The Gmandel parallel software

Edscott Wilson García
edscott@imp.mx
Programa de Matemáticas Aplicadas y Computación
Instituto Mexicano del Petróleo
México

December 3, 2009

Abstract

Keywords. Benchmark, cluster, fractal, Mandelbrot, Julia.

1 Introduction

1.1 The problem

Computer programs constructed for the purpose of benchmarking clusters [*reference other related software*] for the most part requiere the entire cluster system to be available for the tests. This is all very fine when a new system is assembled and has not yet been put into production. Once the cluster has been set in production status, system utilization may run between 80–100 percent, 24/7, with jobs in queue. Further changes to the operating system kernels, compiler, or message passing software such as PVM or MPI [*cite these*] cannot be consistently evaluated. What is needed is a benchmarking tool which does not requiere the entire cluster to be available. Gmandel addresses this specific problem in a novel way. [*Also, remember to point out that the model is scalable*]

[*for gmandel 1.4: fix input window, show equation being iterated, allow for scaling of graphic window*]

The structure of this paper is the following: In this introductory section we introduce the main definitions related to Jaulia and Mandelbrot sets and their generalizations. In second section we describe the software developed to deal with generalized sets in a cluster environment. In third section we invite the reader to take a journey into rarely explored parts of generalized Mandelbrot and Julia sets, computed quite efficiently and precisely with our cluster. Finally, in two appendices we chracterize the region of interest, in the complex plane, related with the generalizations of the Mandelbrot set.

2 Definitions

The Julia set is comprised of the points on the complex plane whose orbit never escapes to infinity upon the iteration $x_{i+1} = x_i^2 + c$, where $i \in \mathbb{N}$ and $x_i, c \in \mathbb{C}$.

The Mandelbrot set comprises the points on the complex plane whose orbit never escapes to infinity upon the iteration $x_{i+1} = x_i^2 + x_1$, where $i \in \mathbb{N}$ and $x_i, c \in \mathbb{C}$.

A generalized Mandelbrot set comprises the points on the complex plane whose orbit never escapes to infinity upon the iteration $x_{i+1} = x_i^q + x_1$, where $i \in \mathbb{N}$, $q \in \mathbb{N} - \{0, 1\}$ and $x_i, c \in \mathbb{C}$.

An α-generalized Mandelbrot set comprises the points on the complex plane whose orbit never escapes to infinity upon the iteration $x_{i+1} = \alpha x_i^{q_1} + (1-\alpha) x_i^{q_2} + x_1$, where $i \in \mathbb{N}$, $q_1, q_2 \in \mathbb{N} - \{0, 1\}$, $\alpha \in [0, 1] \subset \mathbb{R}^+$ and $x_i, c \in \mathbb{C}$.

Let us put the above notions in a more technical notation. A sequence of complex numbers $(z_n)_{n \geq 0}$ *tends to infinity* if $\forall C > 0\ \exists n_C \in \mathbb{N}$: $n \geq n_C \ \Rightarrow\ \sqrt{\overline{z}z} = |z_n| > C$. In this case, let us write $\lim_{n \to +\infty} z_n = \infty$. For any $z_0 \in \mathbb{C}$ and $r > 0$ let $D(z_0, r)$ be the closed disk centered at z_0 with radius r: $D(z_0, r) = \left\{ z \in \mathbb{C} \,\middle|\, \sqrt{\overline{(z - z_0)}(z - z_0)} \leq r \right\}$.

For any non-negative integer exponent $q \in \mathbb{N}$ and any complex number $c \in \mathbb{C}$, let $f_{qc} : \mathbb{C} \to \mathbb{C}$ be the map $z \mapsto z^q + c$. For any $z_0 \in \mathbb{C}$ let $\mathcal{Z}(z_0, q, c) = (Z_n(z_0, q, c))_{n \geq 0}$ be the sequence defined iteratively as $Z_0 = z_0$ and $\forall n \in \mathbb{N}$: $Z_{n+1} = f_{qc}(Z_n)$. The *q-Mandelbrot set centered at* z_0 is the set of complex numbers c such that $\mathcal{Z}(z_0, q, c)$ does not tend to infinity:

$$M_q(z_0) = \left\{ c \in \mathbb{C} \,\middle|\, \lim_{n \to +\infty} Z_n(z_0, q, c) \neq \infty \right\} \tag{1}$$

$M_2(0)$ is the very well known Mandelbrot set, which is a fractal included in the disk $D(0, 2)$. The *q-Julia set shifted by* c is the set of complex numbers z_0 such that $\mathcal{Z}(z_0, q, c)$ does not tend to infinity:

$$J_q(c) = \left\{ z_0 \in \mathbb{C} \,\middle|\, \lim_{n \to +\infty} Z_n(z_0, q, c) \neq \infty \right\} \tag{2}$$

It can be seen that, for any integer exponent $q \geq 4$, $M_q(0) \subset D(0, 2^{\frac{1}{q-1}})$, i.e. the q-Mandelbrot set centered at the origin is included within the disk of radius $2^{\frac{1}{q-1}}$ centered also at the origin.

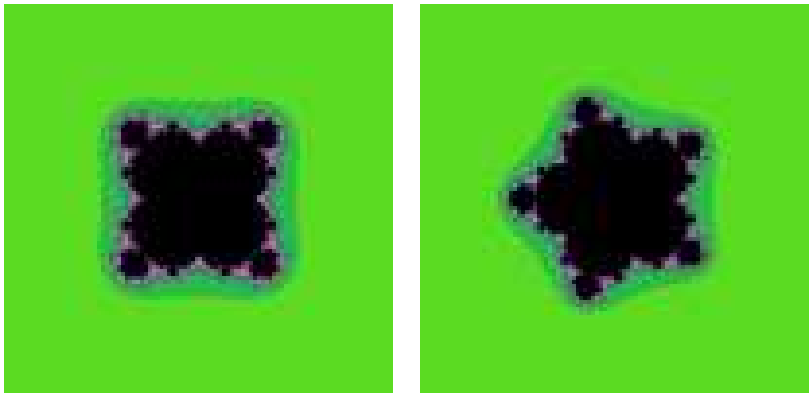
In fact, as $q \to +\infty$, $M_q(0)$ will approximate the unit disk $D(0, 1)$.

