Physica 28D (1987) 358–370
North-Holland, Amsterdam

EFFICIENT COMPUTATION OF JULIA SETS AND THEIR FRACTAL DIMENSION

Dietmar SAUPE
Department of Mathematics, University of California, Santa Cruz, CA 95064, USA and Institut für Dynamische Systeme, Fachbereich Mathematik, Universitat Bremen, 2800 Bremen-33, Fed Rep Germany

Received 27 January 1987
Revised manuscript received 26 May 1987

The computation of the fractal dimension is straightforward using the box-counting method However, this approach may require very long computation times If the Julia set is the connected common boundary of two or more basins of attraction, then a recursive version of the box-counting method can be made storage- and time-efficient The method is also suitable for the computation of the Julia sets We apply the method to verify a result of D Ruelle regarding the dimension of Julia sets of $R(z) = z^2 + c$ for small $c \in \mathbb{C}$, to Newton's method for complex polynomials of degree 3 and to a sequence of Julia sets from the renormalization transformation for hierarchical lattices We also discuss the computation of Julia sets and their information dimension by the inverse iteration method In all examples tested we find that the information dimension is less than the fractal dimension

1. Julia sets and dimension

In many areas of natural sciences such as phase transitions in physics Julia sets are fast becoming an important issue in studying nonlinear phenomena. These studies are typically done as computer experiments using the iteration of certain nonlinear complex functions $R(z)$ The corresponding Julia set can be interpreted as a set of singularities which bounds regions of qualitatively different behaviour of the iterative system.

At the beginning let us briefly recall the definition of Julia sets and some basic facts. If $R(z)$ is a nonlinear rational function in $\mathbb{C}$ then its Julia set is defined as the set of points $z \in \mathbb{C} \cup \{\infty\}$ at which R is not normal in the sense of Montel, i.e there does not exist a neighborhood of z in which $R(z), R^2(z) = R(R(z)), R^3(z) = R(R^2(z)), \ldots$ is an equicontinuous sequence of functions. The Julia set is nonempty and completely invariant under R. Moreover, we have the following properties: Let $z^* = R(z^*)$ be an attractive fixed point of R, i.e. $|R'(z^*)| < 1$ Define the basin of attraction of z^* as $A(z^*) = \{z \in \mathbb{C} \cup \{\infty\} | R^k(z) \to z^* \text{ as } k \to \infty\}$. Then the Julia set J is the boundary of $A(z^*)$:

$$J = \partial A(z^*). \tag{1}$$

Let $z^* = R(z^*)$ be a repelling fixed point of R, i.e. $|R'(z^*)| > 1$. Then the Julia set is the closure of all preimages of z^*:

$$J = \mathrm{cl}\{z \in \mathbb{C} | R^k(z) = z^* \text{ for some } k\}. \tag{2}$$

Let $D \subset \mathbb{C}$ be an open set having a nonempty intersection with the Julia set J. Then there exists an integer n such that

$$J = R^n(D \cap J) \tag{3}$$

In this partial list of properties of Julia sets we have restricted to those facts that are used in the following. For a comprehensive discussion of Julia sets and complex analytic dynamics we refer to the survey [1] and the book [8] The first two of the above properties will be used for the numerical approximation of Julia sets The last fact constitutes the self-similarity of Julia sets: The structure of the whole Julia set is already contained in an arbitrary small fraction of it, which may be one of the facts, that had motivated B. Mandelbrot to call these sets and others with similar properties fractals.

The self-similarity is a homogenuity property and, thus, the notion of dimension is a natural one for Julia sets and it defines a characteristic number attached to any Julia set. Ruelle [10] regards the study of the Hausdorff dimension of Julia sets as one of the fundamental theoretical problems in deterministic chaos in differentiable dynamical systems. Unfortunately it seems to be very hard to obtain rigorous results in that direction. Ruelle's remarkable work [9] stands alone (see section 3.3). In this paper we attempt to design efficient algorithms for the numerical computation of the dimension of Julia sets.

There are two concepts of dimension both of which apply to Julia sets: *metric dimension* and "*probabilistic*" *dimension*, which depends on a natural measure of Julia sets. There are two different approaches to metric dimension: the *Hausdorff dimension* and the *capacity* of sets.

The definition of Hausdorff dimension goes back to F. Hausdorff in 1919. If X is a subset of a metric space and $d>0$ then the d-dimensional outer measure $m_d(X)$ is

$$m_d(X) = \lim_{r\to 0} \inf\left\{ \sum_{i\in I} (\operatorname{diam} S_i)^d \right\},$$

where the infimum is taken over all finite coverings of X by sets S_i of diameter less than $r>0$ There is a unique number $d=d_{\mathrm{H}}$ such that $m_d(X)=\infty$ for $d<d_{\mathrm{H}}$ and $m_d=0$ for $d>d_{\mathrm{H}}$. This number is the Hausdorff dimension

$$d_{\mathrm{H}} = \sup\{ d>0 | m_d(X)=\infty \}.$$

The capacity is a simplified version of the Hausdorff dimension and it lends itself for numerical computation. For the definition of the capacity of a set in Euclidean space let $N(r)$ be the minimum number of cubes of length $r>0$ that are needed to cover the set. Then the capacity d_{C} is defined as

$$d_{\mathrm{C}} = \lim_{r\to 0} \frac{\log N(r)}{\log 1/r}. \tag{4}$$

The capacity of the well known Cantor set which is obtained by consecutive deletion of middle thirds of intervals is easily seen to be equal to $\log 2/\log 3$. From the definition it follows that the Hausdorff dimension d_{H} is a lower bound for the capacity d_{C}:

$$d_{\mathrm{C}} \geq d_{\mathrm{H}}.$$

However, it is believed that both dimensions typically have the same value, which is called the *fractal dimension* d_{F}

