Projects in Chaotic Dynamics: Spring 2011

Elizabeth Bradley, Editor

Technical Report CU-CS 1081-11

University of Colorado
Department of Computer Science
Boulder CO 80309-0430 USA

May 2011

Table of Contents

Allison Brown
May 10, 2011

A "Knead" For Chaos: The Culinary Kneading Process as an Explanatory Metaphor in Chaotic Systems

Introduction

Kneading is frequently used as a metaphor in chaotic texts. Authors make various claims about how kneading demonstrates different chaotic behaviors, most particularly sensitive dependence on initial conditions , but those claims do not appear to have not been verified. My project examines the various claims authors make, as well as their validity. I will explain the experiments I performed with regard to these claims, and provide analysis of their results. I will also briefly discuss kneading theory, and how it can be used to gain information about bifurcation of unimodal mappings (among many other uses and applications). I will finish by outlining conclusions and possible future work.

Kneading as Metaphor in Dynamical Systems

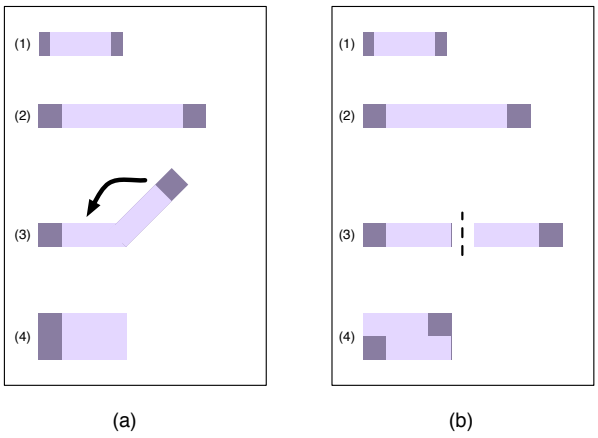


Figure 1: (a) A diagram depicting the steps for the stretch and fold method of kneading; (b) A similar diagram for the stretch, cut, and paste method.

Those who use kneading to illustrate chaotic concepts have two definitions of the steps that are repeated to make up the kneading process. These methods can be simply described as the "stretch and fold" and "stretch, cut, and paste" techniques. In the first method, a blob of dough is stretched in one dimension until it has doubled in length, then folded in half and compressed. This process is repeated as necessary. The second method also stretches the dough to twice its length in one dimension, but then cuts that piece of dough in half and stacks one piece on top of the other. Diagrams of both of these methods can be found in figure 1.

Steven Strogatz asserts that if a drop of food dye is added to a mass of dough, the dye will evenly color that mass after the dough has been kneaded using the stretch and fold method[5]. Peitgen et al. state in [4] that kneading with the stretch and fold method will guarantee a that "pocket of spices" added in one spot will be evenly distributed throughout the doughs mass. The

dye and the cluster of spices each represent nearby initial conditions, and the even distribution of each demonstrates sensitive dependence on initial conditions.

Peitgen et al. describe the second (stretch, cut, and paste) method of kneading and state that, while in theory the methods should have different and distinct "iteration behaviors," the respective dynamic behaviors of the two methods are actually quite similar[4]. They continue by describing an idealized situation that neglects dough thickness and describes how the two methods can be used interchangeably.

Experiments

To test the claims made by Strogatz, Peitgen et al., and others in the field, I designed a two experiments. In the first, I tested whether a drop of dye added to dough would evenly color the dough after many iterations of kneading. In the second, if the first proved successful, I added clusters of beads to the dough, and carefully tracked where each bead ended up. Since it is not possible to ignore dough thickness, I did not test the interchangeability of the two methods as described in Peitgen et al.'s idealized kneading process.

Experiment 1: Dye Diffusion

For this experiment I added a drop of dye to two different batches of dough. I kneaded each batch about 13 times[1] with one of methods described in the previous section. The experiment's purpose was to determine if, after a period of time, both methods of kneading would have evenly dispersed the dye throughout the dough mass.

Method 1: Stretch and Fold

The progress of the dye throughout the kneading process can be seen in figure 2. It seems fairly clear that Strogatz is right, and that after several iterations of kneading, a drop of dye will be evenly distributed through the entire volume of dough.

Method 2: Stretch, Cut, and Paste

The progress of the dye throughout this second kneading process can be seen in figure 3. It again seems fairly clear that after several iterations, the drop of dye will be evenly distributed.[2] It appears that Peitgen et al. may be right, and that the two methods are dynamically equivalent.

Experiment 2: Bead Trajectories

Since both parts of the previous experiment indicated that kneading does create sensitive dependence on initial conditions, I moved on to create a second experiment that tracked the dynamics in the bulk. This time, I added a cluster of differently colored beads to two batches of dough. In order to measure how different initial conditions progressed through the dough mass, I designed a "dough squisher," depicted in figures 4 and 5. The purpose of the squisher is twofold. First, I wanted to be able to flatten the dough as uniformly as possible. Traditional, culinary kneading's stretching

[1] I used a homemade play dough for my experiments. (See the final section of this paper for an explanation of why I made this choice.) This type of dough requires no kneading, so I kneaded until dye was evenly distributed, which took approximately 13 "kneads" in both cases. Bread dough might require more kneading, see [2] for more.

[2] The color is somewhat washed out in the lower two pictures. In reality, the doughs were similarly dark in pigment by the end of the kneading processes.

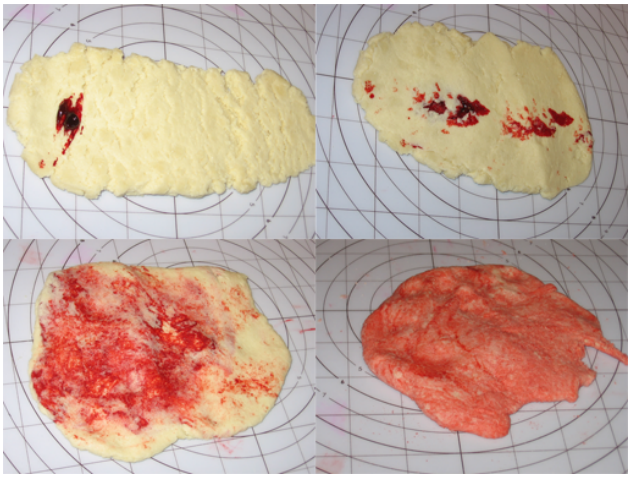


Figure 2: A drop of dye progressing through a mass of dough using the stretch and fold method; shown at the "stretch" phase of the process. Clockwise from upper left: just after dye was added, after 1 iteration, after about 7 iterations, and after kneading was complete and dye was fully distributed (about 13 iterations).

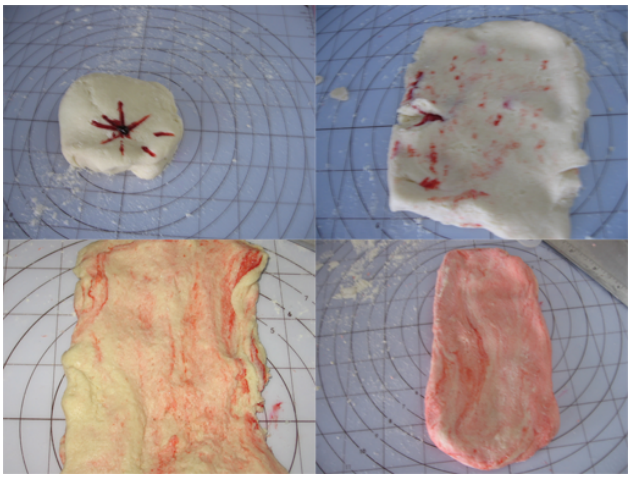


Figure 3: As in figure 2, pictures showing a drop of dye progressing through a mass of dough; though this time using the stretch, cut, and paste method, again shown at the "stretch" phase of the process. Clockwise from upper left: just after dye was added, after 1 iteration, after about 7 iterations, and after kneading was complete and dye was fully distributed(about 13 iterations).

phase is not necessarily uniform but is a more organic and variable process. The stretching phase described in metaphors for chaos is usually described more rigidly. Since adding an element of consistency in that phase would not change the end result of culinary kneading[3], it seemed best to try to replicate the kneading processes described by the authors. Second, I needed to track the bead positions as accurately as possible. Each side of the squisher has a grid to make this process easier (see again figure 5). The kneading processes for this experiment were identical to those previously described, except that the stretching phase of each was performed in the squisher.

[3]Tested by the author in the kitchen.

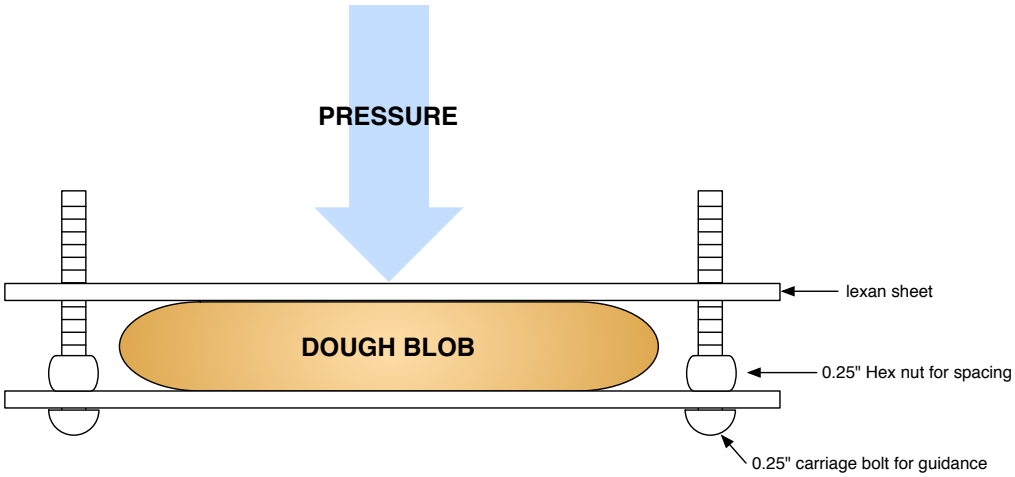


Figure 4: The "dough squisher" from the side.

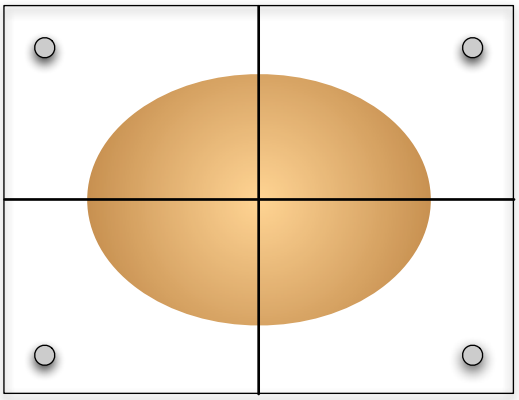


Figure 5: Top view of the "dough squisher."

Method 1: Stretch and Fold

The pictures of individual bead progress through the dough using this kneading method can be seen in figure 14. As can be seen in those images, the beads begin in a close cluster of starting positions but take different paths through the dough mass, which again demonstrates sensitive dependence on initial conditions . An image with all positions of all beads, found in figure 6, further confirms Strogatz's statement that closely clustered initial conditions will travel throughout the dough mass. This is more clearly demonstrated by the experiment with dye.

Method 2: Stretch, Cut, and Paste

The pictures of individual bead progress through the dough can be seen in figure 15. As shown in those figures, the beads start with close initial conditions but follow different trajectories through the dough, which demonstrates sensitive dependence on initial conditions in this system. As with the stretch and fold method, an image (figure 7) that shows every position of every bead over time shows that Strogatz's potentially previously untested statement holds for this kneading method.

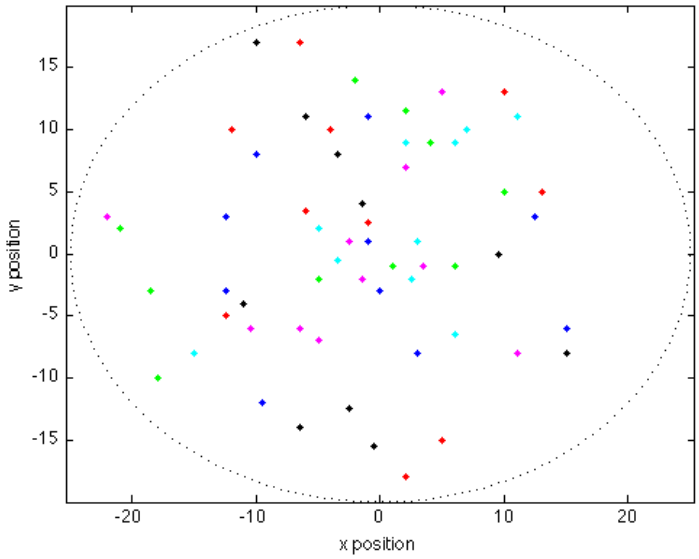


Figure 6: Each bead position at each stretch and fold kneading iteration superimposed on one figure; dotted black circle represents edge of dough.

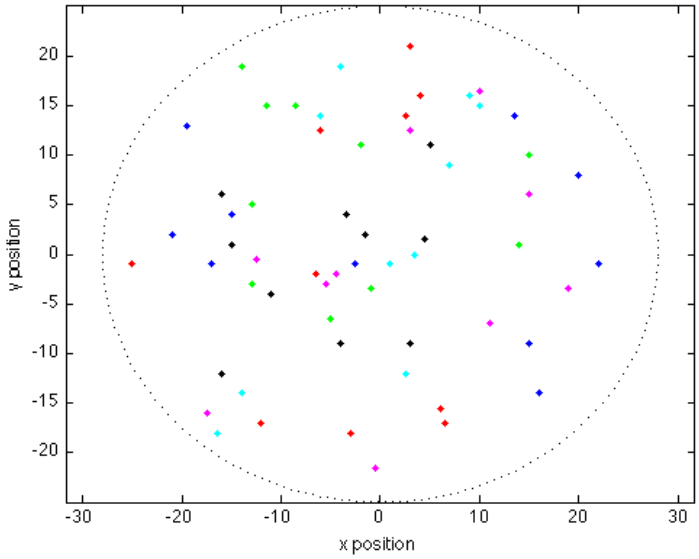


Figure 7: Each bead position at each stretch, cut, and paste kneading iteration superimposed on one figure; dotted black circle represents edge of dough..

Bead Positions as Poincaré Sections

Initially, I had thought to somehow record the full trajectories of beads as they traveled through the dough mass, but time and resource constraints made such an endeavor out of the scope of this project. Instead, as described above, I opted to track bead positions at the stretching phase of kneading using the dough squisher. Bead positions were recorded at this regular interval, which

is arguably the natural period of both kneading methods, so it is appropriate to consider those positions as temporal Poincaré sections of the flow. The image in figure 8 shows the positions of a single bead over ten iterations of stretch-fold kneading. There is no evidence of limit cycles or periodicity here, just the sensitive dependence on initial conditions described previously.

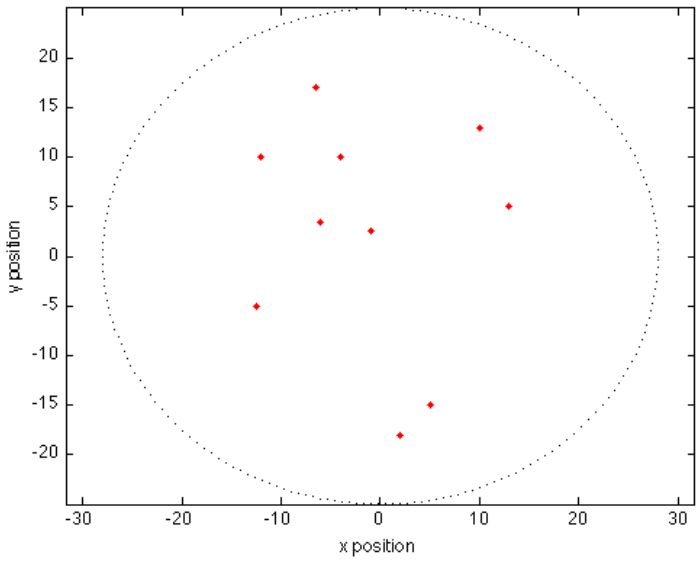


Figure 8: Positions of a single bead over each iteration of the stretch and fold kneading method; dotted black circle represents edge of dough. Such information can be treated as a temporal Poincaré section for the stretch and fold kneading method, and appears to indicate characteristics of chaotic behavior.

We can see similar behavior in figure 9, which shows the positions of a single bead over ten iterations of stretch-cut-paste kneading. Again, there is no evidence of limit cycles or periodicity, so this method also appears to exhibit chaotic behavior. In both cases it is possible that the system settles down to some form of non-chaotic behavior, but it seems unlikely.

Problems Encountered

I encountered one significant as I performed my experiments. The stretching described for both kneading methods indicates the dough should double in length, but maintain the same width (shown on the left in figure 10). The dough squisher "stretches" the dough equally in all directions (right side of the same figure). Even in traditional kneading, a baker will experience some sideways expansion as he/she kneads the dough (see figure 11). It is impossible to stretch the dough in one direction without experiencing some increase in other directions. Bakers deal with this inevitability by periodically rotating the dough mass ninety degrees. In my case, however, performing such a rotation would not fully solve the problem. I could manually re-form the dough into a ball, but this would introduce additional dynamics and obscure some of the very dynamics I was trying to observe. I decided it would be best to perform the folding (or cutting and pasting) steps twice. So, my actual kneading processes were most accurately described as stretch, fold, and fold (figure 12) and stretch, cut and paste, and cut and paste(figure 13). This slight alteration seemed more likely to preserve most of the dynamics than re-forming the dough ball, and solved the problem of dough expanding in several directions.

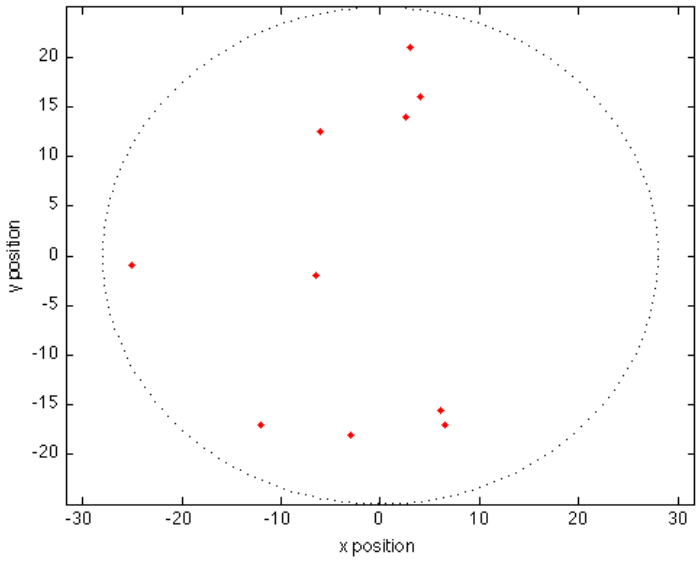


Figure 9: Positions of a single bead over each iteration of the stretch, cut, and paste kneading method; dotted black circle represents edge of dough. As with figure 8, such information can be treated as a temporal Poincaré section for the kneading method, and appears to indicate characteristics of chaotic behavior.

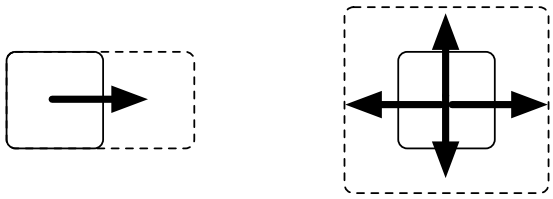


Figure 10: Left side, ideal stretching behavior; right side, actual behavior from "dough squisher."

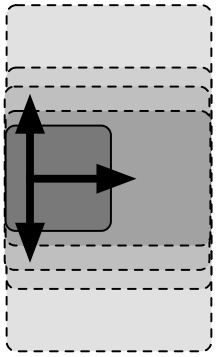


Figure 11: Extra expansion as dough is kneaded in traditional manner; a progressive widening.

Milnor-Thurston Kneading Theory

Since my project is focused on how kneading is used in non-culinary fields, I chose to also examine kneading theory, a rough formalization of kneading as a mathematical model. Kneading theory

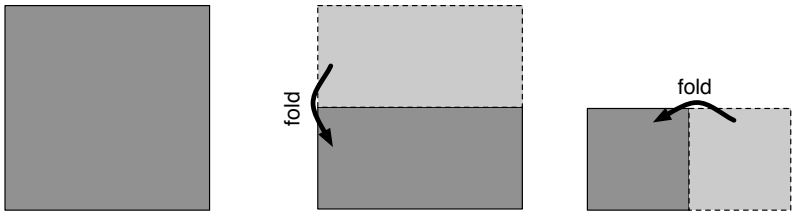


Figure 12: The altered procedure for the stretch and fold method of kneading to avoid the problems of extra expansion in all directions: stretch, fold, and fold.

