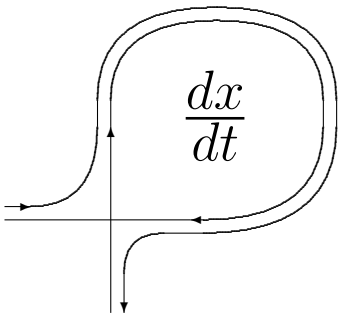


ДИФФЕРЕНЦИАЛЬНЫЕ УРАВНЕНИЯ
И
ПРОЦЕССЫ УПРАВЛЕНИЯ
N 3, 2006
Электронный журнал,
рег. N П23275 от 07.03.97

http://www.neva.ru/journal
e-mail: jodiff@mail.ru

Моделирование динамических систем

РАЗРАБОТКА И РЕАЛИЗАЦИЯ АЛГОРИТМОВ ПОСТРОЕНИЯ СИМВОЛИЧЕСКОГО ОБРАЗА

Е.И.Петренко

Россия, 198904, Санкт-Петербург, Петродворец, Университетский пр., 28,
Санкт-Петербургский Государственный Университет,
математико-механический факультет,
e-mail: zhene@mail.ru

Реферат.

Главной причиной интереса к изучению динамических систем послужило их все более возрастающее значение при исследовании процессов связанных с окружающим миром. Развитие теории динамических систем стимулировало активную разработку алгоритмов их исследования, а бурный рост возможностей вычислительной техники позволил широко применять методы компьютерного моделирования.

Для компьютерного моделирования и исследования динамической системы мы предлагаем использовать *символический образ* [5], который представляет собой ориентированный граф и является дискретизацией исходной динамической системы. Он строится по заданному покрытию фазового пространства ячейками $\{D_i\}$. Вершины графа соответствуют ячейкам, дуги — связям между ними, а именно: вершины i и j соединяются дугой, если образ ячейки D_i при действии динамической системы пересекается с ячейкой D_j. Многие задачи исследования динамических систем могут быть сведены к задачам исследования построенного ориентированного графа — символического образа.

В данной работе раскрываются аспекты реализации построения символического образа. Приводится способ введения координат, представления ячеек. Описаны четыре метода построения образа ячейки. Приводится их анализ и сравнение производительности.

1 Определения

Пусть M — подмножество m-мерного пространства $\mathbb{R}^m$. Как правило, M является замкнутым ограниченным множеством (компактом) или гладким многообразием в $\mathbb{R}^m$. Пусть $\mathbb{Z}$ — множество целых чисел и $\mathbb{R}$ — множество вещественных. Динамической системой называется непрерывное отображение $\Phi(x,t)$, где $x \in M,\ t \in \mathbb{Z}$ (или $t \in \mathbb{R}$), такое, что

$\Phi : M \times \mathbb{Z} \to M$ (или $\Phi : M \times \mathbb{R} \to M$),

$$\begin{aligned} \Phi(x,0) &= x, && (1)\\ \Phi(\Phi(x,t),s) &= \Phi(x,t+s), && (2) \end{aligned}$$

где $t,\ s$ принадлежат $\mathbb{Z}$ (или $\mathbb{R}$). Переменная t трактуется как время, а многообразие M называется фазовым пространством. Если $t \in \mathbb{Z}$, то мы получаем динамическую систему с дискретным временем, которая называется каскадом или дискретной динамической системой. Часто дискретные динамические системы порождены итерационными процессами вида $x_{n+1} = f(x_n)$ или разностными уравнениями. В случае $t \in \mathbb{R}$, мы имеем дело с системой с непрерывным временем, которая называется непрерывной динамической системой. Иногда непрерывные динамические системы называются потоками. Как правило, непрерывные динамические системы порождены автономными системами дифференциальных уравнений $\dot{x} = f(x)$.

Мы будем рассматривать дискретные динамические системы, порожденные итерационными процессами вида $x_{n+1} = f(x_n)$, где f — диффеоморфизм, D — компакт, $f : D \subset \mathbb{R}^m \to D$ задает динамическую систему с дискретным временем.

Определение 1 *Степень отображения будем обозначать, и определять следующим образом.*

$$\begin{aligned} f^0(x) &= x, \\ f^p(x) &= f^{p-1}(f(x)), p \geq 2, \\ f^{-p}(x) &= (f^{-1})^p(x), p > 0. \end{aligned} \qquad (3)$$

Определение 2 *Пусть $x_0 \in D$. Положительной траекторией или орбитой точки x_0 будем называть множество*

$$\left\{x_0, x_1 = f(x_0), x_2 = f(x_1) = f^2(x_0), \ldots\right\}. \tag{4}$$

Отрицательная орбита или траектория точки x_0 — это множество

$$\left\{x_0, x_1 = f^{-1}(x_0), x_2 = f^{-1}(x_1) = f^{-2}(x_0), \ldots\right\}. \tag{5}$$

Определение 3 *Если $f(x_0) = x_0$, то такая точка называется неподвижной точкой динамической системы.*

Например, рассмотрим динамическую систему, заданную отображением

$$f : x \longrightarrow 2006x. \tag{6}$$

Неподвижной точкой этой динамической системы является точка $x_0 = 0$.

Определение 4 *Если $f^k(x_0) = x_0$, при $k > 0$ и для любого $0 < p < k$ $f^p(x_0) \neq x_0$, то тогда точка x_0 называется k-периодической.*

При численном моделировании динамических систем оказывается удобным рассматривать ε-траектории.

Определение 5 *Пусть задан набор точек $\{x_0, x_1, \ldots\}$. Тогда будем говорить, что задана ε-траектория отображения f, если*

$$|x_{i+1} - f(x_i)| < \varepsilon, \tag{7}$$

для всех $i \geq 0$

При помощи ε-траектории мы можем задать погрешность определения орбиты или погрешность вычисления значения функции. Аналогично можно определить (p, ε)-периодическую точку, ε-неподвижную точку. Например, при $\varepsilon = 2$, ε-траекторией системы (6) будет являться траектория $\{1, 2005, 4022031, \ldots\}$

2 Символический образ динамической системы

Пусть $C = \{D_1, \ldots, D_k\}$ — конечное покрытие компакта D замкнутыми множествами. Множества D_i будем называть *ячейками* и $f : D \to D$ — диффеоморфизм. По множеству D, диффеоморфизму f и множеству ячеек C можно

построить ориентированный граф следующим образом: каждой ячейке D_i соответствует узел i графа. Между узлами i и j есть ориентированное ребро тогда и только тогда, когда $f(D_i) \cap D_j \neq \emptyset$. Таким образом построенный ориентированный граф будем называть *символическим образом* динамической системы, заданной диффеоморфизмом f.

Практически любую задачу исследования динамической системы можно свести к задаче исследования ориентированного графа — символического образа [5]. Например, можно локализовать неподвижные точки, периодические траектории, оценивать спектр Морса динамической системы [4]. Между исходной системой и символическим образом существуют следующие связи:

- траектории системы образуют пути на символическом образе;
- символический образ отражает глобальную структуру динамической системы;
- символический образ является конечным приближением динамической системы;
- точность этого приближения зависит от максимального размера ячейки.

Введем несколько параметров символического образа. Будем обозначать:

- d — максимальный диаметр ячейки;
- r — нижняя граница символического образа. Определяется как минимальное расстояние между $f(D_i)$ и множеством ячеек разбиения, которые не пересекаются с образом ячейки D_i.

Символический образ можно рассматривать как конечную дискретную аппроксимацию динамической системы. При этом более мелкое покрытие порождает более точную аппроксимацию. С помощью процесса последовательного подразбиения элементов покрытия можно строить последовательность символических образов и, тем самым, уточнять структурные характеристики системы.

Описанный метод может быть успешно применен к решению следующих задач [5]:

- локализация периодических траекторий заданного периода;
- построение периодической траектории;

- локализация цепно-рекуррентного множества;
- построение (положительно, отрицательно) инвариантного множества;
- построение аттрактора и его области притяжения;
- построение фильтраций и точной последовательности фильтраций;
- определение структурного графа динамической системы;
- оценка топологической энтропии;
- оценка показателей Ляпунова;
- оценка спектра Морса;
- построение изолирующих окрестностей инвариантных множеств.

Определение 6 *Последовательность z_k вершин графа G называется допустимым путем (или просто путем), если для любого k граф G содержит ориентированное ребро $z_k \to z_{k+1}$. Путь называется периодическим, если последовательность $\{z_k\}$ является периодической.*

Существует естественная связь между допустимыми путями на символическом образе G и ε-траекториями отображения f. Можно сказать, что допустимый путь является следом ε-траектории, причем обратное также верно. Справедлива следующая теорема [5]:

Теорема 1 *[5] Справедливы следующие утверждения для символического образа G:*

1. *Если последовательность $\{z_k\}$ есть путь на символическом образе G и $x_k \in D_{z_k}$, тогда последовательность $\{x_k\}$ есть ε-траектория гомеоморфизма f для любого $\varepsilon > q + d$, где q и d — наибольшие диаметры ячеек D_i и их образов $f(D_i)$ соответственно. В частности, если последовательность $\{z_k\}$ есть периодический путь на символическом образе, то последовательность $\{x_k\}$ есть ε-периодическая траектория.*
2. *Если последовательность $\{z_k\}$ есть путь на символическом образе G, тогда существует последовательность $\{x_k\}$, $x_k \in M(z_k)$, которая есть ε-траектория гомеоморфизма f для любого $\varepsilon > d$.*

3. Пусть последовательность $\{x_k\}$ есть ε-траектория гомеоморфизма f, $\varepsilon < r$ и $x_k \in D_{z_k}$, где r нижняя граница символического образа. Тогда последовательность $\{z_k\}$ есть допустимый путь на символическом образе G. В частности, если последовательность $\{x_k\}$ есть ε-периодическая траектория, то последовательность $\{z_k\}$ есть периодический путь на символическом образе G.

3 Анализ символического образа

При исследовании динамических систем наибольший интерес представляют неподвижные точки, периодические орбиты, а также цепно-рекуррентные множества.

Определение 7 *Точка x называется цепно-рекуррентной, если для любого $\varepsilon > 0$ существует периодическая ε-траектория, проходящая через x. Цепно-рекуррентным множеством называется множество всех цепно-рекуррентных точек.*

Цепно-рекуррентное множество инвариантно, замкнуто и содержит возвращающиеся траектории всех типов [5]. В частности цепно-рекуррентное множество содержит периодические орбиты.

Пусть Q — цепно-рекуррентное множество. $Q(\varepsilon)$ — множество точек ε-траекторий. Тогда из определения цепно-рекуррентного множества и множества ε-траекторий следует $Q \subseteq Q(\varepsilon)$ и $Q(\varepsilon_1) \subseteq Q(\varepsilon_2)$, при $\varepsilon_1 \leq \varepsilon_2$.

Определение 8 *Вершина символического образа называется возвратной, если существует периодический путь, проходящий через нее. Две вершины символического образа называются эквивалентными, если существует периодический путь, проходящий через них.*

Согласно определению, множество возвратных вершин разбивается на несколько классов эквивалентности. Ясно, что каждый периодический путь ξ находится в некотором классе, который однозначно определяется по ξ.

Рассмотрим те ячейки D_i, для которых вершина i является возвратной. Такое множество зависит от выбранного покрытия C и от наибольшего диаметра ячейки d. Поскольку зависимость от d для нас наиболее важно, обозначим

$$\begin{aligned} P(d) &= \{x \in D_i : i \text{ — возвратные}\}. \\ T(d) &= \{x \in D_k : k \text{ — невозвратные}\}. \end{aligned} \tag{8}$$

Теорема 2 *[5] Справедливы следующие утверждения:*

1. *Множество $P(d)$ является замкнутой окрестностью цепно-рекуррентного множества. Кроме того, $P(d)$ состоит из ε-периодических точек для любого $\varepsilon > q + d$, т. е.*

$$P(d) \subset Q(\varepsilon),\ \varepsilon > q + d. \tag{9}$$

2. *Цепно-рекуррентное множество Q совпадает с пересечением множеств $P(d)$ для всех положительных d:*

$$Q = \bigcap_{d>0} P(d). \tag{10}$$

3. *Точки из множества $T(d)$ не являются цепно-рекуррентными. Кроме того, если $\varepsilon < r$, то не существует ε-периодической траектории, проходящей через точку x множества $T(d)$, т.е.*

$$Q(\varepsilon) \bigcap T(d) = \emptyset,\ \varepsilon < r. \tag{11}$$

Множество $T(d)$ является замкнутым по построению и пара $P(d)$, $T(d)$ образует замкнутое покрытие D. Следовательно, множество $P(d)\backslash T(d)$ является окрестностью цепно-рекуррентного множества Q. Из теоремы 2 следует включение:

$$Q \subset Q(\varepsilon_1) \subset D \setminus T(d) = P(d) \setminus T(d) \subset P(d) \subset Q(\varepsilon_2), \tag{12}$$

где $\varepsilon_1 < r < d < q + d < \varepsilon_2$.