$$d_{\mathrm{F}} = d_{\mathrm{C}} = d_{\mathrm{H}}.$$

The probabilistic dimension is based on a natural measure for Julia sets. From property (2) we see that the Julia set can be interpreted as the closure of the union of all points visited by inverse iteration of $R(z)$ starting at a repelling fixed point. Of course we have to consider all branches of $R^{-1}(z)$ in this process. The natural measure is obtained by taking the relative frequency with which the inverse iteration visits a section of the Julia set. More precisely we have [5]: There is an invariant Borel measure μ with support J. For any $x\in \mathbf{C}\cup\{\infty\}$ except for at most two points μ is the weak limit of the $\mu_{m,x}$ where

$$\mu_{m,x} = \frac{1}{d^m} \sum_{z\in R^{-m}(x)} \delta_z, \tag{5}$$

where δ_z is the unit mass distribution concentrated at $z\in\mathbf{C}\cup\{\infty\}$ and d is the degree of the rational function R, i.e. $d=\max(\deg P, \deg Q)$ when $R=P/Q$ and P, Q are relatively prime.

There are several ways to define a probabilistic dimension with respect to a natural measure. In the case of chaotic attractors they will typically yield the same results. For Julia sets we are only considering the *information dimension* d_I which is a generalization of capacity. Again let $N(r)$ be the minimum number of squares of size r needed to cover the Julia set. Then the information dimension d_I is

$$d_I = \lim_{r\to 0} \frac{I(r)}{\log 1/r}, \tag{6}$$

where

$$I(r) = \sum_{1}^{N(r)} P(Q_i) \log \frac{1}{P(Q_i)}$$

and $P(Q_i)$ is the probability contained in the ith square Q_i. $I(r)$ is the information in the sense of information theory which is necessary to specify a point on the Julia set within accuracy r Thus, the dimension d_I measures how fast that information increases as r goes to 0 Note, that the information dimension is equal to the capacity if all probabilities $P(Q_i)$ are equal In general we have $d_I \le d_C$. For certain chaotic attractors (not Julia sets) it is conjectured that both dimensions must be the same [4]. We will test numerically whether this holds for Julia sets alike For a detailed discussion of dimensions in the context of attractors we recommend the survey [4]

In the following we give two efficient algorithms for the computation of the fractal and information dimension. In the course of the actual computation one naturally obtains a list of points approximating the Julia set. Therefore our methods also provide at least a first step to solve the problem posed in [1]· "Design good algorithms to generate pictures of Julia sets."

2. Methods for the computation of Julia sets and dimensions

To compute the fractal dimension of Julia sets we need to know the minimal number of squares $N(r)$ of length or resolution r required to cover the set We first remark that it suffices to only consider squares from a priori fixed grids· Let us denote by $N'(r)$ the number of squares from a fixed grid of resolution r (i e grid size r) that are necessary to cover the Julia set. The Julia set contained in an arbitrarily placed square of size r can be covered by at most 4 squares from a fixed grid of resolution r Thus we have $N'(r) \le 4N(r)$ and in the limit we obtain

$$d_F' = \lim_{r\to 0} \frac{\log N'(r)}{\log 1/r} \le \lim_{r\to 0} \frac{\log 4N(r)}{\log 1/r} = d_F.$$

By definition of $N(r)$ we also have $d_F \le d_F'$ and thus $d_F = d_F'$

Thus we may choose grids of increasing resolution $r = 2^{-k}$, $k = 0, 1, 2, \ldots$ (we say that the resolution increases as $r \to 0$) and ask how many squares have nonempty intersection with the Julia set This number we again call $N(r)$. The display of these squares in black and all others in white then yields a picture of the Julia set in the given resolution. Thus, we call the squares of the grid *pixels*. For the information dimension we must additionally compute the probability with respect to the natural measure of the set which is contained in each pixel

2.1 *Two basic algorithms BSM and IIM*

Two algorithms based on properties (1) and (2) are immediately clear. Assume that the underlying map $R(z)$ has at least two attractive fixed points. (If $R(z)$ has an attractive cycle of least period n, then $R^n(z)$ has at least n attractive fixed points and we may replace R by R^n.) Property (1) states that J is the boundary of the basin of attraction of such a fixed point of R If a pixel has vertices that belong to different basins then it is clear that it must also contain a basin boundary point, i e a point of the Julia set. Conversely, it is clear that given a point in J and $\varepsilon > 0$ we can find a pixel in the ε-neighborhood of that point which has vertices belonging to different basins if only the resolution is sufficiently small.

If the assumption of the existence of at least two attractors is not fulfilled, then we may instead test for Siegel disks, Herman rings or parabolic basins (see [2]). However, we point out that there are cases where the map $R(z)$ has only one basin, e.g. the Julia set of $R(z) = z^2 + \mathrm{i}$ is a dendrite and its complement is the basin of attraction of ∞. The boundary scanning method does not apply to these maps.

After scanning all points of a grid we obtain a list of pixels approximating the Julia set. We follow the notation in [8] and call this algorithm the boundary scanning method (BSM).

If the fixed grid contains N^2 pixels, then the BSM algorithm requires computations for each of the $(N+1)^2$ vertices on the grid. Thus, the compute time is quadrupled when the resolution r is halfed. This leads to unacceptable compute times if N is large, say 10000 as it may have to be for computation of fractal dimension.

