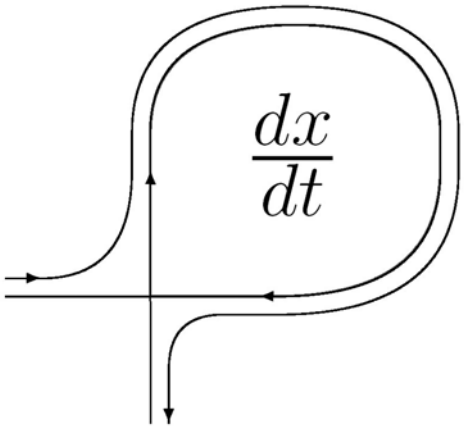

ДИФФЕРЕНЦИАЛЬНЫЕ УРАВНЕНИЯ
И
ПРОЦЕССЫ УПРАВЛЕНИЯ
№ 3, 2007
Электронный журнал,
рег. № П2375 от 07.03.97
ISSN 1817-2172

http://www.newa.ru/journal
http://www.math.spbu.ru/user/diffjournal
e-mail: jodiff@mail.ru

Моделирование динамических систем

РЕАЛИЗИЦИЯ АЛГОРИТМОВ ПОСТРОЕНИЯ ГИПЕРКОМПЛЕКСНЫХ МНОЖЕСТВ ЖЮЛИА И МАНДЕЛЬБРОТА

РАБА Н.О.

Россия, 198504, Санкт-Петербург,
Петродворец, Университетский пр., дом 28,
Санкт-Петербургский Государственный Университет
математико-механический факультет
e-mail: no13@inbox.ru

Аннотация

В данной статье описываются методы и алгоритмы визуализации гиперкомплексных множеств Жюлиа и Мандельброта: быстрый, но дающий не очень хорошие результаты метод обратных итераций (подходящий только для визуализации множества Жюлиа), или более точный, но требующий намного больше вычислений метод трассировки лучей. Для метода трассировки лучей приводятся два способа нахождения пересечения луча с множеством: усовершенствованный классический и использующий оценку расстояния до множества. Также дается оценка скорости и качества работы алгоритмов.

1 Введение

Множества Жюлиа квадратичной функции (и других функций) существуют не только на комплексной плоскости, но и в пространствах с большими размерностями, в частности, в гиперкомплексном пространстве. Так как комплексная плоскость – это подмножество гиперкомплексного пространства, то любое комплексное множество Жюлиа существует в гиперкомплексном пространстве и имеет продолжение вне комплексной плоскости. Гиперкомплексное множество Жюлиа содержит больше информации, чем его комплексное подмножество.

Визуализация гиперкомплексных множеств Жюлиа более сложная задача по сравнению с комплексными множествами. Гиперкомплексное пространство – это четырехмерное пространство, а человек воспринимает только трехмерное (в действительности, проекцию трехмерного – двухмерное). Поэтому необходимо проецировать четырехмерную фигуру на плоскость.

В [1] и [2] дается определение фракталов, комплексных множеств Мандельброта и Жюлиа, двоичного разбиения, описываются основные алгоритмы для их построения. В [3] дается теоретическая основа и алгоритмы для построения гиперкомплексных множеств Жюлиа и Мандельброта. Алгоритм построения множеств Жюлиа с помощью метода обратных итераций приведен в [4], где также объясняются некоторые полезные для этого метода свойства кватернионов. В [5] описывается метод трассировки лучей с использованием оценки расстояния для построения гиперкомплексных множеств Жюлиа. В [6] описывается классический метод трассировки лучей.

В данной статье для метода трассировки лучей предложены два способа нахождения пересечения луча с множеством. Первый – усовершенствованный классический, работающий быстрее классического при одинаковой точности (либо дающий большую точность при одинаковом времени работы). Второй способ использует оценки расстояния, приведенные в [3].

2 Кватернионы

Кватернионы (гиперкомплексные числа) были введены Гамильтоном в 19 веке. Множество гиперкомплексных чисел обозначается **H**.

По аналогии с комплексным числом, кватернион $\mathbf{q} = \mathbf{r} + \mathbf{a}\cdot\mathbf{i} + \mathbf{b}\cdot\mathbf{j} + \mathbf{c}\cdot\mathbf{k}$, где **i**, **j**, **k** – мнимые единицы ($\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = -\mathbf{1}$), **r**, **a**, **b**, **c** – вещественные. Таким образом, гиперкомплексные числа порождаются вещественной единицей **1** и тремя мнимыми **i**, **j**, и **k**.

Обозначения:

$\mathbf{Re(q) = r}$ – вещественная часть **q**, $\mathbf{Im_i(q) = a}$, $\mathbf{Im_j(q) = b}$, $\mathbf{Im_k(q) = c}$,

$\mathbf{Pu(q) = Im_i(q)\cdot i + Im_j(q)\cdot j + Im_k(q)\cdot k = a\cdot i + b\cdot j + c\cdot k}$ – мнимая часть **q**,

$\mathbf{S = \{q \in H: |q| = 1\}}$ – сфера радиуса **1** в **H**,

$\mathbf{S^2 = \{q \in H: |q| = 1, Re(q) = 0\}}$.

Кватернион, у которого вещественная часть нулевая, называется чистым кватернионом. Если **q** – чистый кватернион, то **Pu(q) = q**.

Пусть **P** – пространство (трехмерное) чистых кватернионов. Тогда $\mathbf{H = R \oplus P}$, так как **q = Re(q) + Pu(q)**, $\mathbf{Re(q) \in R}$, $\mathbf{Pu(q) \in P}$.

$\mathbf{S^2}$ можно определить по-другому: $\mathbf{S^2 = \{q \in P: |q| = 1\}}$ – сфера чисто мнимых точек радиуса **1**.

Гиперкомплексное пространство – продолжение комплексной плоскости (по аналогии с тем, что комплексная плоскость – продолжение вещественной прямой). Комплексные числа – гиперкомплексные, у которых $\mathbf{Im_j = Im_k = 0}$ (вещественные, если еще и $\mathbf{Im_i = 0}$).

Иногда кватернионы представляют в виде

- вещественного числа **r** и 3-х мерного вектора **(a, b, c)** (разложение **H** на **R** и **P**: $\mathbf{r \in R}$, $\mathbf{(a,b,c) \in P}$)
- $\mathbf{c_1 + c_2\cdot j}$, где $\mathbf{c_1}$, $\mathbf{c_2}$ – комплексные, $\mathbf{c_1 = r + a\cdot i}$, $\mathbf{c_2 = b + c\cdot i}$

- **R·(A + B·u) = R·cos(θ) + R·sin(θ)·u**, **A**, **B**, **R**, **θ** – вещественные, **R** – модуль кватерниона (см. далее), $\mathbf{A^2 + B^2 = 1}$, $\mathbf{u \in S^2}$.

2.1 Действия с кватернионами

Пусть $q = r + a \cdot i + b \cdot j + c \cdot k$, $q_1 = r_1 + a_1 \cdot i + b_1 \cdot j + c_1 \cdot k$, $q_2 = r_2 + a_2 \cdot i + b_2 \cdot j + c_2 \cdot k$

Сложение и вычитание выполняются покомпонентно:

$$q_1 + q_2 = (r_1 + r_2) + (a_1 + a_2) \cdot i + (b_1 + b_2) \cdot j + (c_1 + c_2) \cdot k$$

$$q_1 - q_2 = (r_1 - r_2) + (a_1 - a_2) \cdot i + (b_1 - b_2) \cdot j + (c_1 - c_2) \cdot k$$

Умножение кватернионов похоже на умножение полиномов с тремя переменными **i**, **j**, **k**, но со специфическими свойствами:

$$i^2 = j^2 = k^2 = -1, \quad i \cdot j = k, \quad j \cdot k = i, \quad k \cdot i = j, \quad j \cdot i = -k, \quad k \cdot j = -i, \quad i \cdot k = -j .$$

Поэтому умножение не коммутативно.

$$q_1 \cdot q_2 = (r_1 r_2 - a_1 a_2 - b_1 b_2 - c_1 c_2) + (r_1 a_2 + a_1 r_2 + b_1 c_2 - c_1 b_2) \cdot i +$$
$$(r_1 b_2 + b_1 r_2 + c_1 a_2 - a_1 c_2) \cdot j + (r_1 c_2 + c_1 r_2 + a_1 b_2 - b_1 a_2) \cdot k$$

Сопряжение: $q^* = r - a \cdot i - b \cdot j - c \cdot k$

Модуль: $| q | = \sqrt{q \cdot q^*} = \sqrt{r^2 + a^2 + b^2 + c^2}$

Свойство: $| q_1 \cdot q_2 | = | q_1 | \cdot | q_2 |$

Обратное: $q^{-1} = \dfrac{q^*}{| q |^2}$, $\quad q \cdot q^{-1} = 1$

Деление: $\dfrac{q_1}{q_2} = q_1 \cdot q_2^{-1}$

Скалярное произведение: $< q_1, q_2 > = \mathrm{Re}(q_1 \cdot q_2^*) = r_1 r_2 + a_1 a_2 + b_1 b_2 + c_1 c_2$

Нормализация: $Un(q) = \dfrac{q}{| q |}$, если $q \neq 0$

$Un(q) \in S$, а если $q \in P$, то $Un(q) \in S^2$

Возведение в квадрат: $q^2 = (r^2 - a^2 - b^2 - c^2) + 2ra \cdot i + 2rb \cdot j + 2rc \cdot k$

Возведение в степень $n \in N$: $q^n = \underbrace{q \cdot q \ldots q}_{n}$

Но удобнее представить **q** в виде $R\cdot\cos(\theta)+R\cdot\sin(\theta)\cdot u$, где $u\in S^2$, и воспользоваться формулой: $q^n=R^n\cdot\cos(n\theta)+R^n\cdot\sin(n\theta)\cdot u$

Извлечение корня

Корнем (квадратным) из **q** называется такое **z**, что $\mathbf{z^2 = q}$.

В **R**, так же, как и в **C**, возведение в квадрат – отображение 2 к 1 (везде, кроме **0**). Но в **H** это не так. Здесь возможно отображение бесконечного числа точек в одну. Например, $\mathbf{q^2}$**:** $\mathbf{S^2 \to -1}$. В общем, любая сфера чисто мнимых точек с радиусом $\mathbf{r \in R_+}$ ($\mathbf{S^2\cdot r}$), свертывается в одну точку на $\mathbf{R_-}$ (а именно, в точку $\mathbf{-r^2}$).

Их этого следует, что корней из **q** может быть бесконечное число.

Возможны следующие случаи:

1) $\mathbf{q = 0} \Rightarrow$ 1 корень равный **0**
2) $\mathbf{q = r}$ ($\mathbf{r \in R}$, $\mathbf{r > 0}$) $\Rightarrow$ 2 корня, $\pm\sqrt{r}$
3) $\mathbf{q = r}$ ($\mathbf{r \in R}$, $\mathbf{r < 0}$) $\Rightarrow$ бесконечное множество корней
$$S^2\cdot\sqrt{-r}=\{p\in H:|\,p\,|=\sqrt{-r},\ \mathrm{Re}(p)=0\}=\{p\in P:|\,p\,|=\sqrt{-r}\}$$
4) $\mathbf{q \in H}$, $\mathbf{Pu(q) \neq 0} \Rightarrow$ 2 корня: $r+\frac{\mathrm{Im}_i(q)}{2r}+\frac{\mathrm{Im}_j(q)}{2r}+\frac{\mathrm{Im}_k(q)}{2r}$, где $r=\pm\sqrt{\frac{\mathrm{Re}(q)+|\,q\,|}{2}}$

2.2 Повороты 3-х мерного пространства

Одно из самых интересных свойств кватернионов – это возможность представлять повороты 3-х мерного пространства.

Представим 3-х мерное пространство $\mathbf{R^3}$ в виде пространства чистых кватернионов **P**. Если **g** – единичный кватернион ($|\mathbf{g}| = \mathbf{1}$), то определим отображение **ρ**: $\mathbf{R^3 \to R^3}$ уравнением

$$\mathbf{\rho(Z) = gZg^{-1}}, \text{ где } \mathbf{Z \in P},$$

описывающим поворот 3-х мерного пространства вокруг оси **u** на угол **θ**, где

$$\mathbf{g = \cos(\theta/2) + \sin(\theta/2)\cdot u}$$

Здесь **u** – чистый кватернион единичной длины, следовательно, вектор в 3-х мерном пространстве. Любой единичный кватернион можно представить в таком виде.

Этот метод представления поворотов с помощью кватернионов очень полезен в компьютерной графике для выполнения вращения без использования матриц.

Следствие 1. Для любого кватерниона **q** существует единичный кватернион **p** такой, что $\mathbf{p}^{-1}\cdot\mathbf{q}\cdot\mathbf{p}$ – комплексное число. То есть любой кватернион может быть повернут в комплексную плоскость с помощью соответствующего единичного кватерниона.

Следствие 2. Можно определить ось вращения, являющегося композицией вращений, перемножив соответствующие этим вращениям кватернионы.

2.3 Гиперкомплексные полиномы

Комплексные числа можно рассматривать как подмножество гиперкомплексных, а комплексные полиномы – как гиперкомплексные полиномы. Например, выражение вида $\mathbf{a}\cdot\mathbf{x}^2 + \mathbf{b}\cdot\mathbf{x} + \mathbf{c}$, где $\mathbf{a}, \mathbf{b}, \mathbf{c} \in \mathbf{C}$, является также гиперкомплексным. Однако, некоммутативность умножения кватернионов влечет за собой то, что многие полиномы не могут быть так просто описаны. Например, следующие полиномы не эквивалентны, за исключением случая, когда **a** и **b** вещественные

$$\mathbf{x}^2\cdot\mathbf{a} + \mathbf{b}\cdot\mathbf{x} + \mathbf{c},$$

$$\mathbf{x}^2\cdot\mathbf{a} + \mathbf{x}\cdot\mathbf{b} + \mathbf{c},$$

$$\mathbf{x}\cdot\mathbf{a}\cdot\mathbf{x} + \mathbf{b}\cdot\mathbf{x} + \mathbf{c} \text{ и т.д.}$$

Гиперкомплексные полиномы могут быть определены так

$$p(x) = \sum_{k=0}^{l}\sum_{i=0}^{m} p_{i0}\cdot x\cdot p_{i1}\cdot x\ldots x\cdot p_{ik},$$

где **l** и **m** целые, **l** – степень полинома, $\mathbf{p_{ij}} \in \mathbf{H}$.

