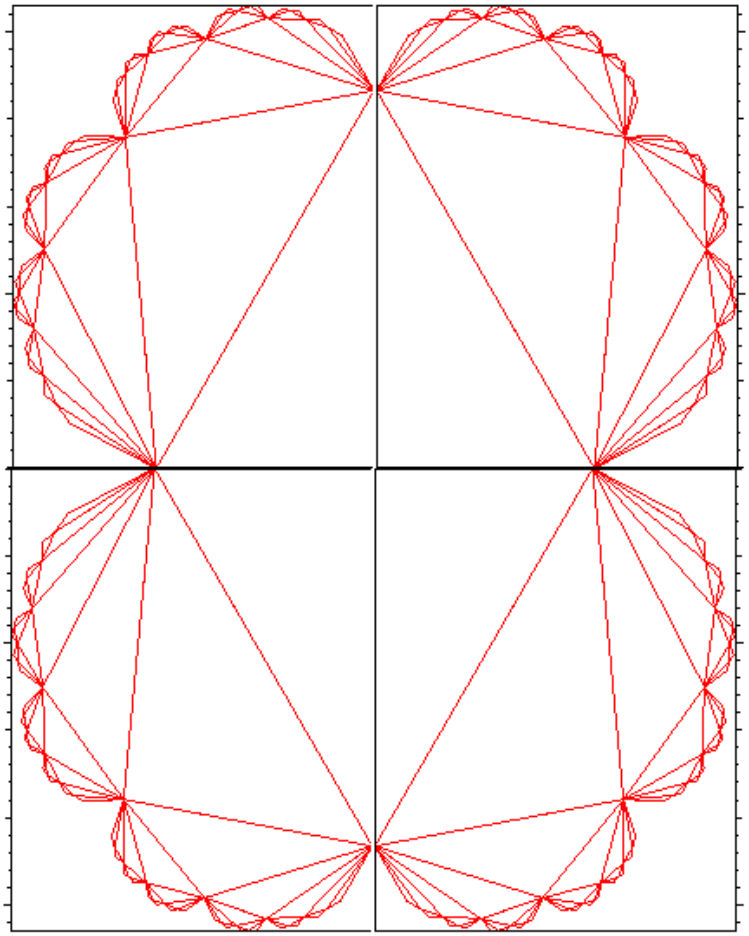
The Search for Fractal Area

Undergraduate Honors Program Research Project



Daniel W. Bruhn
Mentor: Dr. John Franke
NC State University
Summer 2002

This research was conducted at NC State University in the summer of 2002 by the student Daniel Bruhn, under the supervision of Dr. John Franke, a professor in the Department of Mathematics. The research was successful in determining the area of the Julia Set for the dynamical system $Z_{n+1} = Z_n^2 + Z_n$, within a range of ~0.023. Currently, the upper bound stands at 2.94932944974534, the lower bound at 2.926713564103740.

This research began as an investigation into computing the area of the Mandelbrot Set (Figure 1). The initial idea was to derive equations for numerous lemniscates and integrate them, resulting in area values that gradually approached that of the Mandelbrot Set (Figure 2). However, the nature of the Mandelbrot function $Z_{n+1} = Z_n^2 + C$ complicated matters for Maple (the mathematical analysis software the researchers used). Granted, it was a simple matter to generate implicit *xy* forms of the lemniscate equations, but a form that could be integrated was needed. Maple was unable to solve the lemniscate equations explicitly, and a method to generate parametric and/or complex-parameterized forms of the lemniscates failed as well.

It was during this difficulty that discussion of a pseudo-Mandelbrot Set arose. Whereas the M-Set arises from complex iterations of the function $Z_{n+1} = Z_n^2 + C$, the "Dynamical Mandelbrot Set" (the researchers' coined name) is based on $Z_{n+1} = Z_n^2 + Z_n$. The DM-Set (Figure 3) is in reality not a Mandelbrot Set, but rather a Julia Set of the dynamical system defined by $Z_{n+1} = Z_n^2 + Z_n$. It was conjectured and verified that the nature of this equation would make the DM-Set more conducive to lemniscate and area calculation. Thus, the Mandelbrot Set was abandoned in favor of a Julia Set of similar origin.

