REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
15-08-2017	Technical Paper	15-08-2016 – 15-08-2017

4. TITLE AND SUBTITLE

Using a Double-Well Oscillator to Train Binary Neural Networks

5a. CONTRACT NUMBER N/A

5b. GRANT NUMBER N/A

5c. PROGRAM ELEMENT NUMBER N/A

6. AUTHOR(S)

1st Lt Aron Wing, USAF; Rose Rustowicz

5d. PROJECT NUMBER N/A

5e. TASK NUMBER N/A

5f. WORK UNIT NUMBER N/A

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AND ADDRESS(ES)

Air Force Research Laboratory
Information Directorate
525 Brooks Road, Rome, NY 13441

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

N/A

10. SPONSOR/MONITOR'S ACRONYM(S) N/A

11. SPONSOR/MONITOR'S REPORT NUMBER(S) N/A

12. DISTRIBUTION / AVAILABILITY STATEMENT

DISTRIBUTION A; Distribution is unlimited

13. SUPPLEMENTARY NOTES

N/A

14. ABSTRACT

Certain processors specifically designed for neural networks are defined with low-precision weights and activations. Low-precision weights and activations can considerably reduce the power required for computing a neural network. Although the neural network is still capable of generalizing the data to determine relevant features, the loss of precision in the values sometimes results in considerable loss of test accuracy. In this paper, we explore using a dynamical system which introduces transient chaos to the loss function that helps train binary network layers. Implementing theoretical stochastic rounding probabilities on the MNIST data set we improved the test error to state of the art for a binary network. We also show that adding a dynamical equation to the loss function of a network can effectively binarize a network.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF: UNCLASSIFIED			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE	Distribution A	19	Aron Wing
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED			19b. TELEPHONE NUMBER *(include area code)* (315) 330-4424

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

Using a Double-Well Oscillator to Train Binary Neural Network Layers

2d Lt Aron Wing

Miss Rose Rustowicz

Air Force Research Laboratory- Information Directorate

15 August 2017

Abstract

Certain processors specifically designed for neural networks are defined with low-precision weights and activations. Low-precision weights and activations can considerably reduce the power required for computing a neural network. Although the neural network is still capable of generalizing the data to determine relevant features, the loss of precision in the values sometimes results in considerable loss of test accuracy. In this paper, we explore using a dynamical system which introduces transient chaos to the loss function that helps train binary network layers. Implementing theoretical stochastic rounding probabilities on the MNIST data set we improved the test error to state of the art for a binary network. We also show that adding a dynamical equation to the loss function of a network can effectively binarize a network.

1 Introduction

1.1 Motivation

The equation for this paper was developed in order to provide a means with which one could entice activations of neurons in a neuromorphic machine learning algorithm to have binary representations. Neuromorphic processors do not have the floating point precision of GPUs and instead rely on generalizing feature spaces to predict classifications. These feature spaces are transduced from the original format to a processor specific readable format. In this paper the desired format is binary. Because of this motivation we assume the initial conditions of the system have been normalized between 0 and 1 and we will analyze the system using a dense uniform distribution of initial conditions to provide global dynamics without loss of generality.

1.2 Initial System Set-Up

The idea is that if we add error to the system for not having a binary representation, then we may be able to entice it to the desired representation. We began by finding an equation for which one could define the y-axis as the error we will add to the system and the x-axis as the value of an activation or weight in the system. Since we want to entice the activations to train toward binary, we want the error introduced by the equation to be minimum at 0 and 1. We hope that by doing this, the machine learning algorithms will train the values to the lowest error and therefore near 0

or 1. We also wanted to ensure that the system is defined for all values and that the system is smooth and continuous. There are many possibilities, but we started with the simplest one that came to mind:

$$f(x) = (x)(x-1)(2x-1) \qquad \text{Equation 1.1}$$

However, it is clear from figure 1.1 that for every x this system does nothing but round to either 0 or 1. Taking the integral:

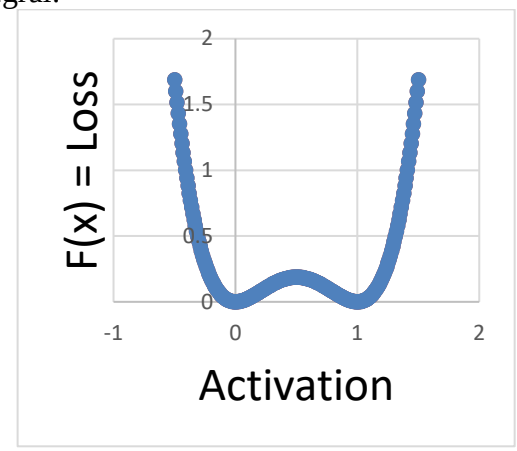


Fig. 1.1

Note: One can naively determine the end-state dynamics of this system by imagining rain drops falling on the figure and rolling down-hill to the bottom of the wells

In order to develop more complicated system behavior, we introduced two new states to the system, namely:

$$f(x) = (x)(x-1)^2 \quad \& \quad f(x) = (x^2)(x-1)$$

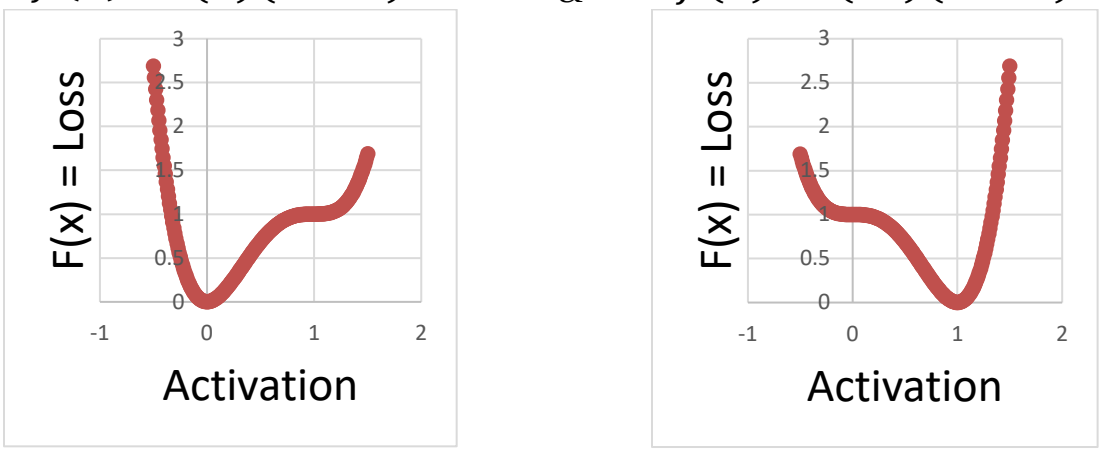


Fig. 1.2

Figure 1.2: These states allow for the system to reach 0 or 1, respectively, for any initial condition.

These three states were combined to provide the following two variable dynamical equation:

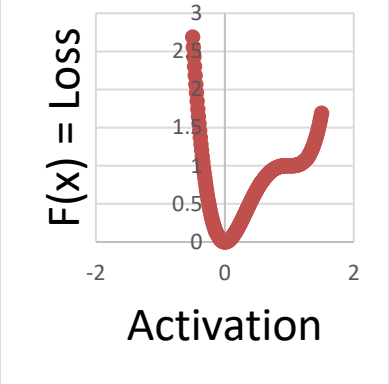
$$f(x,t) = h((x)(x-1)(2x+\sin(t)-1))$$

Equation 1.2

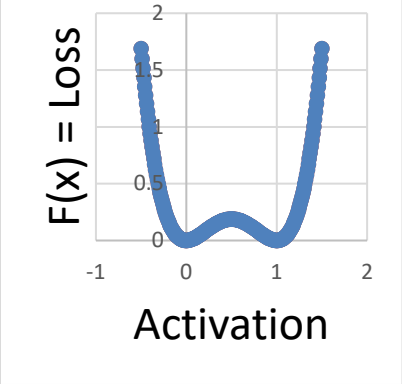
Integrating represents the error introduced by our system. We have obtained our desired system where H = $\frac{h}{6}$ controls the "peakiness" of the system and t is a free time-parameter that oscillates the "hill" across the system. We will progress t as a function of the step during training of a machine learning algorithm. C is a control parameter used to maintain the structure of the system. That is, C = $\frac{h}{6}$ ∗sin(t) if sin(t) > 0 else C = 0. The amount defined by equation 1.3 will be the amount of error we introduce.

$$F(x,t) = H(x)^2[(2x-3)\sin(t) + 3((x-1)^2)] + C$$ Equation 1.3

Sin(t) = -1



Sin(t) = 0



Sin(t) = 1

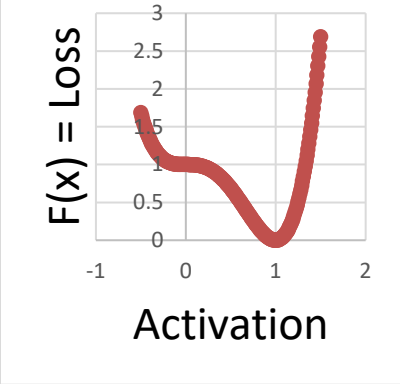


Fig. 1.3

1.3 Initial Observations

We have constructed a system defined by equation 1.3 which we call a modified double-well oscillator.

