Notes on Mandelbrot set
(Draft)

February 14, 2018

Abstract

Math tricks for rendering the Mandelbrot set and other fractals. Most of this stuff (but not everything) has been invented and/or discussed over the last 5 years on `fractalforums.com/.org`.

1 Basics

The Mandelbrot (M) set is generated by the iterations

$$f(z,c) = z^2 + c \tag{1a}$$
$$z_0 = 0 \tag{1b}$$
$$z_{n+1} = f(z_n, c) = z_n^2 + c \tag{1c}$$
$$z'_{n+1} = \frac{\partial f}{\partial z} z'_n + \frac{\partial f}{\partial c} = 2 z_n z'_n + 1 \tag{1d}$$

where all symbols are complex numbers, unless otherwise indicated. $z_n(c)$ is the orbit of c, and the orbits on a (usually rectangular) region of c values (on a pixel grid) are used to make a picture. The derivative $z' = \partial z/\partial c$ is not necessary to compute but useful for coloring and other things. How to go from an orbit to a color is called making a representation function $m(c)$. There are many and only a few fundamental ones will be described.

First of all, let's give m a fixed value (say 0) on any interior point where the orbit does not escape. If $|z_n|$ ever gets bigger than 2 the orbit will escape, and c is a point in the exterior. The simplest representation function m is the iteration at which the orbit escapes, $|z_n| > R$ for some radius $R \geq 2$.

However, this m depends on R and is not smooth, and though we could just chose $R = 2$ it turns out to be better (for some purposes) to choose R large, say 1000000.

As the following tricks work more generally, let's consider a more general $f(z,c)$ where z is now just a 2-vector $z = (x, y)$ and $f(z,c)$ anything that satisfies $|f(z,c)| \approx |z|^q$ for large $|z|$ and some $q > 1$, and $|.|$ is now the Euclidean vector norm (length).

Once the orbits starts diverging and $|z|$ gets big, it will grow as

$$|z_n| = a^{q^n} \tag{2}$$

for some number a and the double logarithm (written as $l(x) = log(log(x))$) goes like

$$l(|z_n|) = l(a) + n\log(q) \tag{3}$$

which is linear, increasing by $\log(q)$ every iteration.

If the orbit has just escaped at $n = N + 1$ we have

$$l(|z_N|) < l(R) < l(|z_{N+1}|) \tag{4}$$

$$m = N + 1 \tag{5}$$

with m the value of the "naive" representation function. To make a smooth m, using (3), define the fractional iteration count $\hat{N}$, by solving $l(|z_{\hat{N}}|) = l(R)$, which gives

$$\hat{N} = (l(R) - l(a))/log(q) = (l(R) - l(|z_N|))/log(q) + N. \tag{6}$$

The constant $l(R)$ does not matter and is even undesirable, so just define the continuous iteration count as

$$m = N + 1 - \log(\log|z_N|)/\log(q). \tag{7}$$

For large enough R this is a smoothly varying function everywhere.

From m we can derive another nice representation function, the distance estimate $d(c)$ which is approximately how far from the boundary c is. First define the Green's or potential function

$$G(c) = e^{-m(c)\log(q)} \tag{8}$$

which is zero only on the boundary (where $m \to \infty$) and undefined when the orbit does not escape. The distance to the boundary of c is just the length of shortest v that solves $G(c+v) = 0$. If we're close, v is small and we can use the linear approximation

$$G(c + v) = G(c) + v \cdot \nabla G(c) = 0 \tag{9}$$

with ∇ the 2-d gradient, which has solution v of length

$$d = |v| = G/|\nabla G| = \frac{1}{|\nabla m|\log(q)} = \frac{|z_N|\log(|z_N|)}{|\nabla|z_N||}, \tag{10}$$

which is the distance estimate. For holomorphic $f(z, c)$ it simplifies to

$$d = \frac{|z_N|\log(|z_N|)}{|z'_N|} \tag{11}$$

with z'_N the (complex) derivative wrt c. The distance estimation looks best when z' is obtained from the iterations, but it is also possible to just compute the orbit and differentiate m numerically. This tends to require something like a 6X oversampling to look good, whereas the analytic distance coloring is fine with just 2X oversampling.

For the M-set one can prove that the distance estimate is within a factor 2 (multiply or divide) of the true distance [8].

Renormalization

For non-escaping points, of particular interest are periodic points which are in the center of a hyperbolic component (a mini body or attached bulb), and pre-periodic points (Misiurewicz points) which become periodic after a number (the pre-period) of iterations. For periodic points of period p we have

$$z_p = 0 \tag{12}$$

so $z_0, \ldots, z_{p-1}$ is repeated over and over. For Misiurewicz points (pp, p) with $pp > 0$ the preperiod we have

$$z_{pp+p} = z_{pp} \tag{13}$$

where pp is the smallest $pp > 0$ for which this holds.

Periodic points can be found using Newton's method to solve $z_p(c) = 0$ if we know p and using (1) for the derivative. Misiurewicz points require some more care to ensure we find a strictly preperiodic point [9].

If we know p and c_p for a periodic point (i.e., inside a hyperbolic component) we can estimate its size relative to the main M-set body (needed to make a picture) by the method of renormalization [20], which transforms p full iterations into 1 iteration of scaled variables in a small region around c_p.