The first step in determining the area of the DM-Set was to use Maple to generate complex-parameterized equations for the lemniscates. This was initially accomplished by a logical approach involving the reverse-mapping relationship between successive lemniscates:

> *We know the parametric equation for lem0* (the outermost lemniscate)*: <2 cos(t), 2 sin(t)>. We know that* $W=Z^2+Z$ *maps points from lem1* (the next lemniscate in the sequence) *to lem0. Solve W in terms of Z, then plug in parametric form of lem0 to get param. eqn. of lem1.*
> --Log, 5/24/02

This method successfully yielded several lemniscate equations. (Computation 1 shows the actual Maple output.) A pattern in the lemniscate equations was clearly evident, thus leading the researchers to develop a succinct pair of recursive formulas for quick and painless generation of the lemniscates (Computation 2). With these formulas, it was possible to generate functions for as many lemniscates as needed. The next step in the process was to devise a method of integrating these equations. Their complex nature proved an obstacle, as there is no definitive method of complex integration. After some attempts to convert the equations into an awkward $re^{\theta i}$ form (which would allow for separation of the real and imaginary components and thus integration using Green's Theorem), Dr. Franke was able to derive a formula for complex integration. His formula uses Green's Theorem, modified for complex functions using complex conjugates (Computation 3).

With this formula secured, three choices for integration lay before the researchers:

> *We can either use the conjugate integration formula* (Computation 3) *with the original complex lemniscate equations, use that same formula with the e-complex-parameterized versions* (a rather awkward form)*, or use Green's Theorem on the x(t),y(t) parameterized versions obtained from the e-parameterized ones* (an awkward use of the awkward form).

Applying the conjugate integration formula to the original complex lemniscate equations was the simplest route, and thus it was chosen as the preferred method of integration. Yet there was one caveat:

…however, all 3 methods generate Maple errors when integrated from -Pi to Pi.
--Log, 5/30/02

This problem was circumvented with the introduction of Simpson's method to approximate the integrals. As it turns out, Maple could not handle the level of recursion needed for exact integration, and cooperated better with a numerical analysis method. To counteract the loss in accuracy generated by moving to a numerical method, the researchers used 15 digit precision and 200 subintervals when utilizing Simpson's method. An optimized Simpson script was then written and yielded several lemniscate area values after a 1-2 day run (Computation 4). This sequence of upper-bound area values, however, seemed to converge very slowly. Thus, in order to gain a fuller picture of the area, the researchers developed a similar method to generate a sequence of lower-bound area values.

The process for lower-bound generation involved starting with a fabricated lemniscate inside the set. An ellipse was chosen, whose major and minor axes corresponded to the distances between the "spikes" in the set (Figure 4). The next "lemniscate" was calculated by determining what points were mapped onto the ellipse under the first iteration of $Z_{n+1} = Z_n^2 + Z_n$ (simply a matter of reverse-iteration). Thus, the second lemniscate consisted of the set of all points that mapped onto the ellipse under the first iteration, the third of all points that mapped onto the ellipse under the second iteration, and so forth. These lemniscates successively grew to fit the Dynamical Mandelbrot Set from the inside, and consequently provided a sequence of lower-bound areas when integrated with Simpson's method (Computation 5). However, it was apparent that this "Lower Simpson" method converged no faster than the Upper Simpson. More precise upper and lower bounds than those of 2.879 and 2.925, respectively, were desired. Attempts to plot the data sets and fit curves to them resulted in impossible limits, such as 1.397,

3.0056, and 3.0255. Clearly, another method of upper- and lower-bound determination was needed – one that could be computed faster and had a better rate of convergence.

It was during this time that the researchers began investigating the numerous "spike points" of the DM-Set. They soon discovered that generating the values for these unique points was a simple process, given that the spikes mapped to other spikes under each iteration. One needed only the spike located at (0,0), and the remaining spikes could be determined from applying a pattern of reverse iterations to (0,0). Given the ease with which they could calculate these spikes, the researchers soon devised a new method to give an approximate lower-bound for the area of the DM-Set. This method consisted of creating triangles with the largest spikes, then adding on smaller triangles by connecting the smaller spikes (Figure 5). As more triangles were added, the area of this polygonal figure would approach that of the set, and hopefully provide a geometric sequence (Figure 6) from which the exact area could be derived.

This was the most effective method of area calculation, but turned out to be one of the most challenging to implement in Maple. The first step was to generate a large amount of spike points, which involved stepping the first spike (0,0) through a process of reverse iteration. Once all the needed spike points were generated, the symmetry of the DM-Set was applied and the points below the real axis were eliminated (Figure 7). The remaining points were sorted into a list based on their clockwise orientation from the center of the set (-0.5,0). Once this was accomplished, Maple was instructed to create triangle objects and evaluate their areas. An optimized script yielded the area values shown in Computation 6. This method (coined "Lower Poly" for Lower-bound Polygonal) converged much faster than the Lower Simpson method. It also *computed* much faster than the Lower Simpson, calculating several values in a matter of minutes, while the Lower Simpson took hours. (This was due to the simplicity of using

triangles.) An evaluation of these area values, while they did not provide the researchers with a simple geometric sequence, led the researchers to estimate the area of the DM-Set as somewhere in the realm of 2.93.