The supreme of the distances of points in $M_q(0)$ to the origin are realized by the vertexes of a regular polygon:

$$P_{qj} = 2^{\frac{1}{q-1}} \exp\left[i \left(\pi + j \frac{2\pi}{q-1} \right) \right], \qquad j = 0, \ldots, q-2 \tag{3}$$

and the minimum of the distances of points in the complement of $M_q(0)$ to the origin are realized by the vertexes of a regular polygon:

$$p_{qj} = r_q \exp\left[i \left(j \frac{2\pi}{q-1} \right) \right], \qquad j = 0, \ldots, q-2 \tag{4}$$



(a) $q = 5$ (b) $q = 6$

Figure 1: Annuli of interest in the generalized Mandelbrot sets.

where $r_q \in]0, 1[$ is the radius of convergence of the series given by equation. (7) in appendix A.

Let G_q be the regular polygon with vertexes $(P_{qj})_{j=0}^{q-2}$ and let g_q be the regular polygon with vertexes $(p_{qj})_{j=0}^{q-2}$. G_q is drawn within the bounding disk $D(0, 2^{\frac{1}{q-1}})$ which contains $M_q(0)$. g_q is drawn within the disk $D(0, r_q)$ which lies entirely in $M_q(0)$. G_q and g_q seem to be reflected each other, i.e. the vertex P_{q0} lies in the real negative axis of the complex plane while the vertex p_{q0} lies in the real positive axis. Thus the interesting part of $M_q(0)$ lies within the annulus $A_q = \{z \in \mathbb{C} |\ r_q \leq |z| \leq 2^{\frac{1}{q-1}}\}$. This can be seen in the figure 1, with odd and even values of $q = 5$ and $q = 6$.

3 Parallel calculations

3.1 The parallel algorithm and partitioning strategy

Since each pixel on the fractal image is independent from its neighbors, a divide-and-conquer mechanism is best suited for the problem. The image is divided into columns, and these columns are assigned to individual computer nodes by means of message passing techniques. Each individual computer node then performs a second division of the problem by means of `POSIX` threads so as to generate the requested number of lightweight processes acting upon shared memory. In this manner the rows are partitioned by the separate threads acting upon them. When any individual computer node finishes each assigned column, a message with the column results are posted to the process which has control of the graphic interface.

3.2 Communication and synchronization

To minimize the startup communication, a single message block is sent to each remote heavyweight process. In this message block each remote process child learns the parameters of the graphical image to be calculated, the number of heavy weight siblings, the quantity of threads to use, and the individual child identification number. With such identification number and the number of siblings, each child knows exactly which columns has to undertake.

To return the calculated values and to build an image fragment with them, each process parent shall define an array and directly read into this memory device the column vectors as they are sent by the children. The message identification will correspond to the column number so that the parent will read them as soon as they are available without expecting to receive them in any particular order. As soon as each child has finished sending the calculated columns it will send an additional message which contains the total number of floating point operations realized so that the parent process can tabulate the benchmark MFLOPS.

The initial message is sent coded as bytes, assuming a homogeneous cluster conformation. The resultant vector columns are encoded as integer values, and the number of floating point operations as a long unsigned.

As pointed out earlier, each heavyweight child further divides the problem by means of threads. Each child thread will solve only the rows assigned to it, which it can calculate by examining its own thread identification number and the amount of sibling threads which will be running. Since the threads are acting on shared memory without any overlapping, each thread stores the results obtained directly into the vector which will be sent back via message passing to the process in charge of generating the final image. In order to eliminate the need for a mutex —which significantly deteriorates performance— in the tabulation of the number of floating point operations, each thread stores the number of floating point operations in a local variable which will be returned to the parent as a reference upon exiting.

With threads, there is one consideration which must be made to avoid a race condition. Within the structure which is sent to each child thread is the thread identification number. Since this data structure is defined within the parent thread, the memory location is shared. It is therefore imperative that the child has time to read this data before the parent modifies it for the next thread. This is done by introducing a semaphore within the data structure which will be set by the child thread after securing the data within its local stack. This will allow the parent to create the next thread. Again, this approach is significantly faster than using operating system semaphores.

3.3 Benchmarking clusters with full CPU utilization

This is done using the dual message-passing/shared-memory feature of `Gmandel`. The Linux operating system does not distinguish between heavyweight processes and lightweight processes while it assigns CPU time to different processes. If all processes have the same priority, which is generally the case, it is sufficient to request a high number of threads. In this manner the operating system will not distinguish between the threads and other processes competing for CPU time. For example, let us look at an individual computer node before the `Gmandel` run:

```
1:30pm up 50 days, 5:46, 1 user, load average: 2.00, 2.00, 2.00
35 processes: 32 sleeping, 3 running, 0 zombie, 0 stopped
CPU0 states: 99.4% user, 0.1% system, 0.0% nice, 0.0% idle
CPU1 states: 100.0% user, 0.0% system, 0.0% nice, 0.0% idle
Mem: 1028356K av, 894896K used, 133460K free, 10888K shrd, 606324K buff
Swap: 2048276K av, 0K used, 2048276K free 69508K cached
PID USER PRI NI SIZE RSS SHARE STAT %CPU %MEM TIME COMMAND
```

```
10957 mmartine 17 0 48364 47M 10704 R 99.9 4.7 1383m nwchem.luis
10956 mmartine 16 0 48648 47M 11324 R 99.7 4.7 1368m nwchem.luis
11863 edscott 9 0 948 948 764 R 0.1 0.0 0:00 top
```