The second algorithm is based on property (2): The inverse iteration of R (all branches) starting at a repeller of R produces a dense subset of points in J. If d is the degree of the rational map R, then each point z has d preimages counting multiplicities. Thus, the number of computed preimages n increases rapidly with the maximal number of levels l allowed in this recursion:

$$n = \sum_{k=1}^{l} d^k.$$

Thus, l can only be of moderate size, $l = 24$ with $d = 2$ may already be at the limit

The inverse iteration is most conveniently done in a recursive scheme. For each preimage and each grid under consideration we have to compute the pixel in which the point lies and then increment a counter for that pixel. We propose to work with square grids of increasing size of 2^{2k} pixels, $k = 2,\ldots,k_{\max}$. In order to save storage we do not keep counters for all pixels in memory. Only the relevant counters, i.e. those that correspond to pixels which carry at least one point of the inverse iteration may be stored in a quad tree, each node of which corresponds to a square region in $\mathbf{C}$ and contains four counters which hold the number of preimages that fall into one of the four quadrants of the region. Each node also contains pointers to the four daughter nodes which correspond to the four quadrants.

Upon completion of the inverse iteration the quad tree will be traversed to obtain for each resolution $r = 2^{-k}$ the number $N(r)$ of pixels containing part of the Julia set and the information $I(r)$ based on the approximation of the probabilities by the distribution of the preimages.

The amount of required storage depends on the resolution $r = 2^{-k_{\max}}$ and on the fractal dimension d_F since the number of pixels approximating the Julia set scales as $2^{k_{\max} d_F}$. This approach would allow resolutions up to about 2^{-13} or 2^{-14} depending on the fractal dimension and the available machine memory.

2.2. *The modified boundary scanning method (MBSM)*

Let the basins of attraction for R be numbered from 1 to l and let us call the number of the basin to which a grid point belongs the *label* of that grid point If a pixel has vertices that do not all carry the same label, then the pixel intersects the Julia set and we call it *completely labeled.*

Since label computation is the major time consuming factor in the BSM algorithm we try to avoid it in regions bounded away from the Julia set. This can be done in a straightforward way in the case that the Julia set is connected. We may assume that the connectivity of the Julia set carries over to its approximation in terms of pixels. This is not rigorously so but as the resolution becomes finer our pixel approximations will still converge to the Julia set.

The algorithm proceeds as follows: We maintain a stack of pixels which are candidates for being completely labeled. At the beginning we scan just the boundary of the grid of vertices, obtain a list of completely labeled pixels along the boundary, and push their neighboring pixels onto the candidate stack. If no completely labeled pixels have been found along the boundary, then we must provide a first completely labeled pixel in the interior as a seed, whose neighbors we push onto the stack. Of course, only admissible pixels are pushed onto the stack, i e. pixels that are inside the defined grid and that were not already tested. We then work on the stack processing each pixel in the following three steps.

1) compute all those labels of the vertices of the candidate pixel that are not already known,
2) note whether the pixel is completely labeled;
3) if it is completely labeled, then push all its admissible neighbors onto the stack.

The algorithm is completed on exhaustion of the stack. The notion of neighbors of pixels may be defined in different ways. We have used the eight pixels which are adjacent horizontally, vertically or diagonally. By expanding this list of neighbors we get a version that is more careful in exploiting the connectivity of the Julia set. In order to check if there are any completely labeled pixels missed out by the above algorithm we can simply remove the 'if' clause in step 3 and obtain the usual BSM algorithm

We remark that the assumption on the connectivity of the Julia set can easily be checked if R is a polynomial. The Julia set is connected if and only if no finite critical point, i e. a zero of the derivative R', is attracted by ∞ (see [1]).

The algorithm as described above is useful for an efficient computation of pictures of connected Julia sets in a resolution of 1000 by 1000 or even 2000 by 2000 pixels. For the computation of fractal dimension, however, it is not suitable because its storage requirements exceed the available computer memory on most machines The introduction of a quad tree structure as described in the last section for the storage of only those pixels and labels that have been tested alleviates the storage problem to some extent

We have opted for an alternative scheme with a fixed and very small storage requirement which depends on neither the resolution nor the fractal dimension. Our approach recursively uses the

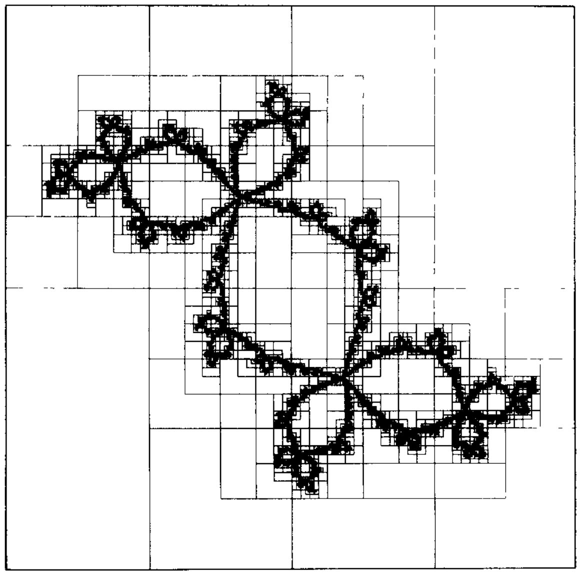
Fig 1 Pixels computed by the modified boundary scanning method (MBSM) for $R(z)=z^2-0\,12+0\,74i$ In each outlined square the recursive subalgorithm is performed

above described algorithm for a grid of relatively small size $N^2, N=2^{k_0}, k_0 \leq 7$ First, the above algorithm is used for computation of completely labeled pixels at the resolution $r=2^{k_0}$ If any pixels were found then the region of interest in $\mathbb{C}$ is subdivided into four quadrants of equal size Each of the quadrants is spanned by grid of again $N^2, N=2^{k_0}$ pixels and this recursive algorithm is invoked for each subregion In each level of the recursion the actual resolution is increased by a factor of 2 A limiting resolution $r=2^{-k_{\max}}$ is prescribed. Let us call this method the modified boundary scanning method (MBSM). It is illustrated in fig 1.