The researchers had succeeded in devising three methods thus far: an upper-bound Simpson, lower-bound Simpson, and lower-bound polygonal. Determined to finish the job, Daniel Bruhn developed an upper-bound polygonal method, to bring the total number of approaches to four. The Upper Poly approach utilized the appearance of *tangent points* (Figure 8) on the outer lemniscates of the Dynamical Mandelbrot Set. The tangent points could be calculated in the same manner as the spike points, and connecting them provided a similar structure of triangles (Figure 9). By continually increasing the number of tangent points utilized, the Upper Poly script generated a decreasing sequence of area values (Computation 7). This sequence, like the Lower Poly, also suggested a DM-Set area of approximately 2.93, and provided an additional upper bound. This was the last method developed, and due to limitations on computer power, additional runs of the four methods could not produce any more values. Thus, the area of the Dynamical Mandelbrot Set was successfully calculated to lie between 2.92671356410374 and 2.94932944974534.

APPENDICES

Appendix A: Computations

Computation 1:

$$dyn0cpx := 2\cos(t) + 2\,I\sin(t)$$

$$dyn1cpx := -\frac{1}{2} + \frac{1}{2}\sqrt{1 + 8\cos(t) + 8\,I\sin(t)},\ -\frac{1}{2} - \frac{1}{2}\sqrt{1 + 8\cos(t) + 8\,I\sin(t)}$$

$$dyn2cpx := -\frac{1}{2} + \frac{1}{2}\sqrt{-1 + 2\sqrt{1 + 8\cos(t) + 8\,I\sin(t)}},$$
$$-\frac{1}{2} - \frac{1}{2}\sqrt{-1 + 2\sqrt{1 + 8\cos(t) + 8\,I\sin(t)}},$$
$$-\frac{1}{2} + \frac{1}{2}\sqrt{-1 - 2\sqrt{1 + 8\cos(t) + 8\,I\sin(t)}},$$
$$-\frac{1}{2} - \frac{1}{2}\sqrt{-1 - 2\sqrt{1 + 8\cos(t) + 8\,I\sin(t)}}$$

$$dyn3cpx := -\frac{1}{2} + \frac{1}{2}\sqrt{-1 + 2\sqrt{-1 + 2\sqrt{1 + 8\cos(t) + 8\,I\sin(t)}}},$$
$$-\frac{1}{2} - \frac{1}{2}\sqrt{-1 + 2\sqrt{-1 + 2\sqrt{1 + 8\cos(t) + 8\,I\sin(t)}}},$$
$$-\frac{1}{2} + \frac{1}{2}\sqrt{-1 - 2\sqrt{-1 + 2\sqrt{1 + 8\cos(t) + 8\,I\sin(t)}}},$$
$$-\frac{1}{2} - \frac{1}{2}\sqrt{-1 - 2\sqrt{-1 + 2\sqrt{1 + 8\cos(t) + 8\,I\sin(t)}}},$$
$$-\frac{1}{2} + \frac{1}{2}\sqrt{-1 + 2\sqrt{-1 - 2\sqrt{1 + 8\cos(t) + 8\,I\sin(t)}}},$$
$$-\frac{1}{2} - \frac{1}{2}\sqrt{-1 + 2\sqrt{-1 - 2\sqrt{1 + 8\cos(t) + 8\,I\sin(t)}}},$$
$$-\frac{1}{2} + \frac{1}{2}\sqrt{-1 - 2\sqrt{-1 - 2\sqrt{1 + 8\cos(t) + 8\,I\sin(t)}}},$$
$$-\frac{1}{2} - \frac{1}{2}\sqrt{-1 - 2\sqrt{-1 - 2\sqrt{1 + 8\cos(t) + 8\,I\sin(t)}}}$$

Computation 2:

$$s_0 := x \to -\frac{1}{2} + \frac{1}{2}\sqrt{4x + 1}$$

$$s_1 := x \to -\frac{1}{2} - \frac{1}{2}\sqrt{4x + 1}$$

Computation 3:

$$Area = \int_{-\pi}^{\pi} i\left(\frac{Z(t)-\overline{Z}(t)}{2}\right)\left(\frac{\frac{dZ}{dt}-\overline{\left(\frac{dZ}{dt}\right)}}{2}\right)dt$$

Computation 4:

```
12.5663706143592
6.25856923705022
4.39803599839599
3.66261604575748
3.32344637633564
3.15232608588506
 3.0607851771532
 3.0096705975802
2.98012001953313
2.96250823690325
```

Computation 5:

```
1.36034952317566
 1.9137858786494
 2.2853898315865
2.51759612300105
2.66017283751829
2.74833160907172
2.80384385575988
2.83962345236596
2.86327838563603
2.87932532489819
```