And during the `Gmandel` run:

```
1:46pm up 50 days, 6:03, 1 user, load average: 12.73, 8.52, 6.04
61 processes: 34 sleeping, 27 running, 0 zombie, 0 stopped
CPU0 states: 99.0% user, 0.1% system, 0.0% nice, 0.0% idle
CPU1 states: 100.0% user, 0.0% system, 0.0% nice, 0.0% idle
Mem: 1028356K av, 895104K used, 133252K free, 10888K shrd, 606324K buff
Swap: 2048276K av, 0K used, 2048276K free 69544K cached
PID USER PRI NI SIZE RSS SHARE STAT %CPU %MEM TIME COMMAND
12448 edscott 20 0 784 784 532 R 11.7 0.0 0:02 mandel_pvm_a
12449 edscott 20 0 784 784 532 R 11.7 0.0 0:02 mandel_pvm_a
12451 edscott 20 0 784 784 532 R 11.7 0.0 0:02 mandel_pvm_a
12455 edscott 20 0 784 784 532 R 11.7 0.0 0:02 mandel_pvm_a
12457 edscott 20 0 784 784 532 R 11.7 0.0 0:02 mandel_pvm_a
12446 edscott 19 0 784 784 532 R 10.7 0.0 0:02 mandel_pvm_a
12450 edscott 20 0 784 784 532 R 10.7 0.0 0:02 mandel_pvm_a
12447 edscott 18 0 784 784 532 R 9.8 0.0 0:02 mandel_pvm_a
12452 edscott 20 0 784 784 532 R 7.8 0.0 0:02 mandel_pvm_a
10956 mmartine 15 0 48648 47M 11324 R 6.8 4.7 1381m nwchem.luis
10957 mmartine 14 0 48364 47M 10704 R 5.8 4.7 1395m nwchem.luis
12453 edscott 20 0 784 784 532 R 5.8 0.0 0:02 mandel_pvm_a
12454 edscott 20 0 784 784 532 R 5.8 0.0 0:02 mandel_pvm_a
12456 edscott 20 0 784 784 532 R 5.8 0.0 0:02 mandel_pvm_a
12458 edscott 20 0 784 784 532 R 5.8 0.0 0:02 mandel_pvm_a
12459 edscott 20 0 784 784 532 R 5.8 0.0 0:02 mandel_pvm_a
12460 edscott 20 0 784 784 532 R 5.8 0.0 0:02 mandel_pvm_a
12461 edscott 20 0 784 784 532 R 5.8 0.0 0:02 mandel_pvm_a
12462 edscott 20 0 784 784 532 R 5.8 0.0 0:02 mandel_pvm_a
12463 edscott 20 0 784 784 532 R 5.8 0.0 0:02 mandel_pvm_a
12464 edscott 20 0 784 784 532 R 5.8 0.0 0:02 mandel_pvm_a
12465 edscott 20 0 784 784 532 R 5.8 0.0 0:02 mandel_pvm_a
12466 edscott 20 0 784 784 532 R 5.8 0.0 0:02 mandel_pvm_a
12467 edscott 20 0 784 784 532 R 5.8 0.0 0:02 mandel_pvm_a
12468 edscott 20 0 784 784 532 R 5.8 0.0 0:02 mandel_pvm_a
12469 edscott 20 0 784 784 532 R 5.8 0.0 0:02 mandel_pvm_a
```

Which is an unequal allotment of CPU time which benefits programs which combine message passing with shared memory, in Beowulf class clusters.

3.4 Performance model

3.4.1 Shared memory

In the calculation of many types of fractals, as in the case of the Mandelbrot and Julia sets, the time it takes depends on several factors. Over each pixel of the image —or point of the

grid— a series of iterations will be performed. The number of iterations per pixel is not known beforehand. In fact, the number of iterations is the unknown to be solved to render the fractal image in color. The time it takes one iteration on each pixel is constant, since it only depends on the number of floating point operations performed and the processor speed. This number of floating point operations per iteration, which we shall call c, determines a computation time $t = c/f$ where f is the number of floating point operations per second (FLOPS) that the processor in question is capable of realizing.

For a given grid determined by the selected coordinates upon the complex plane, there exists a fraction α of points for which it cannot be determined that the orbit escapes after a certain number of iterations.

Let $\mathbf{L} = \{L_1, L_2, L_3, \ldots\}$ be an increasing sequence of integer indexes. Let $\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \alpha_3, \ldots\}$ be the sequence of proportions of elements whose orbit does not escape for each entry in the sequence $\mathbf{L}$. Then as $L_i \to \infty$, there exists a limit value α such that $\alpha_i \to \alpha$. For any points whose orbital sequence escapes, we have that for almost all i the number of iterations is strictly less than L_i, and for those whose orbit does not escape the number can be reached L_i. Therefore the computation time is

$$t \leq \frac{\alpha_i c L_i}{f} + \frac{(1-\alpha_i) c L_i}{f}.$$

And the time to calculate any point in the grid is less than or equal to $\frac{cL_i}{f}$. The maximum time for the limit $\alpha > 0$ is $\frac{cwhL_i}{f}$, where $w \times h$ are the grid dimensions.

It can be seen that the value of α varies according to the region of the complex plane that is being examined, thus parameter α cannot be known beforehand. On the other hand, the parameters w and h remain constant while generating the fractal image. Therefore the only parameter that can vary is L_i, to resolve areas with more detail close to the border of the Mandelbrot set.

With respect to the shared memory component of `Gmandel`, all threads act upon the same space of memory. For this segment of calculation there exists a fraction of pixels whose orbit does not escape. To each thread there corresponds an α_i which indicates the fraction of points whose orbit does not escape. If each processor has an equivalent number of points, then the α corresponding to L will be $\alpha = \sum_i \alpha_i$, and the computation time will be determined by the processor with the highest value for α_i, being this time $t \leq \frac{\alpha_i cL}{f}$.

Analyzing the source code for the Mandelbrot set, it is easy to determine that $c = 8$ is the count of the floating point instructions. Analyzing the results that were generated in a two processor SMP computer, and ignoring run times which were too short, we obtain the results shown in table 1 where the value of the highest α_i is also tabulated (since this determines the acceleration). To achieve the maximum acceleration, the ideal number of processors is equivalent the the maximum of w y h, being that if it is w, the partitioning should be done by columns and if it is h the partitioning should be by rows. But since any useful fractal image should have at least a 300 pixel width, a SMP computer which could provide the maximum acceleration would have to have 300 processors —which is a machine that is very difficult to come about. This shows that the shared memory approach can only be thought of as a partial solution to the problem of fractal image generation and that further work must be done along the lines of message passing between different computers.

L	serial	2 threads	acceleration	α_{max}
3200	11	6	1.8	0.555
6400	21	13	1.6	0.625
12800	42	28	1.5	0.667
25600	85	51	1.7	0.588
51200	170	83	2.0	0.500

Table 1: Values of acceleration rates with respect to L.

3.4.2 Message passing

The message passing approach is to do a double partitioning, one by columns and one by rows. The message passing partition is done along the columns of the grid. Each remote PVM process will do the calculations and send the results to the parent PVM process to conform the final image. Also, each remote PVM process will do a shared memory partitioning along the rows of each column. This implies w messages of length $4h$, using 32 bit integers. The time lost to latency will be given by $w\lambda$. The time lost due to data length will be $4h\beta$, where β is the bandwidth. The computation time will be therefore marked off by:

$$t \leq w\lambda + 4h\beta + \frac{\bar{\alpha}cLwh}{fN},$$

where N is the number of PVM nodes and $\bar{\alpha}$ is the maximum fraction of points whose orbit does not escape for any of the remote PVM processes.