Отметим, что множества $P(d)$ не являются монотонными по d, т. е. из условия $d_1 > d_2$ не обязательно следует, что множество $P(d_1)$ содержит $P(d_2)$. Однако, если C_2 является подразбиением покрытия C_1, то $P(d_2) \subset P(d_1)$. Это свойство лежит в основе следующего алгоритма.

3.1 Алгоритм локализации цепно-рекуррентного множества

1. Строим исходное покрытие C компакта D. Находим символический образ G отображения f. Заметим, что ячейки исходного покрытия могут иметь произвольный диаметр d_0.

2. Выделяем на графе G возвратные вершины $\{i_k\}$. Если множество таких вершин пустое, значит локализуемое цепно-рекуррентное множество является пустым и процесс его локализации прекращается. Иначе, используя их, находим замкнутую окрестность $P = \{x \in D_{i_k} : i_k$ — возвратные$\}$ цепно-рекуррентного множества Q.

3. Разбиваем ячейки $\{D_{i_k}\}$, соответствующие возвратным вершинам символического образа и, таким образом, определяем новое покрытие.

4. Строим символический образ G для нового покрытия.

5. Переходим ко второму пункту, если размеры ячеек построенного символического образа достаточно велики.

Повторяя процесс последовательного измельчения покрытия, мы получаем последовательность окрестностей P_0, P_1, $P_2, \ldots$ цепно-рекуррентного множества Q и последовательность наибольших диаметров d_0, d_1, $d_2, \ldots$ ячеек, соответствующих возвратным вершинам символического образа для покрытия C_k. Следующая теорема обосновывает описанный алгоритм локализации множества Q.

Теорема 3 *[5] Последовательность множеств $P_0, P_1, P_2, \ldots$ обладает следующими свойствами:*

1. *Окрестности P_k вложены друг в друга, т. е.*

$$P_0 \supset P_1 \supset P_2 \supset \ldots \supset Q, \tag{13}$$

2. *Если наибольшие диаметры $d_k \to 0$ при $k \to \infty$, то*

$$\lim_{k\to\infty} P_k = \bigcap_k P_k = Q. \tag{14}$$

Таким образом, описанный алгоритм дает монотонно убывающую последовательность окрестностей, сходящуюся к цепно-рекуррентному множеству.

Пусть $G(V, E)$ — ориентированный граф, V — множество вершин, E — множество ребер.

Определение 9 *Вершины v_1 и v_2 сильно связаны в G, если существует путь из v_1 в v_2 и из v_2 в v_1. Если все вершины в ориентированном графе сильно связаны, то G называется сильно связным.*

Как было показано в [5], задача о локализации цепно-рекуррентного множества заданной динамической системы сводится к исследованию соответствующего символического образа и выделению на нем классов возвратных вершин. Из определения возвратной вершины следует, что компоненте сильной связности соответствует объединение классов возвратных вершин. Таким образом, выделение таких классов на графе эквивалентно нахождению компонент сильной связности. Для этого использован алгоритм Тарьяна [7], который обладает достаточно хорошей оценкой сложности: $O(n+m)$, где n — количество узлов, m — количество ребер.

3.2 Алгоритм выделения компонент сильной связности

Алгоритм основан на обходе графа в глубину и использует два стека "стек" и "маршрут". Стек "маршрут" содержит путь от начальной вершины до текущей. Каждая новая исследуемая вершина опускается в стек "маршрут", а при возвратах — извлекается. В "стек" добавляются все просмотренные вершины. Все элементы найденной компоненты сильной связности удаляются после ее окончательного формирования.

Заведем счетчик вершин с некоторым начальным значением и припишем к каждой вершине 2 числовых параметра: "номер" и "связка". Поле "номер" определяется простой последовательной нумерацией вершин по мере их обхода алгоритмом. Поле "связка" для произвольной вершины хранит номер другой вершины, которая была нумерована раньше. Если рассматриваемая вершина является корнем дерева поиска компоненты сильной связности, то значения полей "номер" и "связка" совпадут. Заметим, что значение поля "связка" всегда меньше или равно значению поля "номер". Схему работы алгоритма можно представить следующим образом.

- **Шаг 1.** Выбрать произвольную нерассмотренную вершину v, т.е. вершину для которой значение поля "номер" не было инициализировано.
- **Шаг 2.**
  - Положить вершину v в стеки "стек" и "маршрут".
  - Увеличить счетчик вершин на 1.
  - Присвоить полям "номер" и "связка" этой вершины значение счетчика.

- **Шаг 3.** Выбрать некоторое нерассмотренное ребро, выходящее из вершины v и рассмотреть вершину (w), в которую оно ведет.
  - Если ребро идет в не рассмотренную ранее вершину, положить $v = w$ и перейти на **Шаг 2**.
  - Если ребро идет в уже рассмотренную вершину, перейти на **Шаг 4**.
  - Если вершина v не имеет неисследованных выходов и значение поля "связка" меньше значения поля "номер", перейти на **Шаг 5**.
  - Если вершина v не имеет неисследованных выходов и значение поля "связка" равно значению поля "номер", перейти на **Шаг 6**.
- **Шаг 4.**
  - Если значение поля "номер" вершины w меньше значения поля "номер" вершины v и w находится в стеке "стек", тогда положить значение поля "связка" вершины v равным минимуму из значений полей "связка" вершин v и w.
  - Перейти на **Шаг 3**.
- **Шаг 5.**
  - Извлечь вершину v из стека "маршрут" и рассмотреть вершину, которая оказалась на вершине стека (u).
  - Положить значение поля "связка" вершины u равным минимуму из значений полей "связка" вершин v и u. Положить v равным u и перейти на **Шаг 3**.
- **Шаг 6.**
  - Взять все вершины с вершины стека "стек" до вершины v включительно и поместить их в новую компоненту сильной связности. Больше эта компонента изменяться не будет.
  - Извлечь вершину v из стека "маршрут". Если в итоге "маршрут" окажется пустым, перейти на **Шаг 1**, иначе положить v равным вершине на вершине "маршрута" и перейти на **Шаг 3**.

Заметим, что мы можем получить компоненту сильной связности, состоящую из одной вершины, при этом у нее нет ребра, ведущего в нее саму. Для того чтобы не рассматривать такие "компоненты сильной связности", было добавлено дополнительное условие на шаге формирования компоненты

сильной связности (**Шаг 6**). А именно, если на верхушке стека "маршрут" лежит текущая вершина v, то она единственная в своей компоненте. Согласно описанию алгоритма она становится компонентой сильной связности. Если у этой вершины есть ребро идущее в неё, то это компонента сильной связности, иначе это просто проходящая вершина.

На **Шаге 3** алгоритма требуется находить необработанные ребра. Для этого в каждой вершине графа будем хранить индекс последнего обработанного ребра.

На **Шаге 4** требуется проверять, находится ли данная вершина в стеке "стек". Для ускорения этой проверки введем в каждой вершине флаг. Флаг будет установлен, если вершина находится в стеке. Для того, чтобы выяснить, содержится ли вершина в стеке будет достаточно посмотреть на значение этого флага.

На **Шаге 6** для формирования компоненты сильной связности будем создавать новую структуру данных графа для сохранения выделенных вершин. Если для дальнейшего исследования требуются ребра, то после завершения работы алгоритма локализации компонент сильной связности, можно восстановить ребра в выделенной компоненте сильной связности.

Обозначим G исходный граф. Пусть для каждой компоненты связности графа G_i задан набор узлов N_i, входящих в нее. Тогда для построения графов компонент сильной связности G_{N_i} графа G следует выполнить следующий алгоритм:

- Для каждой вершины v компоненты сильной связности N_i найти соответствующую ей вершину v' в исходном графе G.
- Для всех ребер вида $v' \to w'$, w' вершина G, если вершина $w' \in N_i$, добавить это ребро в граф G_{N_i}.

После завершения процесса локализации компонент сильной связности, мы получаем набор из графов-компонент сильной связности исходного графа. Каждый такой граф соответствует цепно-рекуррентному множеству исходной системы [5]. Теперь исходный граф можно удалить из памяти.

4 Представление ячейки

Ячейка представляется координатами ее вершин и центра. В реализации алгоритма удобно рассматривать одинаковые ячейки, тогда информацию о раз-

мерах ячейки можно хранить отдельно и всего один раз. Каждая ячейка в таком случае представляется точкой ее *"верхнего левого угла"* единственным образом.

Для избежания ошибок при работе с плавающей арифметикой на компьютере, рассматривается целочисленная система координат, за единицу длины, в которой принимается размер ячейки.

Введем обозначение $[a,b] = [a_1,b_1] \times \cdots \times [a_m,b_m]$.

Возьмем $a, b \in \mathbb{R}^m$, такие, что $a \neq b$, тогда

$$D = [a,b] \subset \mathbb{R}^m. \tag{15}$$

Зафиксируем m положительных чисел $p_1, p_2, \ldots, p_m$ и разобьем каждый отрезок $[a_i,b_i]$ на p_i равных частей:

$$\begin{array}{rcl} d_i & = & \frac{b_i-a_i}{p_i}, \\ W_i^j & = & [a_i+(j-1)d_i, a_i+jd_i], \\ [a_i,b_i] & = & W_i^1 \cup W_i^2 \cup \cdots \cup W_i^{p_i}, \\ W_i^{j_1} \cap W_i^{j_2} & = & \emptyset, j_1 \neq j_2. \end{array} \tag{16}$$

Определим множество всех ячеек разбиения.

$$W = \{W_1^{j_1} \times W_2^{j_2} \times \cdots \times W_m^{j_m} | 1 \leq j_k \leq p_k, 1 \leq k \leq m\}. \tag{17}$$

Элемент множества W будем называть ячейкой. *Целочисленной координатой* ячейки $c \in W$ будет набор значений $(j_1, j_2, \ldots, j_m)$, с которыми эта ячейка входит в множество W.

Пусть $a, b \in \mathbb{R}^k$, $(p_i)_{i=1}^k$ — набор положительных чисел. Будем обозначать через $W([a,b],p)$ множество ячеек, построенное описанным выше способом.

Использование представления ячеек в целочисленной системе координат позволяет уменьшить объем памяти, требуемой для хранения ячейки. Для вычисления координат вершин ячейки в исходной системе координат проводится линейное преобразование координат.

При построении последовательности символических образов ячейки, соответствующие вершинам символического образа разбиваются на равное количество частей. Удобно задавать количество частей по каждой координате отдельно. Пусть задан вектор разбиения ячеек $r = (r_1, r_2, \ldots, r_m)$, $r_i > 0$. Тогда каждой ячейке c с координатами $(x_1, x_2, \ldots, x_m)$ соответствуем множество ячеек $(x_1r_1 + j_1, x_2r_2 + j_2, \ldots, x_mr_m + j_m)$, где $j_i \in [0, r_i) \cap \mathbb{Z}$.

5 Структура данных символического образа

Процесс построения символического образа представляет собой последовательную обработку ячеек. Для экономии памяти оказывается удобным хранить в графе только вершины, соответствующие обработанным ячейкам.

Для того чтобы обеспечить быстрое построение символического образа требуется, чтобы структура данных, представляющая ориентированный граф, была устроена так, чтобы оптимально выполнялись следующие операции:

- быстрый поиск вершин. При добавлении новой вершины, нужно проверить, не была ли эта вершина добавлена в граф раньше. И если была, то нужно просто вернуть указатель на нее и не производить никаких дополнительных действий;
- быстрый поиск ребер. При добавлении нового ребра, следует проверить, не было ли добавлено это ребро раньше.
- оптимальное использование памяти. Построенная нами структура данных должна работать с большими графами.

Граф представляется с помощью списка инциденции, т.е. списка вершин, каждая из которых содержит список исходящих из нее ребер. Для ускорения поиска списки ребер каждой вершины и список узлов хешируются при помощи хэш-таблиц.

6 Методы построения образа ячейки

Построение символического образа основано на построении образа ячейки. На рис. 1 показан образ ячейки (заштрихован темным цветом). Понятно, что в зависимости от выбора ячеек покрытия, попадающих в этот образ, мы будем получать различные графы.