(1) It is obvious from this construction that 0 and 1 are, generally, stable orbits of the system. If the free parameter t is slow the system will predictably approach the stable orbits of 0 or 1 as the system "sloshes" back-and-forth.
(2) If the parameter H is too large, the "sloshing" will be extreme and the system will become unstable.

1.4 Initial Questions

(1) Is this system always predictable, that is does there exist sensitive dependence on the initial conditions (SDIC)?
(2) How large is "too" large for parameter h?
(3) Are there parameters for which there exists chaos?
(4) Can this function be added to the loss function, or error, of a neural network to train binary layers?

2 Modeling and Simulation

All following simulations in this paper were conducted using a first order Euler approximation map to the system. That is

$$x_{n+1} = x_n - \varepsilon f(x_n, t_n) \qquad \text{Equation 2.1}$$

Where f is equation 1.2, ε = .01 and $tStep = t_{n+1} - t_n$, so that

$$x_{n+1} = x_n - \varepsilon h((x_n)(x_n - 1)(2x_n + \sin(t_n) - 1)) \qquad \text{Equation 2.2}$$

Note that, the selection of different epsilon is not necessary as it is h and tStep that dominate the dynamical behavior of this system; that is, we could always set a constant α and let $\varepsilon =$ Eα then setting $h := \frac{h}{\mathrm{E}}$ returns equation 2.1 with epsilon fixed at α. The overall dynamics will be conserved, just the values of h for which they occur will change. Whenever we refer to F, or Equation 1.3 from this point on in this paper, we mean the approximation based on Equation 2.2.

Notation presented in this paper: $x^i{}_j$ refers to the i[th] activation or weight at the j[th] step.

2.1 Initial Observation Simulations

First we let the initial value of t, t(0), and tStep be constants and raised the "peakiness" parameter, h, slowly until the dynamics of the system change.

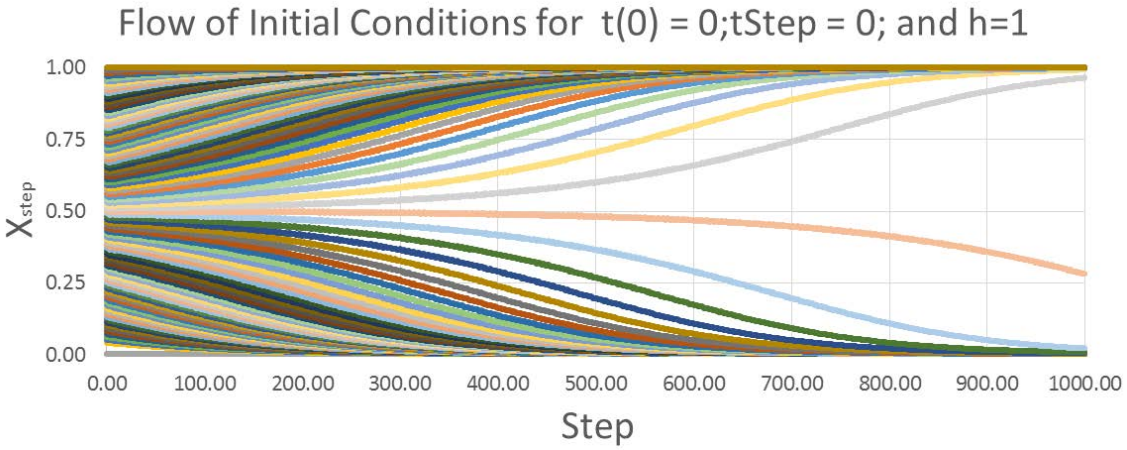


Fig. 2.1

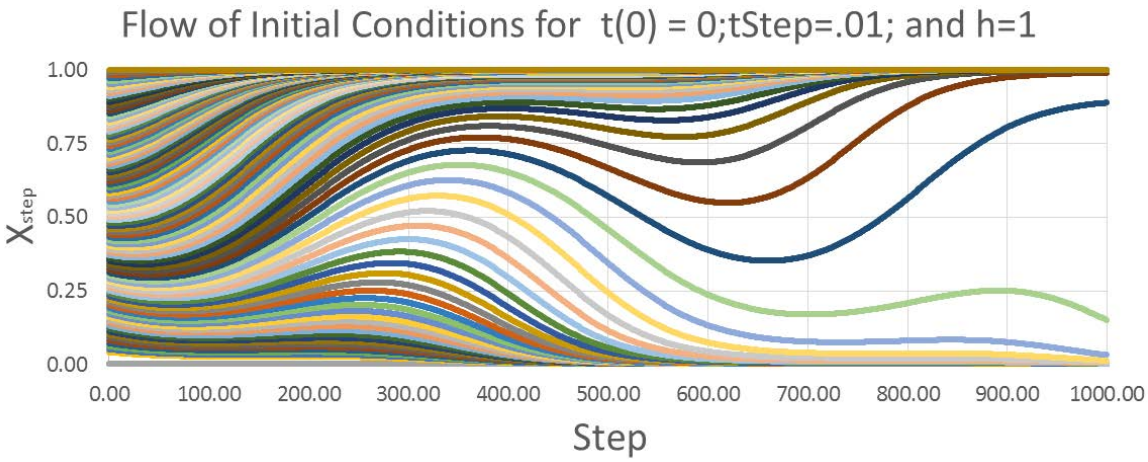


Fig. 2.2

Figures 2.1 and 2.2 show that for small t and h, all initial conditions approach the stable orbits 0 or 1 predictably based on how t is initialized. However, if we continue to increase h the behavior of the system will become unstable. We did not prove, but feel it is safe to assume based on simulation, that there exists some t(0) and tStep values for which $x^i{}_\infty$ approaches $\pm\infty$. Section 3 explores the stable regions of F in more detail.

2.2 Initial Questions Simulations

This section will begin to explore system stability which will lead into system states for which there exists SDIC, chaos and transient chaos. We slowly let h grow we noticed that the system dynamics did not differ from figures 2.1 and 2.2 until around h = 75 where the trajectories begin to peak over the stable orbits at 0 and 1. However, at h =75 the end state is still predictable and there are no periodic orbits (see figure 2.3). The system maintains this structure until about h = 160 where the trajectories peak over the stable orbits, then oscillate back across to the opposite attractor (see figure 2.4). It is at about h = 160 that the system begins to exhibit SDIC (see figure 2.5). Through simulation we determine that the maximum h for any random t(0) and tStep is approximately 190.5

Although the dynamics have changed considerably and have shown signs of transient chaos (fig. 2.5), the stable orbits at 0 and 1 still dominate in the system and attract all trajectories as stable points. This behavior goes away around h=170 when the system begins to generate period 8 stable orbits around 0 and 1 for tStep = pi/2 (see figure 2.6). 53 of 100 trajectories from randomly initialized conditions are shown in figure 2.6; the other 47 trajectories have period 8 orbits around the stable orbit at 1.

