Exceptional service in the national interest

Sandia National Laboratories

Staying on the path

Fred Rothganger

U.S. DEPARTMENT OF ENERGY
NNSA National Nuclear Security Administration
LDRD LABORATORY DIRECTED RESEARCH & DEVELOPMENT

Sandia National Laboratories is a multi-mission laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000. SAND NO. 2011-XXXXP

Problems

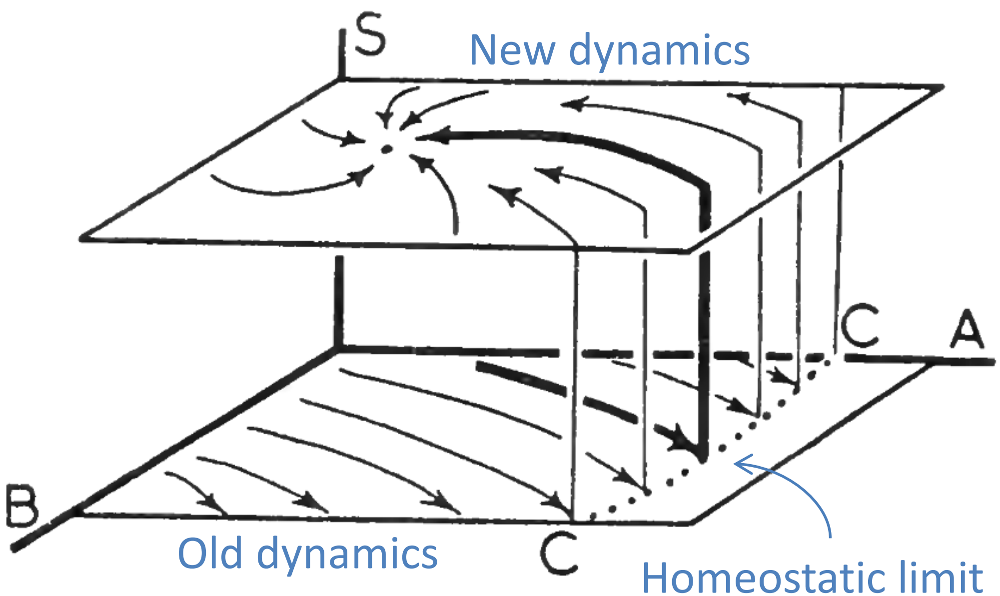
Neuro-inspired computing faces three key challenges:

1) Reducing energy per bit—how to produce reliable results with unreliable hardware?
2) Creating algorithms to solve problems currently only solved by the human brain.
3) Porting algorithms between substantially different architectures.

➡ Computing with dynamical systems offers potential solutions to all three problems.

Nature computes with dynamical systems

Biological systems respond to a problem by making internal changes to their dynamics. This can be called "learning".