The recursive algorithm may be tuned according to the following remarks.

At each stage certain labels are computed and at the following stages of finer resolution some of those labels have to be recomputed. To avoid this costly redundancy we can simply pass the label array on to the recursive subalgorithm which at the beginning has to set its own label array appropriately.

Moreover, all labels situated along the boundary of the grid must be computed in each stage. This information may not only be used in the next lower calls to the recursive scheme. After the algorithm goes back to a coarser grid and then advances to finer resolutions it may eventually use a grid neighboring another one for which the algorithm had already been completed. Then previously stored boundary labels can be used to set the boundary labels of the present grid thus avoiding recalculations.

The recursive algorithm is not pursued in quadrants that do not contain any completely labeled pixels. This causes the method to loose all pixels in that quadrant that are completely labeled with respect to some finer resolutions. It is the author's experience that only very few pixels are lost in this fashion. In an attempt to avoid this defect we may abort the recursion in a quadrant only if no completely labeled pixels were found in two consecutive levels of resolution.

2.3. *The modified inverse iteration method (MIIM)*

The weakness of the inverse iteration method, namely the failure to render those pixels covering the Julia set and carrying only a relatively small probability with respect to the natural measure, can be partially overcome by an adaptive strategy. It is an approach due to H.W. Ramke (unpublished), and many of the pictures of Julia sets contained in [8] were generated with this method. For completeness we briefly describe the basic idea.

The inverse iteration method would give a very accurate picture of Julia sets if one could allow the recursive scheme to proceed up to levels of hundreds and thousands of preimages of a repeller. To accomplish this in acceptable time we have to cut off the inverse iteration at points where we do not expect any improvement of our approximation of the Julia set by further inverse iteration. For each pixel that contains preimages of the repeller we therefore define a maximal number $N_{\max}$ of preimages allowed for that pixel. Any further preimages that fall into such a pixel are then exempted from further inverse iteration. The numbers $N_{\max}$ chosen will decide on the quality of the result and also how long the method takes. The simplest approach is to choose a constant $N_{\max}$ for all pixels. An improved strategy is to take into account how much the pixel is stretched or contracted by a branch of the inverse map R^{-1} or to make use of an approximation of the invariant measure.

One advantage of this method over the boundary scanning method is that it is applicable also in cases where the other method is not, namely for Julia sets that do not separate two or more components of its complement. Examples are not connected Julia sets or dendrite like Julia set as for $R(z) = z^2 + \mathrm{i}$.

2.4. *Calculating dimensions*

For the computation of fractal dimension the boundary scanning methods (BSM, MBSM) provide the numbers $N(r)$ of pixels of resolutions $r = 2^{-k}$, $k = k_0, \ldots, k_1$ that intersect the Julia set. The sequence

$$\delta_{\mathrm{F},k} = \frac{\log(N(2^{-k}))}{k \log 2}$$

converges to the fractal dimension d_{F}. However, convergence is very slow since in practise we have for small $r > 0$

$$N(r) \approx c\left(\frac{1}{r}\right)^{d_{\mathrm{F}}}, \quad c > 0$$

and, thus

$$\delta_{\mathrm{F},k} - d_{\mathrm{F}} \approx \frac{\log c}{k \log 2}.$$

This error goes to 0 as slowly as $1/k$. This observation leads to the improved estimates

$$d_{\mathrm{F},k} = \frac{1}{\log 2} \log \frac{N(2^{-k})}{N(2^{-k+1})}, \tag{7}$$

which also converge to d_F.

The actual computation of the information dimension d_I of Julia sets is more problematic since the approximation of

$$d_I = \lim_{r\to 0} \frac{I(r)}{\log 1/r}$$

is twofold: In addition to the limit in the formula we have that the computation of $I(r)$ requires knowledge of the natural measure which again is defined as a limit. The probability P contained in a pixel $Q \subset \mathbb{C}$ is

$$P(Q) = \lim_{n\to\infty} P_n(Q), \quad P_n(Q) = \frac{m_n(Q)}{s_n},$$

where $m_n(Q)$ is the number of preimages $R^{-k}(z)$ of a repeller z with $k \le n$ that fall into the pixel Q and s_n is the total number of preimages of order n or less, i.e.

$$s_n = \sum_{k=0}^{n} d^k,$$

where $d = \deg R$. Let us define $I_n(r)$ as the approximate information obtained at resolution r and using n levels of inverse iteration:

$$I_n(r) = \sum_{i=1}^{N(r)} P_n(Q_i) \log \frac{1}{P_n(Q_i)}.$$

Experimental studies suggest that the convergence of $I_n(r)$ to $I(r)$ is linear. Therefore we propose to use the extrapolation

$$I \approx I_n - \frac{(I_n - I_{n-1})^2}{I_n - 2I_{n-1} + I_{n-2}}.$$

Finally, approximations for d_I are obtained using the same acceleration of convergence as in the case of fractal dimension, namely

$$d_{I,k} = \frac{1}{\log 2}\left(I(2^{-k}) - I(2^{-k+1})\right). \tag{8}$$

3. Numerical results

Computer programs for all algorithms discussed in this paper have been developed on a UNIX based graphics workstation in the C programming language. The length of the code is about 850 lines for the MBSM algorithm which includes the BSM algorithm as a special case and 750 lines for the IIM and MIIM algorithms.

3.1. *Three test functions*

In order to test the proposed algorithms we choose examples from three families of Julia sets. The degrees of the rational maps are 2, 3 and 4.