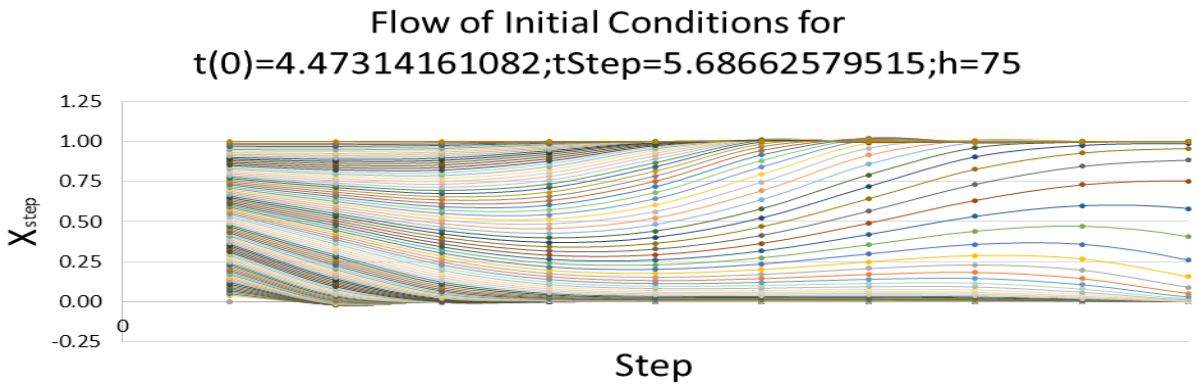


Fig. 2.3 Stable predictable flow

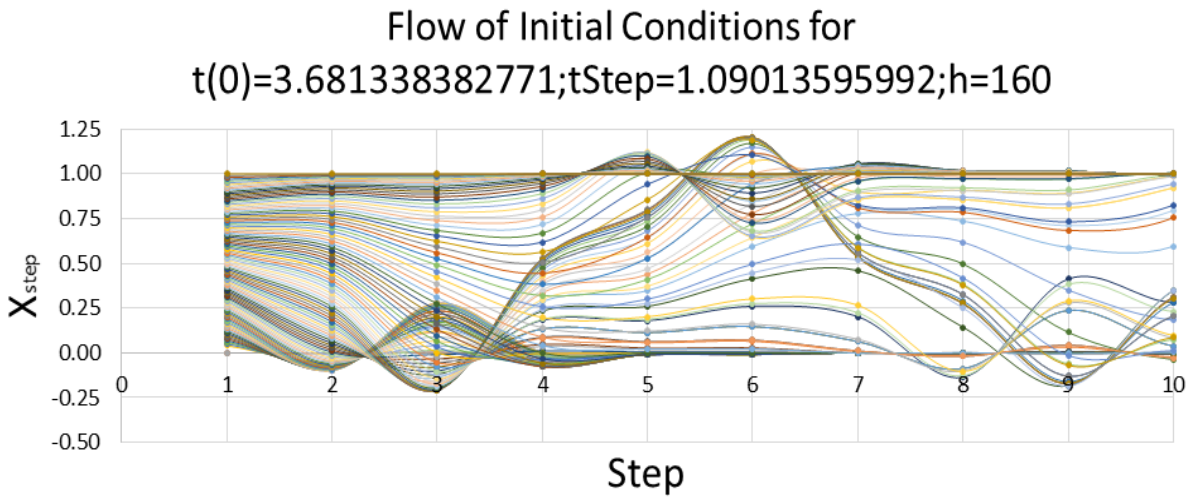


Fig. 2.4 Oscillating Dynamic Flow

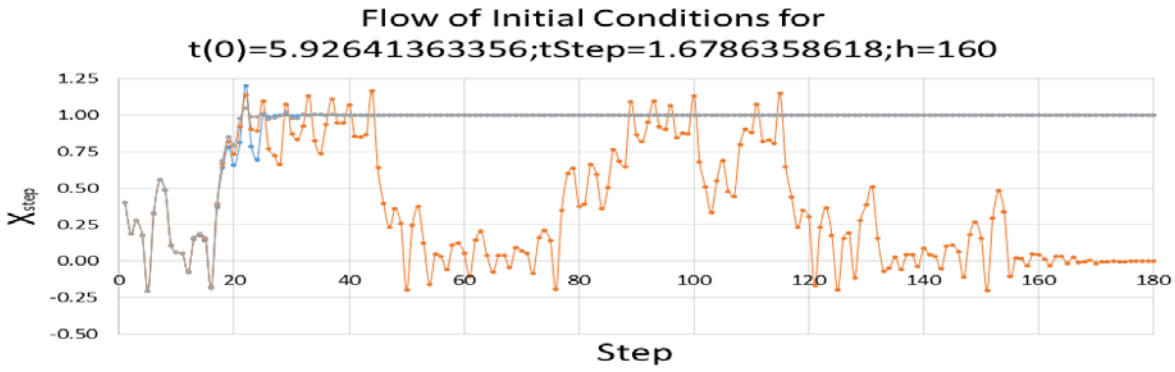


Fig. 2.5 SDIC $x^1{}_0$ =.4010 $x^2{}_0$ =.4011 and $x^3{}_0$ =.4012

	$x^1{}_0 = .42$	$x^2{}_0 = .43$	$x^3{}_0 = .44$	$x^4{}_0 = .45$
Step	x_{step}	x_{step}	x_{step}	x_{step}
984	**-0.0078819528**	**0.0095668443**	**0.9927510065**	**0.9927510065**
985	-0.0075566346	0.0097409521	1.0169397068	1.0169397068
986	0.0072489935	-0.0084494277	0.9904331557	0.9904331557
987	-0.0169397068	0.0206455587	0.9902590479	0.9902590479
988	0.0095668443	-0.0078819528	1.0084494277	1.0084494277
989	0.0097409521	-0.0075566346	0.9793544413	0.9793544413
990	-0.0084494277	0.0072489935	1.0078819528	1.0078819528
991	0.0206455587	-0.0169397068	1.0075566346	1.0075566346
992	**-0.0078819528**	**0.0095668443**	**0.9927510065**	**0.9927510065**
993	-0.0075566346	0.0097409521	1.0169397068	1.0169397068
994	0.0072489935	-0.0084494277	0.9904331557	0.9904331557
995	-0.0169397068	0.0206455587	0.9902590479	0.9902590479
996	0.0095668443	-0.0078819528	1.0084494277	1.0084494277
997	0.0097409521	-0.0075566346	0.9793544413	0.9793544413
998	-0.0084494277	0.0072489935	1.0078819528	1.0078819528
999	0.0206455587	-0.0169397068	1.0075566346	1.0075566346
1000	**-0.0078819528**	**0.0095668443**	**0.9927510065**	**0.9927510065**

Table2.1

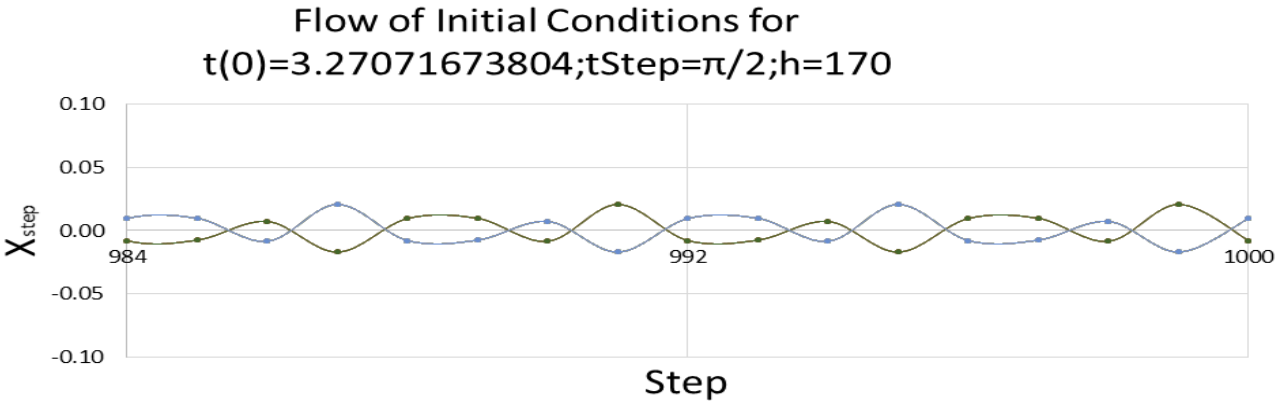


Fig. 2.6 Pictured are 53 unique initial condition trajectories period-8 oscillating around 0 (see table 2.1).