6.1 Линейный метод

В этом методе мы оцениваем возможные значения функции на заданной ячейке. Для этого производятся следующие действия (рис. 2):

- берем ячейку разбиения D_i;

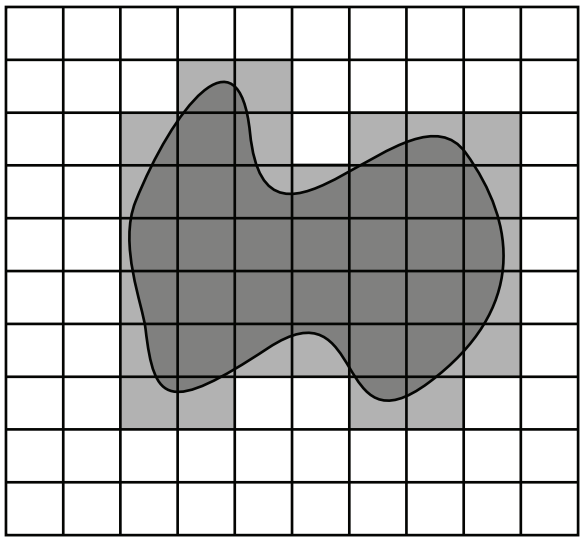


Рис. 1: Разбиение множества D, образ ячейки.

- вычисляем значения функции системы f на вершинах ячейки D_i;
- строим минимальный прямоугольник E, ориентированный по осям координат, который содержит образы вершин ячейки D_i;
- образом ячейки будем считать множество ячеек покрытия, которые пересекаются с построенным прямоугольником E.

Для ускорения работы этого метода значения функции системы в вершинах ячейки вычисляются с точностью до членов первого порядка при разложении этой функции в ряд Тейлора в окрестности центра ячейки x_0, т.е.

$$\tilde{f}(x) = f(x_0) + Df(x_0)(x - x_0), \tag{18}$$

где x — вершина ячейки.

Чтобы увеличить вероятность того, что построенный этим методом образ ячейки содержит все точки образа ячейки, можно ввести коэффициент расширения полученного прямоугольника.

6.2 Точечный метод

Пусть дано число $n \in \mathbb{N}^m$, т.е. $n = (n_1, n_2, \ldots, n_m)$, где $n_i \in \mathbb{N}$. Тогда точечный метод с дроблением n действует следующим образом. В ячейке равномерно берем n_i точек по i-й координате. Образом ячейки будем называть

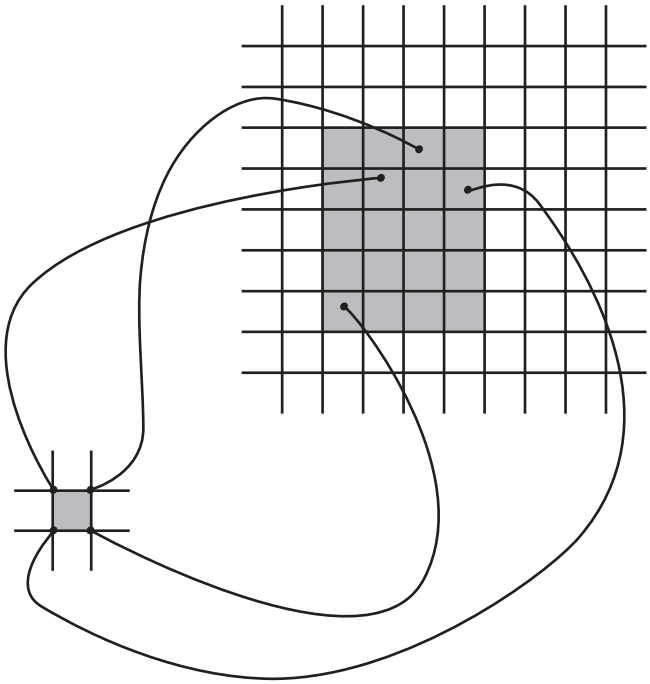


Рис. 2: Построения образа ячейки. Линейный метод.

набор ячеек, в которые попали образы равномерно брошенных точек (рис. 3). Результат и время работы метода зависит от выбора значения вектора n.

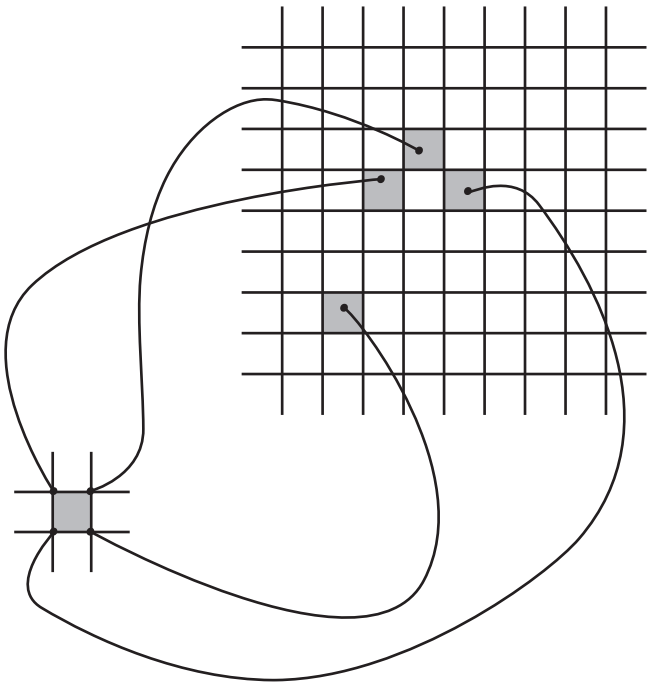


Рис. 3: Построение образа ячейки при $N = (2, 2)$. Точечный метод.

6.3 Улучшенный точечный метод

Пусть даны число $n \in \mathbb{N}^m$ и $b \in [0,1]^m$, тогда улучшенный точечный метод с дроблением n и наложением b действует аналогично точечному методу с дроблением n. Образу равномерно брошенной точки x будем сопоставлять ячейку, которая содержит образ $f(x)$ и все ячейки, которые находятся ближе чем $\frac{b_i d_i}{2}$ от этой точки, где i номер оси, в направлении которой идет отрезок-граница ячейки.

Если образ некоторой равномерно брошенной точки попадет на светло-серую область ячейки (рис. 4), то тогда добавляем к результату не только эту ячейку, но и соседнюю, как показано на рис. 4. Если точка лежит близко от нескольких границ ячейки, то следует добавлять несколько ячеек (рис. 4).

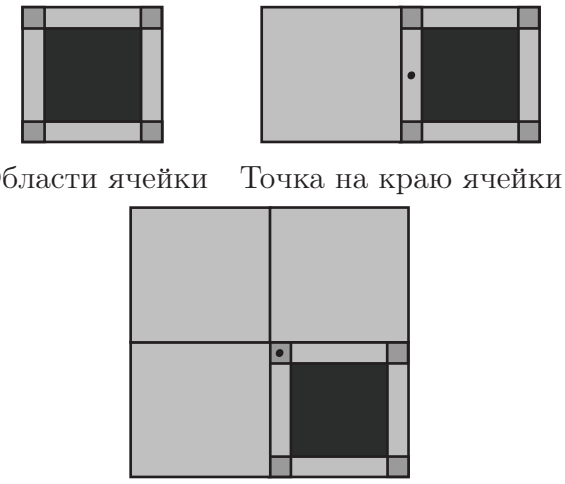


Рис. 4: Точки на краю ячейки

Для поиска соседних ячеек был разработан специальный алгоритм, который довольно эффективно вычисляет целочисленные координаты всех ячеек, которые должны быть добавлены.

6.4 Адаптивный метод

Недостаток описанных выше методов построения образа ячейки заключается в том, что эти методы вычисляют функцию в заранее определенных точках. Предлагается новый метод построения образа ячейки, который будет вычислять значение функции только в тех точках, выбранных в процессе работы алгоритма, в которых это действительно необходимо, при этом количество

таких точек, по которым строится образ ячейки тоже будет зависеть от поведения системы на данной ячейке. Такой способ построения образа ячейки будем называть *адаптивным методом.*

Адаптивный метод работает по принципу *точечного метода* (см. 6.2). Единственное отличие его заключено в том, что точки, в которых вычисляются значения функции берутся не равномерно, как раньше, а исходя из поведения системы.

Определение 10 *Пусть есть множество точек $P \subseteq D$ и рефлексивное симметричное отношение ψ на множестве P. Будем называть такое отношение ψ отношением соседства. Пусть задана функция правой части системы $f = (f_1, f_2, \ldots, f_m)$. Фиксируем $\varepsilon = (\varepsilon_i)_{i=1}^m$, $\varepsilon_i > 0$. Тогда будем называть множество P ε-множеством для функции f и отношения ψ, если для любых двух соседних ($\psi(x, y)$) точек $x, y \in P$ выполнено соотношение*

$$|f_i(x) - f_i(y)| \leq \varepsilon_i. \tag{19}$$

Будем обозначать такое множество через $P^{\varepsilon}_{f,\psi}$.

Определение 11 *Пусть задан набор m положительных чисел $\varepsilon = (\varepsilon_i)_{i=1}^m$. Тогда ε-окрестностью точки $x = (x_1, x_2, \ldots, x_m) \in \mathbb{R}^m$ будем называть множество*

$$[x_1 - \varepsilon_1, x_1 + \varepsilon_1] \times [x_2 - \varepsilon_2, x_2 + \varepsilon_2] \times \cdots \times [x_m - \varepsilon_m, x_m + \varepsilon_m] \subset \mathbb{R}^m. \tag{20}$$

Пусть задано множество $P^{\varepsilon}_{f,\psi}$, и ячейка D_j, все вершины которой принадлежат множеству P. Тогда образ ячейки будет состоять из ε-окрестностей образов точек, которые лежат в $P \cap D_j$. При этом если элементы ε достаточно малы, т.е.

$$\varepsilon_i < \frac{d_i}{2}, \tag{21}$$

то можно искать ячейки по принципу, описанному в *улучшенном точечном методе* (см. 6.3).

Введенное отношение соседства ψ можно представить как неориентированный граф, где вершины соответствуют точкам множества P, а $\psi(x, y)$ эквивалентно существованию ребра между вершинами x и y или совпадению этих вершин $x = y$.

6.5 Алгоритм построения графа для $P^{\varepsilon}_{f,\psi}$

Предположим, задано начальное множество точек и граф, определяющий соотношение соседства для точек множества. Зафиксируем $\varepsilon > 0$.

Определение 12 *Вектором длины ребра r между узлами графа x и y будем называть вектор пространства $\mathbb{R}^m$, компонентами которого будут расстояния между проекциями образов точек x и y под действием функции системы f.*

$$length(r)_i = |f_i(x) - f_i(y)|. \tag{22}$$

На множестве векторов длин можно ввести отношение порядка. Будем говорить, что вектор a больше вектора b, если это соотношение выполнено покомпонентно.

Определение 13 *Обозначим через $E(n)$ множество всех вершин, которые соединены одним ребром с вершиной n.*

Для того, чтобы заданное множество удовлетворяло требованиям определения, нужно проверить, что все ребра не длиннее ε. Представим все ребра исходного графа в виде упорядоченного по убыванию длины ребра списка L. В качестве длины будем брать сумму модулей компонент вектора длин.

Пока список L не пуст, возьмем ребро r из начала списка и удалим его из списка. Если ребро r между точками x и y такое, что вектор его длины больше ε, выполним следующее:

1. Добавим в граф разбиения новую вершину p с координатами $(\frac{x_i+y_i}{2})_{i=1}^m$.
2. Удалим ребро r из графа.
3. Добавим ребра (x, p) и (y, p) в граф и в список L, с сохранением порядка.
4. Добавим ребра к вершинам, принадлежащим $E(x) \cap E(y)$ в граф и список L с сохранением порядка.