Consider an orbit $z_n, z_{n+1}, \ldots z_{n+p}$, so $z_n = z_{n+p} = 0$. Next perturb this orbit by taking $c = c_p + b$ with $|b|$ very small (a "ball"). The iteration moves this ball around, until it returns at iteration p but with changed size. For the perturbed orbit Z we write $Z = z + w$ and we get

$$Z_k = z_k + w_k \tag{14a}$$
$$z_{k+1} = z_k^2 + c_p \tag{14b}$$
$$w_{k+1} = w_k^2 + 2z_k w_k + b \tag{14c}$$
$$z_0 = 0 \tag{14d}$$
$$w_0 = 0. \tag{14e}$$

As b is tiny (as small as necessary), so is w_k (the deviation from the periodic orbit under small shift in c) and the term w_k^2 can be neglected compared to $2z_k w_k$, unless $z_k = 0$. We get the linearized form

$$w_{k+1} = 2z_k w_k + b \quad \text{when } z_k \neq 0 \tag{15a}$$
$$w_{k+1} = w_k^2 + b \quad \text{when } z_k = 0 \tag{15b}$$
$$z_0 = 0 \tag{15c}$$
$$w_0 = 0. \tag{15d}$$

If we now iterate from n to $n+p$ we get

$$w_{n+1} = w_n^2 + b \tag{16a}$$
$$w_{n+k} = 2z_{n+k-1}w_{n+k-1} + b \tag{16b}$$
$$k = 2, \ldots, p \tag{16c}$$

The linear recursion (16b) can be solved explicitly [19] giving

$$w_{n+p} = \Lambda(w_n^2 + \beta b) \tag{17a}$$

where

$$\lambda_m = \prod_{k=1}^{m}(2z_{n+k}) \tag{17b}$$
$$\Lambda = \lambda_{p-1} \tag{17c}$$
$$\beta = 1 + \sum_{k=1}^{p-1}\frac{1}{\lambda_k}. \tag{17d}$$

We have $Z_n = w_n$ and $Z_{n+p} = w_{n+p}$ since $z_n = z_{n+p} = 0$, and writing $b = c - c_p$ we get

$$Z_{n+p} = \Lambda(Z_n^2 + \beta(c - c_p)) \tag{18}$$

which can be renormalized in terms of new variables

$$\hat{Z} = \Lambda Z \tag{19a}$$
$$\hat{c} = \beta\Lambda^2(c - c_p) \tag{19b}$$
$$\hat{Z} \leftarrow \hat{Z}^2 + \hat{c} \tag{19c}$$

which will give the usual zoomed out M-set shape when $\hat{c}$ is inside the body or bulbs, which means the actual c values $c - c_p$ should be in a similar shaped region but shrunk by a scale/rotation factor $\frac{1}{\beta\Lambda^2}$. The method generalized to more general functions $f(z, c)$, just replace $2z_i$ where it occurs by $\partial f/\partial z$ (or a Jacobian matrix for non-analytic f). See [10].

2 Complex differentiation

For a nice introduction to complex analysis see [1].

Functions $f(z)$ can also be considered functions $f(x, y)$ with $z = x + iy$, and $f(z) = u(x, y) + iv(x, y)$ with u, v, x, y real.

A function $f(z)$ is called holomorphic or analytical if it satisfied the Cauchy-Riemann conditions $u_x = v_y$ and $u_y = -v_x$ with subscripts indicating partial differentiation. Loosely speaking it is analytical if $f(z)$ has a formula involving only z but

not its complex conjugate $\bar{z}$. Any function $f(x, y)$ can be recast in the form $f(z, \bar{z})$. Analytic functions satisfy $\partial f/\partial \bar{z} = 0$. We have the following conversions

$$\frac{\partial}{\partial z} = (\frac{\partial}{\partial x} - i\frac{\partial}{\partial y})/2 \tag{20a}$$

$$\frac{\partial}{\partial \bar{z}} = (\frac{\partial}{\partial x} + i\frac{\partial}{\partial y})/2. \tag{20b}$$

When dealing with analytical functions f we often have to compute derivatives of absolute values of functions, which are functions from complex numbers to reals and obviously not analytical. Some useful results derivable from the above are

$$\frac{\partial |f|^2}{\partial z} = \bar{f}\frac{\partial f}{\partial z} \tag{21a}$$

$$||\nabla |f||| = |\frac{\partial f}{\partial z}| \tag{21b}$$

where f is holomorphic, $|.|$ is the complex absolute value, $||.||$ is the Euclidean 2-vector norm, and ∇ is the gradient operator (vector of partial derivatives wrt x and y).

3 Rendering methods

To compute a representation function involves selecting a rectangular window in the complex plane, fill it with a pixel grid of c values, and for each c iterate (1) up to some maximum number of iterations. Implementing this using normal floating point (FP) numbers (usually 64 bit double precision) works only if the pixel spacing is bigger than the round off error, $h > \epsilon$ with ϵ the machine precision, about 2.10^{-16}. This roughly corresponds to a maximum magnification of about 10^{12} for a high resolution 10000×10000 image.

For deeper zooms we can replace the usual FP numbers by "arbitrary precision" numbers, implemented in software. This however results in a tremendous slowdown.

Recently [17] a different approach, called "perturbation theory (PT) and series approximation (SA)", was suggested and implemented, using variable splitting and Taylor expansions. This method does not work directly, but needs some sophisticated error correction methods, which has been developed by various people on the fractal forum `fractalforums.com` and still continues on its successor, `fractalforums.org`.

3.1 Perturbation theory

Let c range over a small window (e.g., of size 10^{-1000}) and pick one (reference) point c_r, for example the center though there are probably better selection methods. Now split the variables occurring in (1) into large and small parts.

$$c = c_r + b \tag{22a}$$

$$z_k = X_k + w_k \tag{22b}$$

where b and w_k are small. In terms of the new variables we have

$$x_k = round(X_k) \tag{23a}$$
$$w_{k+1} = w_k(w_k + 2x_k) + b \tag{23b}$$
$$z_{k+1} = x_{k+1} + w_{k+1} \tag{23c}$$
$$z'_{k+1} = 2(x_k + w_k)z'_k + 1 \tag{23d}$$
$$X_{k+1} = X_k^2 + c_r \tag{23e}$$
$$X_0 = 0 \tag{23f}$$
$$w_0 = 0 \tag{23g}$$
$$z'_0 = 0 \tag{23h}$$

where X and c_r are multiple precision (MP) variables, and all other variables are FP, which may need to be extended to prevent under and overflow. This makes it a bit slower than normal FP (about a factor 4), but not by as much as using MP. The "round" operator converts from MP to FP.

