Exploiting Chaos for Computation

A new direction in harnessing chaos

Sudeshna Sinha

The Institute of Mathematical Sciences, Chennai

homepage: http://www.imsc.res.in/∼sudeshna

In the past two decades a “new science”, known popularly as chaos, has given us deep insights into previously intractable, inherently nonlinear, natural phenomena

Chaos has caused a fundamental reassessment of the way in which we view the physical world

For instance, certain seemingly simple natural nonlinear processes, for which the laws of motion are known and completely deterministic, can exhibit enormously complex behavior

Often appearing as if they were evolving under random forces rather than deterministic laws

One consequence is the remarkable result that these processes, although completely deterministic, are essentially unpredictable

Blending of three distinct approaches:

- Powerful analytical mathematical methods to treat functional recursion relations, to solve certain nonlinear partial differential equations, or to describe complex geometrical structures
- Experimental Mathematics: numerical simulations to give qualitative insights into problems that are at present analytically intractable
- High precision experimental observations of nonlinear phenomena in many different natural and man-made systems arising in a variety of conventional disciplines

Illustration of this tripartite approach : discovery of the universality in unimodular one-dimensional maps

Nonlinear science is inherently interdisciplinary

Impacting upon traditional subjects ranging through all the physical and biological sciences, mathematics and engineering

Initially the focus was on Suppressing Chaos

With greater understanding came the ability to manipulate nonlinear phenomena, and so the focus has shifted in recent times to Exploiting Chaos

- Chaos provides a rich variety of behaviors :

  Can concievably serve as a versatile pattern generator

- Ability to readily switch behaviours may provide flexibility

Need a control mechanism that enables us to exploit the richness of chaos in a direct and efficient manner

Thresholding (Clipping, Dynamic Range Limiter) as a strategy for extracting a wide range of stable patterns from a chaotic system

Sinha & Biswas, Physical Review Letters, 1993;
Sinha, Physcial Review E, 1994;
Murali & Sinha, Physical Review E, 2003

Consider a general dynamical system $\dot{\underline{x}} = F(\underline{x})$

Choose a state variable to be monitored

Threshold Mechanism is triggered whenever the value of the variable exceeds a critical threshold x^*

The variable is then re-set to x^*

If $x > x^*$ then $x = x^*$

The dynamics continues till the next occurrence of the variable exceeding the threshold

Principle : Restricts available phase space

Dynamic Range Limiter

- Prunes chaotic temporal sequences to stable desired patterns
- Chaos advantageous as it possesses a rich range of temporal patterns which can be clipped to different behaviours

For example for the chaotic logistic map $f(x) = 4x(1-x)$

Different regular dynamical patterns obtained for different thresholds x^* on the variable x

- $x^* < 0.5$: Fixed point
- $0.5 < x^* < 0.809$: Period 2
- $0.809 < x^* < 0.85$: Period 4
- $x^* = 0.86$: Period 6
- $x^* = 0.88$: Period 7
- $x^* = 0.9$: Period 9

- Exact calculation of the period corresponding to a certain threshold
- Answer the reverse (important) question as well: what threshold do we need to set in order to obtain a certain period

Sinha, 1999

Starting point of the analysis : the chaotic system, being ergodic, is guaranteed to exceed threshold at some point in time, at which point its state is re-set to x^*

One then studies the forward iterations of the map, starting from this state $x = x^*$, i.e.

$$f_0(x^*), f_1(x^*) \ldots$$

where $f_k(x^*)$ is the k^{th} iterate of the map

Specifically for the logistic map $f(x) = 4x(1-x)$:

- $k = 0$; $f_0(x^*) = x^*$
- $k = 1$; $f_1(x^*) = 4x^*(1-x^*)$
- $k = 2$; $f_2(x^*) = 4(4x^*(1-x^*))(1-4x^*(1-x^*))$

In general

$$f_k(x^*) = f \circ f_{k-1}(x^*) = f \circ f \circ \ldots f \circ (x^*)$$

where threshold value $0 < x^* < 1$

When $f_k(x^*) > x^*$ we have a k cycle : as this implies that the k^{th} iterate exceeds the threshold x^* and is re-set to x^*

$x^* \;=\; f_0(x^*)$ is the first point in the cycle

k - Cycle : $x^*,\ f_1(x^*),\ f_2(x^*),\ \ldots\ f_{k-1}(x^*)$

For instance, for the logistic map, in the range $0 \leq x^* \leq \frac{3}{4}$, $f_1(x^*) > x^*$

So the chaotic system is clipped back to x^* at every iterate, yielding a period 1 fixed point

In the range $\frac{3}{4} < x^* < 0.9$, $f_1(x^*) < x^*$ but $f_2(x^*) > x^*$

This imples that the second iterate of the map (starting from $x = x^*$) exceeds threshold and is adapted back to x^*, thus giving rise to a period 2 cycle

- Thus the cycle at each value of threshold is the smallest k such that the k^{th} iterate of the map (starting from $x_0 = x^*$) is greater than x^*, i.e.

$$f_k(x^*) > x^*$$

- The chaotic element can then yield a wide variety of regular cyclic behaviour :

  Period of the cycle depends on threshold level

- Periodicity enforced on the sequences as thresholding acts as a re-setting of initial conditions

- In threshold space we can find windows of various periods

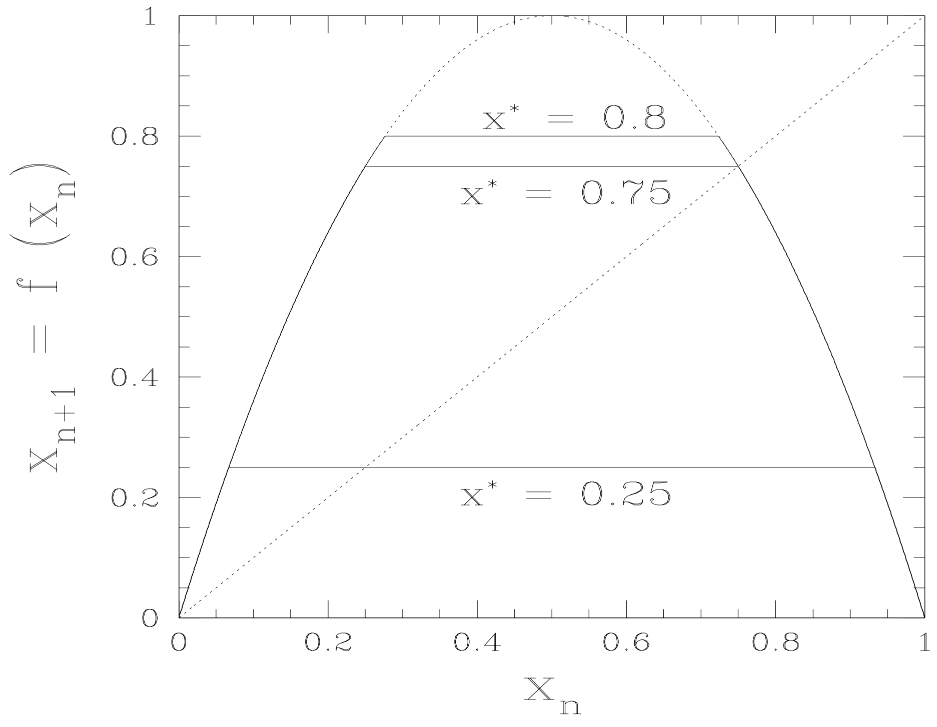
  These are intervals where the following is satisfied:

  Period $P(x^*) = k$ iff $f_k(x^*) \geq x^*$ and $f_l(x^*) < x^*$ for all $l < k$

- $P(x^*)$ is a piecewise continuous function of x^*

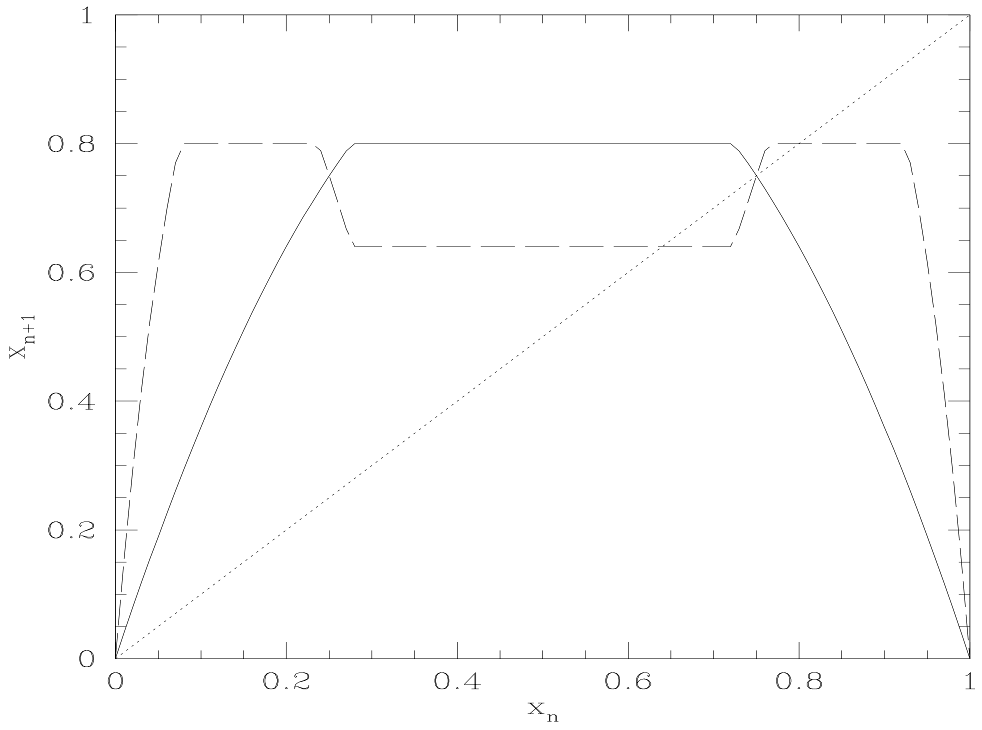
- Can formulate the different solutions using the inverse map: symbols L and R

First iterate $x_{n+1} = f(x_n)$ for the chaotic map under thresholding (the "be-headed map")