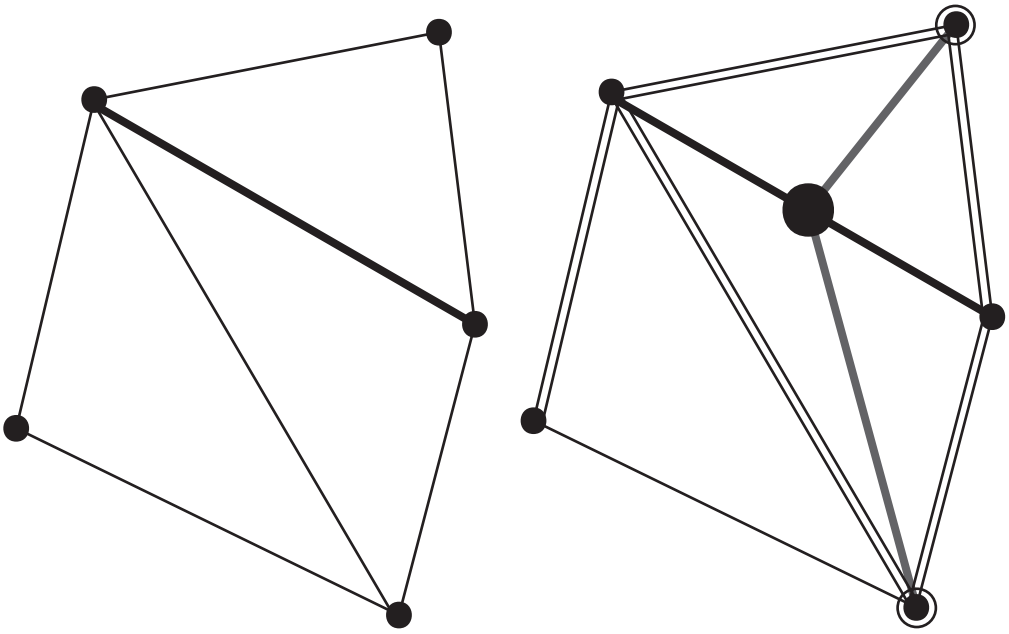
Основная сложность в данном алгоритме возникает на шаге 4. Нужно определить, какие ребра добавлять к новому узлу графа, т.е. следует определить расширение отношения ψ на новую точку p. Следует наиболее полно расширить отношение ψ, чтобы учесть все расстояния от новой точки p до остальных точек графа.

На практике возникают ситуации, когда описанный процесс работает очень медленно из-за большого количества добавлений новых вершин в граф. В этом случае мы вводим ограничение на количество вершин в графе разбиения. А именно: множество ребер графа делится на два класса: ребра с известной длиной и ребра, длина которых не определена. Алгоритм построения образа ячейки по графу разбиения работает аналогично приведенному выше, однако, при построении ε-окрестности для вершины этого графа берем ε равным длине наибольшего ребра. Можно приписать каждому ребру его длину и хранить упорядоченный список ребер для каждой вершины. В вершинах можно хранить значение образа точки. Введенные оптимизации позволили увеличить скорость работы алгоритма.

Пример работы алгоритма для 2-х мерного случая

Рассмотрим граф для множества P, которое состоит из 4-х точек (рис. 5). Предположим мы проверяем длину выделенного жирно ребра. Тогда, согласно алгоритму, следует ввести новую точку и некоторое число новых ребер.

Пометим двойной линией ребра, которые имеют одну общую вершину с рассматриваемым (выделенным жирно) ребром. Тогда, согласно алгоритму, следует добавить новые ребра только к тем вершинам, которые соединены ребрами с двумя вершинами рассматриваемого ребра (рис. 5).



Исходный граф Дробление ребра

Рис. 5: Разбиение длинного ребра.

Выделим такие вершины кругом. Тогда появляются два новых ребра, которые нарисованы серым цветом. Эти ребра будут добавлены в конец списка L. На следующих шагах алгоритм проверит длины оставшихся ребер.

6.6 Начальное разбиение ячейки

Вершины графа разбиения соответствуют точкам фазового пространства исследуемой системы. Результат и скорость работы адаптивного метода зависит от выбора начального разбиения ячейки. Мы предлагаем следующие графы разбиения ячейки для одномерного, двумерного и трехмерного случаев (рис. 6):

- для одномерного случая достаточно взять взять граф с одним ребром и двумя вершинами, которые соответствуют границам одномерной ячейки;
- в двумерном случае будем рассматривать граф из 5 вершин — центр ячейки и ее четыре вершины и 8 ребер, которые связывают соседние вершины;
- в трехмерном случае мы рассматриваем все вершины ячейки, все центры ее граней и центр ячейки. Соединяем ребром соседние вершины.

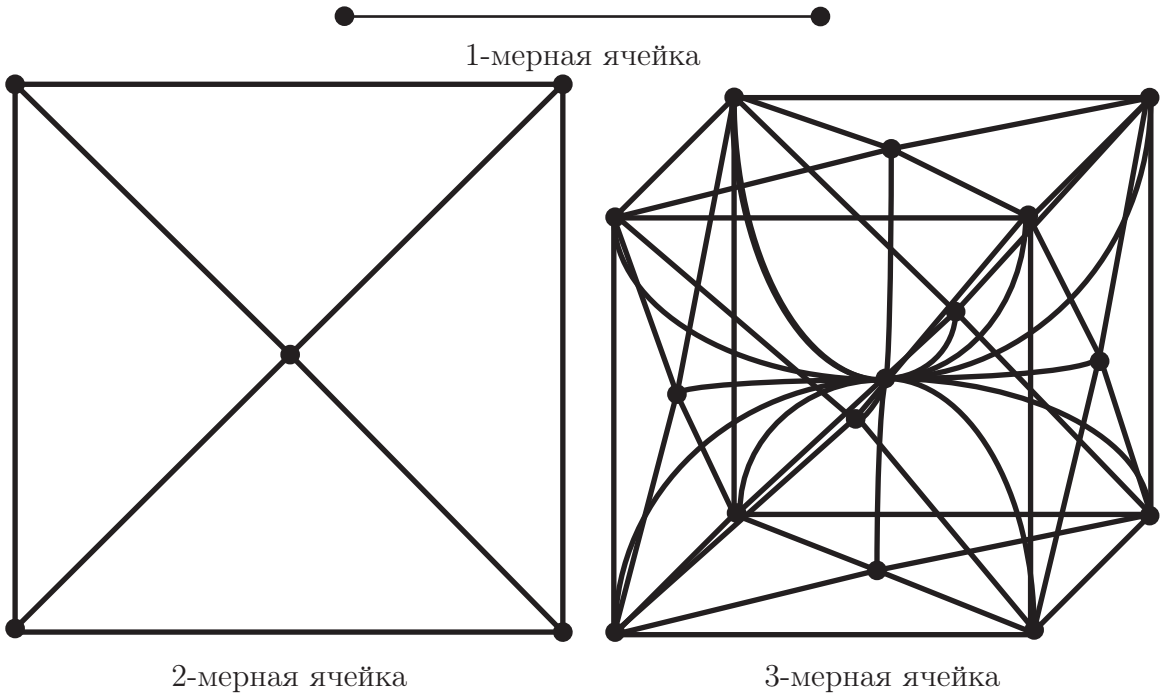


Рис. 6: Графы начального разбиения

7 Примеры

В этой части приведены примеры построения цепно-рекуррентных множеств динамических систем с помощью построения последовательности символических образов с использованием разных методов построения образа ячейки. Приведены рисунки полученных цепно-рекуррентных множеств и таблицы сравнений работы различных методов построения образа ячейки. В таблице также показано время работы алгоритма и количество компонент сильной связности графа.

Количество шагов процесса подразбиения символического образа определяется экспериментально для каждой системы. Следующий шаг построения символического образа проводится, если для этого хватит оперативной памяти компьютера.

На каждом шаге построения символического образа будем разбивать исходную ячейку на 2 части по каждой координате. Точечный метод будет строить образ ячейки по 2 точкам по каждой координате, улучшенный точечный метод использует наложение, равное 10% от размера ячейки. Ограничение на количество точек для построения образа ячейки адаптивным методом не устанавливается.

Все эксперименты были проведены на компьютере `Intel Pentium 4 3Ghz`, `1Gb` оперативной памяти. Операционная система `Microsoft Windows XP SP2`.

7.1 Отображение Хенона [10]

Рассмотрим систему:

$$\begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} 1 - ax^2 + by \\ x \end{pmatrix}, \tag{23}$$

при $a = 1.4$, $b = 0.3$.

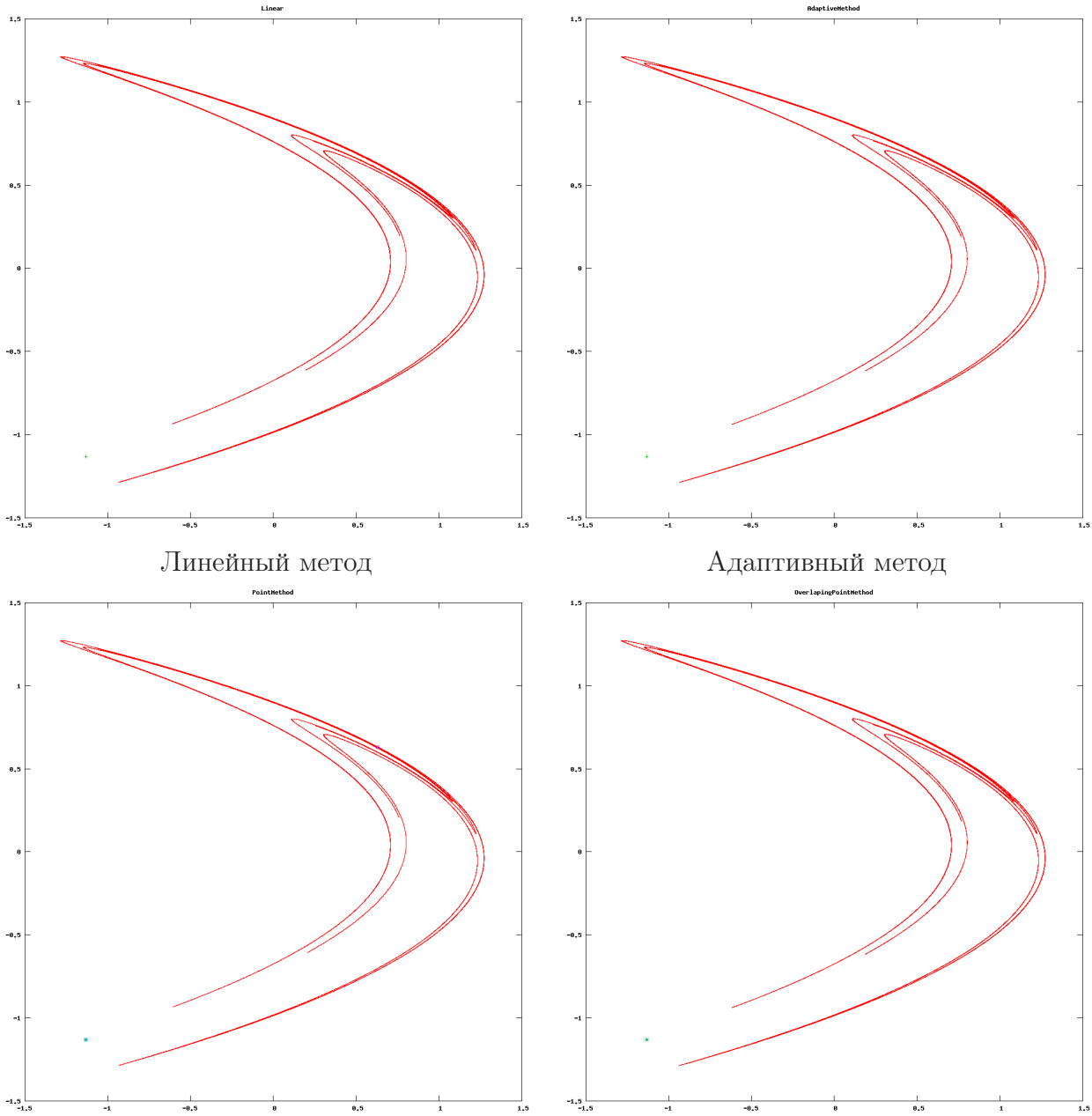
Известно [5], что при заданных параметрах у этой системы существует инвариантное множество в области $[-1.5, 1.5] \times [-\frac{10}{3}, \frac{10}{3}]$. Возьмем $D = [-10, 10] \times [-10, 10]$. Проведем 13 шагов построения символического образа. Сведем результаты вычислений в таблицу. На каждом шаге будем выполнять построение символического образа и его анализ, согласно алгоритму, приведенному в 3.1, т.е. выделение цепно-рекуррентного множества с помощью

алгоритма выделения компонент сильной связности.

Метод	Кол-во узлов	Кол-во ребер	Время работы	Кол-во компонент
Линейный метод	572 435	5 509 083	24 047 ms	2
Точечный метод	298 397	853 915	40 656 ms	5
Улучшенный точечный метод	505 152	3 877 289	465 563 ms	3
Адаптивный метод	507 637	5 171 045	464 844 ms	2

Каждым методом было сделано по 13 шагов. Начальное разбиение — по 10 ячеек на каждую координату. На каждом шаге ячейка разбивается на 2 части по каждой координате.