This "splitting" method is common in numerical solutions of differential equations, see for example [7].

3.2 Series approximation

The second ingredient, the SA, is based on the observation that since b is small, and $w_n(b)$ (23b) is a degree 2^{n-1} polynomial in b, we should be able to discard some of these higher power terms by truncating the polynomial to a lower order K. This is not an optional refinement, as the saving in computation time is often two orders of magnitude (factor 100). To avoid scaling issues it's best to expand in $\hat{b} = b/b_{max}$ with $b_{max} = \max(|b|)$ So we approximate

$$w_n(b) \approx \sum_{k=1}^{K} a_{kn}\hat{b}^k \tag{24a}$$

where the coefficients a_{kn} can be found by substitution (full derivation in Section 3.2.2) in (23b) and collecting terms of the same order in $\hat{b}$, resulting in [11]

$$k = 1, \ldots, K \tag{24b}$$
$$a_{k1} = b_{max}\delta_{k1} \tag{24c}$$
$$a_{k,n+1} = 2x_n a_{kn} + b_{max}\delta_{k1} + \sum_{j=1}^{k-1} a_{jn}a_{k-j,n} \tag{24d}$$

where δ_{k1} is the Kronecker delta. All these variables are FP (possibly with extended range if b_{max} is very small). Once K is chosen (not sure how) we have to figure out up to which iteration M we can use it and compute w_M from (24a), skipping

iterations $1, \ldots, M-1$ and continue with normal iterations (23b) after that. M should be chosen to keep errors small, and a method to determine these is described and discussed here [2].

Note that the minimum number of iterations in the image is an upper bound for M as for $n > M$ some points will diverge and will have no Taylor expansion.

3.2.1 Theoretical SA error analysis

Evaluating w_M using (24a) introduces several errors which need to be controlled. The calculation of the coefficients a_{kn} has rounding errors, truncating the full w_M polynomial has a truncation error, usually called the remainder, and evaluating (24a) has rounding errors. The usual method to evaluate (24a) is through Horner's method, and a more accurate method is described in [16] which also reviews error analysis for the Horner method.

For a Taylor expansion $P_k(z)$ to order k of function $f(z)$ (a polynomial in the case at hand) around $z = 0$ we have (exactly)

$$f(z) = P_k(z) + R_k(z) \tag{25a}$$

$$P_k(z) = \sum_{i=0}^{k} \frac{f^{(k)}(0)}{k!} z^k \tag{25b}$$

where there are several exact expressions for the remainder R_k:

$$R_k(z) = \frac{f^{(k+1)}(\xi)}{(k+1)!} z^{k+1}, \ \textit{for some } |\xi| < |z| \tag{25c}$$

$$R_k(z) = \frac{z^{k+1}}{2\pi i} \oint_{\partial U} \frac{f(u)}{u^{k+1}(u-z)} du \tag{25d}$$

where the integral is over the boundary of a region U that includes all values of z we're interested in.

(25d) allows us to compute a bound on R_k. In terms of the problem at hand (25d) reads

$$R_k(b) = \frac{b^{k+1}}{2\pi i} \oint_{\partial U} \frac{w_M(u)}{u^{k+1}(u-b)} du. \tag{26}$$

Let's assume we're rendering a ball of radius $r = \max(b)$ around 0 (or around the reference point in terms of the full variables). Let U be a ball of radius αr, $\alpha > 1$. An upper bound for the integrand is

$$\left| \frac{w_M(u)}{u^{k+1}(u-b)} \right| \leq \frac{\max_U(|w_M|)}{(\alpha r)^{k+1} r}, \tag{27a}$$

using $|u-b| \geq r.$ and $r \leq |b|$. Path length is $2\pi\alpha r$, and $\max(w_M) = 4$ if nothing has escaped yet in U. so

$$R_k(b) \leq \frac{\max_U(|w_M|)r^{k+1}\alpha r}{(\alpha r)^{k+1}r} = \max_U(|w_M|)/\alpha^k. \tag{27b}$$

This proves that at least the Taylor expansion converges up to M where nothing has escaped. Not sure if there is a practical use for this analysis.

3.2.2 Practical SA error analysis

Probably more promising is the ball method explained in [2], as this takes into account the actual iterations for w. Ball method is explained in Section 4.1.

Here's my derivation, which agrees with results from [2]. Some notational simplifications:

$$\sum_i = \sum_{i=1}^{K} \tag{28a}$$

$$\sum_{ij-} = \sum_{\{i,j\in 1,\ldots,K | i+j\leq K\}} \tag{28b}$$

$$\sum_{ij+} = \sum_{\{i,j\in 1,\ldots,K | i+j> K\}} \tag{28c}$$

Iteration indices omitted are n.

For an SA to order K, we write for the ball variable

$$[w_n] = \sum_i a_{in}\hat{b}^i + r_n E \tag{28d}$$

$$[w_{n+1}] = \sum_i a_{i,n+1}\hat{b}^i + r_{n+1} E. \tag{28e}$$

The following steps derive recursion relation for the $a_i(n)$ and r_n.

$$[w_{n+1}] = [w_n]([w_n] + 2X_n) + b_{max}\hat{b} = \tag{28f}$$

$$(\sum_i a_i\hat{b}^i + rE)(\sum_i a_i\hat{b}^i + rE + 2X) + b_{max}\hat{b} = \tag{28g}$$

$$\sum_{ij} a_i a_j \hat{b}^{i+j} + \sum_i (2Xa_i + b_{max}\delta_{i1})\hat{b}^i + (r^2 + 2r(\sum_i a_i\hat{b}^i + X))E = \tag{28h}$$

$$\sum_{ij-} a_i a_j \hat{b}^{i+j} + \sum_i (2Xa_i + b_{max}\delta_{i1})\hat{b}^i + \tag{28i}$$

$$+ (r^2 + 2r(\sum_i a_i\hat{b}^i + X))E + \sum_{ij+} a_i a_j \hat{b}^{i+j}. \tag{28j}$$

