Staying on the path

Fred Rothganger
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.
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

\[ f(X,p) = (1 - p) e(X) + ph(X) = 0 \]

- Nonlinear systems
- Complex polynomial systems → Eigenvalue problems

Special hardware (xbar, spiking chip) could track the manifold.

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 \( \lambda \), and columns of the identity matrix are the associated eigenvectors \( x \).

Definition of eigenpair \((x, \lambda)\): \( Ax = \lambda x \)

Homotopy formulation: 
\[
\begin{bmatrix}
(1 - p)D + pA \\
1 - x^H x
\end{bmatrix}
\begin{bmatrix}
x - \lambda x
\end{bmatrix} = 0
\]

Jacobian: 
\[
\partial f(x, \lambda, p) = 
\begin{bmatrix}
(1 - p)D + pA - \lambda \\
-2x^T
\end{bmatrix}
\begin{bmatrix}
x \\
(A - D)x
\end{bmatrix}
\]

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 problem is some corner in this space.

The answer is another corner.

In between, the computer jumps from one corner to another based on its logic.
Leave the edges of the binary hypercube?
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
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

- Algorithm
- Network Model (LIF Neurons, Synapses with delay)
- Neural Compiler
  - STPU Backend
  - Internal Simulator
  - C Backend
  - Xyce Backend
    - STPU FPGA hardware
    - C++ compiler
    - Netlist file
    - Executable program
    - Xyce
N2A – Unified modeling framework

Attributes

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

Equations

\[ V' = \frac{(G \times (V_{rest} - V) + I_{inj})}{C} \]

G = 0.3

...