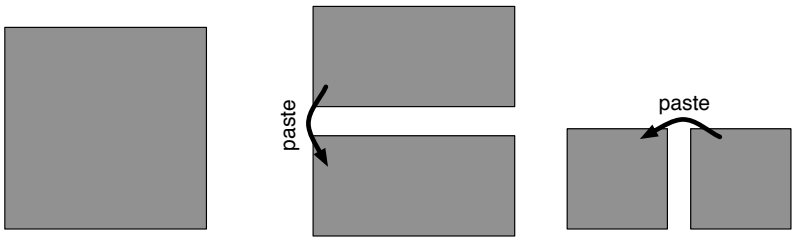


Figure 13: The altered procedure for the stretch, cut, and paste method of kneading to avoid the problems of extra expansion in all directions: stretch, cut and paste, and cut and paste.

provides tools to qualitatively examine functions that map an interval to itself, which is analogous to culinary kneading. That is to say, just as dough is stretched and folded (or cut and pasted) onto itself and its original surface area, the functions studied by kneading theory map a closed interval onto itself. Formally, kneading theory is a calculus used to describe the "qualitative behavior of the successive iterates of a piecewise monotone mapping." Somewhat more simply put, kneading theory is a set of tools that allows us to describe the qualitative details from sequential results from a mapping on a closed interval, where that mapping is strictly increasing or strictly decreasing on subintervals of the initial interval[3].

Kneading theory works by creating several different invariants that make use of characteristics of a given piecewise monotone function (henceforth called f). The *cutting* and *lap invariants* are variations on power series. Finding the limit of the cutting invariant provides information about the topological entropy of f. The more-interesting and important invariant is the *kneading invariant*, which in turn relies on a *kneading matrix*. The matrix contains information about the combinatorial and topological invariants of f.

Kneading theory and symbolic dynamics can be combined to explain the details of bifurcation diagrams for unimodal maps.[6] First, we define a partition of the interval on which the map is defined. If we use the quadratic map ($x_n = \alpha x_{n-1}(1 - x_{n-1})$, $0 \leq \alpha \leq 4$), we set up the partition as: $I_0 = [0, 0.5)$, $C = 0.5$, and $I_1 = (0.5, 1]$, then define s_n as follows:

$$s_n = \begin{cases} 0 & \text{if } x_n \in I_0 \\ C & \text{if } x_n = C \\ 1 & \text{if } x_n = I_1 \end{cases}$$

We define the *symbol sequence* (or *itinerary*) of a given starting point x_0 as $S(x_0) = \{s_0, s_1, s_2, \dots\}$. We also define an ordering ($\prec$) on symbol sequences that preserves their ordering on the interval, so if $S(x) \prec S(y)$ then $x < y$, and define the order of the symbols as $0 < C < 1$. We can call a symbol sequence s valid if for some point a on the interval $S(a) = s$. The *kneading sequence* $K(f)$ of a unimodal mapping f is the itinerary of $f(C)$, so that $K(f) = S(f(C))$.

We can obtain two theorems from this setup. First, given a symbol sequence s, if $K(f)$ is not periodic, and the sequence obtained by shifting the elements of s to the left (and discarding the first element) $\prec K(f)$ for any number of shifts larger than zero, then s is valid itinerary for f. Second, given two sequences s and t and that t represents a valid itinerary, then if we can shift t left some number of times m, where $m \geq 0$, so that it will always hold that $s \prec t$, no matter how many times s is shifted, then s is a valid sequence. These theorems show us that the kneading sequence of a mapping indicates which period orbits exist for that mapping. In the case of the quadratic mapping, the kneading sequence increases on the ordering $\prec$ as α increases. Kneading theory can be combined with other properties of that mapping to obtain a more detailed description of how periodic orbits emerge as α increases.[6]

This explanation (and the example with symbolic dynamics of unimodal maps) have been rather coarse. More detail on kneading theory and how it is used can be found in the original paper by Milnor and Thurston ([3]), an article by one of their colleagues on Scholarpedia ([1]), and this helpful set of notes from a talk given at the University of Minnesota ([6]).

Future Work and Conclusions

My original experiment plan was to map the diffusion of dye and trajectory of beads through bread dough. Bakers knead dough to form chains of gluten within the dough mass. Kneading the dough orients the gluten chains in orderly arrays and encourages weak, side-by-side bonds to form. Both of these things contribute to gluten strength, and strong gluten helps bread to rise.[2] Underknead the dough, and the chains will be underdeveloped. Conversely, overwork the dough, and the chains will begin to break down again. Bread dough's gluten chains most likely have some impact on the dynamics as dye and beads travel through the dough; to eliminate that state variable, I chose to use homemade play dough[4]. The artificial dough's dynamics change as the dough heats up, but this can be controlled by pre-chilling the dough in the refrigerator, so that the experiment can be completed before the dough heats up.

It might be interesting, though probably not necessary and certainly difficult, to repeat the experiments with bread dough. Strogatz, Peitgen et al., and others make their assertions about the kneading processes they describe, not about the specific process of kneading bread dough. Strogatz actually specifically mentions a type of pastry dough, but his interest there is the layers of dough that are maintained, not in the other dynamics of that dough. Because that dough is particularly difficult to create, due to its delicate nature, it seems unlikely that kneading that dough would create the "culinary fractal attractor" Strogatz describes, but it would take a more talented baker than I to test that particular theory. Suffice it to say that, since most authors are really more concerned with the generic process of kneading non-specific dough, the experiments I have performed seem sufficient to test their claims. The process of kneading dough definitely demonstrates sensitive dependence on initial conditions , and also seems to demonstrate other behavior characteristic of chaotic systems.

References

[1] Toby Hall. Kneading theory. `http://www.scholarpedia.org/article/Kneading_theory`, March 2011.

[4] Recipe for homemade play dough is 1 part salt, 1 part water, and 2 parts flour.

[2] Harold McGee. *On Food and Cooking: The Science and Lore of the Kitchen*. Scribner, New York, NY, revised edition, 2004.

[3] John Milnor and William Thurston. On iterated maps of the interval. In James Alexander, editor, *Dynamical Systems*, volume 1342 of *Lecture Notes in Mathematics*, pages 465–563. Springer Berlin / Heidelberg, 1988.

[4] Heinz-Otto Peitgen, Harmut Jürgens, and Dietmar Saupe. *Chaos and Fractals: New Frontiers of Science*. Spring-Verlag New York, Inc, New York, NY, 1992.

[5] Steven H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry, and Engineering*. Westview Press, Cambridge, MA, 1994.

[6] Warren Weckesser. Symbolic dynamics in mathematics, physics, and engineering: Notes on a talk presented by dr. nicholas tufillaro. `http://www.ima.umn.edu/~weck/nbt/nbt.html`, April 2011.

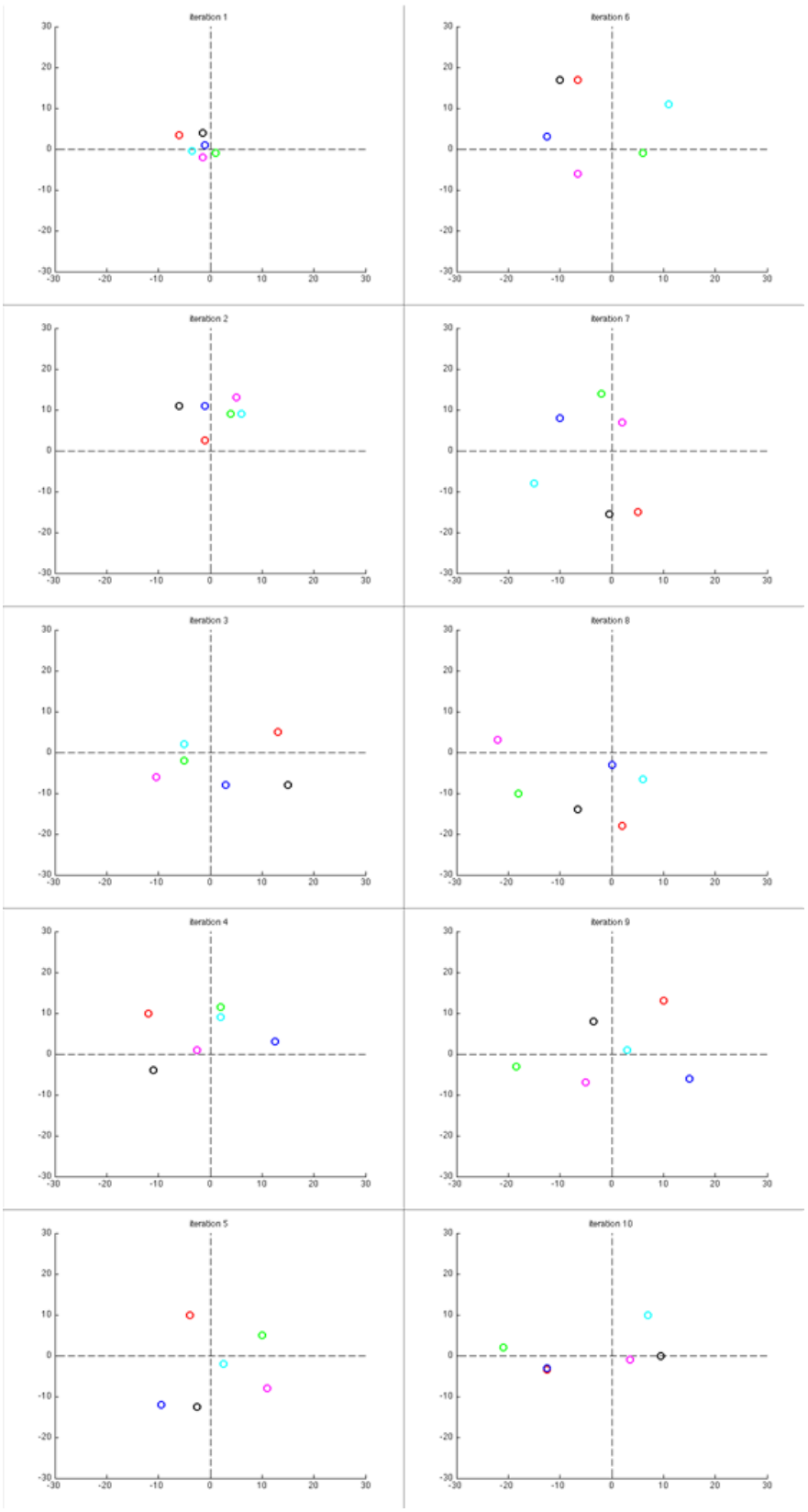


Figure 14: Beads progressing through a dough mass with the stretch and fold method, iterations 1-5 in the first column, 6-10 in the second.

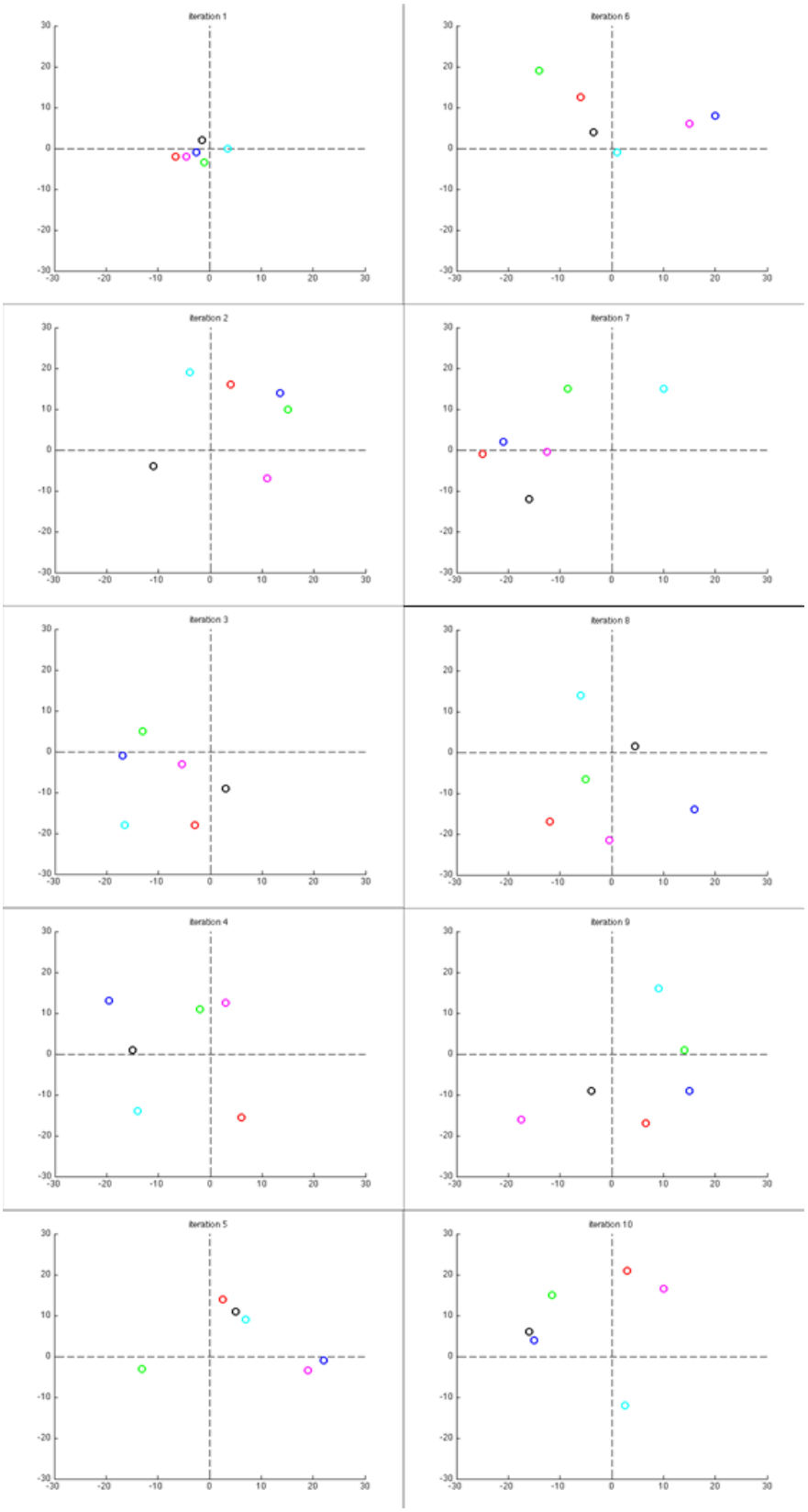


Figure 15: Beads progressing through a dough mass with the stretch, cut, and paste method, iterations 1-5 in the first column, 6-10 in the second.

Wiring Chaos:
The challenges of modeling chaos in a circuit

Bruce Deakyne
Chaotic Dynamics, Spring 2011
Professor Elizabeth Bradley
University of Colorado Boulder
deakyne@colorado.edu

Abstract

Before the advent of modern computers, mathematicians used analog circuits to model non-integrable differential equations. This paper investigates the feasibility, functionality, and accuracy of several systems including the solar system, three-body system, and Lorenz attractor. Due to their highly nonlinear nature and circuit limitations, the solar system and three-body systems are not practical to build, with component costs as high as $50,000 and $4,300 respectively. However, the Lorenz attractor can be built and analyzed; even using "economy" components, its output covers the correct attractor. An error analysis concludes that while the Lorenz circuit's output may have the correct shape, high accuracy components are required for the system to follow the correct trajectory for even just a short period of time. Given these challenges, only a few practical applications of chaotic circuits exist, mainly those that leverage error such as random number generators. Unfortunately, in most other functions, digital computing drastically outperforms analog circuits.

1 Introduction

Mathematicians have studied differential equations for centuries. However, only recently has progress been made in solving systems of non-integrable equations. In the past few decades, computers have allowed researchers to implement discrete approaches, such as the 4^{th}-order Rungé-Kutta solver. Before the age of computers however, researchers discovered they could build controllable physical systems with the same differential equations as those they were studying. One such method is the use of electronic circuits; utilizing integrator designs and feedback, nearly any first order differential equation can theoretically be simulated.

This paper explores the validity and accuracy of those circuits. The initial thought was to model the solar system using these techniques; however, for reasons discussed in section 2, a more linear system called the Lorenz attractor is the topic of much of the analysis. The purpose of this paper is to determine if

there any benefit to using electronic circuits; is the digital environment superior in every way, or are there still applications for the analog domain?

2 Initial Project & Design Challenges

The original conception for this paper was to model the solar system using a nonlinear analog circuit. Through the many challenges encountered in the process, a new project emerged whose purpose was to describe the difficulties, error, and validity of a circuit model.

2.1 N-body gravitation

To understand the overall design of the solar system model, the interaction between two bodies, described by Newton's law of universal gravitation, should be examined.

$$\vec{F}_{12} = \frac{Gm_1m_2(\overrightarrow{r_2} - \overrightarrow{r_1})}{|\vec{r}|^3} \qquad \text{(Equation 1)}$$

The two-body problem, derived from the forces in Equation 1 is fairly straightforward. There are few enough state variables that chaos is not possible; the bodies orbit each other or one crashes into the other. A more interesting problem is that of the interaction of three bodies; the motion of just one of the bodies is as follows (1):

$$\ddot{\vec{r}}_1 = -G\left(m_2\frac{\overrightarrow{r_1} - \overrightarrow{r_2}}{|\overrightarrow{r_1} - \overrightarrow{r_2}|^3} + m_3\frac{\overrightarrow{r_1} - \overrightarrow{r_3}}{|\overrightarrow{r_1} - \overrightarrow{r_3}|^3}\right) \qquad \text{(Equation 2)}$$

2.2 Calculation of magnitude

Notice that the positions in Equation 2 are actually vectors. Approximating space as a two dimensional plane yields two coordinates for each body's position for a total of 6 overall in the 3-body problem. The difficult portion of Equation 2 is the magnitude cubed in the denominator; the first of the two terms in Equation 2 can be found expanded in Equation 3. This equation can also be described in block diagram form, as seen in Figure 1.

$$\frac{1}{((x_2 - x_1)^2 + (y_2 - y_1)^2)^{3/2}} \qquad \text{(Equation 3)}$$

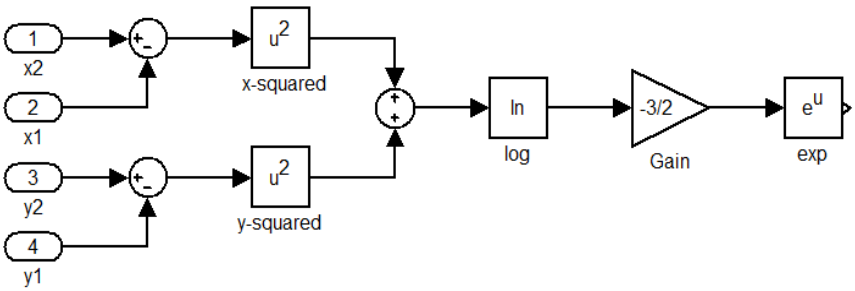


Figure 1: Computing magnitude in block diagram form

2.3 Computing magnitude in circuits

Perhaps the most significant challenge in designing circuits to model differential equations is the difficulty in relatively simple, nonlinear mathematical operations such as multiplication, division, and logarithms. These operations cannot be performed with passive components; they require premade integrated circuits (ICs). Consider the block diagram in Figure 1. Each of the four nonlinear operations requires a separate IC; these components range in cost from $30 to $210 each (2). The magnitude sub-circuit, depending on the component accuracy, would cost between $160 and $600 each. In order to implement the three-body problem with a circuit, three magnitude sub-circuits and an additional 12 multipliers are required, a total cost in the range $840 and $4,300. Expanding this to the nine bodies of the solar system requires 36 magnitude sub-circuits and 144 multipliers; even using the lowest quality components, this circuit would cost over $10,000.

2.4 Challenges of chaos in a circuit

Sections 2.1 through 2.3 detail the size and expense challenges of chaos in a circuit. Though the solar system and three-body problem were both a bit too expensive to explore, there are many more chaotic systems with fewer nonlinear terms that are much easier to model with circuits. One such system is the Lorenz attractor, the topic of section 3.

3 The Lorenz Attractor

In 1960, Edward Lorenz sought to model the convection rolls in the atmosphere in order to understand and predict weather. His solution is comprised of three first order non-linear differential equations with three free parameters *a*, *r*, and *b*. The equations are as follows:

$$\dot{x} = a(y - x)$$

$$\dot{y} = rx - xz - y$$

$$\dot{z} = xy - bz$$

Under the correct conditions, the system becomes chaotic; a small perturbation to the initial conditions yields a significant difference in the long-term results.

3.1 Design of the Lorenz Circuit

Using the Lorenz equations, the first step in the circuit design is a simple block diagram, shown in Figure 2. The variables *x*, *y*, and *z* are fed through summing nodes, multipliers, and gains to form the first order differential equations; they are then integrated to provide the output, which is fed back to the beginning.