Example 1. *The quadratic map*

The best known dynamics of iterations of a rational function are for the polynomial

$$R(z) = z^2 + c,$$

where $c \in \mathbb{C}$ is a complex parameter. It is also the only map where something is known about the fractal dimension of its Julia sets. For $c = 0$ we have that the Julia set J is the unit circle which has dimension 1. For parameters $c = w/2 - w^2/4$ where $|w| < 1$ one of the fixed points of R is an attractor. The boundary of the basin of attraction of the fixed point, thus the Julia set is a Jordan curve which bounds the basin of the fixed point from the basin of ∞. Ruelle has shown [9] that the Hausdorff dimension of J is given by

$$d_H = 1 + \frac{|c|^2}{4\log 2} + \text{h.o.t.} \tag{9}$$

Example 2. *Relaxed Newton method*

In several papers (e.g. [2, 7, 8, 11]) pictures of Julia sets of the relaxed Newton iteration

$$N_h(z) = z - h\frac{p(z)}{p'(z)}$$

applied to polynomials $p(z)$ and with $h > 0$ have appeared. Motivated by images for $p(z) = z^3 - 1$

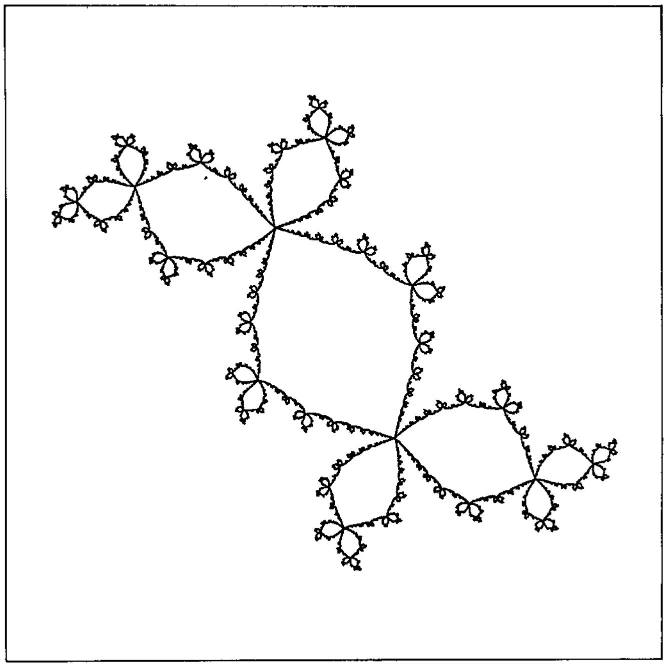

Fig 2 Julia set for R_1 $d_F = 1\ 39$

and $h = 0.5, 1.0, 1.5$ we conjectured in [7] that the dimension of the Julia set is an increasing function with h. In [2, 11] the family of polynomials of degree 3 which can be restricted to

$$p_\lambda(z) = z^3 + (\lambda - 1)z - \lambda,$$

with $\lambda \in \mathbb{C}$, has been studied. Here we choose a priori a fixed parameter $\lambda \in \mathbb{C}$ and compute the Julia sets and their dimensions for several values of h.

Example 3. *Renormalization group transformation for hierarchical lattices*

In 1983 the French physicists B. Derrida, L. DeSeze and C. Itzykson studied the renormalization of temperature for a hierarchical model of magnetism [3]. They discovered that complex phase boundaries can be identified as Julia sets of the renormalization transformation of temperature. [8] contains a series of such Julia sets for the transformation

$$R(z) = \left(\frac{z^2 + q - 1}{2z + q - 2} \right)^2,$$

where q is the number of Potts states in the original model but is treated as a real or complex parameter.

3.2. *Calculating Julia sets*

From each of the examples we choose one function and compute their Julia sets with a resolution of 1024 by 1024 pixels. We are using

$$R_1(z) = z^2 - 0.12 + 0.74\mathrm{i},$$

$$R_2(z) = N_h(z) = z - h\frac{p(z)}{p'(z)},$$

$$p(z) = z^3 + (\lambda - 1)z - \lambda,$$

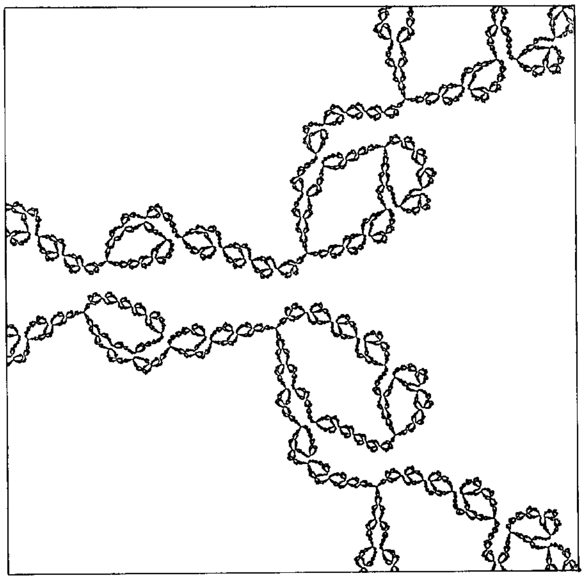

Fig 3 Julia set for R_2 $d_F = 1\ 54$

with $h = 1.5$ and $\lambda = 1.1 - 0.1\mathrm{i}$, and

$$R_3(z) = \left(\frac{z^2 + q - 1}{2z + q - 2}\right)^2, \quad q = 1\ 2$$

Figs. 2 to 4 show the corresponding Julia sets computed by the MBSM algorithm. For R_1 we have a periodic attractor of period 3 besides the attractor ∞ R_2 is the relaxed Newton method for $p(z)$ which exhibits 3 attractive fixed points, the roots of p R_3 has a Julia set of very high fractal dimension. The fractal dimensions are 1.39 for R_1, 1.54 for R_2 and 1.81 for R_3

Table I lists the performance of the modified boundary scanning method versus the simple boundary scanning method. We note that in both cases the algorithms produce the same number of pixels approximating the Julia sets except for example 1 where a few pixels were missed, a small price to pay for the savings of roughly three quarters of compute time Naturally, these savings decrease with increasing fractal dimension. Thus, in example 3, where the dimension is 1.81, the MBSM algorithm needed three quarters of the time that the BSM algorithm used.