In (28i) we have only powers of $\hat{b}$ up to K and we equate this with the first term of the right hand side of (28e), which gives the SA coefficient recursion (24) after resolving the $i+k \leq K$ constraint. The term (28j) is a ball plus something we want to reduce, so we replace that with a ball and equate that with the second term in (28e) and we get

$$r_{n+1} = r_n^2 + 2r_n|\sum_i a_{in}\hat{b}^i + X_n)| + |\sum_{ij+} a_{in}a_{jn}\hat{b}^{i+j}|. \tag{28k}$$

In [2, 3] the recursion was in terms of $R_n = r_n/|b|^{K+1}$ (no connection to the R in (25a)).

As (28k) depends on b it's still not useful because if we'd had to evaluate it for every b we'd be better off without the SA altogether. From the bound $|\hat{b}| \leq 1$ and other inequalities we can obtain the pixel independent error bound as (slightly abusing notation by redefining r)

$$r_n = 0 \; if \; 2^{n-1} \leq K \tag{28l}$$

$$r_{n+1} = r_n^2 + 2r_n(\sum_i |a_{in}| + |X_n)|) + \sum_{ij+} |a_{in}a_{jn}|. \tag{28m}$$

3.2.3 SA error analysis in practice

Now that we know the truncation error (but have ignored rounding errors altogether) the question is what to do with this. One option is to impose backward stability. Error Δw is related to error in Δc through

$$\Delta w = w' \Delta c \tag{29}$$

so a good criterion could be

$$\Delta w_n/|w'_n| = r_n/|w'_n| = \Delta c < h \tag{30}$$

with h the pixel radius, but $|w'_n|$ depends on b and a pixel by pixel test negates the purpose of the SA, to speed up things. Unfortunately $|w'_n|$ (obtained simply from the series of w) has no lower bound, so it's unclear how to proceed. Current practice is to set $|w'| \geq 1$ which is valid except inside hyperbolic components.

If we set $|w'| = 1$ we are effectively directly working with the forward error r_n, which is not necessarily bad. The backwards error is not really the visible error, as this is determined by the behaviour just before escaping, if the orbit escapes, or, if the orbit does not escape and is stable, there is no direct visual interpretation of Δc, and any Δc that does not move out of the orbit attractor is fine for rendering purposes. (Unless you want to do interior coloring.) While this works, the SA should in principle be applied to as close to the minimum iterations in the image as possible, and the

truncation error based method does not get as close as a more heuristic method based on probes. Another problem is that truncation error is just one estimated error and in practice the rounding error from evaluating the SA polynomial, assuming all series coefficients to be accurate to machine precision, is larger by many orders of magnitude than the truncation error, putting the justification of this method on shaky grounds.

A more heuristic method checks selected points for comparison with a full iteration of the orbit using PT. As theoretical upper bounds on the error seem too conservative, and there is probably no way to rigorously guarantee accuracy everywhere except by brute force methods which negate the purpose of the SA, this is a reasonable approach.

As the polynomial has largest errors on the boundary, the four corner points of the image are good candidates for probe points. For each probe point we also calculate the PT orbit and compare at each SA iteration for accuracy, terminating when at any probe we get $\Delta w/|w'| > h$ with h the pixel radius.

Occasionally this does not suffice and the SA is incorrect somewhere in the center. As the error is potentially largest where $|w'|$ is small, which happens near minibrots, a good method is to compute the roots of the SA polynomial or its derivative and use those as probe points, adding probes as you go along. If you find a root c_r it will have $|w'| < 1$, inside a component, and if used as-is will terminate the iteration very quickly as the backward error will be very large. However the root c_r is not on a pixel in general, and it should be shifted to the nearest pixel. In images without visible minibrots this is sufficient as now the derivate will be fairly large again. If there is a minibrot in the image such a shift may still stay in the minbrot with small derivative. As we want the probe to be just outside the mini we should shift it until $|w'| \approx 1$. This can be achieved by the shift

$$c_r \to c_r + e^{i\theta}/w'' \tag{31}$$

with w'' the second derivative obtained from the SA polynomial and θ an arbitrary phase factor determining the direction of the shift. This method works well and allows more iterations to be skipped than any other known method at present.

3.3 Glitch detection

The perturbation method sometimes works as-is, but usually some pixel regions (c values) do not render correctly with the given reference orbit, and require a different reference point. Obviously if the reference orbit escapes before the pixel we are computing the calculation will fail. This can be avoided by ensuring c_r has a non-escaping orbit, which means finding an hyperbolic component in the given image, which is not too difficult.

Harder is to automatically (i.e., without visual inspection) detect when (23b) is going to fail. At first glance it seems this will happen when the FP number w gets big enough that the addition of the b drops out (when $|b| < \epsilon|w|$) but in practice this is sometimes fine. The orbits of w in a neighborhood of $c_r + b$ can still stay

distinct. A problem will arise when they don't, as then a whole region will "flatten out". This reasoning led to the Pauldelbrot glitch detector [4] (the errors are usually called "glitches").

Once b drops out, the iteration for w and $w' = \frac{\partial w}{\partial b}$ become

$$w_{k+1} = w_k(w_k + 2x_k) \tag{32a}$$
$$w'_{k+1} = 2(x_k + w_k)w'_k. \tag{32b}$$

If w' ever becomes zero, nearby orbits will merge, and this will happen if $|x_k + w_k = z_k| = 0$. So whenever the orbit z_k becomes small we may have a problem. The Pauldelbrot criterion for a glitch to occur is

$$|x_k + w_k|/|x_k| < \tau \tag{33}$$

with the tolerance τ about $10^{-6} - 10^{-3}$, depending on how complicated the image is. The intuition is that the presence of a dip in z indicates the influence of a nearby mini different from the reference mini, which explains the division by $|x_k|$. The method works well, but detects many "false glitches" (that had no problem), is not well motivated theoretically, and requires an empirically determine tolerance parameter.