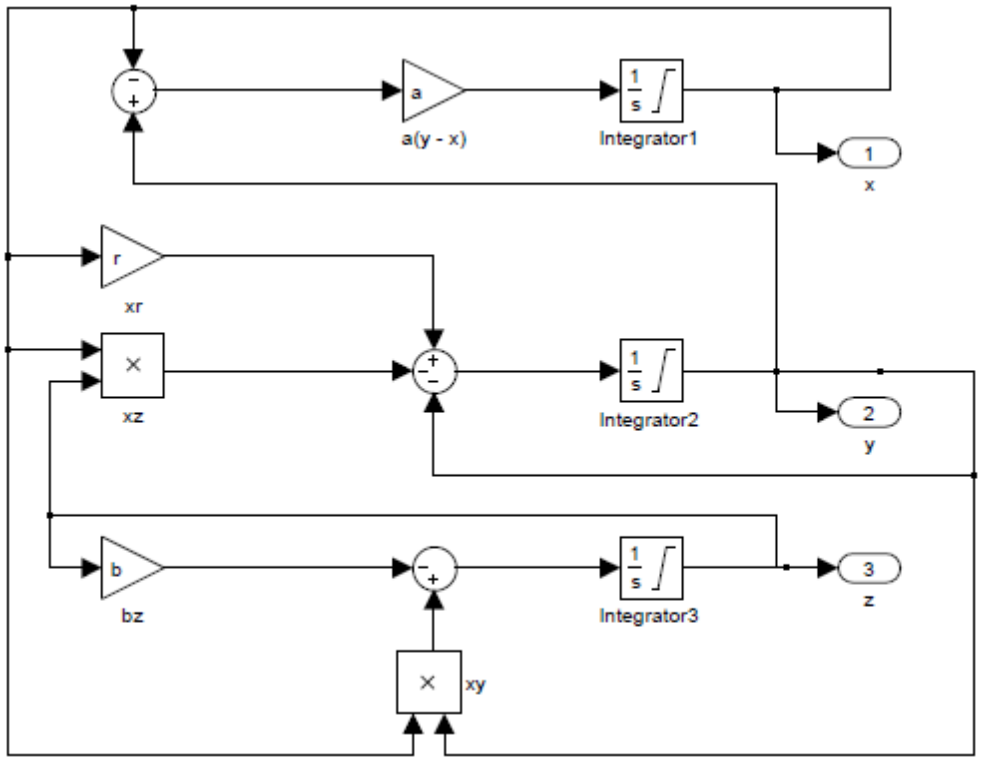


Figure 2: Block diagram for the Lorenz attractor circuit

With the general circuit flow, the exact circuit diagram can be laid out using a design and simulation program such as Orcad. Integration is fairly straightforward; an op-amp with capacitive feedback will integrate the sum of the input voltages. It is important to note that this does invert (change the sign) of the signal and that the multiplier chips also have the option of inverting the signal depending on which input is tied to ground. For the ideal design, it is desired that the output of the first integrator be $-x$ instead of x, which avoids using an additional inverter. The circuit design is shown in Figure 3.

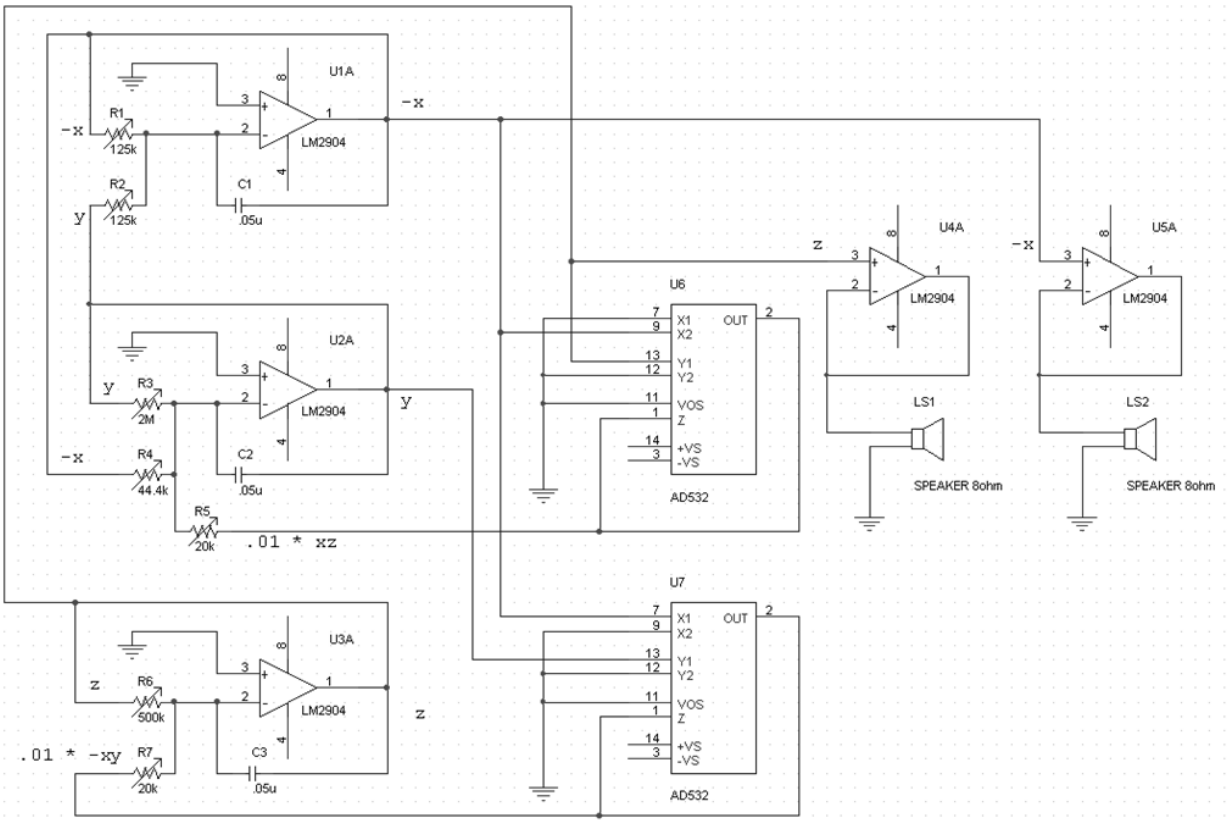


Figure 3: Schematic for the Lorenz Attractor Circuit

The three sub-circuits on the left are the integrators for *–x*, *y*, and *z*. The two ICs in the middle are the multiplier chips, and the two sub-circuits on the right are the speakers, attached to *–x* and *z* variables. The capacitance of the integrators affects the frequency the circuit oscillates at (3); as capacitance increases, the natural frequency decreases. Though a few different values were tried experimentally, $C_{1,2,3}$ = .05μF was the most practical in terms of oscillation frequency and components available. The resistor values are chosen based off of the parameters a = 16, r = 45, and b = 4 as follows:

$$R_1 = R_2 = \frac{1}{10}\frac{1}{aC_1} = 125k\Omega \qquad R_5 = \frac{1}{1000}\frac{1}{C_2} = 20k\Omega$$

$$R_3 = \frac{1}{10}\frac{1}{C_2} = 2M\Omega \qquad R_6 = \frac{1}{10}\frac{1}{bC_3} = 500\text{k}\Omega$$

$$R_4 = \frac{1}{10}\frac{1}{rC_2} = 44.4k\Omega \qquad R_7 = \frac{1}{1000}\frac{1}{C_3} = 20k\Omega$$

Note that the terms have a $1/10$ coefficient, which comes from the normalization of voltages to 0.1v. The voltage supplied to each integrated circuit is only ±15v, which is therefore the maximum value for any variable. Since the Lorenz attractor usually lies in values above 15 in magnitude, the voltage measured will be $1/10$ of the actual value of the variable in the Lorenz attractor. The $1/1000$ terms, which appear

on the resistance values of the nonlinear terms, are a combination of the $1/10^2$ from the two normalized variables multiplied together as well as the multiplier circuit's output specification, which is the product of the two input voltages divided by 10.

Another notable feature of the circuit is that the resistors are actually potentiometers; their resistance was adjustable. In section 3.3, the implications of error are discussed. As normal resistors carry a ±5% error, the potentiometers allowed an adjustment to bring the actual value to less than $\pm 10^{-2}$% error.

3.2 Results and Analysis

With a circuit build, all that is left is to measure is the output. Theoretically, turning the circuit on would yield initial conditions of $(x_0, y_0, z_0) = (0, 0, 0)$, which is an unstable fixed point. However, due to error discussed in section 3.3, the system ends up on a trajectory diverging from the fixed point and eventually finds the attractor. There are three levels of questions that describe how well the circuit models the Lorenz equations. First, is the output chaotic? Second, does the chaos resemble the Lorenz attractor with the same parameters? Finally, given exact initial conditions, can the circuit give an accurate long-term solution? Unless otherwise stated, all plots are of $-x/10$ on the horizontal axis and $z/10$ on the vertical axis.

The first question is easily answered based on Figure 4, the oscilloscope output of the circuit. The graph clearly resembles the Lorenz attraction, but by how much? This moves the focus to the second question, how accurately the circuit's attractor's shape matches that of the theory. In order to calculate this, a simulation utilizing the 4th-order Rungé-Kutta solver with the same system and similar initial conditions is implemented using MATLAB. This graph (thin black line) is then scaled to size of the oscilloscope output (gray background) and overlaid in Figure 5 to show how well the two matched. The two cover roughly the same attractor; the main difference spawns from the incorrect initial condition in the MATLAB simulation, which causes an overshoot in the upper-left-hand side of the graph and fills in the right hole more than the oscilloscope's output.

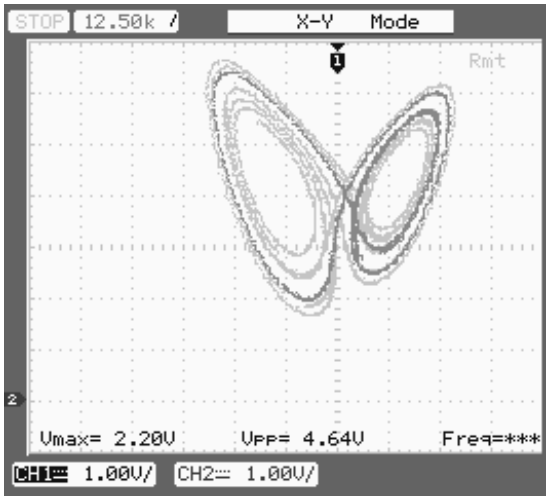


Figure 4: Oscilloscope output, *z* vs. *–x*

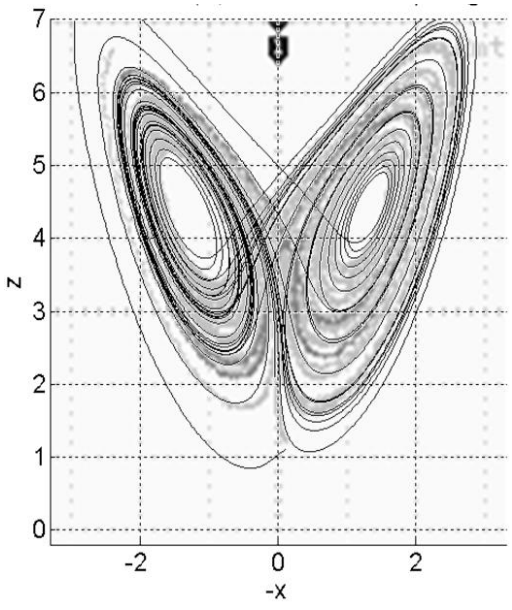


Figure 5: Oscilloscope output with RK4 overlay

3.3 Sources of error in a chaotic circuit

The output of the circuit is chaotic and matches the general shape of the Lorenz attractor. However given an exact initial condition, how accurately will the circuit traverse the true trajectory? This is rather difficult to do with the circuit itself, but can be done using a MATLAB model. First, the sources of error in a circuit must be considered. Three main sources of error exist in chaotic circuits: passive component tolerance, integrated circuit accuracy, and electromagnetic effects.

All passive circuit components, such as resistors and capacitors, have a set tolerance in a range of around 0.1% to 10%. The components used to build the Lorenz attractor in section 3.2 were accurate to ±5% of their stated value. For this reason, variable resistors were used and adjusted within approximately $\pm 10^{-2}$% of the required resistances. However, variable capacitors were not available at the time of the circuit's construction, yielding a ±5% capacitance error. One method for further fine tuning the parameters, though not used, would be to calculate the decay time of an RC circuit. Take, for example, the calculation for R_1 in section 3.1; a cross-multiplication yields the value of the equation parameter *a*.

$$a = \frac{1}{10}\frac{1}{R_1 C_1}$$

The inverse of product of resistance and capacitance is simply a time constant. By supplying the series variable resistor and capacitor with a low frequency square wave and measuring the voltage output between the two components on an oscilloscope, the rise time and RC time constant can be calculated. The resistance can be adjusted until the time constant is proportionally close to the parameter *a* within a satisfactory error range. Using this method, the error can be reduced to the limit of the oscilloscope's accuracy, potentiometer's sensitivity, or operator's patience.

The error in integrated circuits is slightly more difficult to deal with. Four types of error contribute to the inaccuracy of a multiplier, including input and output offset as well as a scale factor (2)**.** This error can be mitigated, but at great cost. The multiplier used in the Lorenz circuit of section 3.1 is Analog Devices AD532J; the accuracy of this IC is at worst ±2% and is typically around ±1.5% (4). As of 1973, Analog Devices produced a high accuracy multiplier whose error was guaranteed less than ±0.2%. However, the price of accuracy is quite steep; the "economy" AD532J multiplier costs around $30 per chip while a single high accuracy AD427J multiplier costs seven times that: $210 (2).

The final type of error mentioned is difficult to account for, especially in bread-boarded circuits. The electric and magnetic fields established in a circuit allow it to work properly and predictably. However, they can also work in unexpected ways. For example, the capacitance between two typical wires 4cm long and 1cm apart is on the order of .05pF. Even if all other error were absent, this still amounts to approximately an additional 10^{-4}% to the capacitors attached to the integrators. The physics of transistors must also be taken into account in integrated circuits. The current across the transistor is temperature dependent; therefore, as the circuit runs and dissipates heat, its behavior may change slightly. Most integrate circuit's datasheets discuss the output variations based on temperature. For example, the

multipliers used in the Lorenz circuit in section 3.2 lists the maximum temperature error as ±0.04% per °C (4).

3.4 Effects of error in a chaotic circuit

To better understand the effects of the sources of error presented in section 3.3, three MATLAB simulations compare the unaltered Lorenz equation to the results when the parameters *a*, *r*, and *b* have been given a certain level of inaccuracy. The output plot details the magnitude of the difference between the points in the two systems at a given time. The figures shown will demonstrate the difference in outcomes of the theoretical and physical systems if the initial conditions are completely accurate.

The first error tested is the easiest to achieve in the circuit, 2%; this could come from only slightly corrected resistors and capacitors or the "economy" multiplier IC described in section 3.3. This error is still an understatement of the Lorenz circuit built in section 3.2, as it assumes capacitive accuracy is smaller than that of those used in the circuit. Figure 6 clearly illustrates how the new system produces slightly different results from the very beginning. This disparity grows as time progresses, and by around 4 seconds in, the two systems are on different trajectories.

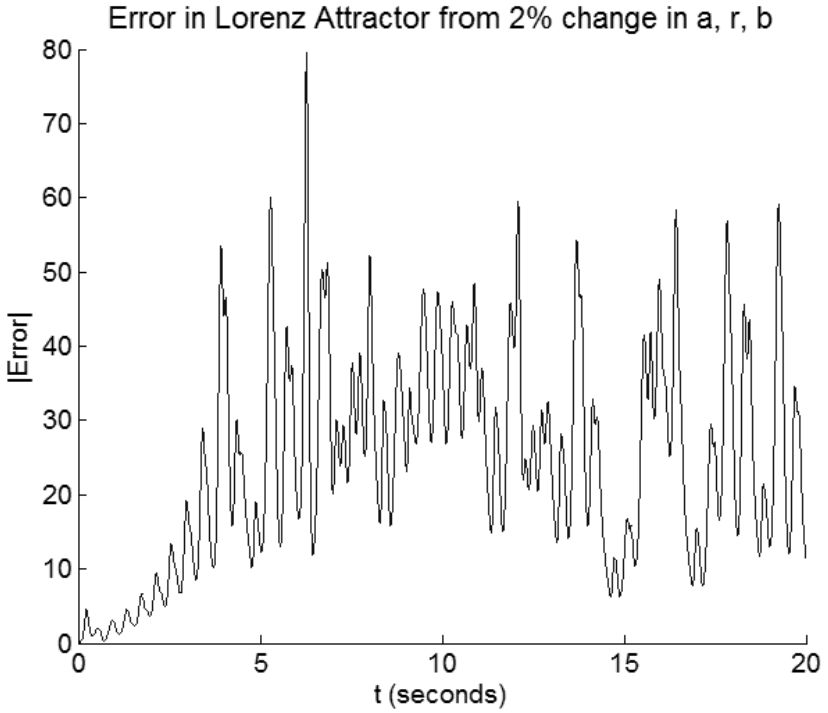


Figure 6: Resulting error from a 2.0% perturbation in the Lorenz attractor parameters

The next error examined, 0.2%, would result from more finely tuned resistor and capacitor values, as well as much more expensive, high accuracy multipliers. Unlike the previous simulation, Figure 7 demonstrates how the two systems are very closely in synchronization for the first few seconds. After around two seconds, the error grows to noticeably above zero and the trajectories quickly diverge.

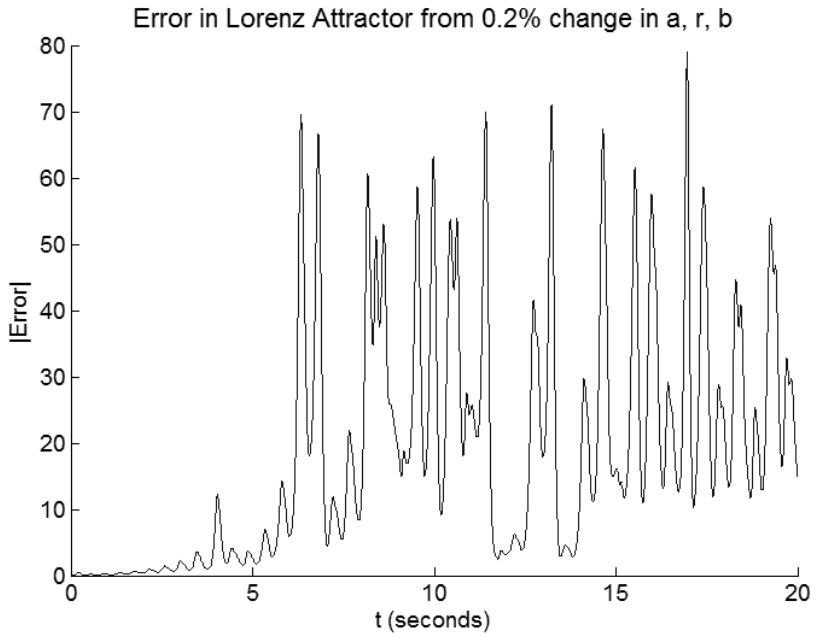


Figure 7: Resulting error from a 0.2% perturbation in the Lorenz attractor parameters

The final error examined is 10^{-4}%; this represents only small electromagnetic effects of capacitance between wires and assumes that by some miracle, the passive circuit components and integrated circuits have been adjusted to perfection in a temperature controlled environment. Even with this, Figure 8 shows the systems still diverge after around 8 seconds. However, it would be possible to follow the exact behavior for at least a small time period.

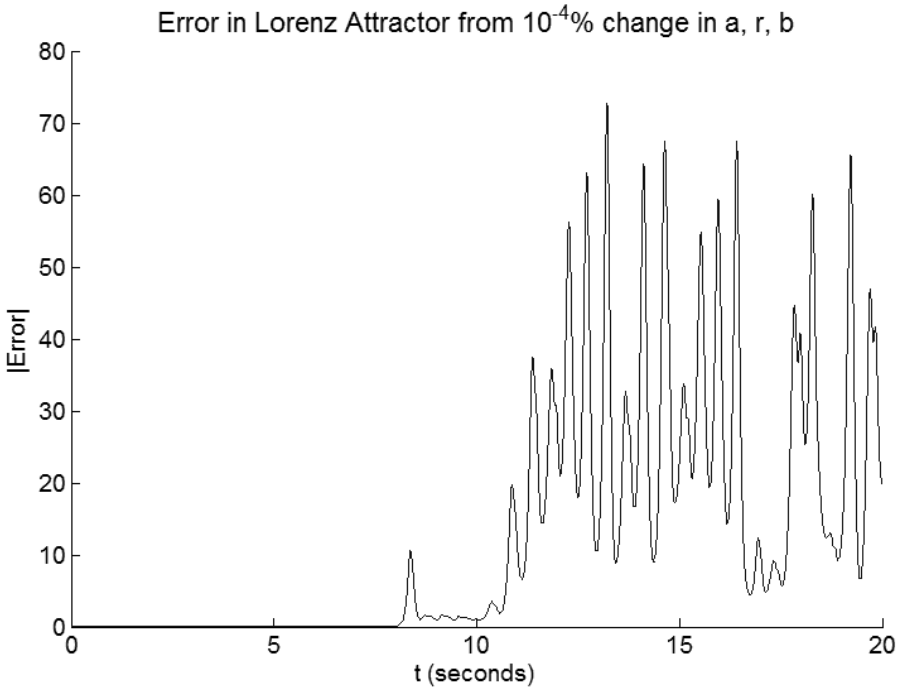


Figure 8: Resulting error from a 10^{-4}% perturbation in the Lorenz attractor parameters

In error around and below 0.2%, it seems very likely that given accurate initial conditions, the system would maintain the correct trajectory for a short period of time.

4 Conclusion

Non-linear circuits have been used for decades to model systems of equations; however, as discussed in this paper, the accuracy of such systems is limited.

4.1 Implications of error

Sections 3.3 and 3.4 explain the sources of error in a circuit model and the resulting differences in the theoretical versus physical systems. In order to achieve an accurate model, even for a brief second, high precision multipliers are necessary; the two multipliers ICs for the Lorenz circuit would cost over $400 in total. This is likely just as expensive as a microprocessor with 4^{th}-order Rungé-Kutta solver implementation embedded. An application such as the magnitude for a 3-body problem could cost upwards of $600 each. This implies a high accuracy three-body circuit could total over $4,000, certainly more expensive than modern desktop computer with high computing capacity. Additionally, a 0.2% error does not begin to compare to machine epsilon limitations, which are on the order of 10^{-16}% of the common Lorenz variable values.

