# Converging on the Area of the Mandelbrot set

Daniel Bittner and Winston Cheong

Rowan University

November 20, 2013

# Outline

# Student Research (Summer 2013)

- Daniel Bittner - CS Major.
- Winston Cheong - Math and CS Major.
- Done in collaboration with Dr. Hieu Nguyen

# Outline

# Definitions

## Definition (Mandelbrot set)

The set of values $c \in \mathbb{C}$ where the sequence $\{z_n\}$, defined by

$$z_{n+1} = z_n^2 + c, \quad z_0 = 0,$$

remains bounded.

## Definition (Bounded)

A sequence $\{z_n\}$ is bounded if there exists some real number $L$ such that

$$|z_n| \leqslant L$$

for all $n$.

# Pictoral Representation

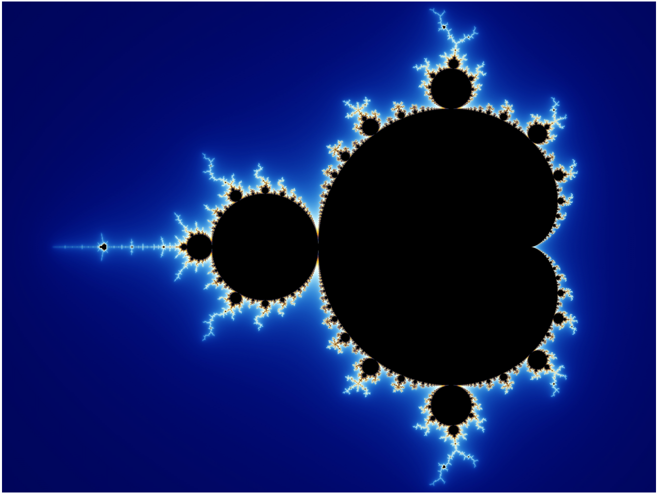


Figure: Pixels are colored according to how rapidly the sequence diverges

# The open problem

What is the exact area of the Mandelbrot set?

# The open problem

What is the exact area of the Mandelbrot set?

Approaches

- Pixel counting - 1.50659188849(28) (Thorsten Förstemann 2012)

# The open problem

What is the exact area of the Mandelbrot set?

**Approaches**

- Pixel counting - 1.5065918849(28) (Thorsten Förstemann 2012)
- Laurent series - 1.72 using 500K terms (Ewing, Schober)

# The open problem

What is the exact area of the Mandelbrot set?

**Approaches**

- Pixel counting - 1.5065918849(28) (Thorsten Förstemann 2012)
- Laurent series - 1.72 using 500K terms (Ewing, Schober)

**Our goal** Improve the estimates of the upper bound of the area using the Laurent series method by calculating 1 million b terms.

# The open problem

What is the exact area of the Mandelbrot set?

**Approaches**

- Pixel counting - 1.5065918849(28) (Thorsten Förstemann 2012)
- Laurent series - 1.72 using 500K terms (Ewing, Schober)

**Our goal** Improve the estimates of the upper bound of the area using the Laurent series method by calculating 1 million b terms.

# The open problem

What is the exact area of the Mandelbrot set?

**Approaches**
- Pixel counting - 1.5065918849(28) (Thorsten Förstemann 2012)
- Laurent series - 1.72 using 500K terms (Ewing, Schober)

**Our goal** Improve the estimates of the upper bound of the area using the Laurent series method by calculating 1 million b terms.

**Result**
- 1.(697) using 1.7M terms (B-C-N)

# How it works

## Theorem ([Ewing-Schober, 1992])

*Area is given by*

$$A_r = \pi \left[ 1 - \sum_{m=1}^{\infty} m|b_m|^2 \right]$$

*with* $b_m = \beta_{0,m+1}$, *where* $\beta_{n,m}$ *is defined as*

$$\begin{cases} \beta_{n,0} = 1 \\ \beta_{n,m} = 0 \qquad (1 < m < 2^{n+1} - 1, n \geqslant 1) \\ \beta_{n,m} = \dfrac{1}{2} \left[ \beta_{n+1,m} - \displaystyle\sum_{k=2^{n+1}-1}^{m-2^{n+1}+1} \beta_{n,k}\beta_{n,m-k} - \beta_{0,m-2^{n+1}+1} \right] \\ \quad \text{when } (m \geqslant 2^{n+1} - 1, n \geqslant 0) \end{cases}$$