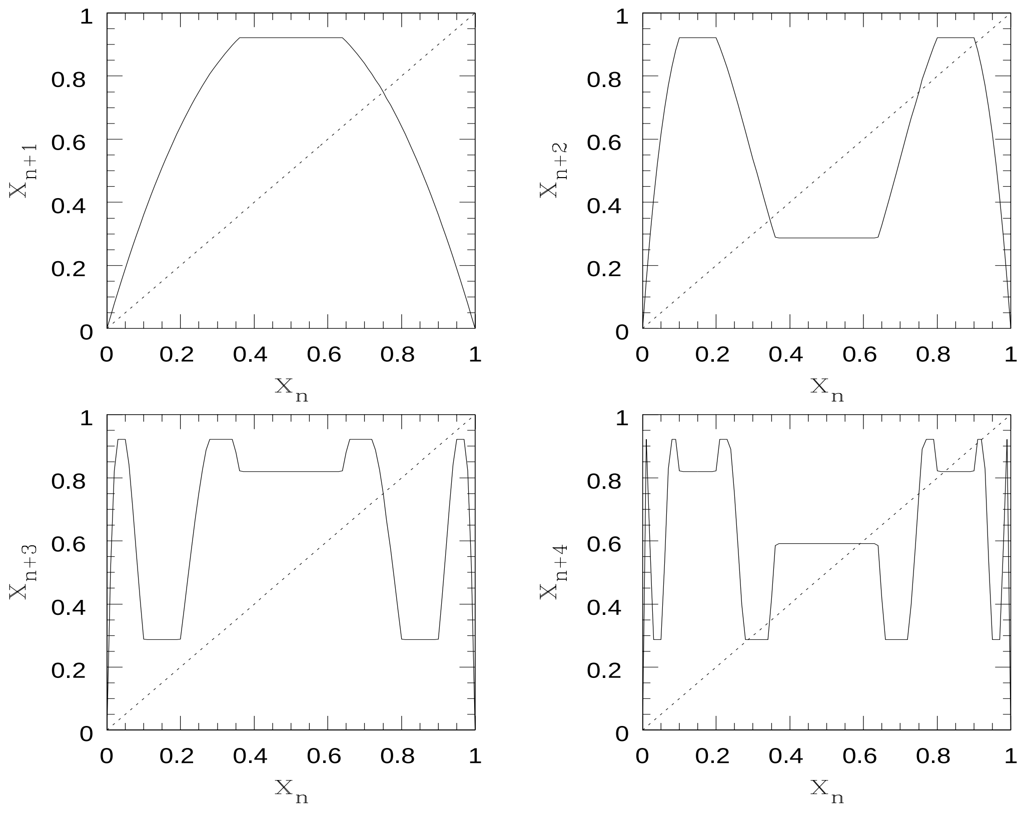
Intersection of the plateau with the 45^0 line (i.e. the $x_{n+1} = x_n$ line) yields a superstable fixed point of period 1

Iterates x_{n+1} (—) and x_{n+2} (- - -) of chaotic map under thresholding : $x^* = 0.8$



Intersection of the flat portion of the map x_{n+2} with the 45^0 line yields a superstable fixed point of period 2

Threshold value : 0.922



Intersection of the flat portion of the map x_{n+4} with the 45^0 line yields a superstable fixed point of period 4

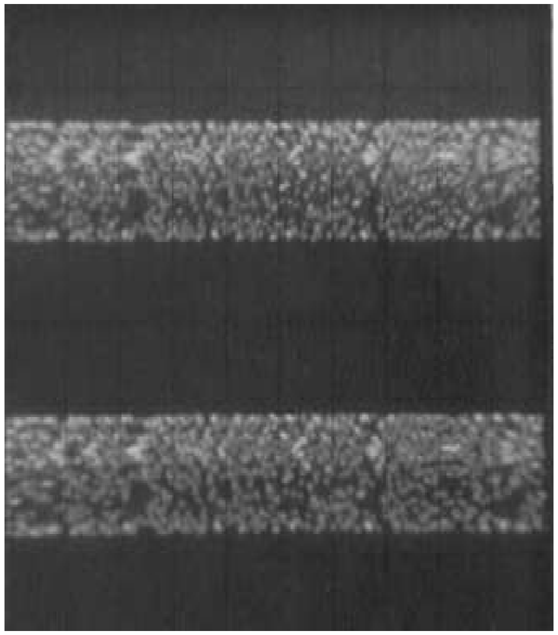
For chaotic maps it can then be analytically shown :

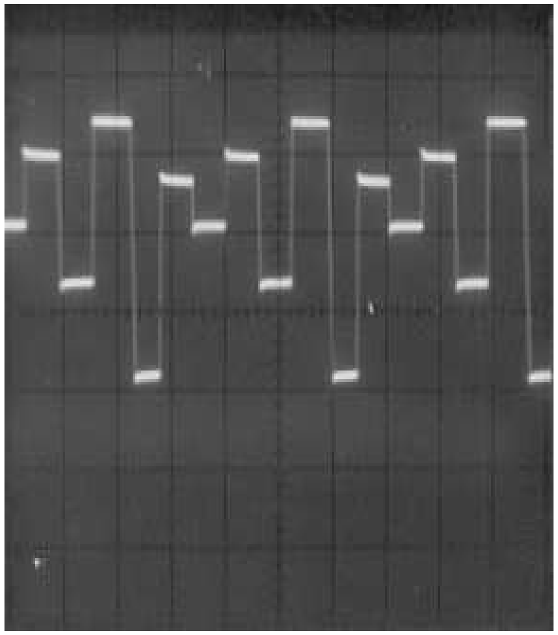
- Thresholding clips chaos to desired time sequences : yields periods of all orders
- The system is trapped in a super-stable cycle the instant it exceeds threshold

Sinha & Biswas, Physical Review Letters, 1993

Sinha, Physical Review E, 1993;   Physics Letters A, 1994;
Also reviewed in   Int. J. of Modern Physics, 1995

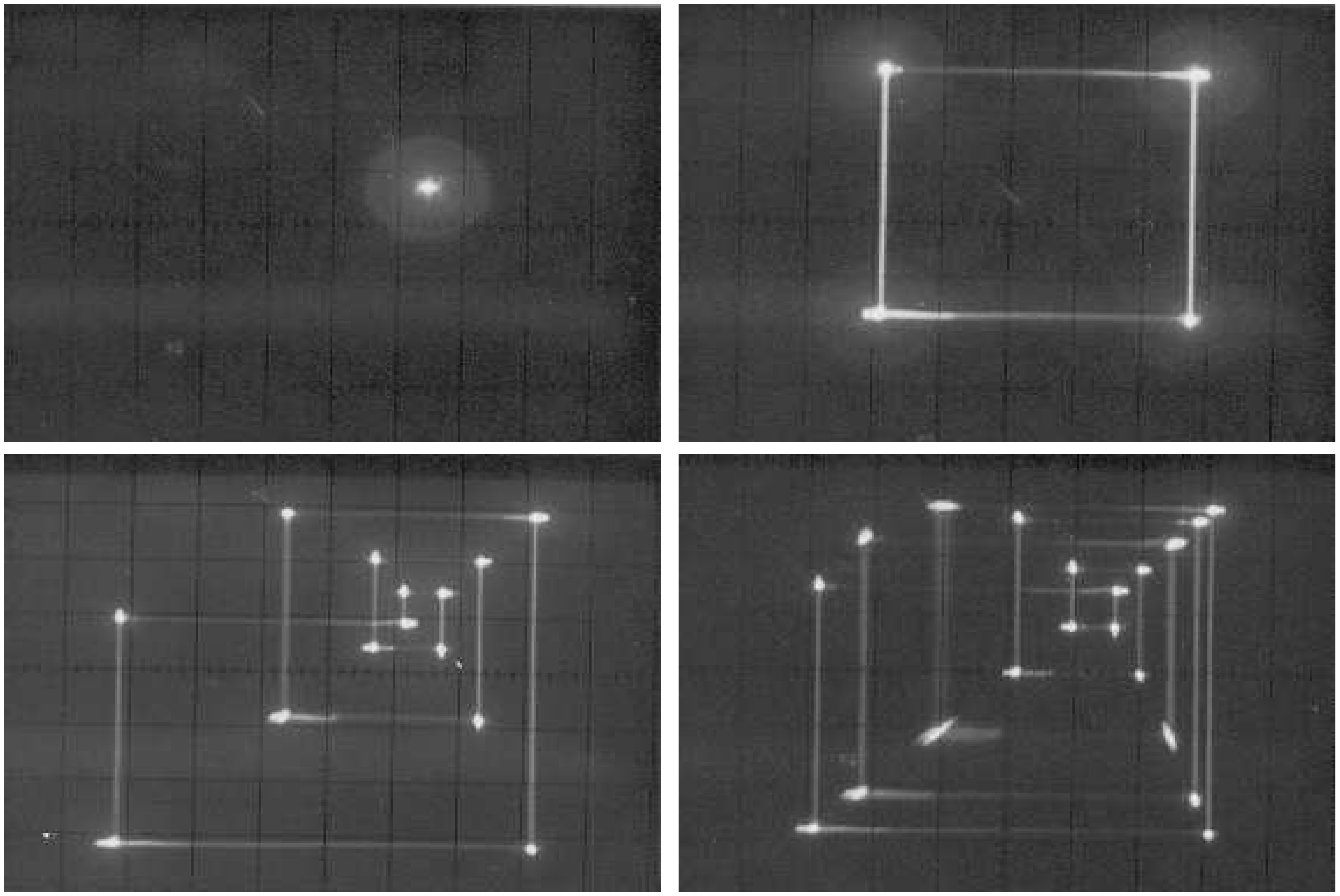
Experimental verification of clipping chaos to periods of wide ranging orders


Chaotic Trace


6 - Cycle

Circuit Realization of the Logistic Map

Murali, Sinha and Ditto, Physical Review E, 2003



Complete agreement with theoretical analysis

- Exact relations for the position and width of the periodic windows in threshold parameter space :

  Provides a look-up table to directly extract widely varying temporal patterns

- Yields a wide range of response patterns from the same module

  Thus useful for designing components that can switch flexibly between different behaviours

- Requires no run-time computations
- Transience is extremely short; Very robust
- Controller simple

Does thresholding work beyond iterative 1d maps?

Can continuous time higher dimensional (possibly hyper-chaotic) systems be clipped?