4.2 Potential applications

Though there are many difficulties in implementing chaotic systems with circuits, these can be leveraged and utilized for certain applications. Consider a random number generator; often, it is simply an algorithm that provides a difficult to predict yet uniform distribution of values. For that reason, many random number generators are often referred to as pseudo-random. If a chaotic system with uniform distribution could be modeled in a high frequency circuit, it could quite effectively generate a random number without the possibility of prediction. All the challenges of reducing error, discussed in sections 3.3 and 3.4, would not be necessary. In fact, the error would magnify the random behavior, as sensitive initial conditions mean a slight change to the system would result in dramatic overall outcome differences. One such random number generator implementation is the logistic map in circuit form, driven by a discrete clock cycle. This requires a single multiplier and has relatively uniform distribution (5)**.**

One might ask: would a chaotic system qualify as truly random, or is it pseudo-random? The Oxford English Dictionary defines random as "not sent or guided in a particular direction; … without method or conscious choice" (6). Though a perfect model chaotic system is unpredictable, it is not necessarily random. However, with the introduction of physical error to a system, especially that of transistor behavior at the electron level, the system is no longer guided; though a method would be in place to ensure fairly uniform distribution, the error could nullify the conscious choice portion of the definition as well. Using this method, a true random number generator could be placed in the processor hardware itself.

One final benefit of analog circuits is their ability to oscillate at high frequencies. Though the Lorenz circuit from section 3 has a relatively low frequency, it can be increased by reducing the value of the capacitors in the integrators (3). A MATLAB simulation with a small time-step can take longer to compute the trajectory than the value of the total trajectory time. Though it might be less accurate than a computer simulation, the analog circuit has the speed advantage.

4.3 Final thoughts

Analog circuits were a popular method for studying chaos before the advent of modern computing. However, as seen in this paper, they have many drawbacks. In section 2, the n-body problem was explored and deemed infeasible due to the complexities in designing a circuit to compute a relatively simple vector magnitude. Section 3 introduced the Lorenz circuit and sources of error in it. Section 3.4 determined that to get on the correct trajectory, even for a brief second, an error less than approximately 0.2% is required. In order to meet this specification, the Lorenz attractor circuit must use high accuracy integrated circuits.

Tying this back into the gravitation problems from section 2 would imply the cost of the three-body circuit at over $4,000 and the solar system at over $50,000. Note that similar accuracy can be easily and quickly achieved with a low-end processor on the range of $500. The only feasible applications for chaotic circuits seem to be those such as random number generators, which can actually leverage the error and uncertainty in their functionality. Despite these few purposes, it seems modern computers have overtaken their analog predecessors in accurate modeling of chaos.

5 References

1. Bradley, Elizabeth. *Classical Mechanics: Notes for CSCI4446/6446.* Boulder, CO : s.n., 1999.
2. Analog Devices, Inc. *Nonlinear Circuits Handbook.* Norwood : s.n., 1974.
3. —. AD532. *Chip Catalog.* [Online] 2001. [Cited: April 15, 2011.] http://www.chipcatalog.com/Analog/AD532.htm.
4. random, n., adv., and adj. *OED Online.* [Online] Oxford University Press, March 2011. [Cited: April 26, 2011.] http://www.oed.com/view/Entry/157984?rskey=QWAKo5&result=1&isAdvanced=false.
5. Chen, Wai-Kai. Feedback, Nonlinear, and Distributed Circuits. Chigaco : CRC Press, 2009.
6. Horowitz, Paul. Lorenz Attractor. *Horowitz Group.* [Online] 3 2, 2003. [Cited: April 20, 2011.] http://frank.harvard.edu/~paulh/misc/lorenz.htm.

iMath: ”Impressionistic Math
A Quantitative Approach to a Qualitative Problem

Frank Jones
francis.jones@colorado.edu

May 19, 2011

1 Abstract

Fractal art is found in numerous domains, from video games to movies to impressionistic art galleries. Typically fractals impact art in three ways: as the algorithmic basis for efficiently mimicking naturally occurring fractals such as trees, clouds and mountains, as a component in an artist's overall composition, or (on the merits of the fractals inherent beauty) as a stand-alone image. This work presents a variation on the second and third themes. Instead of seeking inspiration from, or randomly searching for impressionistic fractals by hand, I propose *iMath* (Impressionistic Math) a framework for generating fractal images that are "inspired" by input imagery.

2 Introduction

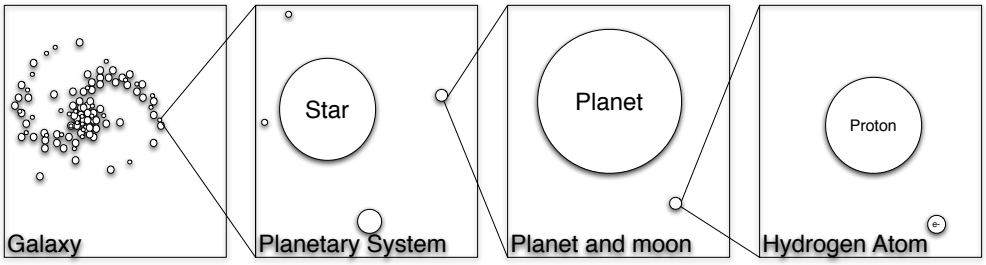
Fractals are everywhere! Since their popularization by Benoit Mandelbrot and others in the late 1970s and 1980s the value of fractals to both art and science has become increasingly apparent. Far more than an intriguing occurrence of the natural world, the mathematical and aesthetic properties of fractals have been successfully applied to problems as quantitative as image compression [7] and as qualitative as impressionistic art [8, 9, 10]. It is the aesthetic and qualitative properties of fractals that have inspired this project.

This paper presents the basis for iMath, a tool for generating impressionistic fractal artwork using existing images as a guide. The novelty of this work is that it represents an attempt to automate a process that appears to be strictly human. Understanding the concepts presented here requires at least a cursory knowledge of fractals and their application/utilization in artistic domains. The following sections present a high-level introduction to such information. Following this introductory material the iMath application is discussed and some preliminary results are presented. Finally I present my observations and conclusions drawn from the project in its current state, including proposals for possible future work.

2.1 Fractals

A fractal is a pattern or shape that exhibits self-similarity within its structure. Numerous examples of fractals exist in nature. Clouds, mountain ranges, streams, and even the universe exhibit the characteristics of a fractal. As an example, consider a galaxy (see figure 1.

Figure 1: The fractal nature of the universe



From a distance an observer would perceive numerous masses orbiting around a concentrated central region. As the scene is magnified it becomes apparent that the individual masses are in fact planetary systems; many with multiple planets orbiting a concentrated region of mass called a star. Further magnification reveals that it is common for such planets to be orbited by one or more moons. Continued magnification of these moons will eventually reveal the structure of the atom - clouds of electrons orbiting concentrated regions of mass... and so on. Mathematically fractals can be constructed that exhibit such self-similarity infinitely. That is - a person can zoom in on the fractal's structure indefinitely with no loss of complexity.

A fractal's structure arises from the fact that it does not occupy the entirety of the region it is bounded by. In other words, the structure of a fractal is defined as much by the regions that it does NOT occupy as the regions that it does. This gives rise to one of the signature characteristics of fractals; they do not occupy integer dimensions. Rather than being 1,2, or 3 dimensional objects fractals may be .5, 1.3 or 2.666 dimensional etc.. As will be explained below, iMath utilizes dimensionality as an approximation of complexity.

Numerous well known techniques exist for approximating the dimensionality of an object. iMath employs a technique for calculating what is known as the box dimension. Fractal dimension, as well as several techniques for approximating it, is explained in [11]; only a cursory introduction to box dimension is provided here.

2.1.1 Box Dimension

To calculate box dimension the object of interest is first circumscribed within a hyper-cube of the required dimensionality. In the context of imagery, the hyper-cube is simply a bounding square around the image or region of interest. In suc-

cessive steps, the box is subdivided into smaller and smaller boxes, and at each step the number of boxes that contain part of the image are counted. The box dimension is defined as a ratio between the size of the boxes (ε) and the number of such boxes occupied by the image as the size of ε approaches 0.

$$d = lim\frac{lnN(\varepsilon)}{ln(\frac{1}{\varepsilon})}, \quad as \quad \varepsilon \to 0 \tag{1}$$

Obviously it is impractical in most cases (and impossible in the case of a digitized image) to meaningfully subdivide the object of interest beyond some finite point. Thus except in cases where the box dimension has a close formed analytical solution an approximation is obtained by plotting $ln(\frac{1}{\varepsilon})$ vs. $lnN(\varepsilon)$ and using linear regression to determine the slope of the transient in the curve[1]. iMath utilizes this approximation to calculate the box dimension of individual regions of the input image. The regions are defined by partitioning the input image into an N X M grid of equally sized squares. Further details concerning this process are provided in section 3.2.1.

2.1.2 The Mandelbrot Set

One of the most famous fractals is the well known Mandelbrot set, which was discovered by Benoit Mandelbrot while studying the behavior of the iterative equation:

$$z = z^2 + c \tag{2}$$

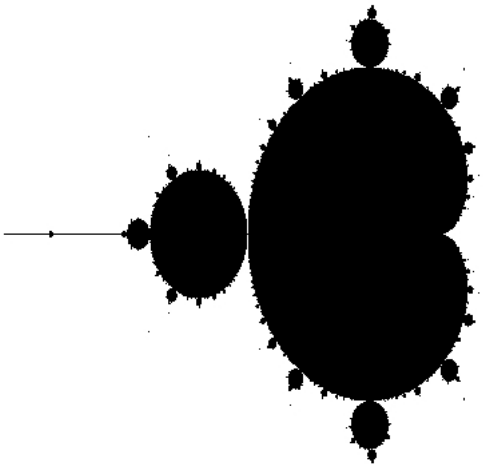
where z begins at zero and c is a complex value.

The Mandelbrot set is defined as the set of points for which this equation does NOT diverge as the number of iterations approaches infinity. Rather than formally proving that a point doesn't diverge, artists and scientists generally approximate such a proof by simply iterating the equation up to some predetermined number of iterations at which point if the value of z has not diverged then the point is classified as being part of the set, otherwise it is not. The Mandelbrot equation can be visualized by representing the real and imaginary components of the values of c in the complex plane as the X and Y coordinates in a two-dimensional image.

[1] iMath assumes that all values of ε sampled are in the transient, though this is not strictly correct. Fortunately, since the value is used in a relative sense (i.e. compared to other values computed in the same way) the desired effect is still achieved)

By coloring all of the points in the set black and those outside the set white, an image similar to figure 2 is generated.

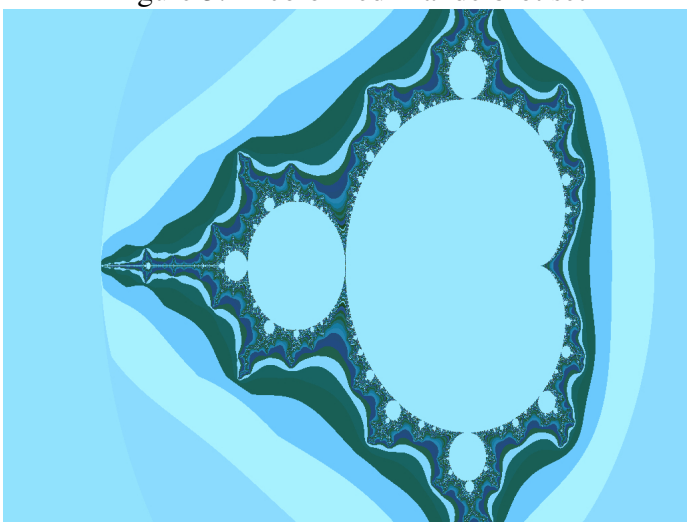
Figure 2: The Mandelbrot set



Though complex, the image displayed in figure 2 is not particularly impressive. Further detail (and beauty) can be added to this plot by implementing a mapping between the behavior of the points outside of the Mandelbrot set with values in a color map. A common approach is to establish a correlation between the colors and the rate at which the points diverges from 0. The rate of divergence is commonly approximated by simply marking the iteration at which the value of z exceeds a certain threshold. A low count would suggest rapid divergence and a high count slower divergence. Using this information, more-intricate plots can be created by assigning different colors to the pixels in accordance with their divergence rates. Figure 3 demonstrates the result of this technique.

Countless coloring formulas have been proposed. Some attempt to smooth transitions between colors (see [12]) while others attempt to introduce 3-dimensional elements to the fractal, as in [6]. iMath employs a coloring formula driven by the input image itself. This technique is introduced in section 3.2.2.

Figure 3: A colorized Mandelbrot set



2.2 Fractal Art

The beauty of fractals has not been lost on the world of artists. One need only "google" the term "fractal art" to discover that in the few decades since their introduction, countless fractals have found their way into computer wallpapers, picture frames, and even art galleries, purely on the merit of their intrinsic mathematical beauty.

The complexity and diversity of fractals that can be generated from relatively simple mathematical equations such as the Mandelbrot set has resulted in an explosion of fractal generating computer applications such as Apophys, Ultra Fractal, XaoS and countless "home-grown" software systems. These programs simplify the process of generating fractal images by performing the math behind the scenes. The user need merely set and adjust a set of simplified parameters and the resulting fractal is generated.

2.2.1 Impressionistic Art

Beyond the realm of "pure fractals" the world of impressionistic artists has also learned to exploit the complexity and structure of fractals. Artists such as Renata Spiazzi, Terry Wright, and Rogert McGregor utilize fractals in various forms and stages of their artistic process. These artists may use fractals as a source of inspi-

ration or include them in a piece in such an obscured form that such that only the artist would recognize its presence or influence explicitly [8, 9, 10].

2.2.2 Computer graphics and animation

Beyond their artistic beauty, the fact that complex fractals can be generated from simple equations and initial conditions makes them an ideal method of storying and generating complex imagery - on the fly if necessary. Fractal landscape generation in particular has become a widely accepted method for generating beautiful, varied and realistic looking terrain for digital imagery, video-game graphics, and even full length movies. The role of fractals in generating such content is discussed in [1] and [5]

3 iMath

In preceding sections I presented methods in which fractals are incorporated into artwork, serving either as components of the art themselves, or as a source of inspiration for the final piece. In the previous case the generalized approach involves an iterative three step process in which:

1. A human enters parameters into a computer program.
2. The software generates a fractal based on the input parameters.
3. A human evaluates the fractal and adjusts the input parameters.

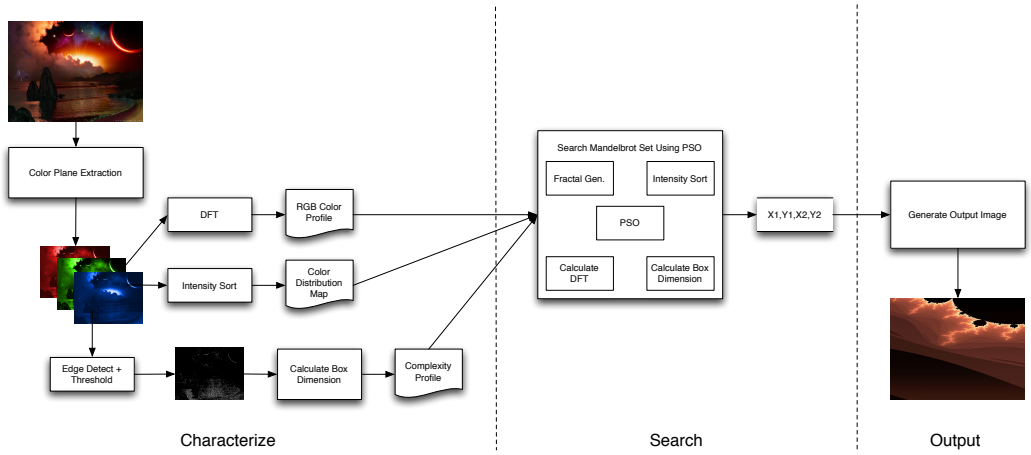
This process iterates until a noteworthy image is discovered. This fractal, or collection of fractals, is subsequently named and a work of art is officially born. Often such pieces have impressionistic qualities that are reflected in the names they bear.

Rather than simply generate fractals to be perused by humans, iMath attempts to *automatically* generate impressionistic art by creating fractal images influenced by the characteristics of a "seed" image provided by the user.

3.1 Architectural Model

Figure 4 presents the high-level architecture of iMath.

Figure 4: High level architecture of iMath



As can be seen in the figure, a three step process is employed.

1. Characterize the input image
2. Search for parameters that result in fractals that approximate the input image attributes
3. Output the fractal

The following sections discuss in greater detail how these steps are accomplished.

3.2 Conceptual Considerations

As the end goal of iMath is to produce *impressionistic* fractals, it is necessary to quantify characteristics of an image that contribute to the impressions left on a viewer. This work investigates the effectiveness of utilizing fractal dimension and color distribution for this purpose. The fact that techniques exist for quantifying these attributes adds to their appeal as starting points.

Aside from the sample images presented below, no attempt is made in this work to validate the impressionistic value of these metrics. It is probable that better characteristics and/or combinations of characteristics exist; however, performing an assay to determine such a feature set is beyond the current scope of this work. Rather, iMath represents a first attempt at establishing a foundation upon which (among other things) such a study might be constructed.

3.2.1 Complexity

One of the assumptions underlying this work is that complexity is an important factor in human processing of imagery. More specifically, the assumption utilized by iMath is that the difference in complexity between one region and another within an image conveys impressionistic information. For example a bowl of popcorn with its many edges and shadows produces a vastly different effect than that of a flat wall painted with the same component colors.

Edges play an important role in conveying both complexity and information in human vision. Often it is the edges of an image that convey the majority of the meaningful information. Consider for example the fact that numerous drawings, technical and otherwise, rely exclusively upon edges (lines) to convey the requisite information. iMath leverages the connection between visual information and edges to quantify the complexity of regions in an input image. This metric is computed using a two-stage process.

First a simple Gaussian edge detect filter is applied to the input image. The resulting image is then thresholded to produce a black and white "line drawing" of the original image (see figure 4). Next, the filtered image is divided into M X N square regions of equal size and the box dimension of each region is calculated. Region sizes experimented with in this work were typically 8, 16, or 32 pixels square [2]. The collection of dimensions for all of the regions constitutes a "complexity profile". This profile is used to guide iMath's efforts to generate an image with "similar" qualities.

3.2.2 Color Distribution

The second attribute of an image characterized by iMath is its distribution of color. Multiple techniques for quantifying the color attributes of an input image were attempted in the development of iMath. For the sake of brevity only the most

[2]There appears to be no "silver bullet" regarding the optimal size of this window; the window size that generates the most pleasing results for one image may perform poorly on another

effective method observed is presented in this work. Understanding it requires a basic understanding of the modern application of wave theory to digital imaging.

Images As Waves Digital images can be interpreted as a waveform. To understand this interpretation, consider how intensity is represented in an 8-bit grayscale image. Each row of the image is represented as a set of intensity values (one for each pixel) on the range [0 - 255] with 0 representing full black and 255 full white. These values can be interpreted as discrete samples taken of some type of wave, such as a sound wave, with the values representing measurement samples, such as air pressure values, taken at fixed intervals. By extending this concept along the columns the entire image can be considered a single 2-dimensional waveform. The usefulness of this interpretation will become apparent shortly.

In the 19th century, French mathematicianJean Baptiste Fourier theorized that any given waveform, no matter how complex, could be constructed through the summation of simple sines and cosines of various frequencies, amplitudes and modulations. This sum is known as a Fourier series. The Fourier Transform and its derivatives such as the Fast Fourier Transform (FFT) and the Discrete Fourier Transform (DFT), perform the decomposition of a waveform into its constituent frequencies, providing their amplitude and modulation coefficients.[2]

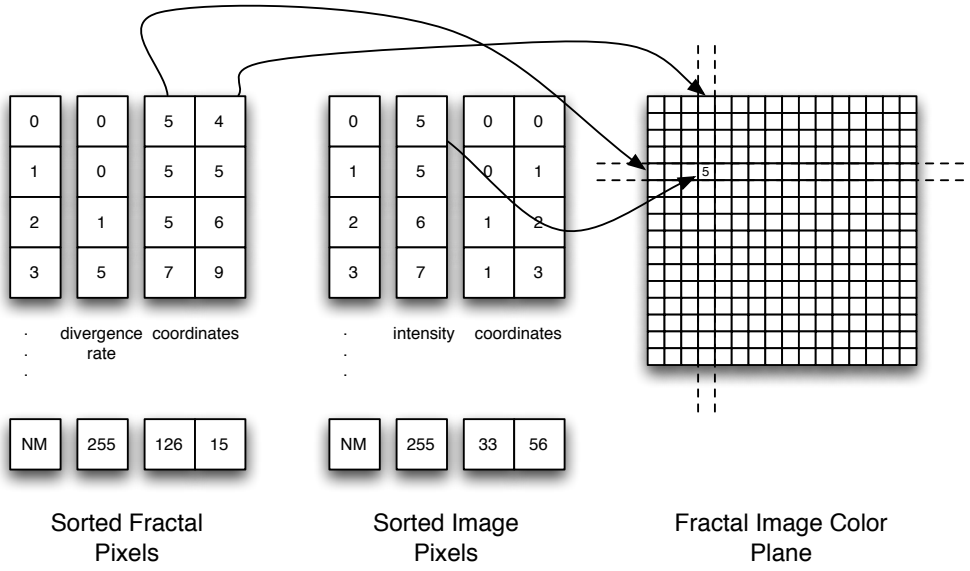
By applying the aforementioned interpretation of an image as a waveform, Fourier Transforms can be used to analyze and manipulate images, originally represented in the spacial domain, in the frequency domain. Numerous benefits arise from the ability to do this and the technique is commonly used in digital imaging applications. The reader is referred to [2] for more in-depth coverage of both Fourier transforms and their application to digital imaging.

iMap utilizes the DFT of the input image in a manner similar to the complexity profile introduced in section 3.2.2. The input image is first separated into its red, green and blue (RGB) color planes. Each of these planes is an array of intensity values identical in size to the original image. The Fourier series (FS) of each of the three planes is calculated and stored as the first part of the image's color profile. The FS is a good fit for this task as it provides a quantifiable baseline by which two images can be compared and the quantities involved contribute to the image as a whole (i.e., each constituent frequency is applied to the entire image vs. being localized to an individual pixel).

