PSEUDORANDOM NUMBERS FOR CONFORMAL MEASURES

MANFRED DENKER[1], JINQIAO DUAN[2] AND MICHAEL MCCOURT[2]

1. MATHEMATICS DEPARTMENT
PENNSYLVANIA STATE UNIVERSITY STATE
COLLEGE, PA 16802, USA
E-MAIL: DENKER@MATH.PSU.EDU

2. DEPARTMENT OF APPLIED MATHEMATICS
ILLINOIS INSTITUTE OF TECHNOLOGY, CHICAGO, IL 60616, USA
E-MAIL: DUAN@IIT.EDU

Abstract. We propose a new algorithm for generating pseudorandom (pseudo-generic) numbers of conformal measures of a continuous map T acting on a compact space X and for a Hölder continuous potential $\phi : X \to \mathbb{R}$. In particular, we show that this algorithm provides good approximations to generic points for hyperbolic rational functions of degree two and the potential $-h \log |T'|$, where h denotes the Hausdorff dimension of the Julia set of T.

1. Introduction

Conformal measures for rational maps were introduced by Sullivan ([13]) in 1983 following ideas of Patterson ([12]) for the case of limit sets of Fuchsian groups. Existence and uniqueness of such measures has been shown in [6] for a wide class of rational maps. These measures are in general singular with respect to Lebesgue measure and have no explicitly computable distribution function. There are a few papers dealing with the numerical computation of these (mostly fractal) measures (e.g. [3]), but there is no work done concerning the construction of generic points according to the following definition.

Definition 1.1. *Let (X, T) be a continuous dynamical system on a compact space X and let ν be a T-invariant probability measure on the Borel field of*

Date: April 17, 2009 (Revised version).

2000 *Mathematics Subject Classification.* Primary 37A50; Secondary 37F10, 37F15, 86A05, 60H15.

Key words and phrases. Generic point, complex dynamical system, conformal measure, invariant measure, Julia set.

X. A point $x \in X$ is called generic if for every continuous function $h \in C(X)$

$$\lim_{n\to\infty} \frac{1}{n}\sum_{k=0}^{n-1} h(T^k(x)) = \int h d\nu. \tag{1.1}$$

The convergence rate in this theorem can be arbitrarily slow, as discussed in [11]. It depends on the function h. A probabilistic error bound can be obtained from the central limit theorem or large deviation results; see [4] for a general description of the central limit theorem problem in dynamical systems, and in particular [5] about this issue for rational functions.

Accordingly, we call a point $y \in X$ pseudo-generic for ν if the equation (1.1) holds up to some prescribed precision or error. The construction of pseudorandom numbers by the linear congruential method is also based on the iteration theory of maps of the interval. These points are as well pseudo-generic, hence one may use the notion of pseudorandom numbers as well in the present situation.

The aim of this note is to define and analyze an algorithm for computable pseudorandom points. In many applications, a sequence generated by the iteration of a pseudo-generic point will produce points which can be viewed as asymptotically independent realizations of independent identically distributed random variables. This follows whenever the map is a weakly dependent sequence of random variables.

Let X be a compact metric space and $T : X \to X$ be continuous. A conformal measure m for a continuous function $\phi \in C(X)$ is a probability measure satisfying

$$m(T(A)) = \int_A \exp[\phi(x)]m(dx) \tag{1.2}$$

for every measurable set A with the property that T restricted to A is invertible. This definition is equivalent to the requirement that

$$\int g(x)m(dx) = \int_A g(T(x)) \exp[\phi(x)]m(dx),$$

for every bounded continuous function g and any measurable set A such that T is invertible on A and the support of g is contained in $T(A)$. We shall call (1.2) the conformal equation. Examples for such measures are provided by rational maps (see [6] among others) or self-similar measures on fractal sets (see [9] among others).

Given an invariant measure μ, equivalent to a conformal measure m, one can construct a pseudo-generic point by the method of least square estimation, i.e.

by minimizing

$$\sum_{g\in\mathfrak{G}}\left(\frac{1}{n}\sum_{k=0}^{n-1}g(T^k(x))-\int g d\mu\right)^2$$

over a suitable subclass $\mathfrak{G}$. However, the integral involved is not known and has to be computed by other means. It can be approximated in some cases using the Perron-Frobenius operator

$$Ph(x)=\sum_{T(y)=x}h(y)\exp[-\phi(y)]$$

and the projection to the eigenspace of the maximal eigenvalue of this operator. Here we follow another approach using a discretized version of the conformal equation together with a least square estimate. In this way no integral or calculation of eigenspaces is involved in the algorithm. The algorithm is explained and analyzed in section 2 in general terms.

Sections 3 and 4 are devoted to the special case of hyperbolic rational functions of degree two. We demonstrate how the algorithm is implemented in this case. Other cases can be investigated in a similar way.

The algorithm requires the computation of points in X and the density of the invariant measure μ with respect to the conformal measure m at specific points. Hyperbolic rational maps on the Riemann sphere $S^2=\overline{\mathbb{C}}$ are characterized by the requirement that their Julia sets $J(T)$ does not contain parabolic or critical points (see [1]). It is known that these maps have a unique conformal measure m for every Hölder-continuous potential $\phi: J(T)\to\mathbb{R}$ ([13]), and that there is a unique equivalent, ergodic and T-invariant probability measure μ. This property will guarantee the the pseudo-generic points are approximating integrals with respect to the invariant measure and that the Perron-Frobenius operator can be used to find the density $d\mu/dm$ at specific points. Unfortunately, this operator requires to calculate the Hausdorff dimension of $J(T)$ which we do here numerically. Moreover, since repelling periodic points are dense in $J(T)$ we are able to construct dense sets of points in the Julia set.

It is important to remark that the algorithm needs precise numerical calculations. We discuss this issue in Section 4.

2. Least squares and the conformal equation

In this section we describe an algorithm leading to explicitly computable pseudorandom points for a dynamical systems. We start listing the assumptions we impose to hold: Let $T: X\to X$ be a continuous map on some compact metric space X with metric $d(\cdot,\cdot)$ and let $\phi\in C(X)$ be a continuous function. Assume the following conditions to hold:

A. (1) There exists a unique conformal measure m for T and ϕ.
   (2) There exists a unique ergodic, T-invariant measure μ which is equivalent to m.
   (3) The Radon-Nikodym derivative $f(x) = \frac{dm}{d\mu}$ has a continuous version defined on X with modulus of continuity $\omega : \mathbb{R} \to \mathbb{R}$.