Since w and h remain constant, and λ and β are constants that depend upon the hardware, we can write the generalizes constants:

$$\begin{aligned} k_1 &= w\lambda + 4h\beta \\ k_2 &= \frac{whc}{f}, \end{aligned}$$

with which the mark-off equation for computation time becomes:

$$t \leq k_1 + \frac{k_2 L\bar{\alpha}_{max}}{N}.$$

Nevertheless each remote PVM process makes private use of shared memory to take advantage of the fact that they may be SMP nodes with two or more processors available. Thus the computation time factor becomes:

$$t \leq k_1 + \frac{k_2 L\alpha_{max}}{nN},$$

where α_{max} refers to the maximum fraction of points whose orbit does not escape that corresponds to the threads for each PVM process and n i the number of processors available at each computer. With the results obtained on a Beowulf class cluster at the IMP (Mexican Institute of Petroleum) conformed by 125 dual Pentium-3 @ 1 GHz connected by switched

L	time in seconds	acceleration	α_{max}	processors
25600	6	14.2	0.070	20
51200	10	17.0	0.059	20
102400	20	17.0	0.059	20
204800	40	17.0	0.059	20
25600	2	42.5	0.024	80
51200	3	56.7	0.018	80
102400	7	48.6	0.021	80
204800	12	56.7	0.018	80
25600	1	85	0.012	140
51200	2	85	0.012	140
102400	4	85	0.012	140
204800	7	97	0.010	140

Table 2: Values of acceleration rates.

Gigabit Ethernet, we obtained the following results with the values shown at table 2 for α_{max}. For the message passing scenario with PVM, the optimum configuration is with remote SMP nodes. The number of columns would correspond to the width of the array, w, and the number of processors on each SMP node would equal the number of rows, h. A configuration of this nature would be extremely difficult to achieve. Nonetheless, the parameters of `Gmandel` can be moved to accommodate any number of remote PVM nodes with any number of processors by node. This is done by toggling the values of remote heavyweight processes and threads per heavyweight process.

4 A voyage into the seldom explored Mandelbrot

4.1 The generalized Mandelbrot set

The images in figure 2 correspond to the generalization of the Mandelbrot set. They refer to the exponents $q = 2$ through $q = 9$, plus $q = 13$. One detail to notice is that the number of bulbs grows. In the case where $q = 2$, there is one large bulb, while with $q = 3$ there are two large bulbs. It is easy to see that the number of large bulbs will equal $q - 1$. Another interesting detail is with the butt. The number of butts increase with q and is equal to $q-1$. But there is always a butt on the positive x-axis, which progressively gets smaller. What happens when q increases without bound? In figure 3 we may take a look at $q = 30, 120, 1200$. What's happening? The generalized Mandelbrot set tends to the unit circle as was pointed at the end of section 1. And we must also recall the Mandelbrot is further tied to the circle with the appearance of the irrational number π. This number, which represents the ratio of the diameter to the circumference of a circle, appears in different manner, as shown by Dave Boll [2, 4], and proved by Aaron Klebanoff [11]. This further ties the Mandelbrot set, and specifically the generalized Mandelbrot set, to an alternate definition of the concept of a circle. This is so because the circumference is not really smooth, but can be as smooth as we

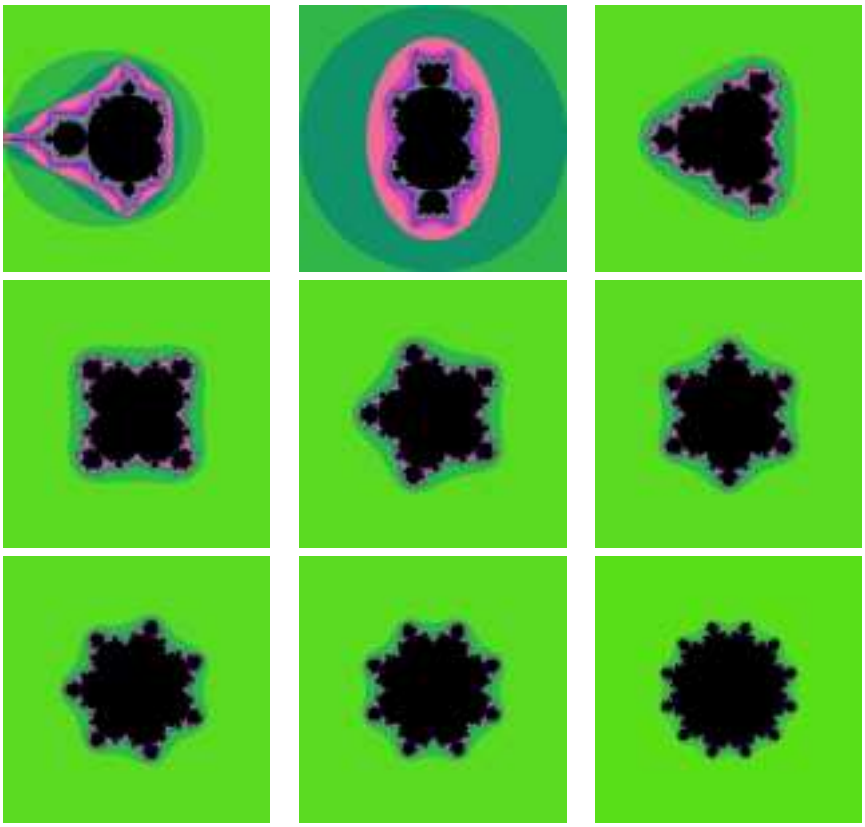


Figure 2: Generalized Mandelbrot sets for $q \in \{2, \ldots, 9, 13\}$.

Figure 3: Generalized Mandelbrot sets for $q \in \{30, 120, 1200\}$.