# $\beta_{n,m}$ Values

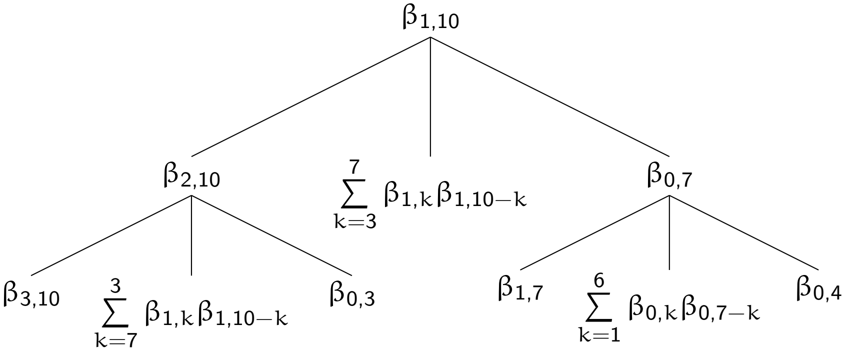
| n \ m | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | $-\frac{1}{2}$ | $\frac{1}{8}$ | $-\frac{1}{4}$ | $\frac{15}{128}$ | 0 | $-\frac{47}{1024}$ | $-\frac{1}{16}$ | $\frac{987}{32768}$ | 0 | $-\frac{3673}{262144}$ |
| 1 | 1 | 0 | 0 | $-\frac{1}{2}$ | $\frac{1}{4}$ | $-\frac{1}{16}$ | 0 | $-\frac{47}{256}$ | $\frac{1}{16}$ | $\frac{15}{2048}$ | 0 |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | $-\frac{1}{2}$ | $\frac{1}{4}$ | $-\frac{1}{16}$ | $\frac{1}{8}$ |
| 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Necessary Terms

A calculation up to 1,000,000 terms must contain

| | |
|---|---|
| Zeros | 2,048,536 zeros |
| $\beta$ Values | 17,951,464 values |
| $\beta_{0,m}$ Value calls | 1,000,001,000,000 calls |

# Calculation Example



# Visualizing the $\beta$ series

| n \ m | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * |
| 1 | 1 | 0 | 0 | * | * | * | * | * | * | * | * | * | * | * | * | * | * |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | * | * | * | * | * | * | * | * | * | * |
| 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | * | * |

$$\boxed{\beta_{n,m}} \; \frac{1}{2}\left( \boxed{\beta_{n+1,m}} - \boxed{\sum_{k=2^{n+1}-1}^{m-2^{n+1}+1} \beta_{n,k}\beta_{n,m-k}} - \boxed{\beta_{0,m-2^{n+1}+1}} \right)$$

# Visualizing the $\beta$ series

| n \ m | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * | * |
| 1 | 1 | 0 | 0 | * | * | * | * | * | * | * | * | * | * | * | * | * | * |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | * | * | * | * | * | * | * | * | * | * |
| 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | * | * |

$$\boxed{\beta_{n,m}} = \frac{1}{2}\left( \underline{\beta_{n+1,m}} - \boxed{\sum_{k=2^{n+1}-1}^{m-2^{n+1}+1} \beta_{n,k}\beta_{n,m-k}} - \underline{\beta_{0,m-2^{n+1}+1}} \right)$$

# Iterative Method

| N\M | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | $\beta_{0,1}$ | $\beta_{0,2}$ | $\beta_{0,3}$ | $\beta_{0,4}$ | $\beta_{0,5}$ | $\beta_{0,6}$ | $\beta_{0,7}$ | $\beta_{0,8}$ |
| 1 | 1 | 0 | 0 | $\beta_{1,3}$ | $\beta_{1,4}$ | $\beta_{1,5}$ | $\beta_{1,6}$ | $\beta_{1,7}$ | $\beta_{1,8}$ |
| 2 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | $\beta_{2,7}$ | $\beta_{2,8}$ |
| 3 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# Known results