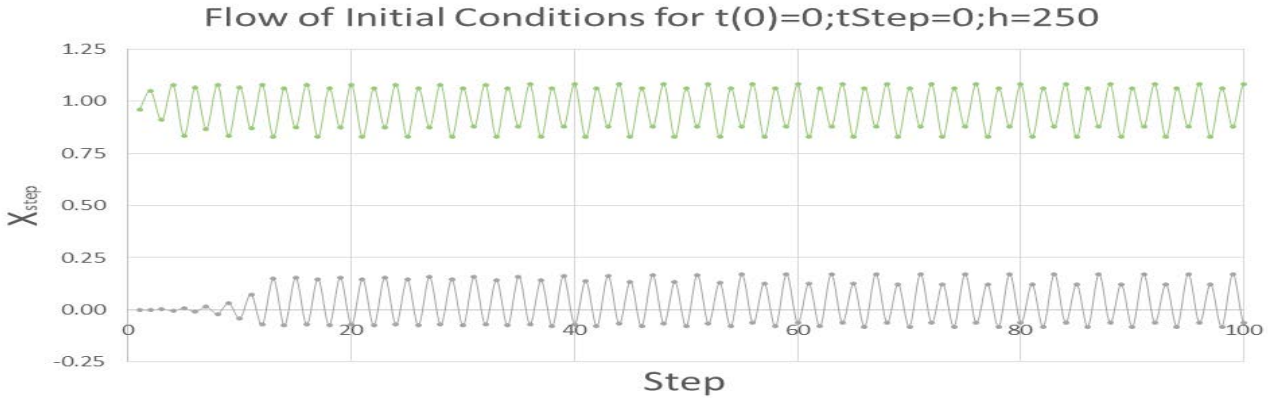


Fig. 2.7 with initial conditions $x^1{}_0 = .0012$ and $x^2{}_0 = .97$

Figure 2.7 and table 2.2 shows the rise of period-4 oscillations in the for h = 250. Note that, as shown in table 2.2, every initial condition will reach a steady state of period 4 oscillations {-.08, .17, -.06, .12} if $x_0 < .5$ or {1+.08,1-.17,1+.06,1-.12}={1.08,.83,1.06,.88} if $x_0 > .5$.

	$x^1{}_0 = .61$	$x^2{}_0 = .72$
Step	x_{step}	x_{step}
991	**0.877572856879239**	**1.061760827832230**
992	1.080402612634390	0.877572856879239
993	0.828312884504610	1.080402612634390
994	1.061760827832230	0.828312884504610
995	**0.877572856879239**	**1.061760827832230**
996	1.080402612634390	0.877572856879239
997	0.828312884504610	1.080402612634390
998	1.061760827832230	0.828312884504610
999	**0.877572856879239**	**1.061760827832230**
1000	1.080402612634390	0.877572856879239

Table 2.2

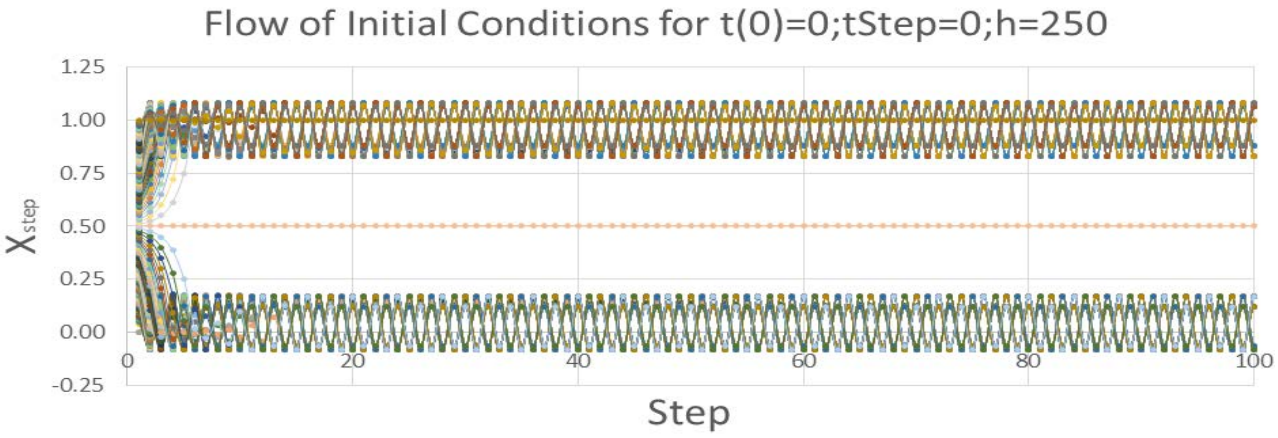


Fig. 2.8

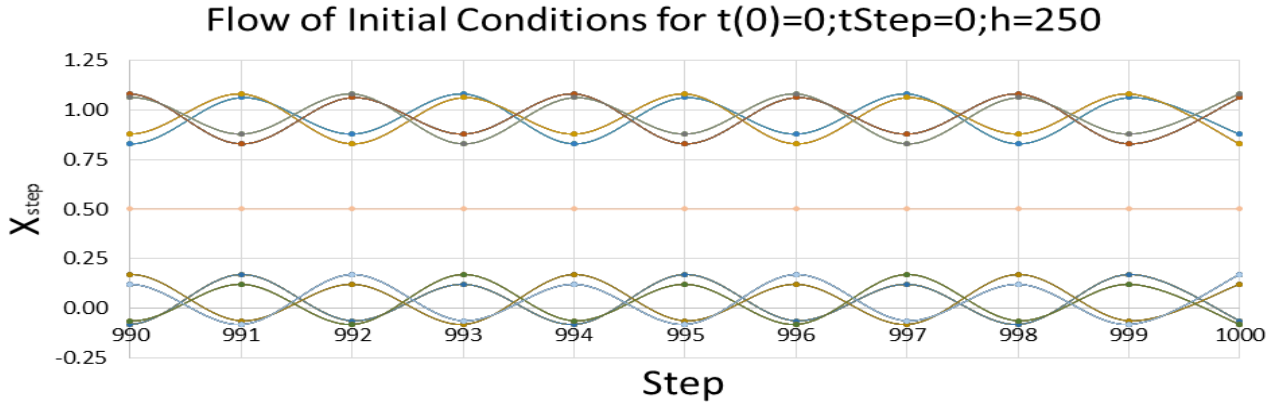


Fig. 2.9

Figures 2.8 and 2.9 show the propagation of 100 unique conditions stabilizing to harmonious period 4 orbits. However, adding a small change to tStep the system in this intermediate range of h brings rise to the question of sensitive dependence on initial conditions. Figures 2.10 and 2.11 show the system beginning to destabilize and exhibit some aperiodic behavior. Although both figures eventually settle to a stable orbit we question the conditions under which the 0 and 1 attractors will dominate the system. If tStep is increased further the system will destabilize completely and initial conditions will escape to infinity.

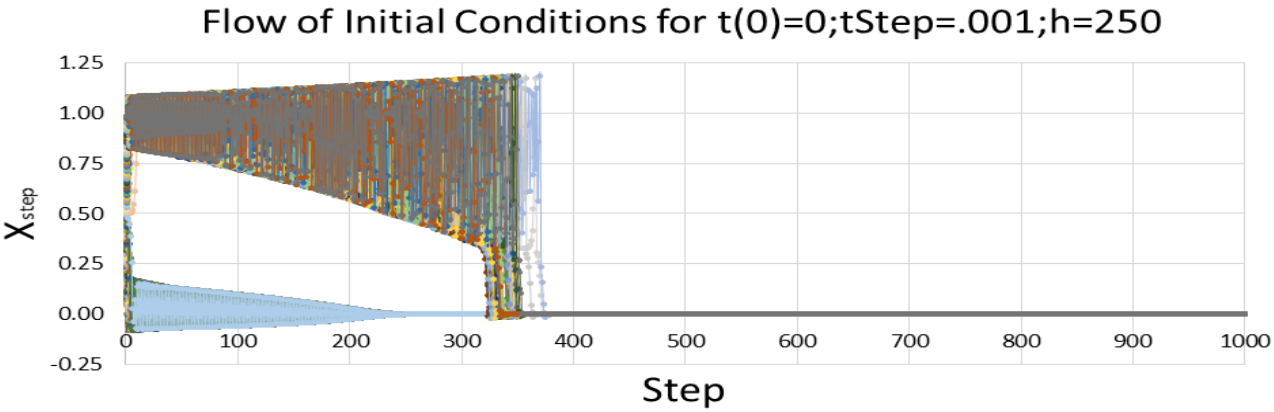


Fig. 2.10

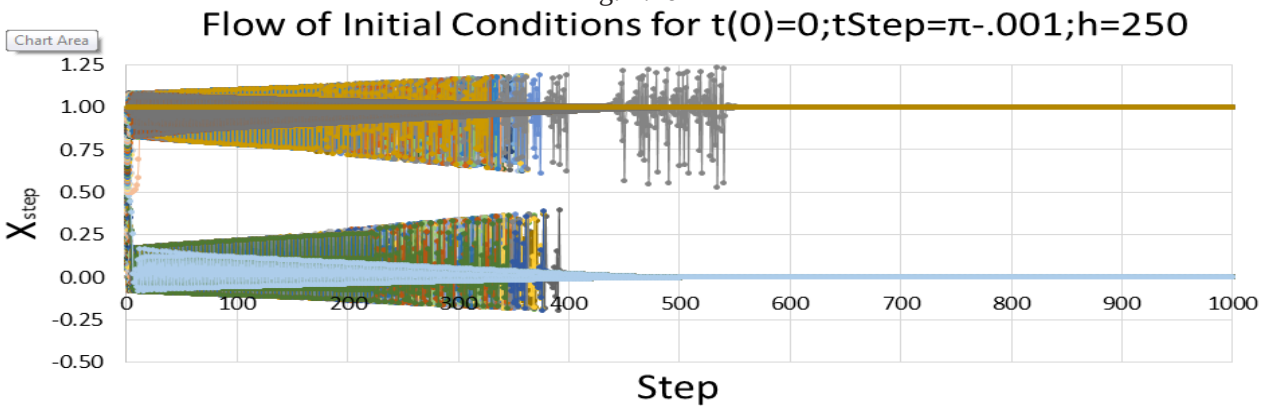


Fig. 2.11

Up to now we have seen that eventually all the trajectories will settle at or oscillate about the stable points or stable orbits at 0 or 1. However if we set h to the globally stable max around 190.5 and choose tSTep as pi/2 we will see a new behavior emerge.