Computation 6:

```
 .86602540378444
1.74721940171576
2.28334939378986
2.57246200644142
2.72679382232865
2.81090196269776
2.85803110736268
2.88519607659812
2.90128641160427
2.91107018661681
2.91717230786105
2.92099847125921
2.92352703245995
2.92536716697634
2.92671356410374
```

Computation 7:

```
3.43353654706353
3.29951100747568
3.21258409525696
3.13278509462622
3.07817665258830
3.03475502842711
3.00687376657682
2.98544908642022
2.97141512047291
2.96093717327182
2.95427910078466
2.94932944974534
```

Appendix B: Figures

Figure 1: The Mandelbrot Set

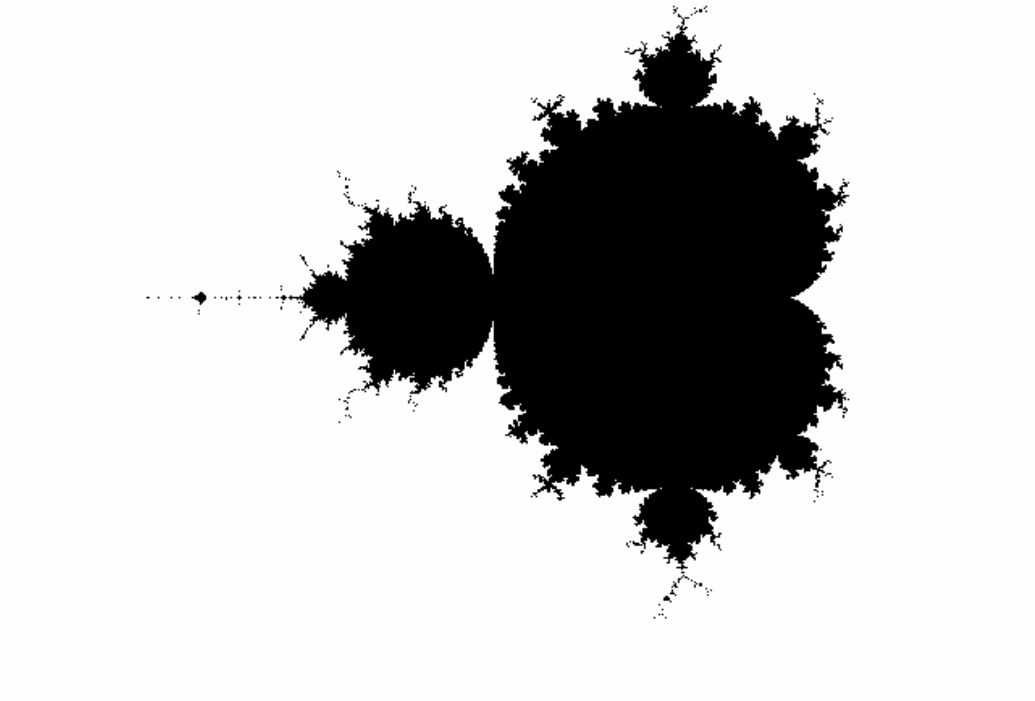


Figure 2: Mandelbrot lemniscates

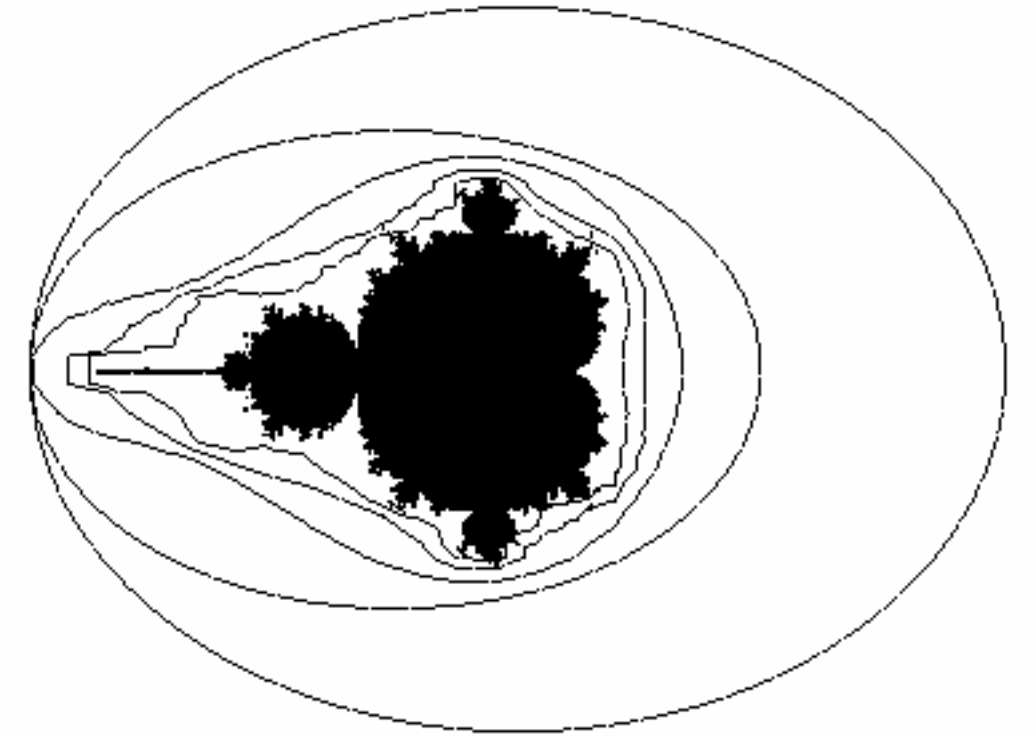


Figure 3: The Dynamical Mandelbrot Set

Figure 4: DM-Set with interior lemniscates