## Theorem ([Ewing-Schober, 1992])

1. *If* $m = (2k+1)2^{\nu}$, *with* $k, \nu \in \mathbb{Z}$, $k \geqslant 0, 2^{\nu} \geqslant k+3$, *then* $b_m = 0$.
2. *If* $m = (2^{\nu+1} - 3)2^{\nu}$ *and* $\nu \geqslant 1$, *then* $b_n = -\frac{1}{m}\binom{2^{\nu-3/2}}{2^{\nu}-1}$.

## Theorem (Ahmed-N)

*Let* $n$ *and* $m$ *be integers such that* $n \geqslant 0$ *and* $2^{n+1} - 1 \leqslant m \leqslant 2^{n+2} - 3$. *Then for every* $p \in \mathbb{N}$, *we have*

$$\beta_{n,m} = \beta_{n+p,m+2^{n+1}(2^p-1)} = -\frac{1}{2}\beta_{0,m-2^{n+1}+1}$$

# New results I

## Theorem

*Let $n$ be a non-negative integer. Then*
$\beta_{n,2^{n+2}-2} = -\frac{1}{2}(\beta_{0,2^{n+1}-1} + \frac{1}{4})$

# New results II

Proof.
Recall that $\beta_{n,m} = -\frac{1}{2}\beta_{0,m-2^{n+1}+1}$ for $n \geqslant 0$ and $2^{n+1} - 1 \leqslant m \leqslant 2^{n+2} - 3$. We have

$$\begin{aligned}
\beta_{n,2^{n+2}-2} &= \frac{1}{2}\left[\beta_{n+1,2^{n+2}-2} - \sum_{k=2^{n+1}-1}^{2^{n+1}-1} \beta_{n,k}\beta_{n,2^{n+2}-2-k} - \beta_{0,2^{n+1}-1}\right] \\
&= \frac{1}{2}\left[0 - \beta_{n,2^{n+1}-1}^2 - \beta_{0,2^{n+1}-1}\right] \\
&= \frac{1}{2}\left[-\frac{1}{4}\beta_{0,0}^2 - \beta_{0,2^{n+1}-1}\right] \\
&= -\frac{1}{2}\left[\beta_{0,2^{n+1}-1} + \frac{1}{4}\right] \qquad \square
\end{aligned}$$

# New results III

**Theorem**

*Let $n$ be a non-negative integer. Then* $\beta_{n,2^{n+2}} = \frac{1}{16}$

# New results IV

Proof.
Recall that $\beta_{n,m} = -\frac{1}{2}\beta_{0,m-2^{n+1}+1}$ for $n \geqslant 0$ and $2^{n+1} - 1 \leqslant m \leqslant 2^{n+2} - 3$. We have

$$\begin{aligned}
\beta_{n,2^{n+2}} &= \frac{1}{2}\left[\beta_{n+1,2^{n+2}} - \sum_{k=2^{n+1}-1}^{2^{n+1}+1} \beta_{n,k}\beta_{n,2^{n+2}-k} - \beta_{0,2^{n+1}+1}\right] \\
&= \frac{1}{2}\left[-\frac{1}{2}\beta_{0,1} - 2\beta_{n,2^{n+1}-1}\beta_{n,2^{n+1}+1} - \beta^2_{n,2^{n+1}} - b_{0,2^{n+1}}\right] \\
&= \frac{1}{2}\left[-\frac{1}{2}\beta_{0,1} - \frac{1}{2}\beta_{0,0}\beta_{0,2} - \frac{1}{4}\beta^2_{0,1} - 0\right] \\
&= \frac{1}{2}\left[-\frac{1}{2}(-1/2) - \frac{1}{2}(1)(1/8) - \frac{1}{4}(-1/2)^2\right] \\
&= 1/16 \quad \square
\end{aligned}$$