We also implemented the simplest version of the modified inverse iteration method (fixed $N_{\max}$) and tested it on the three examples In order to apply the method to the relaxed Newton method (example 2), a coordinate transformation may be applied such that the unbounded Julia set is mapped to a bounded domain in $\mathbb{C}$. This is to ensure that no inverse iterations are lost by means of points which lie outside of the region discretized by the grid We apply the linear fractional (Moebius) transformation $M(z) = z/(z - 1)$ which maps ∞ to 1 and the attractive fixed point 1 to ∞ (see fig 5).

For examples 1 and 2 we find that increasing the threshold $N_{\max}$ as expected improves the quality of the result which we can quantify as the number of pixels found. The pictures obtained are comparable to those derived from the MBSM algorithm. However, in example 3 the relationship between $N_{\max}$ and the number of pixels found is not clear at all (see table II). The best result is for $N_{\max} = 5$, see fig. 6 We note, that in this example the MBSM algorithm (fig. 4) clearly outperformed the MIIM algorithm

3 3. *Calculating fractal dimension*

In this section we are applying the MBSM algorithm to compute fractal dimensions For the first test function $R(z) = z^2 + c$ we choose 53 parameters $c = re^{\mathrm{i}\phi}$ such that R has an attractive fixed point besides ∞. Thus, these parameters are taken from the cardiod shaped main body of the Mandelbrot set Ruelle's formula (9) yields a second order approximation of the fractal dimension and it is the intention of this experiment to check the formula. We use a maximal resolution of 2^{-14} i.e. 16384 by 16384 pixels. Table III lists the results. Each dimension was computed as the average over the last three obtained approximations in the sense of (7), i.e. for the resolutions 2^{-14}, 2^{-13}, 2^{-12}. For fixed r the measured fractal dimension varies with ϕ, but on the average we find that Ruelle's formula is quite accurate. It differs from the measured average by at most 0.01 for $r \leq 0\ 4$.

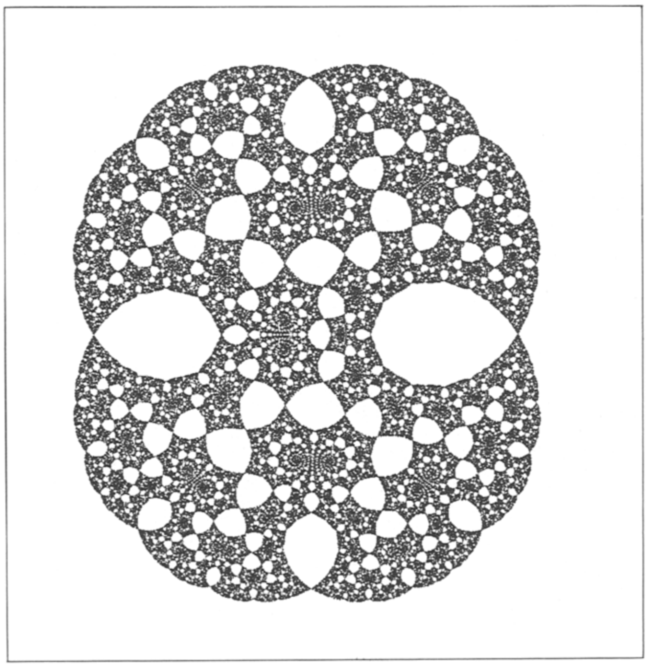

Fig 4 Julia set for R_3 $d_F = 1\ 81$ Compare with fig 53e in [8]

Table I
Performance of MBSM versus BSM algorithms The numbers of iterations are in millions

	Example 1		Example 2		Example 3	
	Pixels	Iterations	Pixels	Iterations	Pixels	Iterations
MBSM	20351	1 3	59519	2 5	233400	11 0
BSM	20379	5 0	59519	10 4	233400	15 1

Table II
Number of pixels found by modified inverse iteration (MIIM) with N_{max} as a parameter in the method

N_{max}	Example 1	Example 2	Example 3
2	14153	32809	55900
5	20442	47819	76468
10	22656	52599	20819
20	23427	56172	22828

The performance of the MBSM algorithm for the example 2 (relaxed Newton method for degree 3 polynomial) is summarized in table IV and illustrated in figs. 7 and 8. Even though we have chosen a maximal resolution of only 2^{-12} (4096 by 4096 pixels) the computed dimensions seem to be very accurate in most cases. The results confirm our conjecture that the fractal dimension is monotonically increasing as h grows from 0 to 2.

Results for the third example (renormalization group transformation) are reported likewise in table V. The Potts parameter q chosen correspond to figures published in [8].