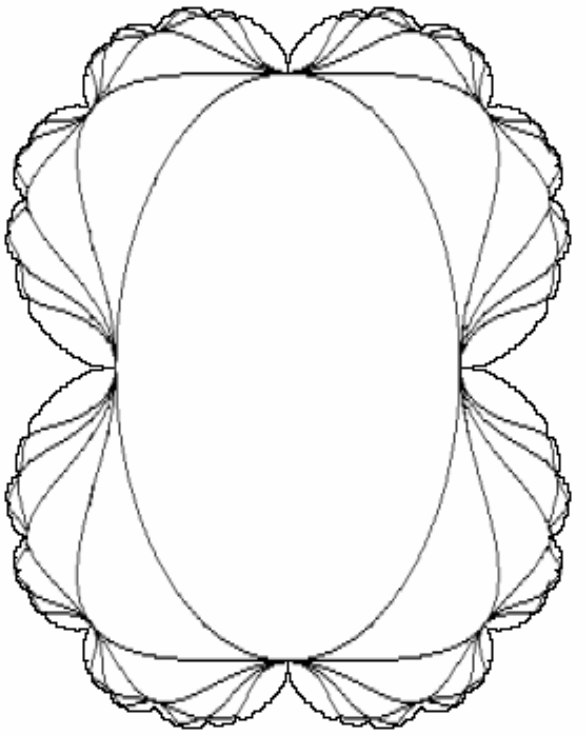


Figure 5: Spike-connected triangles

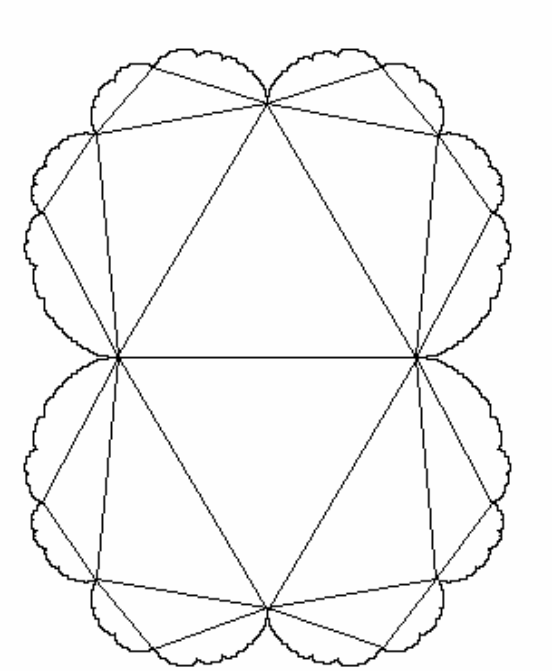


Figure 6: Sequence of triangles

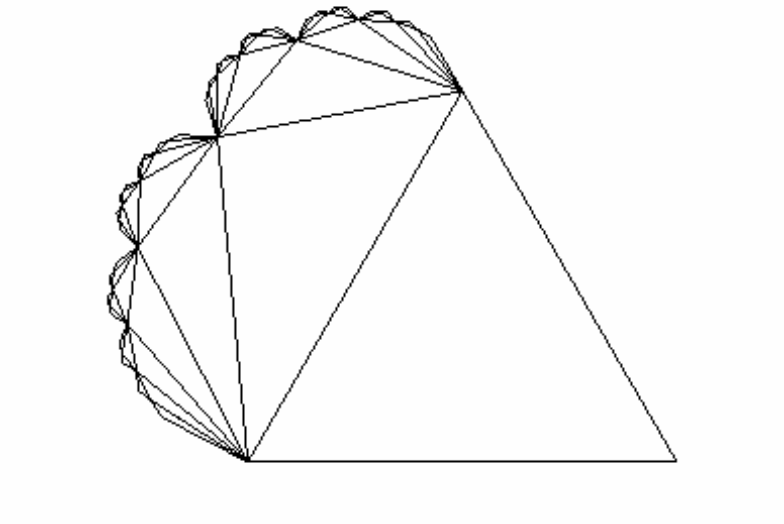


Figure 7: Isolated spike points

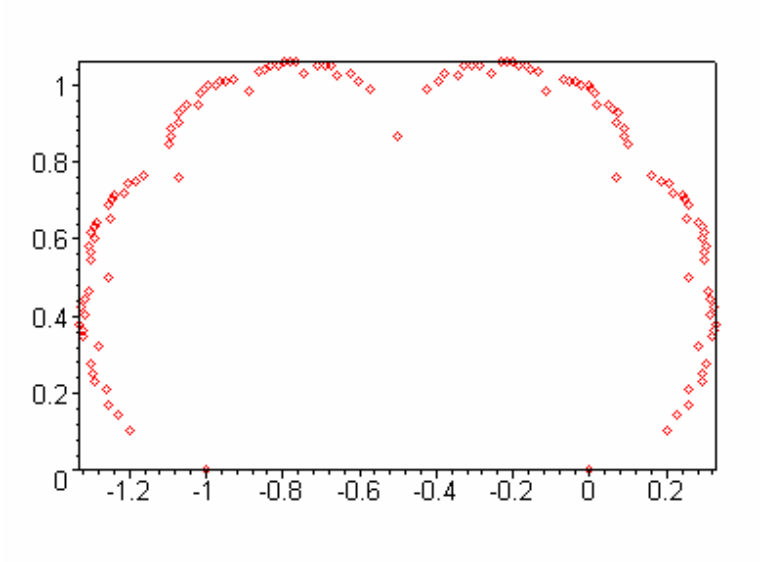


Figure 8: DM-Set tangent points (emphasized)

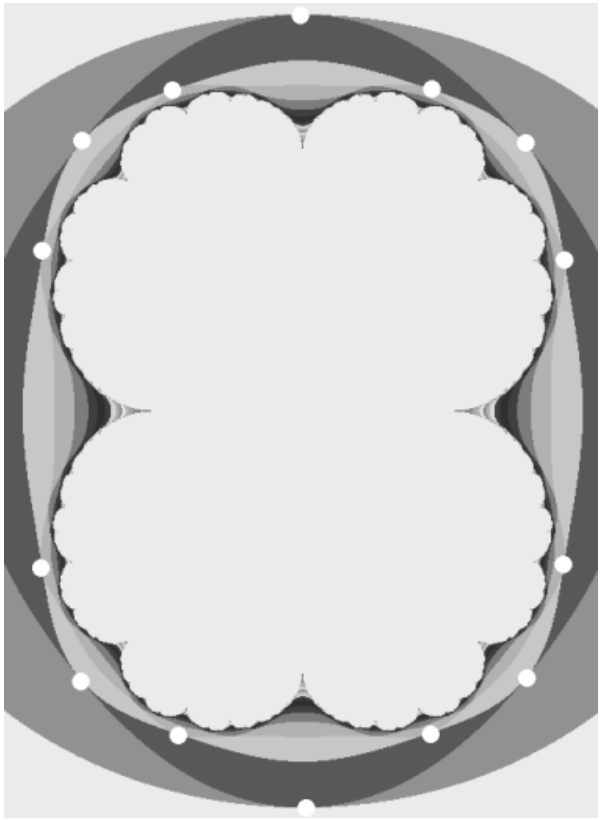
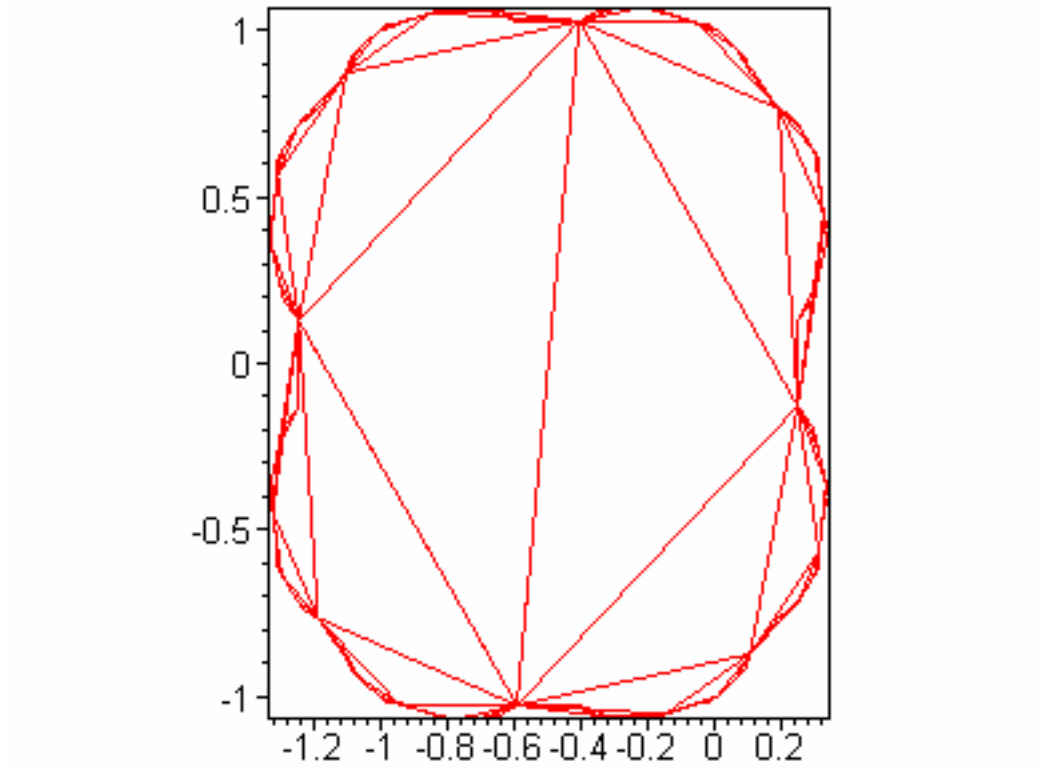