# New results V

## Theorem

*Let* $n \geqslant 2$ *be an integer. Then* $\beta_{n,2^{n+2}+2} = -\frac{1}{2}\beta_{0,2^{n+1}+3}$.
*Let* $n \geqslant 2$ *be an integer. Then* $\beta_{n,2^{n+2}+4} = -\frac{1}{2}\beta_{0,2^{n+1}+5}$.
*Let* $n \geqslant 3$ *be an integer. Then* $\beta_{n,2^{n+2}+6} = -\frac{1}{2}\beta_{0,2^{n+1}+7}$.

# Outline

# Processing Power

- cc1.rowan.edu, Rowan cluster, 8 quad-core processers, 32 GB of ram
- 2 GB 800 MHz DDR2 SDRAM, 2.13 Ghz Intel Core 2 Duo, Mac OS X 10.6.8
- 32 GB 1066 MHz DDR3 RAM, 3.2 Ghz quad-core Intel Xeon, Mac OS X 10.7.5

# Procedure

- Calculated $\beta$ values are stored into a file
  - Stored as double-precision floating point decimal
- Terms calculated in batches of 10,000 or 100,000
  - Batches begin by reading stored values to rebuild the data structure and resume calculation.

# Language comparison

| Language | Execution | Typing | Scripting | Compiling |
|---|---|---|---|---|
| C++ | Compiled | Static | Procedural, Functional, Obj Oriented | Machine Code |
| Java | Compiled | Static | Obj Oriented | Bytecode |
| Python | Interpretative | Dynamic | Procedural, Functional | Non-compiled |

Table: Language comparison

# Language Timings Table

| Terms $b_m$ | Python(sec) | Java(sec) | C++(sec) | Terms $b_m$ | Python(sec) | Java(sec) | C++(sec) |
|---|---|---|---|---|---|---|---|
| 100 | 0.01 | 0.04 | 0.02 | 1100 | 15.47 | 6.41 | 5.78 |
| 200 | 0.05 | 0.11 | 0.11 | 1200 | 21.95 | 8.66 | 7.45 |
| 300 | 0.16 | 0.33 | 0.27 | 1300 | 30.10 | 11.86 | 9.43 |
| 400 | 0.39 | 0.47 | 0.51 | 1400 | 40.23 | 15.35 | 11.54 |
| 500 | 0.81 | 0.74 | 0.85 | 1500 | 52.49 | 19.67 | 13.88 |
| 600 | 1.55 | 1.01 | 1.30 | 1600 | 67.49 | 25.15 | 16.55 |
| 700 | 2.70 | 1.61 | 1.88 | 1700 | 85.27 | 31.68 | 19.61 |
| 800 | 4.43 | 2.23 | 2.59 | 1800 | 106.36 | 39.10 | 23.29 |
| 900 | 6.96 | 3.17 | 3.46 | 1900 | 131.02 | 47.62 | 27.58 |
| 1000 | 10.55 | 4.70 | 4.50 | 2000 | 159.725 | 58.17 | 32.49 |

# Memoization

Each $\beta_{n,m}$ calculation requires $m - 2^{n+1} + 2$ other $\beta$ terms.

Storing and calling previously computed $\beta$ values from a data structure reduces calculation time

| Terms $b_m$ | pure recursion(sec) | Terms $b_m$ | stored values(sec) |
|---|---|---|---|
| 10 | 2.56 | 1000 | 6.17 |
| 11 | 9.20 | 1100 | 7.03 |
| 12 | 43.26 | 1200 | 9.60 |
| 13 | 164.38 | 1300 | 11.95 |
| 14 | 594.49 | 1400 | 13.63 |

# Half Convolution

## Definition (Half Convolution)

$$\sum_{k=2^{n+1}-1}^{m-2^{n+1}+1} \beta_{n,k}\beta_{n,m-k}$$

$$= \begin{cases} 2(\sum_{k=2^{n+1}-1}^{m/2-1} \beta_{n,k}\beta_{n,m-k}) + (\beta_{n,m/2})^2 & \text{if } m \text{ even} \\ 2(\sum_{k=2^{n+1}-1}^{(m-1)/2} \beta_{n,k}\beta_{n,m-k}) & \text{if } m \text{ odd} \end{cases}$$