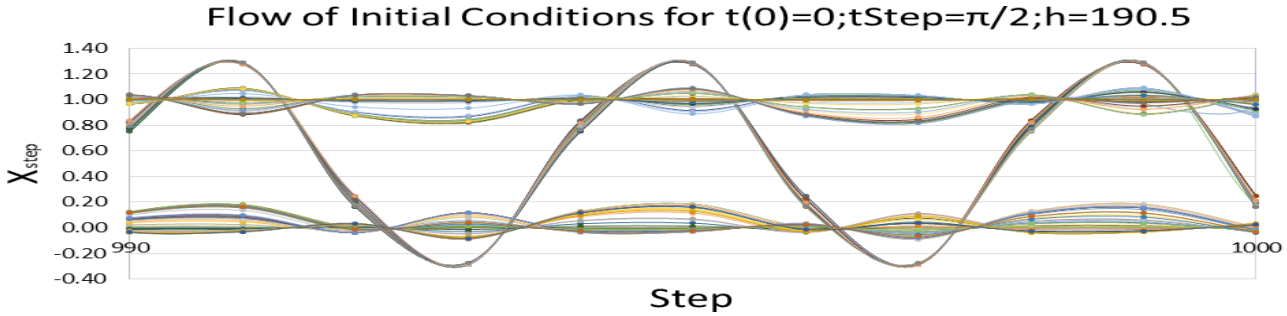


Fig. 2.12

The system is now overall still stable, but the 0 and 1 attractors no longer dominate the system, as shown in figure 2.12. The paper has concluded with simulating the initial observations and is now poised to begin answering the initial questions. Of primary concern is determining the valid ranges of tStep and h for which the system is always stable. This will naturally lead us down a road to discovery answering the remaining questions.

3 Chaotic Behavior

Continuing on our path of discovery we set h to a locally stable max of 205. Here, we notice that the orbits 0 and 1 have completely destabilized, but the system overall remains stable with trajectories that fill the space (0,1) (see step 994 of figure 3.1). Now that the conditions of chaos have been visually satisfied with sensitive dependence on initial conditions and dense covering of the domain, we begin our chaotic analysis.

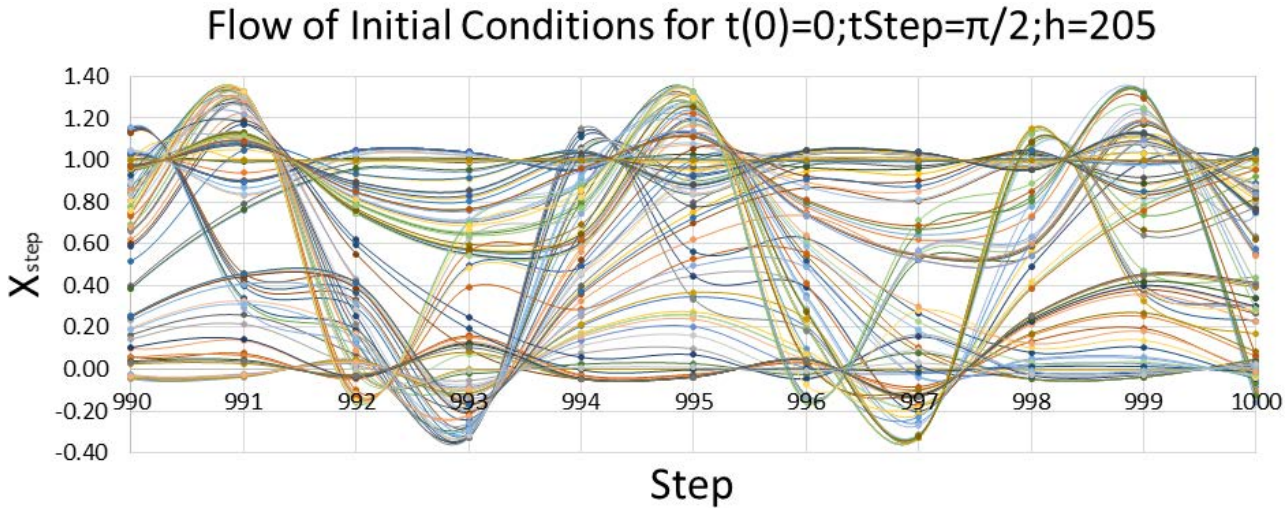


Fig. 3.1

However, if we zoom out a little and take a look at the behavior for 100 steps prior (fig 3.2), we notice that the system is apparently periodic on a global scale. Moreover, this system becomes unstable for different t(0) and therefore does not apply well to our motivation, as we must assume random initial conditions (i.e. t(0) = rand*2*pi) in order for our system to be applied to a neural network. Therefore, not only is this system not entirely chaotic, it is not practical for our application. We will need to keep our h value under the global max to ensure the system is robust under any initial conditions.