Color Mapping The second component of the input image's color profile is a set of sorted lists of the pixels in each of the images RGB color planes. In section

2.1.2 I mentioned the need to assign colors to the divergence rate values generated by iMath's fractal generator; these three lists are the method by which this is accomplished. The pixels of the fractal image are sorted according to their divergence rates and a simple one-to-one one mapping against the seed image's lists is then used to assign intensities to RGB color planes of the fractal. The fractal is now a 24-bit RGB color image. Figure 5 illustrates how this color mapping is done.

Figure 5: Color mapping technique



With the RGB values of the fractal image established, the same techniques utilized to generate the complexity and color profiles of the input image are applied to the fractal. These values are then used to evaluate how well the fractal mimics the complexity and color profiles of the input image.

3.3 Search

Utilizing the concepts mentioned thus far iMath is able to mimic the three-step process that humans often employ when searching for impressionistic fractals. In short: generate, evaluate, adjust. Because color mapping is managed in a fixed manner, as described in section 3.2.2, and the equation generating the divergence values is currently fixed, only four parameters can be adjusted: the coordinates

of the upper-left and lower-right corners of the bounding box within which the Mandelbrot equation is to be evaluated. The nonlinear behavior of the Mandelbrot equation makes the task of "tuning" these parameters extremely non-trivial. Unlike a human, iMath does not have the benefit of intuition or any such "sense" of how to modify these coordinates and clearly an exhaustive search is out of the question. iMath employs particle swarm optimization (PSO) to search for good coordinates[3]. The reader is referred to [3, 4, **?**] if they are unfamiliar with PSO.

iMath utilizes the ring topology and inertia variant of PSO. Every 10 iterations the best parameters discovered by the swarm thus far are used to re-initialize all of the particles in the swarm, with the exception of the current swarm leader, to new locations within the window defined by these coordinates. I will refer to these 10 iteration intervals as generations. The intended effect of this re-initialization is that the swarm is more inclined to "zoom-in" on the Mandelbrot set within the regions where progress is being made. This modification to simple PSO resulted in qualitative improvements in the output images.

The fitness of the fractal images represented by particles during PSO is embodied in two error values:

- The complexity profile error = the sum of squared errors over the values in the fractal's complexity profile relative to the input image's profile.
- The color profile error = the sum of squared errors over the DFT values of the RGB color planes of the fractal relative to those of the input image.

The lower the error the better the fitness; in this case, optimization amounts to minimizing these error values. The complexity and color errors are given equal weight in determining the fitness of a particle relative to the rest of the swarm. When two particles are compared it is possible for a tie to occur due to one particle being superior relative to complexity error and the other being superior relative to color error. Such a tie is broken by comparing the ratios of the corresponding scores between the two particles. For instance if particle A has a complexity profile error value that is half as large as particle B's, and particle B has a color profile error that is one third as large as particle A's, then particle B is deemed to be superior.

The swarm size and number of generations the PSO algorithm is to perform are selected by the user when iMath is invoked.

[3]Numerous nonlinear optimization/search algorithms such as a genetic algorithm might just as easily have been applied.

4 Results

Here I present samples of iMath's performance on 4 input images. These images were take from multiple sources on the web as follows.

1. "Before The Storm":
   `http://www.spacewallpapers.net/wallpapers/albums/CG/before_the_storm_1600.jpg`
2. "Cataclysm" :
   `http://browse.deviantart.com/fractals/?q=cataclysm#/d12dm22`
3. "e7a16-7321":
   `http://www.wallpaper4computer.com/wallpaper/?autumn/1024/e7a16-7321`
4. "Bridge004":
   `http://www.fwallpaper.net/buildings-bridge_004.html`

The images are arranged as pairs with the seed image on the left, the impressionistic fractal generated by iMath on the right.

Figure 6: Input Image "Before The Storm"

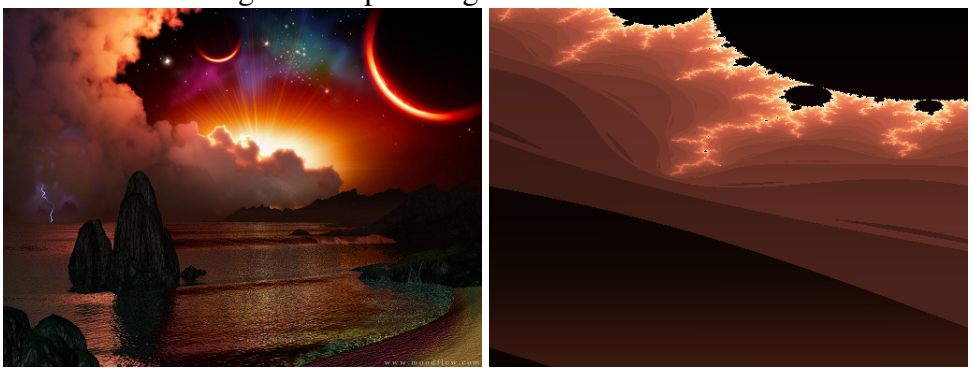


Figure 7: Input Image “Cataclysm”

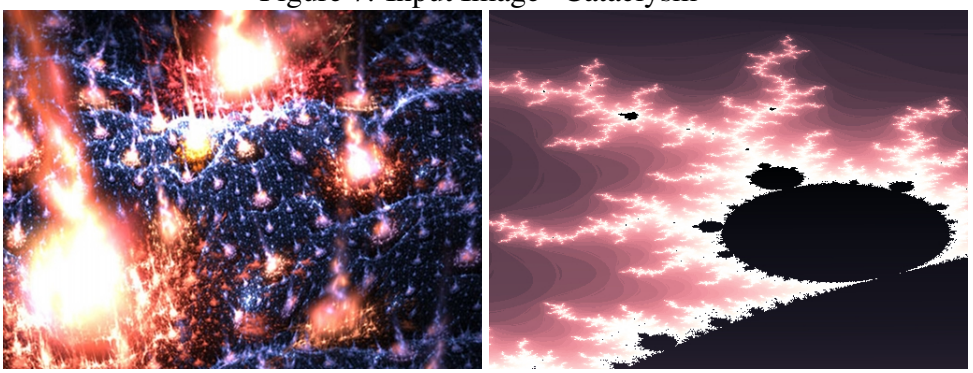


Figure 8: Input Image “e7a16-7321”

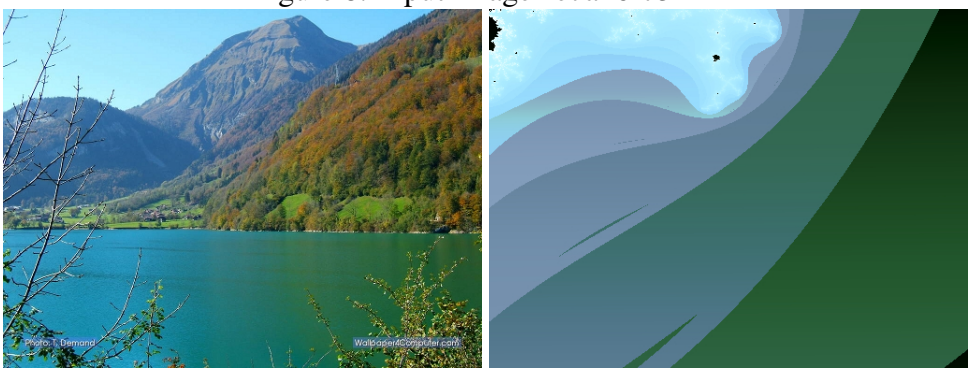
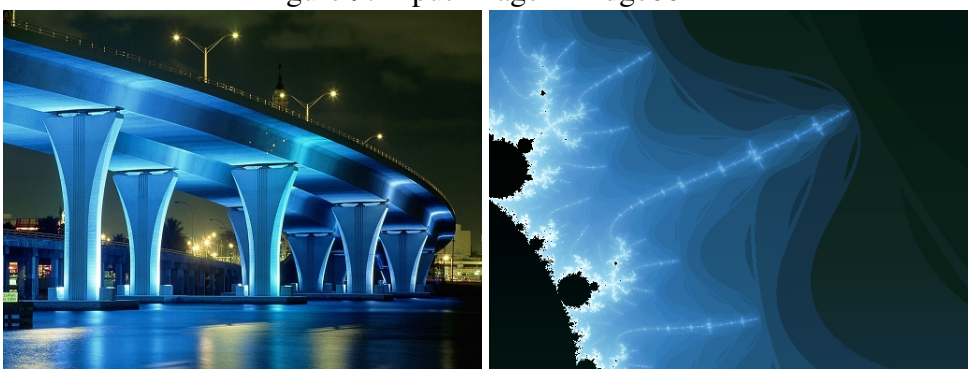


Figure 9: Input Image “Bridge004”



5 Conclusions and Future Work

This work has presented iMath, an application for generating impressionistic fractal art based upon a user-provided seed image and two user-tuned parameters. I have shown that iMath succeeds in automatically generating fractals that convey attributes of color and complexity that are influenced by the seed image. Some of these images are impressive in their beauty and (in my opinion) qualify as impressionistic renderings of the input images.

Several weaknesses with the approaches taken in iMath are apparent. For certain types of images the techniques employed work well but for others they fail miserably. Using box dimension as a means of quantifying complexity prevents excessive oversimplification in the final results but generally fails to capture any meaningful structure such as arcs or lines etc. Though the final images typically have a color distribution that is similar to that of the input image at times significant elements such as the presence of a singular but significant detail in the input image is not always preserved in the output image. A method of identifying such outliers (which our eyes consider important) and weighting the fitness of a candidate image accordingly is needed.

The fact that iMath's fractal generator is fixed (i.e. the parameters of the equation cannot be changed) limits the type of geometry that can be produced. Additionally, the fact that the entire image is generated from a single fractal is at odds with the fact that the individual components of a scene are typically fundamentally varied, even if they are fractals (or instance clouds vs. trees). A logical next step would be to segment the input image and attempt to generate fractals for these segments and conjoin them in the final output image.

These and other issues, such as experimenting with alternative search methods, are the focus of future work on this project.

References

[1] Farès Belhadj and Pierre Audibert. Modeling landscapes with ridges and rivers. In *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '05, pages 151–154, New York, NY, USA, 2005. ACM.

[2] R. Gonzalez, R. Woods. *Digital Image Processing 3rd Edition*. Pearson Education, Inc, 3rd edition, 2008.

[3] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948, 1995.

[4] J. Kennedy and R. Mendes. Population structure and particle swarm performance. In *Proceedings of the Evolutionary Computation on 2002. CEC '02. Proceedings of the 2002 Congress - Volume 02*, CEC '02, pages 1671–1676, Washington, DC, USA, 2002. IEEE Computer Society.

[5] Peter E. Oppenheimer. Real time design and animation of fractal plants and trees. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '86, pages 55–64, New York, NY, USA, 1986. ACM.

[6] Bulusu Rama and Jibitesh Mishra. Generation of 3d fractal images for mandelbrot set. In *Proceedings of the 2011 International Conference on Communication, Computing & Security*, ICCCS '11, pages 235–238, New York, NY, USA, 2011. ACM.

[7] Dietmar Saupe and Raouf Hamzaoui. A review of the fractal image compression literature. *SIGGRAPH Comput. Graph.*, 28:268–276, November 1994.

[8] G. Singh. Fractal fracas. *Computer Graphics and Applications, IEEE*, 25(3):4 – 5, may-june 2005.

[9] Gary Singh. The spiral effect. *Computer Graphics and Applications, IEEE*, 29(2):4 –5, march-april 2009.

[10] Gary Singh. Alive and digital. *Computer Graphics and Applications, IEEE*, 30(3):4 –5, may-june 2010.

[11] Steven H. Strogatz and Ronald F. Fox. Nonlinear dynamics and chaos: With applications to physics, biology, chemistry and engineering. *Physics Today*, 48, 1995.

[12] Ningping Sun, Ryo Miyazaki, and Naoki Yoshida. Complex mapping with the interpolated julia set and mandelbrot set. In *ACM SIGGRAPH ASIA 2010 Posters*, SA '10, pages 49:1–49:1, New York, NY, USA, 2010. ACM.

Poincaré Tomography

Ron Kneusel, CSCI 5446, April 2011

Abstract

A Poincaré section of a three dimensional state-space trajectory is a plane showing the location of the places where the trajectory crosses the plane. In Poincaré tomography, true projections are made of the state space by assigning an attenuation coefficient (μ) to the trajectory points and calculating line integrals along specified directions. For any fixed angle, the output is a projection image. With a complete 360° set of projection images and filtered backprojection, it is possible to reconstruct the original state-space trajectory, with possibly modified emphasis of points due to the attenuation coefficient. This paper explores some of the possibilities of this technique.

1 Introduction

Computed tomography (CT) is a common process in medical imaging [7]. The most common version of computed tomography is x-ray transmission tomography where a beam of x-rays is passed through the patient's body producing a projection detected on the other side. If many such projections are acquired, it is possible to reconstruct the x-ray absorption map of the interior of the body. There are several possible ways to do this reconstruction, with the most prominent being filtered backprojection (FB) which is described in detail below.

This report presents an application of FB to the visualization of 3D state-space trajectories of chaotic systems. Section 2 describes the application and algorithms developed for visualizing the trajectories. Section 3 presents the projections and volumes generated. Section 4 concludes the report with a discussion of the results and possible future work.

Filtered Backprojection

If a beam of x-rays is transmitted through an object and the resulting shadow image is collected, a projection is produced. A single projection is equivalent to a classical x-ray image with overlapping structures within the object piled on top of each other and with an x-ray density related to the type and thickness of material through which the x-rays pass. From a classical physics perspective, the x-ray beam is attenuated according to Beer's Law,

$$\frac{\mathrm{d}I}{\mathrm{d}s} = -\mu(x)I$$

where I is the intensity, x the material thickness, $\mathrm{d}s$ in the direction of x, and μ is the attenuation coefficient, specific to the type of material [1]. The attenuation is also a function of x-ray energy, $\mu = \mu(E)$, where $E = h\nu$. The integration of Beer's Law leads to

$$\log \frac{I}{I_0} = -\int_l \mu \,\mathrm{d}x$$

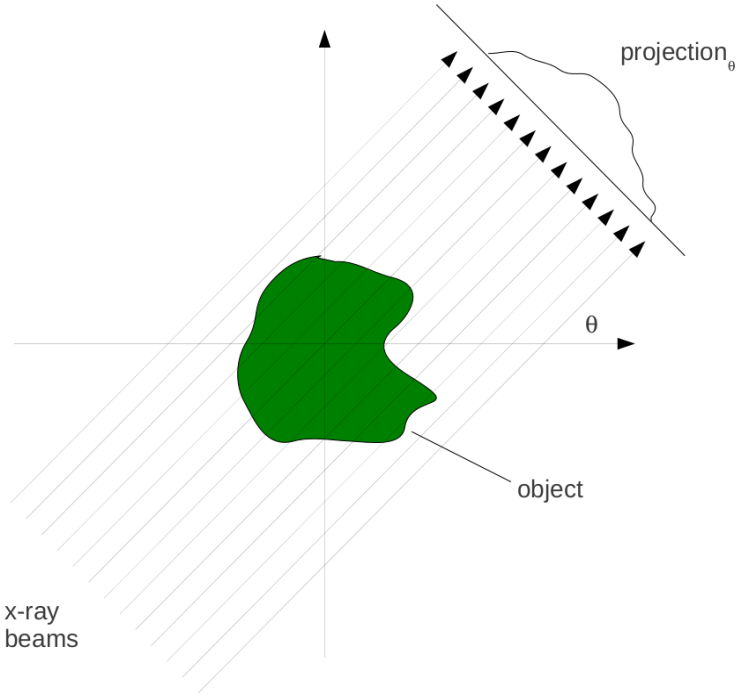


Figure 1: Parallel-beam projections built from a set number of x-ray sources (or a single source scanned along the projection direction). This work used a parallel-beam approach with the number of x-ray sources equal to the desired reconstructed volume length.

where dx is along the direction of the x-ray beam and I_0 is the incident x-ray intensity. The line integral along a particular direction can be determined by measuring the incident and remaining intensity. This relationship, in a simplified form, will be used below to generated projection images from state space trajectories.

Modern CT systems use point x-ray sources and fan-beam or cone-beam geometry, which complicates reconstruction. In this work, the classical first-generation CT strategy of parallel-beam projection is used, as shown in Figure 1.

The figure shows a 2D image with a 1D projection. A volume in 3D projects to a 2D image made up of a stack of 1D projections, i.e., the 2D projection (xy plane) is repeated for each slice (z). The object or x-ray source is then rotated by a fixed angle and a new projection image is made. The number of projections and the rotation angle are explored below.

A collection of projections for a full 360° rotation can be reconstructed via filtered backprojection. In backprojection, a projection is "smeared" back across the output image (or slice in 3D) at the angle at which it was acquired. The "smearing" adds the projection value along that line to any existing image value. This process is then repeated for all projections. While simple backprojection is sufficient to recover some insight about the internal structure of the x-rayed object, the backprojection process introduces a point-spread function which is convolved with the object and introduces blurring. The blurring can be corrected by first filtering the projections with a high-pass filter. This, in effect, adds negative regions to the edges of the projections thereby cancelling out the blurring when summed over all projections. The filter is best applied in Fourier space as a multiplication of the projection by a ramp filter (ideal case, suitable here). This was

```
1. for each 2D projection:
2.     proj = empty array (2D)
3.     for each slice:
4.         generate an array representing the slice
5.         rotate the slice array by the proper angle (2D)
6.         calculate the projection along the x direction (1D)
7.         store the 1D projection in the output array (proj)
8.     write the projection array, proj, to disk
```

Figure 2: The projection algorithm.

done for all the volume reconstructions in this paper.

The projection and reconstruction steps described above are essentially those of the Radon transform [9]. However, in this work, no pre-existing Radon transform, or its inverse, were used; all steps were coded directly from the algorithm.

Related Work

As of this writing, no literature specific to the application of tomographic reconstruction of 3D state space trajectories has been found. The application below includes utilities for generating known data sets from the Lorenz [8] and Rössler [10] systems, as well as applying principal component analysis (PCA) to reduce the dimensionality of the input data. These are well-known techniques and not novel. Other authors have investigated other techniques for presenting high-dimensional chaotic data [4] [5] [2] [3].

2 Methods

This work consisted of three main parts: the generation of projection images with desired characteristics through the manipulation of the attenuation coefficient, μ, the reconstruction of projection images for volume rendering, and the construction of a helper application to load and process input data for the first two steps. All code for this work was written using IDL (Interactive Data Language, ITT Visual Information Solution, Boulder, CO).

Projection

As shown in Figure 1, simulation of a projection image is accomplished through a line integral passing through the state space. For the case here, projection proceeded as in Figure 2 , as rotating the slice array is equivalent to rotating the object itself. The rotation code used bicubic interpolation. This is likely the source of some of the rendering error that will be seen below.

The slice array is built from the actual trajectory points and the discretization of the volume encompassing the trajectory points. The box containing each point in the trajectory is first calculated. If that box is in the slice currently being processed, the attenuation value, μ, for that point is added to any value already present in the slice array for that box. The box size is determined by the

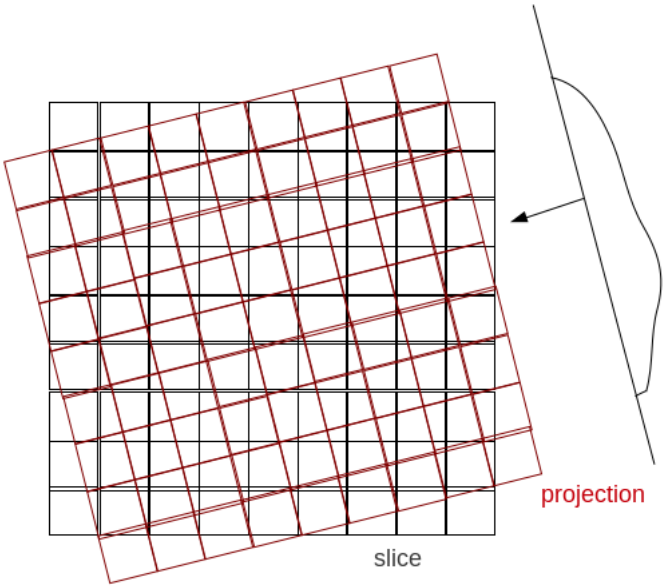


Figure 3: Slice reconstruction via summation of projections. The projection is replicated to produce a projection array. The array is rotated by the appropriate angle and summed to the slice array. This process is repeated for each projection.

number of slices and is such that the largest extent of the trajectory will be covered by the desired number of boxes.

The attenuation value for each point is the key in determining the appearance of the reconstructed slices and rendered volumes. In the simplest, case a constant value was used ($\mu = 1$) thereby making the shading of each output projection "image" directly related to the number of trajectory points along the projection line. Varying μ by position or some other attribute might enhance regions of the reconstruction where a larger value was present and diminish regions with smaller μ. This is directly analogous to the increased absorption of x-rays by dense tissue (bone) versus less dense tissue (organs) in CT imaging of the human body.