2.4 Гиперкомплексные множества Жюлиа и Мандельброта

Пусть $\mathbf{f}$: $\mathbf{H} \to \mathbf{H}$ (например, $\mathbf{f(h)} = \mathbf{f_q(h)} = \mathbf{h}^2 + \mathbf{q}$).

Бассейн притяжения бесконечно удаленной точки (∞) для отображения **f**

$$\mathbf{A_f}(\infty) = \{\mathbf{h} \in \mathbf{H}: |\mathbf{f}^{\mathbf{n}}(\mathbf{h})| \to \infty, \text{ если } \mathbf{n} \to \infty\}.$$

Множество Жюлиа для отображения **f** – это граница множества $\mathbf{A_f}(\infty)$., т.е.

$$\mathbf{J(f) = j\,A_f(\infty)},$$

наполненное множество Жюлиа

$$\mathbf{K(f) = H \setminus A_f(\infty) = \{\, h \in H:\ |f^n(h)|} \text{ ограничен, если } \mathbf{n \to \infty\}}.$$

Множество Мандельброта

$$\mathbf{M = \{q \in H:\ 0 \in K(f_q)}, \text{ где } \mathbf{f_q(h) = h^2 + q\}}$$

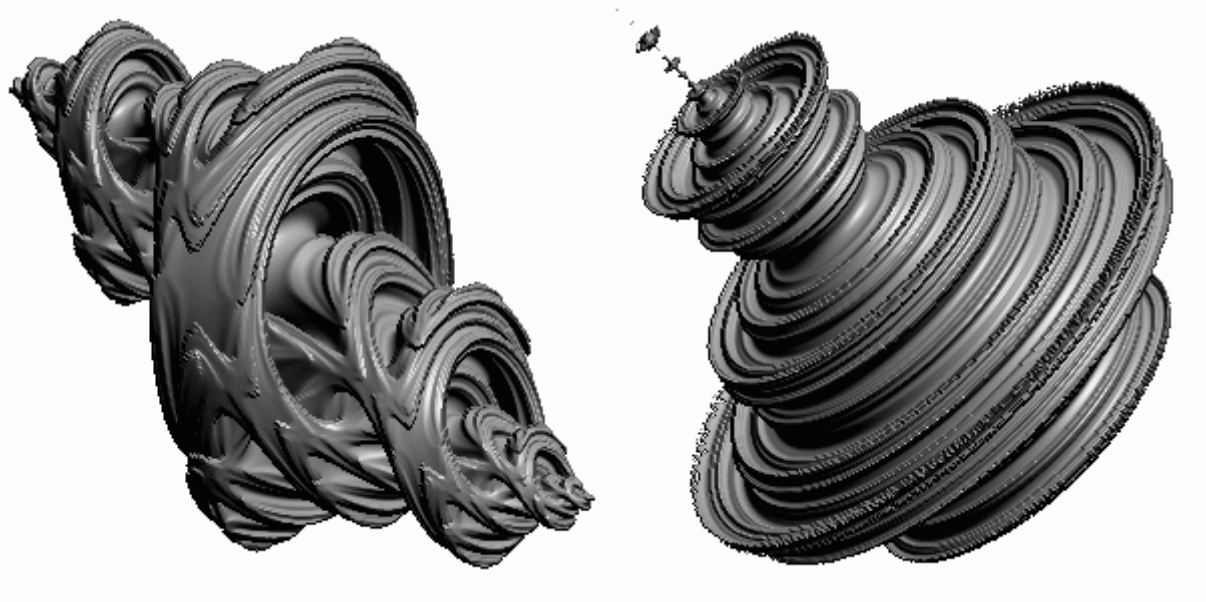

Рис.1. Гиперкомплексные множества Жюлиа и Мандельброта

2.5 Оценка расстояния (для комплексного множества Жюлиа и Мандельброта)

Подробное доказательство оценок расстояния для комплексных и гиперкомплексных множеств Жюлиа и Мандельброта приведено в [3].

По существу, на комплексной плоскости формулы оценки расстояния для множеств Жюлиа и Мандельброта одинаковы. Поэтому будем рассматривать только множества Жюлиа.

Пусть $\mathbf{c \in M}$ (комплексное множество Мандельброта), тогда $\mathbf{K_c}$ (наполненное комплексное множество Жюлиа для отображения $\mathbf{f_c(z) = z^2 + c}$) связно. Пусть $\mathbf{D = \{z \in C:\ |z| \le 1\}}$ – диск единичного радиуса с центром в начале координат на комплексной плоскости. В этом случае существует взаимнооднозначное биголоморфное отображение

$$\varphi_c: C \setminus K_c \to C \setminus D,$$

для которого $\varphi_c(z) \to \infty$, когда $z \to \infty$.

$$\begin{array}{ccc} C \setminus Kc & \xrightarrow{\varphi_c} & C \setminus D \\ f_c \downarrow & & \downarrow f_0 \\ C \setminus Kc & \xrightarrow[\varphi_c]{} & C \setminus D \end{array}$$

Другими словами, $f_0(z) = \varphi_c(f_c(\varphi_c^{-1}(z)))$, то есть итерация f_0 вне D эквивалентна итерации f_c вне K_c.

Определим функцию потенциала (потенциал) для K_c:

$$G(z) = \ln(|\varphi_c(z)|)$$

и функцию времени убегания:

$$\varepsilon = -\ln(G(z)).$$

Существенно, что K_c связно, иначе φ_c не существовало бы.

Пусть $d(z, K_c)$ – расстояние от точки $c \in C$ до множества Жюлиа K_c. Тогда расстояние $d(z, K_c)$ между точкой z, лежащей вне K_c, и K_c удовлетворяет

$$\frac{\sinh(G(z))}{2e^{G(z)}\,|G'(z)|} < d(z, K_c) < \frac{2\sinh(G(z))}{|G'(z)|},$$

где $G(z)$ – потенциал в точке z.

Приближение этого неравенства может быть записано как

$$\frac{|z_n|}{2|z_n|^{\frac{1}{2^n}}|z'_n|}\ln(|z_n|) < d(z, K_c) < \frac{|z_n|}{|z'_n|}\ln(z_n)$$

и имеет большое практическое значение для построения множеств Жюлиа. Оно также обобщается на кватернионы.

2.6 Оценка расстояния (для гиперкомплексного множества Мандельброта)

Пусть M_4 обозначает гиперкомплексное множество Мандельброта, M_2 – комплексное. Пусть $q \in H$, тогда существует $p \in H$, $|p| = 1$ такое, что $p^{-1}qp = c \in C$ (см. пункт 2.2). Назовем c *соответствующим комплексным значением* для q.

Для любого $q \in H$ и его соответствующего комплексного значения $c \in C$,

$q \in M_4$, если и только если $c \in M_2$.

Для любого $\mathbf{q} \in \mathbf{H}$ и любого $\mathbf{p} \in \mathbf{H}$, $|\mathbf{p}| = 1$,

$\mathbf{q} \in \mathbf{M_4}$, если и только если $\mathbf{pqp^{-1}} \in \mathbf{M_4}$.

Для любого $\mathbf{c} \in \mathbf{C}$, расстояние от $\mathbf{c}$ до $\mathbf{M_4}$ равно расстоянию от $\mathbf{c}$ до $\mathbf{M_2}$:

$\mathbf{d(c, M_4) = d(c, M_2)}$.

Для любого $\mathbf{q} \in \mathbf{H}$ и его соответствующего комплексного значения $\mathbf{c} \in \mathbf{C}$,

$\mathbf{d(q, M_4) = d(c, M_2)}$.

Расстояние от точки $\mathbf{q}$, лежащей вне гиперкомплексного множества Мандельброта $\mathbf{M_4}$, до $\mathbf{M_4}$ ограничено

$$\frac{|Z_n|}{2|Z_n|^{\frac{1}{2^n}}|Z'_n|}\ln(|Z_n|) < d(q,M_4) < \frac{|Z_n|}{|Z'_n|}\ln(|Z_n|),$$

где $\mathbf{Z_n = F^n(Z_0)}$, $\mathbf{Z_0 = 0}$, $\mathbf{F(Z) = Z^2 + q}$ ($\mathbf{F: H \to H}$).

2.7 Оценка расстояния (для гиперкомплексного множества Жюлиа)

Расстояние $\mathbf{d(h_0, K_q)}$ между точкой $\mathbf{h_0}$, лежащей вне $\mathbf{K_q}$, и $\mathbf{K_q}$ удовлетворяет следующему неравенству

$$\mathbf{d(h_0, K_q) > \alpha\, |h_n| / D(|h_n|)},$$

где $\mathbf{h_0} \in \mathbf{H}$, $\mathbf{D(|h_n|) = 2|h_{n-1}|D(|h_{n-1}|)}$, $\mathbf{K_q}$ – наполненное множество Жюлиа для $\mathbf{f_q(h) = h^2 + q}$ ($\mathbf{h}$, $\mathbf{q} \in \mathbf{H}$), $\boldsymbol{\alpha}$ – положительная вещественная константа.

2.8 Притяжение/отталкивание множества Жюлиа

Множество Жюлиа ($\mathbf{J = J(f_c)}$) – отталкивающее по отношению к $\mathbf{f_c}$. Это значит, что орбиты точек, находящихся около $\mathbf{J}$, уходят далеко от него. И, наоборот, множество $\mathbf{J}$ – притягивающее по отношению к обратной функции $\mathbf{f_c^{-1}(q)} = \sqrt{q - c}$ (орбиты точек, находящихся далеко от $\mathbf{J}$, приближаются к нему), поэтому погрешность вычислений уменьшается при последовательных итерациях $\mathbf{f_c^{-1}}$.

2.9 j-k эквивалентность

Динамика $\mathbf{F(h) = c_1 h^2 + c_2}$ ($\mathbf{c_1}$, $\mathbf{c_2} \in \mathbf{C}$) не зависит от угла $\boldsymbol{\theta}$ в

$$\mathbf{h = z_1 + e^{i\theta} z_2 j},$$

где $\mathbf{z_1}$, $\mathbf{z_2} \in \mathbf{C}$.

В частности, пусть $\mathbf{g_\theta(z_1 + z_2 j) = z_1 + e^{i\theta} z_2 j}$, тогда $\mathbf{g_\theta(F(g_{-\theta}(h))) = F(h)}$.

Можно рассматривать функцию $\mathbf{g_\theta(h)}$ как вращение 4-х мерного пространства, которое фиксирует комплексную плоскость. Видно, что 4-х мерное множество Жюлиа получается с помощью вращения 3-х мерного множества Жюлиа вокруг комплексной плоскости.

Это свойство 4-х мерного множества Жюлиа называется *j-k эквивалентностью*.

2.10 Множества Жюлиа квадратичного семейства

Классическое гиперкомплексное множество Жюлиа определяется функцией

$$\mathbf{f_q(z) = z^2 + q},$$

где $\mathbf{q} \in \mathbf{H}$. Рассмотрим случай $\mathbf{q = c} \in \mathbf{C}$. Поэтому

$$\mathbf{f_c(z) = z^2 + c}$$

расширение комплексной функции.

Так как комплексная плоскость – это подмножество гиперкомплексного пространства, то любое комплексное множество Жюлиа существует в гиперкомплексном пространстве и часто имеет продолжение вне комплексной плоскости.

Когда $\mathbf{c} \in \mathbf{R}$, гиперкомплексное множество Жюлиа – всего лишь поверхность вращения в $\mathbf{H}$ 2-х мерного множества Жюлиа вокруг вещественной оси.

Если $\mathbf{c}$ имеет мнимые компоненты, то продолжение не тривиальное. Гиперкомплексное множество Жюлиа содержит больше информации, чем его комплексное подмножество.

Множества Жюлиа для $\mathbf{f_c(z) = z^2 + c}$ могут быть визуализированы в 3-х мерном пространстве с помощью нахождения пересечения 4-х мерного объекта с 3-х мерным пространством, натянутом на векторы $\mathbf{1}$, $\mathbf{i}$, $\mathbf{j}$ (из-за j-k эквивалентности).

Существуют соответствующие $\mathbf{q}$ ($\mathbf{q} \in \mathbf{H}$), единичный кватернион $\mathbf{p}$ ($|\mathbf{p}| = \mathbf{1}$) и комплексное число $\mathbf{c}$, такие, что $\mathbf{pqp^{-1}} = \mathbf{c}$. Тогда множество Жюлиа $\mathbf{J_q}$ для $\mathbf{f_q(z)} = \mathbf{z^2} + \mathbf{q}$ ($\mathbf{q} \in \mathbf{H}$) может быть получено с помощью поворота в гиперкомплексном пространстве множества Жюлиа $\mathbf{J_c}$ для $\mathbf{f_c(z)} = \mathbf{z^2} + \mathbf{c}$ ($\mathbf{c} \in \mathbf{C}$).

3 Алгоритмы построения гиперкомплексных множеств

3.1 Метод обратных итераций

Простейшим методом построения множеств Жюлиа является метод обратных итераций. Он использует тот факт, что множество Жюлиа является притягивающим для обратной функции. Этот метод не требует больших вычислений и использует малое количество памяти, но не позволяет получать мелких деталей. Изображения, полученные с помощью этого метода – облака из точек, плохо отображают пространственную структуру множества.

