layer structured cortical microcircuit model
- Two input cell groups
    - Key
    - Data
- Output cell group
- Spatio-temporal spike trains as fundamental unit of information


To Layer 6 (then 4) of higher region
From Layer 6 of higher region
Layer 1
Layer 2
Layer 3
Layer 4
Layer 5
Layer 6
Output to Layer 1 of lower region
From Layer 2,3 of lower region

Stimulus patterns



My Ph.D. research: genetic model search

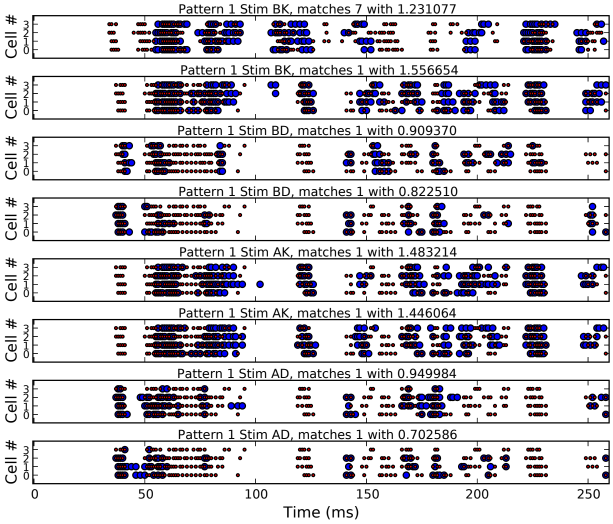
```
genelist=[]
genelist.append((ga.gene.list_gene([2,3,4,5,6,7,8]), 'ntapecells'))
genelist.append((ga.gene.float_gene((.005, .05)), 'allmaxcond'))
genelist.append((ga.gene.float_gene((.005,.3)), 'allF'))
genelist.append((ga.gene.float_gene((.005,.3)), 'allD'))
genelist.append((ga.gene.float_gene((.005,.3)), 'poshebbdeltause'))
genelist.append((ga.gene.float_gene((.005,.3)), 'neghebbdeltause'))
genelist.append((ga.gene.list_gene([(float(x)/1000.0) \
                    for x in range(200, 801, 100)]), 'stimdur'))
genelist.append((ga.gene.list_gene(range(20, 120, 10)), 'interspike'))
genelist.append((ga.gene.float_gene((.05, 1.0)), 'eresprob'))
genelist.append((ga.gene.float_gene((.05, 1.0)), 'eremprob'))
genelist.append((ga.gene.float_gene((.05, 1.0)), 'l1emprob'))
genelist.append((ga.gene.float_gene((.05, 1.0)), 'esthalprob'))
genelist.append((ga.gene.float_gene((0.0, 1.0)), 'L1L5prob'))
genelist.append((ga.gene.float_gene((0.0, 1.0)), 'L6L4prob'))

all_genes=[]
for (g, parmname) in genelist:
    all_genes+=g.replicate(1)
    mychrom.append(parmname)
```

My Ph.D. research: genetic model search

```
this_genome.performance=EOFitness
gnm=this_genome(all_genes)
pop=ga.population.population(gnm)
pop.evaluator=my_pop_evaluator()
galg=ga.algorithm.galg(pop)
settings={'pop_size':16,'p_replace':.8,'p_cross': .8, \
          'p_mutate':'gene', 'p_deviation': 0.,'gens':64,\
          'rand_seed':0,'rand_alg':'CMRG'}
galg.settings.update(settings)
galg.evolve()
print "best:", galg.pop.best()
```

My Ph.D. research: automated pattern matching



Phase space transitions in a recurrent network

Dr. Doursat:

"My goal is to explore bistability and other firing modes of the network through a survey of parameter space (especially E/I to E/I connections) and, from there, reveal what difference synaptic augmentation (SA) and/or reciprocal connections can make (changes in phase space landscape; easiness of phase transitions) when they are added to the model."

Setting up the experiment in Brainlab

- Convert plain .in file into a script
- Create experiment script
- Create data analysis/graphing

Tighter integration with NCS

Idea: eliminate NCS parsing module, and most of NCS' biological knowledge

- Simpler code; just a "flat" simulation engine that accepts GCList
- Less redundancy of code and documentation
- NCS would be more easily used with unstructured networks
- Easier to add new features

Short-term enhancements to Brainlab

- Some BRAIN.methods become brainlab functions
- Data loading and report plotting standardized, expanded
- Subdirectories for job execution (at least as an option)
- Support for new queueing environments
- Convenience functions for executing multiple jobs
- Automated installation, configuration, testing
- Expanded examples library

Use, so far

- A colleague in our lab used an earlier version of Brainlab
- I have made very extensive use of Brainlab in my own experiments
- A colleague in our lab has recently begun using Brainlab in his experiments
- Another lab is beginning to use NCS, and expressed interest in Brainlab

Acknowledgements

Thanks to:

- My committee
- Dr. Harris, chair
- Dr. Goodman, for the Brain Lab
- Brain Lab colleagues