The projection along $\hat{x}$ is then simply the summation of the pixel values for the rotated slice array along that direction. This produces the final 1D projection vector that is added to the output projection array. The process is identical, for a given slice, to that shown in Figure 1, where θ is always zero.

Reconstruction

Reconstruction of slices from projections are accomplished using filtered backprojection. The specific algorithm followed is in Figure 4. The 1D projections for the slice being reconstructed are backprojected by replication producing an array of the same size as the output slice. This array is rotated by the necessary angle and added to the existing output slice. This is process is repeated for each projection and the resulting slice is written to disk. See Figure 3.

```
1. for each slice:
2.     for each projection, angle in that slice:
3.         fp = Filter(projection)
4.         slice += Backproject(fp, angle)
```

where `Filter` is,

```
1. fp = FFT of projection zero-padded to 4x original length
2. ramp = 45-degree ramp for positive and negative frequencies
3. p = inverse FFT of (fp*ramp)
```

and `Backproject` is,

```
1. t = 1D projection replicated across columns
2. t = rotation of t by given angle
3. slice += t
```

Figure 4: Slice reconstruction from projections.

Application

The helper application consists of a small window with several tabs and makes uses of the pre-existing visualization tools available in IDL. The relevant screens of the application are shown in Figure 5.

Projections are specified by number of slices, which controls the 2D projection size, and by the number and step size of the projection angles. A full 360° rotation with 1° steps was used for most of the volumes rendered for this paper. In some cases a step size of 0.5° was used instead. Each projection image was stored on disk as an image file for display and as raw data.

Reconstruction requires only the source directory containing the projection data and the angular step size used. The resulting reconstructed slices are then stored on disk in a separate directory. These can be loaded into IDL and quickly assembled into a volume which is always a cube of size n where n is the number of slices. For rendering, the data cube is scaled to bytes.

High-dimensional (> 3) data in TISEAN-compatible format [6] can be loaded through the 'File' menu. If the application detects more than three dimensions, the dimensionality reduction dialog is presented. The user can then select specific dimension numbers (zero-based) to load, or apply principal component analysis (PCA) to select the three directions that describe the largest amount of variation in the data.

As a convenience, the application allows the user to directly calculate data sets using the Lorenz or Rössler systems. The initial condition and parameter values can be entered and run for a given number of points using either fourth order Runge-Kutta (RK4) or adaptive RK4. The resulting trajectory can be viewed in 3D and written to disk for use in other applications.

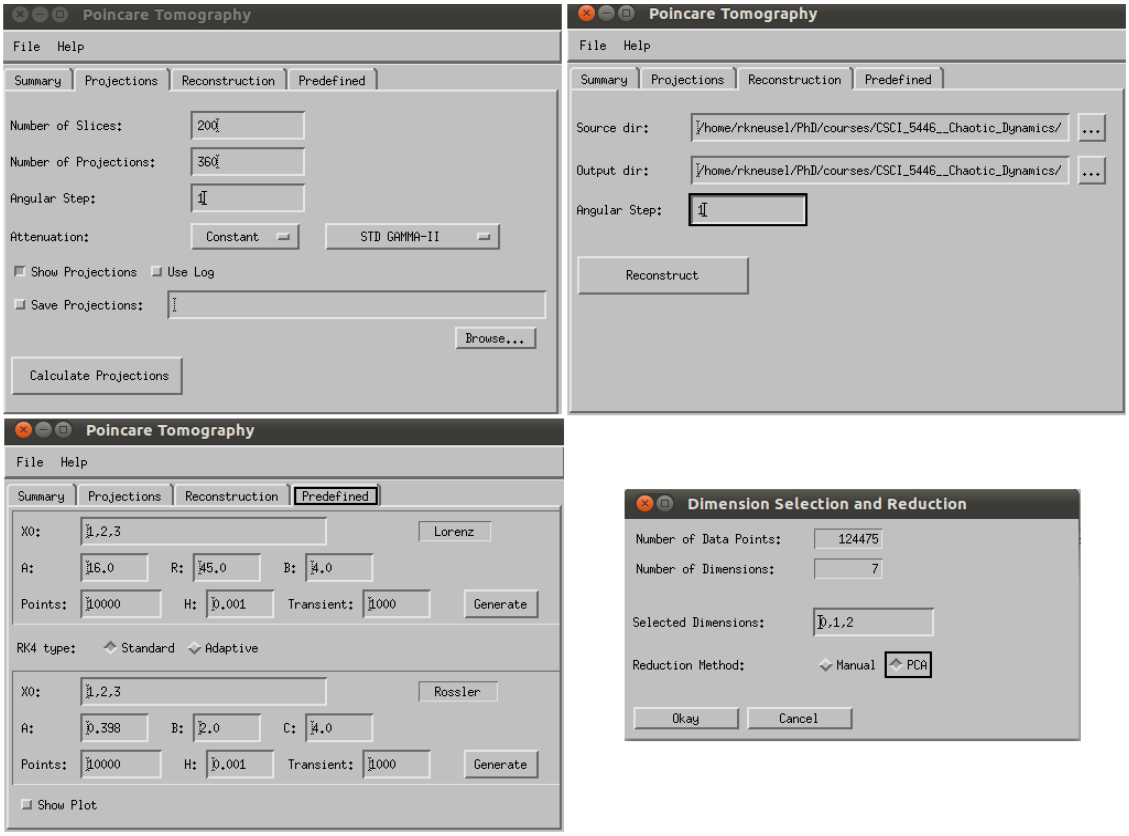


Figure 5: The helper application for Poincare Tomography. (Clockwise from the upper left) The projections are specified by the number slices (projection image size, square), the number of projection angles, and the step between the projections (degrees). Reconstruction is from projections to slices given a directory of projection images and the angular step between them. Higher-dimensional data is reduced to 3D via selection or specific dimensions or the application of PCA. Lastly, predefined data can be generated for the Lorenz and Rössler systems internally.

3 Results

The projection/reconstruction process described above was used in a series of experiments to determine the usefulness the process for the trajectories of chaotic systems. First, the 2D projections themselves are presented. Second, volume renderings of the reconstructed slices are presented. Both of these will be discussed in Section 4.

Projections

A sample projection of the Lorenz attractor is shown in Figure 6. A color table is applied to show the effect of the attenuation. The number of slices was 800 which results in an 800x800 projection image. The attenuation coefficient of each point was fixed at $\mu = 1.0$ so that the shading is representative of the density of trajectory points along the projection direction. The color table is provided in the figure. The projection data was scaled to the range $[0, 255]$ before application of the color table. It is clear that the most point-dense region is near the center of one of the "butterfly" wings, as expected.

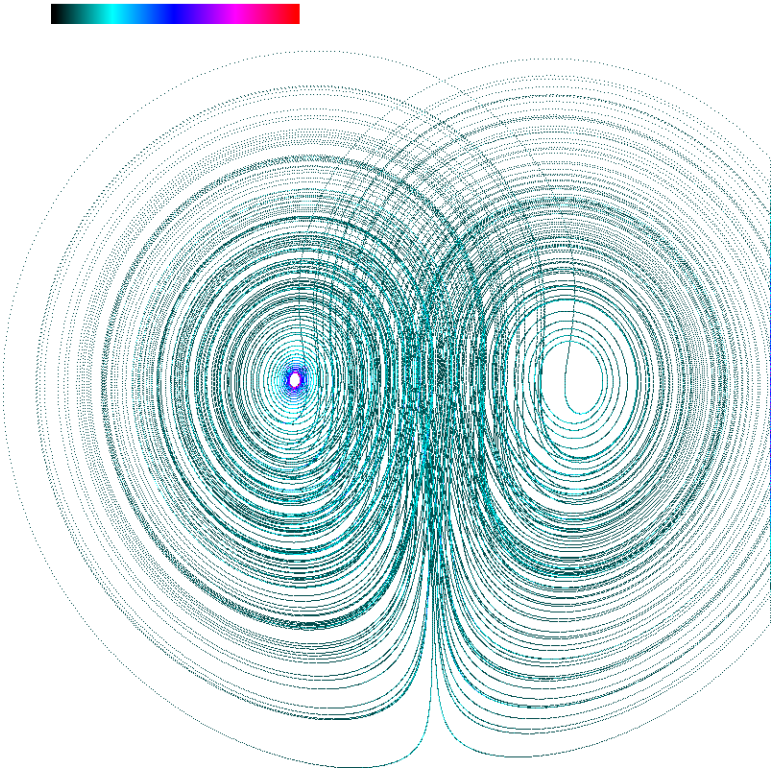


Figure 6: A projection with constant $\mu = 1.0$ for all points. Lorenz attractor ($A = 16, R = 45, B = 4$), 800 slices, 150,000 points (RK4), first 1000 discarded. The color table (low to high, left to right) shows the shading.

Figure 7 shows a projection for the Rössler system where the attenuation coefficient is a function of the distance from the origin: the larger the distance, the larger the value of μ.

Volume Rendering

The full set of projection images, covering all 360° in 1° or 0.5° increments, were reconstructed to produce a set of output slices along $\hat{z}$. This set of slices was then loaded into IDL for display as a rendered 3D volume. A data filter was applied to remove low value pixels from the volume in order to reduce the diffuse background artifacts.

The volume reconstruction itself uses alpha blending for the composite function. Volume rendering requires passing a ray through the object in 3D to create the pixel as seen on the computer screen. When passing through the volume, there are choices as to how the final pixel value is calculated. In alpha blending, the following recursive equation is used,

$$d_{n+1} = \alpha s + d_n(1 - \alpha)$$

where s is the source pixel in the volume along the ray, d is the previous output pixel, and α is the blending factor with a default of 0.5. It is important to distinguish between a volume rendering and plotting in 3D of a set of points.

First, consider Figure 8. In this figure, the right volume represents the Lorenz system (parameters in the figure) when a constant μ value is used. On the left, the same trajectory is reconstructed

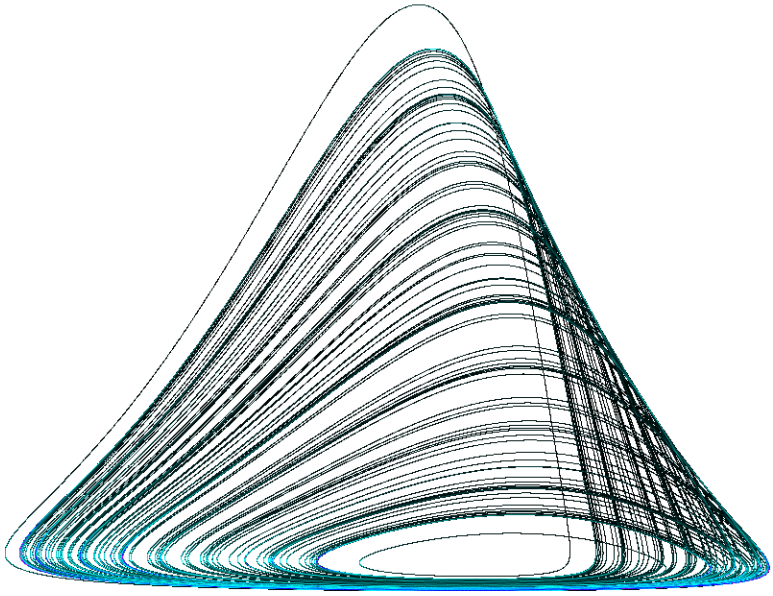


Figure 7: The Rössler system ($A = 0.398, B = 2, C = 4$), 800 slices, μ constant, 600,000 points (RK4), first 1000 discarded.

but this time the μ value changes with the position along $\hat{x}$. It is clear that the smaller attenuation results in a smaller, hence dimmer, volume reconstruction, as expected.

In Figure 9 a larger reconstruction of the Lorenz system is presented. In this case, an 800x800x800 volume is rendered. It is filtered to remove volume elements less than 2 (after byte scaling to $[0, 255]$) which reduces the diffuse background seen in the 200x200x200 volumes.

A color table can be applied to the rendered image, i.e., the image displayed after alpha blending; this is shown in Figure 10. The color table makes for a cleaner image but it is important to remember that the colors show the effect of the alpha blending as well as the effect of any change in μ value.

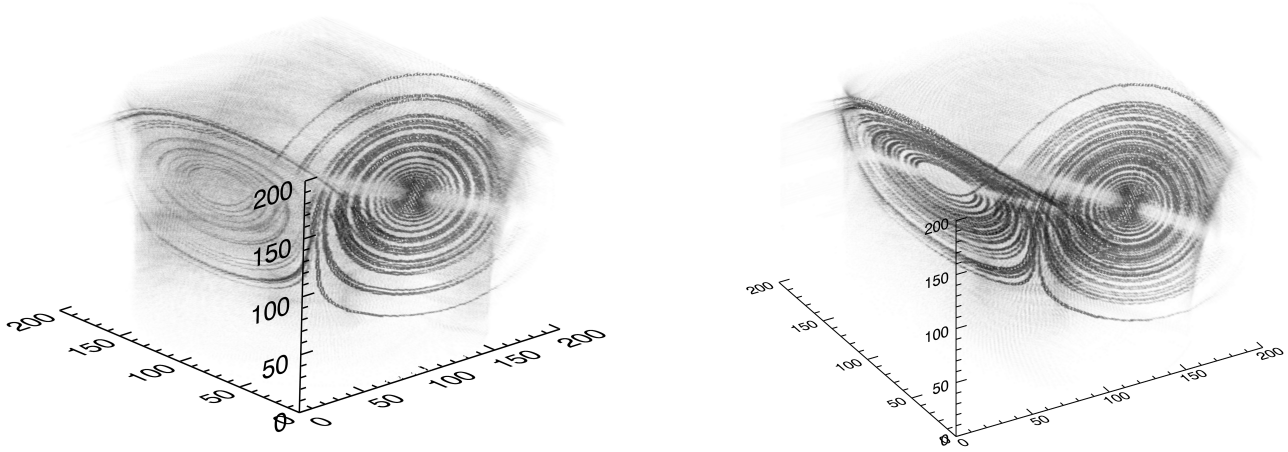


Figure 8: Volume reconstructions of the Lorenz system. For both, 30,000 points, ($A = 16, R = 45, B = 4$), 200 slices resulting in a 200x200x200 volume. On the left, $\mu = 0.25$ for $x > 0$ and $\mu = 1$ otherwise. On the right, $\mu = 1$ always. The effect of the μ value is easily seen. The volumes are somewhat small and reconstruction effects have not been removed by filtering.

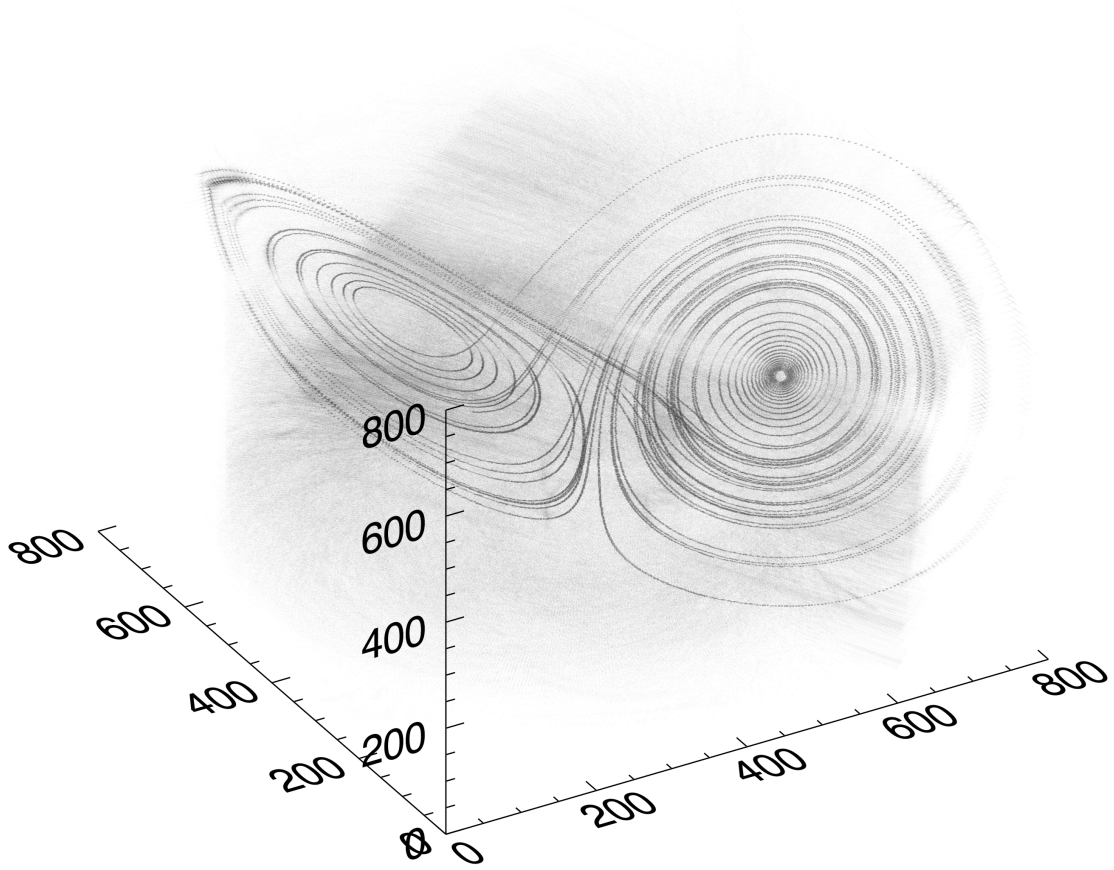


Figure 9: The Lorenz system with the same parameters as in Figure 8 but reconstructed to 800x800x800 and filtered to remove volume values less than 2.0. The filtering removes much of the diffuse cloud of points and the larger reconstruction volume shows sharper discrimination between points.

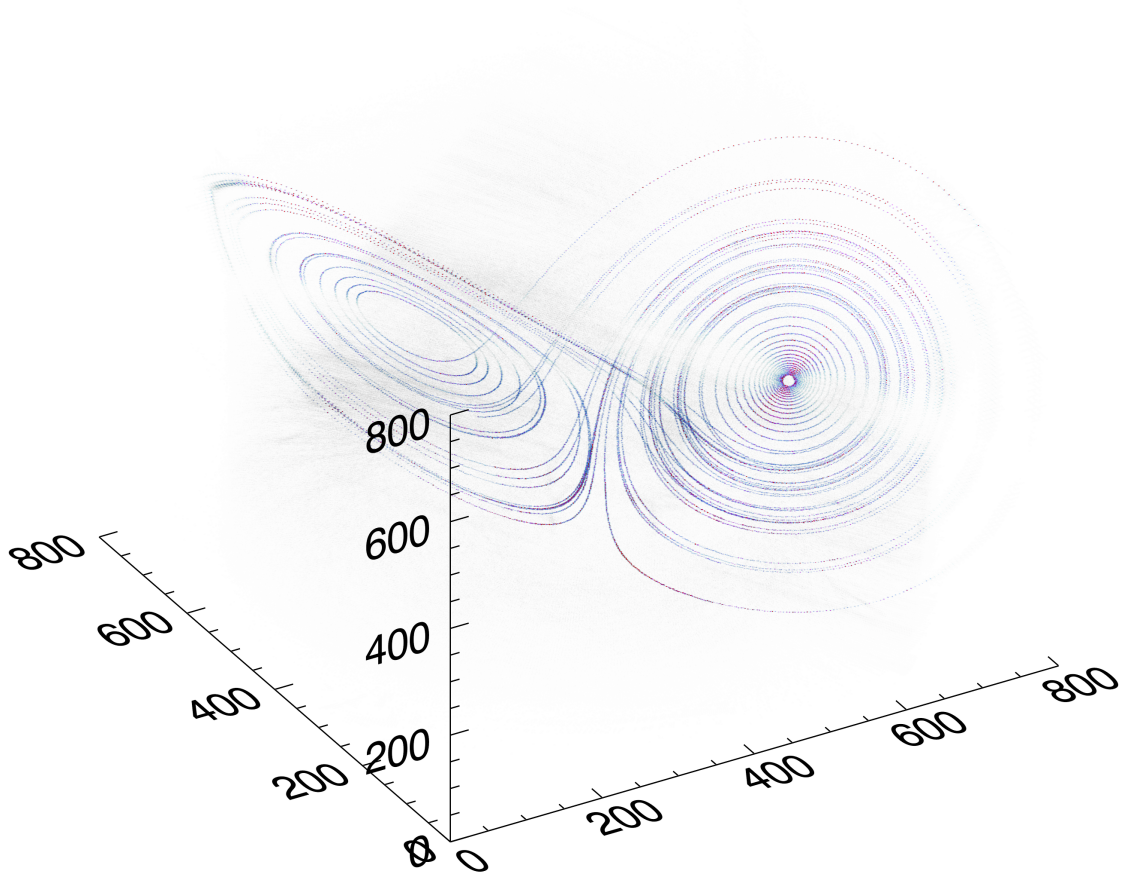


Figure 10: The same as Figure 9 but applying a color table to the rendering. This produces a cleaner looking image but may obscure the shading due to any changes in the μ value as the color selection is based on the alpha blending result.

4 Discussion

The projections shown in Section 3 are useful on their own. Instead of a truly flat projection, as is found by ignoring some dimensions while plotting others, the projections show some essence of the original structure of the trajectory, in much the same way that a classic chest x-ray shows the internal structure of the body. The ability to make such projections in any orientation through the trajectory, which is not fully explored here, could add insight to the state space structure. The addition of color to the projections, either through a color table as shown here, or through the application of a more sophisticated color assignment scheme. This is a possible future direction to add still more to what can be gained through visual inspection.