Приведем иллюстрации построенного цепно-рекуррентного множества. Построенные цепно-рекуррентрые множества получились практически одинаковые, отличие видно только по количеству узлов. Самый маленький граф получается при использовании точечного метода.



Линейный метод

Адаптивный метод

Точечный метод

Улучшенный точечный метод

Рис. 7: Результаты вычислений. Отображение Хенона.

7.2 Отображение Икеда [11]

Отображение имеет вид:

$$\tau(x,y) = C_1 - \frac{C_3}{1+x^2+y^2} \tag{24}$$

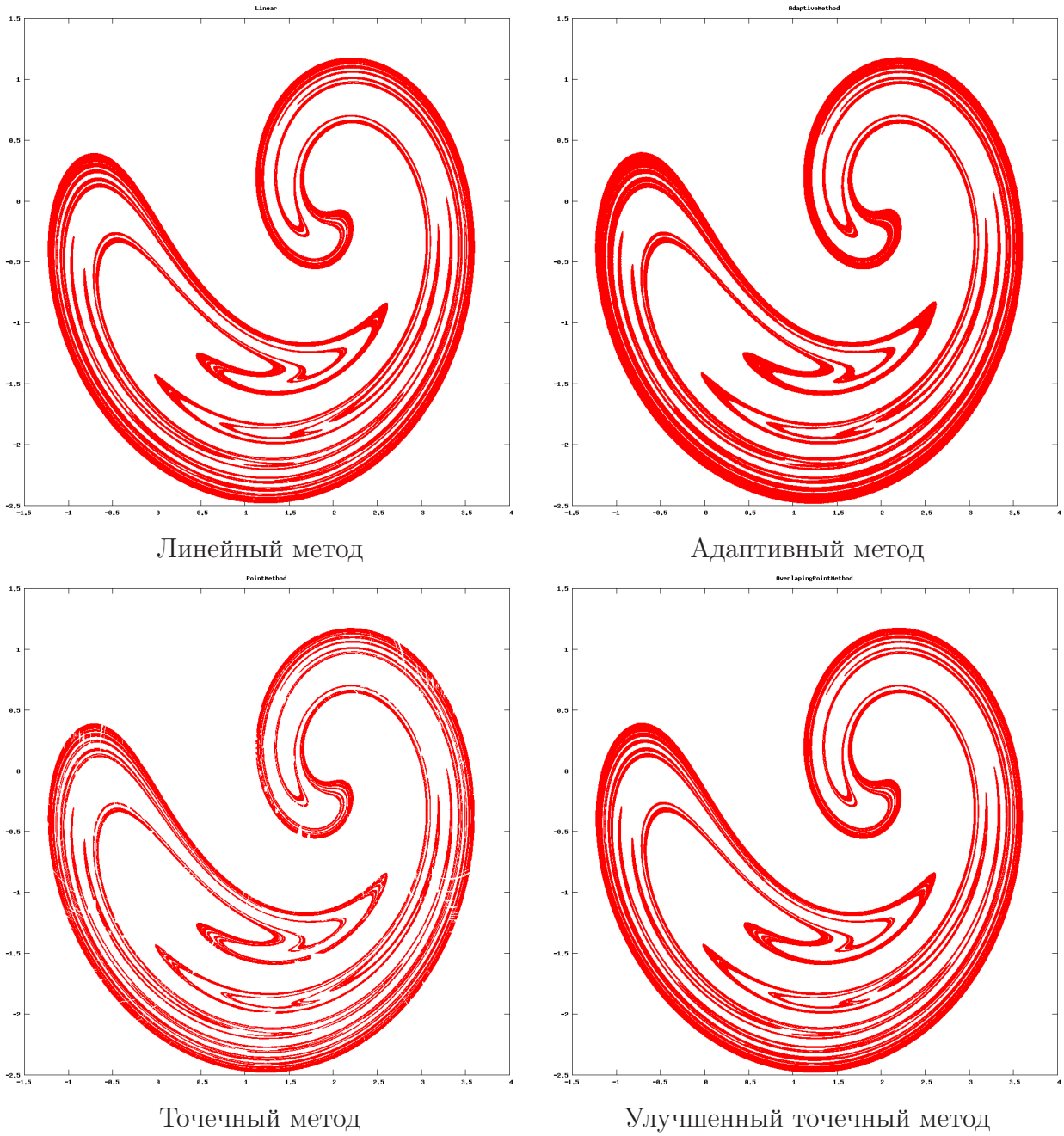
$$\begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} d - C_2(x\cos\tau(x,y) - y\sin\tau(x,y)) \\ C_2(x\sin\tau(x,y) + y\cos\tau(x,y)) \end{pmatrix}, \tag{25}$$

где $d = 2$, $C_1 = 0.4$, $C_2 = 0.9$, $C_3 = 6$.

Пусть $D = [-10, 10] \times [-10, 10]$. Отображение Икеда возникает при моделировании оптических носителей (кристаллов) информации. Оно было рассмотрено и исследовано в [11, 13]; было показано, что при заданных параметрах это отображение имеет цепно-рекуррентное множество.

Метод	Кол-во узлов	Кол-во ребер	Время работы	Кол-во компонент	Кол-во шагов
Линейный метод	3 117 236	25 988 943	172 484 ms	2	11
Точечный метод	1 732 774	4 378 370	112 359 ms	4	11
Улучшенный точечный метод	2 949 314	21 666 315	512 969 ms	2	11
Адаптивный метод	1 017 624	11 740 358	640 609 ms	2	10

Результаты вычисления символического образа отображения Икеда приведены на рис. 8. Видно, что результат, полученный точечным методом немного более “дырявый”, по сравнению с результатами, полученными другими методами. Согласно [13], наиболее хорошие результаты дали линейный и адаптивный методы.



Линейный метод | Адаптивный метод

Точечный метод | Улучшенный точечный метод

Рис. 8: Результаты вычислений. Отображение Икеда.

7.3 Множество Жюлиа для $r = -0.12 + 0.74i$ [6, 12]

В исследованиях Пайтгена и Рихтера [6] были приведены алгоритмы и результаты построения множеств Жюлиа и наполненных множеств Жюлиа для комплексного отображения

$$z \to z^2 + r. \tag{26}$$

Система, эквивалентная (26), [12]:

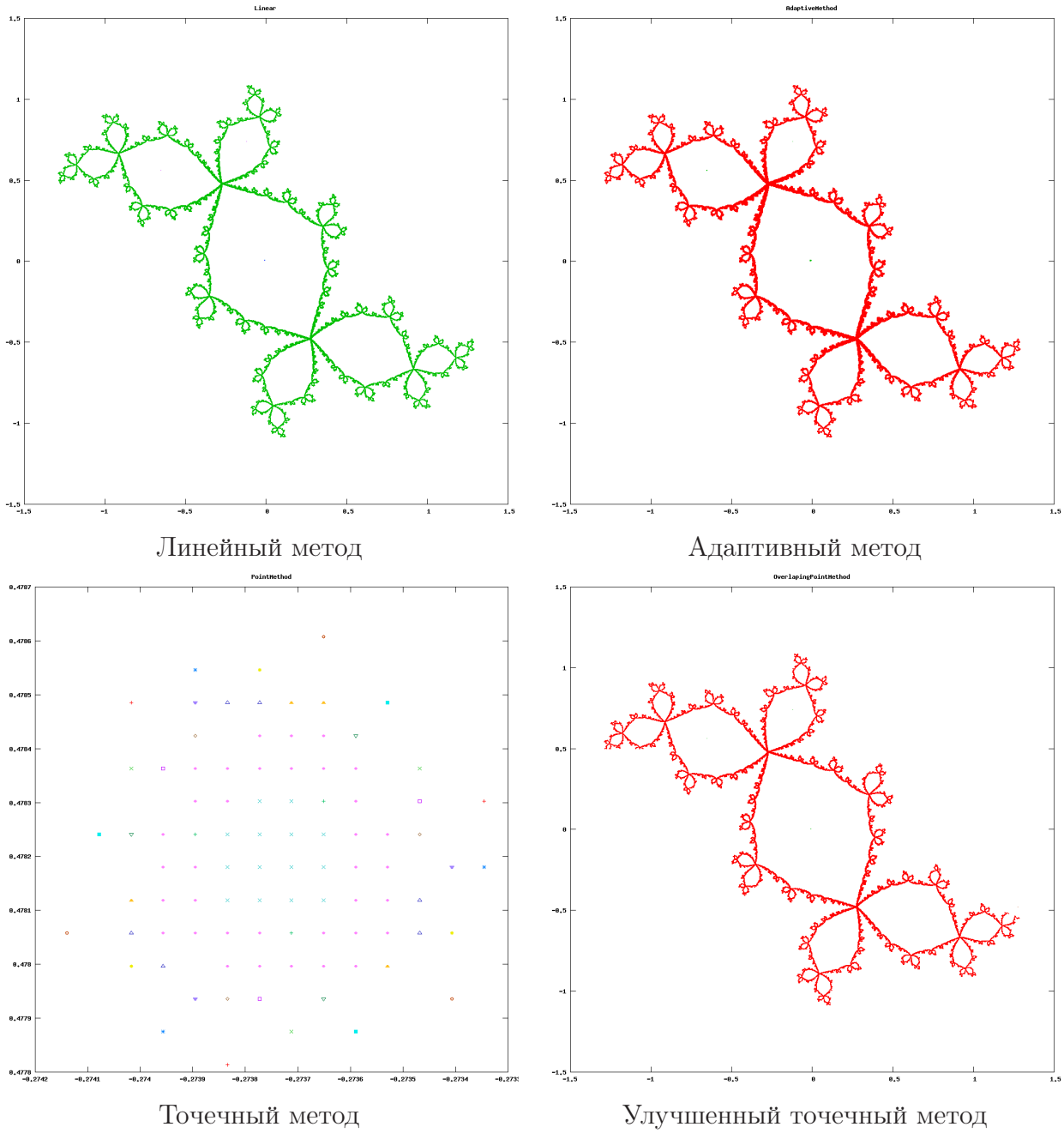
$$\begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} x^2 - y^2 & + & re \\ 2xy & + & im \end{pmatrix}, \tag{27}$$

где $re = -0.12$, $im = 0.74$

Описанные выше методы построения инвариантных множеств с помощью символического образа дали похожие результаты.

Метод	Кол-во узлов	Кол-во ребер	Время работы	Кол-во компонент	Кол-во шагов
Линейный метод	2 412 630	20 337 231	1 174 000 ms	4	13
Точечный метод	92	140	1 406 ms	15	15
Улучшенный точечный метод	2 164 382	14 282 466	844 844 ms	6	13
Адаптивный метод	1 068 206	12 145 512	2 058 797 ms	2	12

На этом примере точечный метод показал очень странный результат. Это можно объяснить тем, что количество точек, взятых для построения образа ячейки слишком мало. Изображения построенных цепно-рекуррентых множеств приведены на рис. 9.



Линейный метод | Адаптивный метод

Точечный метод | Улучшенный точечный метод

Рис. 9: Результаты вычислений. Отображение $z \to z^2 + r$, $r = -0.14 + 0.74i$.

7.4 Множество Жюлиа в случае параболической неподвижной точки [6]

Рассмотрим отображение на комплексной плоскости

$$f : z \to z^2 + \exp^{2\pi i \frac{1}{20}} z. \tag{28}$$

Для k-периодической точки z_0 ее орбита представляет собой множество точек $\{z_0, f(z_0), f^2(z_0), \ldots, f^{k-1}(z_0)\}$ (цикл периода k).

Определение 14 *Пусть z_0 периодическая точка периода n. Собственным значением точки x_0 для отображения f называется комплексное число*

$$\lambda = (f^n)'(z_0)$$

По правилам дифференцирования сложной функции, мы получаем, что это число одинаково для любой точки цикла.

Периодическая или неподвижная точка z_0 называется:

- **притягивающей**, если $0 < |\lambda| < 1$,
- **отталкивающей**, если $|\lambda| > 1$,
- **сверхпритягивающей**, если $\lambda = 0$,
- **нейтральной**, если $|\lambda| = 1$.

Нейтральные точки в свою очередь тоже можно классифицировать. Пусть $|\lambda| = 1$ и λ — собственное значение точки z_0 для отображения f, тогда его можно переписать в виде

$$\lambda = e^{2\pi i\alpha},\ \alpha \in [0, 1]. \tag{29}$$

Неподвижная точка z_0 называется **параболической**, если $\alpha \in Q$.