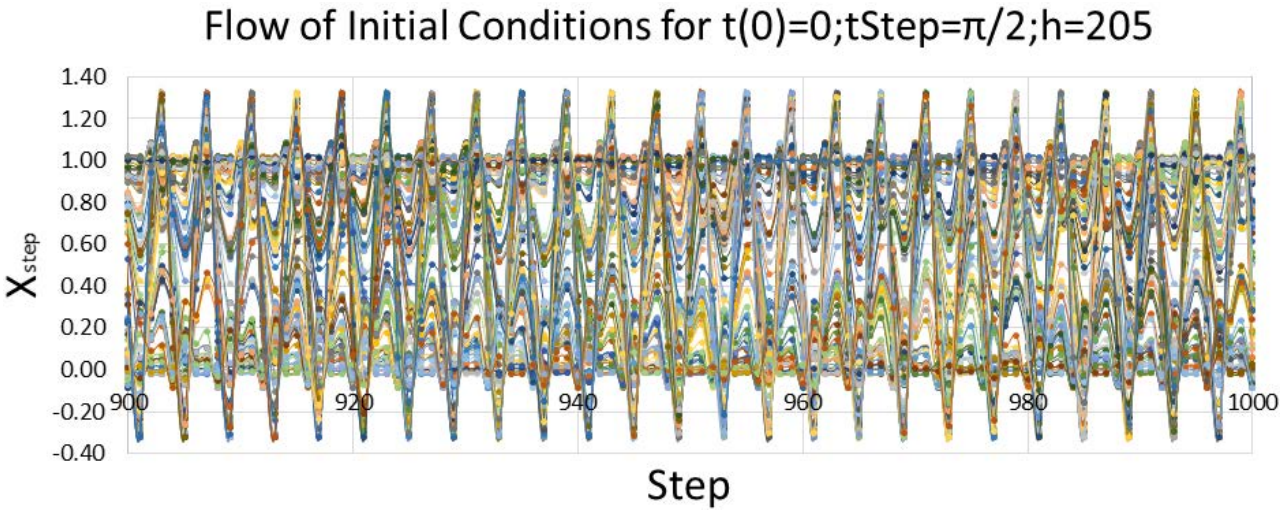


Fig. 3.2

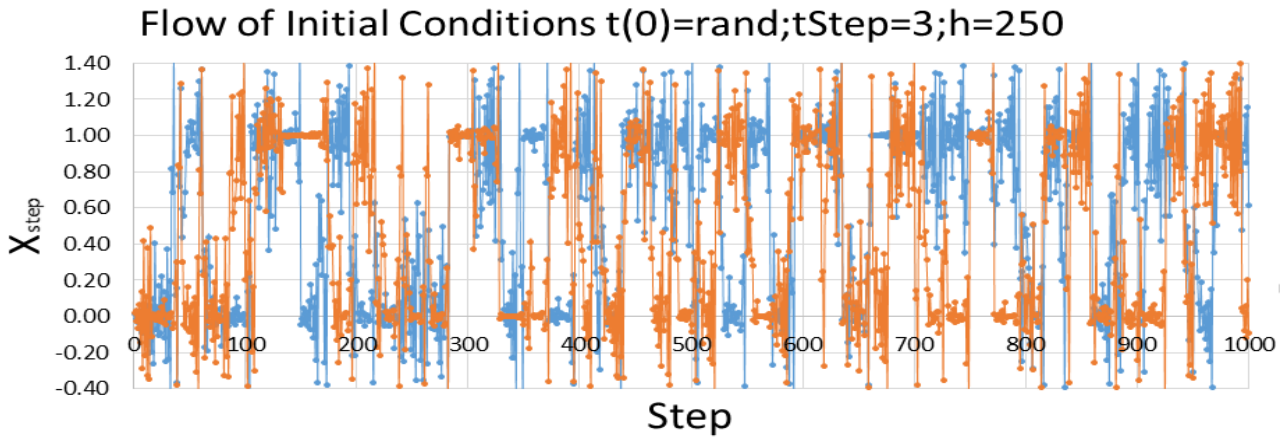


Fig. 3.3

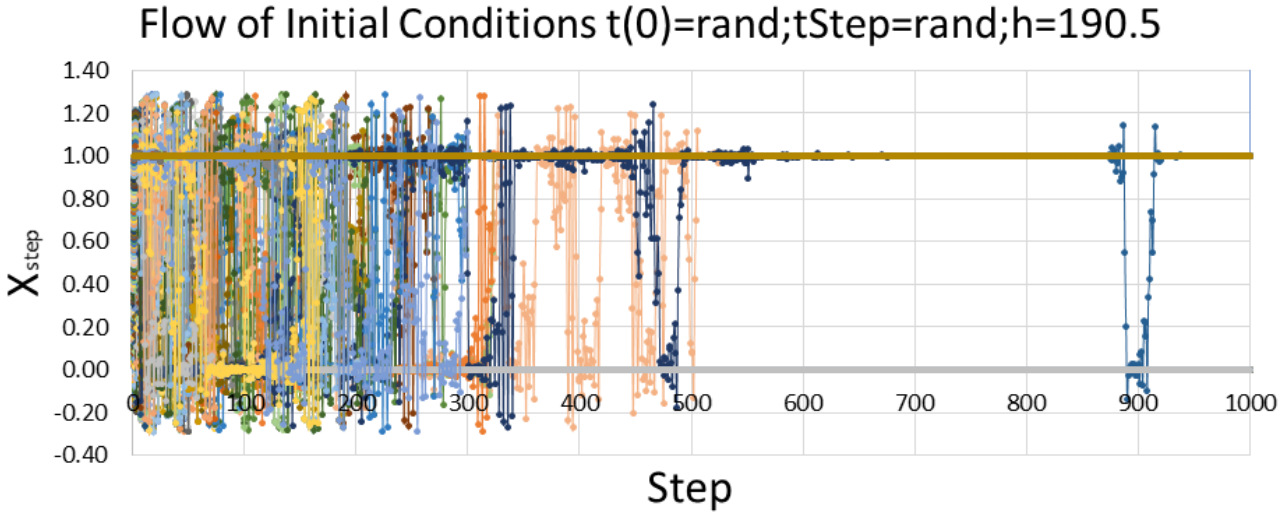


Fig. 3.4

When tStep is not a multiple of pie, the aperiodicity is removed, and complete chaos ensues as t goes to infinity (fig 3.3). However, we are limited on our selection of t(0) and tStep. When h is set at 190.5 and tStep and t(0) are allowed to take on random conditions $(0,2\pi)$ transient chaos persists in the system. Furthermore, although the perturbation must be fairly large, a slight push away from the stable orbit will cause the trajectory to begin jolt off on another transiently chaotic orbit before stabilizing again on 0 or 1. In figure 3.4 a .01 perturbation was added to a random trajectory at step 867.

The kind of behavior pictured in figure 3.4 is what we are hoping to see based on the motivation of creating this dynamical system. Theoretically, the activations, plotted in figure 3.4 on the y-axis, are chaotically flowing through the system, and settle around points which provide the smallest loss, F(x,t). However, small perturbations from back-propagation will destabilize a trajectory which has settled near at a non-optimal local minima. The paper will now transition to determining if there are some intervals of tStep that h could be increased without losing global stability for a random t(0). We know that the system is globally stable at h = 190.5; but what regions are “locally” stable for larger h?