B. For each $n, p \in \mathbb{N}$, $p \geq n$, there are finite sets $X_{n,p} \subset X$, finite sets $\mathfrak{A}_n$ of measurable subsets of X and continuous functions $g_A \in C(X)$ $(A \in \mathfrak{A}_n)$ such that
   (1) Every set $A \in \mathfrak{A}_n$ satisfies $A \in \sigma(\mathfrak{A}_{n+1})$ (i.e. is a union of elements in $\mathfrak{A}_{n+1}$) and every point in X lies in at most a^* elements form $\mathfrak{A}_n$, where a^* is independent of n.
   (2) $d_n := \sup_{A \in \mathfrak{A}_n} \max\{\operatorname{diam}(A), \operatorname{diam}(T(A))\} \to 0$ as $n \to \infty$.
   (3) $\operatorname{supp}(g_A) \subset T(A)$ and $0 \leq g_A \leq 2$.
   (4) The sigma fields $\sigma(\{g_A : A \in \mathfrak{A}_n\})$ and $\sigma(\{g_A \circ T \cdot 1_A : A \in \mathfrak{A}_n\})$ generate the Borel field of X as $n \to \infty$. Each g_A with $a \in \mathfrak{A}_n$ is approximated arbitrarily close by a linear combination of functions in $\mathfrak{A}_{n+l}$ for $l \geq 1$ sufficiently large.
   (5) For each n, $X_{n,p} \subset X_{n,p+1}$ and
$$D_{n,p} = \sup_{x \in X} \inf_{y \in X_{n,p}} \max_{1 \leq i \leq p} d(T^i(x), T^i(y)) \to 0$$
   as $p \to \infty$.

The algorithm for fixed $n \in \mathbb{N}$ assumes the existence of m, μ, $\mathfrak{A}_n$, g_A $(A \in \mathfrak{A}_n)$ and the sets $X_{n,p}$. It proceeds as follows:

(1) Choose $z_A \in A$ for $A \in \mathfrak{A}_n$ and compute $f(z_A)$ and $f(T(z_A))$.
(2) Let $p = n$. For $x \in X_{n,p}$ compute $\beta_n^2(x)$ by
$$\sum_{A \in \mathfrak{A}_n} \left(f(T(z_A)) \sum_{k=0}^{p-1} g_A(T^k(x)) - f(z_A) \sum_{k=0}^{p-1} g_A(T^{k+1}(x)) e^{\phi(T^k(x))} \right)^2 . \tag{2.1}$$
(3) If $\min_{x \in X_{n,p}} \frac{1}{p^2} \beta_n^2(x) \leq 5a^* \omega(d_n)$ stop and go to step 4, if not set $p = p+1$ and continue with step 2.
(4) Let $\beta_n^2 = \min_{x \in X_{n,p}} \beta_n(x)$. Choose $x_n^* \in X_{n,p}$ minimizing this expression, i.e.
$$\beta_n^2(x_n^*) = \beta_n^2.$$

Remark 2.1. *(1) The algorithm requires to apply several subroutines explained by examples in the following sections:*
- *Calculation of the sets $X_{n,p}$ and the distances $D_{n,p}$ for each fixed n.*
- *Calculation of the sets $\mathfrak{A}_n$, d_n and a^*.*
- *Calculation of the functions g_A.*
- *Calculation of the density at points z_A and $T(z_A)$.*

(2) There is a simpler algorithm which may not work in general but is easier to implement and used in later sections. In this case all sets A have no mass on their boundaries (because these will be a finite union of points and the conformal measure has no atoms).

The simplification puts all functions g_A to indicator functions, $g_A = 1_{T(A)}$ and uses only one set $X_{n,n}$.

We need to show that the algorithm stops eventually and that the resulting points x_n^* are pseudorandom. This will be accomplished in the following two propositions.

Proposition 2.2. *The algorithm stops eventually.*

Proof. For $x \in X_{n,p}$ we have

$$\begin{aligned}
&\sqrt{\frac{1}{p^2}\beta_n^2(x)}\\
= &\left[\sum_{A\in\mathfrak{A}_n}\left(f(T(z_A))\sum_{k=0}^{p-1} g_A(T^k(x)) - f(z_A)\sum_{k=0}^{p-1} g_A(T^{k+1}(x))e^{\phi(T^k(x))}\right)^2\right]^{1/2}\\
= &\left[\sum_{A\in\mathfrak{A}_n}\left(f(T(z_A))\sum_{k=0}^{p-1} g_A(T^k(x)) - \int g_A dm\right.\right.\\
&\left.\left.+\int_A g_A(T(u))e^{\phi(u)}m(du) - f(z_A)\sum_{k=0}^{p-1} g_A(T^{k+1}(x))e^{\phi(T^k(x))}\right)^2\right]^{1/2}\\
\leq &\left[\sum_{A\in\mathfrak{A}_n}\left(f(T(z_A))\sum_{k=0}^{p-1} g_A(T^k(x)) - f(T(z_A))\int g_A d\mu\right.\right.\\
&\left.\left.+ f(z_A)\int_A g_A(T(u))e^{\phi(u)}\mu(du) - f(z_A)\sum_{k=0}^{p-1} g_A(T^{k+1}(x))e^{\phi(T^k(x))}\right)^2\right]^{1/2}\\
+ &\left[\sum_{A\in\mathfrak{A}_n}\left(f(T(z_A))\int g_A d\mu - \int g_A f d\mu\right)^2\right]^{1/2}\\
+ &\left[\sum_{A\in\mathfrak{A}_n}\left(f(z_A)\int_A g(T(u))e^{\phi(u)}\mu(du) - \int_A g_A(T(u))e^{\phi(u)}f(u)\mu(du)\right)^2\right]^{1/2}\\
\leq &\left[\sum_{A\in\mathfrak{A}_n}\left(f(T(z_A))\sum_{k=0}^{p-1} g_A(T^k(x)) - f(T(z_A))\int g_A d\mu\right.\right.\\
&\left.\left.+f(z_A)\int_A g_A(T(u))e^{\phi(u)}\mu(du) - f(z_A)\sum_{k=0}^{p-1} g_A(T^{k+1}(x))e^{\phi(T^k(x))}\right)^2\right]^{1/2}\\
+ &\ 4a^*\omega(d_n).
\end{aligned}$$

Since μ is ergodic there exists a generic point for μ. Let z denote such a point. Choose $x \in X_{n,p}$ such that $d(T^i(z), T^i(x)) \leq D_{n,p}$, for each $1 \leq i \leq p$. Then for any g_A by the triangle inequality

$$\begin{aligned} & |\int g_A d\mu - \frac{1}{p}\sum_{k=0}^{p-1} g_A(T^k(x))| \\ \leq \quad & |\int g_A d\mu - \frac{1}{p}\sum_{k=0}^{p-1} g_A(T^k(z))| + \omega_{g_A}(D_{n,p}) \end{aligned}$$

which converges to 0 as $p \to \infty$, where ω_h denotes the modulus of continuity for the function h. Likewise

$$\begin{aligned} & |\int_A g_A(T(u))e^{\phi(u)}\mu(du) - \frac{1}{p}\sum_{k=0}^{p-1} e^{\phi(T^k(x))} g_A(T^{k+1}(x))1_A(T^k(x))| \\ \leq \quad & |\int_A g_A(T(u))e^{\phi(u)}\mu(du) - \frac{1}{p}\sum_{k=0}^{p-1} e^{\phi(T^k(x))} g_A(T^{k+1}(x))1_A(T^k(z))| + \omega_{e^\phi g_A \circ T}, \end{aligned}$$

which tends to 0 as well as $p \to \infty$ (note that the support of $g_A \circ T$ is inside of A, so that the discontinuity of A is of no relevance). The indicator of A has to be kept here. If g_A is the indicator of $T(A)$, then $g_A \circ T(x)$ is one if and only if x is a preimage of a point in $T(A)$. But there are more than the points in A as preimages and we want only those in A.