Пусть $\mathbf{n}$ – количество генерируемых точек.

Выбирается произвольная начальная точка. После $\mathbf{m}$ итераций обратной функции точка становится очень близкой к множеству $\mathbf{J}$. Если выбрать начальную точку принадлежащей $\mathbf{J}$, то $\mathbf{m}$ можно принять равной $\mathbf{0}$. Каждая последующая итерация точки выводится на экран.

Гиперкомплексный метод обратных итераций отличается от классического не только использованием гиперкомплексных вычислений, но и учетом некоторых свойств $\mathbf{H}$ (например, особенности вычисления корней) и гиперкомплексных множеств Жюлиа (например, j-k эквивалентность).

Подробное описание метода обратных итераций можно найти в [3] и [4].

3.2 Метод трассировки лучей

Этот метод требует намного больше вычислений, но позволяет лучше показывать структуру множества, увеличивать фрагменты изображения. Он позволяет строить не только множества Жюлиа, но и множество Мандельброта.

Главная часть этого метода – это получение точки пересечения луча с поверхностью (границей) множества и определение нормали к поверхности в этой точке.

Поскольку границы множества Мандельброта и почти всех множеств Жюлиа недифференцируемы, то нормали к этим границам не определены. Мы приводим алгоритм вычисления нормалей к приближенным границам.

Луч задается начальной точкой $\mathbf{q_0}$ и направлением **dir** ($\mathbf{q_0}$, **dir** $\in$ **H**, |**dir**| = **1**, $\mathbf{q_0}$ не принадлежит $\mathbf{K(f_c)}$ (или **M**, если строится множество Мандельброта)). Точка **h** на луче, находящаяся на расстоянии **dist** от начала луча, определяется по следующей формуле: $\mathbf{h = q_0 + dir \cdot dist}$ (см. рис. 2).

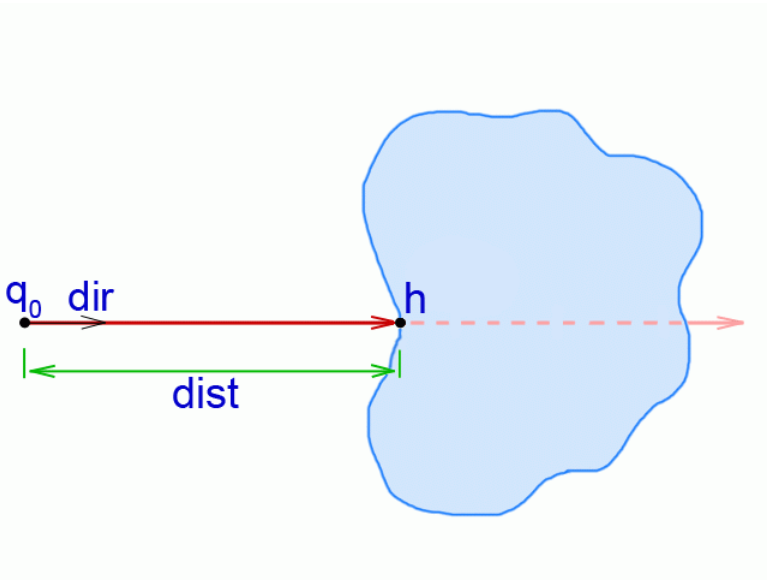

Рис.2. Пересечение луча с границей множества

3.3 Первый способ (классический) нахождения расстояния от начала луча до точки пересечения.

Дополнительно задаются максимальное расстояние **maxD**, дальше которого пересечение не ищется, и количество разбиений **n** (можно интерпретировать как точность). Определяется шаг по расстоянию $\Delta\mathbf{D} = \mathbf{maxD} / \mathbf{n}$. С этим шагом мы "идем" по лучу и проверяем принадлежность точки на луче (на шаге **s**: $\mathbf{h_s} = \mathbf{q_0} + \mathbf{dir}\cdot\Delta\mathbf{D}\cdot\mathbf{s}$) наполненному множеству Жюлиа.

Если на шаге **s** точка $\mathbf{h_s}$ принадлежит $\mathbf{K(f_c)}$ (алгоритмы определения принадлежности точки множеству Мандельброта и наполненному множеству Жюлиа приведены в Приложении), то считается, что расстояние от начала луча до точки пересечения равно $\mathbf{dist} = \Delta\mathbf{D}\cdot(\mathbf{s} - \mathbf{1})$ (см. рис. 3). Если после **n** шагов условие принадлежности не выполняется, то считается, что луч не пересек множество, и можно положить **dist** равным **maxD**. Подобный алгоритм нахождения расстояния используется в [6].

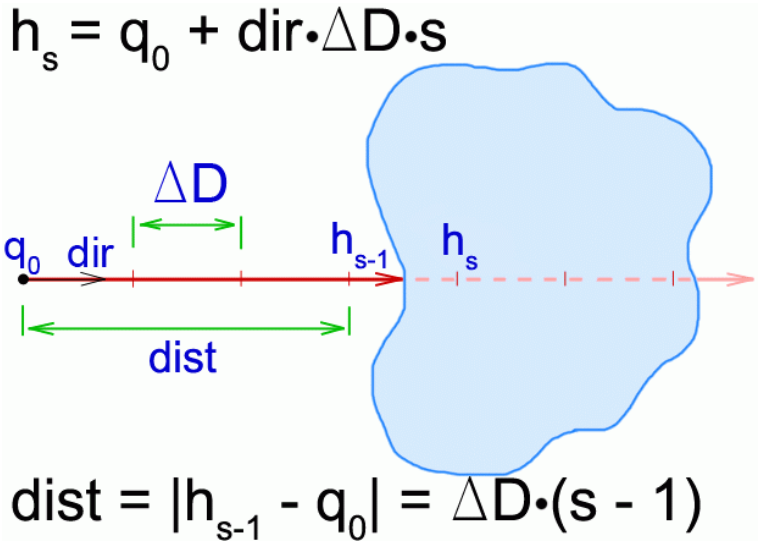

Рис.3. Классический способ нахождения расстояния

Недостатки данного способа нахождения пересечения:

1) с возрастанием точности (количества разбиений **n**) пропорционально возрастает время поиска пересечения;

2) даже если луч пересекает $\mathbf{K(f_c)}$, и расстояние от начала луча до точки пересечения меньше **maxD**, пересечение может быть не найдено (такое случается, когда длина отрезка луча внутри $\mathbf{K(f_c)}$ меньше $\Delta\mathbf{D}$; см. рис.4);
3) может быть найдено не первое пересечение луча с $\mathbf{K(f_c)}$ (та же причина).

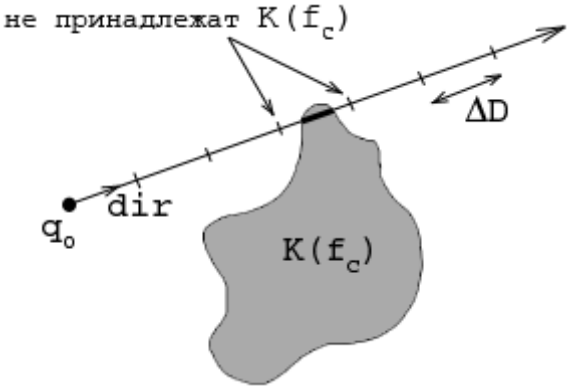

Рис.4. Пересечение не найдено

3.4 Усовершенствование первого способа

Для увеличения точности вычисления расстояния можно рекурсивно вызвать процедуру нахождения расстояния от начала луча до точки пересечения с $\mathbf{K(f_c)}$. Выбирается луч с направлением $\mathbf{dir'} = \mathbf{dir}$ и началом $\mathbf{q_0'} = \mathbf{q_0} + \mathbf{dir}\cdot\Delta\mathbf{D}\cdot(\mathbf{s} - \mathbf{1})$, $\mathbf{maxD'} = \Delta\mathbf{D}$, $\mathbf{n'}$ можно выбрать и не равным **n**, но ≥ 2. Если для этого луча получилось расстояние **dist'**, то для начального луча $\mathbf{dist} = \Delta\mathbf{D}\cdot(\mathbf{s} - \mathbf{1}) + \mathbf{dist'}$ (см. рис. 5).

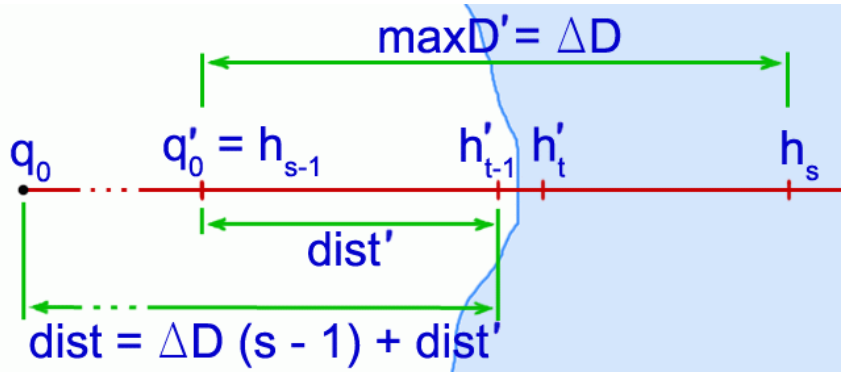

Рис.5. Усовершенствование классического способа

3.5 Второй способ

Пусть есть функции $\mathbf{d_{min}}$ и $\mathbf{d_{max}}$, которые для каждой точки **q** выдают минимальную и максимальную оценку расстояния от **q** до множества $\mathbf{K(f_c)}$, (см. пункты 2.5 – 2.7). Примечание: на расстояние $\mathbf{d_{min}(q)}$, от **q** (**q** не принадлежит $\mathbf{K(f_c)}$) нет ни одной точки из $\mathbf{K(f_c)}$.

1. Выбирается точка $\mathbf{q = q_0}$ ($\mathbf{q_0}$ не принадлежит множеству).
2. Вычисляется $\mathbf{dMin = d_{min}(q)}$ и $\mathbf{dMax = d_{max}(q)}$.
3. $\mathbf{q = q + dir \cdot dMin}$ (**q** по-прежнему не принадлежит множеству).
4. Если $\mathbf{|dMax - dMin| < \varepsilon}$ (заданная погрешность), то точка пересечения **q** найдена (точнее **q** – это точка, находящаяся в **ε**-окрестности границы множества). Расстояние от начала луча до точки пересечения $\mathbf{dist = |q_0 - q|}$. Выход.
5. Если $\mathbf{|q_0 - q| > maxD}$ (см. предыдущий алгоритм) то пересечения нет. Положим **dist = maxD**. Выход.
6. Переход к 2.

Есть еще один вариант этого алгоритма, в котором не используется функция $\mathbf{d_{max}}$. В этом варианте в пункте 2 **dMax** не вычисляется, а проверка в пункте 4 заменяется на $\mathbf{dMin < \varepsilon}$. Такой алгоритм будет работать быстрее, но с меньшей точностью определять расстояние. В качестве **dMin** удобно брать оценку из пункта 2.7. В этом случае **dMin** вычисляется итеративно с помощью приведенных в этом пункте формул (параметр **n** выбирается, а затем константа **α** подбирается опытным путем)

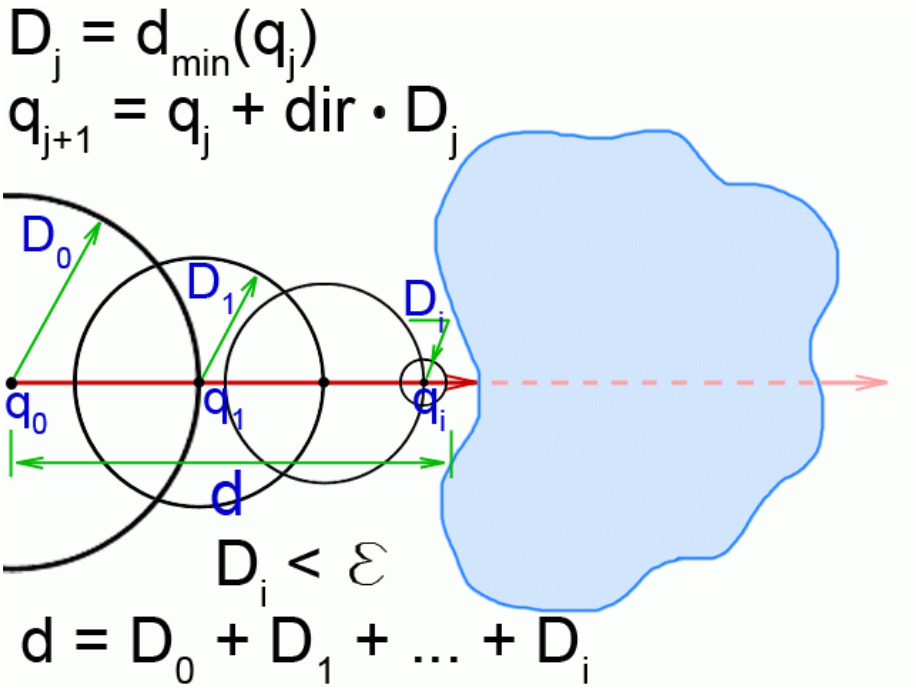

Рис.6. Способ, использующий оценку расстояния

3.6 Сравнение способов нахождения расстояния до точки пересечения

При одинаковой точности во втором способе (использующем оценку расстояния) за счет переменного шага по лучу пересечение находится за меньшее число шагов, чем в классическом способе, и примерно за такое же, как в усовершенствованном классическом. Также во втором способе отсутствуют недостатки 2) и 3) классического.

3.7 Переход от 4D к 2D, вычисление матрицы расстояний

Определим направление проектирования (вектором **zDir**), и проекционную плоскость **P** (точкой **center** на этой плоскости и двумя векторами **xDir**, **yDir**; **P** имеет размерность 2). Векторы **center**, **xDir**, **yDir**, **zDir** $\in$ **H**, причем **xDir**, **yDir**, **zDir** ортонормированные.