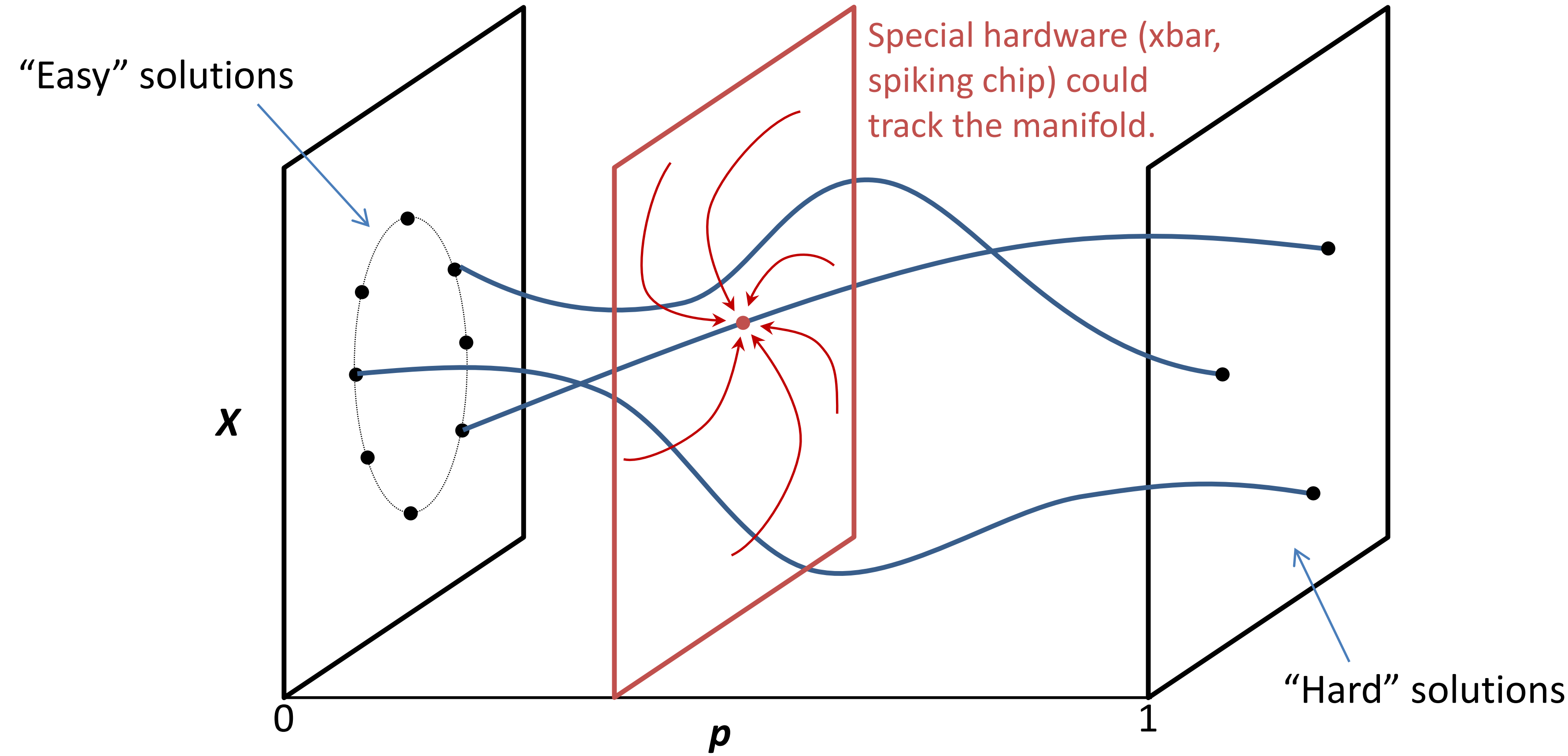


Homeostat – a biologically-inspired computer
W. Ross Ashby, pioneer in cybernetics, 1947

Dynamical systems *can* solve numerical problems

$$f(\boldsymbol{X}, p) = (1-p)e(\boldsymbol{X}) + ph(\boldsymbol{X}) = 0$$

- Nonlinear systems
- Complex polynomial systems
  → Eigenvalue problems



By construction, easy and hard solutions are always on the same manifold.
Unlike an ODE, *f* always tells us distance to the manifold.

Eigenvalue problem

A – square matrix of arbitrary complex values
D – diagonal matrix of "random" values. These are initial eigenvalues λ, and columns of the identity matrix are the associated eigenvectors $\boldsymbol{x}$.

Definition of eigenpair ($\boldsymbol{x}$,λ): $\boldsymbol{Ax} = \lambda\boldsymbol{x}$

Homotopy formulation: $$f(\boldsymbol{x}, \lambda, p) = \begin{bmatrix} \big((1-p)\boldsymbol{D} + p\boldsymbol{A}\big)\boldsymbol{x} - \lambda\boldsymbol{x} \\ 1 - \boldsymbol{x}^H\boldsymbol{x} \end{bmatrix} = \boldsymbol{0}$$

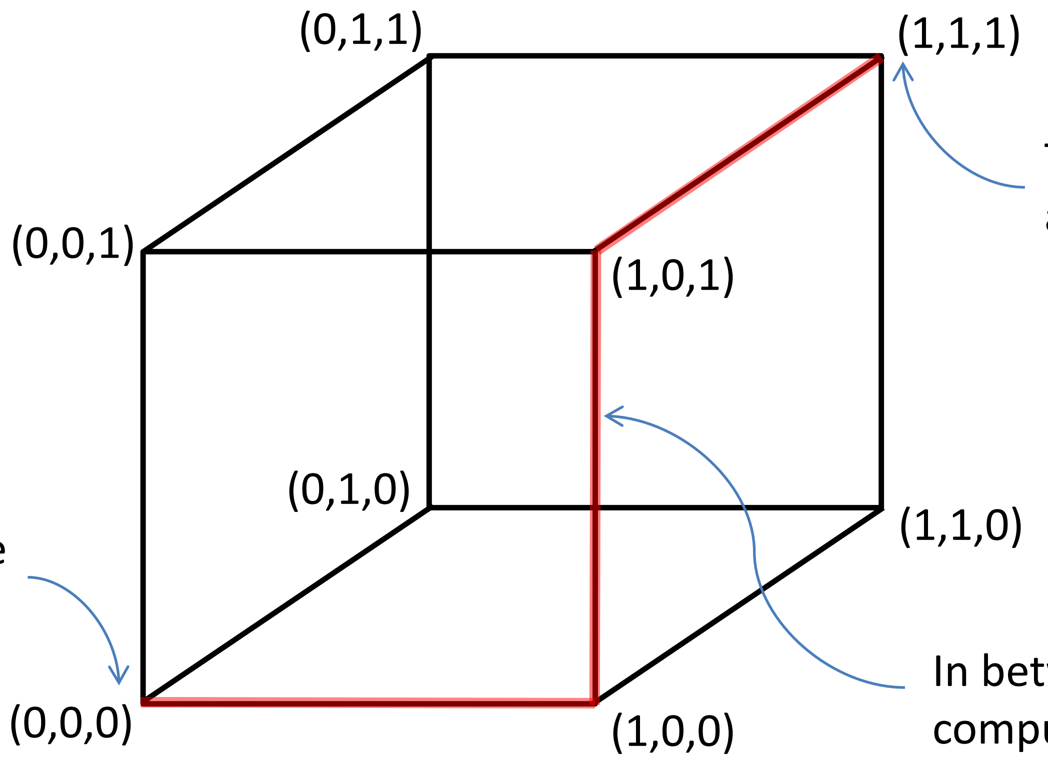
Jacobian: $$\partial f(x, \lambda, p) = \begin{bmatrix} (1-p)\boldsymbol{D} + p\boldsymbol{A} - \lambda & -\boldsymbol{x} & (\boldsymbol{A} - \boldsymbol{D})\boldsymbol{x} \\ -2\boldsymbol{x}^T & 0 & 0 \end{bmatrix}$$