Since $\mathfrak{A}_n$ is a finite set, not changing with p, we see that

$$\limsup_{p\to\infty} \frac{1}{p^2}\beta_n^2 \leq 4a^*\omega(d_n),$$

which implies that the algorithm stops eventually. □

Proposition 2.3. *Let x_n^* and $p = p(n) \geq n$ ($n \in \mathbb{N}$) be constructed according to the algorithm, and assume that A and B hold. Then for every continuous function g we have*

$$\lim_{n\to\infty} \frac{1}{p(n)} \sum_{k=0}^{p(n)-1} g(T^k(x_n^*)) = \int g d\mu.$$

Proof. Define

$$\nu_n = \frac{1}{p(n)} \sum_{k=0}^{p(n)-1} \delta_{T^k(x_n^*)},$$

where δ_z denotes the point mass in $z \in X$. Then $\{\nu_n : n \in \mathbb{N}\}$ is relatively compact in the weak topology of measures. Let ν be an accumulation point.

Then ν is an invariant measure and for $A \in \mathfrak{A}_n$

$$
\begin{aligned}
& |\int g_A f d\nu_{n+l} - \int_A e^{\phi(u)} f g_A(T(u)) \nu_{n+l}(du)| \\
\leq & \sum_{g_B : b \in \mathfrak{B} \subset \mathfrak{A}_{n+l}} |f(T(z_B)) \int g_B d\nu_{n+l} - f(z_B) \int_B e^{\phi(u)} g_B(T(u)) \nu_{n+l}(du)| + o_l(1) \\
\leq & \beta_{n+l} + o_l(1).
\end{aligned}
$$

Therefore, letting $l \to \infty$ along a suitable subsequence, $d\widetilde{m} = f d\nu$ satisfies the conformal equation, hence is conformal. Since m is unique as a conformal measure it equals $\widetilde{m}$. Moreover, ν is equivalent to m, and must be equal to μ, since the latter is unique as well. This shows that ν_n converges weakly to μ, proving that x_n^* is a sequence of pseudorandom points. $\square$

3. Conformal measure on a Julia set

We describe a specific example in this section, which will be used to show how the general algorithm can be applied.

Consider the rational map $T : C \to C$ defined by $T(z) = z^2 + \frac{1}{8}$. Its Julia set $J(T)$ is a bounded compact set in C (under the induced (Euclidean) topology).

A Julia set $J(T)$ is the closure of the set of repelling periodic points and inverse images of T^n, $(n \geq 1)$ are dense as well ([1]). Thus the repelling periodic points and its preimages are dense inside the Julia set $J(T)$, and every point in $J(T)$ can be realized as a limit of some sequence of preimages of each repelling periodic point.

The set of points y such that the iterates of y under T, T^2, T^3 and so on eventually hit a fixed repelling periodic point z is dense in the Julia set. Therefore, it is possible to construct many points in the Julia set by taking preimages. This can be done in different ways. The most convenient is to calculate inverse branches f_1 and f_2 (for a quadratic polynomial) as maps defined on the Julia set. Then we can iterate all possible finite combinations $f_1 \circ f_2 \circ f_1 \circ f_1 ...$, where the sequence of f_1 and f_2's are arbitrary choices. This is the naive way, since the computation creates too many data. We need for later purpose a certain depth of the iteration procedure, much longer than the forward iteration done later. In order to accomplish this one takes random choices of the two maps over a long string of iterations. This gives one point in the Julia set and one needs to estimate the errors in this calculation.

We now discuss the discretization, i.e., a random mesh on the Julia set.

Take a repelling periodic point z_0, say a fixed point: $T(z_0) = z_0$. Let us discretize the Julia set $J(T)$, i.e., generate a random mesh or random lattice S over it, as its computational representation.

The inverse of T has two analytical branches, f_1 and f_2. We backward iterate T but select the inverse branches randomly. Let l be a large positive integer. Define a sample space

$$\Omega = \{1,2\}^l = \{\omega = (\omega_1, \omega_2, \cdots, \omega_l) \ : \ \omega_i = 1 \text{ or } 2\}.$$

Starting from z_0, a random backward iteration of l steps of T can be represented as

$$f_{\omega_l} \cdots f_{\omega_2} f_{\omega_1}(z_0),$$

where $\omega_1, ..., \omega_l$ are chosen randomly with equal probability. Let Ω_0 be a randomly chosen set of $\omega_1, ..., \omega_l$. Its cardinality is the the size of the random mesh. Now we define a random mesh of the Julia set $J(T)$ defined by Ω_0 as

$$S = \{z : \ z = f_{\omega_l} \cdots f_{\omega_2} f_{\omega_1}(z_0), \omega = (\omega_1, \omega_2, \cdots, \omega_l) \in \Omega_0\}. \tag{3.1}$$

We take the Borel sets A to be small balls in $J(T)$ centered around some points in S. Let us take a representative subset S_0 of S and take A as balls centered on points in S_0. This is a finite family of balls and it is arranged to cover $J(T)$:

$$\mathcal{A} := \{A = B(z^*, \delta) : \text{ ball with center } z^* \in S_0 \text{ and radius } \delta > 0\}. \tag{3.2}$$

Let h denote the Hausdorff dimension of the Julia set $J(T)$. We shall consider the conformal measure m associated to the potential $h \log |T'|$, which is a well defined Lipschitz continuous function on $J(T)$, since T is hyperbolic. The transfer operator (Perron-Frobenius operator) for the the map T and the potential is $P : C(J(T)) \to C(J(T))$ defined as

$$Pg(z) = \sum_{y \in T^{-1}(z)} g(y) |T'(y)|^{-h}. \tag{3.3}$$

Iteration yields

$$P^n g(z) = \sum_{y \in T^{-n}(z)} g(y) |(T^n)'(y)|^{-h}. \tag{3.4}$$

It is known that there exists a unique conformal measure with respect to this potential, always denoting it m. Moreover, T is ergodic and has a unique finite invariant measure μ (on $J(T)$) that is equivalent to the conformal measure m.