The exact projection implementation outlined in this paper cannot be considered state-of-the-art, however. The reconstructed volumes show artifacts, as do virtually all image reconstruction techniques, that could perhaps be mitigated by refining the projections. Of necessity, the projections are only approximations of the trajectory, as the state space must be discretized. Because of this, two or more trajectory points will appear in the same location in the projection image. This overlapping is not all negative, perhaps. One could imagine a scenario where the projection image size, combined with the pixel values (which need not be integer), might scale in a way that is related to the capacity dimension of the chaotic system. A single projection may not be sufficient for this purpose, but a small *set* of projections, perhaps along the orthogonal directions, might allow such a calculation. This is also a topic for future work.

Reconstruction of projections into a volume for visualization was also demonstrated in this paper. It is clear that there are artifacts in the reconstructions, however. In part, some error is expected as filtered backprojections is only completely accurate in the limit that the number of projections goes to infinity. Also, as mentioned above, the size of the projection images also greatly affects the results. In reconstruction, one is attempting to build a volume representing what is in reality a mathematical ideal, a cloud of points. After reconstruction, the points have an existence in three dimensions through the voxel size of the volume.

Visually, the larger the projection image, the cleaner the rendered volume. From a comparison of Figure 8 to Figure 9, it is clear that some volume artifacts are due to the size of the projection images. Also, in the smaller volumes, on the edges of the attractor, there are streaks which are likely due to a combination of finite number of projections (and angular separation of projections) and the size of the boxes the state space is broken into.

The exact cause of the diffuse haze in the reconstructed volumes is unclear. Imperfections in the choice of the ramp filter shape will lead to imperfect cancellation of the blurring introduced in the backprojection process. If this is the case, residual information will be left in places in the volume where there should be none. A curious artifact is also seen in some volume renderings. This is the appearance of two light colored spokes starting near the center of the Lorenz attractor running radially outward.

It is known that iterative tomographic reconstruction [7] produces superior results to filtered backprojection. However, until recently, the computational cost of iterative reconstruction has prevented its acceptance in commercial tomographic systems. It is possible that many of the artifacts seen in this paper would be greatly reduced or eliminated by using iterative reconstruction; still another area for possible future work.

The visualization of high-dimensional data often makes use of interactive techniques [4]. Future

work will look at the introduction of such a system applied to the techniques described here. For example, one might be able, with a mouse and a 3D region of interest, to select a particular part of state space and generate shaded projections or reconstruct the a small region in 3D. The combination of user selection of locations, assignment of attenuation coefficients, projections (2D), reconstructions (3D) and color could perhaps greatly assist the researcher interested in gaining insight into the structure of a complex system.

This paper concerns the application of tomographic principles to the 3D state space of a chaotic system. It was shown that it is possible to create projections of the points in the trajectory by simulation of the x-ray transmission process and the assigning of attenuation values to the points. In this sense, the technique was successful. Yet, in another sense, this is a technique in search of a specific use. Perhaps further investigation of the items mentioned above will reveal an ideal situation in which to use Poincaré tomography.

References

[1] C. Epstein, *Introduction to the Mathematics of Medical Imaging*, SIAM, 2nd ed, 2008.

[2] E. Groller, *Application of Visualization Techniques to Complex and Chaotic Dynamical Systems*, Proceedings of 5th Eurographics Workshop on Visualization in Scientific Computing, Rostock, Germany, 1994.

[3] E. Groller, *Interactive Exploration of Dynamical Systems*, Technical University of Vienna, Austria.

[4] K. Gruchalla, *Progressive Visualization-Driven Multivariate Feature Denition and Analysis*, Ph.D. Disseration, University of Colorado, Boulder, CO, 2009.

[5] K. Gruchalla, E. Bradley, *Stream Hulls: A 3D Visualization Technique for Chaotic Dynamical Systems*, University of Colorado, Boulder, CO, (unpublished?)

[6] R. Hegger, H. Kantz, and T. Schreiber, *Practical implementation of nonlinear time series methods: The TISEAN package*, CHAOS 9, 413, 1999.

[7] J. Hsieh, *Computed Tomography: Principles, Design, Artifacts, and Recent Advances*, SPIE Press, 2003.

[8] E. Lorenz, *Deterministic nonperiodic flow*, J. Atmos. Sci., Vol 20, pp. 130-141, 1963.

[9] J, Radon, P.C. Parks (translator), *On the determination of functions from their integral values along certain manifolds*, IEEE Transactions on Medical Imaging, Vol 5, No. 4, pp. 170176, 1986.

[10] O. Rössler, *An Equation for Continuous Chaos*, Physics Letters 57A (5): 397398, 1976.

Preliminary Methods and Results on the Intelligent Exploration of Reachability Sets

Erik Komendera
Department of Computer Science
University of Colorado at Boulder

May 10, 2011

Abstract

This paper introduces a new technique for intelligently exploring the reachability sets of spacecraft in arbitrary systems. A reachability set for Δt is the set of trajectories in state space that result from a set of perturbations on a set of trajectories. The chaotic nature of most gravitational systems makes it difficult to quickly characterize regions in the reachability set. Artificial intelligence routines can assist in exploring regions of interest. With a search function and a fitness function, certain regions of the reachability can be explored quickly and in greater detail, allowing an on-board spacecraft to determine its own course without input from Earth-based controllers. In this paper, I report on initial explorations with a simple search function and two fitness functions, analyze the reachability set of an arbitrary starting point in the reduced three-body problem, and discuss future work.

1 Introduction

In space mission planning, the reachability set for Δt is the set of all possible positions and velocities obtainable through the consumption of fuel at any times between the t_0 and $t_0 + \Delta t$. This determines the entire region of space that can be reached in a finite amount of time, and is incredibly important in determining impact and escape scenarios. The state space for gravitational systems has 6 dimension, 3 for the position vector and 3 for the velocity vector. In state space, a ΔV fuel burn corresponds to a jump along the velocity coordinates of magnitude ΔV. The reachability set for a single burn is therefore the set of points reachable in Δt from the trajectories that are within the ΔV sphere at the time of the burn. In space missions, Δt might represent the extent of certainty in future planning, or the lifetime of the spacecraft.

Starting from an initial position and velocity, a naive simulation method for exploring the reachability set is to apply ΔV on evenly-spaced intervals along a subset of a sphere of radius ΔV. Evenly-spaced test trajectories may miss complicated regions of space while over-sampling uncomplicated regions. Humans can scan data and identify interesting features of the reachability set, such as a high level of deformation and variability, and use that information to refine the search. In the absence of humans, artificial intelligence techniques may stand in proxy and make high-level decisions on which regions of the reachability set deserve more attention. This can be used to plan a complicated trajectory in a short amount of time. A spacecraft in orbit has a small window of opportunity to plan ahead, and Earth may be too far for fast communication. For example, a spacecraft without contact with Earth may need to quickly determine a course of action that does not result in an impact or an escape.

Some trajectories with close starting conditions in chaotic space may diverge wildly after a short amount

of time, or they may have similar fates. Thus, an intelligent search algorithm must be able to distinguish between regions that have a high level of divergence and regions that share similar trajectories and fates. A large region of impacts on a body, for example, may not require exhaustive exploration. However, "holes" may exist in such regions, so an intelligent search algorithm must be able to find holes. Finally, the search algorithm must be fast enough to be practical.

Several researchers have devised methods to partition state space into such regions. They include Feng Zhao's ([12]) algorithm for partitioning a state space into flowpipes characterizing regions of similar behavior, separated by boundaries. This work developed into the Spatial Aggregation Language ([1], [9], [11]), which can autonomously aggregate a set of data into a set of clusters. The work of Robins et al., ([5], [6], [7]) examines the topology of data points and uses this information to accurately group related data. These methods share with my approach the desire to accurately classify high-dimensional sets from a finite number of data points.

This paper describes a simple kd-tree search algorithm that selectively increases search resolution along subsets of the space of starting conditions. The kd-tree search algorithm is a selective search algorithm used to explore only those regions that are deemed "interesting". This paper also describes two fitness functions that define whether a region is worth exploring. One looks for the boundaries between regions with differing end conditions (impact, escape, unknown); the other looks to see if the count of close passes to bodies exceeds a threshold. The fitness function is used by the kd-tree algorithm to selectively explore regions. Combined, they represent the first iteration of the intelligent search algorithm. These algorithms are applied to a single case in a restricted three-body system, and the resulting space partition is examined in detail. I then discuss possible variations of the search and fitness functions, and how these initial explorations will affect future research.

Algorithm 1 kd-tree search algorithm

```
levels ← ∅
sc ← Grid of starting positions and velocities
for i = level → maxlevels do
  for all s ∈ sc do
    s ← ⟨s, Trajectory(s)⟩
  end for
  newsc ← ∅
  for all s ∈ sc do
    nei ← NeighborsOf(s)
    if Fitness(s, nei) → True then
      newsc = newsc ∪ SubdivisionOf(s, nei)
    end if
  end for
  levels ← levels ∪ sc
  sc ← newsc
end for
return levels
```

2 Algorithm Descriptions

This section will describe in detail the search algorithm that selectively explores the reachability set, and the two fitness functions used by the search algorithm to determine which regions require more exploration.

2.1 Kd-tree Search Algorithm

I devised a kd-tree search algorithm [2] to efficiently explore reachability sets. This algorithm was easy to code, and thus was able to give me early feedback on the intelligent exploration approach.

The algorithm currently assumes a grid layout of starting trajectories is used. The grid provides an easy and consistent way to check groups of neighboring trajectories by looking at their indices. Subdivision of the grid is also simple and allows subdivided points to share the same indexing notation for future neighbor groupings. The grid corresponds to ΔV along the velocity axes.

A high-level description of the algorithm is in Algorithm 1. The algorithm runs passes on an several sets of starting points. Each pass creates a new "level", in which all the points have the same resolution and

share the same indexing notation. No levels share points. The interesting regions from current level are used to define the points examined in the next level. The algorithm terminates after a defined number of levels have been explored, and returns the set of *levels* explored by the algorithm.

The variable *sc* refers to the current level (set of trajectories), and *s* refers to a single trajectory in *sc*. *nei* is a set of points in *sc* that are neighbors to *s*, and *newsc* is a temporary variable for the next level.

$Trajectory$ runs a single starting condition and returns the full trajectory. It implements a Runge-Kutta 4th-order integrator of this system, with a timestep size of $0.001s$. The trajectory terminates either upon impact, escape, or after a set amount of time.

The $NeighborsOf$ function returns the points in *sc* that, together with *s*, constitute a 2x2x...x2 hypercube with *s* having the minimal index. In 2D space, the neighbers are a 2x2 square with *s* in the lower left corner.

The $Fitness$ function returns true if a set of trajectories warrants more investigation on the next level. Currently, this is either the end result or minima function.

$SubdivisionOf$ returns a 2x2x...x2 grid of points for each point in *s* and *nei*. The $SubdivisionOf$ function implements the kd-tree, in which the subdivided points are the children of a point (either *s* or a point in *nei*). Each point in the subdivided space has one of its vector elements varied by $\pm d/4$, in which d is the distance between adjacent points on the parent level.

I modularized the algorithm implementation to allow for the easy swapping of subroutines. The functions $Trajectory$, $NeighborsOf$, $Fitness$, and $SubdivisionOf$ can be modified without altering the search algorithm.

2.2 End Result Fitness Function

The End Result fitness function looks for differences in end conditions between groups. These differences point to a boundary between regions that impact a body, escape, or terminate without resolution. For any trajectory that impacts a body of finite size, there is a set of nearby trajectories that also hit the body in different places. By looking for boundaries, large regions with similar end results (such as all impacting a single body) will not be explored in more detail than necessary.

Algorithm 2 End result fitness function

```
sq ← s ∪ nei
if ∃s1, s2 ∈ sq | EndResult(s1) ≠ EndResult(s2)
then
  return True
else
  return False
end if
```

Algorithm 2 provides a high-level description of the end result fitness function. This function looks for boundaries between end results. An end result can be an impact with any body in the system, an escape, or an undecided result for a finite time. The boundary indicates regions of interest by excluding regions that share similar results. This algorithm mimics an edge-detect algorithm found in many image-editing software packages.

The *EndResult* function determines impacts by checking to see if the trajectory is within the radius of either body. The trajectory is considered escaped if the Keplerian energy $E = (v + w \times r) \cdot (v + w \times r)/2 - 1/|r| > 0$. The check approximates a single point mass at the origin, in which v is the velocity, w is the rotational velocity of the frame, and r is the distance to the origin. The escape check is performed at greater than 10 units; at this distance and further, the approximation is valid.

My hypothesis is that in chaotic gravitational systems, the boundary between end result regions is a fractal. If so, there exist regions in the state space in which arbitrarily small perturbations in the application of ΔV may produce differing end results. I explore this in Section 5.

2.3 Minima Count Fitness Function

The Minima Count fitness function identifies highly-curved trajectories as regions of interest. A signature of curved trajectories is the number of times

Algorithm 3 Minima count fitness function

```
sq ← s ∪ nei
for all pt ∈ sq do
  m ← Count of all minima of trajectory of pt relative to each body.
  if m ≥ threshold then
    return True
  else
    return False
  end if
end for
```

the trajectory has a distance minimum with respect to a body. The minima count is a simple metric that measures the curvature of space. This fitness function eliminates regions of space in which trajectories are straight and/or terminate quickly.

Algorithm 3 provides a high-level description of the minima count fitness function. In this function, minima can be considered synonymous with periapses.

I hypothesize that this function is useful in locating stable trajectories that do not impact a body.

3 System Description

As a test case for this approach, I used the well-known 2D Circular Restricted Three-Body System [8]. This system uses a frame of reference positioned at the center of mass of the system, rotating at the angular velocity at which the two major bodies orbit one another. The mass of the third body (the spacecraft) has negligible influence on the other two bodies. The system is represented by Equations 1-4. The variables are non-dimensionalized; the distance between bodies is 1, and the sum of the masses is 1. Although I use seconds to denote time, it too is non-dimensional.

$$x' = v_x \tag{1}$$

$$y' = v_y \tag{2}$$

$$v'_x = -\frac{\mu(\mu+x-1)}{((\mu+x-1)^2+y^2)^{3/2}} - \frac{(1-\mu)(\mu+x)}{((\mu+x)^2+y^2)^{3/2}} + 2v_y + x \tag{3}$$

$$v'_y = -\frac{\mu y}{((\mu+x-1)^2+y^2)^{3/2}} - \frac{(1-\mu)y}{((\mu+x)^2+y^2)^{3/2}} - 2v_x + y \tag{4}$$

$$\mu = \frac{m_1}{m_1+m_1} \tag{5}$$

For all experiments, I set the mass of the smaller body $\mu = 0.2$. Smaller μ values reduce the smaller body's influence, asymptotically approaching the integrable Kepler equations of motion.

Because the reachability set is complete set of locations that can be reached by burning fuel from a starting point, it was appropriate to use a single starting position for all tests. I chose a point between the two bodies and on the axis: $x_0 = 0.5, y_0 = 0$. For determining impacts, I set the radius of each body to be 0.1 unit.

I varied only v_{x0} and v_{y0}. As initial speed increases to ∞, gravitational influence on the trajectory curve goes to 0 in the time the body remains in the system, and thus the trajectory will either impact one of the two bodies on a straight line (outer edges of Figure 1), or will escape the system. Thus, the focus of the algorithm should be in a bounded region of initial velocity space.

4 Experiments

Starting with the initial position given in Section 3, I examined the initial velocity space to locate the expected fractal region. I set the initial grid to be a 10x10 grid of points in (v_x, v_y) space, ranging from -2.5 to 2.5 on both axes. This region of starting velocities contains in its entirety a complicated set of boundaries. Using the end result fitness function, the starting level of 10x10 points can be seen in the left panel of Figure 1. This produced a grid of 100 points, which are colored as squares according to the end result fitness function (Algorithm 2). All 2x2 squares with differing end results were subdivided into 4 points per point until seven levels were produced. The kd-tree search algorithm examined 77028 points in total. If every point on every one of the seven levels were examined, it would total 546100. Thus, the kd-tree search algorithm and the end re-

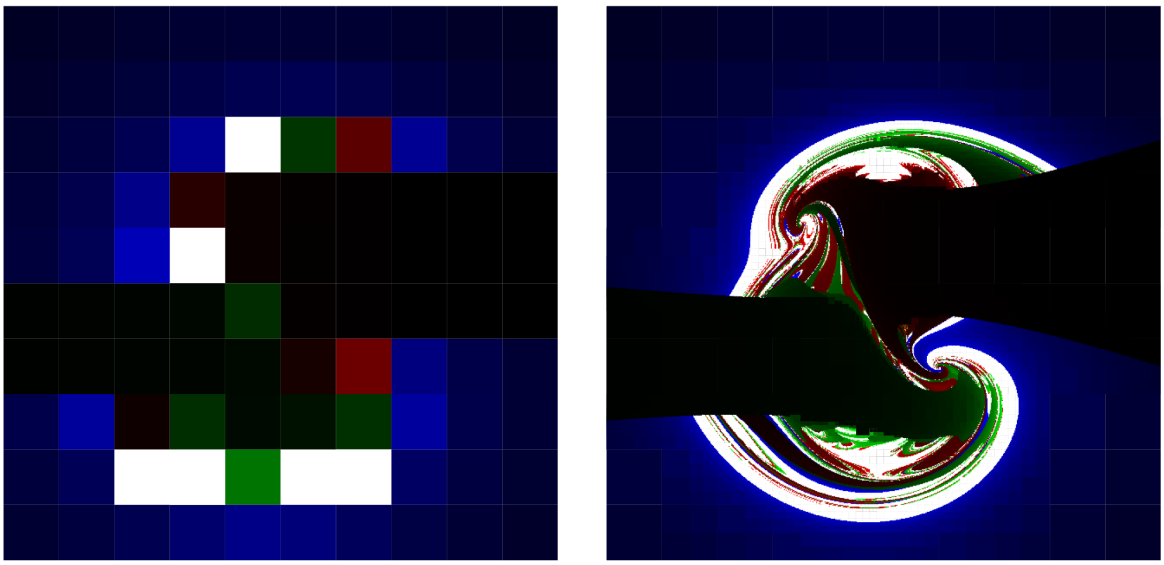


Figure 1: The initial velocity end result space after one pass (left) and seven passes (right) at 20 seconds each, ranging from -2.5 to 2.5 on both the X-axis (v_x) and the Y-axis (v_y). Blue indicates an escaped trajectory, green indicates an impact on the more massive left body, and red indicates an impact on the less massive right body. The intensity of the color represents the time taken to record an end result. White indicates that after 20 seconds, the trajectory did not impact or escape. The large green-black region on the left is the left body impact zone; likewise, the large red-black region on the right is the right body impact zone.

sult fitness function only examined 14.1 percent of the possible search space. The selective nature of the algorithm can be seen in Figure 1 on the outside edges; several contiguous escape (blue) regions were not examined in further detail on future passes. Thus, the algorithm focused only on the border regions near the center of the image. While I did not examine the same region with the minima count fitness function, I used this overall image to determine regions that I could then explore in more detail with both fitness functions.

4.1 Closer Examinations

The overall characterization of the initial velocity space revealed regions of interest. I chose to examine two regions in Figure 1.

One such exploration focused on a large contiguous white region centered around the starting velocity $(0, 1.15)$ units/s. Figure 2 zooms in on this region. The white region indicates that the trajectories did not impact or escape, and thus are likely highly-curved. Thus, I chose to explore with the minima count fitness function, deciding that a minimum total of 40 minima required more exploration. I felt 40 was a good number to isolate this region from surrounding areas. This choice is reflected in the granularity of the images. The lower center region has a multitude of minima for both bodies, as indicated by the yellow hue.

The other exploration focused on the spiral region near the starting velocity of $(0.56, -0.84)$ units/s, as shown in Figure 3. This region warranted more interest because it contains a spiral. A never-ending

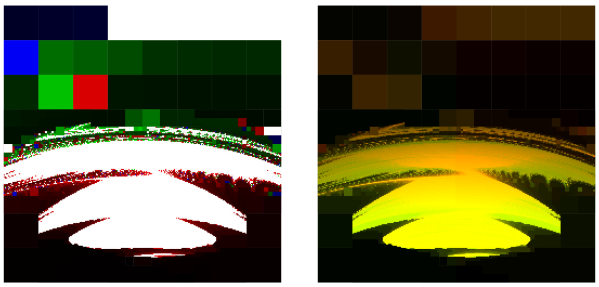


Figure 2: A region in the vicinity of $(0, 1.151)$ units/s, with each point simulated for 200 seconds. The left panel has the same end result coloring scheme described in Figure 1. The right panel uses the following color scheme: the green intensity is the ratio of left-body minima to 100, and the red intensity is the ratio of right-body minima to 100. Nearly equal combinations of red and green are shown as yellow. Lower left corner at $(-0.34, 0.96)$, upper left corner at $(0.34, 1.64)$

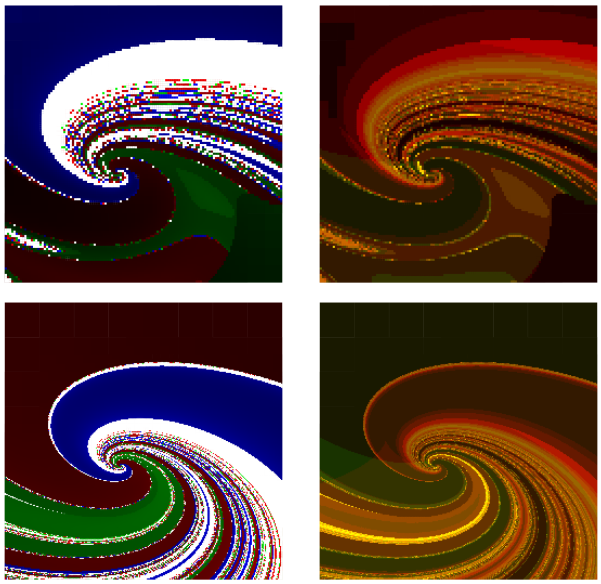


Figure 3: Top left: a close look at the spiral, with the end result coloring scheme. Top right: the minima colorization. Bottom left: a closer look at the spiral, with the end result coloring scheme. Bottom right: the minima colorization. Top row: lower left corner at $(0.37, -1.02)$, upper right corner at $(0.83, -0.57)$. Bottom row: lower left corner at $(0.5487, -0.8564)$, upper right corner at $(0.5713, -0.8335)$.

spiral is indicative of fractal structure. Here, I chose to zoom in to the spiral twice to see if the spiral existed at small levels.