Gradient of f^2: $$\nabla f(x, \lambda, p) = 2(\partial f)^T f$$

Secant method for prediction.
Gradient descent with backtracking line search for correction.

Digital computers are dynamical systems

Imagine a trillion-dimensional hypercube …

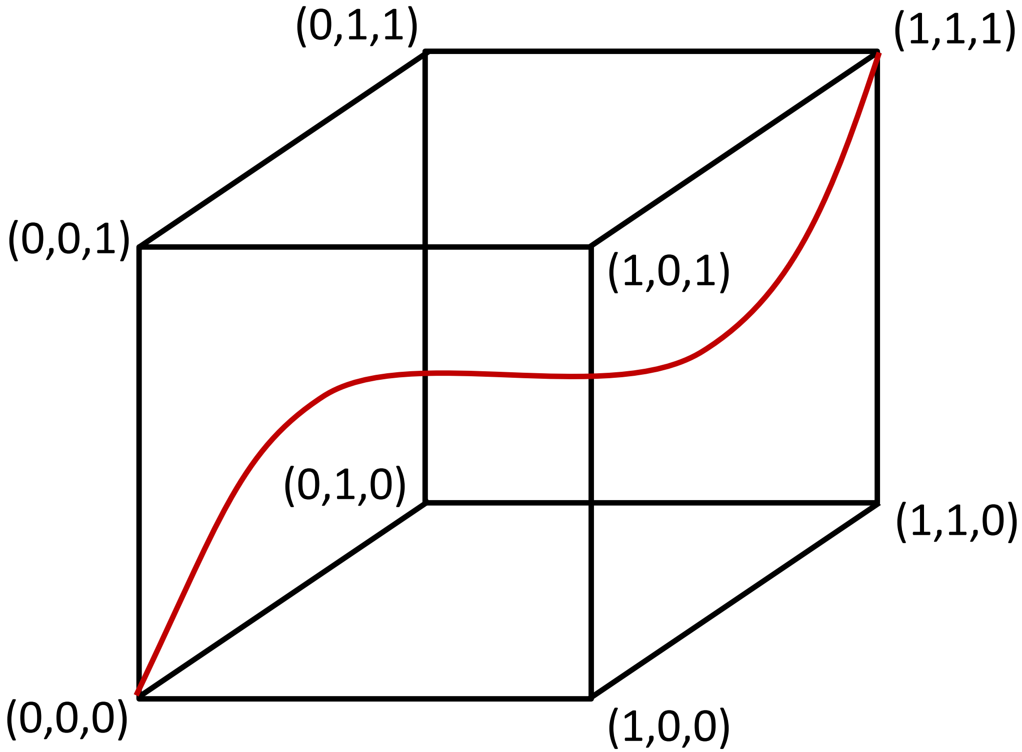


The answer is another corner.

The problem is some corner in this space.

In between, the computer jumps from one corner to another based on its logic.

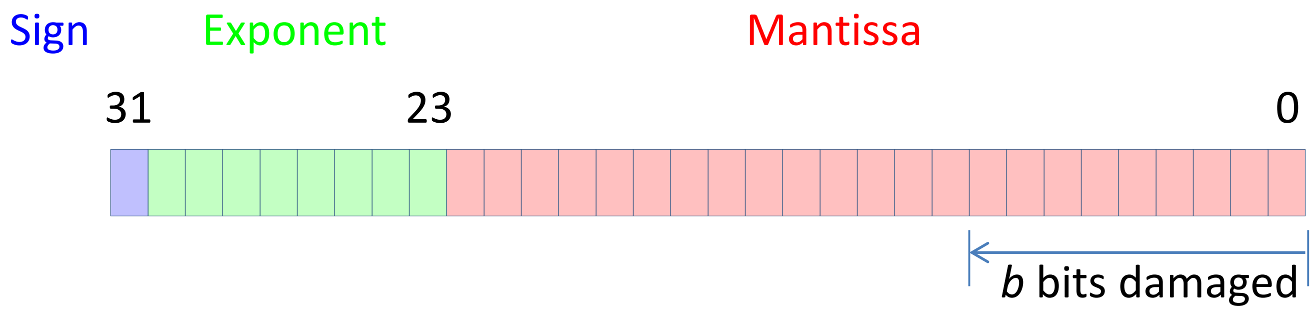
Leave the edges of the binary hypercube?