The Hausdorff dimension of the Julia set can be calculated as follows. It is known that

$$\sum_{n=0}^{\infty} \sum_{T^n(y)=x} |(T^n)'(y)|^s$$

converges for $s < -h$ and diverges for $s > -h$. We shall use a slight variant of this fact to determine h:

$$\sum_{T^n(y)=x} |(T^n)'(y)|^s$$

converges only for $s = -h$. The Hausdorff dimension h is approximately $= 1.00735$ for the map $z \mapsto z^2 + \frac{1}{8}$.

The starting point for the optimization is the defining equation for a conformal measure (1.2). We evaluate this equation for balls $A \subset J(T)$. It is known that m has no atoms for our special quadratic map considered here, since the Julia set is a Jordan curve. Thus $m(\partial B) = 0$ for open sets and we can construct pseudo-generic points from the equation

$$m(T(A)) = \int_A |T'|^h dm \tag{3.5}$$

directly, where A are balls.

Since $\mu \sim m$, by the Radon-Nikodym theorem $\frac{d\mu}{dm} = f(z)$ (density of μ w.r.t. m), $\mu(TA) = \int_{TA} f dm$ or

$$m(TA) = \frac{1}{f(Tz^{**})}\mu(TA) = \int_A |T'|^h dm = \frac{1}{f(z^*)}\int_A |T'|^h d\mu$$

for some point z^* and z^{**} by the intermediate value theorem since the density is continuous. These equations hold approximately for all z replacing z^* and z^{**} if the sets A and $T(A)$ are small enough.

Thus, on small balls A in the Julia set, we have

$$\begin{aligned} m(T(A)) &= \frac{1}{f(T(z^*))}\int_{T(A)} d\mu \\ &= \frac{1}{f(T(z^*))}\lim_{n\to\infty}\frac{1}{n}\sum_{k=0}^{n-1} 1_{T(A)}(T^k(z)), \qquad (3.6) \\ \int_A |T'|^h dm &= \frac{1}{f(z^*)}\int_A |T'|^h d\mu \\ &= \frac{1}{f(z^*)}\lim_{n\to\infty}\frac{1}{n}\sum_{k=0}^{n-1} 1_A(T^k(z))\ |T'(T^k(z))|^h, \qquad (3.7) \end{aligned}$$

where $z \in J(T)$ is in a full $\mu-$measure subset in $J(T)$.

Therefore, we have the fundamental equation

$$\frac{1}{f(T(z^*))}\lim_{n\to\infty}\frac{1}{n}\sum_{k=0}^{n-1} 1_{T(A)}(T^k(z)) = \frac{1}{f(z^*)}\lim_{n\to\infty}\frac{1}{n}\sum_{k=0}^{n-1} 1_A(T^k(z))\ |T'(T^k(z))|^h \tag{3.8}$$

We will find such a z approximately by the method of least squares, i.e. finding the minimizer for

$$\min_{z\in S}\sum_{A\in\mathcal{A}}\left|\frac{1}{f(T(z^*))}\frac{1}{n}\sum_{k=0}^{n-1}1_{T(A)}(T^k(z))-\frac{1}{f(z^*)}\frac{1}{n}\sum_{k=0}^{n-1}1_A(T^k(z))\,|T'(T^k(z))|^h\right|^2 \quad (3.9)$$

where -more precisely- z^* depends on the corresponding ball A.

For calculating the density $f(z^*)$ we use the transfer operator and the well know equation

$$f(z)=\lim_{n\to\infty}P^n1(z)=\lim_{n\to\infty}\sum_{y\in T^{-n}(z)}|(T^n)'(y)|^{-h} \quad (3.10)$$

We choose points $z^*\in S_0$ in the discretization step and open balls around these points as choices of the sets A. The equation (3.10) is used to calculate the densities at z^* and $T(z^*)$.

We check numerically whether the minimizer in the optimization problem is pseudo-generic.

For any continuous and bounded function on $J(T)$, and for any generic point z in $J(T)$, we should have

$$\int g d\mu=\lim_{n\to\infty}\frac{1}{n}\sum_{k=0}^{n-1}g(T^kz),\ a.e.-\mu, \quad (3.11)$$

for $g\in L^1(J(T))$. In fact, in the next section, we test this for the function $g=|z|$. For z obtained in our numerical procedure, we compute $\lim_{n\to\infty}\frac{1}{n}\sum_{k=0}^{n-1}g(T^k(z))$ for some large n. Repeated calculation for different sets S, S_0, random choices and n will show that the average does not vary considerably. Choosing the backwards iteration randomly and S_0 randomly may be seen as the analog of a seed in the construction of random numbers.

4. Numerical results

4.1. **Determining the Hausdorff Dimension h for the Julia Set.** Before we can hope to evaluate the Perron-Frobenius operator for a generic point in $J(T)$, we must determine the appropriate Hausdorff dimension. There should be only one value h which allows for convergence of the limit described by (3.10) to a value $f(z)\in(0,\infty)$. This h will be different for each rational map but since we only consider one such map here, $T(z)=z^2+\frac{1}{8}$, we need only determine the dimension of the resulting fractal (Fig. 1).

We begin by taking the only repelling fixed point of T as a test value to determine h. Below is a table which shows the sequence of $f_n(z)$ values (see §4.4) as $n\to\infty$.

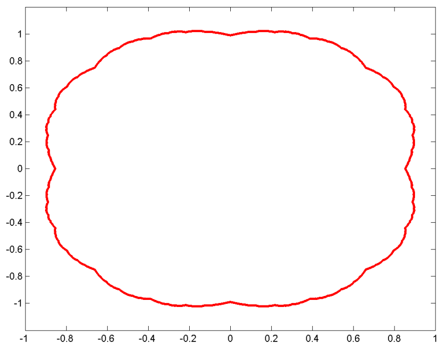


FIGURE 1. Julia Set of the mapping $T(z) = z^2 + 1/8$

Convergence of the Density Function

$z_0 = .8356$	$h = 1.00$		$h = 1.00735$		$h = 1.01$	
	$f_n(z_0)$	f_n/f_{n-1}	$f_n(z_0)$	f_n/f_{n-1}	$f_n(z_0)$	f_n/f_{n-1}
$n = 3$	1.3029	1.0983	1.2922	1.0935	1.2884	1.0918
$n = 5$	1.4132	1.0299	1.3884	1.0250	1.3796	1.0232
$n = 10$	1.4865	1.0059	1.4245	1.0009	1.4028	0.9991
$n = 15$	1.5256	1.0051	1.4258	1.0000	1.3914	0.9982
$n = 20$	1.5644	1.0050	1.4258	1.0000	1.3789	0.9982

FIGURE 2. Table to experimentally determine the Hausdorff dimension.