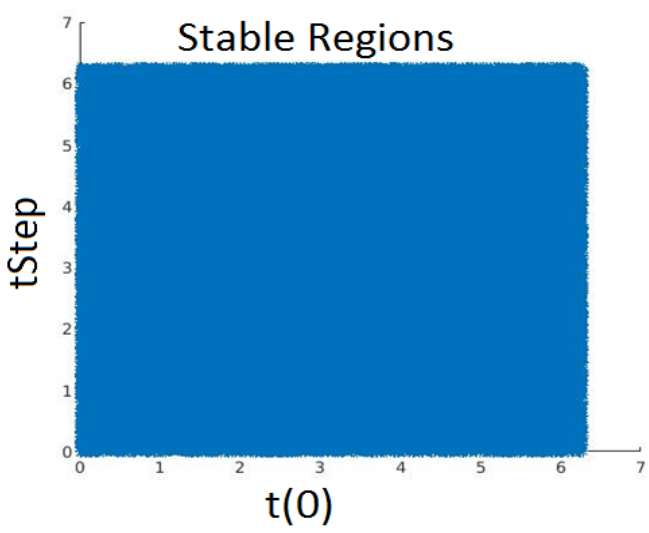


Fig 3.5 h = 190.5

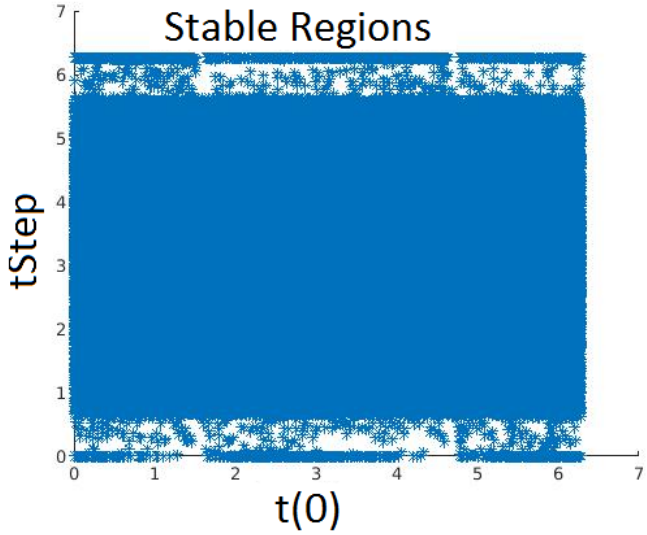


Fig. 3.6 h = 191.0

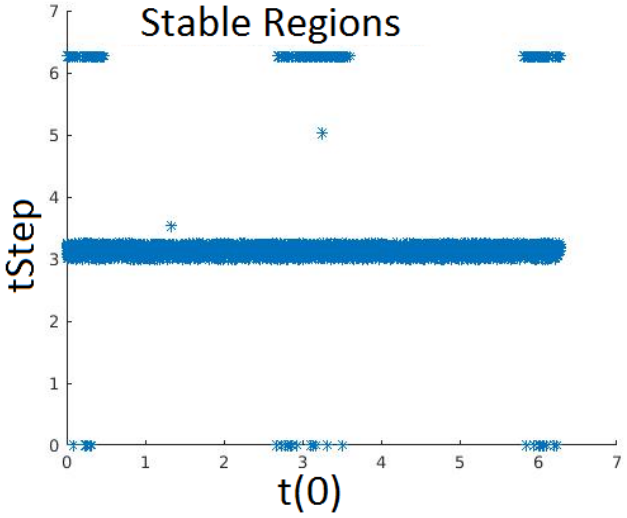


Fig. 3.7 h = 265

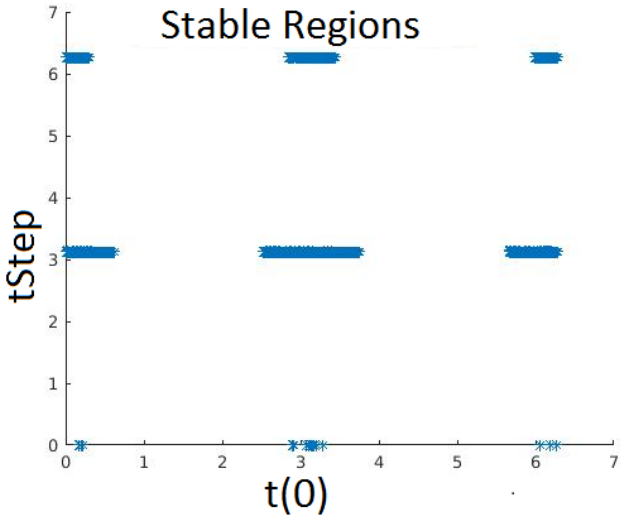


Fig. 3.8 h = 300

We see that the stable regions shrink to a volume of zero as h grows and is no longer visible at this resolution. We notice that the system is always stable within some interval of tStep = 0 and pi until h is greater than approximately 265. For larger h, the system is only stable for particular t(0). Below we plot the valid x values vs their respective loss, F(x,t), to examine the behavior further. Since we noticed something special about tStep = pi we set it constant and plot the valid responses. We assume that there should be a positive δ and ϵ, where δ =: | N*pi - δ | (N a positive integer), that for any h if tStep < δ the system will remain stable for all $x_0 <$ ϵ at any t(0). The derivative of F(x,t) approaches 0 as ϵ approaches zero. Therefore, we have a loose proof that although the volume of the region approaches to zero, there is some structure that remains.