Backup

FlakyFloat

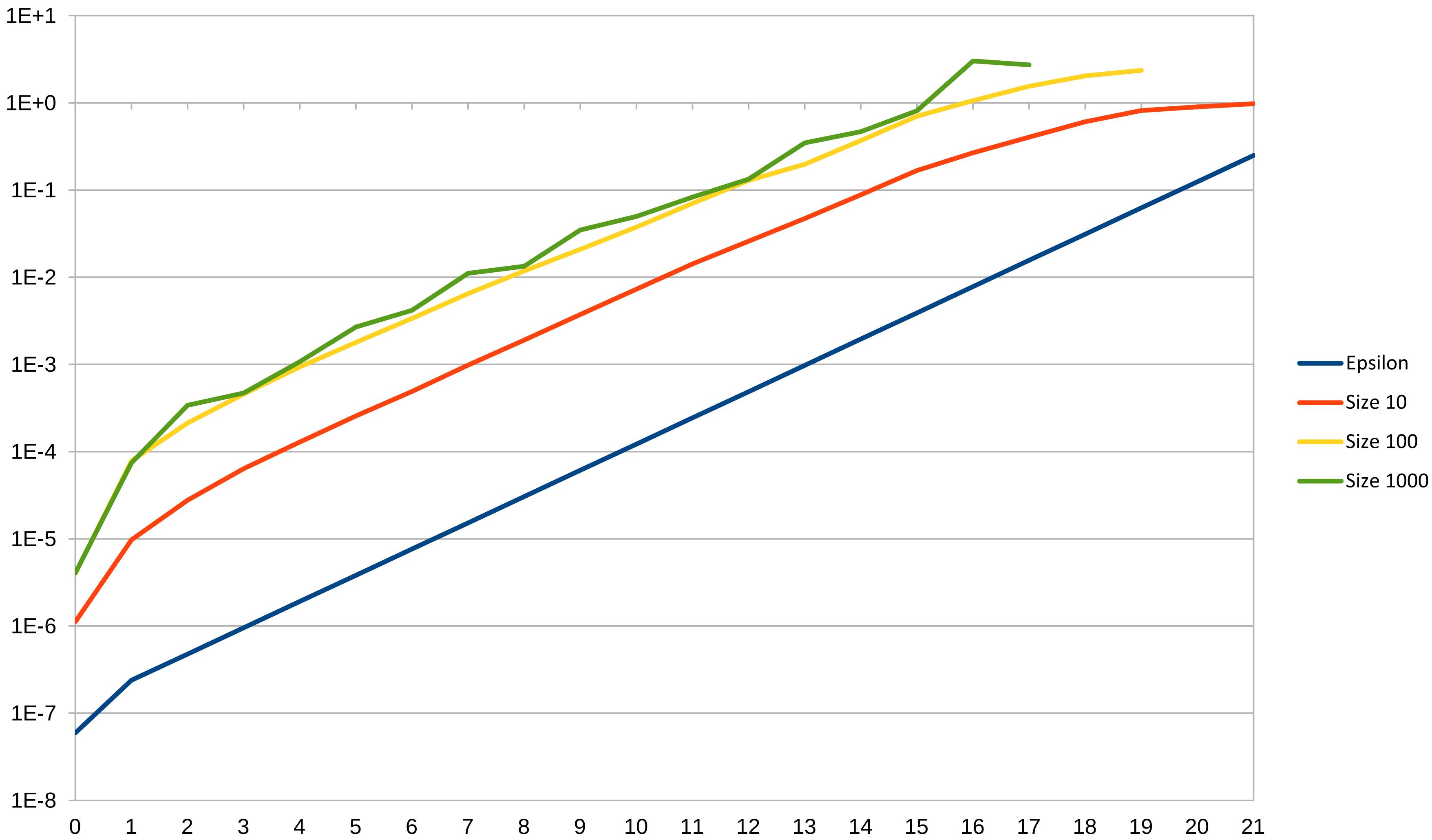
Compiled CLAPACK-3.2.1 with g++
Replaced FORTRAN REAL type with C++ class FlakyFloat