It is clear that the approximate value $h = 1.00735$ allows for convergence of the Perron-Frobenius operator, and that values too small or too big yield unbounded or zero answers respectively. For the remainder of this project we will approximate h as such. More rigorous discussion of Hausdorff dimension computation can be found in [9]. If n increases the results become better in generally. Statistical rigourous methods use the Grassberger and Procaccia correlation dimension and has been developed by Cutler and Dawson or Denker and Keller. The above approach is sufficient for hyperbolic maps.

4.2. **Creating the Computational Lattice.** Now that we have determined the appropriate Hausdorff dimension, we compute the z^* which define the A (the covering of the Julia set) and the lattice S. Given an m, the z^* are all the points for which $z_0 = T^m(z^*)$, which can be computed directly without much difficulty. The second block of code in Appendix A.1 generates the points z^* using $z_0 = .8536$. Logically, there are 2^m points which generate the covering of the Julia Set, since there are 2^m pre-images in $T^{-m}(z_0)$. An example of a covering of the Julia set is below.

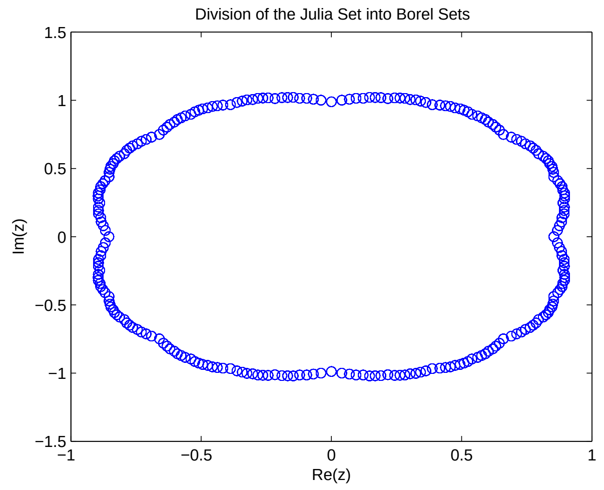


FIGURE 3. For $m = 8$ there are 256 z^* points which center the circles $A \in \mathcal{A}$ which cover $J(T)$.

Each of the points in S is determined by an inverse iteration backwards from a point randomly chosen from the z^*. $|S| = \ell$, and each point is inverse iterated backwards N times from the initial randomly chosen point. At each inverse iteration only one pre-image is chosen and stored, since the other pre-image is of no consequence for determining the final lattice. The appropriate Matlab code to create the lattice is the function `makelattice` which is found in Appendix A.2.

4.3. **Numerical Error Analysis in Random Mesh Generation.** In §4.2, we use the random mesh S to discretize the Julia set $J(T)$. So at least, we would need to make sure that points in S are (approximately) inside $J(T)$, when ℓ is large enough. Numerical error here comes mainly from computing the inverse branches $f_1 = F_1, f_2 = F_2$ of $T = z^2 + \frac{1}{8}$. Let us show that this numerical process is stable, i.e., the total error is bounded ([10]).

Let F denote either one of F_1 and F_2. The error analysis for numerically iterating each function is similar. Let the computer's unit roundoff error be, for example with double precision, 10^{-16}. Let us ignore the error in computing the initial repelling fixed point z_0.

Denote $\tilde{F}(z_0)$ be the computed value of $F(z_0)$. Then

$$|\tilde{F}(z_0) - F(z_0)| \le |F(z_0)|\epsilon_0, \ \epsilon_0 \le 10^{-16}.$$

Denote $z_1 = F(z_0), \tilde{z}_1 = \tilde{F}(z_0)$. In the following ϵ_i's denote relative roundoff errors at various steps of computation. Then

$$\begin{aligned}
|\tilde{F}(\tilde{z}_1) - F(z_1)| &= |\tilde{F}(\tilde{z}_1) - F(\tilde{z}_1) + F(\tilde{z}_1) - F(z_1)| \\
&\leq |\tilde{F}(\tilde{z}_1) - F(\tilde{z}_1)| + |F(\tilde{z}_1) - F(z_1)| \\
&\leq |F(z_1)|\epsilon_1 + |F'(z_1)(\tilde{z}_1 - z_1)| \\
&\leq |F(z_1)|\epsilon_1 + |F'(z_1)|\,|F(z_0)|\epsilon_0, \quad \epsilon_1 \leq 10^{-16}.
\end{aligned}$$

As long as F and its derivative is bounded in the (bounded) Julia set, the right hand side of the above error estimate is upper bounded by a constant multiplying $\epsilon \leq 10^{-16}$. Thus when we iterate n times of F, the relative roundoff error ([10]) is approximately $n\epsilon$. The random selection between F_1 and F_2 does not change this error order.

4.4. **Efficiently Computing the Transfer Operator.** Now that we have determined our computational lattice we must compute the transfer operator for obtaining $f(z)$ which is the density as defined by the Radon-Nikodym derivative. This is a computationally sensitive segment of the procedure because it requires approximating a limit which becomes exponentially more expensive to compute. We can make this somewhat easier by noting that we consider only one mapping, and that $T'(z)$ is a linear function.

$$T(z) = z^2 + \frac{1}{8} \quad \Rightarrow \quad T'(z) = 2z \tag{4.1}$$

The chain rule allow us to say the following,

$$(T^n)'(z) = (T \circ T^{n-1})'(z) = T'(T^{n-1}(z))T'(T^{n-2}(z))...T'(T(z))T'(z). \tag{4.2}$$

Below we substitute (4.1) and (4.2) into (3.10) and use the chain rule technique mentioned to simplify the evaluation of the transfer operator. The final simplification occurs because

$$|uv| = |u||v| \qquad \forall u, v \in \mathbb{C}$$

Note that the set $T^{-n}(z)$ includes all 2^n (possible non-unique) values on the Julia set for which $T^n(T^{-n}(z)) = z$.

$$\begin{aligned}
f(z) &= \lim_{n\to\infty} \sum_{y\in T^{-n}(z)} |(T^n)'(y)|^{-h} \\
&= \lim_{n\to\infty} \sum_{y\in T^{-n}(z)} |T'(T^{n-1}(y))...T'(y)|^{-h} \\
&= \lim_{n\to\infty} \sum_{y\in T^{-n}(z)} |2T^{n-1}(y)...2y|^{-h} \\
&= \lim_{n\to\infty} 2^{-hn} \sum_{y\in T^{-n}(z)} \left|\prod_{k=0}^{n-1} T^k(y)\right|^{-h} \\
&= \lim_{n\to\infty} 2^{-hn} \sum_{y\in T^{-n}(z)} \left(\prod_{k=0}^{n-1} \left|T^k(y)\right|\right)^{-h} \qquad (4.3)
\end{aligned}$$