Точка на плоскости **P** задается следующим образом:

$$\mathbf{q} = \mathbf{center} + \mathbf{x}\cdot\mathbf{xDir} + \mathbf{y}\cdot\mathbf{yDir},$$

где **x**, **y** $\in$ **R** – координаты этой точки в базисе (**xDir**, **yDir**).

Также определим минимальное и максимальное расстояния до проекционной плоскости **P** (**minDist** и **maxDist** соответственно) и стороны прямоугольной области проектирования (**dx** и **dy**) с центром **center** на этой плоскости. **minDist**, **maxDist**, **dx**, **dy** $\in$ **R**, **minDist** < **maxDist**, **dx** > **0**, **dy** > **0** (**minDist** и **maxDist** могут быть меньше 0). Проектирование будет производиться только на эту прямоугольную часть плоскости **P**. Проектироваться будет только та часть множества, которая находится между плоскостями $\mathbf{P}_{\mathbf{min}}$ и $\mathbf{P}_{\mathbf{max}}$, где $\mathbf{P}_{\mathbf{min}}$ и $\mathbf{P}_{\mathbf{max}}$ – смещения плоскости **P** на векторы **minDist**·**zDir** и **maxDist**·**zDir** соответственно. Эти параметры определяют область **A** в **H**:

$$\mathbf{A = \{h \in H \mid h = center + x \cdot xDir + y \cdot yDir + z \cdot zDir}, \text{ где}$$

$$\mathbf{-dx/2 \le x \le dx/2, -dy/2 \le y \le dy/2, minDist \le z \le maxDist\}}.$$

Таким образом, будет отображаться только та часть множества, которая попала в область **A**.

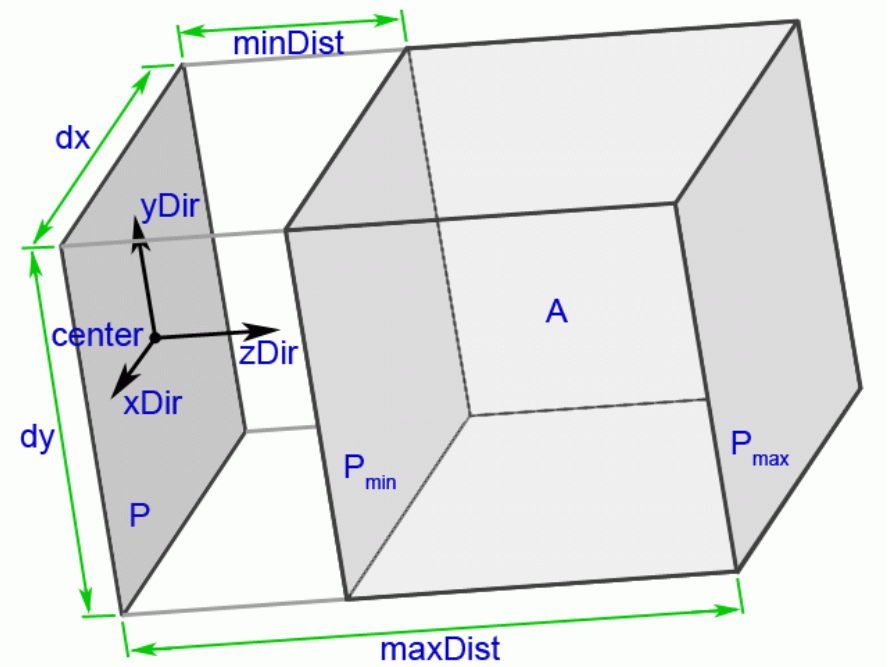

Рис.7.

Пусть надо получить изображение с разрешением **xRes** × **yRes**. Вычислим матрицу расстояний **m** (размерностью **xRes** × **yRes**). Для элемента **m[a, b]** вычислим

$$\mathbf{q_{ab}^{\,0} = center + dx \cdot (a/xRes - 0.5) \cdot xDir + dy \cdot (b/yRes - 0.5) \cdot yDir + minDist \cdot zDir}.$$

Для луча с началом $\mathbf{q_{ab}}^{0}$ и направлением **zDir** найдем расстояние $\mathbf{d_{ab}}$ от $\mathbf{q_{ab}}^{0}$ до точки пересечения с $\mathbf{K(f_c)}$ (любым методом; **maxD = maxDist – minDist**). Занесем это значение в матрицу расстояний (**m[a, b] = d_{ab}**).

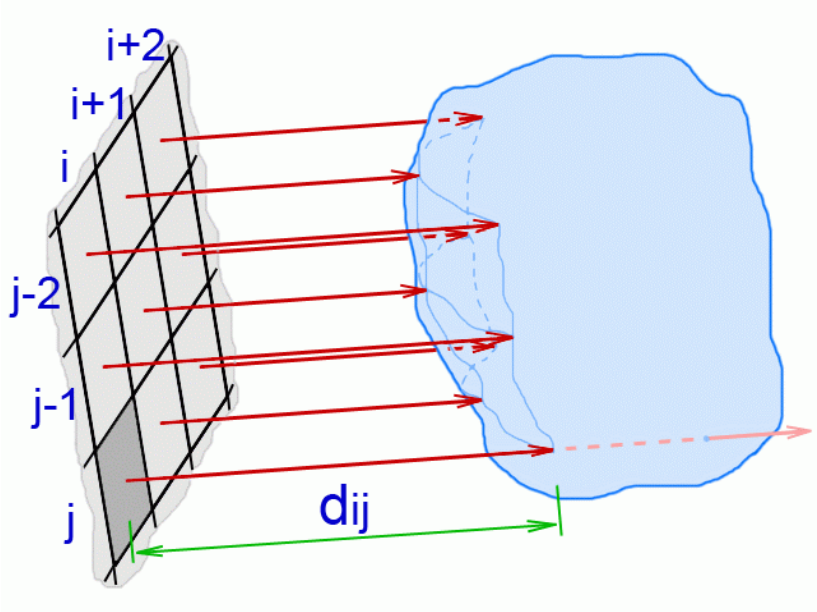

Рис.8.

3.8 Вычисление нормалей

Как уже упоминалось ранее, мы будем вычислять нормали к приближенной границе (поверхности) множества. Приближенная поверхность – “гладкая” поверхность, проходящая через точки $\mathbf{q_{ab}}$ = **center + dx·(a/xRes – 0.5)·xDir + dy·(b/yRes – 0.5)·yDir + (minDist + m[a, b])·zDir**, принадлежащие окрестности реальной границы множества.

Вычислим трехмерную нормаль (в подпространстве, натянутом на векторы **xDir**, **yDir**, **zDir**) к этой поверхности в точке $\mathbf{q_{ab}}$. Ненормированные координаты **(x, y, z)** этой нормали имеют вид:

$$\mathbf{x = (m[a + 1, b] – m[a – 1, b]) / \Delta x},$$

$$\mathbf{y = (m[a, b + 1] – m[a, b – 1]) / \Delta y},$$

$$\mathbf{z = 2},$$

где $\mathbf{\Delta x = dx / xRes}$, $\mathbf{\Delta y = dy / yRes}$.

После этого вектор **(x, y, z)** нужно нормировать.

3.9 Получение изображения

Пусть задан источник света, то есть задано направление **lightDir** на него (трехмерный нормированный вектор) и его цвет **(red, green, blue)**. Пусть $\mathbf{normal_{ab}}$ – нормаль, соответствующая паре **(a, b)** (то есть нормаль к поверхности в точке $\mathbf{q_{ab}}$). Пусть $\mathbf{brightness_{ab} = lightDir \cdot normal_{ab}}$ (скалярное произведение векторов **lightDir** и $\mathbf{normal_{ab}}$, равняется косинусу угла между этими векторами, так как их модули равны 1).

Если $\mathbf{brightness_{ab} \le 0}$, то пиксель **(a, b)** красится черным, иначе цветом $\mathbf{(brightness_{ab} \cdot red,\ brightness_{ab} \cdot green,\ brightness_{ab} \cdot blue)}$. Можно использовать несколько источников цвета. В этом случае цвета от каждого источника суммируются.

4 Программа “Quaternions”

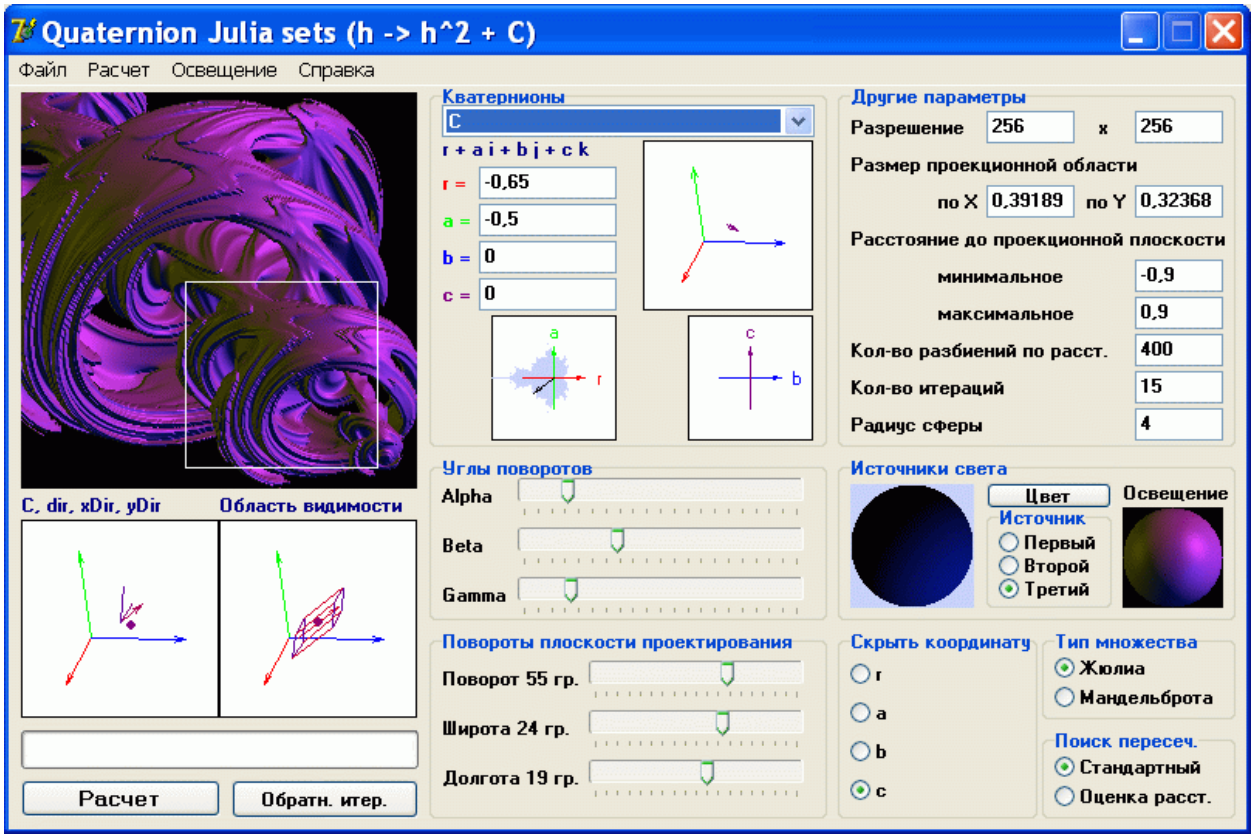

Рис.9. Программа “Quaternions”

Возможности построения

- множеств Жюлиа для отображения $\mathbf{h_{n+1} = h_n^2 + c}$ методом обратных итераций;
- множеств Мандельброта и Жюлиа для отображения $\mathbf{h_{n+1} = h_n^2 + c}$ с методом трассировки лучей (возможен выбор из 2 способов нахождения точки пересечения).

Возможности реализации

- несколько способов «навигации» по фракталам;
- сохранение и загрузка параметров фракталов (в файл / из файла специального формата);
- сохранение построенных фракталов как изображения (bmp);
- измерение скорости работы программы: общего времени построения (расчета) фрактала; среднего времени, затрачиваемого на вычисление одного пикселя изображения; среднего количества пикселей, вычисленных за одну секунду;
- освещение тремя цветными источниками света;
- зеркальное отражение поверхностью фрактала произвольной картинки.

5 Оценка скорости и качества работы алгоритмов

Скорость работы алгоритмов (трассировки лучей с использованием различных способов поиска пересечения луча с множеством) оценивалась с помощью программы “Quaternions”. Среднее количество пикселей, вычисленных за одну секунду, было выбрано в качестве параметра скорости работы. Все измерения были выполнены на одном компьютере (чтобы условия были максимально

похожими). Качество работы алгоритмов оценивалось визуально по получаемой картинке.