No exact results : must rely on numerics and experimentation

Nonlinear third order ordinary differential equations

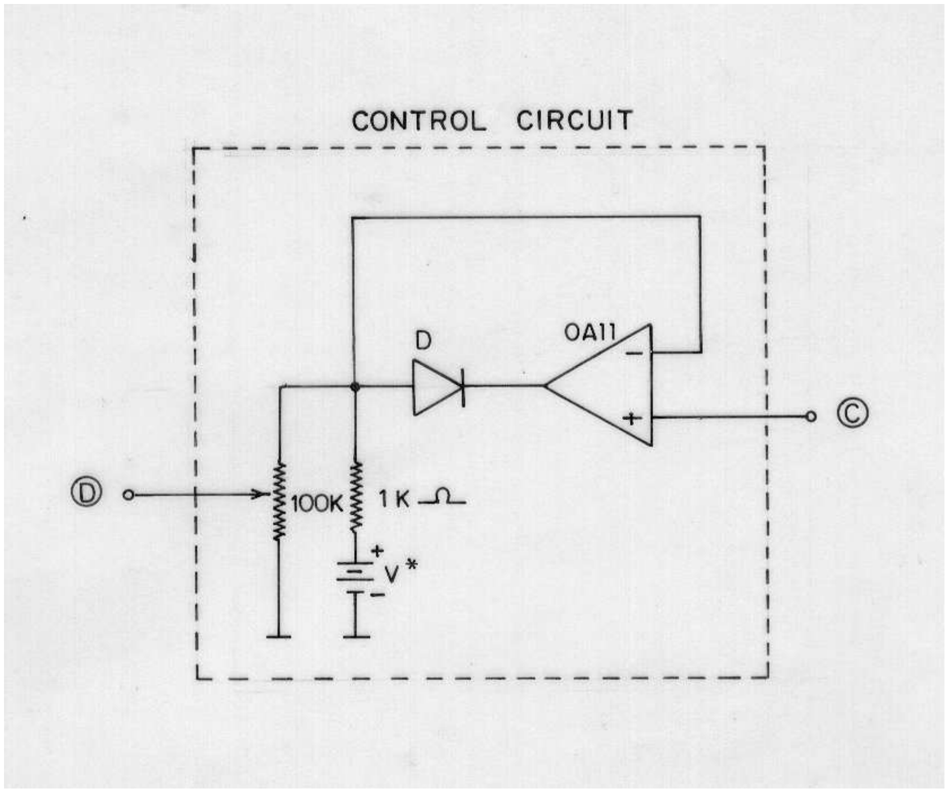
$$\frac{d^3x}{dt^3} + A\,\frac{d^2x}{dt^2} + \frac{dx}{dt} = G(x)$$

where $G(x)$ is a piecewise linear function:

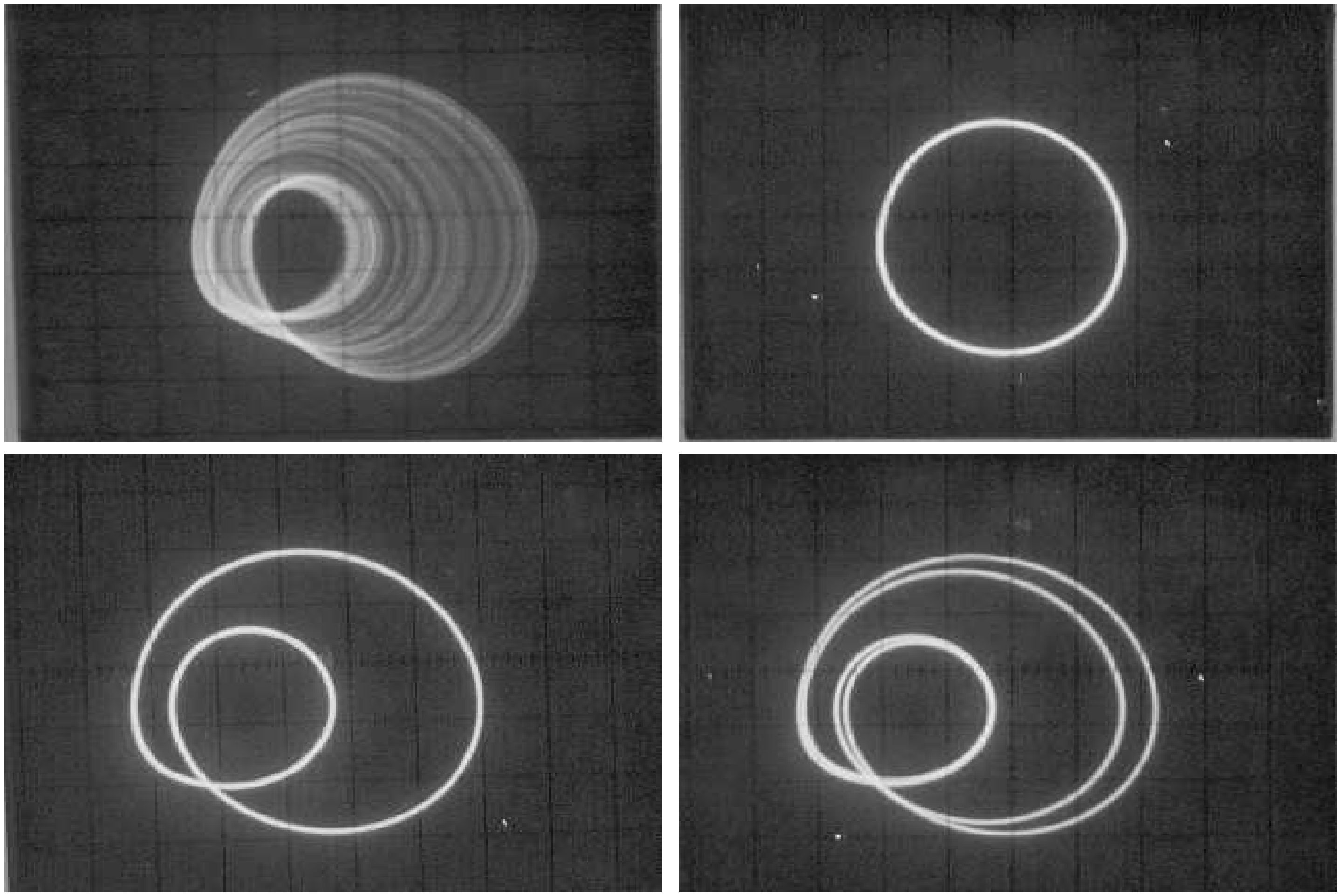
$$G(x) = B|x| - C$$

with $B = 1.0, C = 2.0$ and $A = 0.6$

Precision Clipping Circuit for Thresholding



Circuit realization of coupled third order nonlinear differential equations



Double scroll chaotic Chua's attractor given by the following set of (rescaled) 3 coupled ODEs

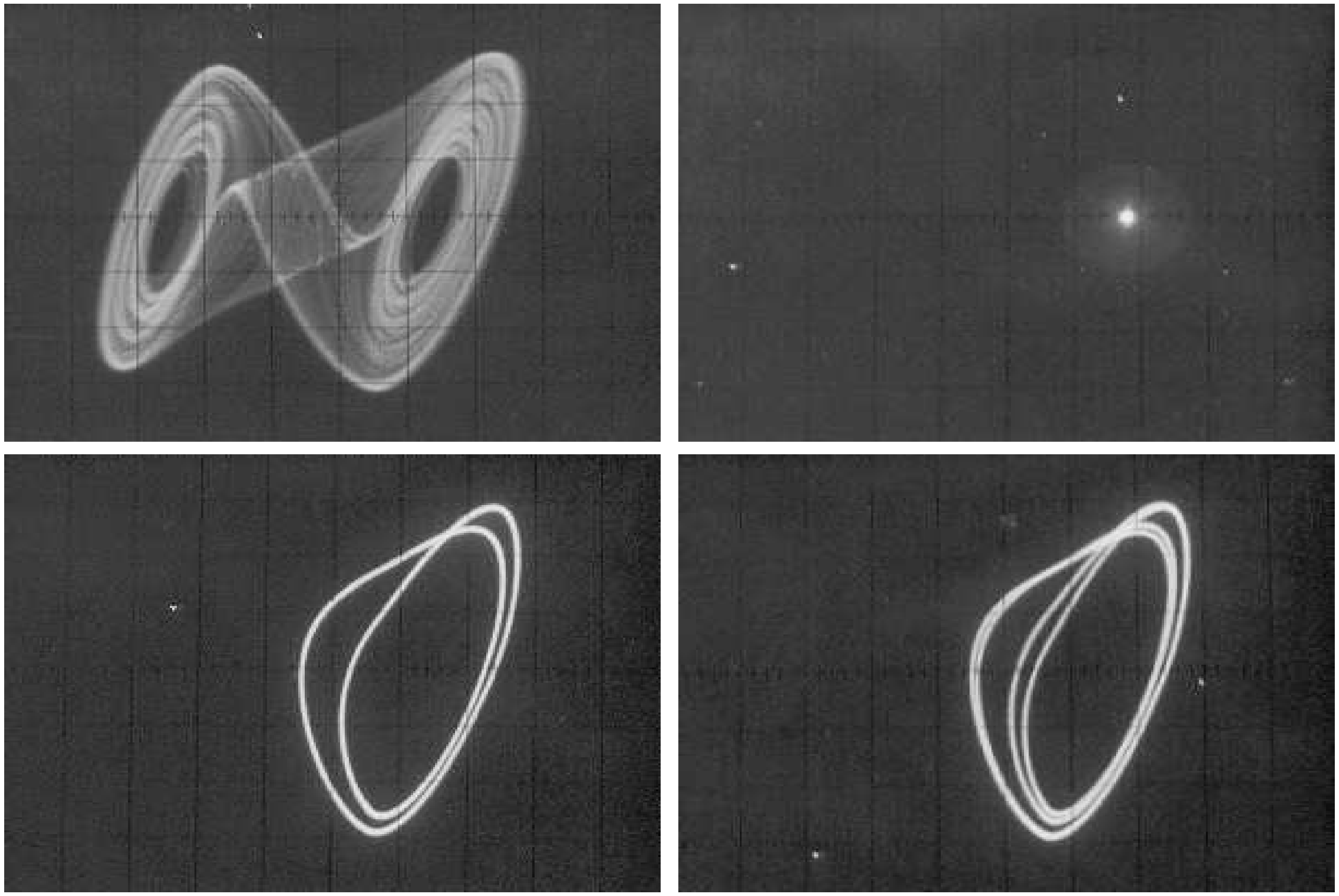
$$\begin{aligned} \dot{x} &= \alpha(y - x - g(x)) \quad (1)\\ \dot{y} &= x - y + z \quad (2)\\ \dot{z} &= -\beta y \quad (3) \end{aligned}$$

The piecewise linear function

$$g(x) = bx + \tfrac{1}{2}(a-b)(|x+1| - |x-1|)$$

Parameters: $\alpha = 10.$, $\beta = 14.87$, $a = -1.27$ and $b = -0.68$

Thresholding Chua’s Circuit



Murali and Sinha, Physical Review E, 2003

Hyperchaotic electrical circuit

Constitutes a stringent test of the control method since the system posseses more than one positive lyapunov exponent, and so more than one unstable eigendirection has to be reigned in by thresholding a single variable