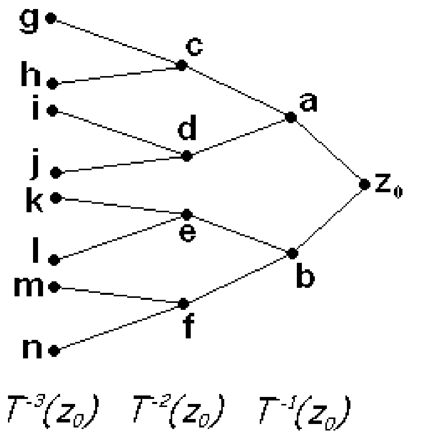
We can further simplify this by noting that we are not interested in the actual values of $T^{k-n}(z)$ but rather only their moduli. For each value $y \in T^{-n}(z)$ there is a value $-y \in T^{-n}(z)$ since both the positive and negative square roots must be considered. Because of this we need only calculate the contribution of half the pre-images to the summation and then double it since $|y| = |-y|$.

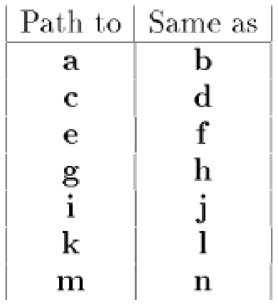
If we let T_+^{-n} denote only the positive square root pre-images of all 2^{n-1} pre-images in $T^{-(n-1)}$ (note that $|T^{-(n-1)}| = |T_+^{-n}|$), we can modify (4.3)

$$\begin{aligned}
f(z) &= \lim_{n\to\infty} 2^{-hn} \sum_{y\in T^{-n}(z)} \left(\prod_{k=0}^{n-1} \left|T^k(y)\right|\right)^{-h} \\
&= \lim_{n\to\infty} 2^{-hn} \left[2 \sum_{y\in T_+^{-n}(z)} \left(\prod_{k=0}^{n-1} \left|T^k(y)\right|\right)^{-h}\right] \\
&= \lim_{n\to\infty} 2^{-hn+1} \sum_{y\in T_+^{-n}(z)} \left(\prod_{k=0}^{n-1} \left|T^k(y)\right|\right)^{-h} \qquad (4.4)
\end{aligned}$$

In the Matlab code to execute this we take advantage of the fact that we need only calculate and store half the moduli needed by the same logic used before. This is seen in graph theory and Figure 4 shows how we utilize the fact that several branches in the tree have the same product.

Using this logic we wrote the function `densop` to approximate the limit in (4.4) to 10^{-4} accuracy. If the appropriate h value is not used, the program will





Path to	Same as
a	b
c	d
e	f
g	h
i	j
k	l
m	n

FIGURE 4. Mutliple pre-image paths yield the same product in computing the density in (4.4)

likely fail to converge and run in perpetuity. See Appendix A.3 for the Matlab implementation.

For those who are interested in the speed of this algorithm, ours is certainly not the fastest possible implementation. Each time `densop` is called it recalculates pre-images which may have already been determined. In addition, new memory is allocated in each iteration above as well as at the start of each call to `densop`. All the code in this project is written to test the algorithms described above and to emphasize readability; this has resulted in a decrease in efficiency which will be the topic of a future project.

4.5. **Solving the Optimization Problem.** Recall the optimization problem we need to solve: given z^* and δ_A (to define the Borel Set $A = B(z^*, \delta_A)$) and n

$$\min_{z \in S} \sum_{A \in \mathcal{A}} \left| \frac{1}{f(Tz^*)} \frac{1}{n} \sum_{k=0}^{n-1} 1_{T(A)}(T^k z) - \frac{1}{f(z^*)} \frac{1}{n} \sum_{k=0}^{n-1} 1_A(T^k z) \left| T'(T^k z) \right|^h \right|^2 .$$

One thing to note is that the optimization is to occur on a discrete lattice, S, thus we need only test a finite number of points, ℓ, to find the solution. Another point of interest is that $T^k(z)$ is evaluated during construction of the lattice so no new function evaluations take place. Also there is no need to set $n > \ell$ since $T^\ell(z) = z_0$ for all $z \in S$.

To determine if a point z is in $B(z^*, \delta_A)$ we simply test $|z - z^*| < \delta_A$, thus

$$1_A(z) = \begin{cases} 1 & |z - z^*| < \delta_A \\ 0 & \text{else} \end{cases} \tag{4.5}$$

Testing if z is in $B(z^*, \delta_{T(A)})$ is more difficult. To do so we state that

$$\begin{aligned} z &\in T(A), \\ T^{-1}(z) &\in A, \end{aligned} \tag{4.6}$$

and therefore test whether either pre-image of z is in A. The result is that

$$1_{T(A)}(z) = \begin{cases} 1 & |T_{\pm}^{-1}(z) - z^*| < \delta_A \\ 0 & \text{else} \end{cases} \tag{4.7}$$

where $T_{\pm}^{-1}(z)$ is either the positive or negative preimage of z.

Appendix A.4 is the Matlab implementation of the optimization procedure. This code simply runs through every point in $z \in S$ and returns the point which minimizes (3.9) for summations of a given length n. It also returns the β_n value described by (2.1).

4.6. **Testing the Pseudorandom Points.** In order to determine if the z which satisfies (3.9) is a pseudorandom point we test its time average in equation (3.11). We use the simple test function $g(x) = |x|$ for which the integral can be approximated deterministically; the lhs of (3.11) is the average distance of points in $J(T)$ from the origin. When all the pre-images for various values of m are averaged together, we see below that the result approaches the limit $\int |z| d\mu \approx 1.001379$.

Integral Limit

m	$\int \lvert z\rvert d\mu$
1	0.853553
5	0.990741
10	1.001044
15	1.001369
20	1.001379
21	1.001379

FIGURE 5. Table describing the approximate solution to (3.11).