3.4. *Calculating information dimension*

In comparison to the computation of fractal dimension we find that the information dimension is much harder to calculate. The inverse iteration method has to produce many millions of points

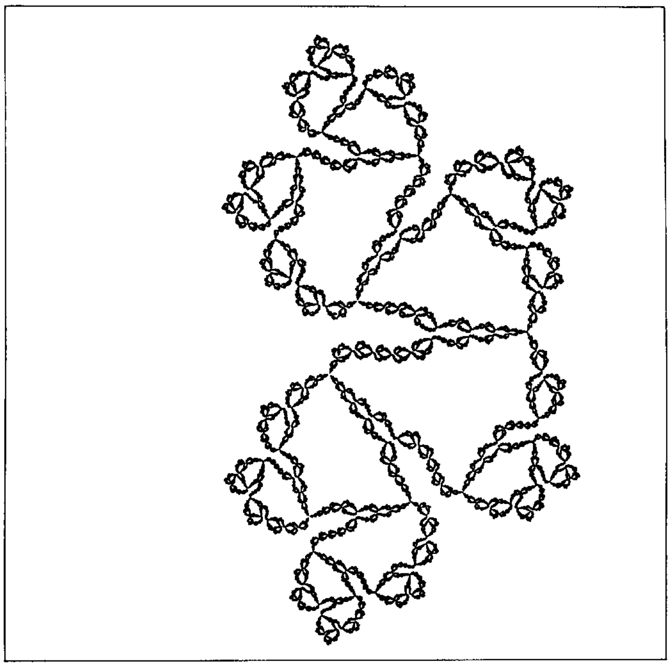

Fig 5 Julia set of R_2 in transformed coordinates obtained by MIIM ($N_{max} = 20$)

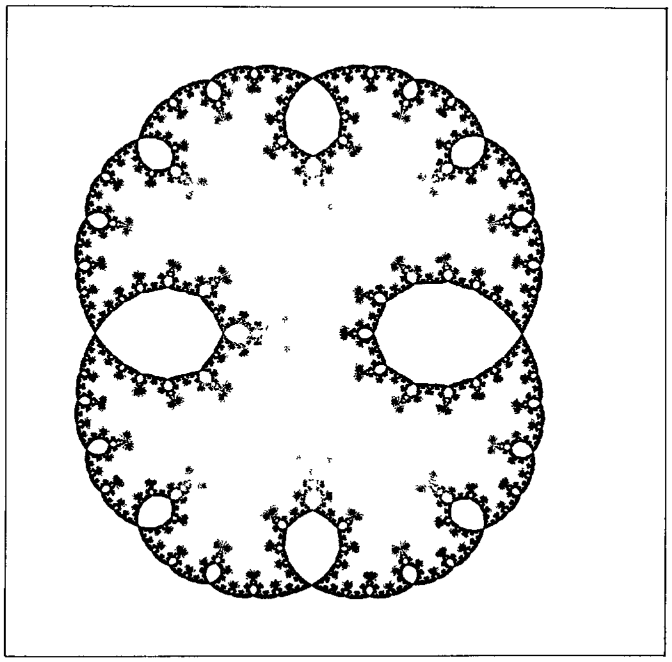

Fig 6 Julia set of R_3 obtained by MIIM ($N_{max} = 5$)

Table III
Fractal dimensions of Julia sets of $R(z) = z^2 + re^{i\phi}$ Maximal resolution is 2^{-14} The results agree with the expected values from Ruelle's formula (last column) The computation of the dimension for a parameter c required between 6 and 346 million iterations of R CPU time varied between 22 and 228 minutes

	ϕ											
r	0	$\frac{1}{9}\pi$	$\frac{2}{9}\pi$	$\frac{3}{9}\pi$	$\frac{4}{9}\pi$	$\frac{5}{9}\pi$	$\frac{6}{9}\pi$	$\frac{7}{9}\pi$	$\frac{8}{9}\pi$	π	Average	$1+\frac{r^2}{4\log 2}$
0 1	1 006	1 005	1 007	1 004	1 003	1 002	1 003	1 002	1 002	1 001	1 004	1 004
0 2	1 022	1 024	1 023	1 015	1 017	1 011	1 014	1 007	1 009	1 010	1 015	1 014
0 3		1 086	1 052	1 044	1 039	1 033	1 029	1 026	1 025	1 024	1 040	1 033
0 4			1 143	1 091	1 070	1 056	1 051	1 049	1 048	1 050	1 070	1 058
0 5				1 194	1 131	1 106	1 091	1 081	1 075	1 070	1 107	1 090
0 6					1 329	1 188	1 156	1 131	1 119	1 114	1 170	1 130
0 7								1 270	1 225	1 182	1 226	1 177

Table IV
Fractal dimension for $R(z) = N_h(z) = z - hp(z)/p'(z)$, where $p(z) = z^3 + (\lambda - 1)z - \lambda$, $\lambda = 1\ 1 - 0\ 1i$ Maximal resolution is 2^{-12} Dimensions are computed as averages over the approximations for resolutions $2^{-10}, 2^{-11}, 2^{-12}$ The last column shows the standard deviation σ for this average Iterations are in millions One million of them typically took 5 to 6 minutes of CPU time

h	Iter	d_F	σ	h	Iter	d_F	σ
0 1	7 5	1 242	0 046	1 1	10 8	1 443	0 002
0 2	6 4	1 254	0 019	1 2	12 5	1 464	0 003
0 3	6 2	1 290	0 015	1 3	15 2	1 485	0 002
0 4	6 3	1 309	0 015	1 4	18 7	1 512	0 004
0 5	6 5	1 326	0 015	1 5	24 0	1 536	0 003
0 6	6 9	1 345	0 011	1 6	31 2	1 565	0 003
0 7	7 2	1 360	0 003	1 7	44 8	1 603	0 002
0 8	7 7	1 369	0 024	1 8	71 4	1 638	0 002
0 9	8 5	1 401	0 001	1 9	146 5	1 678	0 002
1 0	9 6	1 424	0 002				