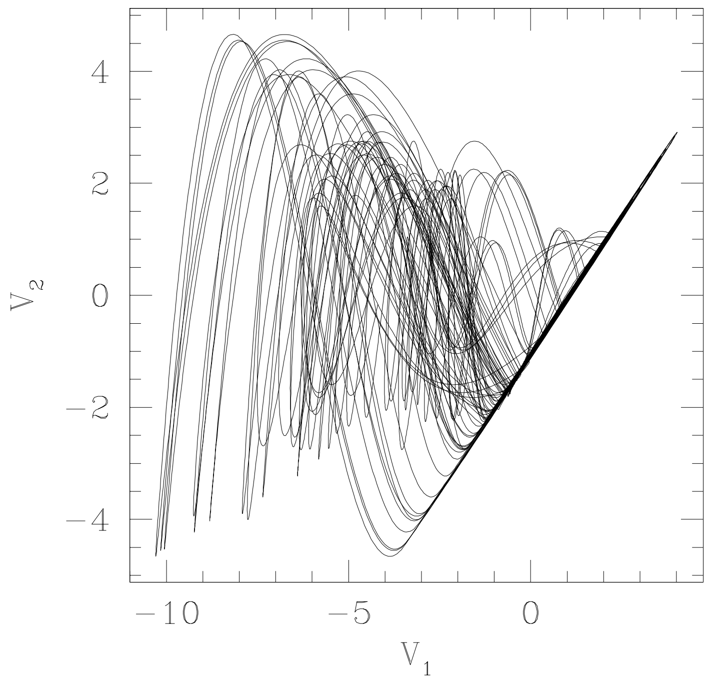
Consider the realisation of four coupled nonlinear (rescaled) ODEs of the form:

$$\begin{aligned}
\dot{x}_1 &= (k-2)x_1 - x_2 - G(x_1 - x_3) \\
\dot{x}_2 &= (k-1)x_1 - x_2 \\
\dot{x}_3 &= -x_4 + G(x_1 - x_3) \\
\dot{x}_4 &= \beta x_3
\end{aligned}$$

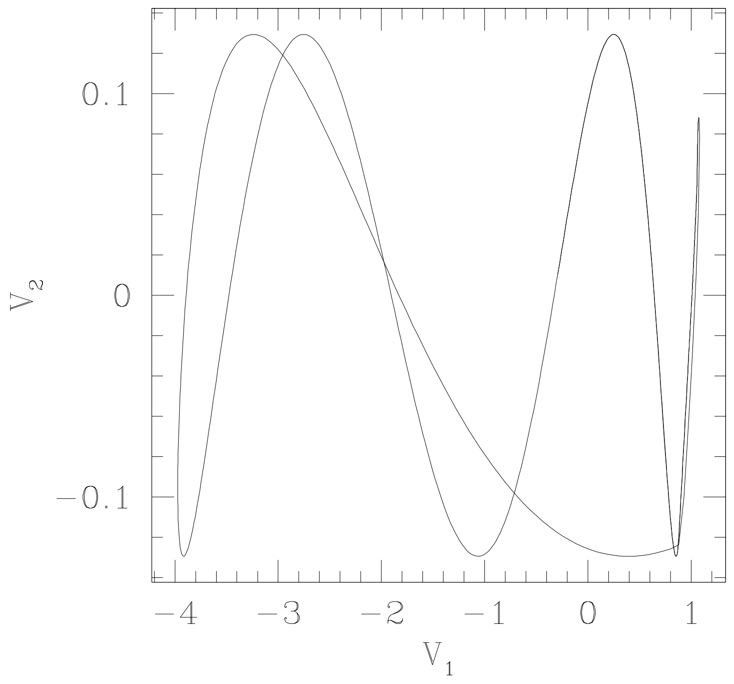
where

$G(x_1 - x_3) = \frac{1}{2}b[|x_1 - x_3 - 1| + (x_1 - x_3 - 1)]$
with $k = 3.85, b = 88$ and $\beta = 18$

Hyper Chaotic Attractor

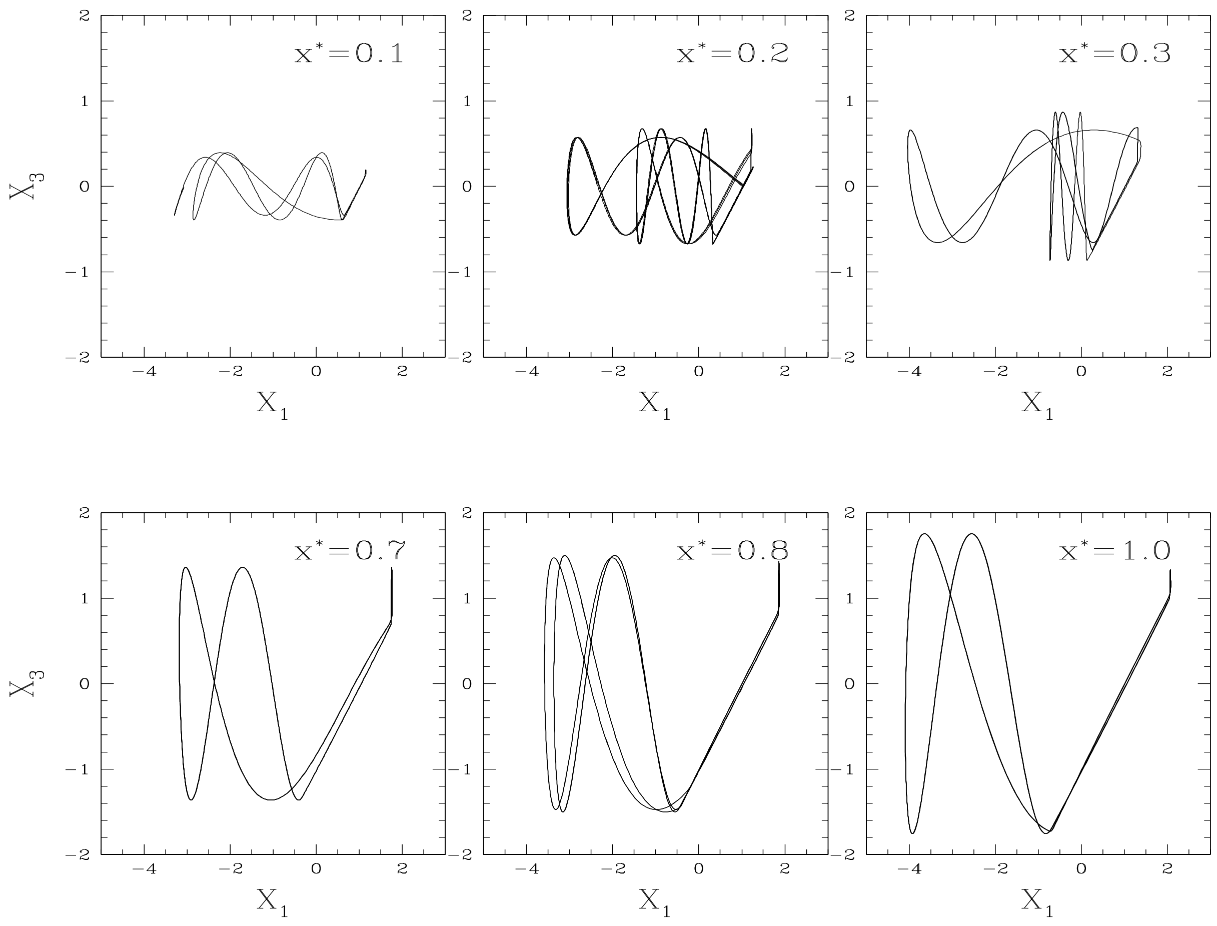


Controlled Orbit



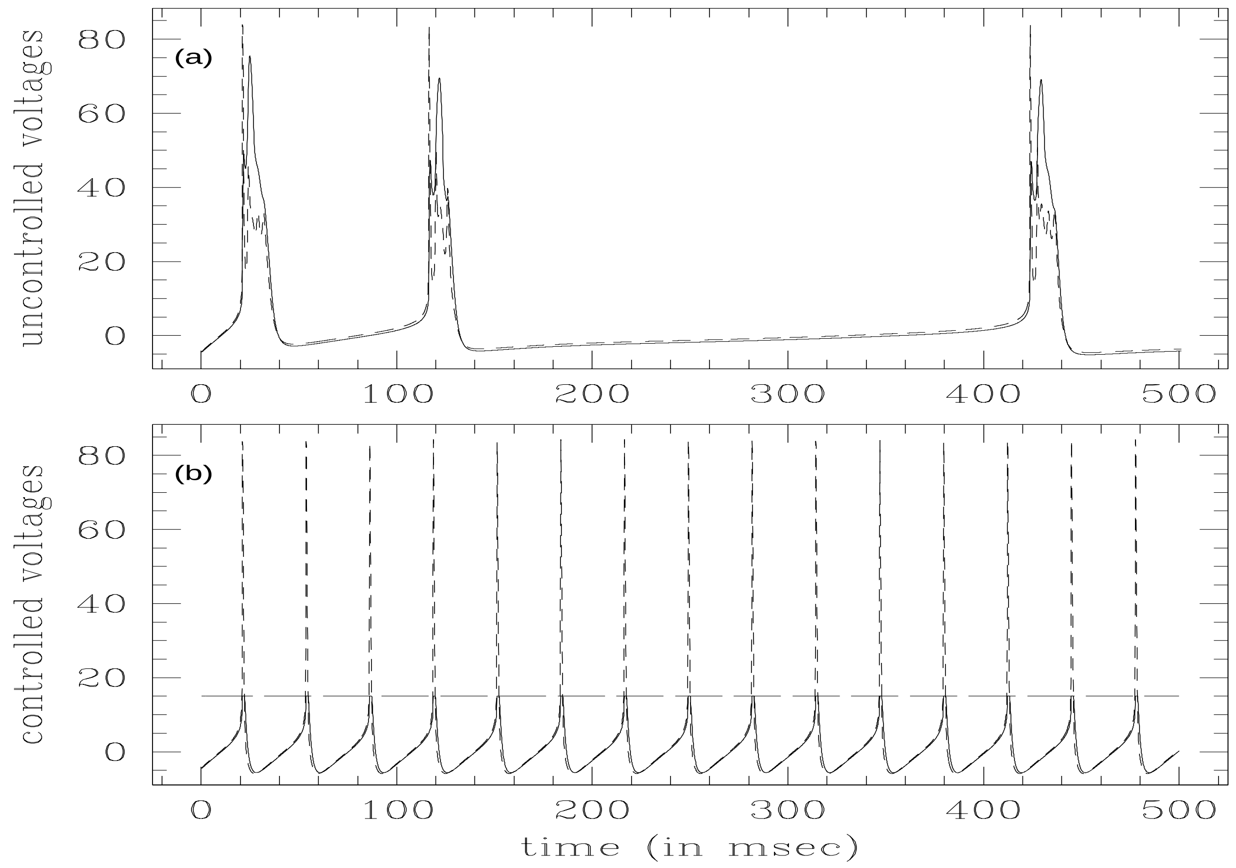
Murali and Sinha, Physical Review E, 2003