A different glitch detector was first proposed in [2], based on a rounding error analysis of (23b), taking into account only the rounding error in the term $w_k + 2x_k$, which is $\epsilon \max(|w_k|, 2|x_k|)$. Here I will derive a more conservative form, taking more errors into account. We set the relative error in x_n and b to ϵ, and the relative error in w to ϵ_r. If no SA is used, $\epsilon_r = \epsilon$, otherwise $\epsilon_r = r_M/|w_M|$ with M the last SA computed w and error r_M. (Injecting SA error in glitch detection has not been tried out, in practice with SA the glitch methods to follow are tweaked by just increasing ϵ by some factor, say 1000.

In (23b), replace x and b by $x + \epsilon|x|$ and $b + \epsilon|b|$ (the rounding error), and w by $w + \alpha\epsilon|w|$ where we can think of ϵ (machine precision) as a ball with undetermined phase, keeping only linear terms in ϵ. to get the error in w, Δw. If no SA is used $\alpha = 1$, but if the error from the last SA iteration would be used for the error in w $\alpha = \frac{r_M}{\epsilon|w_M|}$ with M the number of iterations skipped by the SA.

$$w_{k+1} = w_k(w_k + 2x_k) + b + \Delta w_{k+1} \tag{34a}$$
$$\Delta w_{k+1} = (|w_k|(\alpha|w_k + 2x_k| + \alpha|w_k| + 2|x_k| + |b|)\epsilon. \tag{34b}$$

While it's unclear what limits to put on the "forward error" Δw, the "backward error", the change in c corresponding to the change in w has a clear interpretation: it should be less than half the pixel distance (h). The relation between forward and backward error is

$$\Delta w_{k+1} = w'_{k+1}\Delta b \tag{35}$$

which gives the no-glitch condition

$$\Delta b = \Delta w_{k+1}/|w'_{k+1}| = \frac{|w_k|(\alpha|w_k + 2x_k| + \alpha|w_k| + 2|x_k|) + |b|}{|2w'_k(w_k + x_k) + 1_k|}\epsilon < h \tag{36}$$

or the equivalent glitch condition

$$\frac{|2w'_k(w_k + x_k) + 1_k|}{|w_k|(\alpha|w_k + 2x_k| + \alpha|w_k| + 2|x_k|) + |b|} < \epsilon/h. \tag{37}$$

Strictly speaking, the term denoted 1_k should be 0 if b has dropped out, and 1 if not, but in practice it makes no difference if it's kept or not.

This condition seems to work well, with fewer false positives, but particularly if the SA is used it requires a manually set tolerance by replacing $\epsilon \to \epsilon/\tau$ with tolerance τ about $1 - 10^{-3}$, depending on the complexity of the image and the SA parameters.

We can try to be more rigorous and keep track of the backward errors Δb at each iteration and sum them up (I call this the "step method"), and this does seem to solve the problem for the case of no SA, and we always have $\tau = 1$, at least I am not aware of any exceptions.

The condition (37) looks quite different from (33), but we can see that the condition (37) can be triggered if w'_k becomes small (or rather not very large, as for deep zooms the right hand side is huge) and/or if $w_k + x_k$ becomes small. It is unclear if (37) is ever triggered when (33) is not, with "similar" tolerance.

In actual implementations the absolute values (2 norm of (x, y))in (37) can be approximated by the infinity norm ($\max(|x|, |y|)$) or 1 norm ($|x| + |y|$) as they are not far off from the 2 norm, to avoid taking square roots.

3.4 Glitch resolution

After rendering an image there usually are many glitched points that failed the error test, and the question is how to proceed. To finish processing the glitched pixels a difference reference point c_r is required. This will generate sub-glitches, so many new references will have to be generated. There are several options that all work, but which is the best is unclear at this point. I'll just list the possiblitities that have been tried that I am aware of.

1) Use a higher period hyperbolic center as c_r. Usually (but not always) higher period non-escaping reference orbits generate fewer glitches. One option is to run the period detector beyond the first lowest period and find several more, but this may not be enough and we probably should have used the highest period for the first reference anyways. Another option is subdivision of the rendering rectangle and find new periods and minis in the subregions.

2) Pick a random glitched point and use it as reference. It will generally escape, but it's cheap to compute as we already have the orbit to just before the glitch

occurred. It's also possible to select the "worst" glitch, which sounds better, but if it fails we're out of luck, whereas with the random method we'll eventually get lucky.

3) Based on the observation that using the Pauldelbrot glitch detector the glitched point has a small z_n, assume this minimum indicates the presence of a nearby mini, where this minimum will be exactly zero. So find this mini of period n with one or more Newton iterations and use that. It is unclear if this always works, it could fail if the glitched point is outside the basin of attraction for the Newton method.

4 Finding minis

To locate a mini in a region, we find the period first, then use Newton's method to find the location, and compute the size using the renormalization formula (19b). To find the period in a region D we have to find the smallest p such that the image of D under the map $z_p(D)$ contains 0, as then there will be a point c_0 in D where $z_p(c_0) = 0$.

A method that works well in practice [18] defines D as a polygon (say triangle, or rectangle) and approximates its image under z iterations by the polygon formed by the iterated vertices. Once the deformed polygon surrounds the origin, we have found a period. You can then stop or continue and usually find several more (higher) periods. This is a robust method, and applies to more general $f(z, c)$. It can fail when the vertices escape before a period is detected, and in theory a detected period could be wrong but in practice this is rare if it ever happens at all. It requires the computation of at least 3 (for triangle) full precision orbits.