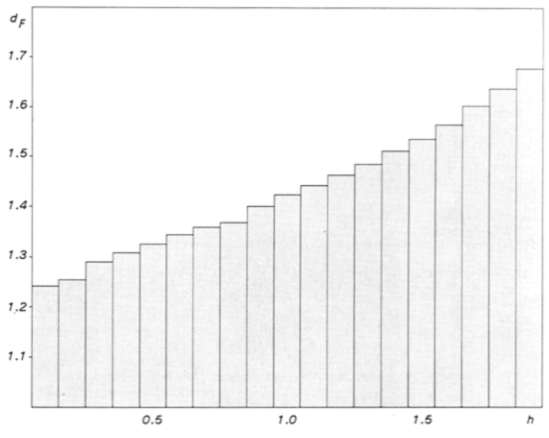

Fig 7 Fractal dimension of Julia sets of N_h as a function of h

before convergence of the information $I(r)$ is achieved. Therefore we can only report results for a few test functions which are summarized in table VI. We first remark that in all cases we have that the information dimension is significantly less than the fractal dimension. Comparing the two cases of example 1 we surprisingly find that although the fractal dimension for the second function is much higher both information dimensions are about the same, namely 1.0. Moreover, even though we have gone through two additional levels of recursion in the second function, i.e. we considered four times as many points (134.2 million), we note that the total number of pixels covering these points is less than the number for the first function.

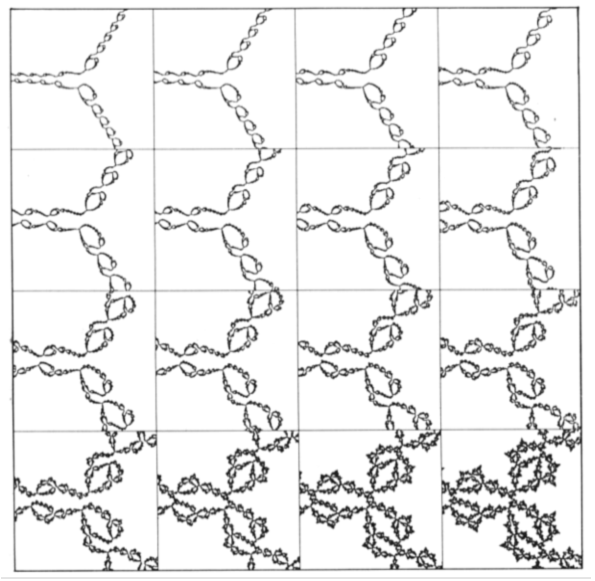

Fig 8 Julia sets for N_h, $h = 0\ 4, 0\ 5, \quad , 1\ 9$

Table V
Fractal dimension of Julia sets of renormalization transformation Maximal resolution is 2^{-12} Iterations are in millions One million of them typically took 2 to 3 minutes of CPU time

q	Iter	d_F	σ	q	Iter	d_F	σ
− 1 0	12 6	1 206	0 007	2 0	68 7	1 566	0 002
− 0 1	49 8	1 334	0 001	2 5	56 7	1 526	0 002
0 0	18 0	1 077	0 002	2 9	58 1	1 525	0 003
1 0	14 1	1 264	0 010	3 0	61 0	1 535	0 005
1 2	582 2	1 807	0 002	3 1	49 0	1 499	0 003
1 6	161 4	1 706	0 001	4 0	18 9	1 336	0 004

Acknowledgements

Thanks are due to Silicon Graphics, Inc. who generously provided an IRIS 3030 workstation on which this research has been carried out. The author also thanks Heinz-Otto Peitgen for supporting this work in all respects.

References

[1] P Blanchard, Complex analytic dynamics on the Riemann sphere, Bull Am Math Soc 11 (1984) 85–141
[2] J Curry, L Garnett and D Sullivan, On the iteration of rational functions Computer experiments with Newton's method, Commun Math Phys 91 (1983) 267–277
[3] B Derrida, L DeSeze and C Itzykson, Fractal structure of zeroes in hierarchical models, J Stat Phys 33 (1983) 559–569
[4] J D Farmer, E Ott and J A Yorke, The dimension of chaotic attractors, Physica 7D (1983) 153–180
[5] M J Ljubich, Entropy properties of rational endomorphisms of the Riemann sphere, Ergod Th & Dynam Sys 3 (1983) 251–385
[6] B B Mandelbrot, The Fractal Geometry of Nature, (Freeman, San Francisco, 1982)
[7] H O Peitgen, D Saupe and F v Haeseler, Cayley's problem and Julia sets, Math Intelligencer 6(2) (1984) 11–20
[8] H O Peitgen and P Richter, The beauty of fractals (Springer, Berlin, 1986)
[9] D Ruelle, Repellers for real analytic maps, Ergod Th & Dynam Sys 2 (1982) 99–108
[10] D Ruelle, Five turbulent problems, Physica 7D (1983) 40–44
[11] D Saupe, Continuous versus discrete Newton's method–A case study, to appear in Acta Appl Math (1987)

Table VI
Information and fractal dimension of Julia sets

	Example 1 $c = -0\ 12 + 0\ 74i$	Example 1 $c = 0\ 32 + 0\ 043i$	Example 2 $\lambda = 1\ 1 + 0\ 1i$ $h = 1\ 5$	Example 3 $q = 1\ 2$
Recurs levels	23	25	13	12
Points (mill)	33 5	134 2	7 2	89 5
Pixels (thsds)	186 2	131 0	135 3	125 8
Resolution (pix)	8192	8192	4096	2048
CPU time (h)	5 5	21 4	15 3	17 0
d_F	$1\ 390 \pm 0\ 004$	$1\ 562 \pm 0\ 002$	$1\ 536 \pm 0\ 003$	$1\ 807 \pm 0\ 002$
d_I	$0\ 97 \pm 0\ 04$	$1\ 00 \pm 0\ 01$	$1\ 23 \pm 0\ 02$	$1\ 28 \pm 0\ 03$