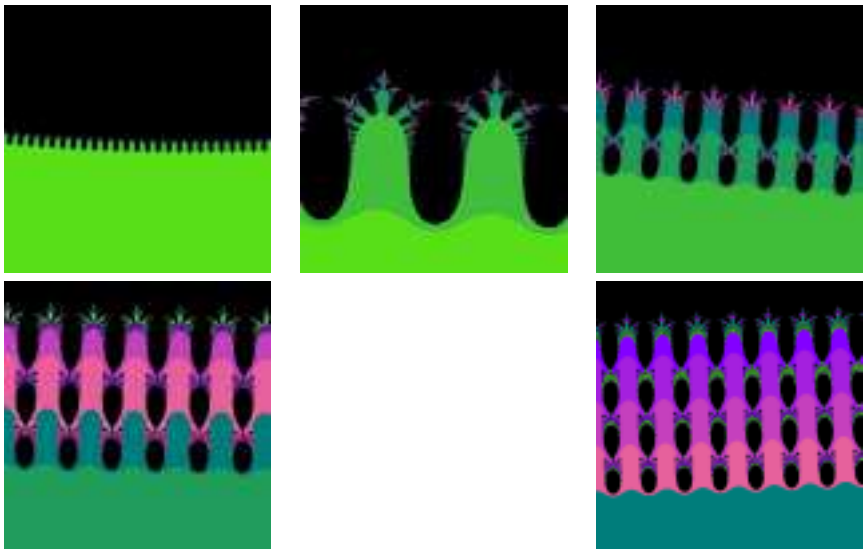


Figure 4: Closest views at apploximatiosn of the unit circle.

want it to be. This is nature's way to make circles since the circumference must eventually come down to atoms.

But what does the smooth part of the Mandelbrot "circle" ($q = 1200$) looks like on enhancement? A closer look can be taken at figure 4, sequentially zooming in on the smoothest part of the graph. Which kind of reminds us of a gas bubbling through a liquid, complete with the bubbles bursting with a splash. Figure 5 displays two zooms at the splash. And then zoom in for two looks at the cloudy part are displayed on figure 6 Who would ever guess that what looks like a circle is really what you observe above? Maybe only those who work in quantum chemistry and who know there are no definite borders between atoms in matter, only electron clouds.

4.2 The generalized Julia Set

Just as with the Mandelbrot set it is possible with the software to alter the exponent on the Julia iteration formula and generate interesting images. The two images shown on figure 7 correspond to a Julia set with $q = 5.$ and the two images on figure 8 correspond to a Julia sets with $q = 13.$

4.3 The alpha generalized Mandelbrot set

The Alpha generalized Mandelbrot set also produces some very interesting images. The two images of figure 9 correspond to an $\alpha = 50, q = (100, 2)$-fractal. Or zooming into the $\alpha = 50, q = (5000, 2)$-fractal image, the four images shown at figure 10 are obtained. And even further with the three images on figure 11.

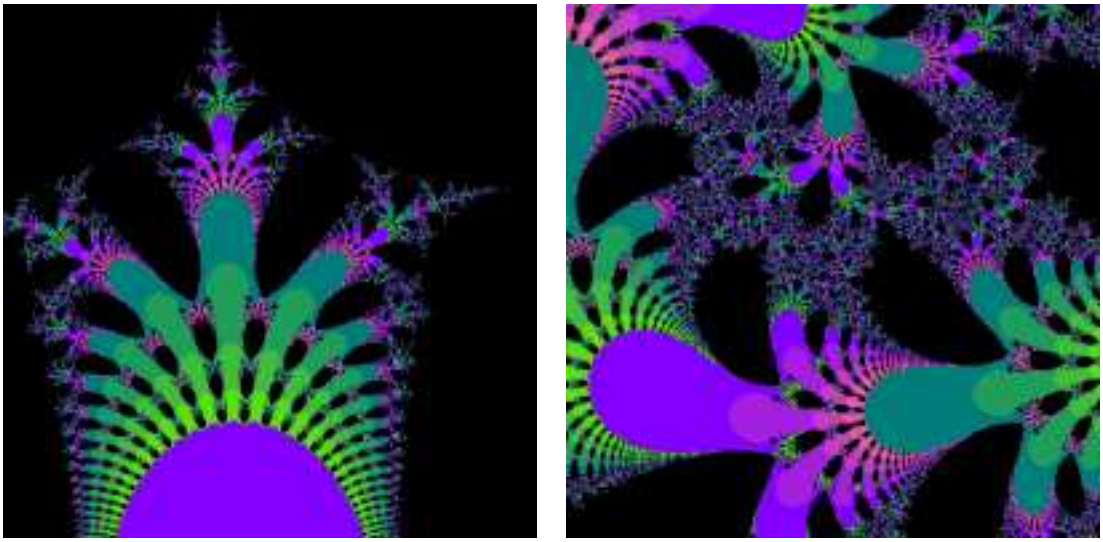


Figure 5: Two zooms at the splash.

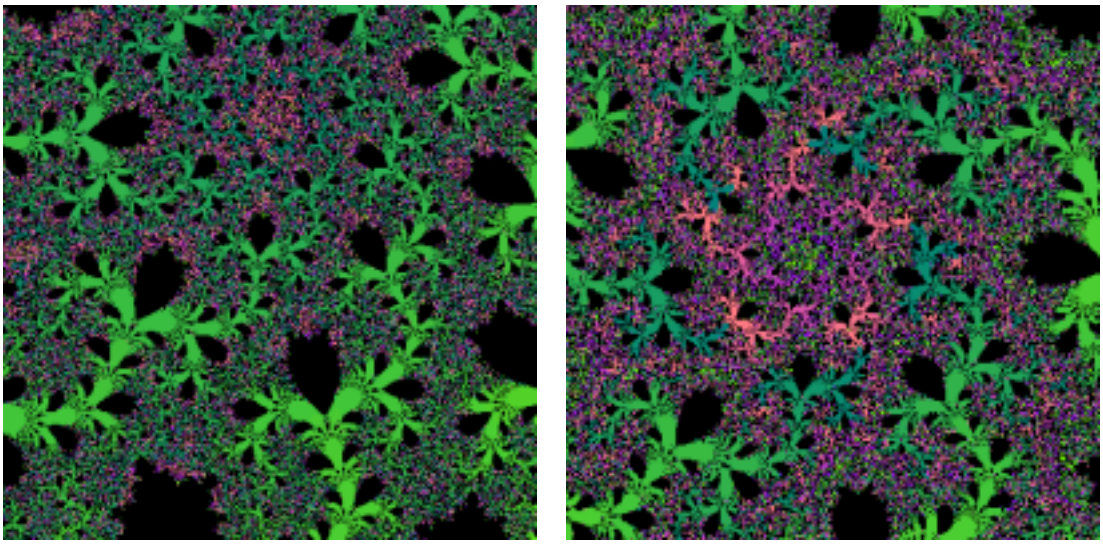


Figure 6: Two looks at the cloudy part.