“Library of Patterns”



Simple thresholding selects out a wide variety of patterns

Pinsky-Rinzel Neuron : Controlling Spiking
8 coupled ODEs : thresholding one variable



Sinha and Ditto, Physical Review E, 2001

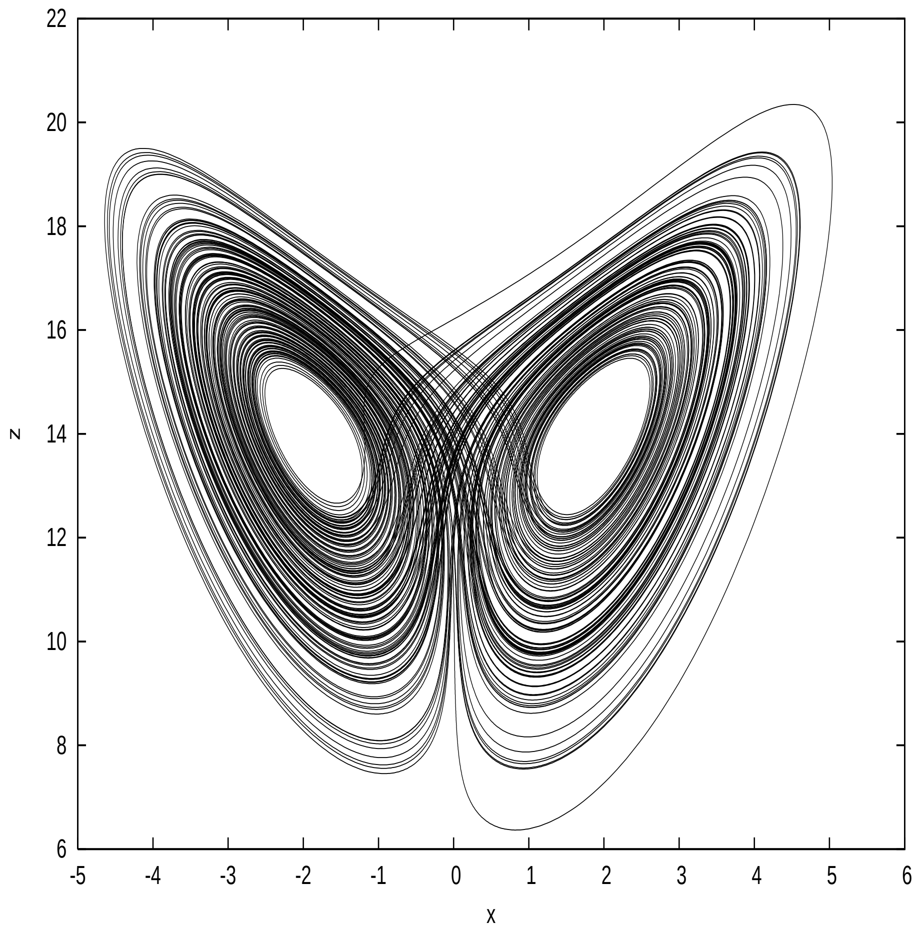
Laser System:

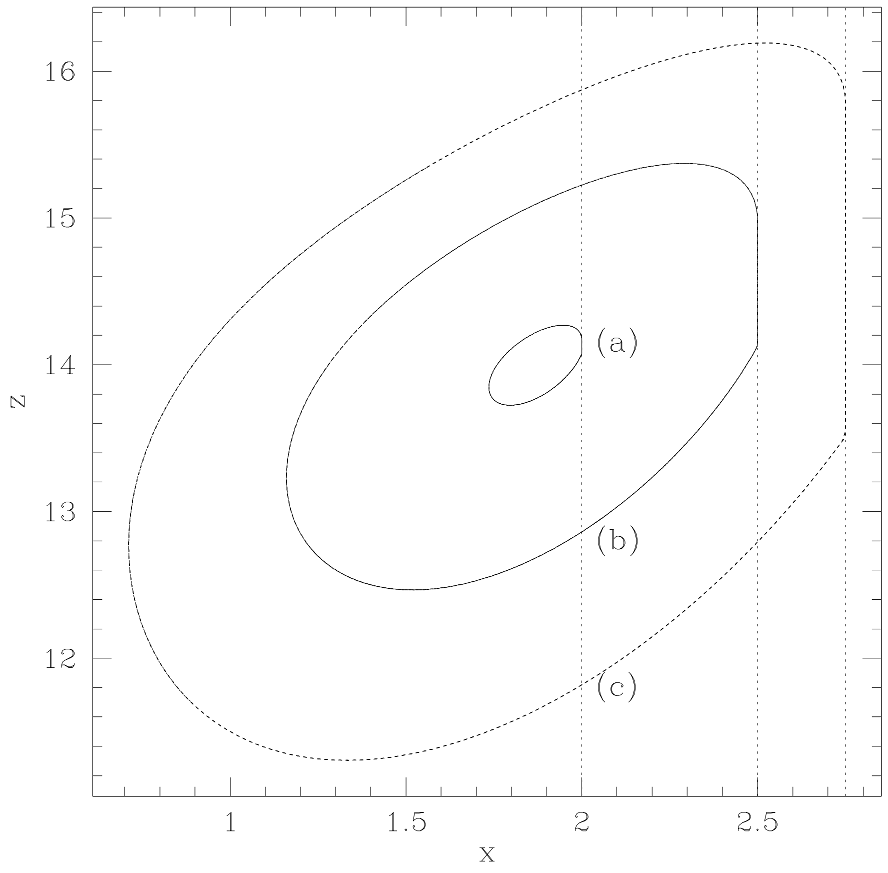
$$\begin{aligned}\dot{x} &= \sigma(y - x)\\ \dot{y} &= rx - y - xz\\ \dot{z} &= xy - bzr\end{aligned}$$

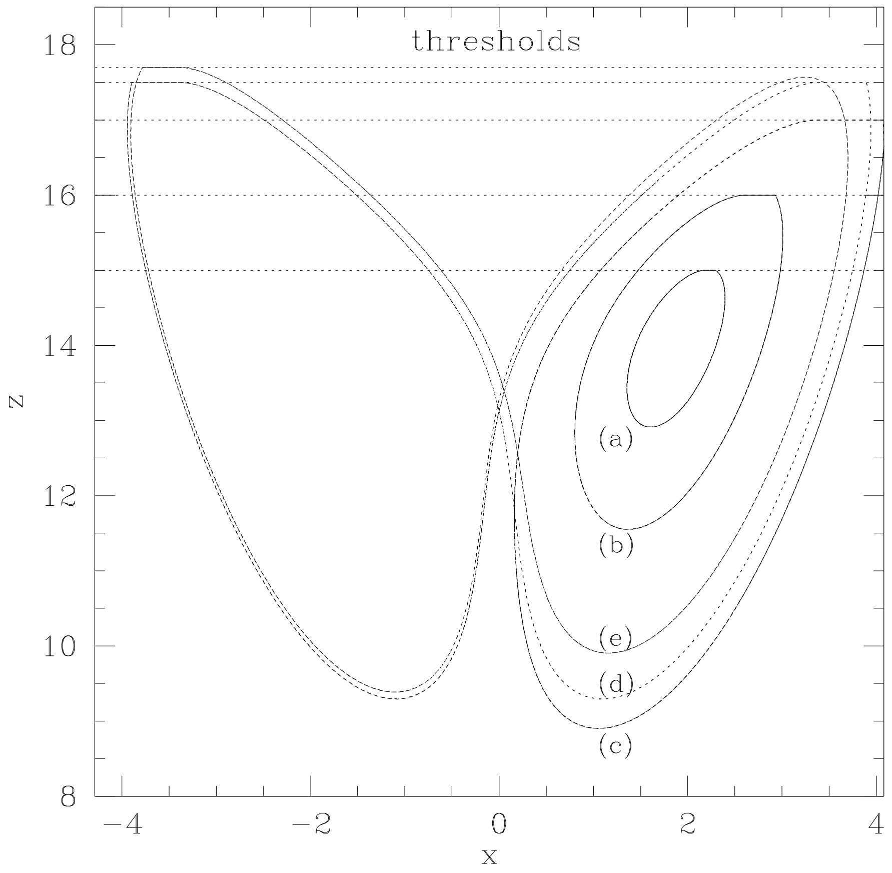
z variable corresponds to the normalized inversion
x and y variables correspond to normalized amplitudes of the electric field and atomic polarisations

Parameter values, obtained by detailed comparison with experiments, for the corresponding coherently pumped far-infrared ammonia laser system are: $\sigma = 2$, $r = 15$ and $b = 0.25$

Laser System: Lorenz-like Attractor




16
15
14
13
12
z
1
1.5
2
2.5
x
(a)
(b)
(c)



Sinha and Ditto, Physical Review E, 1999

Exploiting Chaos to Design Flexible Hardware

Opportunities offered by Chaos:

- Chaos has embedded in it a rich variety of temporal sequences
- Can yield a large range of controlled responses from very simple mechanisms :

  Thus can serve as a versatile pattern generator
- Exploit this flexibility for implementing computational tasks
- Determinism : allows “designing”

Hardware : Threshold activated chaotic elements
Chaotic Chip, Chaotic Processor

Programming these elements consists of fixing thresholds
such that some desired operation is performed
i.e. certain I/O relations are satisfied

Sinha & Ditto, Physical Review Letters, September 1998
Physcial Review E, 1999

Aim :

Demonstrate the direct implementation of all the logic gates which are basic and sufficient components of computer architecture today

Implement all gates using a single chaotic element : with the ability to switch between different operational roles