For holomorphic polynomials (in just z) there is a better method which requires only 1 full precision orbit using "ball arithmetic" [15].

4.1 Ball arithmetic

We consider expressions of the form $[z] = z + rE$ with z a complex number, r a real number and E a "unit ball". We can think of E as being parametrized by polar coordinates $0 \le \rho < 1$ and $0 \le \theta < 2\pi$ as $E = \rho e^{i\theta}$, and $[z]$ represents a region, parametrized by (ρ, θ). It is z surrounded by a ball (disc) of radius r. Arithmetic of "ball" expressions $[z]$ obey some funny rules. I'll introduce (non-standard) notation. $[u] \simeq [v]$ means both regions are the same. $[u] \subseteq [v]$ means region $[u]$ lies in region $[v]$. Now for some rules and examples. Balls E with subscript are independent, having

their own parameters.

$$zE \simeq |z|E \tag{38a}$$
$$E^n \simeq E \ for \ (n > 0) \tag{38b}$$
$$||u| - |v||E \subseteq uE + vE \subseteq (|u| + |v|)E \tag{38c}$$
$$(u + v)E \simeq |u + v|E \tag{38d}$$
$$1/4E \subseteq (E/2) - (E/2)^2 \subseteq 3/4E \tag{38e}$$
$$(E_1/2) - (E_2/2)^2 \simeq 3/4E \tag{38f}$$

The region defined in (38e) is the M-set main cardioid, and the formula shows it is contained in a ball of radius 3/4, and a ball of radius 1/4 fits inside it. Usually for small balls and the M-set function, the upper limits are very close to the actual image under iteration, until the ball gets large and starts escaping.

Functions $f([z])$ of ball expressions are defined only for (possibly infinite) polynomials in z.

We find the period in a ball of radius r_c around c by computing the iteration

$$[z_{k+1}] = z_{k+1} + r_{k+1}E = [z_k]^2 + c + r_cE \tag{39}$$

which gives us $z_k(c + r_cE)$, and whenever $r_k > |z_k|$ the ball contains the origin, and we have found the period. We get

$$z_{k+1} + r_{k+1}E = (z_k + r_kE)^2 + c + r_cE \subseteq z_k^2 + c + (2|z_k|r_k + r_k^2 + r_c)E \tag{40}$$

giving

$$z_{k+1} = z_k^2 + c \tag{41a}$$
$$r_{k+1} = r_k^2 + 2|z_k|r_k + r_c \tag{41b}$$
$$r_0 = 0 \tag{41c}$$
$$if \ r_{k+1} > z_{k+1} \ period = k + 1. \tag{41d}$$

The r iteration can be done in FP. This "first order" method works, but a better method that escapes later (so can detect higher periods) can be constructed using ball arithmetic augmented with a Taylor expansion.

Introduce a new entity $\hat{E}$ which represents a definite but unknown point in the unit disc E. We have $\hat{E} \subseteq E$ which states the obvious fact that any point of $\hat{E}$ is in E. We have

$$z_n(c + r_c\hat{E}) \subseteq z_n(c) + z_n'(c)r_c\hat{E} + r_nE \equiv [z_n] \tag{42a}$$

where r_n can be thought of as being of second order in r_c, which is assumed small. In manipulating such expressions we keep the phase in all linear expressions in $\hat{E}$ but

raising $\hat{E}$ to a power reduces it to E as the phase is now messed up. We get with these reduction rules

$$
\begin{aligned}
&[z_{n+1}] = z_{n+1} + (2z_n z'_n + 1) r_c \hat{E} + r_{n+1} E \\
&[z_{n+1}] = (z_n + z'_n r_c \hat{E} + r_n E)^2 + c + r_c \hat{E} = \\
&z_n^2 + c + r_c^2 |z'_n|^2 E + r_n^2 E + 2 z_n z'_n r_c \hat{E} + r_c \hat{E} + 2 r_n |z_n| E + 2 r_c r_n |z'_n| E = \\
&z_{n+1} + (2 z_n z'_n + 1) r_c \hat{E} + (r_n^2 + 2 r_n (|z_n| + r_c |z'_n|) + r_c^2 |z'_n|^2) E
\end{aligned}
$$

and by equating first and last equations we get

$$
\begin{aligned}
z_{n+1} &= z_n^2 + c && \text{(43a)} \\
z'_{n+1} &= 2 z_n z'_n + 1 && \text{(43b)} \\
r_{n+1} &= r_n^2 + 2 r_n (|z_n| + r_c |z'_n|) + r_c^2 |z'_n|^2 && \text{(43c)} \\
r_1 &= 0 && \text{(43d)}
\end{aligned}
$$

and we have to check if $z_n + (r_c |z'_n| + r_n) E$ (where we have reduced $\hat{E}$) contains the origin, which happens when $r_n + r_c |z'_n| > |z_n|$.

This second order method is more accurate but we have to compute the derivative. This may not be a waste of time as we usually want to follow up with a Newton search for the location of the mini, and we get the first iteration $(-z_n / z'_n)$ for free from this method.

For all these methods the iteration can be continued after finding the smallest period, until the ball or polygon escapes. Often this results in finding more (sometimes dozens) periods in the region.

5 Super series approximation

Using renormalization (see Section 1) we can replace p iterations inside a nucleus of period p with one iteration of the renormalized function (19c) which speeds up the calculation of the interior points by a factor p.

An extension of this idea using higher order polynomial approximation was proposed in [5] and initial results seem promising. Let's call this the super series approximation (SSA).

Consider the PT iteration scheme (23) and assume the reference point c_r is the nucleus of a period p mini. So we have $X_{kp} = 0$, for $k = 0, 1, \ldots$.