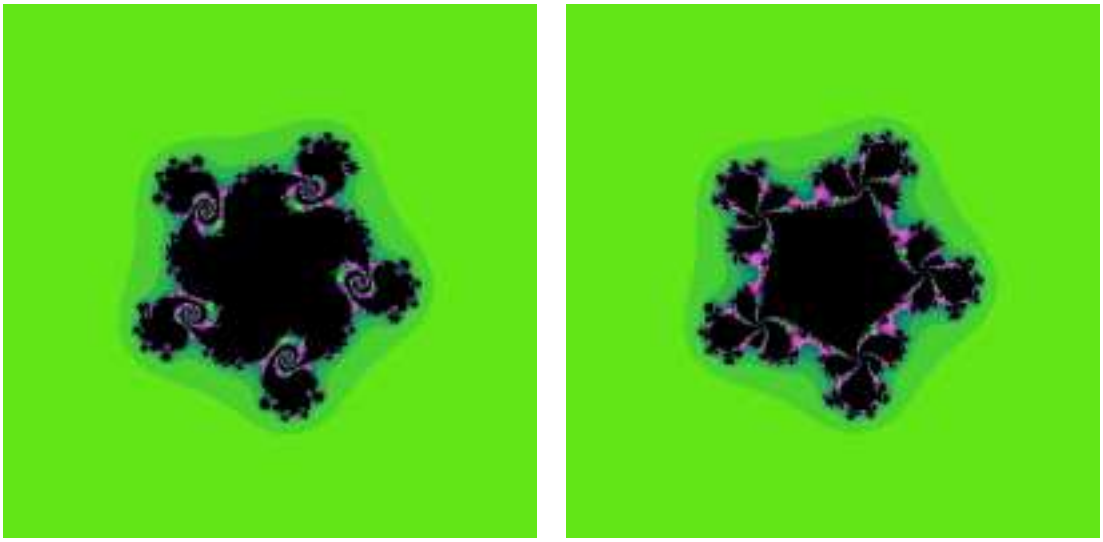


Figure 7: Julia set with $q = 5$.

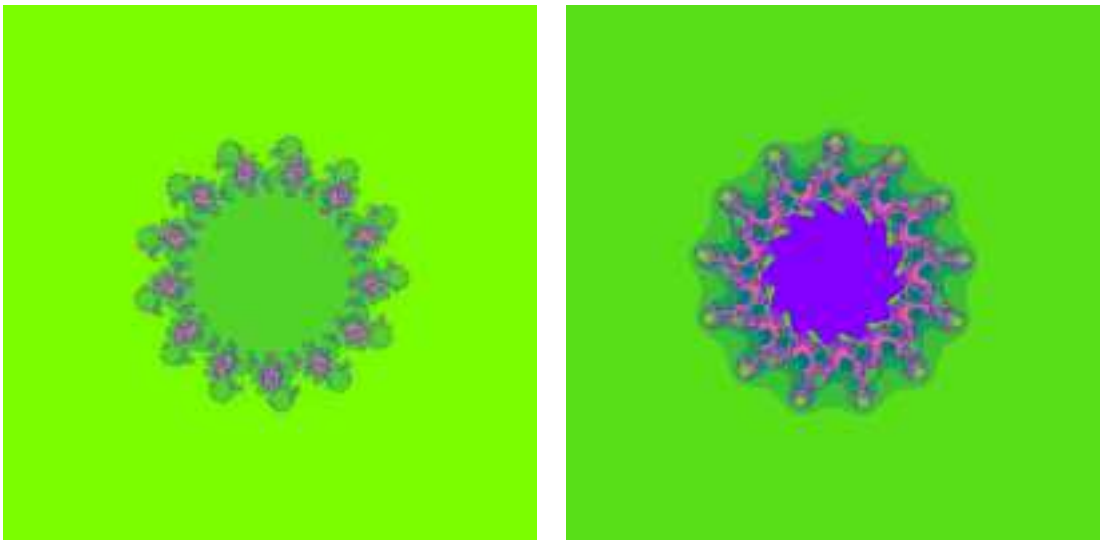


Figure 8: Julia set with $q = 13$

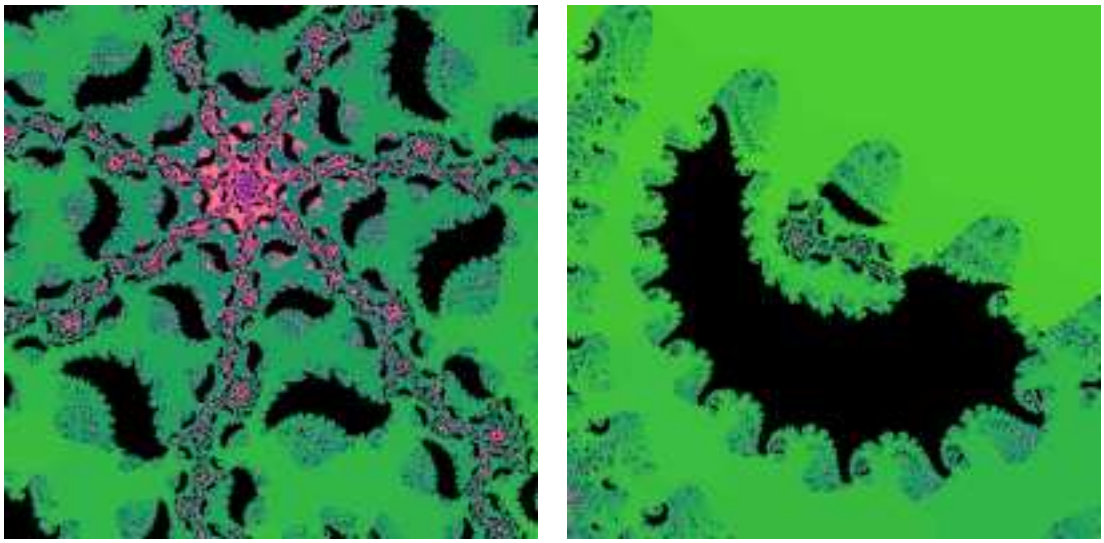


Figure 9: An $\alpha = 50, q = (100, 2)$-fractal.