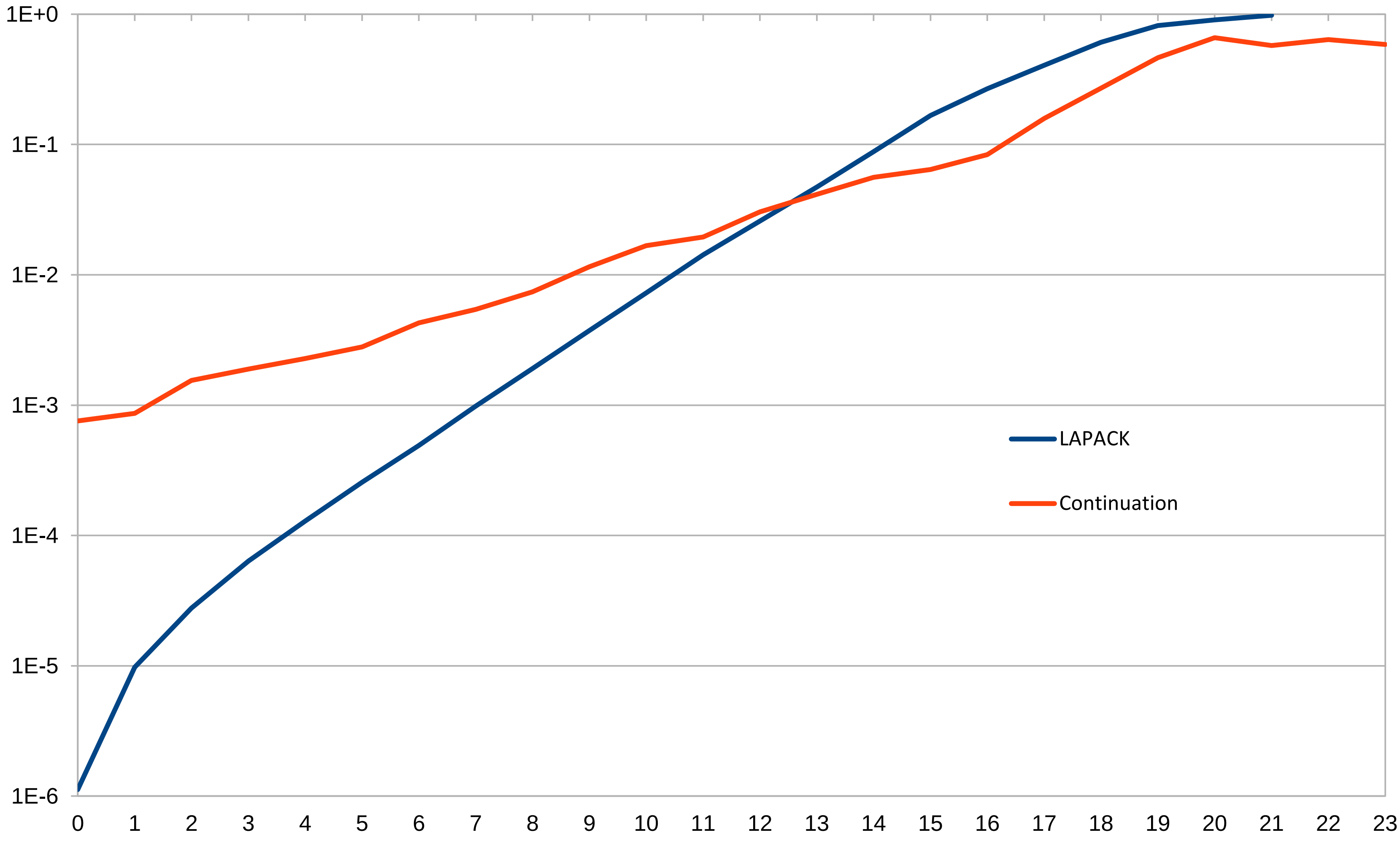
Damage methods:

1. Set to 0
2. Set to random value

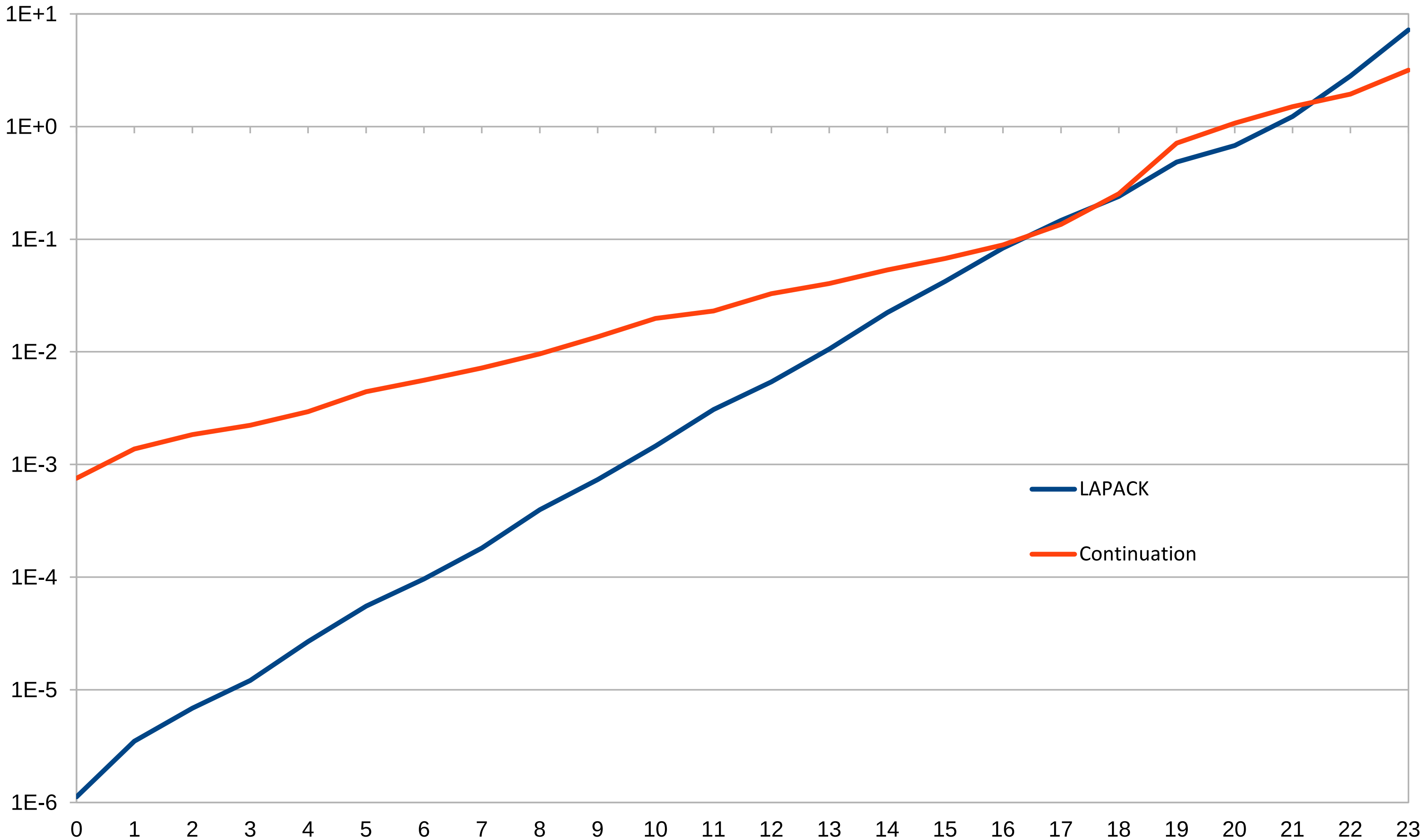
LAPACK under degradation (zeros)



LAPACK vs. Continuation (zeros)