# Half Convolution

## Example

$$\beta_{1,10} \text{ convolution} = \beta_{1,3}\beta_{1,7} + \beta_{1,4}\beta_{1,6} + \beta_{1,5}\beta_{1,5} + \beta_{1,6}\beta_{1,4} + \beta_{1,7}\beta_{1,3}$$

$$\beta_{1,10} \text{ half convolution} = 2(\beta_{1,3}\beta_{1,7} + \beta_{1,4}\beta_{1,6}) + \beta_{1,5}\beta_{1,5}$$

The half convolution will calculate at most $m/2 + 1$ terms of the convolution for a non-zero convolution

# Convolution Timings

| Terms $b_m$ | Convolution(sec) | Half Convolution(sec) |
|---|---|---|
| 1000 | 1.31 | 0.70 |
| 2000 | 6.46 | 3.73 |
| 3000 | 17.94 | 12.15 |
| 4000 | 36.18 | 25.54 |
| 5000 | 61.35 | 44.14 |
| 6000 | 94.28 | 67.05 |
| 7000 | 135.56 | 96.31 |
| 8000 | 186.10 | 131.42 |
| 9000 | 246.01 | 170.17 |
| 10000 | 315.54 | 214.26 |

# Precision

## Theorem (Ewing-Schober)

$2^{2m+3-2^{(n+2)}}\beta_{n,m}$ *is an integer for* $n \geqslant 0$ *and* $m \geqslant 1$
$2^{2m+1}b_m$ *is an integer for* $m \geqslant 0$

If we set $B_{n,m} = 2^{2m+3-2^{(n+2)}}\beta_{n,m}$, then

## Definition (Ewing-Schober)

$$\begin{aligned} B_{n-1,m} &= 2^{2^{n+1}-1}B_{n,m} \\ &\quad - 2^{2^{n+1}-4}\sum_{k=2^n-1}^{m-2^n+1} B_{n-1,k}B_{n-1,m-k} \\ &\quad - 2B_{0,m-2^n+1} \end{aligned}$$

# Doubles and Unlimited Precision Integers

**Definition ($\beta$ and B)**

$\beta_{0,50} = 0.002286919608197$
$B_{0,50} = 890433596703485058699231232$

Integers quickly reach 100's of digits long

- 10,000 integers - 5 MB file size
- 10,000 doubles - 2.9 MB file size

# Precision Timings

| Terms $b_m$ | Integers(sec) | Doubles(sec) |
|---|---|---|
| 1000 | 4.68 | 3.61 |
| 2000 | 33.52 | 16.25 |
| 3000 | 133.78 | 39.19 |
| 4000 | 365.65 | 73.68 |
| 5000 | 777.67 | 119.37 |
| 6000 | 1438.78 | 177.65 |
| 7000 | 2441.06 | 248.63 |
| 8000 | 3880.82 | 330.53 |
| 9000 | 5854.09 | 423.86 |
| 10000 | 8479.87 | 529.62 |

# Multithreading and Parallelization

**Definition (Single-Threaded)**

- Each calculation is executed in order one-by-one
- This is how processing is normally handled

Time is lost waiting for independent calculations to complete

**Definition (Multi-Threading)**

- Multiple calculations are run simultaneously
- Requires multiple processor cores