При заданных параметрах, для исследуемой системы $z_0 = 0$ является неподвижной параболической точкой. Обозначим λ — собственное число функции f в точке z_0. Поскольку $\lambda^{20} = 1$, множество Жюлиа подходит к точке $z_0 = 0$ с 20 различных направлений (между 20 лепестками). Основная проблема моделирования поведения этой системы заключается в потере точности компьютерных вычислений, в результате чего возникают лишние периодические или неподвижные точки.

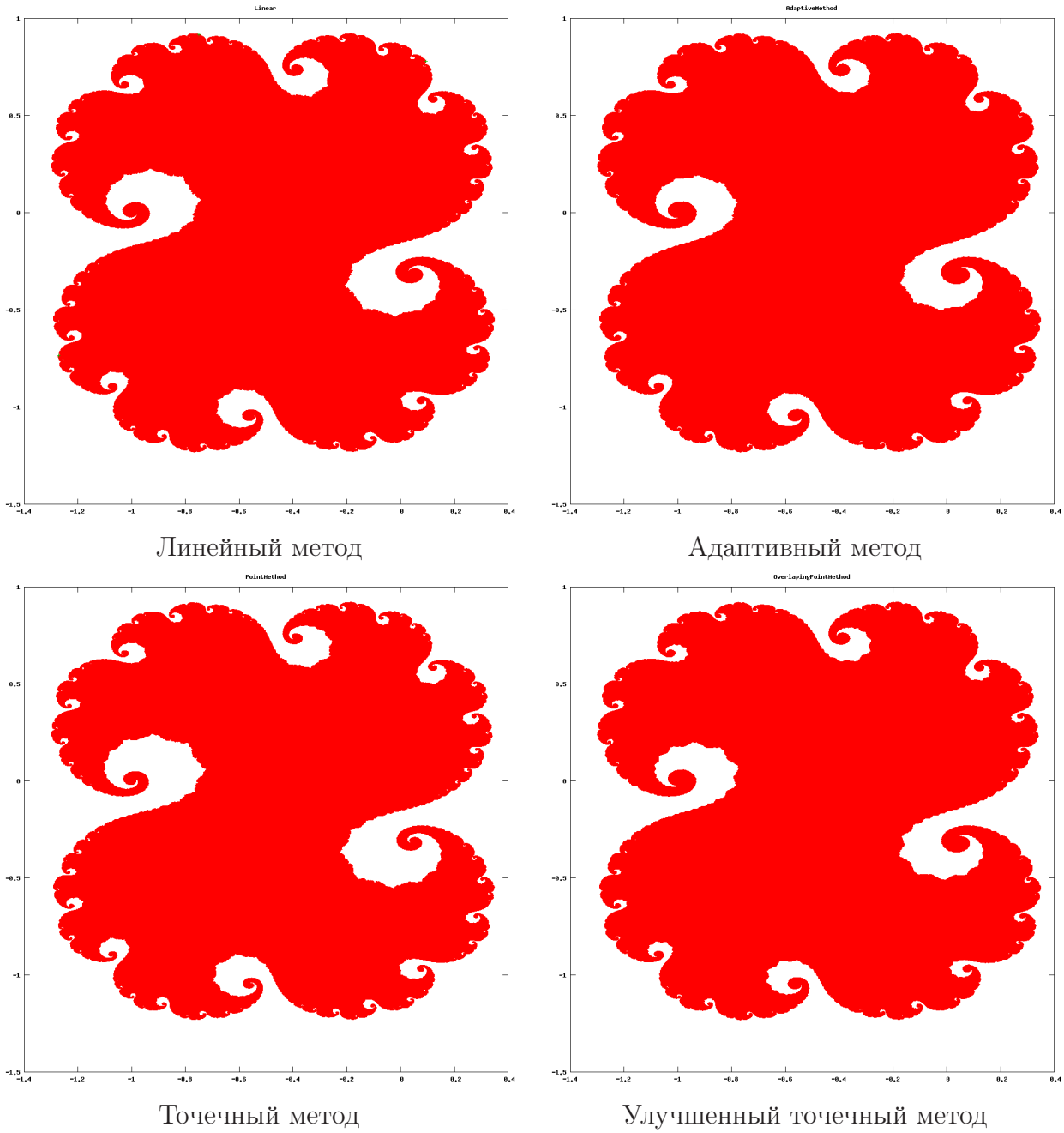
В этом примере техника символического образа для построения цепно-реккурентных множеств позволяет построить лишь наполненные множества Жюлиа, т.е. вышеописанная проблема остается.

Отметим, что в работах [2, 6] были приведены методы решения этой задачи.

Проведем 10 шагов построения символического образа (см. 3.1). Полученные результаты сведем в таблицу:

Метод	Кол-во узлов	Кол-во ребер	Время работы	Кол-во компонент	Кол-во шагов
Линейный метод	2 737 711	24 507 847	182 922 ms	2	10
Точечный метод	2 684 661	14 941 697	1 145 969 ms	1	10
Улучшенный точечный метод	2 801 326	32 547 551	4 795 844 ms	1	10
Адаптивный метод	2 814 564	35 579 838	9 043 422 ms	1	10

Все методы дали визуально схожие результаты (рис. 10). Полученные нами результаты согласуются с результатами, полученными в [2, 6].



Линейный метод | Адаптивный метод

Точечный метод | Улучшенный точечный метод

Рис. 10: Результаты вычислений в случае параболической неподвижной точки.

7.5 Двойное логистическое отображение [9]

Рассмотрим динамическую систему

$$\begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} (1-a)x & + & aby(1-y) \\ (1-a)y & + & abx(1-x) \end{pmatrix}, \tag{30}$$

где $a = 0.38$, $b = 4.11$.

Данная система, была исследована в работах ([1], [9]). Были получены оценки на область притяжения к бесконечно удаленной точке, в частности было показано, что в этой области лежит внешность круга

$$(x-r)^2+(y-r)^2 \le 2r^2, \tag{31}$$

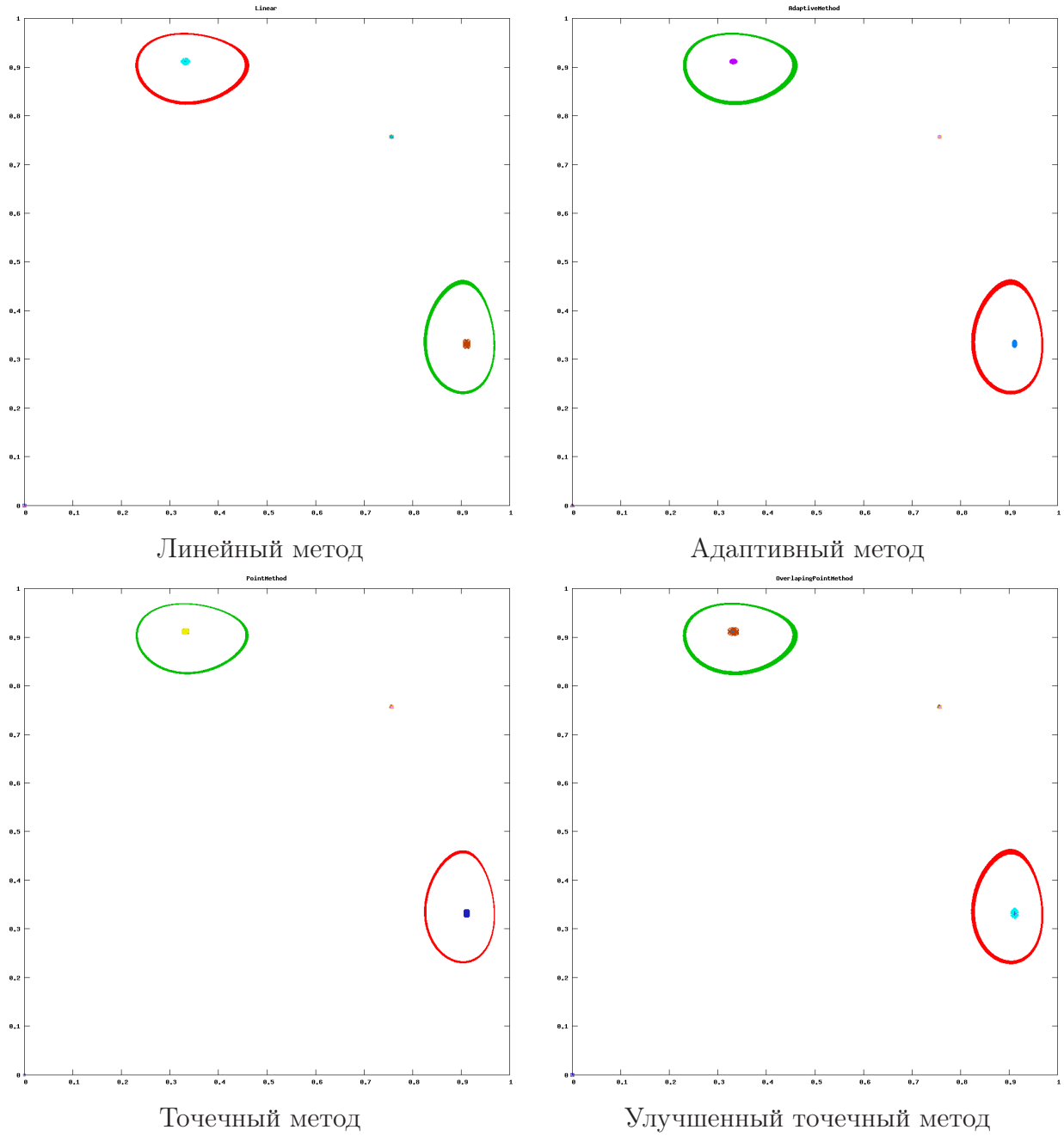
где $r=\frac{1-a+ab}{2ab}$.

При заданных значениях параметров инвариантным множеством является объединением инвариантных кривых бифуркации Хопфа в окрестности двух неподвижных точек, расположенных симметрично относительно главной диагонали.

Проведем построение символического образа и сравним полученные численные результаты.

Метод	Кол-во узлов	Кол-во ребер	Время работы	Кол-во компонент	Кол-во шагов
Линейный метод	393 478	2 715 484	16 875 ms	15	13
Точечный метод	258 874	1 108 540	57 188 ms	13	13
Улучшенный точечный метод	492 115	4 868 381	320 266 ms	13	13
Адаптивный метод	503 102	5 275 606	517 078 ms	13	13

Цепно-рекуррентые множества показаны на рис. 11. На этой системе различия методов построения образа ячейки не видны. Однако по численным показателям отличия все же есть.



Линейный метод

Адаптивный метод

Точечный метод

Улучшенный точечный метод

Рис. 11: Двойное логистическое отображение. $a = 0.38$, $b = 4.11$.