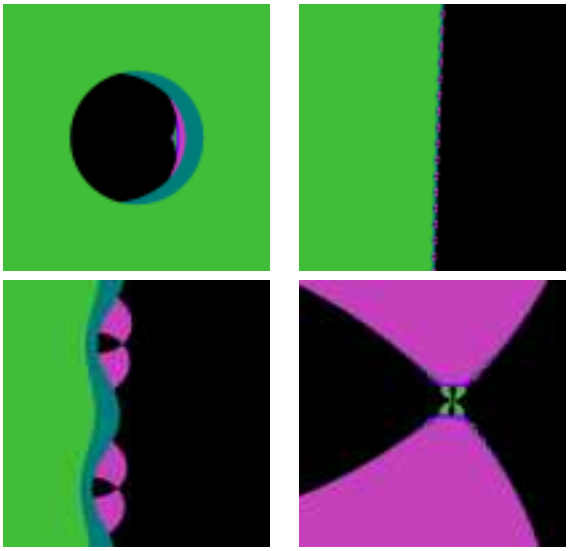


Figure 10: Zooming the $\alpha = 50, q = (5000, 2)$-fractal image.

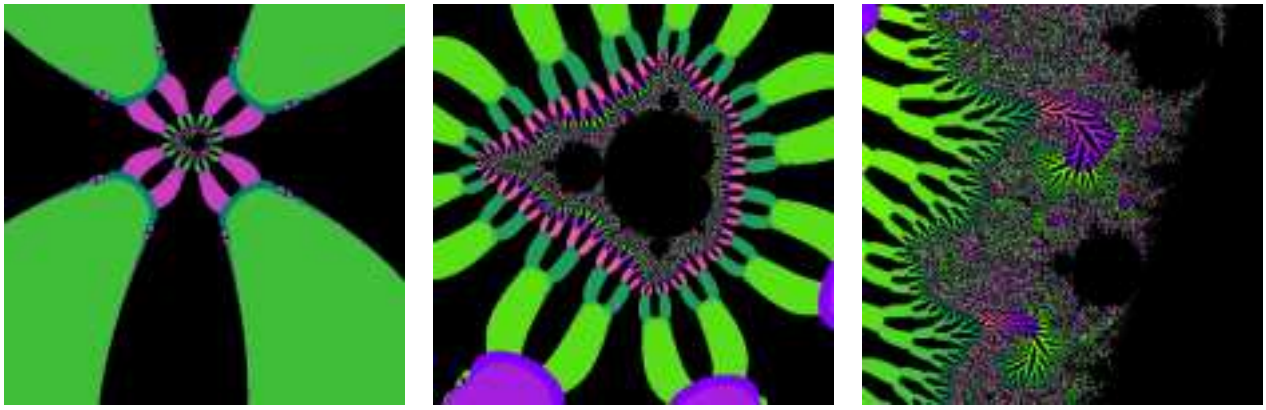


Figure 11: Further zooming.

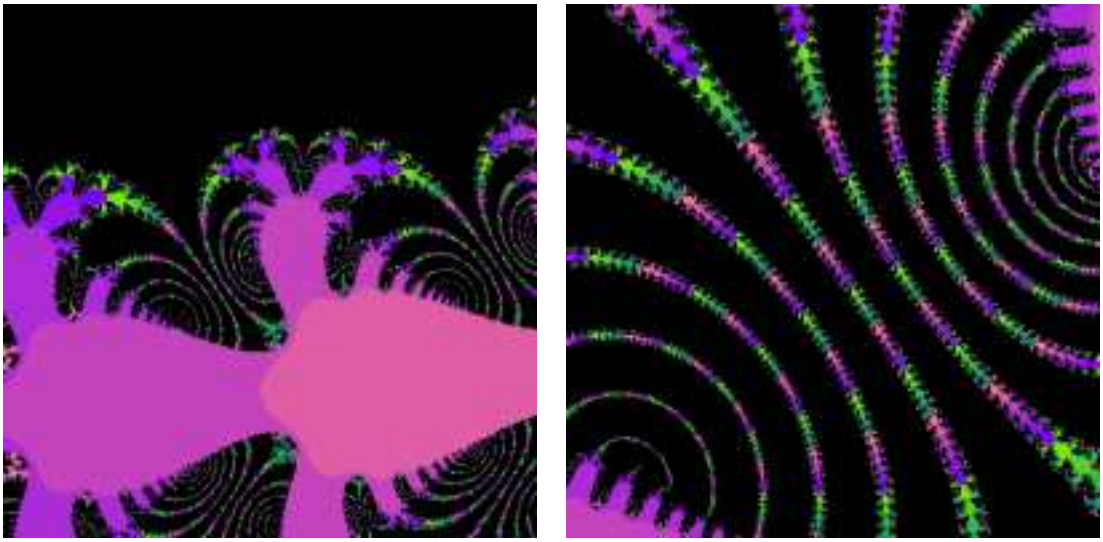


Figure 12: Details of the a $\alpha = 50, q = (2, 1129)$-fractal.

4.4 Statistics calculation

Many of the above images can be calculated within reasonable time on a single computer running Linux or FreeBSD. But some of the images take too long. For each of the two images in figure 12, for example, which represent details of the a $\alpha = 50, q = (2, 1129)$-fractal, the computation time is more than twenty minutes on a Pentium-IV at 1.8 GHz, while using the PVM version of `Gmandel` with 100 dual Pentium-3 computers connected by a Gigabit switch, the image generation is cut down to only 11 seconds. While in the future processor speeds will most certainly be able to match such performance, in the meantime the use of a computer cluster allows a peek at the fractals to be generated on future generation desktop computers.

References

[1] Alfeld, P., *The Mandelbrot Set*, http://www.math.utah.edu/~alfeld/math/mandelbrot/mandelbrot.html

[2] Boll, *D. Pi and the Mandelbrot set*, http://www.frii.com/~dboll/mandel.html

[3] Devaney, R. L. (ed.) *Complex Analytic Dynamics: The Mathematics Behind the Mandelbrot and Julia Sets*, American Mathematical Society, Providence, 1994.

[4] Edgar, G., *Pi and the Mandelbrot set*, http://www.math.ohio-state.edu/~edgar/piand.html

[5] Falconer, K., *Fractal Geometry: Mathematical Foundations and Applications*. John Wiley & Sons, 1990.

[6] Falconer, K., *Techniques in Fractal Geometry*. John Wiley & Sons, 1997.

[7] Fatou, P., Sur l'Itération des fonctions transcendentes entières, *Acta Math.* 47 (1926), 337-370.

[8] Feder, J., *Fractals*. Plenum Press, 1988.

[9] Gardner, M., *Penrose Tiles to Trapdoor Ciphers*, W.H.Freeman and Co., NY, 1989.