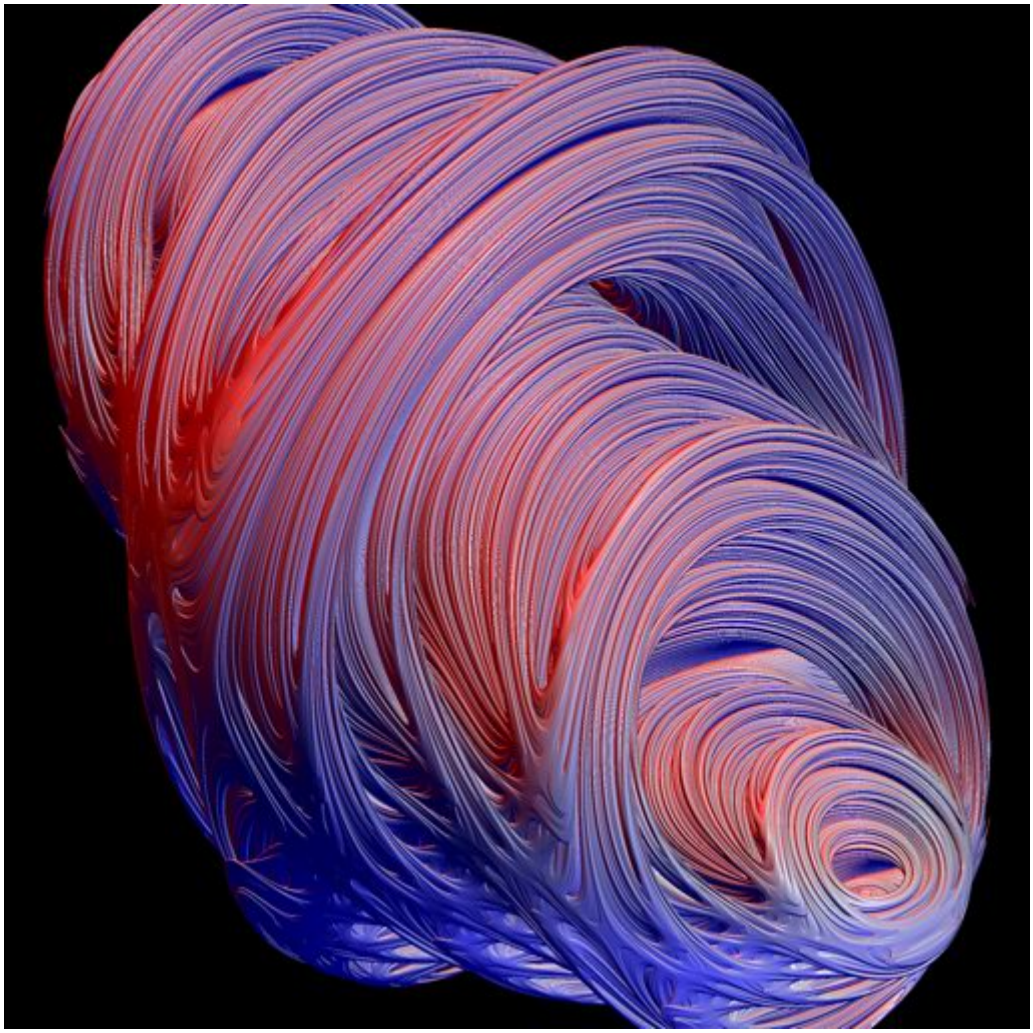

Рис.10.

Оценки скорости работы алгоритмов построения множества, показанного на рисунке 10, приведены

- на графике 1 (используется классический способ поиска пересечения луча с множеством; **maxIter = 20**, **r = 4**, см. пункт **A** Приложения);
- на графике 2 (усовершенствованный классический, два вложенных рекурсивных вызова с **n′ = 10**, см. пункт 3.4; **maxIter = 20**, **r = 4**);

- на графике 3 (оценка расстояний; скорость работы для различных значений параметра **maxIter**, значение константы **α** приведено на графике 4, см. пункт С Приложения).

Из графиков 1 и 2 видно, что алгоритмы, использующие классический (*первый алгоритм*) и усовершенствованный классический (*второй алгоритм*) способы, работают практически с одинаковой скоростью (при равном количестве разбиений второй алгоритм лишь немного медленнее). При увеличении точности (количества разбиений) в 2 раза скорость уменьшается в 2 раза, т.е. время работы этих алгоритмов линейно зависит от количества разбиений. График 3 показывает преимущество *третьего алгоритма* (использующего оценку расстояния): при увеличении точности в 2 раза скорость уменьшается лишь в 1.3 раза (для **maxIter** = 50 и **maxIter** = 80).

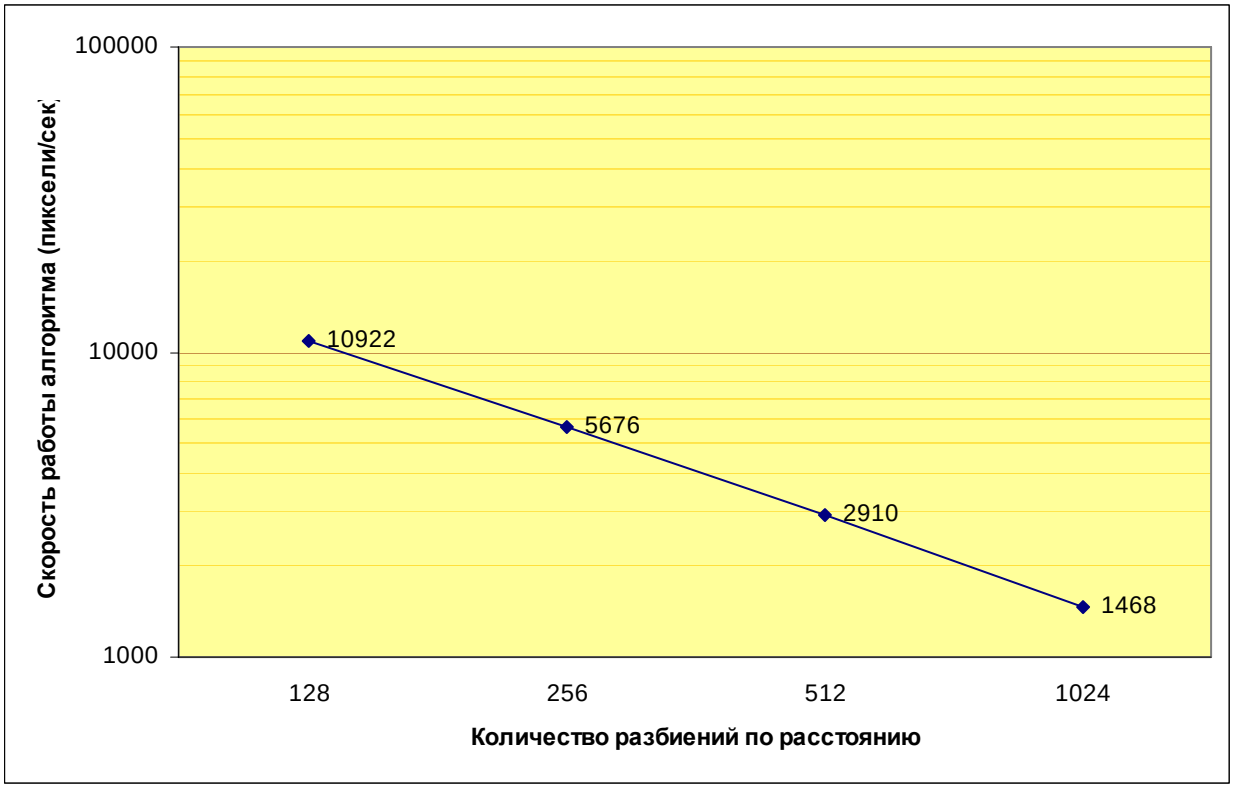

График 1.

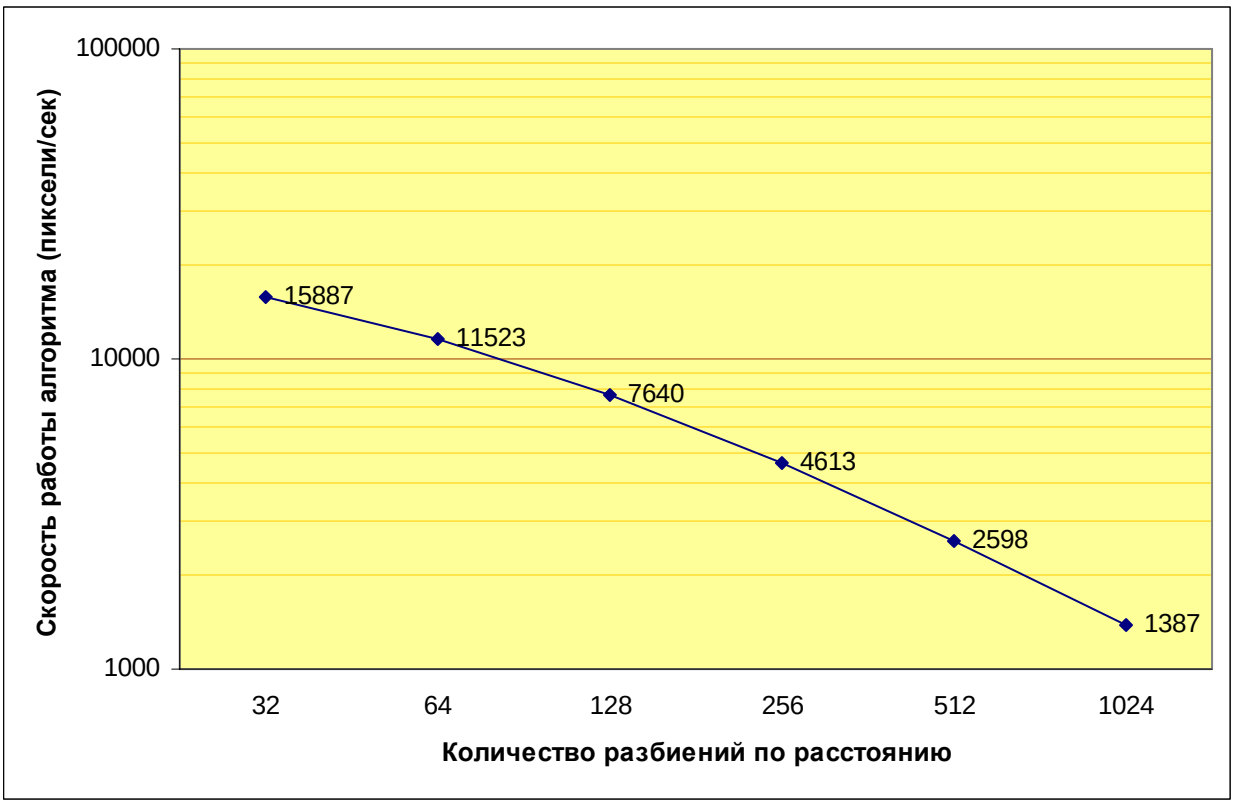

График 2.

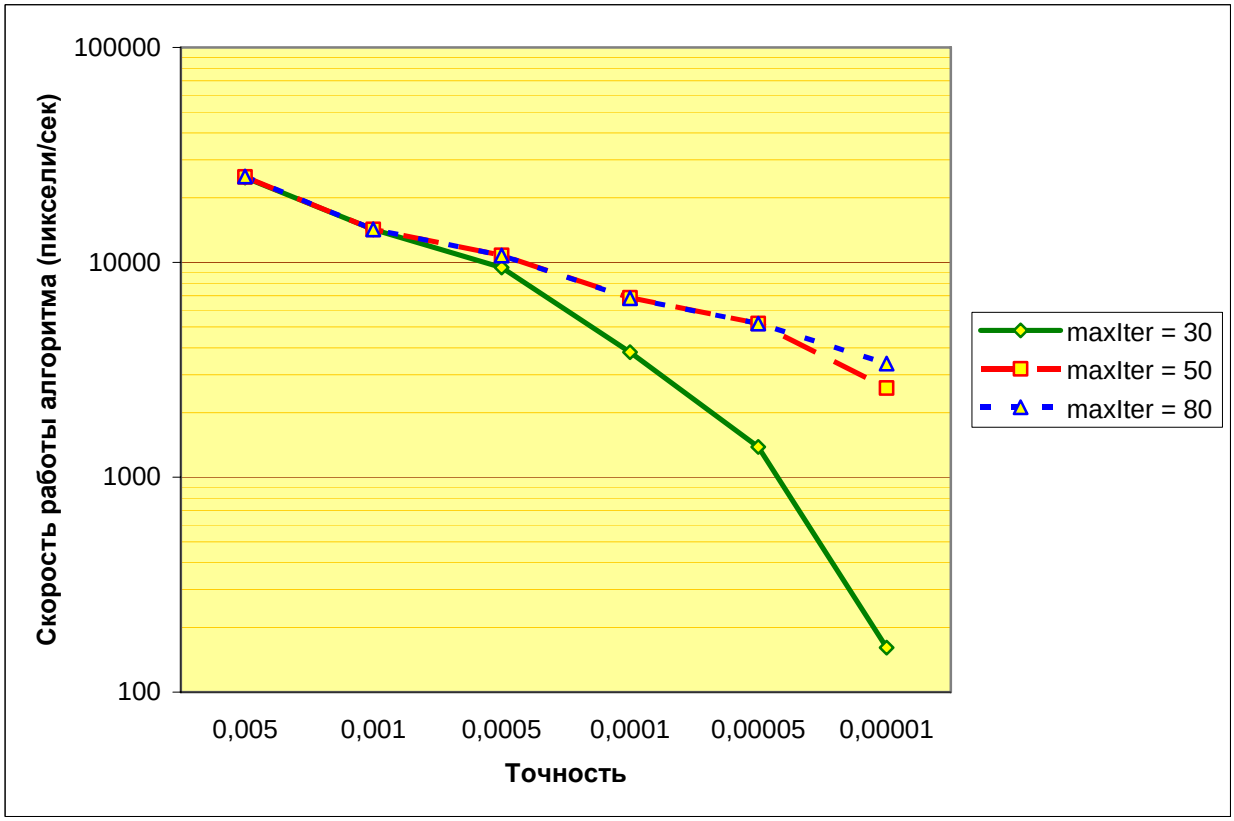

График 3.