Define polynomials $f_k(w, b)$ (w is now just a variable, not related to the w_k from (23)) as follows:

$$f_1(w, b) = w(w + x_0) + b = w^2 + b \tag{44a}$$

$$f_{n+1}(w, b) = f_n(w, b)(f_n(w, b) + 2x_n) + b \tag{44b}$$

and use it to calculate $f_p(w, b)$. We have $w_p = f_p(0, b)$, i.e., one iteration of f_p is the same as p iterations of (23).

The SSA idea is to approximate $f_p(w, b)$ with a lower order bivariate polynomial in w and b of order (K, M):

$$\sum_{ij} \equiv \sum_{i=0}^{K} \sum_{j=0}^{M} \tag{45a}$$

$$\hat{b} \equiv b/b_{max} \tag{45b}$$

$$f_p(w, b) = \sum_{ij} a_{ij}(p) w^i \hat{b}^j + r_p E \tag{45c}$$

where $r_p E$ (using ball notation, see Section 4.1) represents the truncation error and b_{max} is the maximum of $|b|$ in the region of interest. The scaling simplifies formulas though of course not necessary. Note that $f_p(0, b)$ is just the usual SA, skipping p iterations.

Precompute the coefficients a_{ij}, then iterate $f_p(0, b)$ until a yet to be determined bailout. If bailout occurs at super iteration k we have now computed pk normal iterations and can continue from there with normal PT iterations. This is similar to regular SA except different regions now skip by different amounts. Perhaps instead of continuing with normal PT after super bailout there are smarter ways, using SSA from other nuclei.

In the remainder of this section I will now derive the recursion for the SSA coefficients and the truncation error estimate. Just substitute (45) into (44) and match powers and ball terms. To simplify notation let's define $a_{ij} = 0$ for ij "out of range" (negative of bigger than K or M). Also when iteration number is omitted in $a(.)$,

$x_\cdot$, and $r_\cdot$ it is n by implication in the formulas below. For $a_{ij}(1)$ we get the only non-zero coefficients $a_{20}(1) = 1$ and $a_{01}(1) = b_{max}$ and for the recursion we get

$$f_{n+1} = \sum_{ij} a_{ij}(n+1) w^i \hat{b}^j + r_{n+1} E = \tag{46a}$$

$$(\sum_{ij} a_{ij} w^i \hat{b}^j + rE)(\sum_{ij} a_{ij} w^i \hat{b}^j + 2x + rE) + b_{max}\hat{b} = \tag{46b}$$

$$\sum_{ij,kl} a_{ij} a_{kl} w^{i+k} \hat{b}^{j+l} + \sum_{mq} (2x a_{mq} + b_{max}\delta_{0m}\delta_{1q}) w^m \hat{b}^q +$$

$$(r^2 + 2r(x + \sum_{mq} a_{mq} w^m \hat{b}^q))E = \tag{46c}$$

$$\sum_{mq} \left(\sum_{i,j=0}^{(m,q)} a_{ij} a_{m-i,q-j} + 2x a_{mq} + b_{max}\delta_{0m}\delta_{1q} \right) w^m \hat{b}^q + \tag{46d}$$

$$(r^2 + 2r(x + \sum_{mq} a_{mq} w^m \hat{b}^q))E + \tag{46e}$$

$$\sum_{m,q=(0,0)}^{(2K,2M)} (m > K | q > M)(\sum_{i,j=0}^{(m,q)} a_{ij} a_{m-i,q-j}) w^m \hat{b}^q \tag{46f}$$

Where the last sum is over powers greater than K or M. From (46d) we obtain the recursion for the coefficients as

$$a_{mq}(n+1) = \sum_{i,j=0}^{(m,q)} a_{ij}(n) a_{m-i,q-j}(n) + 2x_n a_{mq}(n) + b_{max}\delta_{0m}\delta_{1q} \tag{47}$$

and to obtain the (b and w independent) recursion for r_n we have to put upper bounds on w and b in (46e) and (46f). We have $|\hat{b}| \leq 1$ and $|w| \leq w_{max}(n)$, where a choice could be $w_{max}(n) = |x_n| + R_s$ where R_s is the super escape radius for the f_p iterations. If R_s is small perhaps we can just set $w_{max} = 2$.

With this we get the recursion for r_k as follows:

$$r_{n+1} = r_n^2 + 2r_n(|x_n| + \sum_{mq} |a_{mq}| w_{max}^m(n)) +$$

$$\sum_{m,q=(0,0)}^{(2K,2M)} (m > K | q > M) \sum_{i,j=0}^{(m,q)} |a_{ij} a_{m-i,q-j}| w_{max}^m(n). \tag{48}$$

However, **this does not work** because r just diverges quickly as the bound on $|w|$ is way too loose. Instead we can be less sophisticated and simply estime the error in f_n by the magnitude of the terms with the three next highest powers, so

$$r_n(w, b) = |a_{K,M+1}\hat{b} + a_{K+1,M} w + a_{K+1,M+1} w\hat{b}||w^K \hat{b}^M|. \tag{49}$$

We now compute $a_{ij}(p) \equiv a_{ij}$ up to order $(K+1, M+1)$, using the last three terms only for error estimation. The next step is the error propagation in iterating $f_p = \sum_{ij} a_{ij}(p) w^i \hat{b}^j$. This will give us an error for every f_p iteration at every point b and could be used as a stopping condition.

$$u_0 = 0 \tag{50}$$
$$u_{n+1} = f_p(u_n, b) \tag{51}$$

with $u_n = w_{pn}$. In ball notation we write $[u_n] = u_n + \rho_n E$ which gives

$$[u_0] = 0 \tag{52}$$
$$[u_{n+1}] = u_{n+1} + \rho_{n+1} E = f_p(u_n + \rho_n E, b) + r_p(u_n + \rho_n E, b) E. \tag{53}$$