7.6 Отображение с задержкой [8]

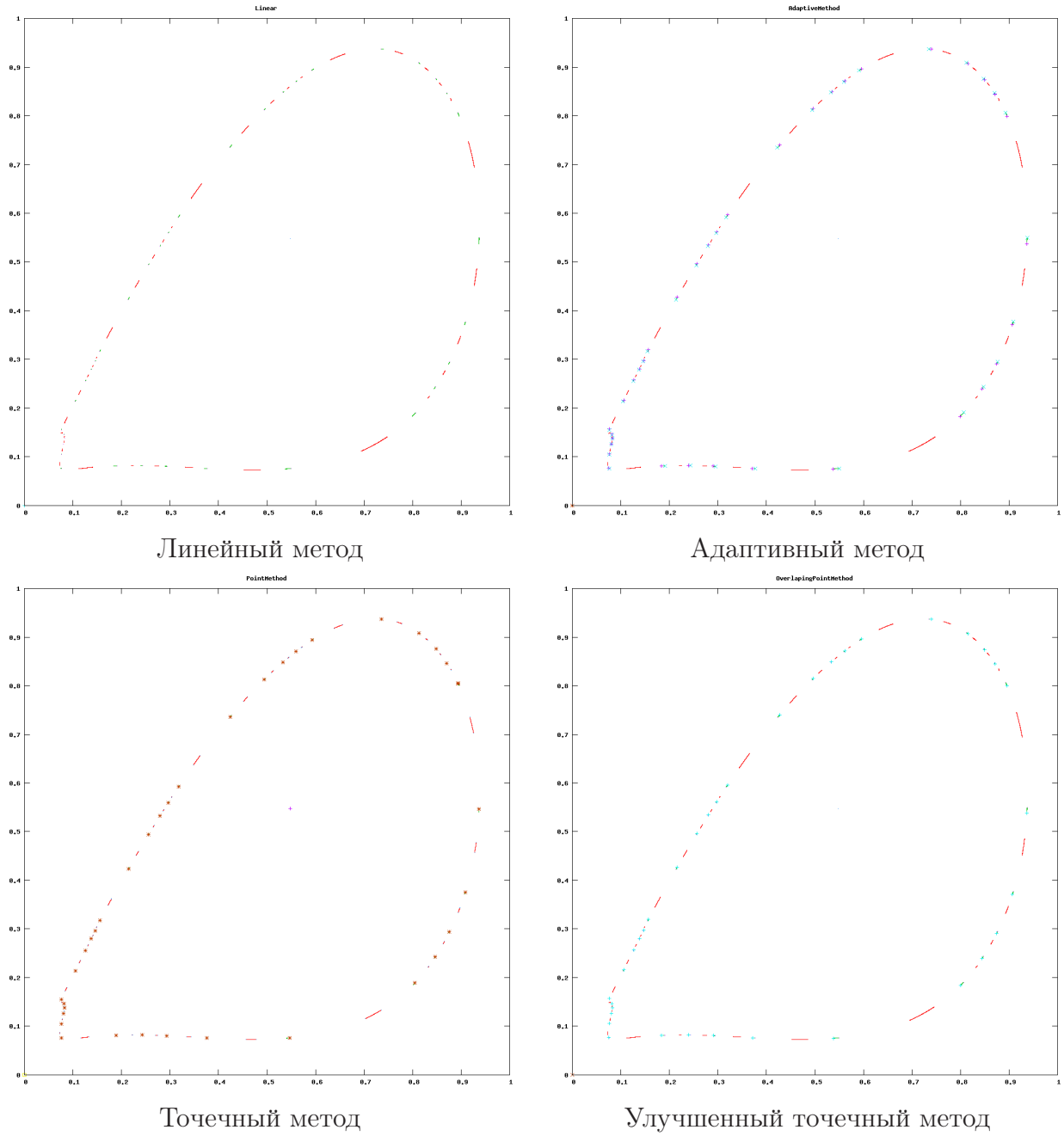
Рассмотрим динамическую систему с задержкой.

$$\begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} y \\ ay(1-x) \end{pmatrix}. \tag{32}$$

Система обладает двумя неподвижными точками $O_1 = (0, 0)$ и $O_2 = (1 - \frac{1}{a}, 1 - \frac{1}{a})$. Начало координат является седловой точкой при $a > 1$. Собственные числа в точке O_1 равны 0 и a. Точка O_2 является фокусом при $a > \frac{5}{4}$. Собственные числа в этой точке равны $\lambda_{1,2} = \frac{1}{2}(1 \pm \sqrt{5 - 4a})$. Фокус устойчив при $a < 2$, а при $a > 2$ фокус теряет устойчивость через бифуркацию Хопфа. Возникающая при этой бифуркации инвариантная кривая разрушается при $a = 2.27$ с появлением странного аттрактора [5]. Кроме того, при этом значении параметра неустойчивая сепаратриса седловой точки касается устойчивой, лежащей на ости OX.

Проведем построение цепно-рекуррентного множества при $a = 2.21$ (рис. 12). Разобьем каждый шаг построения символического образа с меньшим разбиением на два этапа. На первом этапе построения будем делить ячейки только по первой координате на две равные части. На втором — только вторые координаты. Каждый этап представляет собой отдельный шаг построения уточненного символического образа. Такой способ позволяет получить символической образ с более мелким разбиением и затратить на это меньше оперативной памяти, чем аналогичный процесс построения образа ячейки, на каждом шагу которого деление проводилось сразу по нескольким координатам.

Метод	Кол-во узлов	Кол-во ребер	Время работы	Кол-во компонент	Кол-во шагов
Линейный метод	1 489 037	12 863 578	357 859 ms	5	17
Точечный метод	378 579	954 108	311 875 ms	7	17
Улучшенный точечный метод	1 244 114	8 723 417	13 201 563 ms	6	17
Адаптивный метод	1 390 661	13 941 695	18 473 141 ms	6	17



Линейный метод | Адаптивный метод

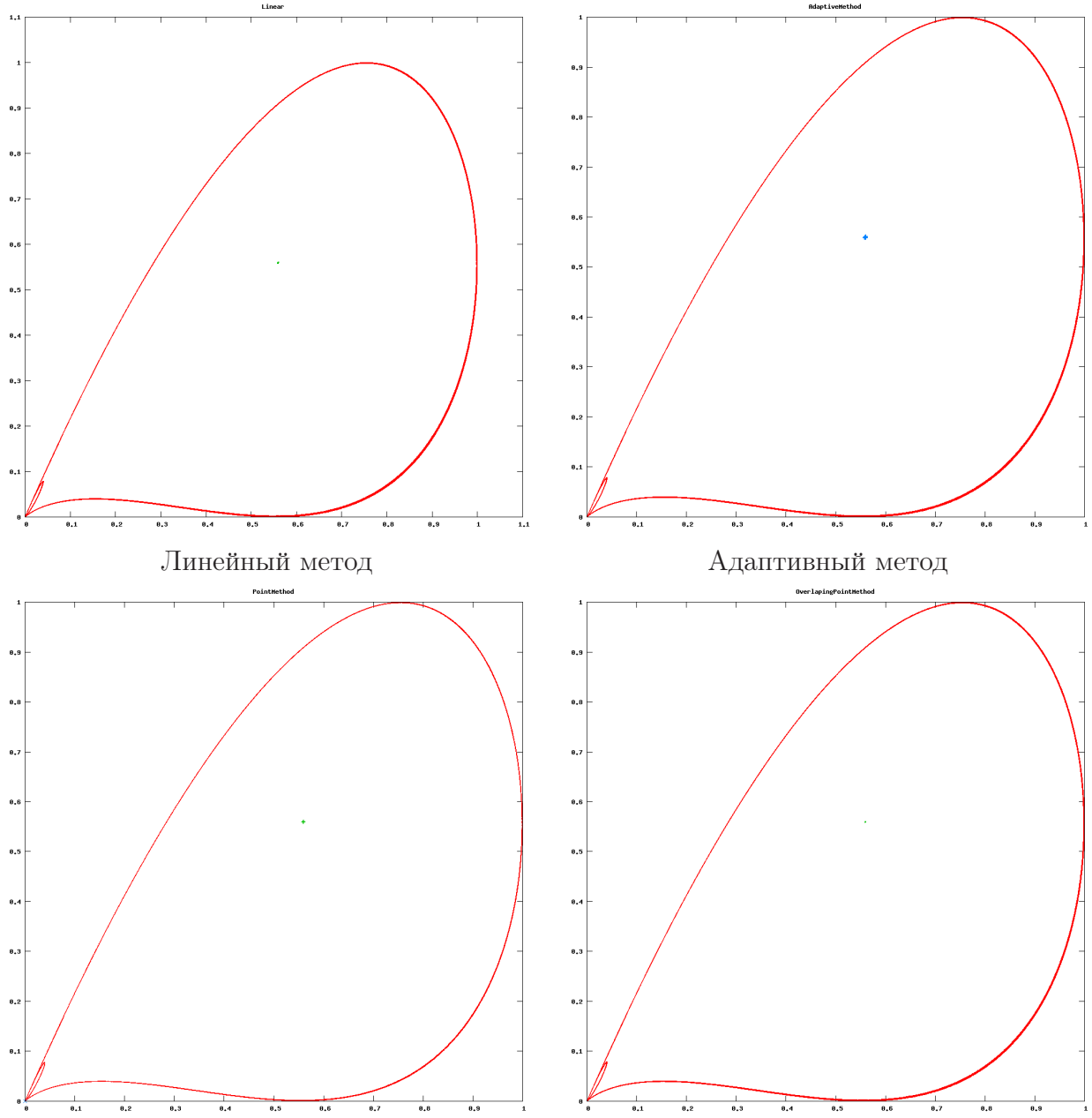
Точечный метод | Улучшенный точечный метод

Рис. 12: Отображение с задержкой, $a = 2.21$.

Проведем построение цепно-рекуррентного множества при $a = 2.27$.

Метод	Кол-во узлов	Кол-во ребер	Время работы	Кол-во компонент	Кол-во шагов
Линейный метод	422 636	3 504 285	17 297 ms	2	13
Точечный метод	185 861	450 270	28 406 ms	3	13
Улучшенный точечный метод	367 115	2 501 061	252 625 ms	2	13
Адаптивный метод	381 186	3 712 343	388 609 ms	3	13

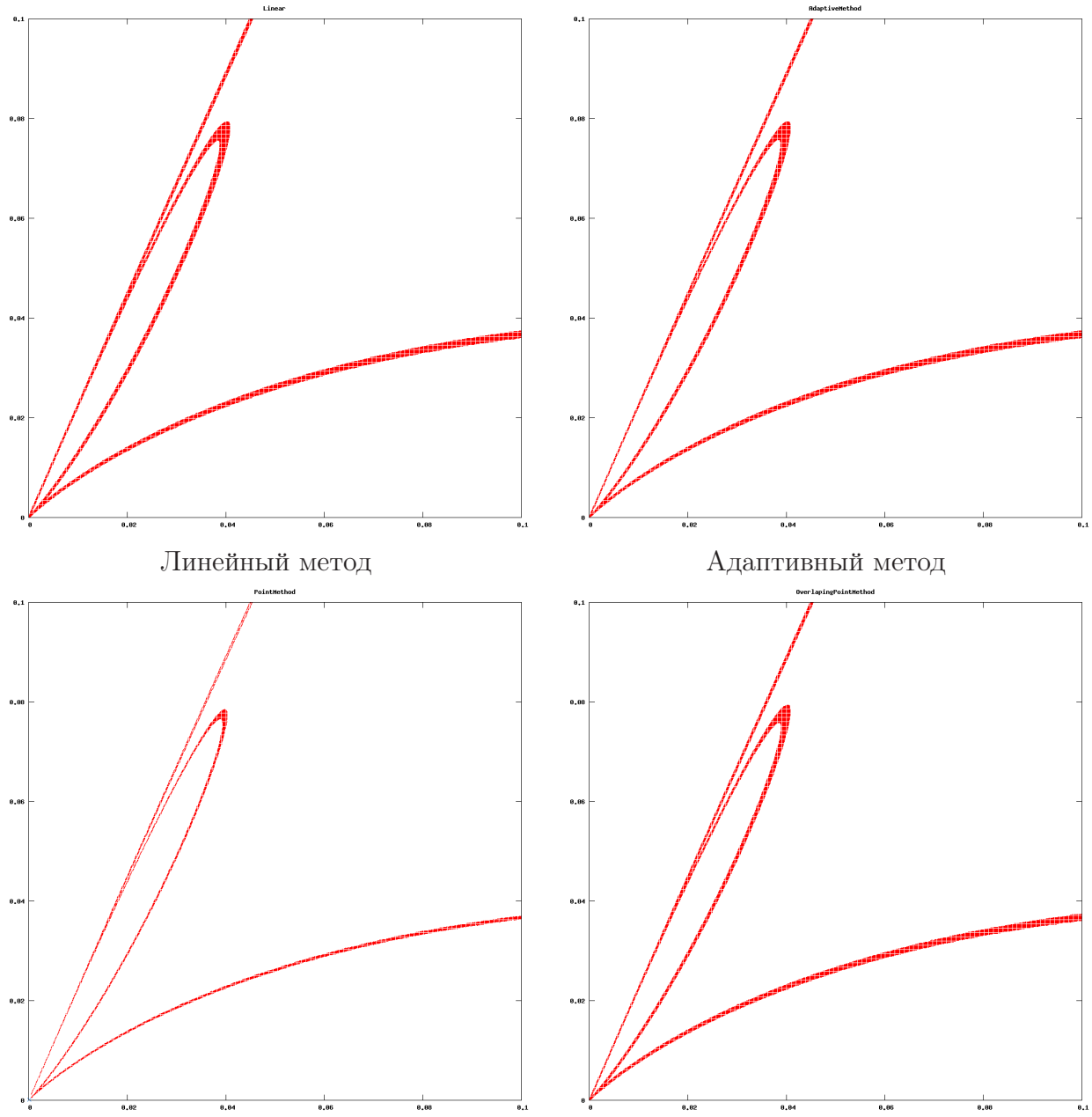
Изображение цепно-рекуррентых множеств дано на рис. 13. На рис. 14 показана область вблизи седловой точки, где возникает сложное поведение ее инвариантных многообразий. Визуально все методы построения образа ячейки работают примерно одинакого. Согласно таблице, самым быстрым методом является линейный метод, самым медленным — адаптивный.