This will allow a more dynamic reconfigurable architecture

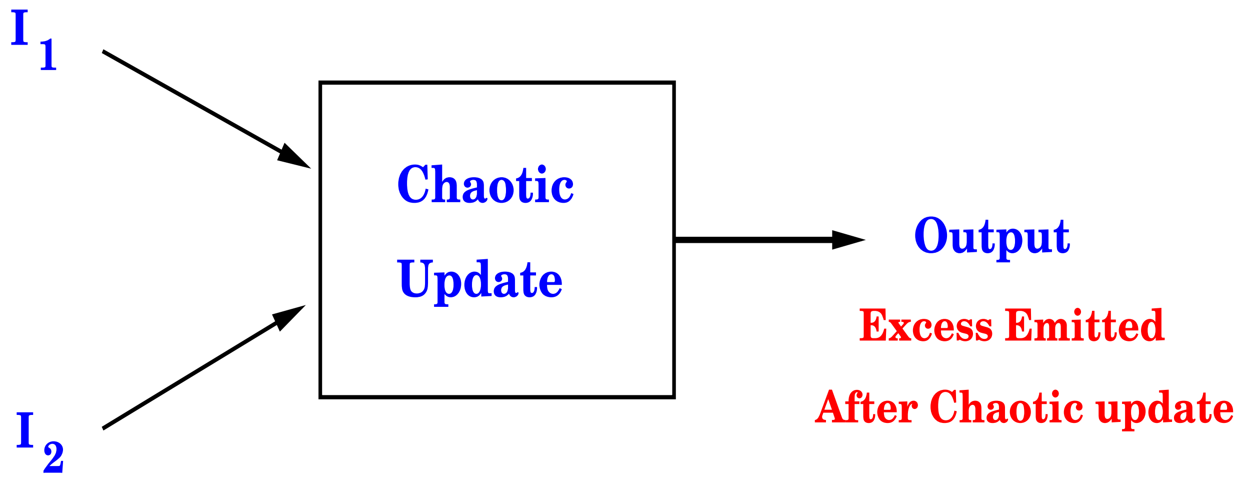
Serve as ingredients of a general purpose device more flexible than statically wired hardware

Chaotic system as a Logic Cell

Inputs : State of the chaotic element $x \quad \rightarrow \quad x_0 + I_1 + I_2$

Output : Obtained by Threshold Mechanism after Chaotic Update $\quad f(x)$

$$\begin{array}{lll} O = f(x) - x^\star & \text{if} & f(x) > x^\star \\ O = 0 & \text{if} & f(x) < x^\star \end{array}$$



The truth table of the basic logic operations :

2 Inputs - 1 Output module

I_1	I_2	AND	OR	XOR	NAND	NOR	I	NOT
0	0	0	0	0	1	1	1	1
0	1	0	1	1	1	0	0	0
1	0	0	1	1	1	0		
1	1	1	1	0	0	0		

The 2 (symmetric) inputs for the AND, OR, XOR, NAND and NOR gates are I_1 and I_2

The 1 input for the NOT gate is I

Necessary and Sufficient conditions to be satisfied simultaneously

AND	OR	XOR
$f(x_0) \leq x^*$	$f(x_0) \leq x^*$	$f(x_0) \leq x^*$
$f(x_0 + I) \leq x^*$	$f(x_0 + I) - x^* = I$	$f(x_0 + I) - x^* = I$
$f(x_0 + 2I) - x^* = I$	$f(x_0 + 2I) - x^* = I$	$f(x_0 + 2I) \leq x^*$

NAND	NOR	NOT
$f(x_0) - x^* = I$	$f(x_0) - x^* = I$	$f(x_0) - x^* = I$
$f(x_0 + I) - x^* = I$	$f(x_0) - x^* \leq I$	$f(x_0 + I) \leq x^*$
$f(x_0 + 2I) \leq x^*$	$f(x_0) - x^* \leq I$	

Robust solutions exist :

Operation	AND	OR	XOR	NAND	NOT
x_0	0	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{3}{8}$	$\frac{1}{2}$
x^*	$\frac{3}{4}$	$\frac{11}{16}$	$\frac{3}{4}$	$\frac{11}{16}$	$\frac{3}{4}$

Scheme has been experimentally verified

Sinha, Munakata & Ditto, Phys. Rev. E, 2002

Munakata, Sinha & Ditto, IEEE Trans. on Circuits and Systems, 2002

Contrast with periodic elements:

It is not possible to extract all the different logic responses from the same element in case of periodic components, as the temporal patterns are inherently limited.

Contrast with random elements:

One cannot design components : need determinism for “reverse engineering”

Only Chaotic dynamics enjoys both

richness
and
determinism

So one can select out all the different temporal responses necessary to obtain all the different logic patterns with a single evolution function

This ability allows us to construct flexible hardware

Implementation of Parallel Logic Operations

Objective : Obtain N clearly defined logic gate response patterns from the N components characterizing the state of a N-dimensional system

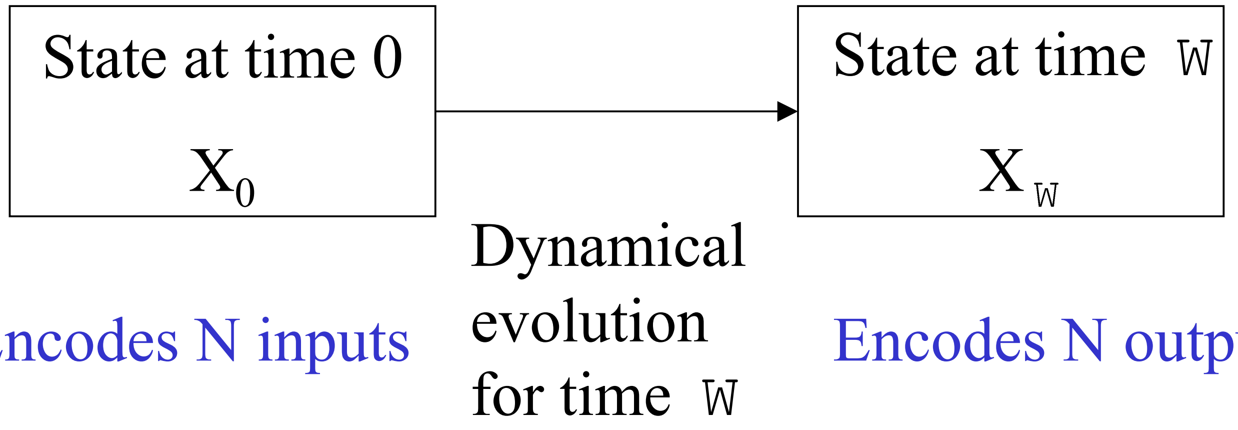
This will enable us to implement N operations in parallel with a single N-dimensional element

Thus one can gain processing power without enhancing space costs

Can execute several operations concurrently

Dynamical System

of dimension N: $\{X\} = (x_1, x_2, \ldots, x_{N)}$



Encodes N inputs

$(I_1, I_2, \ldots, I_N)$

Encodes N outputs

$(O_1, O_2, \ldots, O_N)$

Implementation of parallel logic by a 2-dimensional map

Consider a 2-d model for neurons :

$$x_{n+1} = x_n^2 \, \exp(y_n - x_n) + k$$

$$y_{n+1} = a y_n - b x_n + c$$

Implement bit-by-bit addition in parallel

I_1	I_2	O_1	O_2
0	0	0	0
0/1	1/0	1	0
1	1	0	1

Bit by bit arithmetic addition

Involves 2 logic operations:

O_1 is the first digit of the sum (rightmost) : determined by XOR

O_2 is the carry of the answer :determined by AND

x	variable	implements	XOR
y	variable	implements	AND

Necessary and sufficient conditions for parallelized bit-by-bit arithmetic addition

Initial State	XOR	AND
$x^\star, \quad y^\star$	$x_n < x^\star$	$y_n < y^\star$
$x^\star + I, \quad y^\star + I$	$x_n = x^\star + I$	$y_n < y^\star$
$x^\star + 2I, \quad y^\star + 2I$	$x_n < x^\star$	$y_n = y^\star + I$

I is a common positive constant for the operations

Large bands of solutions exist, satisfying the table of simultaneous conditions

Similar considerations for other parallel logic operations can be straight-forwardly formulated

Sinha, Munakata & Ditto, Phys. Rev. E, 2002

Morphing Dynamic Logic Cell :

Simple mechanism allows one to switch with ease between behaviours emulating different logic gates

Universal General Purpose computing device

This Dynamical Logic Architecture is more versatile than static hardware

Building blocks of Programmable hardware ; Re-configurable hardware

Pre-determined or out-come dependent dynamic logic configuration

Possibility of the hardware design evolving during the computation

Arrays of such flexible units can conceivably be programmed on the run to give the optimal hardware for the task at hand

For instance, may serve flexibly as an arithmetic processing unit or a component of memory, as the need demands, and can be swapped to be one the other

The hope is that reconfigurable chaotic computer chips will enable us to achieve within the same computer chip architecture :

- Flexibility of field programmable gate arrays (FPGA)
- Optimization and speed of application specific integrated circuits (ASIC)
- General utility of a central processing unit (CPU)

CHAOLOGIX : developing a VSLI implementation of chaotic computing in a demonstration integrated circuit chip

First generation of chip: 0.18 μm (January 2007)

Exploiting Nonlinear Dynamics to Encode and Process Information

with Abraham (Aris) Miliotis , William L. Ditto

2007

Information storage is a fundamental function of computing devices

Computer memory is implemented by computer components that retain data for some interval of time

Storage devices have progressed from punch cards and paper tape to magnetic, semiconductor and optical disc storage by exploiting different natural physical phenomena to achieve information storage

For instance, the most prevalent memory element in electronics and digital circuits is the flip-flop or bistable multivibrator which is a pulsed digital circuit capable of serving as a one-bit memory

Namely storing value 0 or 1

More meaningful information is obtained by combining consecutive bits into larger units

Here we will suggest a different direction in designing information storage devices:

We will implement data storage schemes based on the wide variety of controlled patterns that can be extracted from nonlinear dynamical systems

Just as we were able to demonstrate that chaotic systems can be morphed into flexible reconfigurable logic units, we will now demonstrate that they can also be morphed into a more general and versatile version of Content Addressable Memory (CAM)

- We will demonstrate the use of arrays of nonlinear elements to stably encode and store various items of information (such as patterns and strings) to create a database

- Further we will demonstrate how this storage method also allows one to efficiently determine the number of matches (if any) to specified items of information in the database

- So the nonlinear dynamics of the array elements will be utilized for flexible-capacity storage

- Also for pre-processing data for exact (and inexact) pattern matching tasks

Encoding information

We consider encoding N data elements (labeled as $j = 1, 2, \ldots, N$) by N nonlinear elements, with state $X_n^i[j]$ ($j = 1, 2, \ldots, N$)