[10] Julia, G., Iteration des Applications Fonctionelles, *J. Math. Pures Appl.* (1918), 47-245.

[11] Klebanoff, A. *π in the Mandelbrot Set*, http://www.frii.com/~dboll/mandel.pdf

[12] Mandelbrot, B., *The Fractal Geometry of Nature*, Freeman Co., San Francisco, 1982.

[13] Pesin, Y., *Dimension Theory in Dynamical Systems: Contemporary Views and Applications*, Chicago Lectures in Mathematics, Chicago University Press, 1997.

[14] Wilson, E. , *Gmandel*, http://gmandel.sf.net/

A Some remarks on the sequence $\mathcal{Z}(0, q, c)$

Let $q \in \mathbb{N}$ be an integer number greater than 1.

For each $n \in \mathbb{N}$ let $p_{qn} : c \mapsto Z_n(0, q, c)$ be the map that gives the n-th term in Mandelbrot sequence in terms of c, as defined in section 1.

Remark 1 *p_{qn} can be expressed as a polynomial in c of degree q^n of the form*

$$p_{qn}(c) = \sum_{i=0}^{\frac{q^n-1}{q-1}} K_{q,n,1+i(q-1)} c^{1+i(q-1)} \in \mathbb{N}[c] \tag{5}$$

Thus $p_{qn}(c)$ has $\frac{q^n-1}{q-1}+1$ non-null coefficients.
Proof. Let us prove it by induction on n.
Base case. For $n=0$ we have $p_{q0}(c)=c$, which is clearly of the form (5) with $K_{q01}=1$.

Inductive case. Let $n>0$ and suppose that equation. (5) holds for $n-1$. Let us prove it for n.

We observe that the addition of q numbers congruent with 1 modulus $(q-1)$, i.e. of the form $1+i_j(q-1)$, $j=1,\ldots,q$, is of the form

$$\sum_j (1+i_j(q-1)) = q+\left(\sum_j i_j\right)(q-1) = 1+\left(1+\sum_j i_j\right)(q-1)$$

i.e. it is also congruent with 1 modulus $(q-1)$. Thus by expanding $(p_{q,n-1}(c))^q+c$ we get an expression of $p_{qn}(c)$ of the form (5). ∎

Remark 2 *The coefficients* $\left(K_{q,n,1+i(q-1)}\right)_{i=0}^{\frac{q^n-1}{q-1}}$ *can be calculated recursively:*

$$\begin{aligned} K_{qn1} &= 1 \\ K_{q,n,1+k(q-1)} &= \sum\left\{\prod_{j=1}^{q} K_{q,n-1,1+i_j(q-1)} \,\middle|\, \sum_{j=1}^{q} i_j = k-1\right\} \end{aligned} \tag{6}$$

Proof. It follows just by expanding $p_{q,n}(c)=(p_{q,n-1}(c))^q+c$. ∎

Remark 3 *For every integer $q\geq 2$, and every $n,m\in\mathbb{N}$, if $m\geq n$ then $K_{q,n,1+n(q-1)}=K_{q,n,1+m(q-1)}$. In other words, the n-th polynomial fixes the value of the n-th coefficient in all subsequent polynomials.*

Thus, if the sequence $\{Z_n(0,q,c)\}_{n\geq 0}$ is convergent its limit value can be expressed by the series:

$$S(q,c)=\sum_{n\geq 0} K_{q,n,1+n(q-1)}c^{1+n(q-1)} \tag{7}$$

Remark 4 *The number r_q introduced in eq. (4) is the radius of convergence of series $S(q,c)$ in eq. (7). The sequence $\{r_q\}_{q\geq 2}$ starts with the value $r_2=\frac{1}{4}$, is increasing and besides* $\lim_{q\to+\infty} r_q = 1$.

In table 3 we sketch a few approximated values in the sequence $\{r_q\}_{q\geq 2}$, obtained from our calculations of the corresponding q-Mandelbrot set.

B Calculation of radii r_q

When $q=2$, the minimum circle which is contained entirely within the Mandelbrot set is determined by a real number $r_2=0.25$, indeed the point (0.25,0) is the closest to the origin in the border of M_2. As the order of the generalized Mandelbrot set is increased, a sequence

q	r_q	s_q
2	0.25	0.25
3	0.38490	0.148148
10	0.696837	0.038742
100	0.945003	0.003697
1000	0.992116	0.000368
20000	0.999455	0.000018

Table 3: Some values of r_q and s_q.

of real numbers appears which determines the maximum circle contained within the set. The ratio of this circle can be expressed in the form $r_q = s_q^{\frac{1}{q-1}}$. On the other hand, the radius of the minimum circle containing the set is determined by the term $2^{\frac{1}{q-1}}$. Since there arose indeed the unit roots, as the order of q increases the vertices at the border within a distance of r_q to the origin on the complex plane form a regular $(q-1)$-gon. The table 3 indicates some values of r_q and s_q. Since every positive real number has at least one real and positive $(q-1)$-th root, a spike reflecting the minimum value of r_q always appears on the positive real axis. When $q = 2$ there is only one spike. When $q = 3$, two spikes corresponding to the square roots of s_3 show a symmetry with respect to the origin. On $q = 4$, the three spikes correspond to the three cube roots of s_4. In general, there always will be $q-1$ spikes, symmetrically placed around the origin. Since the distance to each of these to the origin is the same, it is sufficient to analyze the behavior of the spike on the positive real axis. This simplifies the problem greatly since we move from a sequence in $\mathbb{C}$ to $\mathbb{R}$.

Let us examine what happens when $q \to \infty$. There are two cases. Case 1 is when $x_0 \geq 1$, and case 2 is when $0 < x_0 < 1$.

Case 1: $x_0 \geq 1$. $x_1 = \lim_{q\to\infty}(x_0)^q + x_0 = \infty$. Nothing more needs to be said.

Case 2: $0 < x_0 < 1$. $\frac{1}{n} = x_0 : n > 1$. $x_1 = \lim_{q\to\infty}(\frac{1}{n})^q + \frac{1}{n} = \frac{1}{n}$. By induction $x_i = \frac{1}{n}\, \forall i \geq 0$. But $\frac{1}{n} < 1 < \infty$, therefore the sequence converges for all values of $x_0 < 1$.

This implies that the limit of the Mandelbrot set when the exponential $q \to \infty$ is nothing less than the unit circle on the complex plane.