There are several factors which contribute to the quality of the solution and the complexity of the algorithm. We will assume here that we already know h and that $m = 8$ is fixed which means that the Julia set is covered by $2^8 = 256$ balls and that $\delta_A = 2^{1-m}$ is also fixed. Assuming that the z^* are already available (which is reasonable because the Borel sets are defined by m), the only parameters which affect the accuracy of the integral are

- ℓ - The number of elements in S, the computational lattice
- N - The depth of the inverse iteration on randomly chosen points z^* to generate S

  - Recall that for $z \in S$, $T^N(z) = z^*$ for exactly one of the 2^m z^*.
- n - The point at which we truncate the least squares limit
- α - The number of pseudorandom trajectories used to evaluate the integral

We can now take a look at how changing these parameters individually affects the speed and accuracy of the algorithm. For all these experiments we have generated 10 computational lattices (ie $\alpha = 10$); for each lattice 1 pseudorandom point minimizes the least squares equation and that is the point whose trajectory we use to compute the integral. μ is the experimental mean and σ is the experimental standard deviation

ℓ	μ	σ ($\times 10^{-3}$)	Time
25	1.00136	0.5848	6
50	1.00179	0.8568	11
100	1.00149	0.8295	20
200	1.00145	0.9279	42
400	1.00122	0.9090	91
800	1.00150	0.9652	205
1600	1.00150	0.5960	413
3200	1.00142	0.4890	826

(A) Fixed $N = 16000$, $n = 100$.

n	μ	σ ($\times 10^{-3}$)	Time
25	1.00117	0.8050	13
50	1.00160	0.6210	15
100	1.00149	0.8295	20
200	1.00116	0.6325	32
400	1.00148	0.8654	70
800	1.00147	0.5014	146
1600	1.00136	0.5816	281
3200	1.00145	0.5851	576

(B) Fixed $N = 16000$, $\ell = 100$.

N	μ	σ ($\times 10^{-3}$)	Time
1000	1.00109	3.264	12
2000	1.00116	2.635	13
4000	1.00153	1.732	14
8000	1.00108	0.8890	16
16000	1.00149	0.8295	20
32000	1.00138	0.4505	30
64000	1.00119	0.3765	48
128000	1.00145	0.1743	85

(C) Fixed $n = 100$, $\ell = 100$.

FIGURE 6. The effect of varying N, n and ℓ.

We can see from Figure 6c that increasing N has the effect of decreasing the standard deviation of the estimator. Figure 7 is a graphic depiction of this. It appears in Figure 6b and Figure 6a that increasing n and ℓ without changing N causes no improvement in σ. This leads us to believe that the

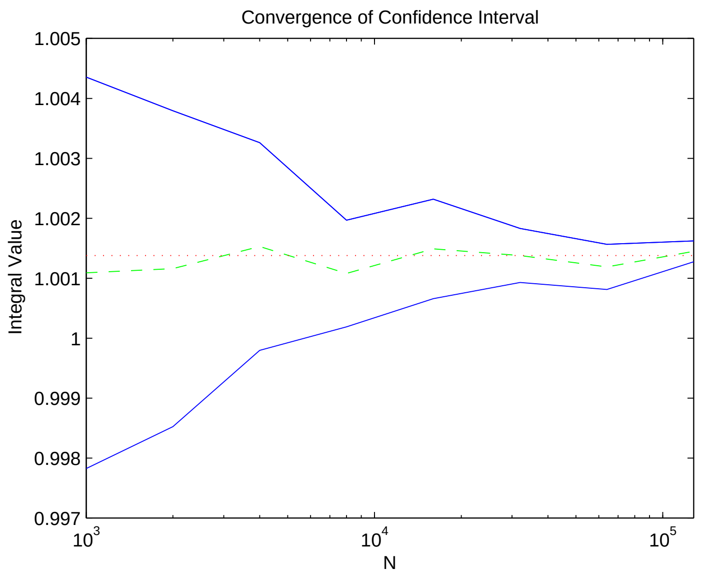


FIGURE 7. Solid - $\mu \pm \sigma$, Dashed - μ, Dotted - true solution

driving force behind accuracy is N for which the complexity of the algorithm increases linearly.

There are limitations to this algorithm because it requires storage of $N+1\times\ell$ terms: this is done to prevent the loss of accuracy from N applications of T on the elements of S. Unfortunately, since there is only one point in S which minimizes (3.9) there is only one pseudorandom trajectory chosen per S generated.

One possible future improvement to this algorithm may be to use several trajectories whose value in (3.9) are close to optimal but not the exact minimum. Their contribution to the estimator can be weighted according to their distance from the optimal value. This would allow the use of multiple trajectories from the same S and not require α versions of S for α trajectories.

Appendix A - Matlab Algorithms

The first file is a script which calls the other functions to generate pseudorandom numbers on the Julia Set for the mapping $T(z) = z^2 + 1/8$. Here is a list of important parameters:

- `z0` - A repelling periodic point which is the start of the inverse iterations
- `m` - 2^m Borel sets are used to cover the Julia Set
- `zstar` - The centers of the Borel sets. These are the $2\hat{}m$ pre-images in the set $T\hat{}\{-m\}(z_0)$
- `S` - The discretization of the Julia Set
- `ell` - The number of points in S
- `N` - All points in S are pre-images in the set $T\hat{}\{-(m+N)\}(z_0)$
- `n` - Summations in the optimization equation are of length n
- `alpha` - The required number of pseudorandom points
- `h` - The Haussdorff dimension

A.1 mainscript.m

```
% Here we consider the mapping T(z)=z^2+1/8
T=inline('x.*x+.125');
z0=fsolve(@(z) T(z)-z,.8); % approximately z0=0.85355339203135
m=8;
ell=100;
N=32000;
n=100;
alpha=30;
h=1.00735;
indmin = zeros(alpha,1);
beta_n = zeros(alpha,1);
caverage = zeros(alpha,1);
% This line makes the random number generator start with the
% same seed always.  The line below will randomize the seed.
rand('state',0);
% rand('state',sum(100*clock));

% We use inverse iteration to create the lattice.  We must
% determine the points in zstar and define the balls which
% cover the Julia Set.
zstar=z0*ones(2^m,1);
for bdec=1:2^m
  bcode=dec2bin(bdec-1,m);
  for i=1:m
    zstar(bdec)=(-1)^(bcode(i)=='1')*sqrt(zstar(bdec)-.125);
  end
```

```
end
% The function densop finds the density at zstar.
% This does not have a time limit, so if the script is
% hanging, densop is a likely source.  Both the density of
% the points in zstar and their images T(zstar) are found.
fzstar=zeros(2^m,1);
fTzstar=zeros(2^m,1);
for k=1:2:2^m
  fzstar(k)=densop(zstar(k),h);
  fzstar(k+1)=densop(zstar(k+1),h);
  fTzstar(k)=densop(T(zstar(k)),h);
  fTzstar(k+1)=fTzstar(k);
end

% Now use makelattice to form the discrete Julia Set.  The
% seeds for the inverse iteration of points on the lattice
% are randomly chosen from zstar.  The computational lattice
% is S(:,N). The final column is an extra inverse iteration
% for evaluating the fundamental equation.  It is important
% to note that the size of S is [ell,N+1] not [ell,N].

% After that we use the optimization function which will return
% the solution to the least squares problem detailed earlier.
% The function below returns imin, the index of the point in S
% which is the solution, and minival which is the residual.
% We should have beta_n/n^2<2*dn*|A_n|*LipConst as described
% in the earlier paper.
for i=1:alpha
  S=makelattice(ell,N,m,zstar);
  [imin(i),beta_n(i)]=opteval(n,S,zstar,fzstar,fTzstar,m,h);
end

% Now we test the ensemble averaging.  Any L1 function can be
% used to test the pseudorandomness of the points found by the
% optimization.  The function must be able to accept vector
% arguments, ie using .* instead of just * for multiplication.
g=inline('abs(x)');
for i=1:alpha
  caverage(i)=mean(g(S(imin(i),1:N)));
end
```