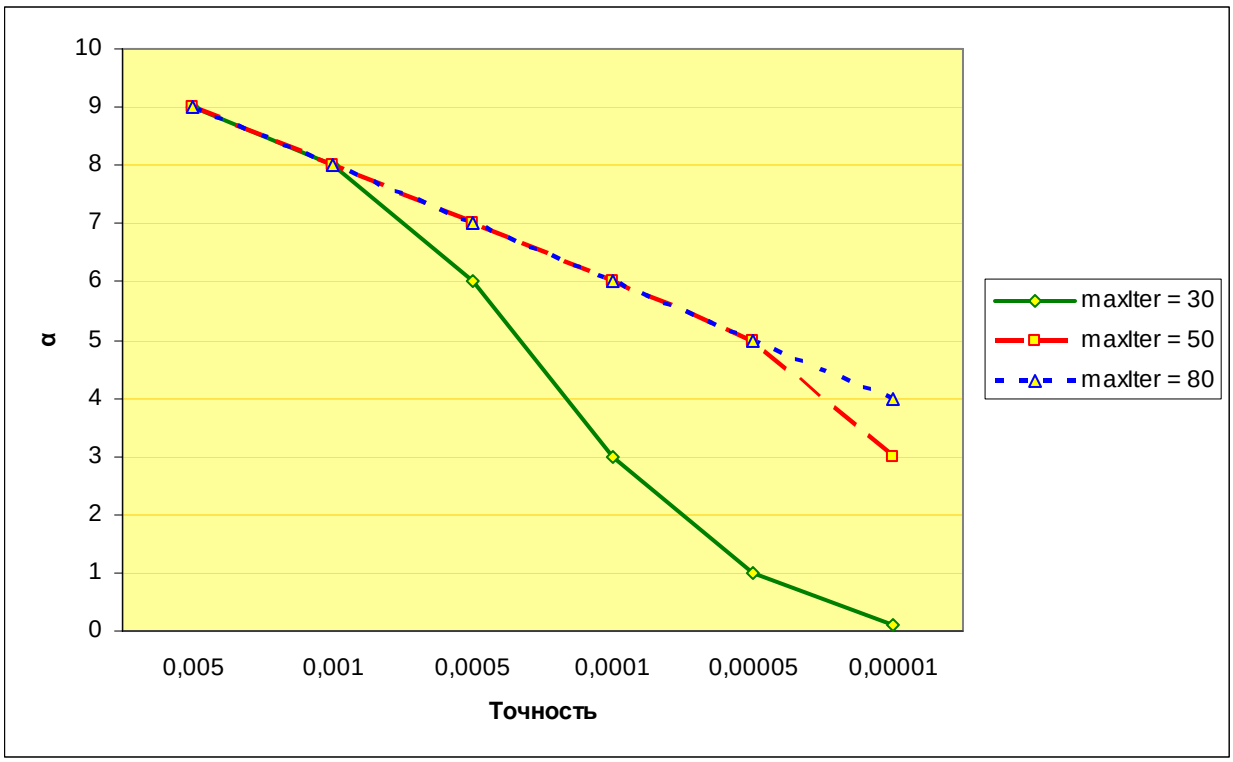

График 4.

Качество изображений оценивалась визуально по увеличенному фрагменту (см. рис. 11).

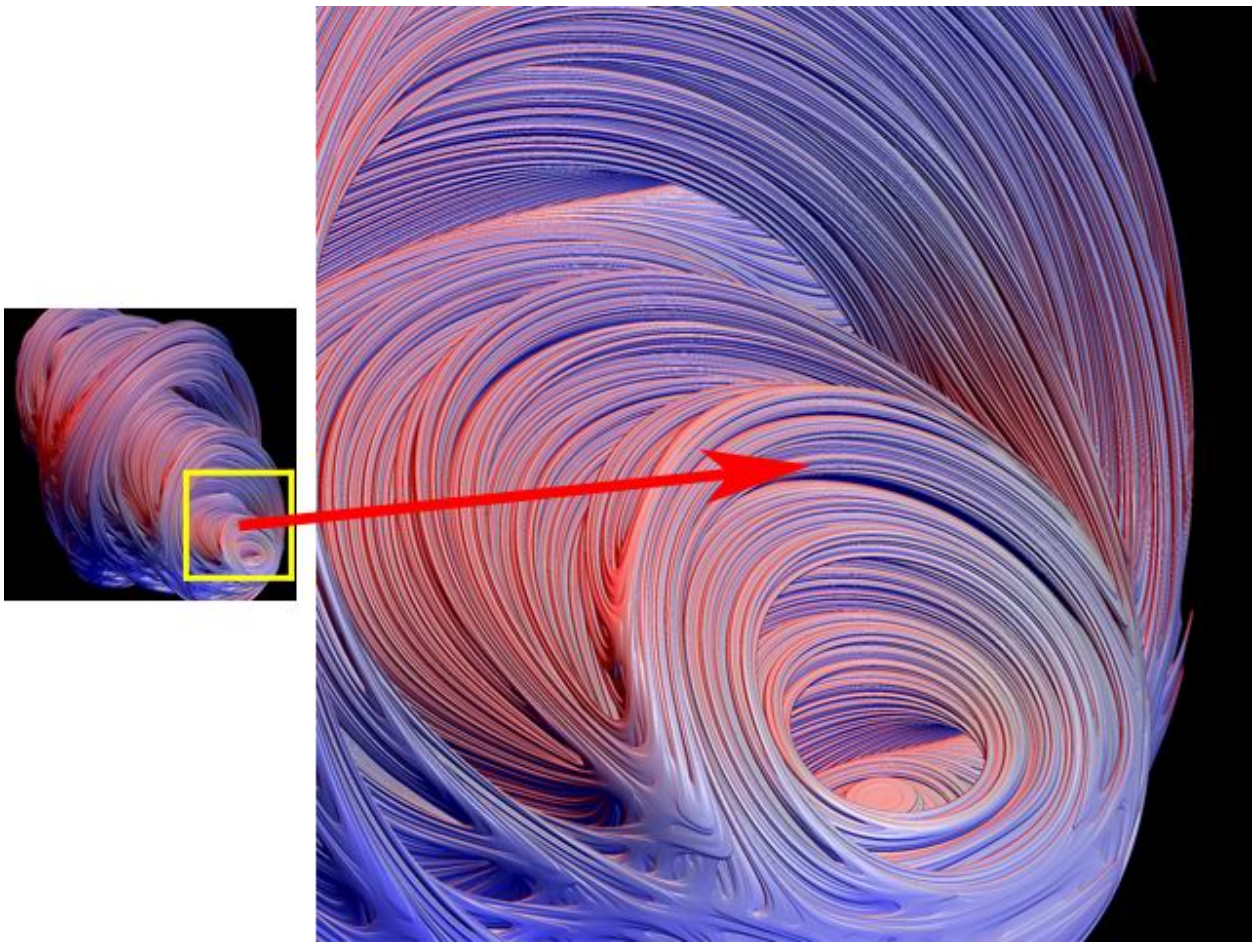

Рис.11.

На рисунке 12 показаны результаты работы первого алгоритма (количество разбиений по расстоянию 128, 256, 512, 1024). Видно, что достаточно хороший результат на данном фрагменте множества алгоритм дает при количестве разбиений 512 и больше (при меньшем количестве очень сильно проявляется “слоистость”).

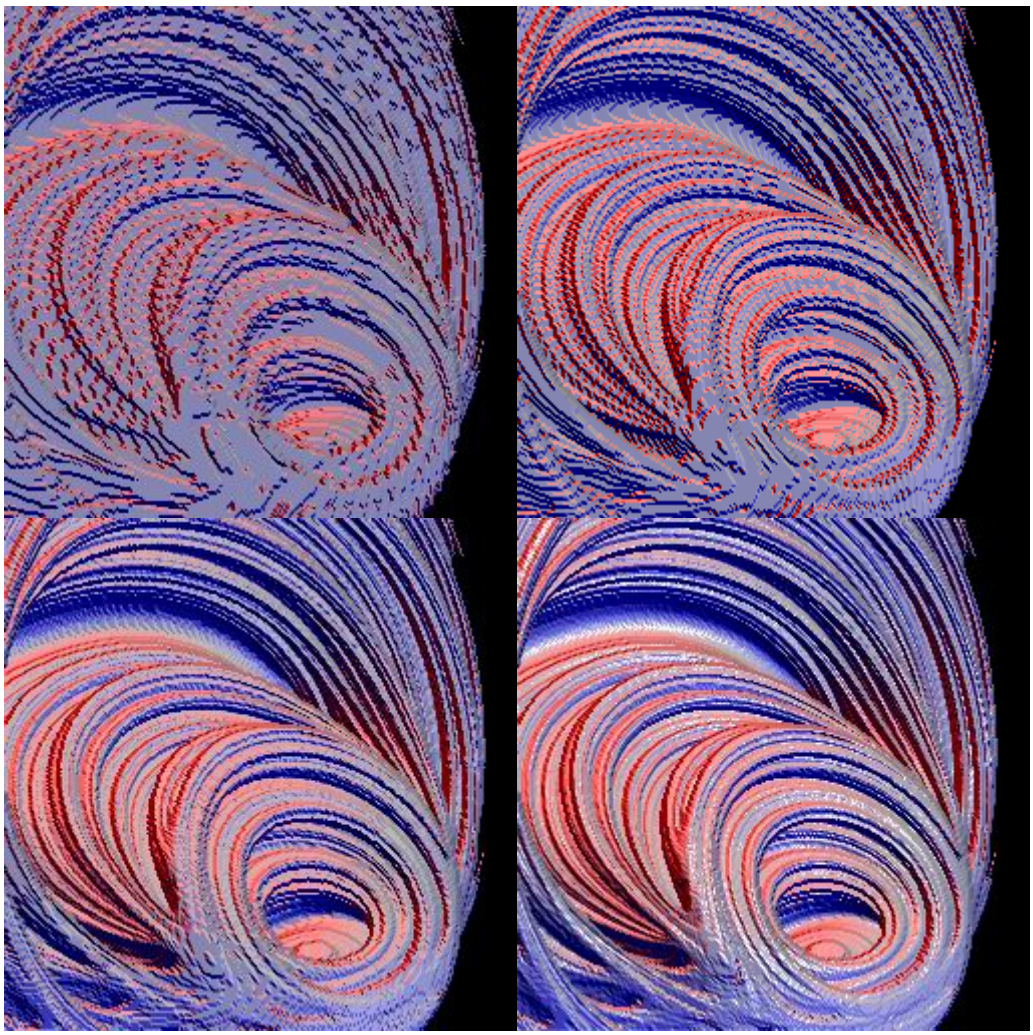

Рис.12.

Результаты для второго алгоритма приведены на рисунке 13 (количество разбиений по расстоянию 32, 64, 128, 256, 512, 1024). Поскольку в усовершенствованном способе используется два вложенных рекурсивных вызова с параметром $\mathbf{n'} = 10$, то реальное количество разбиений в 100 раз больше (10^2). За счет этого исчезает “слоистость”, присущая изображениям, полученным с помощью первого алгоритма. Но при малых значениях количества разбиений начинает проявляться эффект “зазубренности” границ. Хороший результат алгоритм дает при количестве разбиений 256 и больше.

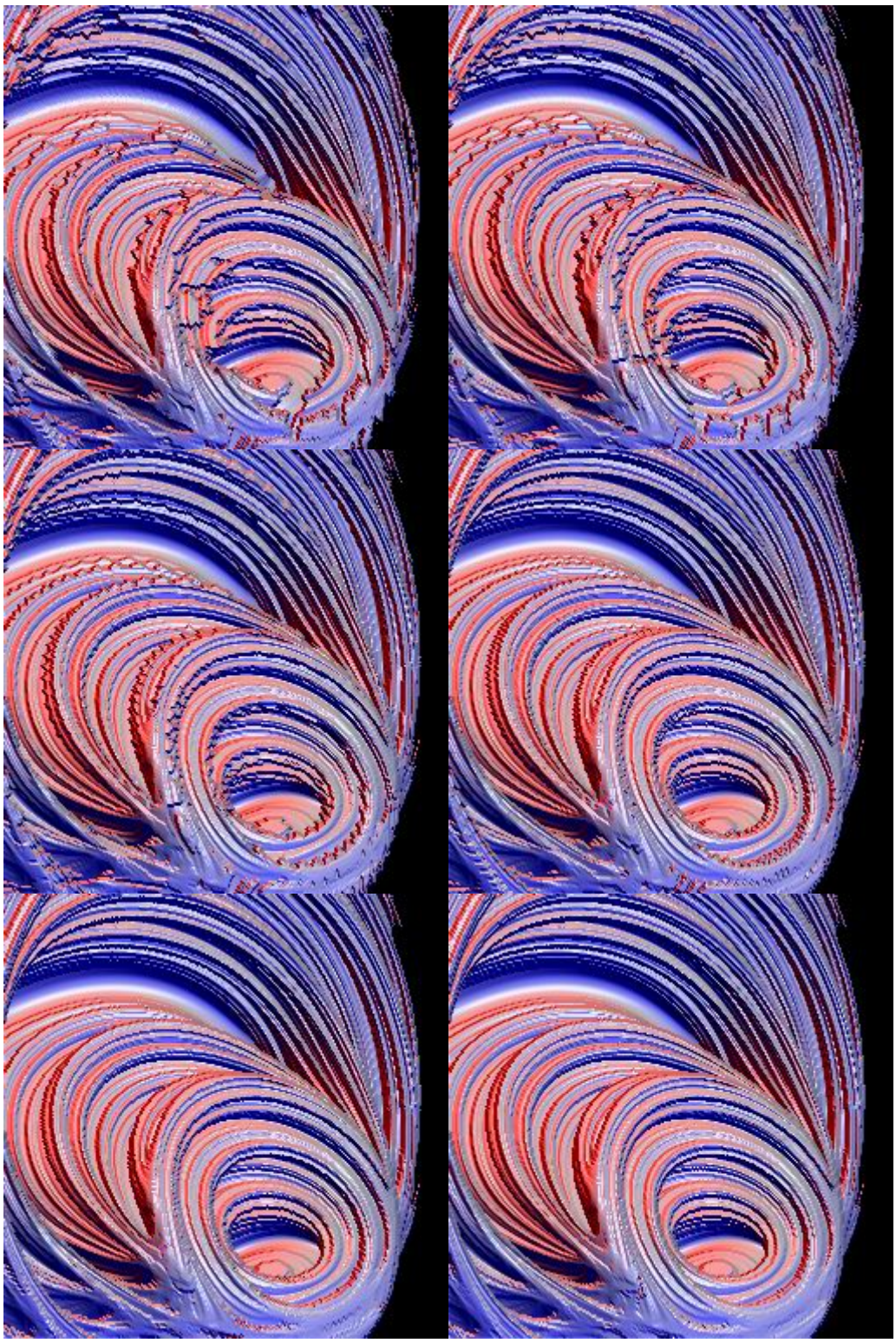

Рис.13.