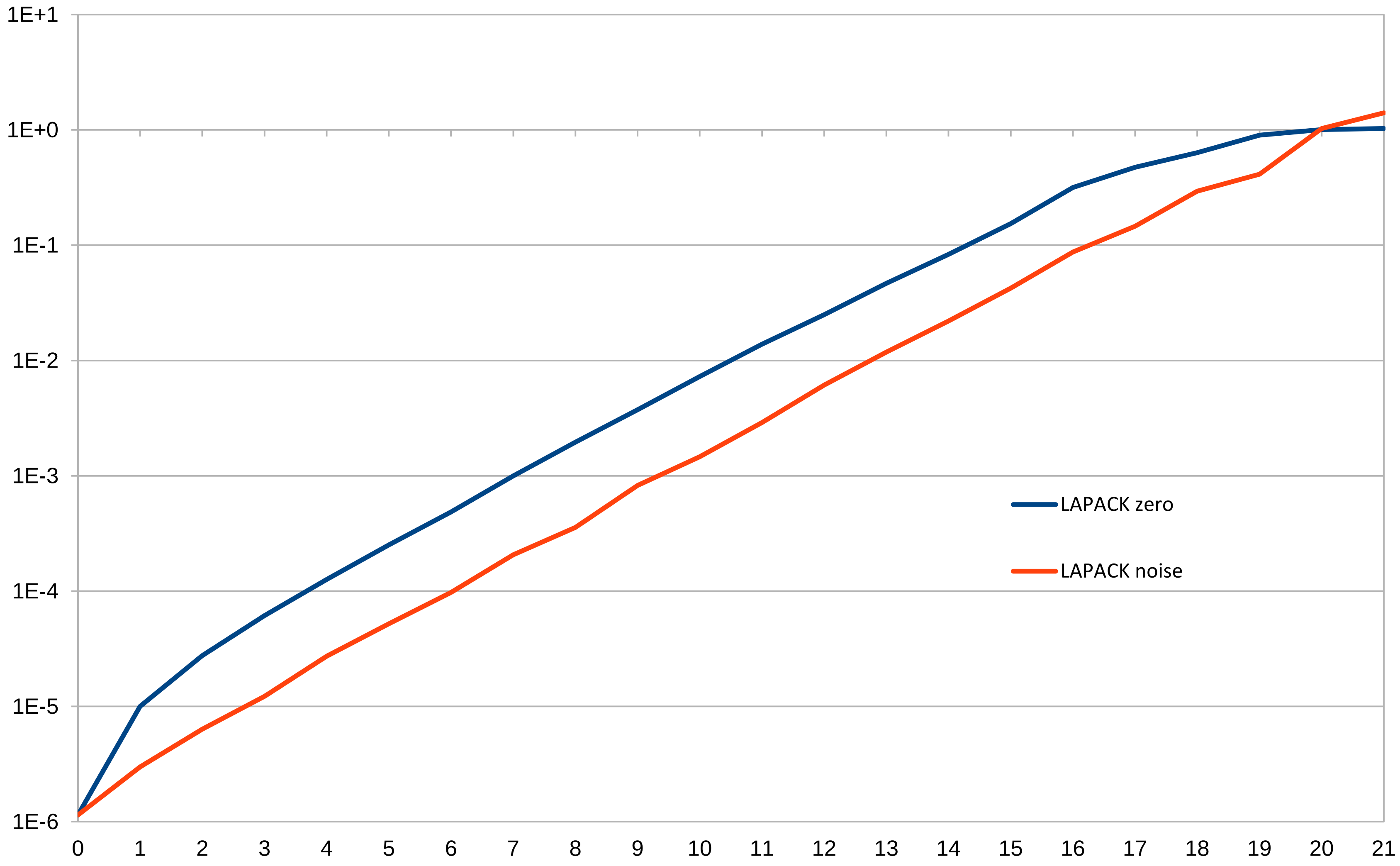


LAPACK vs. Continuation (random)



LAPACK crashed immediately in SLAMCH, so waited to turn on noise until after first call.