Defining $B_i(b) = \sum_{j=0}^{M} a_{ij} \hat{b}^j$, dropping the n in subscripts and b dependency in B_i, and writing $q_1 = a_{K,M+1}$ $q_2 = a_{K+1,M+1}$ and $q_3 = a_{K+1,M+1}$ for notation and collecting terms containing E we get

$$[u_{n+1}] = \sum_{k=0}^{K} k B_k (u + \rho E)^k + |q_1 \hat{b}^{M+1} (u + \rho E)^K + (q_2 \hat{b}^M + q_3 \hat{b}^{M+1})(u + \rho E)^{K+1}| \tag{54a}$$

which we could expand with binomial sums but for now let's just keep the linear terms in ρ (which should be small). We get

$$[u_{n+1}] = u_{n+1} + \sum_{k=1}^{K} |k B_k u^{k-1}| \rho E + |q_1 \hat{b}^{M+1} u^K + (q_2 \hat{b}^M + q_3 \hat{b}^{M+1}) u^{K+1}| E + \left(|K q_1 \hat{b}^{M+1} u^{K-1}| + (K+1)(|q_2 \hat{b}^M| + |q_3 \hat{b}^{M+1}) u^K|) \right) \rho E. \tag{54b}$$

Collecting the E terms gives

$$\rho_0 = 0 \tag{54c}$$
$$\rho_{n+1} = |q_1 \hat{b}^{M+1} u^K + (q_2 \hat{b}^M + q_3 \hat{b}^{M+1}) u^{K+1}| + \left(\sum_{k=1}^{K} |k B_k u^{k-1}| + |K q_1 \hat{b}^{M+1} u^{K-1}| + (K+1)(|q_2 \hat{b}^M| + |q_3 \hat{b}^{M+1}|)|u^K|) \right) \rho_n \tag{54d}$$

6 Other things

6.1 Finding Misiurewics points

A similar method to find islands using period detection and Newton iteration can be used to find Misiurewicz points (pp, p) of period p and preperiod pp, but just solving $z_{pp}(c) = z_{pp+p}(c)$ could find a lower preperiod. This is solved by eliminating those solutions by division [12].

6.2 Atom domains

An atom is a hyperbolic component (approximately in the shape of a cardioid or circle) and its nucleus is that point c inside where the orbit is periodic with period p, so $z_{kp}(c) = 0$ for any k. Outside points near an atom have orbits which look a bit like the atom orbit, except z_p is small instead of zero, z_{2p} a little less small, until the orbit escapes. These dips in the orbit indicate the influence of nearby atoms (there can be dips at multiple periods). The smallest dip can be identified and by doing this you get a picture of the influence regions of the atoms, even if no mini is visible in the picture. Details and algorithm can be found here [13], including a domain size estimate requiring the computation of just the nucleus orbit.

A similar idea can be applied to Misiurewicz domains [14].

6.3 Software

Fast zooming with perturbation theory is implemented in Kalles Fractaler, Mandel-Machine, and UltraFractal, easy to find on the WWW. Code examples can be found by following links from some of the references, and here [6].

References

[1] http://www.cns.gatech.edu/~predrag/courses/PHYS-6124-11/StGoChap17.pdf.

[2] http://www.fractalforums.com/announcements-and-news/*continued*-superfractalthing-arbitrary-precision-mandelbrot-set-rendering-in-ja/msg91505/#msg91505.

[3] http://www.fractalforums.com/announcements-and-news/*continued*-superfractalthing-arbitrary-precision-mandelbrot-set-rendering-in-ja/msg95532/#msg95532.

[4] http://www.fractalforums.com/index.php?topic=18908.msg73027#msg73027.

[5] https://fractalforums.org/fractal-mathematics-and-new-theories/28/another-possible-way-to-accelerate-mb-set-deep-zooming/277/.

[6] https://en.wikibooks.org/wiki/Fractals.

[7] S. Blanes, F. Casas, and A. Murua. Splitting and composition methods in the numerical integration of differential equations. https://arxiv.org/pdf/0812.0377.pdf.

[8] Y. Dang, L. H. Kaufmann, and D. Sandin. Hypercomplex iterations, distance estimation and higher dimensional fractals. https://www.evl.uic.edu/hypercomplex/html/book/book.pdf.

[9] Claude Heiland-Allen. https://mathr.co.uk/blog/2015-01-26_newtons_method_for_misiurewicz_points.html.

[10] Claude Heiland-Allen. https://mathr.co.uk/blog/2016-12-24_deriving_the_size_estimate.html.

[11] Claude Heiland-Allen. https://mathr.co.uk/blog/2016-03-06_simpler_series_approximation.html.

[12] Claude Heiland-Allen. https://mathr.co.uk/blog/2015-01-26_newtons_method_for_misiurewicz_points.html.

[13] Claude Heiland-Allen. https://mathr.co.uk/blog/2013-12-10_atom_domain_size_estimation.html.

[14] Claude Heiland-Allen. https://mathr.co.uk/blog/2015-02-01_misiurewicz_domains.html.

[15] J. Van Der Hoeven. Ball arithmetic (hal-00432152v1). https://hal.archives-ouvertes.fr/hal-00432152v1, 2011.

[16] Philippe Langlois and Nicolas Louvet. Faithful polynomial evaluation with compensated horner algorithm. https://arxiv.org/pdf/cs/0610122.pdf, 2017.

[17] K. I. Martin. Superfractalthing maths. http://superfractalthing.co.nf/sft_maths.pdf.

[18] Robert P. Munafo. http://mrob.com/pub/muency/period.html.

[19] Wikipedia. https://en.wikipedia.org/wiki/Recurrence_relation#Solving_first-order_non-homogeneous_recurrence_relations_with_variable_coefficients.

[20] J.A. Yorke, C. Grebogi, E. Ott, and L. Tedeschini-Lalli. Scaling behavior of windows in dissipative dynamical systems. *Phys.Rev.Lett.*, (54):1095, 1995.