Линейный метод | Адаптивный метод

Точечный метод | Улучшенный точечный метод

Рис. 13: Отображение с задержкой при $a = 2.27$



Линейный метод | Адаптивный метод

Точечный метод | Улучшенный точечный метод

Рис. 14: Отображение с задержкой, $a = 2.27$. Область $[0, 0.1] \times [0, 0.1]$

7.7 3х мерная модель Хищник-Жертва-Суперхищник [14]

В работах [5, 14] рассматривается следующая динамическая система

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} \frac{4.522x\exp(-y)}{1+x\max(\exp(-y),k(z)k(y))} \\ 0.962xy\exp(-z)k(4yz)k(y) \\ 4yz \end{pmatrix}, \tag{33}$$

где $k(t) = \frac{1-\exp(-t)}{t}$.

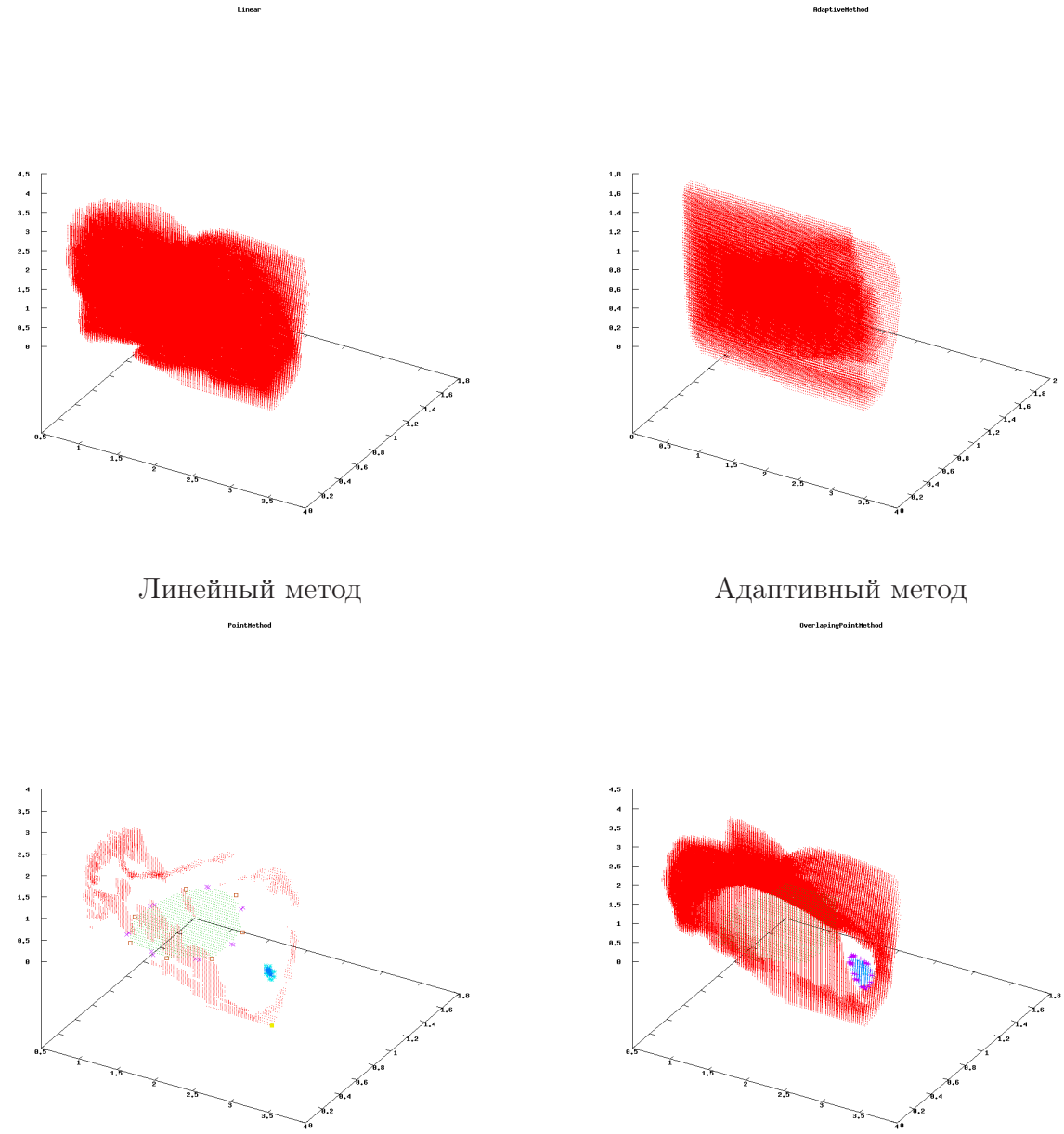
Пусть $D = [0.01, 10] \times [0.01, 10] \times [0.01, 10]$. В этом примере рассматривается трехмерная динамическая система. Для хранения ячейки требуется больше памяти, время вычисления образа одной точки тоже увеличивается. Наиболее хорошие результаты построения инвариантного множества для этой системы можно получить лишь используя все методы построения образа ячейки в совокупности. Эффективным по памяти оказывается на каждом шаге дробить ячейки только на две части (только по одной координате). В этом случае прирост ячеек в построенном графе оказывается минимальным, что позволяет уложиться в оперативную память компьютера и провести больше итераций построения символического образа, тем самым построить символический образ с меньшим разбиением.

Значение константы Липшица системы на множестве D оказывается очень большим, в результате чего адаптивный метод может работать очень медленно. Приходится вводить ограничения на количество узлов в графе представления $P^{\varepsilon}_{f,\psi}$.

Результаты построения цепно-рекуррентного множества приведены в таблице и на рис. 15.

Результаты работы каждого из методов для этого примера сильно отличаются. Видно, что быстрее всего работает точечный метод, и самым долгим является адаптивный метод, несмотря на введенные ограничения на граф (не более 100 вершин в графе адаптивного метода для ячейки).

Метод	Кол-во узлов	Кол-во ребер	Время работы	Кол-во компонент	Кол-во шагов
Линейный метод	287 627	11 399 582	30 219 ms	1	7
Точечный метод	6 744	33 956	5 297 ms	7	7
Улучшенный точечный метод	110 361	3 771 889	43 359 ms	5	7
Адаптивный метод	150 514	6 525 427	1 167 094 ms	1	7



Линейный метод

Адаптивный метод

Точечный метод

Улучшенный точечный метод

Рис. 15: Результаты вычислений. 3-х мерная модель хищник-жертва-суперхищник.

8 Заключение

Сравним средние скорости работы описанных методов для исследованных двумерных систем. Для этого для каждой из двумерных систем для каждого метода вычислим среднее количество ячеек v, ребер e, построенных за 1 миллисекунду (ms), и значение отношения числа ребер к числу ячеек $\frac{e}{v}$.

Усредним эти значения по всем исследованным системам для каждого ме-

тода ($\tilde{v}$, $\tilde{e}$ и $\widetilde{e/v}$). Где под $\tilde{v}$ понимается среднее арифметическое. По каждому из устредняемых параметров вычислим среднее квадратичное отклонение по формуле $\sqrt{\frac{\sum\limits_{i=1}^{N}(v_i-\tilde{v})^2}{N}}$, где N количество значений. Сведем полученные результаты в таблицу.

Из собранных данных видно, что самым производительным оказывается линейный метод. Самым быстрым оказывается точечный метод, но среднее количество ребер на один узел покрытия оказывается довольно маленьким. По этому показателю лидирует адаптивный метод, для которого в среднем на каждую ячейку приходится чуть более 10 ребер. Однако отношение числа ячеек к ребрам меньше всего зависит от системы, если мы применяем линейный метод.

На практике самым удобным оказывается линейный метод. Он позволяет получить достаточно хорошие результаты при минимальных затратах времени. Для приближенного рассмотрения системы удобным оказывается использовать точечный метод или улучшенный точечный метод. Они работают быстрее, но результат получается менее точны. Для некоторых систем оказывается удобным использовать адаптивный метод. Этот метод является наиболее сложным и наиболее медленным, он позволяет получить наибольшее значение отношения среднего числа ребер исходящих из каждого узла. Исходя из среднего квадратичного отклонения среднего числа исходящих ребер из каждого узла можно считать, что результат работы этого метода сильнее зависит от поведения системы, чем результат работы линейного метода.

Метод	Ср. кол-во узлов за ms $\tilde{v}$	Ср. квадр. отклонение $\sqrt{\frac{\sum_{i=1}^{N}(v_i-\tilde{v})^2}{N}}$	Время работы
Линейный метод	**15.802**	**8.6635**	1 945 484 ms
Точечный метод	5.4305	4.7180	1 696 593 ms
Улучшенный точечный метод	2.0527	2.1462	29 589 571 ms
Адаптивный метод	**0.7916**	**0.4803**	31 858 960 ms

Метод	Кол-во ребер за ms $\tilde{e}$	Ср. квадр. отклонение $\sqrt{\frac{\sum_{i=1}^{N}(e_i-\tilde{e})^2}{N}}$	Время работы
Линейный метод	**132.9332**	**73.6511**	1 945 484 ms
Точечный метод	16.0435	11.7427	1 696 593 ms
Улучшенный точечный метод	9.8190	**4.9137**	29 589 571 ms
Адаптивный метод	**8.5444**	5.2827	31 858 960 ms

Метод	Ребер на узел $\widetilde{e/v}$	Ср. квадр. отклонение $\sqrt{\frac{\sum_{i=1}^{N}(\frac{e_i}{v_i}-\widetilde{e/v})^2}{N}}$	Время работы
Линейный метод	8.4534	**0.7656**	1 945 484 ms
Точечный метод	**3.1109**	1.2832	1 696 593 ms
Улучшенный точечный метод	7.3512	**2.9207**	29 589 571 ms
Адаптивный метод	**10.8550**	0.9591	31 858 960 ms

где $N = 7$ количество рассмотренных двумерных систем.

Список литературы

[1] *Ампилова Н.Б., Осипов А.В.* Локальные бифуркации для полного отображения Гардини. Деп. ВИНИТИ 14.06.96,N 1969-В96.

[2] *Ампилова Н.Б., Петренко Е.И.* Алгоритм компьютерного моделирования множества Жюлиа в случае неподвижной параболической точки. // Электронный Журнал Дифференциальные Уравнения и Процессы Управления (http://www.neva.ru/journal), 2,2002

[3] *Осипенко Г.С.* О символическом образе динамической системы // сб. Граничные задачи, Пермь, 1983, с.101-105.

[4] *Осипенко Г.С., Романовский И.В., Ампилова Н.Б., Петренко Е.И.,* О вычисление спектра Морса // Проблемы Математического Анализа, Выпуск 27, Январь 2004, с. 151-169.

[5] *Осипенко Г.С., Ампилова Н.Б.* Введение в символический анализ динамических систем. // Изд. СПбГУ, 2005. ISBN: 5-288-03656-X

[6] *Пайтген Х.О., Рихтер П.Х.* Красота фракталов. // Образы комплексных динамических систем. М, 1993.

[7] *Писсанецки С.* Технология разреженных матриц. // М., Мир, 1988.

[8] *Aronson D.G.,Chory M.A., Hall G.R. et.all* Bifurcation from an invariant circle for two-parameter families of maps of the plane: A computer-assisted study. // Commun.Math.Phys.83,3(1982), p.303-354.

[9] *Gardini L., Abraham R., Record R.J., Fournier-Prunaret D.* A double logistic map. // Int.J.Bif.and Chaos,4,1(1994), p.145-176.

[10] *Henon M.* A two-dimensional mapping with a strange attractor. // Comm. Math.Phys. v.50,69-77(1976).

[11] *Ikeda K.* Multiple-valued stationary state and its instability of the transmitted light by a ring cavity system // Opt. Comm. 1979, Vol.30, p.257-261.

[12] *Julia G.* Sur l'iteration des fonctions rationneles. // Journal de Math.Pure at Appl. 8:47-245. 1918

[13] *Osipenko G.* Numerical Explorations of the Ikeda mapping dynamics // Electronic Journal of Differentional Equations and Control Processes (http://www.neva.ru/journal), Vol.2, 2004

[14] *Linsdrom T.* Dependencies between competition and predation — and their consequences for initial value sensitivity, // SIAM J. Appl. Math. Vol59, pp.1468-1486