На рисунке 14 показаны результаты работы третьего алгоритма, использующего для поиска пересечения луча с множеством оценку расстояния (точность 0.005, 0.001, 0.0005, 0.0001; **maxIter** = 80, качество изображения при других значениях **maxIter** такое же). При небольшой точности (0.005 и 0.001) множество выглядит

немного “сглаженным”. При точности 0.0005 результат хороший, отсутствуют недостатки изображений, построенных с помощью предыдущих алгоритмов. При точности 0.0001 результат наилучший и остается таким при дальнейшем увеличении точности (изменения в качестве изображения настолько незначительные, что это почти невозможно заметить). Первый алгоритм не позволяет добиться такого качества даже при 1024 разбиений по расстоянию, при этом построение данного изображения заняло почти в 5 раз больше времени. Второй алгоритм дает сравнимое качество картинки при 512 разбиений по расстоянию, а работает в 2.6 раза медленнее.

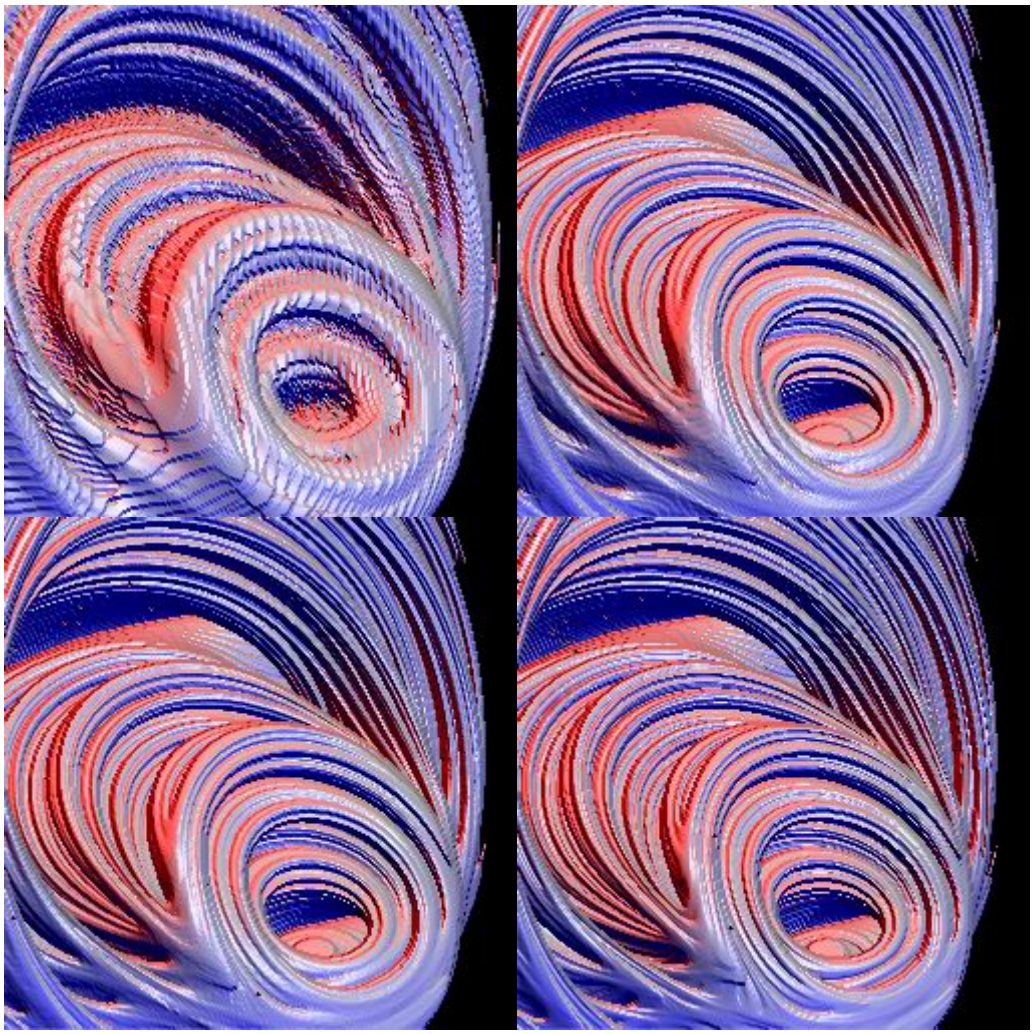

Рис.14.

Помимо исследования зависимости качества и скорости работы алгоритмов от точности были проверены зависимости и от некоторых других параметров.

Для второго алгоритма изучалась зависимость от параметров **maxIter** и **r** (результаты для первого алгоритма очень похожи). На графике 5 показана зависимость скорости работы от **maxIter** (количество разбиений по расстоянию 512, **r** = 4), на графике 6 – от **r (**количество разбиений по расстоянию 512, **maxIter** = 20**)**. На рисунке 15 приведен результат работы алгоритма для **maxIter** 9, 11, 17 и 50, на рисунке 16 – для **r** 0.82, 1, 1.2 и 1.5.

Чем больше **maxIter**, тем больше деталей можно увидеть на изображении. При **maxIter** 20 и больше качество картинки хорошее, видно достаточно мелкие детали. Также хорошее качество картинки достигается при **r** > 1.5.

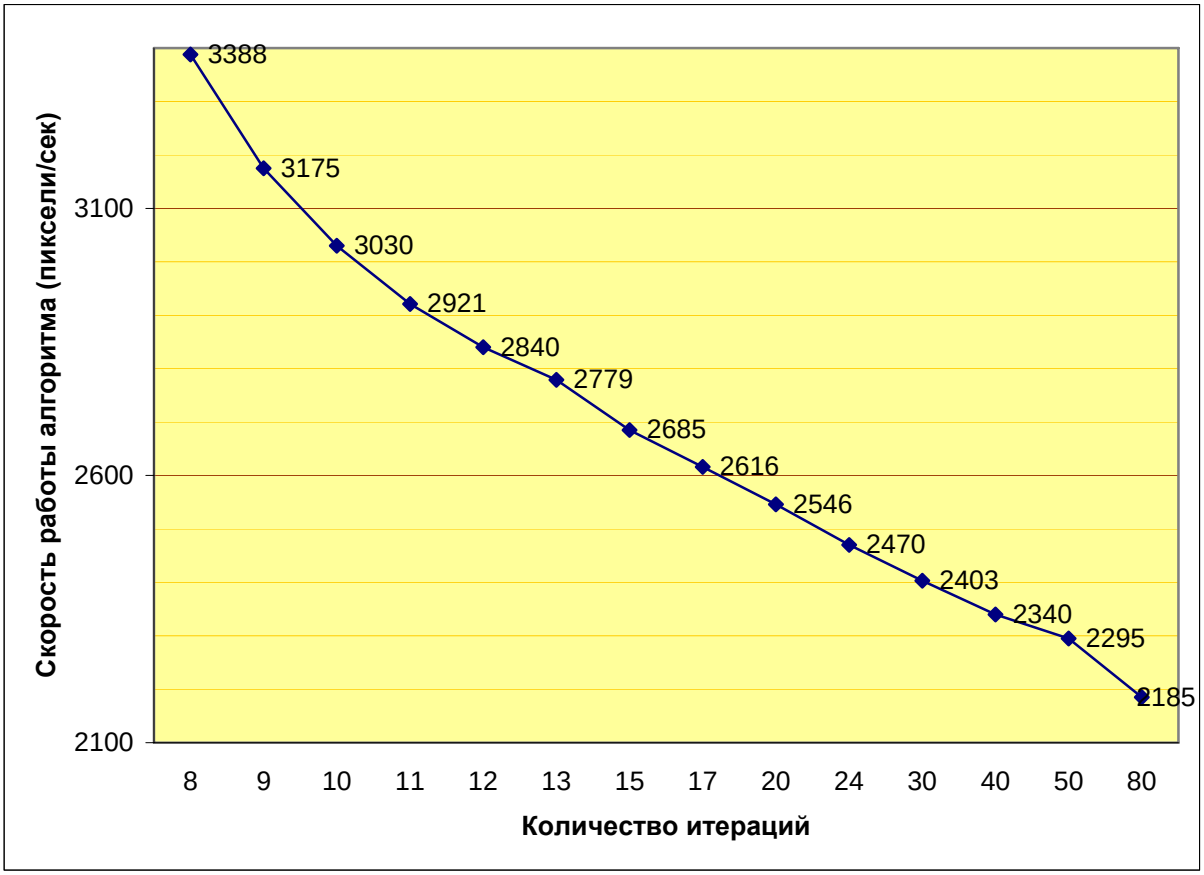

График 5.

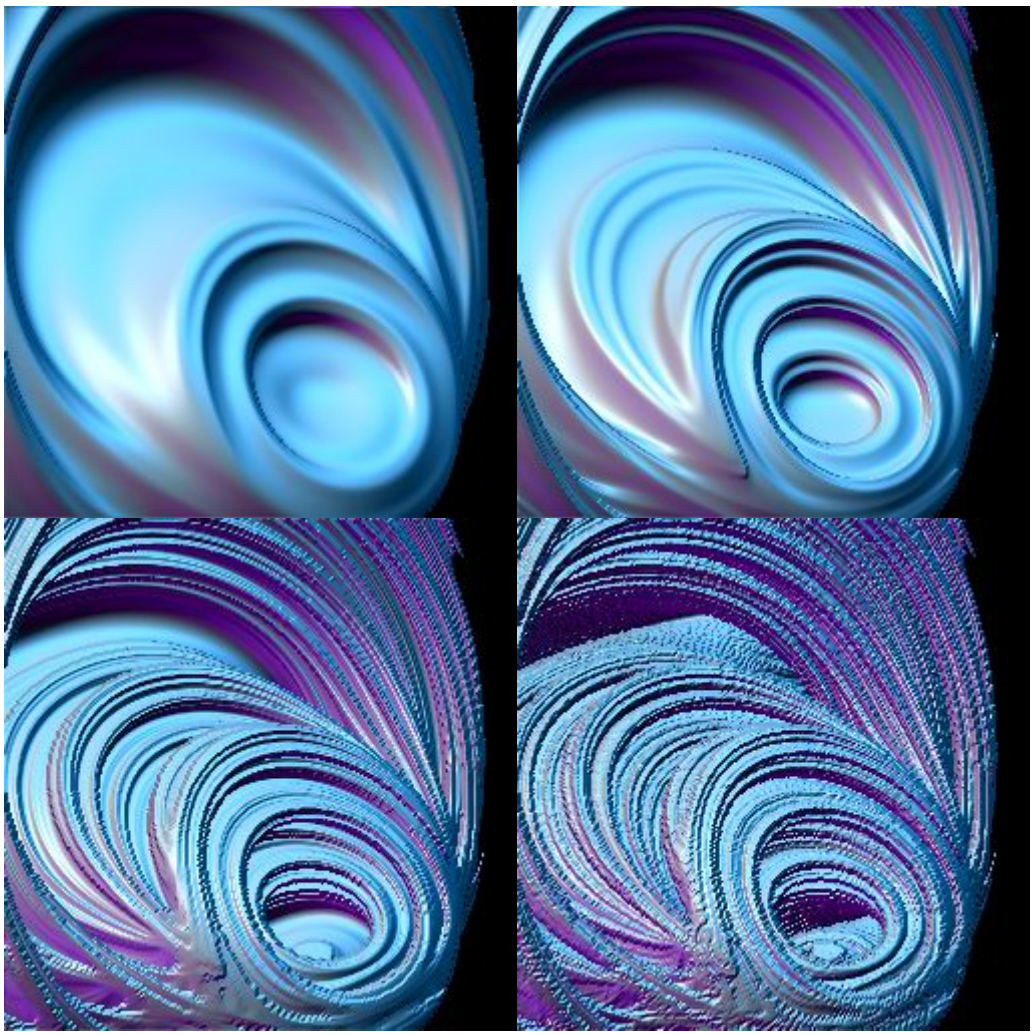

Рис. 15.