I also sampled single trajectories from each of the interesting regions to have a visual representation of the trajectory in addition to its end result or minima count. The descriptions of these, along with comments on the patterns found within the fractal, can be found in Section 5.

5 Results

5.1 Boundary Description

Contiguous impact regions are a projection of the surface of the body to the velocity space. A trajectory that impacts a planet is surrounded in the initial state space by arbitrarily close trajectories that impact arbitrarily closely to the original impact point. The boundary of this region is formed when a trajectory barely misses a planet, and thus its fate is not easily determined without further simulating the trajectory. The boundary is similar for escapes; the boundary is defined by trajectories that have a Keplerian Energy of 0. If each body were a point mass without volume, nearly every starting state would either escape or travel in a never-ending orbit of some sort. The probability of a trajectory falling exactly on a point mass is (likely) zero. Thus, I conjecture that non-point bodies are required for impact boundaries to exist in the state space. I further conjecture that the boundary between trajectories that have an arbitrarily long path, in which the end result may differ for arbitrarily small distances between starting points on nearby trajectories, is fractal.

5.2 Smooth and Fractal-like Regions

Figure 1 shows a compelling fractal-like structure of end result regions. As expected, large starting velocities tend to escape or impact quickly. However, the right panel reveals highly detailed structures. Visible within the image are swirling spirals, contiguous colored regions folded upon themselves, and interesting boundaries between colored regions. The image is not symmetric because neither the system nor the placement of the starting position is symmetric (except that the starting position is on the line between centers of mass). However, symmetry is hinted at in many places. Two major spirals can be found, roughly symmetric to the velocity origin (the lower right one is shown in Figure 3). Spirals are indicative of regions in which sensitivity on initial conditions is a major influence on the end result. A large green curved band stems from the right impact zone and tends to the upper left spiral; a red curved band similarly stems from the left impact zone and goes toward the lower left spiral. Also, large white regions exist both directly above the origin (as shown in Figure 2), and below the origin.

Several smooth boundaries can be found between end result regions, especially regions that border the major impact zones. I hypothesize that these smooth boundaries are the regions in which a spacecraft barely avoids impacting one planet, but soon impacts the other. The smoothness indicates that, at least in these areas, the end-result region boundaries are not fractal.

The major spirals in Figures 1 and 3 do seem to be fractal in nature. The successive zoomed images provided in Figure 3 show remarkable self-similarity. This spiral contains large contiguous bands, separated by white bands dotted with all three end results. I suspect that as I let the simulation time run to ∞, the white bands would shrink and be replaced by ever-smaller contiguous colored regions. I believe that these bands in the spiral contain an infinite number of projected images of each planet. For the purposes of space mission planning, fractal regions should be avoided if the spacecraft has a defined goal; the uncertainty in the starting position and the applied ΔV are more than enough to ensure an unpredictable fate.

5.3 Boundary Fractal Dimension

$$D = \lim_{\epsilon \to 0} \frac{\log N(\epsilon)}{\log \frac{1}{\epsilon}} \tag{6}$$

The fractal dimension of the boundary can be estimated by counting the number of grid points that share a boundary. If the boundary is truly a fractal, the fractal dimension should be in the set $(1, 2)$ in the 2D space of starting velocities. The boundary does not fall conveniently on single points, however, so the boundary may exist on any or all of the points from each subdivided parent point that contains the boundary. The seventh pass of the full fractal image contains 49864 points, each a square of side length $\epsilon = 5/(9 * 2^6) = 0.00868$. The full grid has a length of $5.\bar{5}$ when considering the outer edges of the squares. Normalizing the length by this factor results in $\epsilon = 0.0015625$. Because 49864 points assumes each of the four points of a subdivided sixth-level point has a boundary, I divide by four to provide the lower bound. Following Equation 6, the fractal dimension is therefore likely in the set $[1.460, 1.674]$. Because I performed only seven subdivisions, and the white spaces are included, this range only gives a broad idea of the true value. However, if the dimension is shown to be non-integer on very small scales, the boundary is likely a fractal.

5.4 Spiral Sample Trajectories

Sensitive dependence on initial conditions is very important in the spiral regions. A sampling of points from around the spiral reveal that highly precise starting velocities lead to divergent end conditions after a short amount of time (under 20 seconds). Figure 4 shows a sampling of points taken from multicolored points embedded in one of the white bands on the spiral. Long and complicated paths end up diverging significantly after a certain amount of time. I conjecture that as trajectories close in on the spiral center, the paths taken by nearby trajectories remain similar for very large time spans, orbiting both planets in a large circular orbit for that entire time, but must eventually diverge in their end condition.

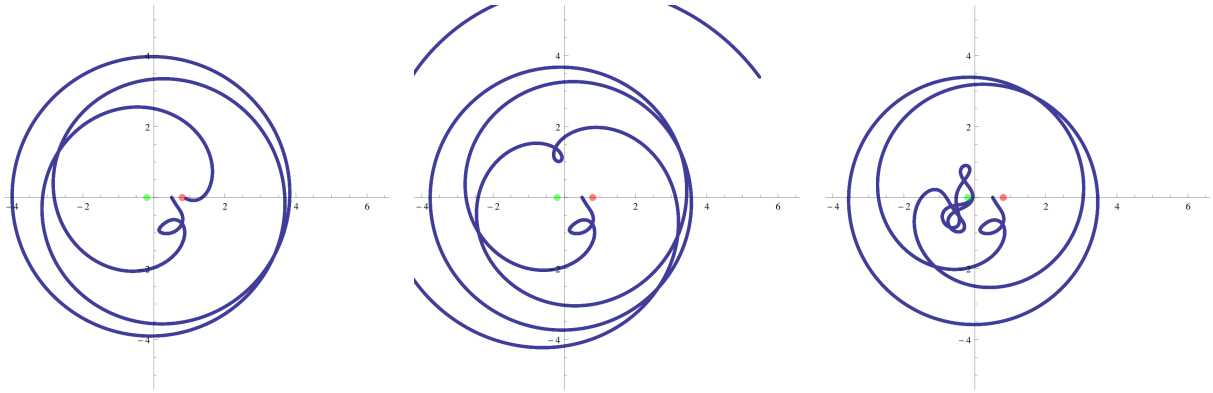


Figure 4: Left: a trajectory near the center of the spiral region in Figure 3, with starting velocity equal to $(0.5573, -0.8446)$, impacts the right planet. Middle: the trajectory that results when v_{x0} is increased by 0.00015, which appears to escape. Right: another increase in v_{x0} of 0.00015 results in an impact on the left body.

5.5 Stable Orbits

The region in Figure 2 is anomalous; it is not determined entirely by the flow of the swirling boundaries around it, so in a sense it stands out. The large amount of yellow indicates that the orbits in this region are stable over the time frame, with multiple minima with respect to both bodies. A random sampling of a point in this region shows an apparently stable orbit (Figure 5). It may be a limit cycle or a quasi-periodic orbit, which may eventually impact or escape after an extended amount of time, or it may be a periodic orbit that is poorly approximated by the Runge-Kutta integrator. I expect that any starting velocity in the region is quasi-stable. This indicates that regions like the one in Figure 2 are desirable for a high level of stability in an orbit around one body. Determining if this region is dense in stable orbits remains to be done. If true, then spacecraft are able to enter a stable orbit without requiring infinite precision on their thrusters, and such velocity regions can be quickly found.

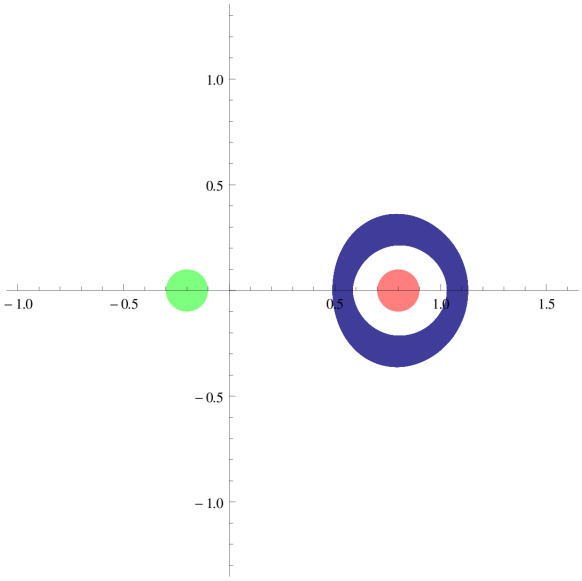


Figure 5: The trajectory with a starting velocity of $(0, 1.151)$. This point was embedded in the lower central yellow region shown in Figure 2.

6 Future Work

The results generated by the kd-tree search algorithm and the fitness functions reveal complicated behavior

and interesting features in the starting velocity space of a reduced three body system. However, there are multiple avenues for improving this approach.

Dimensionality could be increased at the cost of speed. All results included in this document referenced only a single starting position. Other starting points may resemble the included fractals, or may be completely different. Efficient AI techniques are required to examine higher-dimensional spaces in a practical amount of time. The 14.1 percent efficiency of the kd-tree search algorithm is good, but can be improved. Candidate approaches include swarm intelligence [3], in which each data point is a member of the swarm. Genetic algorithms [4] may also be used to evolve a group of points to boundaries. On the other hand, these approachs may not be well-suited for fractal spaces, so a new approach based on these ideas may be necessary.

The methods of Zhao, et al. ([1], [9], [11], [12]), and Robins, et al. ([5], [6], [7]), are advanced clustering techniques that could improve the performance of the search algorithm. These algorithms could be implemented with a fitness function to determine which boundaries should be explored, which regions include stable orbits, and which regions a spacecraft should avoid. For example, for a spacecraft to avoid hitting a body, it could cluster all impact trajectories together and remove them from the valid set of trajectories.

Additional fitness functions may lead to better qualitative analysis of regions. In addition to end result and minima, functions could examine:

- Lyapunov exponents along the length of a trajectory. This may lead to a better analysis of chaotic behavior at the expense of more computations.
- The curvature of a trajectory. This could be a better indicator of curvature than the minima count. Perhaps the curvature calculation could be used in conjunction with minima count.
- The time to impact or escape. This might be important for fast decisions on which regions to avoid.
- Periodicity. This can locate stable orbits in space. Stable orbits can affod spacecraft more time to plan ahead, in addition to being useful for studying a single body.
- Close passes to specific bodies. This could be useful in locating regions in which to perform a gravity slingshot maneuver.

Finally, a suitable integrator is required to give accurate results; the Runge-Kutta integrator does not maintain invariance of total energy. I will use a symplectic integrator [10] for future work.

7 Conclusion

I have defined a simple algorithm that can locate boundaries between regions with different properties (end results and minima included here). I used my algorithm to characterize the velocity space of a single position relative to a restricted three-body system, and has revealed interesting trends in the regions. While I did not prove it, I show that the region has fractal qualities. I also determined that specific regions of the velocity space lead to more unpredictable behavior than other regions. Finally, I described future approaches that will improve the method in numerous ways. I believe this research has the potential to give autonomous spacecraft the quick and accurate means to plot their own missions.

References

[1] C. Bailey-Kellogg, F. Zhao, and K. Yip. Spatial aggregation: Language and applications. In *Proceedings AAAI-94*, pages 517–522, 1994.

[2] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18:509–517, September 1975.

[3] E. Bonabeau, M. Dorigo, and G. Theraulaz. *Swarm Intelligence: From Natural to Artificial Systems.* Oxford University Press, 1999.

[4] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley Professional, 1989.

[5] V Robins, J D Meiss, and E Bradley. Computing connectedness: An exercise in computational topology. *Nonlinearity*, 11(4):913, 1998.

[6] V. Robins, J. D. Meiss, and E. Bradley. Computing connectedness: disconnectedness and discreteness. *Physica D: Nonlinear Phenomena*, 139(3-4):276 – 300, 2000.

[7] V. Robins, J. Abernethy, N. Rooney, and E. Bradley. Topology and intelligent data analysis. *Intelligent Data Analysis*, 8:505–515, October 2004.

[8] S. D. Ross and D. J. Scheeres. Multiple Gravity Assists, Capture, and Escape in the Restricted Three-Body Problem. *SIAM Journal on Applied Dynamical Systems*, 6:576–596, January 2007.

[9] K Yip and F Zhao. Spatial aggregation: Theory and applications. *CoRR*, cs.AI/9608103, 1996.

[10] H. Yoshida. Construction of higher order symplectic integrators. *Physics Letters A*, 150(5-7): 262 – 268, 1990.

[11] F. Zhao. Extracting and representing qualitative behaviors of complex systems in phase space. *Artificial Intelligence*, 69(1-2):51 – 92, 1994.

[12] F. Zhao and MIT-AIL. Automatic analysis and synthesis of controllers for dynamical systems based on phase-space knowledge. *PhD Thesis*, 1992.

Applications of Chaos in Cyptography

Donny Warbritton

May 12, 2011

1 Chaos in Cryptography

For years [2] researchers have attempted to combine chaotic dynamics with cryptography in such a way that the resulting system is on par with modern cryptographic implementations. The characteristics of chaotic systems (sensitive dependence on initial conditions and similarity to random behavior, in particular) seem to make them a prime candidate for obfuscating information in such a way that the result appears random, and similar input data bears little ultimate relation. Despite what appear to be distinguishing advantages of applying chaos to cryptography, we yet await a system that is both sufficiently secure and aptly brisk to compete with current systems [4]. Nonetheless, clever systems that integrate cryptography and chaos exist [2], and their complexity make for interesting analysis.

Chaotic dynamics and cryptography have been historically mismatched, oftentimes due to their unrelated research goals. Because of this, many chaos-based cryptographic systems have been proposed that do not follow best practices laid out in cryptography [3]. On top of this, many of these systems are not only weak, but slow as well. In part, this is due to the computationally expensive multiplications and divisions with floating-point numbers that are common in the creation of chaotic trajectories.

In [2], the authors present both a chaotic Feistel cipher and a chaotic uniform cipher. Both ciphers were designed with cryptographic principles in mind, including resistance to both linear and differential cryptanalysis. The ciphers they describe are largely based on techniques common to traditional ciphers, and they attempt to involve chaos at specific points where it is most warranted, rather than trying to force a strong cryptographic system out of an experiment in chaos. While the authors admit that the cipher is still lacking in some areas, and the link between chaos and cryptography is occasionally tenuous, the attempt is still impressive academically.

Because of the complexity of a modern cipher, chaotic or otherwise, attempts to analyze them will be abandoned in favor of a system that can be presented and discussed within the limits of this paper.

2 Deconstructing a Chaotic Cryptosystem

In order to discuss a chaotic cryptographic system that does not require an advanced knowledge of cryptography, consider the scheme proposed in [1]. As described, this system consists

of two similar rounds, relies on the Lorenz system for pseudo-random data, and has a key that is the concatenation of the initial conditions that uniquely describe the two Lorenz trajectories used in the cipher. (As an aside, the original description of the system is lacking detail in several areas, but it appears unlikely that there exists an interpretation significantly more cryptographically secure than the one described in this paper.)

2.1 Algorithm

For each round of encryption, it is necessary to generate a Lorenz trajectory whose length corresponds to the number of bytes in the plain text. To generate a unique Lorenz trajectory, the XYZ coordinates, along with the Rayleigh coefficient must be specified (σ,β , the time step, and the initial time are all fixed). For each point in the orbit, the XYZ coordinates (treated as double-precision floating point numbers conforming to IEEE 754) are extracted to form a 24-byte binary value.

In the first round, the plain text is stepped through 24-bytes at a time, and the intermediate cipher text is generated by performing a binary XOR on the plain text with the corresponding binary value derived from the chaotic orbit. The second round acts on the intermediate cipher text in a similar manner to the first round. A new trajectory is generated with different initial conditions and the XOR operation is applied as before. Additionally, after each 24-byte chunk has been acted on, it is swapped with a 24 byte chunk somewhere else in the intermediate cipher text. To perform the swap step, the current coordinates of the trajectory (the author does not describe how they should be combined, or if all three should be used) are converted to integers, modded by the length of the input, and swapped.

2.2 Cryptanalysis

In order to simplify the analysis, consider each round individually. In particular, if it can be shown that a round does not contribute to the strength of the cipher, then the remaining round is the sum of the strength. As the first round is the simpler of the two, we will consider it first.

Let us assume that the key space (set of all possible initial conditions) is too expansive to search exhaustively. Then we are left with either strictly the cipher text, or some portion of the plain text and the cipher text that was emitted for a given unknown key.The later case is trivially true if we are allowed to encrypt arbitrary plain text with the secret key, and may be true if we are good at guessing.

Because the plain text is encrypted in blocks of 24 bytes using a unique point on the attractor, by knowing a small portion of the original text and its corresponding cipher text, we can determine an initial condition for this trajectory. If this is not at the beginning of the file, we would simply solve the differential equations for Lorenz with a decreasing time variable until the beginning of the file is reached. We have neglected to mention the Rayleigh coefficient here, but the key space has been so dramatically decreased at this point that it could now be obtained by brute force.

While it appears that the first round could be reversed with a reasonably short amount of information, the second round is more complicated. The swap means that, not only is the

text out of order, but also that different pieces may be XOR'd different numbers of times. Firstly, the shuffle that is used is a poorly implemented Fisher-Yates shuffle. In this case it means that not all permutations of the intermediate cipher are equally probable. Not only that, but converting the floating point values used in the trajectory to integer values means that the space a new swap value can be chosen from isn't very large, and there is a clear bias for some areas of the cube that encompasses the attractor to be chosen more frequently than others. This small key space and poor source of random data, along with the implementation problem, means that reversing the swap may be possible without having any knowledge of the plain text.

If a chosen plain text attack is possible, then discovering the mapping between initial and final positions in the map is trivial: starting with some chosen plain text, perform the encryption and save the result. Next, modify the plain text by one by and encrypt the message again. Because this cipher has no diffusion (the effect of a single plain text bit spreading through the entire cipher), the resulting cipher text will be identical to the first except for a single block. This process can be repeated as many times as necessary to develop a complete mapping. Once again, the key space has been dramatically reduced, and it is reasonable to expect that a brute for search could uncover the desired sequence. This process reveals useful information about the second half of the key - enough to fully reverse the second round.

2.3 Practical Considerations

Due to the implementation of the swapping step in the second round, any portion of the intermediate cipher may need to be accessed throughout the encryption process. This is trivial if the text to be encrypted is small (say, smaller than the size of the cache); however, as the plain text increased, the time required to encrypt it may increase drastically. In particular, if the plain text is larger than the size of the cache, then parts of it will need to be retrieved from main memory, a process on the order of one to two orders of magnitude slower than accessing the cache. Even worse, if the file cannot be fully contained in main memory, random seeks to disk will be required. This could quickly become so prohibitively slow as to appear that it will never finish.

2.4 Conclusion

Although the author is in no way qualified to address serious cryptographic protocols, it is his opinion that this particular cipher is not suited to protecting sensitive information. Although the flaws in both rounds have not yet been combined in such a way that either the plain text or the secret key may be elicited, it appears likely that such a goal may be obtained. It is possible to induce the cipher to reveal information about the secret key being used such that the space in which it lives can be narrowed significantly with relatively little effort on the part of the attacker. Diffusion in the cipher, a property necessary for any strong cryptographic system, seems to be non-existent in this system, leading the author to believe that differential cryptanalysis would be a ripe avenue for attack. Furthermore, the practical considerations discussed above mean that encryption of large files may be unreasonably time consuming anyway.

References

[1] M. Brunel, "A Lorenzian based chaotic encryption scheme" in *Projects in Chaotic Dynamics: Spring 2010.*

[2] Naoki Masuda, Goce Jakimoski, Kazuyuki Aihara, Ljupco Kocarev, "Chaotic Block Ciphers: From Theory to Practical Algorithms, *IEEE Transactions on Circuits and Systems - I: Regular Papers, Vol. 53, No. 6, June 2006.*

[3] Ljupčo Kocarev, "Chaos-Based Cryptography: A Brief Overview, *Circuits and Systems Magazine, IEEE*

[4] J.M. Amigó, J. Szczepanski, L. Kocarev, "Theory and Practice of Chaotic Cryptography", *Physics Letters A, Volume 366, Issue 3, 25 June 2007, Pages 211-216*

[5] Ruming Yin, Jian Yuan, Qiuhua Yang, Xiuming Shan, Xiqin Wang, "Linear cryptanalysis for a chaos-based stream cipher, *World Academy of Science, Engineering and Technology 60, 2009*

[6] Jay Kominek, personal communication, April 2011