Each dynamical element stores one element of the database, encoding any one of M distinct items (labeled as $i = 1, 2, \ldots, M$)

N can be arbitrarily large

M is flexible : determined by the kind of data being stored

- For instance for storing English text one can consider the letters of the alphabet to be the natural distinct items building the database, namely M=26
- For the case of data stored in decimal representation M=10
- For databases in bioinformatics comprised typically of symbols A, T, C, G, one has M=4
- One can also consider longer strings and patterns as the items:

  For instance for English text one can also consider the keywords as the items

Now we demonstrate how a single nonlinear dynamical element can store M items, where M is variable and can be large

This provides the capability for naturally storing data in different bases or in different alphabets or multilevel logic

Now in order to hold information one must confine the dynamical system to a fixed-point behavior:

i.e. a state that is stable and constant throughout the dynamical evolution of the system

We employ a threshold mechanism to achieve this

Typically a wide window of threshold values can be found where the system is confined on fixed points namely the state of the element under such thresholding is stable at $T^i[j]$:

i.e. $X_n^i[j] = T^i[j]$ for all times n

We will use such a window to encode a stable database

So each element is capable of yielding a continuous range of fixed points

As a result it is possible to have a large set of thresholds $\{T^1, T^2, \ldots T^M\}$ each having a one-to-one correspondence with a distinct item of our data

So the number of distinct items that can be stored in a single dynamical element is typically large, with the size of M limited only by the precision of the threshold setting

In particular, consider a collection of storage elements that evolve in discrete time n according to chaotic tent maps:

$$X^i_{n+1}[j] = f(X^i_n[j]) = 2\min(X^i_n[j], 1 - X^i_n[j])$$

with each element storing one element of the given database ($j = 1, \ldots N$)

Each element can hold any one of the M distinct items indicated by the index i

A threshold will be applied to each dynamical element to confine it to the fixed point corresponding to the item to be stored

For this map, thresholds ranging from 0 to 2/3 yield fixed points with the variable $X^i[j]$ held at $T^i[j]$

This can be seen exactly from the fact $f(T^i[j]) > T^i[j]$ for all $T^i[j]$ in the interval $(0, 2/3)$ implying that an iteration of a state at $T^i[j]$ will always exceed $T^i[j]$ and thus be re-set to $T^i[j]$

In our encoding the thresholds are chosen from the interval $(0, 1/2)$ namely a sub-set of the fixed point window (0,2/3)

Defining a resolution r between each integer as:

$$r = \frac{1}{2}\frac{1}{(M+1)}$$

Threshold $T^i[j]$ encoding positive ingtegers $i \in [1, M]$ is

$$T^i = i\, r$$

Lookup map from the encoded number to the threshold

- Therefore we obtain a direct correspondence between a set of integers ranging from 1 to M, where each integer represents an item, and a set of M threshold values
- So we can store N multi-level database elements by setting appropriate thresholds on N dynamical elements

Processing Information

Once we have a given database stored by setting appropriate thresholds on N dynamical elements we can query for the existence of a specific item in the database using one global operational step

This is achieved by globally shifting the state of all elements of the database up by the amount that represents the item searched for

Search operation:

Globally applied shift : $X_n^i[j] \rightarrow X_n^i[j] + Q^k$

where Search Key Q^k is

$$Q^k = \frac{1}{2} - T^k$$

where k is the number being queried for

i.e. value of the search key is simply $\frac{1}{2}$ minus the threshold value corresponding to the item being searched for

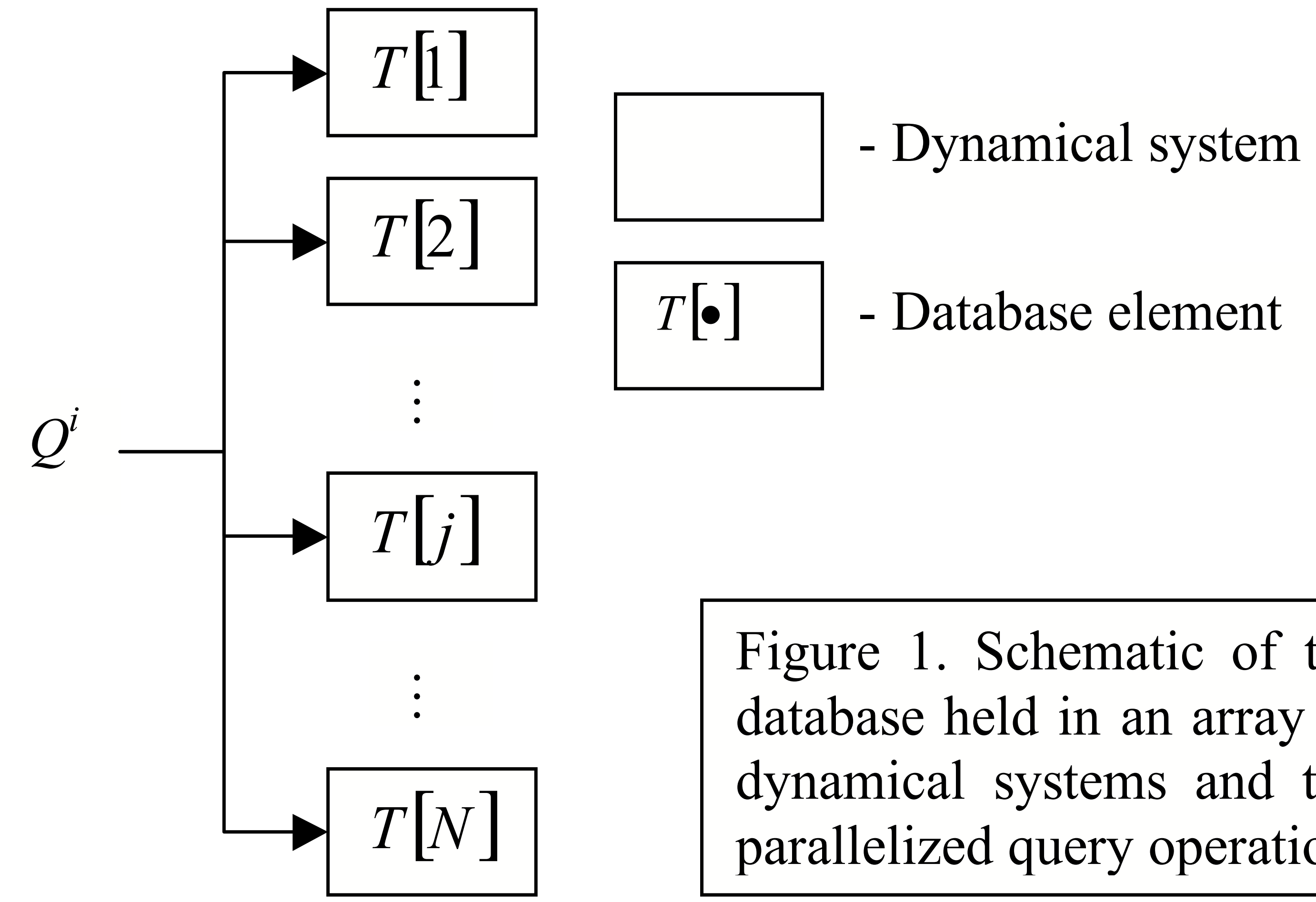


Figure 1. Schematic of the database held in an array of dynamical systems and the parallelized query operation.

Notice that the information item being searched for is coded in a manner complimentary to the encoding of the information items in the database

Much like a key that fits a particular lock

Namely $Q^k + T^i[j]$ adds up to $\frac{1}{2}$

This guarantees that only an item matching the one being searched for will have its state shifted to $\frac{1}{2}$

The value of $\frac{1}{2}$ is special in that it is the only state value that on subsequent update will reach value of 1 which is the maximum state value for the system

So only the elements holding an item matching the queried item will reach extremal value 1.0 on the dynamical update following a search query

Note that the important feature here is the nonlinear dynamics that maps the state $1/2$ to 1, while all other states, both higher and lower than $1/2$, get mapped to values lower than 1: so the matching state is selected out

This scheme provides us with a global monitoring operation on the unsorted database

And pushes the state of all the elements matching the queried item to the unique maximal point, in parallel

- The crucial ingredient here is the nonlinear evolution of the state, which results in folding
- Chaos is not strictly necessary here
- It is evident though, that for unimodal maps higher nonlinearities allows larger operational ranges for the search operation, and also enhances the resolution in the encoding
- For the tent map, specifically, it can be shown that the minimal nonlinearity necessary for the above search operation to work is in the chaotic region
- Another specific feature of the tent map is that its piecewise linearity allows the encoding and search key operation to be very simple indeed