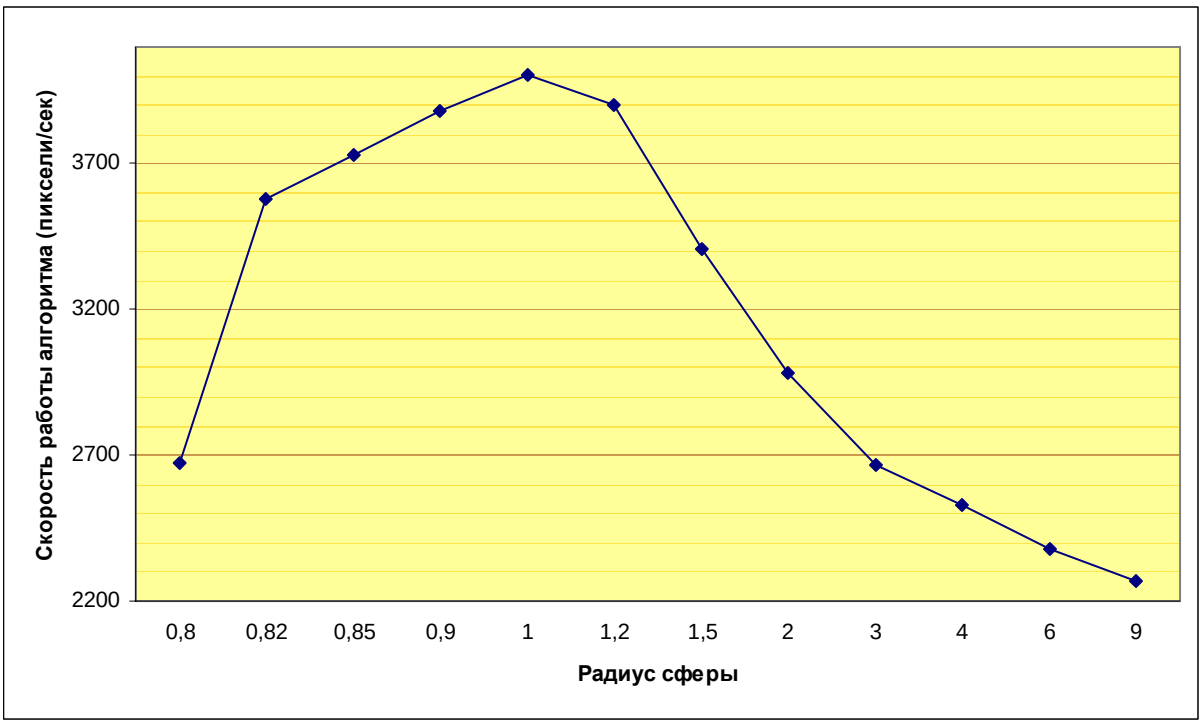

График 6.

Рис. 16.

Для третьего алгоритма была исследована зависимость от **α**. Зависимость скорости работы показана на графике 7. Рисунок 17 показывает влияние **α** на качество изображения (значения **α** 20, 14, 9 и 3). Видно, что при больших значениях **α** изображение – облако из точек – совсем не похоже на то, что должно получится (алгоритм неправильно определяет точки пересечения почти для всех лучей). При уменьшении **α** начинает проявляться реальная структура множества (алгоритм правильно определяет точки пересечения для большинства лучей). При **α** меньшем некоторого критического значения изображение получается хорошим (алгоритм всегда правильно определяет точки пересечения). При дальнейшем уменьшении **α** улучшения картинки не происходит, а вот скорость работы продолжает падать. Поэтому уменьшать **α** дальше критического значения не имеет смысла. Зависимость критического значения **α** (приблизительного, полученного опытным путем) от точности и **maxIter** показана на графике 4.

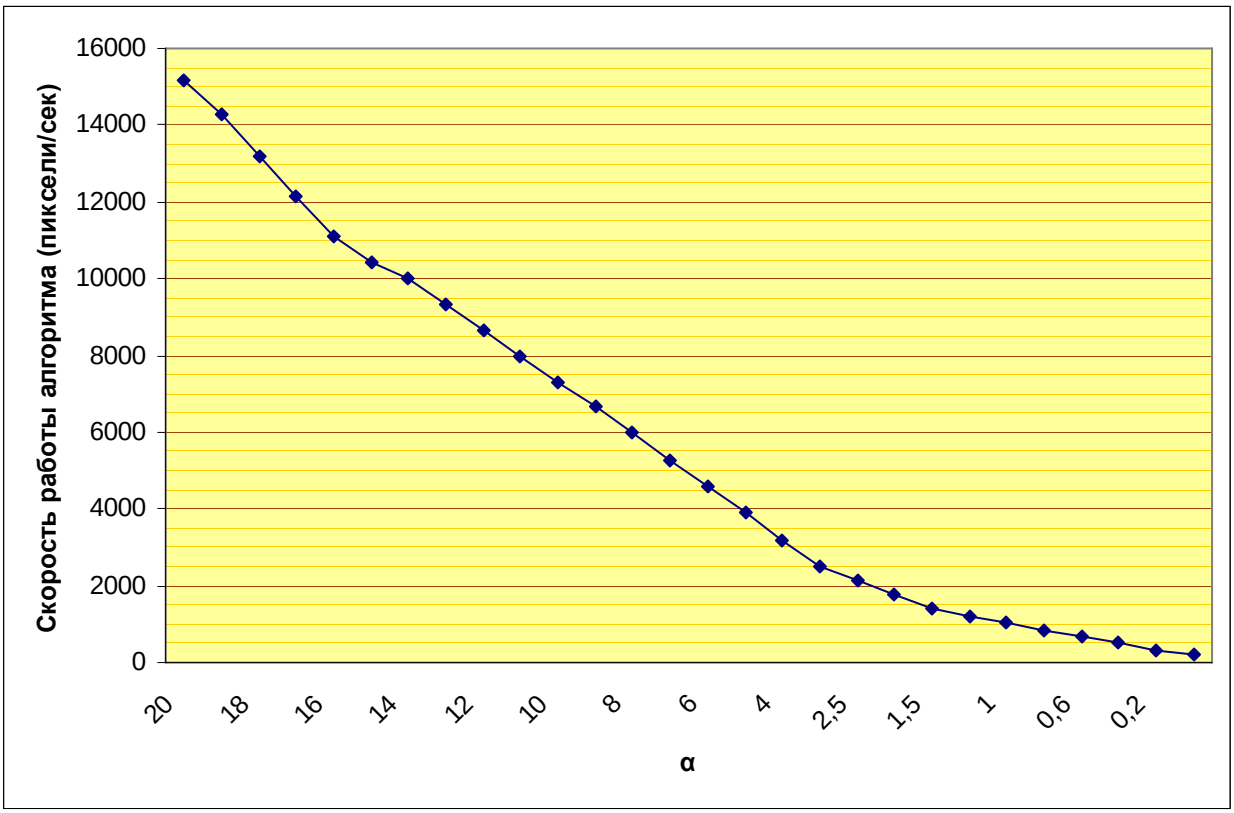

График 7.

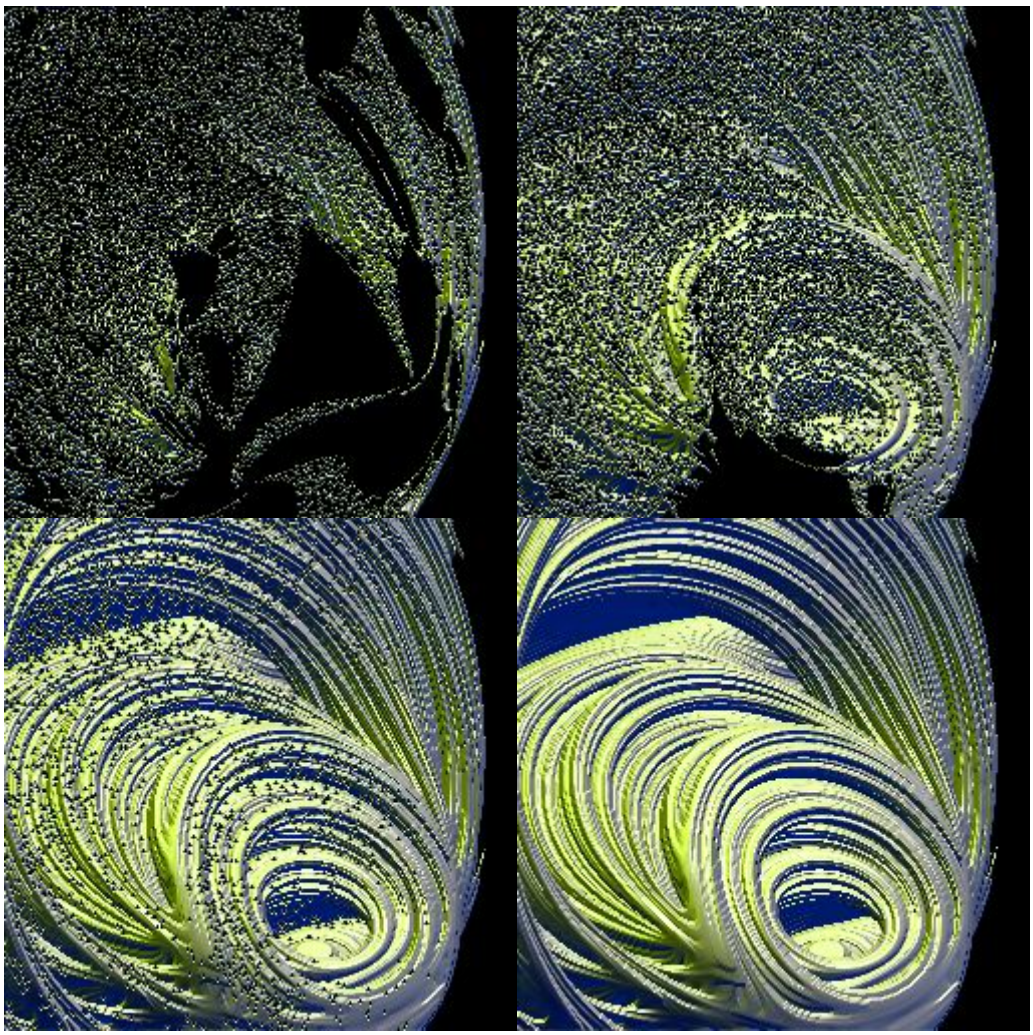

Рис. 17.

Приложение

А Определение принадлежности точки наполненному гиперкомплексному множеству Жюлиа

Надо определить принадлежность $\mathbf{h_0} \in \mathbf{H}$ наполненному множеству Жюлиа для отображения $\mathbf{f: H \to H}$ (например, $\mathbf{f(h) = f_c(h) = h^2 + c}$). Так как наполненное множество Жюлиа $\mathbf{K(f) = \{h \in H: \neg\ |f^n(h)| \to \infty}$, при $\mathbf{n \to \infty\}}$, то точка $\mathbf{h_0}$ принадлежит ему, если последовательность $\mathbf{\{f^n(h_0)\}}$ ограничена. На практике бесконечно итерировать **f** невозможно, поэтому приходится определять ограниченность последовательности приближенно. Выбирается достаточно большие $\mathbf{r \in R}$ и $\mathbf{maxIter \in N}$. Если все элементы подпоследовательности

$\{f^n(h_0)\}_0^{maxIter}$ меньше по модулю **r**, то считается, что вся последовательность ограничена, следовательно, h_0 принадлежит **K(f)**.

```
h ∈ H
m ∈ N
h = h0
for m = 1 to maxIter do
    h = f(h)                    // h = f^m(h0)
    if |h| > r then
        return h0_не_принадлежит_K(f)
return h0_принадлежит_K(f)
```

В Определение принадлежности точки гиперкомплексному множеству Мандельброта

Надо определить принадлежность **c ∈ H** множеству Мандельброта. Так как гиперкомплексное множество Мандельброта $\mathbf{M = \{c \in H: \neg\, f_c^n(0) \to \infty}$, при $\mathbf{n \to \infty\}}$ (например, $\mathbf{f_c(h) = h^2 + c}$), то надо определить ограниченность последовательности $\mathbf{\{f_c^n(0)\}}$. Алгоритм проверки аналогичен предыдущему.

```
h ∈ H
m ∈ N
h = 0
for m = 1 to maxIter do
    h = fc(h)                   // h = fc^m(0)
    if |h| > r then
        return c_не_принадлежит_M
return c_принадлежит_M
```

С Вычисление минимальной оценки расстояния (функции d_{min}) от точки до гиперкомплексного множества Жюлиа

Надо определить минимальную оценку расстояния от точки $\mathbf{h_0 \in H}$, лежащей вне множества $\mathbf{K_q = \{h \in H: \neg\, |f_q^n(h)| \to \infty}$, при $\mathbf{n \to \infty\}}$ ($\mathbf{f_q(h) = h^2 + q}$). В пункте 2.7 была дана следующая оценка

$$\mathbf{d(h_0, K_q) > \alpha\, |h_n| \,/\, D(|h_n|)},$$

где $\mathbf{D(|h_n|) = 2|h_{n-1}|D(|h_{n-1}|)}$, **α** – положительная вещественная константа. Т.е. необходимо вычислить $\mathbf{\alpha\, |h_n| \,/\, D(|h_n|)}$. Выбираются **α ∈ R**, **maxIter ∈ N,**

bigNumber $\in$ **R**. Чем больше **maxIter**, тем точнее оценка. Константа **α** подбирается опытным путем. Константа **bigNumber** зависит от наибольшего вещественного числа, которое можно представить на компьютере (**2·bigNumber**^{2} должно не превышать этого числа).

```
hn ∈ H
n ∈ N
dhn ∈ R
hn = h0
dhn = 1
for n = 1 to maxIter do
    dhn = 2 * |hn| * dhn    // dhn = D(hn), hn = fq^n(h0)
    hn = fq(hn)             // hn = hn = fq(hn–1) = fq^n(h0)
    if |hn| > bigNumber then break
    if dhn > bigNumber then break
return α * |hn| / dhn
```

Список литературы

1. Пайтген Х.-О., Рихтер П.Х. *«Красота фракталов»*, Москва, «МИР», 1993.
2. Ричард М. Кронвер. *«Фракталы и хаос в динамических системах. Основы теории»*, Москва, «Постмаркет», 2000.
3. Yumei Dang, Louis H. Kauffman, Daniel J. Sandin. *Hypercomplex Iterations, Distance Estimation and Higher Dimensional Fractals*. Series on Knots and Everything, vol. 17, 2002.
4. John C. Hart, Louis H. Kauffman, Daniel J. Sandin. *Interactive Visualization of Quaternion Julia Sets*. IEEE, pp. 209-218, 1990.
5. John C. Hart, Daniel J. Sandin, Louis H. Kauffman. *Ray Tracing Deterministic 3-D Fractals*. Computer Graphics, vol. 23, №3, pp. 289-296, 1989.
6. Alessandro Rosa. *Methods and Applications to Display Quaternion Julia Sets*. Electronic Journal Differential Equations and Control Processes, №4, pp. 1-22, 2005.