# Visualizing MultiThreading

| N/M | ... | 15 | 16 | 17 | 18 | 19 |
|---|---|---|---|---|---|---|
| 0 | ... | $\cancel{\beta_{0,15}}$ | $\cancel{\beta_{0,16}}$ | $\cancel{\beta_{0,17}}$ | $\cancel{\beta_{0,18}}$ | $\beta_{0,19}$ |
| 1 | ... | $\cancel{\beta_{1,15}}$ | $\cancel{\beta_{1,16}}$ | $\beta_{1,17}$ | $\beta_{1,18}$ | $\beta_{1,19}$ |
| 2 | ... | $\beta_{2,15}$ | $\beta_{2,16}$ | $\beta_{2,17}$ | $\beta_{2,18}$ | $\beta_{2,19}$ |
| 3 | ... | $\beta_{3,15}$ | $\beta_{3,16}$ | $\beta_{3,17}$ | $\beta_{3,18}$ | $\beta_{3,19}$ |
| 4 | ... | 0 | 0 | 0 | 0 | 0 |

# MultiThreaded Implementation

**Thread** Handles a single column $\beta_{n,m}$ to $\beta_{0,m}$ run in parallel with adjacent columns

**Mutex** Locks calculation of $\beta_{0,m}$ until $\beta_{0,m-1}$ has been calculated

Boost C++ Multi-threading library used to handle the thread generation

**Results**

Multi-Threading is faster once column calculation time savings exceeds absolute cost of thread creation

# SingleThreaded and MultiThreading Comparision

- At small m, initial time and resource costs of thread creation have relatively high cost. Single threaded was faster.
- With large m, total time longer, lowering relative cost of thread creation. MultiThreading becomes preferred approach.
- At approximately 30,000 terms, a single threaded and multi threaded approach will calculate the next 1000 terms in the same amount of time

# MultiThreading Timings

| Terms | SingleThreaded | per 1000 | MultiThreaded | per 1000 | Terms | SingleThreaded | per 1000 | MultiThreaded | per 1000 |
|---|---|---|---|---|---|---|---|---|---|
| 21000 | 79.57 | 79.57 | 86.67 | 86.67 | 61000 | 334.43 | 334.43 | 317.45 | 317.45 |
| 22000 | 166.41 | 86.84 | 183.82 | 97.15 | 62000 | 675.78 | 341.35 | 645.51 | 328.06 |
| 23000 | 258.72 | 92.31 | 287.14 | 103.32 | 63000 | 1024.23 | 348.45 | 975.33 | 329.82 |
| 24000 | 358.61 | 99.89 | 395.01 | 107.87 | 64000 | 1376.50 | 352.27 | 1307.46 | 332.13 |
| 25000 | 477.39 | 118.78 | 510.37 | 115.36 | 65000 | 1738.57 | 362.07 | 1644.9 | 337.44 |
| 26000 | 592.79 | 115.4 | 629.30 | 118.93 | 66000 | 2107.39 | 368.82 | 1990.25 | 345.35 |
| 27000 | 713.81 | 121.02 | 756.41 | 127.11 | 67000 | 2484.35 | 376.96 | 2340.24 | 349.99 |
| 28000 | 836.34 | 122.53 | 886.59 | 130.18 | 68000 | 2894.49 | 410.14 | 2693.84 | 353.60 |
| 29000 | 969.25 | 132.91 | 1022.9 | 136.31 | 69000 | 3355.43 | 460.94 | 3055.19 | 361.35 |
| 30000 | 1111.76 | 142.51 | 1164.45 | 141.55 | 70000 | 3823.92 | 468.49 | 3423.97 | 368.78 |

# Outline

# $\beta$ properties

## Conjecture

*Let* $n \geqslant 0$ *and* $m \geqslant 2^{n+2} - 2$. *Write* $m$ *in the form* $m = 2^n + 2^q k$ *where* $q$ *is a non-negative integer* $q$ *and* $k$ *is an odd positive integer. Set* $p = \left\lceil \frac{2m+3-2^{n+2}}{2^{q+1}-1} \right\rceil$. *Then* $2^p \beta_{n,m}$ *is an integer.*

We see a fractal nature in both the $b$ and $\beta$ values.

# Future Goals

- Continue calculating additional terms of the series
- Implementation of MPI (Message Passing Interface) for multiple processors
    - Inter-processor communication between 4 processors on the CC cluster.

J. Ewing and G. Schober.
The area of the Mandelbrot set.
Numerische Mathematik, **61** (1992), 59 - 72.