To complete the search we now must detect the maximal state at 1.0

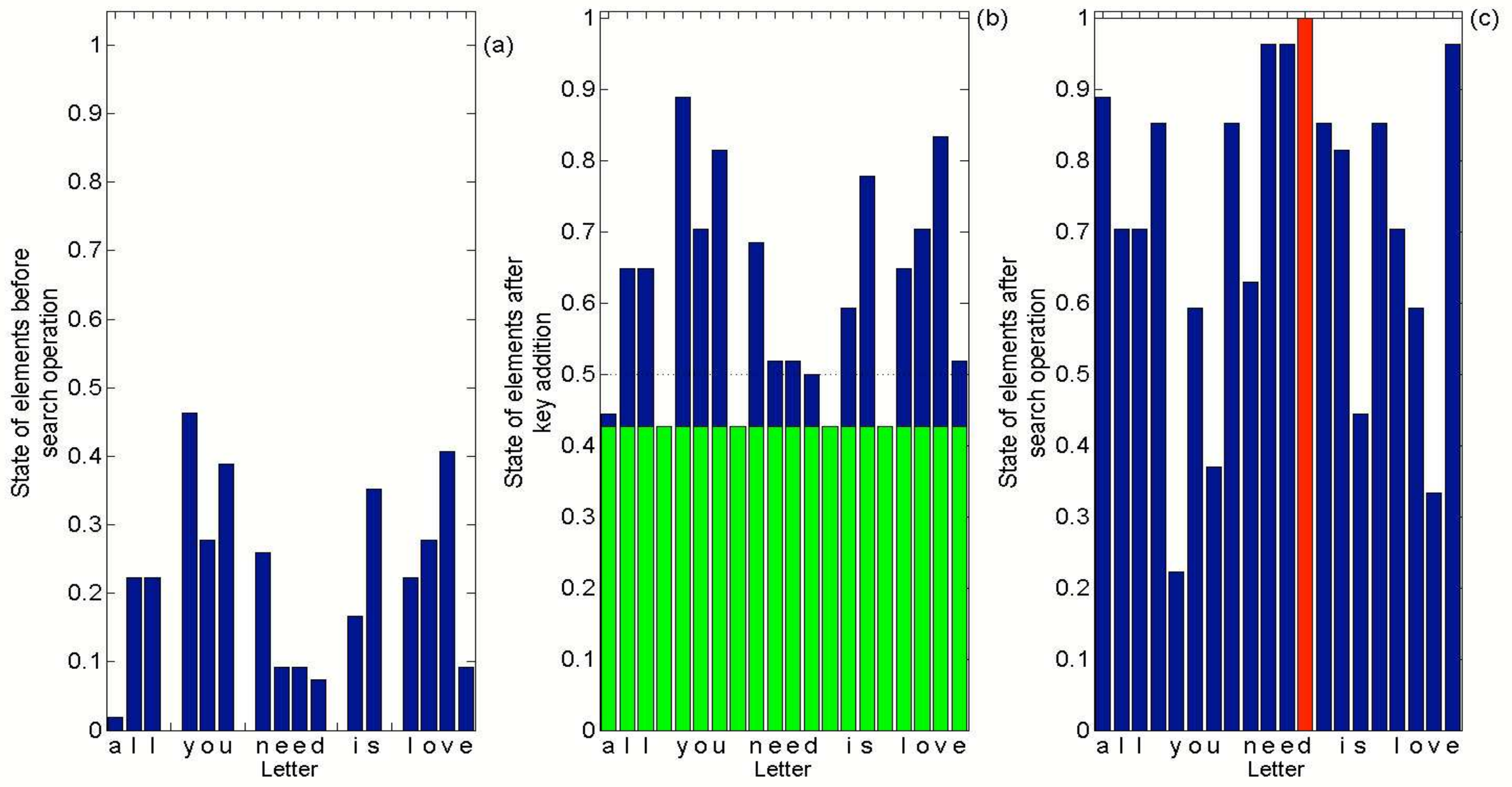
This can be accomplished in a variety of ways

For example, one can simply employ a level detector to register all elements at the maximal state

This will directly give the total number of matches, if any

So the total search process is rendered simpler as the state with the matching pattern is selected out and mapped to the maximal value, allowing easy detection

Representative Example: English language text $M = 26$



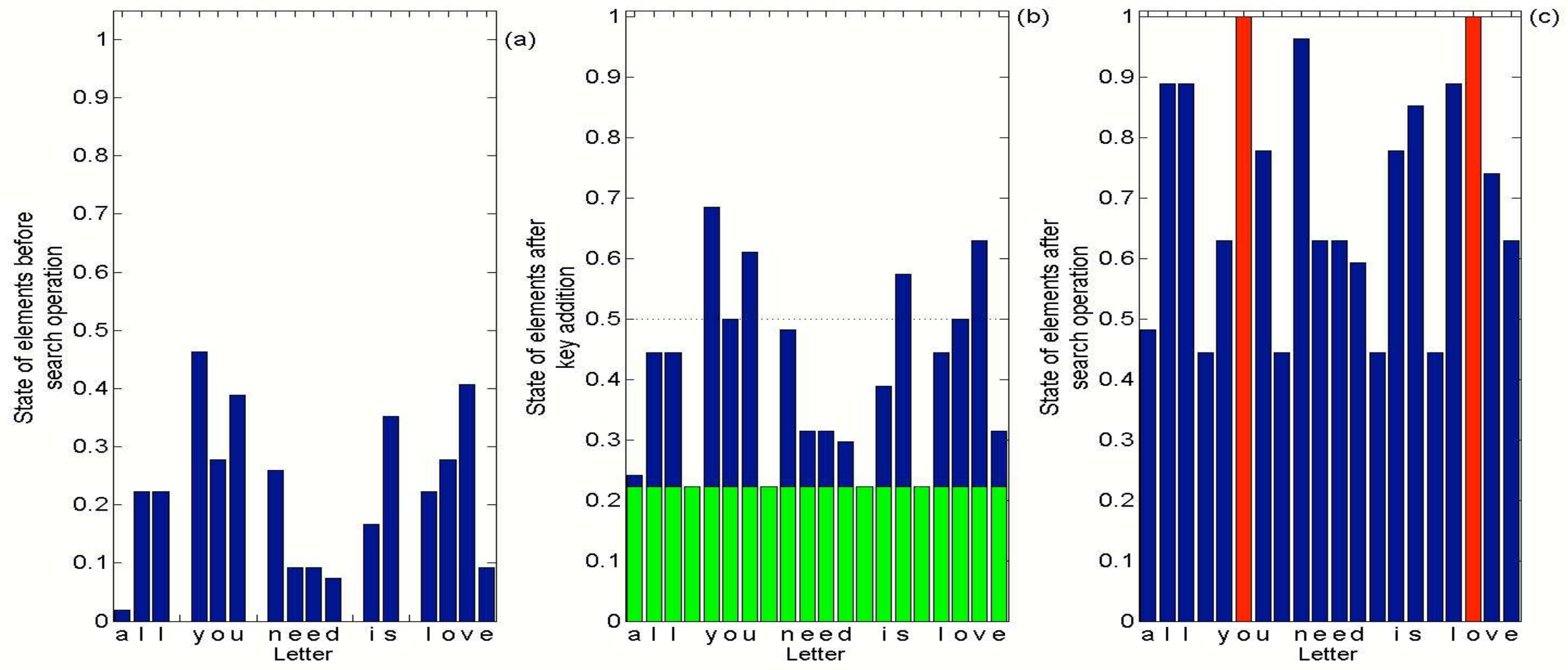


Figure 3. (a) Threshold levels encoding the sentence “all you need is love”, (b) the search key value for letter “o” is added to all elements, (c) the elements update to the next time step. For clarity we marked with red any elements that reach the detection level.

Further, by relaxing the detection level by a prescribed tolerance we can check for the existence within our database of numbers or patterns that are close to the number or pattern being searched for

This allows detection of Inexact Matches

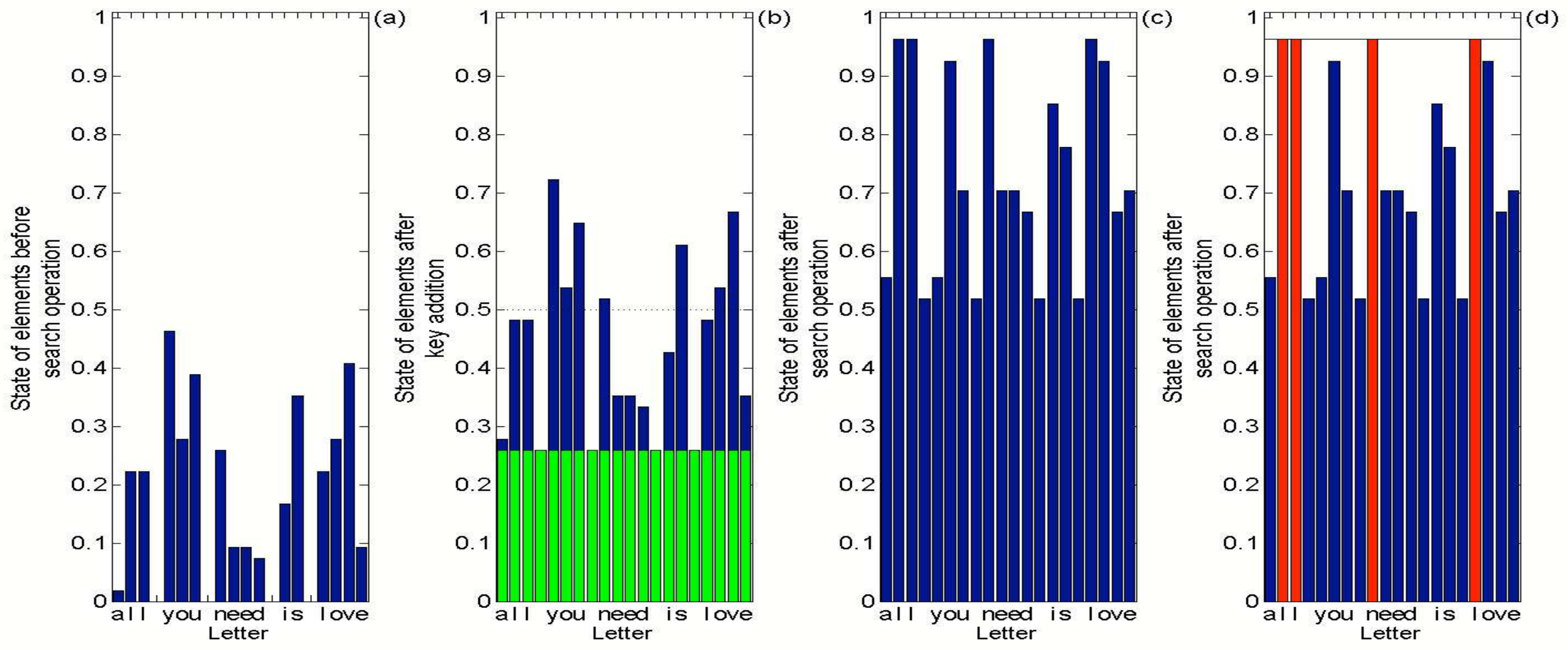


Figure 4. (a) Threshold levels encoding the sentence “all you need is love”, (b) the search key value for letter “m” is added to all elements, (c) the elements update to the next time step. It is clear that no elements reach the detection level at 1. (d) By lowering the detection level we can detect whether “adjacent” items to “m” are present.

A significant feature of this scheme is that it employs a single simple global shift operation

Does not entail accessing each item separately at any stage : great scope for parallelism

Implies that there is no scale-up (in principle) with size N

In terms of the timescales of the processor the search operation requires one dynamical step, namely one unit of the processor's intrinsic update time

So nonlinear dynamics works as a powerful preprocessing tool:

Reduces the determination of matching patterns to the detection of maximal states, an operation that can be accomplished by simple means, in parallel

The search effort is considerably minimized as it utilizes the native processing power of the nonlinear dynamical processors

One can then think of this method as a natural application of a computing machine consisting of chaotic modules

It is also equally potent as a special-applications “search chip”, which can be added on to regular circuitry, and should prove especially useful in machines which are repeatedly employed for selection/search operations

Outlook

- Nonlinear systems are abundant in engineered and natural systems : ranging from fluids to electronics to optics
- Attempted to harness the abundantly available chaotic phenomena for the development of a reconfigurable computing device (morphing chip)