Figure 9: Upper Poly triangle structure



Appendix C: Research Accomplishments Summary

Main Research:

- Located the area of the Dynamical Mandelbrot within a range of ~0.023.
    -Upper bound: 2.94932944974534
    - Lower bound: 2.926713564103740

Side Research:

- Wrote various programs to:
    a) generate the DM-Set and allow for zooming
    b) draw lemniscates of the DM-Set
    c) track points in M-Set through iterations
    d) track points in DM-Set through iterations
    e) locate maps of lemniscate points under any number of iterations
    f) generate a bifurcation diagram for M-Set
- Created Maple worksheet to evaluate errors in various regression models.
- Computed higher order differences/ratios in data sets to predict the next term.
- Proved that the DM-Set is bounded within circle of radius 2 (Triangle Inequality Theorem).
- Wrote a Maple worksheet to generate 2-D and 3-D bifurcation maps for M-Set.
- Located boundary and periodic points in DM-Set.
- Developed method for locating DM-Set boundary points using “limit of spikes.”
- Developed method for cataloguing boundary points using base 2 fractions.

Appendix D: Example Maple Script (Upper Poly)

```
restart:
with(student):
with(geometry):
# Optimized, automated, and disgustingly fast script to generate
# triangle points in one fell swoop
# NOTE:  This script reuses triangle objects to conserve memory.
Digits:=15;
# Some initialization stuff:
# Equations for point generation:
s[0]:=x->-1/2+1/2*sqrt(4*(x+1/2)-1);
s[1]:=x->-1/2-1/2*sqrt(4*(x+1/2)-1);
# Self-explanatory:
printlevel := 0;
# First points:
a[0][0]:=-1;
u[0][0]:=-1;
# To take out the negatives:
t:={-1};
# Boolean formula for ordering based on angle from (-0.5,0):
F:=(x,y)-
>evalb(evalf(arctan(Re(x)+0.5,Im(x)))<evalf(arctan(Re(y)+0.5,Im(y))));
# First set and point:
L0:=[-1];
P0:=[point(C0n1,[Re(L0[1]),Im(L0[1])])];
# Origin (at -0.5):
point(O,[-0.5,0]);
origin:=[O];


# Start the loop:
TArea:=0;
DMArea:=0;
for i from 1 to 30
do
# Generate the points:

for k from 0 to 2^(i-1)-1
do
# Note: Introducing these evalf's early on sped this thing way up:
a[i][2*k]:=evalf(s[0](a[i-1][k]));
a[i][2*k+1]:=evalf(s[1](a[i-1][k]));
od;

# Get rid of stuff when we don't need it anymore:
unassign('a[i-1]');


# Eliminate the points below the real axis:
for k from 0 to 2^i-1
do
if evalf(Im(a[i][k]))>0 then u[i][k]:=a[i][k] else u[i][k]:=-1 end if;
od;
```

```
# Make a list:
L||i := {seq(u[i][k],k=0..(2^i-1))};

# Get rid of the u variables because we don't need em anymore:
unassign('u[i]');

# Take out the -1's:
L||i:=L||i minus t;

# Convert to a set:
L||i:=convert(L||i,list);

# Sort by increasing angle from -0.5:
L||i:=sort(L||i,F);

# Make points:
P||i:=[seq(point(C||i||"n"||k,Re(L||i[k]),Im(L||i[k])),k=1..(2^(i-1)))];

# Don't need the L-sets anymore:
unassign(cat('L',i));

# Now make the triangles:
# This will make the starting point P0[1]
oldset:=i;
oldelmt:=1;

# Make the first triangle here:
if i = 1 then triangle(T1n1,[O,P0[1],P1[1]]); fi;


for k from 1 to 2^(i-2)
do

# If k is odd, then set is 1:
if (k mod 2) <> 0 then setn:=1;
# else the set number is the number of factors of 2 in k, plus 1:
else setn:=1+ifactors(k)[2,1,2];

end if;

elmt:=(k+2^(setn-1))/(2^setn);

triangle(T||2||"n"||k,[P||(i-oldset)[oldelmt],P||i[k],P||(i-setn)[elmt]]);

oldset:=setn;
oldelmt:=elmt;

od;

# Add up the areas:
Area||i:=0;
for k from 1 to 2^(i-2)
do
Area||i:=Area||i + evalf(area(T||2||"n"||k));
od;

# Account for the first triangle:
```

```
if i=1 then Area1:=evalf(area(T1n1)); fi;

TArea:=TArea + Area||i;
DMArea:=TArea*4;

# Now we write the necessary stuff to a file:

if i=1 then f1_id := fopen("LowerPoly.txt",WRITE,TEXT) else
      f1_id:=fopen("LowerPoly.txt",APPEND,TEXT) end if;
fprintf(f1_id, "Area%d = %.15f\nTArea =
%.15f\nDMArea=%.15f\n\n",i,Area||i,TArea,DMArea);
fclose(f1_id);

# This writes it raw:
if i=1 then f2_id := fopen("LowerPolyRAW.txt",WRITE,TEXT) else
      f2_id:=fopen("LowerPolyRAW.txt",APPEND,TEXT) end if;
fprintf(f2_id, "%.15f\n",DMArea);
fclose(f2_id);




# Let's clear some memory:
unassign(cat('Area',i));




# This ends the i loop:
od;
```