LAPACK zeros vs. random



Conclusion

LAPACK is faster and more precise than continuation

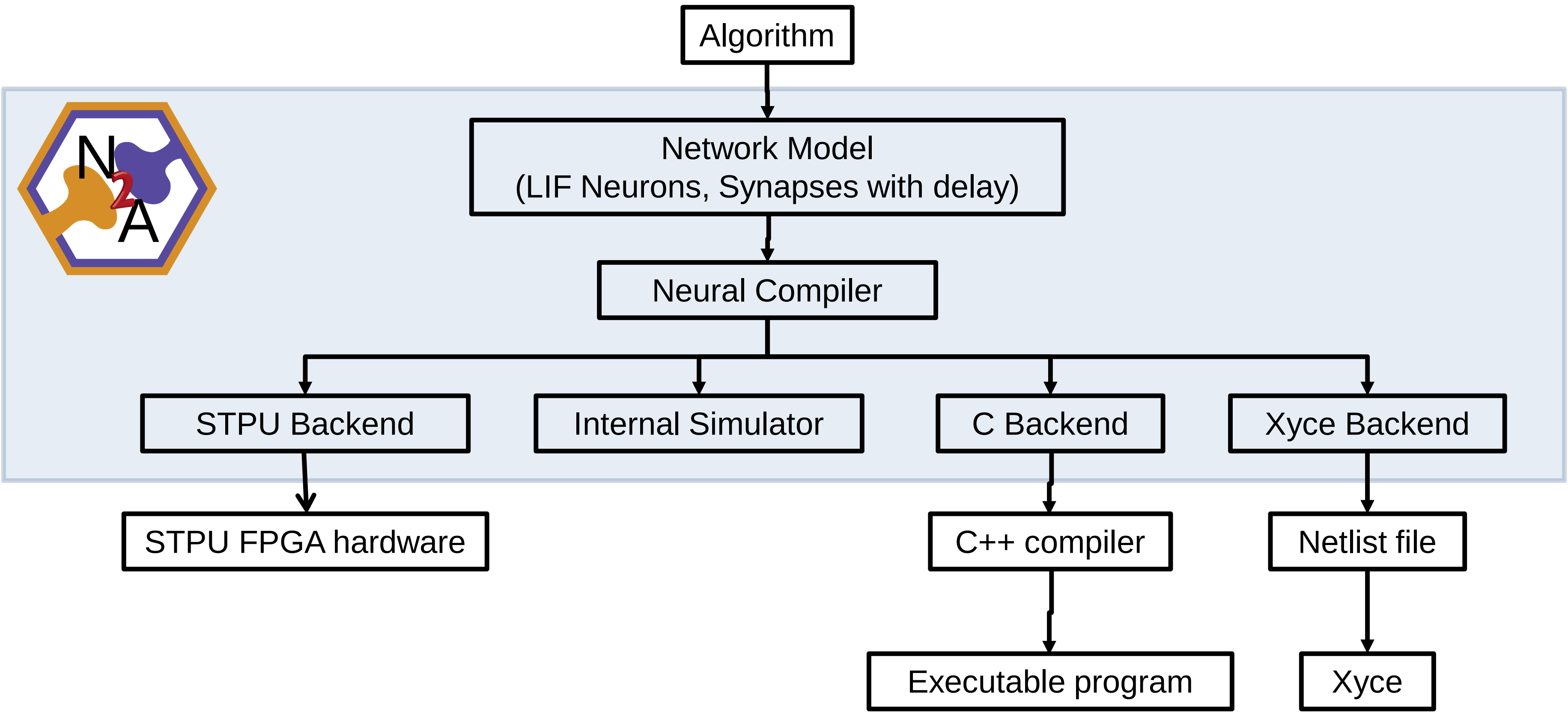
Possibly due to naïve implementation

Continuation is more resilient to noise and loss of precision

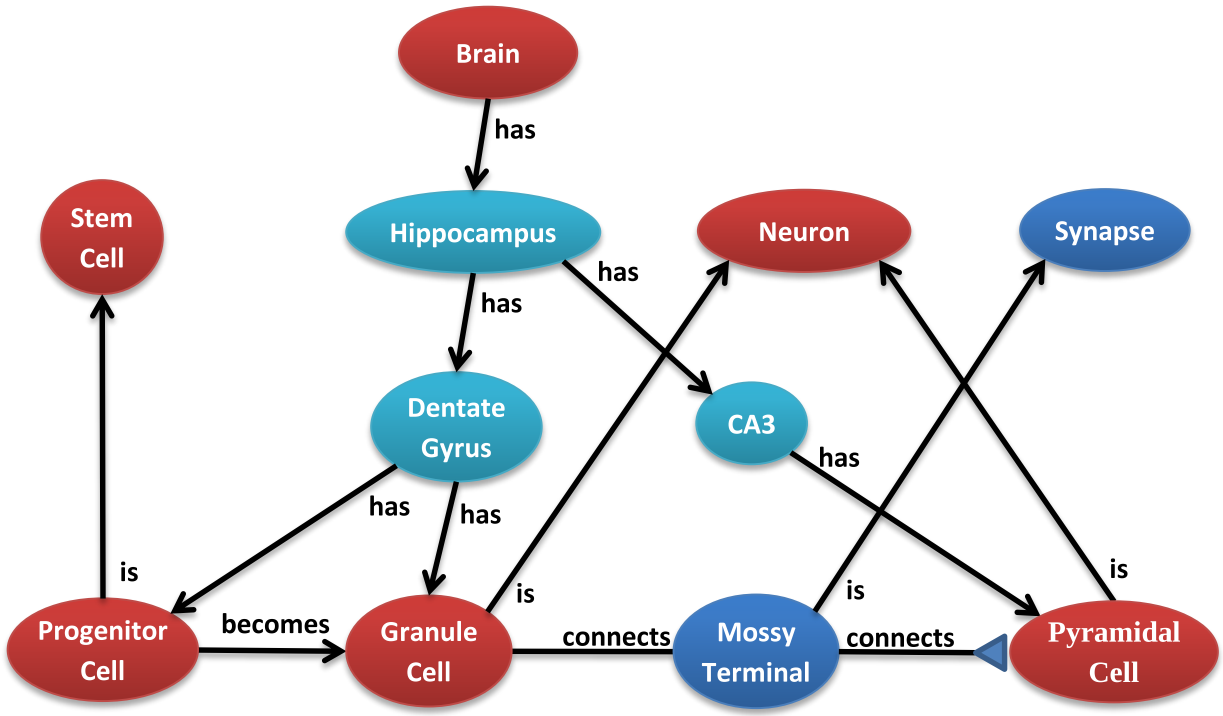
Next steps:

- Actually implement continuation as a dynamical system
  - Small modification to gradient descent
  - Attractor towards p=1
  - Suitable for memristor crossbar – matrices are fixed, or in some cases linearly combined.
- Develop spiking version
- Test on hardware
  - We are assembling a neuromorphic testbed with one of everything: TrueNorth (at LLNL), SpiNNaker, STPU, etc.

N2A – A neural programming language



N2A – Unified modeling framework



Attributes

Name = Hippocampus CA3 pyramidal call
Organism = Vertebrata
Neurotransmitter released = Glutamate
Dendrite Length = 12481.9 +- 2998.9 um

Equations (a type of attribute) → V' = (G * (V_rest - V) + I_inj) / C
G = 0.3
...