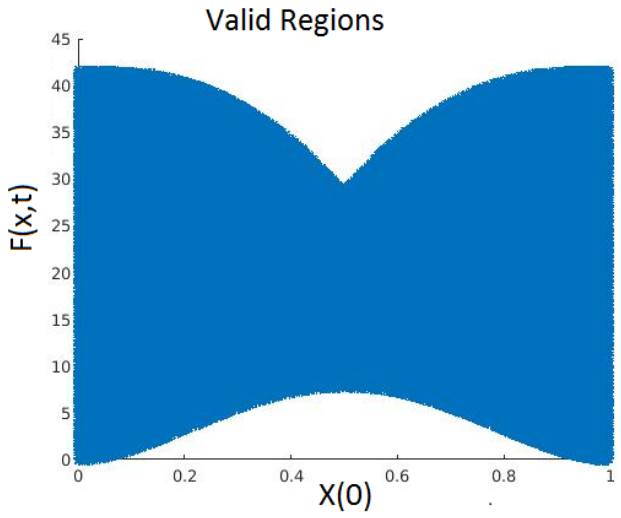


Fig. 3.9 h = 190.5

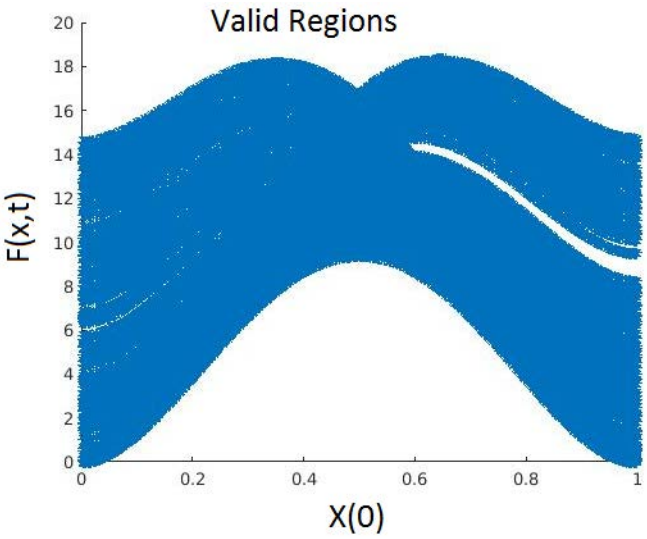


Fig. 3.10 h = 191.0

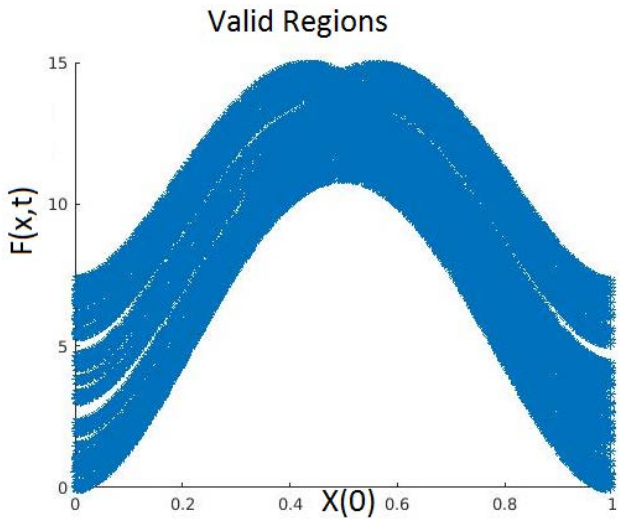


Fig. 3.11 h = 265

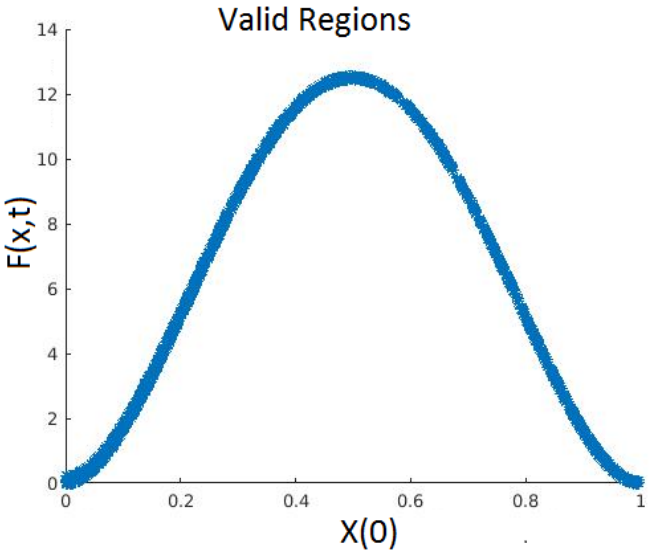


Fig. 3.12 h = 300

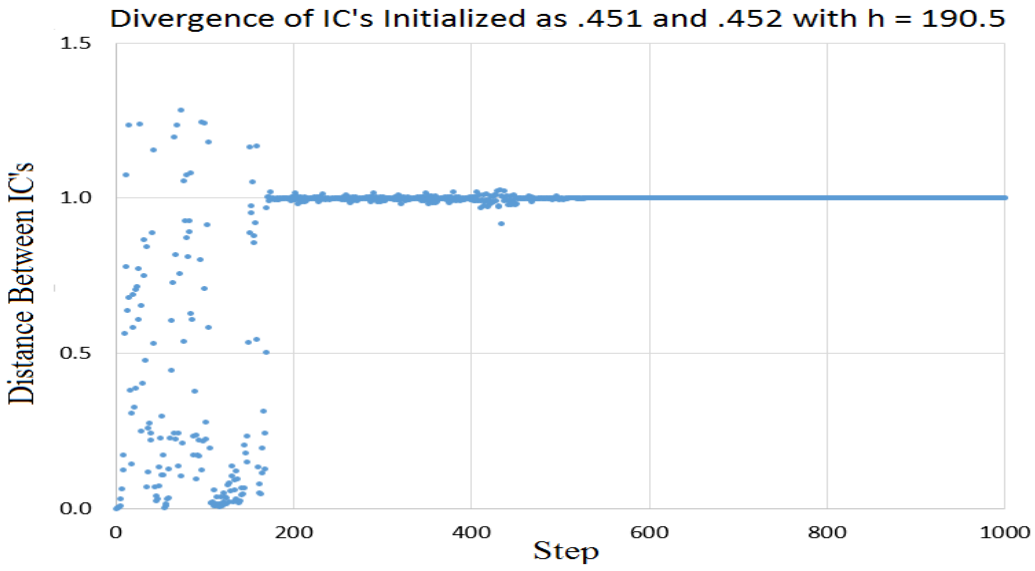


Fig 3.13 Transient chaos

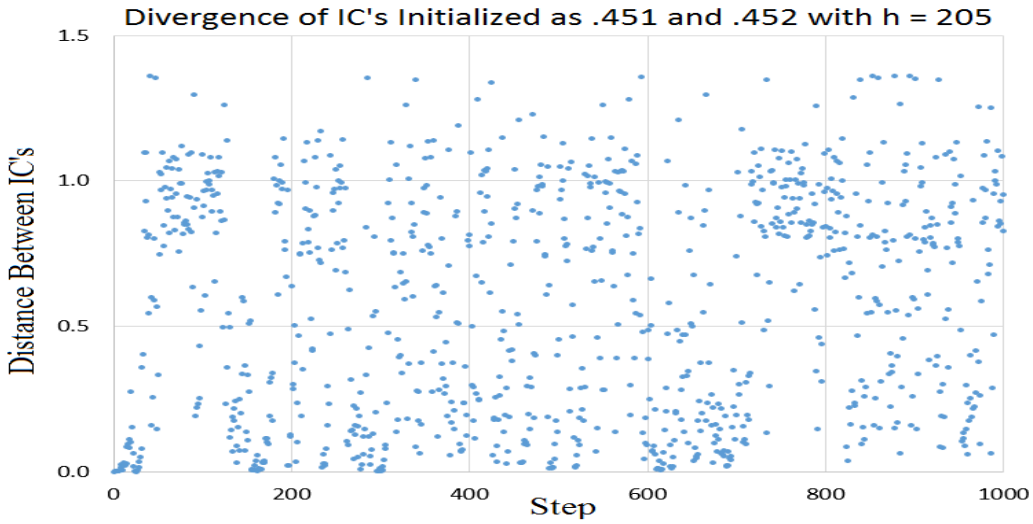


Fig. 3.14 Persistent chaos

Figure 3.13 visualizes the divergence and sensitive dependence of initial conditions when h is low enough that the orbits 0 and 1 are attractors. Figure 3.14 is a visual of when h is locally stable and chaos persists, but the system itself is a chaotic attractor.

4 Experiments

4.1 Stochastic Rounding

4.1 We began exploring how to experiment with our results by looking at BinaryConnect(BinaryConnect) and BinaryNet(BinaryNet). These papers do not attempt to modify the loss function, and instead use a stochastic binarization technique to train binary weight. Namely, the function BinaryConnect and BinaryNet use is a hard sigmoid, and name the process Stochastic Rounding, s.t.:

$$x^1 \quad = Sign(x \quad) = \begin{Bmatrix} +1 \; if \; x \geq 0 \\ -1 \; otherwise \end{Bmatrix}$$

$$x^1 = \begin{Bmatrix} +1 \; with \; probability \; p = \; \sigma(x) \\ -1 \; with \; probability \; 1 - p \end{Bmatrix}$$

$$\sigma(x) = (clip\left(\frac{x+1}{2}\right),0,1)$$

(BinaryNet)

From equation 1.2., treating t as a constant and solving for zero, we looked at the relative space that the equation had a positive, and negative value . We form the conjecture that the probability of rounding can be found as follows:

Solving for t in equation 1.2 we obtain:

$$t = sin^{-1}(1 - 2x), \text{ and } t^* = \pi - t \;\; \text{s.t. } t \;\&\; t^* \in [0,2\pi)$$

Then our conjecture is that the probability to switch rounding is:

$$\text{P(switch)} = \frac{t^* - t}{2\pi} = \frac{1}{2} - \frac{t}{\pi} \qquad \text{(Conjecture 4.1)}$$

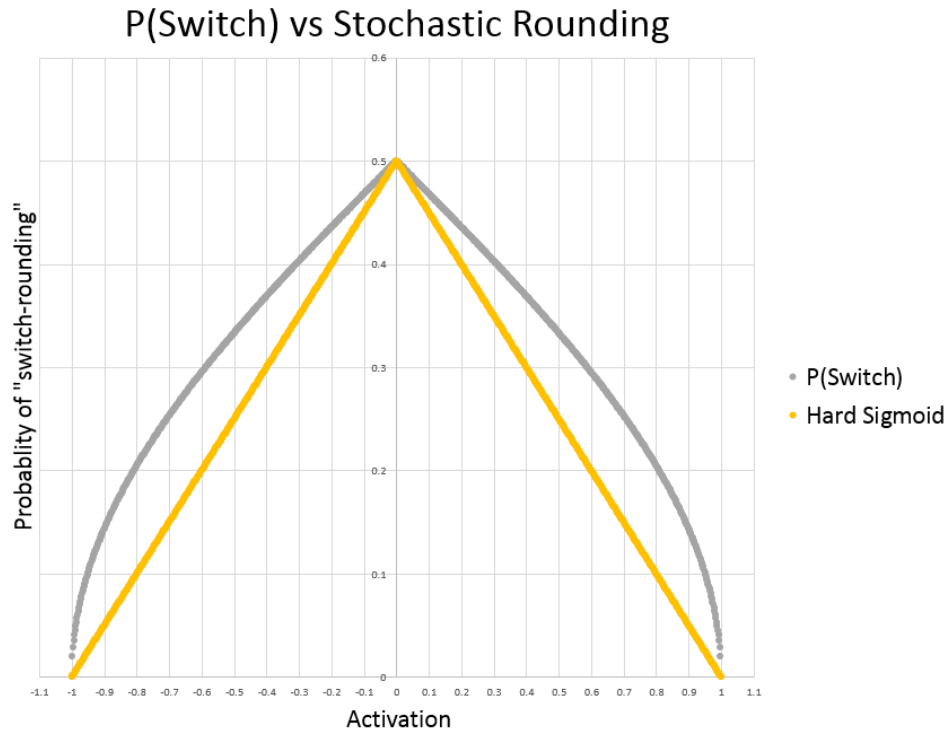


Fig. 4.1: Probability of switch rounding scaled [-1,1]

We implemented this new probability to BinaryNet and BinaryConnect with the following results:

Code Base:	BinaryNet				Binary Connect			
Version:	Original		P(Switch)		Original		P(Switch)	
Data Set:	CIFAR	MNIST	CIFAR	MNIST	CIFAR	MNIST	CIFAR	MNIST
Test Error:	11.39	.99	12.61	1.40	7.92	1.22	9.38	1.15

Table 4.1

While we only out-performed one of the four test sets, we believe there is room to explore optimal probability of switching in future work. The results also gave us promise that our research was progressing well and that adding equation 1.3 to the loss function of a neural network would produce positive results.

4.2 Dynamic Loss

The motivation for developing this equation was to create a binary layer which could be fed into a neuromorphic processor. Therefore, in our experiments we focus on changing a single layer of weights to have a binary representation and examine the results. We added equation 1.3 to the loss function in the TensorFlow "SoftMax" and "Deep" MNIST tutorials for baseline since there are limited works available that reference training through loss-function (TensorFlow). For the "SoftMax" model, we let the x from equation 1.3 be defined as the weights, W, of the network (there is only one layer) and rounded the weights off at the end of training. Less than 10 percent of the available weights took on values of 1, with the remainder taking 0. For the "Deep" model, we let the x from equation 1.3 be defined as the second convolutional layer weights, W_conv2, and rounded the weights off at the end of training. For both models we initialized t at one and increased t by one for every training step, and we initialized H at a locally stable value of 300.

We achieved positive results, losing less than one percent of test accuracy and obtaining a binary representation of the data at a layer. We believe this satisfies our initial research goal of developing function to transduce data for a neuromorphic processor. Our results follow:

Test Accuracy of Model	SoftMax	Deep
Original	91.60	99.16
Loss + Equation 1.3	90.70	99.01

Table 4.2

5 Conclusion

(1) Is this system always predictable; does there exist sensitive dependence on the initial conditions (SDIC)?

We conclude that if 160 > h that this system is always predictable. However, for h larger than 160 the system displays SDIC. If 190 > h > 160 the system is unpredictable, but is only transiently chaotic; trajectories have a steady end-state either orbiting or equal to 0 or 1.

(2) How large is “too” large for parameter h?

We conclude that h > 400 makes the system unstable for nearly all initial conditions.

(3) Are there parameters for which there exists chaos?

If 400 > h > 191 there are regions of initial conditions for which there is persistent chaos, but the system remains stable in the region for all initial conditions.

(4) Can this function be added to the loss of a neural network to train transduction layers?

The function was added to the loss function with positive results. Although there are limited works available to compare our experiments against, we have shown that we can use a dynamic equation to entice a loss function to achieve a binary network.

Bibliography

BinaryNet, arXiv 1602.02830. https://arxiv.org/abs/1602.02830.

BinaryConnect, arXiv 1511.00363. https://arxiv.org/abs/1511.00363.

Strogatz, Steven H., Nonlinear Dynamics and Chaos, page 373. Westview Press, 2015.

Pitt, arXiv 1604.00697. https://arxiv.org/abs/1604.00697.

TensorFlow, GitHub. https://github.com/tensorflow/tensorflow/tree/r1.3/tensorflow/examples/tutorials/mnist.