Appendix E: Glossary

Bifurcation Diagram – a diagram showing the behavior of points within a dynamical system (often used to locate periodic points).

Bounded – does not "escape to infinity."

Complex Number – any number whose expression includes *i* (the square root of -1). Complex numbers are often written in the form a + b*i*.

Complex-Parameterized – used to describe a function of the type Z(t), where Z(t) corresponds to a complex number for each value of the parameter *t*.

Complex Plane – a coordinate system in which complex numbers are represented as points, with their real components corresponding to *x* and imaginary components corresponding to *y*.

Dynamical System – any function that maps a set onto itself. For example, the function $f(x) = x^2 + x$ is a dynamical system, mapping the real numbers into themselves.

Dynamical Mandelbrot Set – the Julia Set for the dynamical system $Z_{n+1} = Z_n^2 + Z_n$, so named because the equation is a true dynamical system, whereas the original Mandelbrot Set equation ($Z_{n+1} = Z_n^2 + C$) is not.

Escape Number – the number of iterations needed for a given point to move outside a circle of radius 2 (after which it is known to increase without bound).

Explicit/Implicit – used to described a function whose range variable is isolated (e.g., $y = x^2 + x$) / used to described a function whose variables are not isolated (e.g., $x^2 + y^2 = 4$).

Fractal – a self-similar picture derived from a mathematical equation.

Green's Theorem – a formula for integrating parameterized functions.

Iteration – the process of compositing a function with itself. For example, given $f(x) = x^2 + x$, the first iteration is $f^1(x) = f(x)^2 + f(x)$, the second is $f^2(x) = f^1(x)^2 + f^1(x)$, etc.

Julia Set – the set of all points in a given dynamical system whose orbits are bounded.

Lemniscate – the figures surrounding and approaching the Mandelbrot Set, created by shading regions of points that have the same escape number.

Mandelbrot Set – the set of all points C whose corresponding Julia Sets in the dynamical system $Z_{n+1} = Z_n^2 + C$ are connected. The Mandelbrot Set can also be described as the set of all points C whose orbits in the system $Z_{n+1} = Z_n^2 + C$ around zero (i.e., $Z_0 = 0$) are bounded.

Orbit – the behavior of a point through multiple iterations.

Parametric – used to describe a function of the type (x(t), y(t)), where the coordinates of the points in the function are defined with respect to a parameter *t*.

Periodic Points – points that return to their original position after a certain number of iterations (the *period*). For example, if the point (-1,0) returns to (-1,0) after 4 iterations under a certain dynamical system, (-1,0) is said to be a point of period 4.

Simpson's Method – a decently-accurate method for approximating integrals.

Spike Points – the spiked "vertices" of the Dynamical Mandelbrot Set.

Tangent Points – locations exterior to the DM-Set at which three consecutive lemniscates touch.