A.2 makelattice.m

```
function S=makelattice(ell,N,m,zstar)
% function S=makelattice(ell,N,m,zstar)
% This function makes the computational lattice that represents
% the Julia Set.  The rest of the matrix values are
% the trajectories taken backwards from a random selection of
% points on the zstar grid.  S(:,N) is the computational
% lattice, S(:,1) is ell randomly chosen points from zstar.
% S(:,N+1) is a preimage of S(:,N) which is needed for
% optimization to test if S(:,N) is in TA.
S = 2*(rand(ell,N+1)>.5)-1;
S(:,1) = zstar(ceil(2^m*rand(ell,1)));
for j=2:N+1
    S(:,j) = S(:,j).*sqrt(S(:,j-1)-.125);
end
```

A.3 densop.m

```
function fz=densop(z0,h)
% function fz=densop(z0,h)
% This function computes the density for the
% mapping T(z)=z^2+1/8.  It does this using a limiting sequence.
fval=[0,-1];
c=1;
cc=1;
pq(1)=z0;
pn(1)=abs(pq(1));
k=2;
while abs(fval(2)-fval(1))>1e-4
  c=[c,2^(k-1)];
  cc=cumsum(c);
  pq=[pq,zeros(1,c(k))];
  pn=[pn,zeros(1,c(k))];
  fval(1)=fval(2);
  fval(2)=0;
  for j=c(k):2:cc(k)
    pq(j)=-1^j*sqrt(pq(fix(j/2))-.125);
    pq(j+1)=-pq(j);
    pn(j)=abs(pq(j));
    pn(j+1)=pn(j);
  end
```

```
    kk=c(k);
    while kk<cc(k)
      kj=kk;
      while kj(length(kj))>2
        kj=[kj,kj(length(kj))/2-(mod(kj(length(kj)),4)>0)];
      end
      fval(2)=fval(2)+(prod(pn(kj)))^-h;
      kk=kk+2;
    end
    fval(2)=fval(2)*2^(-h*(k-1)+1);
    k=k+1;
end
fz=fval(2);
```

A.4 OPTEVAL.M

```
function [indmin,minival]=opteval(n,S,zstar,fzstar,fTzstar,m,h)
% function [indmin,minival]=opteval(n,S,zstar,fzstar,fTzstar,m,h)
% This needs the computational lattice, the transfer operator
% evaluated on the lattice, and the size of the Borel Set A
% around zt. n is the limit truncation which can not be greater
% than N, and h is the Hausdorff dimension.

% All this function does is run through the lattice and calculate
% the optimization equation at each point.  It returns the lowest
% value.  S is the group of trajectories which yield the
% computational lattice.  Specifically S(:,N) is the lattice.
[ell,N]=size(S);
N=N-1; % Recall size(S)=[ell,N+1] although S(:,N) is the lattice.
% The radius of the balls which cover the region is related to m
delta_A=2^(-m+1);

% The lhs part of the summation will find whether either of the
% preimages of S are in A.  This is equal to asking if S is in
% T(A).  The rhs part of the summation tests whether S is in A,
% and then adds the appropriate values.  The rest is just
% evaluating the optimization equation.

Asum = zeros(ell,1);
for j=1:2^m
    spS = sparse(abs(S(:,N-n:N))-zstar(j))<delta_A);
    rhs = 2*sum((spS.*abs(S(:,N-n:N))).^h,2);
    lhs = sum((S(:,N+1-n:N+1)-zstar(j)<delta_A) + ...
              (-S(:,N+1-n:N+1)-zstar(j)<delta_A),2);
    Asum = Asum+(lhs/fTzstar(j)-rhs/fzstar(j)).^2;
end
[minival,indmin] = min(Asum/N);
```

Acknowledgements

This work was partly supported by the NSF Grant DMS-0620539. We would like to thank Xiaofan Li for helpful discussions in numerical error analysis. This work is partly done while J. Duan was a Visiting Professor at Universität Göttingen and while M. Denker is a Distinguished Research Professor at Illinois Institute of Technology.

References

[1] A. F. Beardon, *Iteration of rational functions. Complex analytical dynamical systems.* Graduate Text in Math. 132. Springer, New York 2000.

[2] M. Dellnitz and O. Junge, *Set oriented numerical methods for dynamical systems.* Handbook of dynamical systems, Vol. 2, 221–264, North-Holland, Amsterdam, 2002.

[3] M. Dellnitz, G. Froyland, and O. Junge, *The algorithms behind GAIO-set oriented numerical methods for dynamical systems.* Ergodic theory, analysis, and efficient simulation of dynamical systems, 145–174, 805–807, Springer, Berlin, 2001.

[4] M. Denker, The central limit theorem for dynamical systems. *Dynamical Systems and Ergodic Theory*, edited by K. Krzyzewski. Banach Center Publ. **23**, 33–62. Polish Scientific Publ., Warszawa, 1989.

[5] M. Denker, F. Przytycki and M. Urbański, On the transfer operator for rational functions on the Riemann sphere. *Ergodic Theory and Dynam. Systems* **16**, 1996, 255–266.

[6] M. Denker and M. Urbański, *On Sullivan's conformal measures for rational maps of the Riemann sphere.* Nonlinearity 4 (1991), no. 2, 365–384.

[7] M. Denker and M. Urbański, *On the existence of conformal measures.* Trans. Amer. Math. Soc. 328 (1991), no. 2, 563–587.

[8] K. J. Falconer, *The Geometry of Fractal Sets*, Cambridge University Press, 1985.

[9] J. Kigami, *Analysis on fractals.* Cambridge Tracts in Mathematics, 143. Cambridge University Press, Cambridge, 2001.

[10] D. Kincaid and W. Cheney, *Numerical Analysis*, Second Edition, Brooks/Cole Pub. Company, Boston, 1996.

[11] U. Krengel, On the speed of convergence in the ergodic theorem. *Monatsh. Math.* **86**, 3-6 (1978).

[12] S. J. Patterson, *The limit set of a Fuchsian group.* Acta Math. 136 (1976), no. 3-4, 241–273.

[13] D. Sullivan, *Conformal dynamical systems.* Geometric dynamics (Rio de Janeiro, 1981), 725–752, Lecture Notes in Math., 1007, Springer, Berlin, 1983.