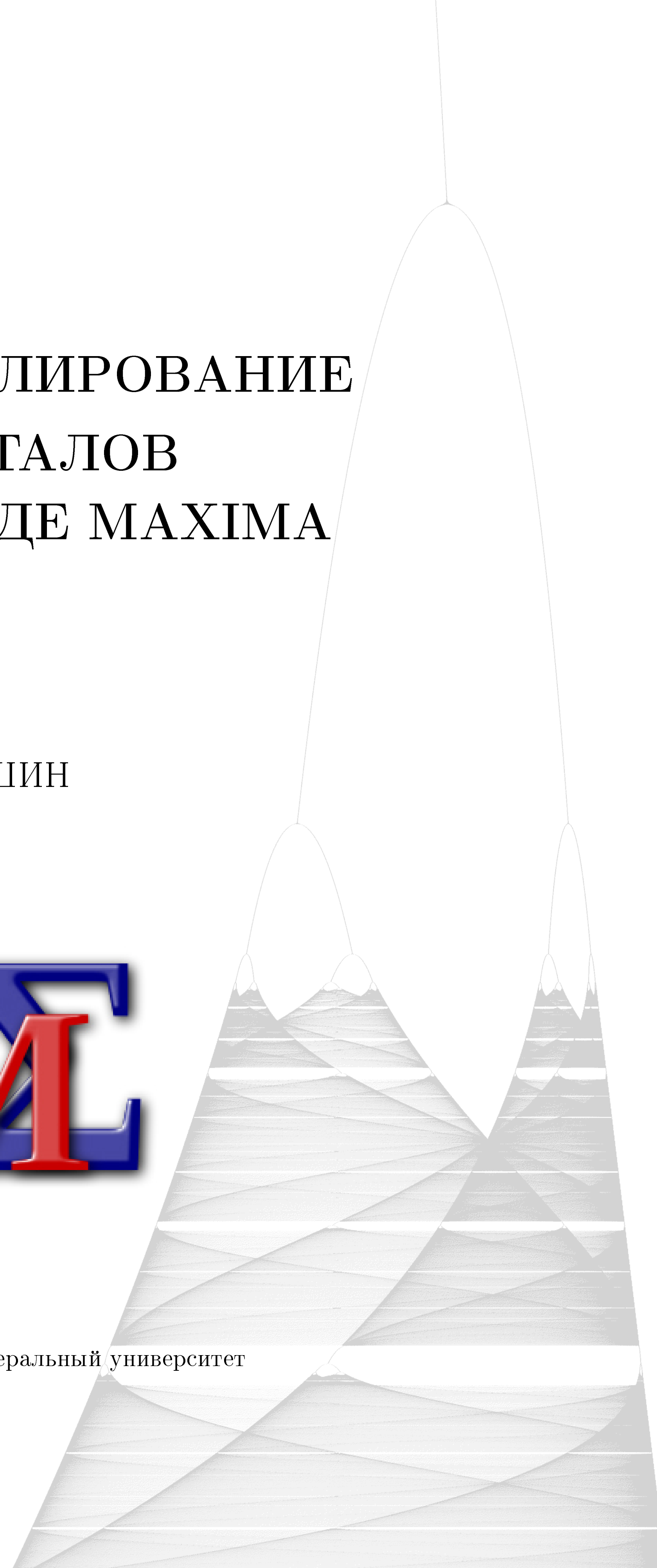
МОДЕЛИРОВАНИЕ ФРАКТАЛОВ В СРЕДЕ MAXIMA

часть II

П.И. ТРОШИН



Казанский федеральный университет

Казань 2012

Оглавление

ВВЕДЕНИЕ

Чем абстрактнее истина, которую ты хочешь преподать, тем сильнее ты должен обольстить ею ещё и чувства.

Фридрих Ницше

Красоту в математике так же трудно формально определить, как и красоту в искусстве, но люди, изучающие математику, обычно не имеют затруднений в её распознавании.

Поль Дирак

Любая формула, включенная в книгу, уменьшает число её покупателей вдвое.

Стивен Хокинг

Настоящее пособие является продолжением части I, в которой изложены методы моделирования в системе компьютерной алгебры Maxima таких фрактальных объектов, как L-системы, системы итерированных функций, множества Мандельброта и Жюлиа, аттракторы в проблеме Кэли, а также метод нахождения фрактальной размерности.

Во II части пособия излагаются методы построения паутинных и бифуркационных диаграмм, а также диаграмм орбит, построение странных аттракторов, кривых и поверхностей фрактального броуновского движения. Содержание этой части сфокусировано на динамических аспектах фрактальной геометрии, или на фрактальных аспектах динамических систем.

В качестве среды моделирования выбрана система компьютерной алгебры Maxima— свободное программное обеспечение, зарекомендовавшее себя в течение длительного времени (проект зародился в 1968 году и стал открытым с 1998 года).

Преимуществами этой системы являются: бесплатная лицензия, кросс-платформенность (реализация в ОС Windows, Linux, Unix, Mac OS X, BSD), открытый код (что понижает количество встроенных ошибок), поддержка языка программирования

Lisp и др.. Maxima выполняет символьные и численные вычисления, строит различные графики, обладает широким набором дополнительных программных пакетов.

Максима имеет несколько графических интерфейсов: XMaxima, wxMaxima и др., а также может работать в режиме командной строки, используя псевдографику. Предполагается, что читатель знаком с основами языка Maxima.

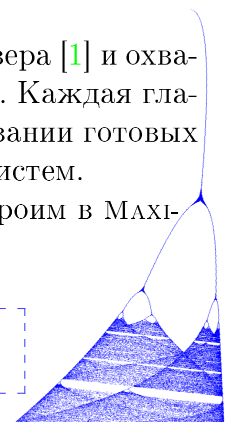
Ввиду сделанного выбора данное пособие может использоваться при изучения применения пакета Maxima в курсе высшей математики. Однако все изложенные алгоритмы читатель легко сможет адаптировать для других языков математических вычислений.

Заметим, что хотя данное пособие посвящено построению фракталов, мы не будем давать строгое определение фрактальным множествам и фрактальной размерности, чтобы избежать однобокого или же, наоборот, слишком пространного и техничного определения. Предполагается, что читатель знаком с широким значением понятия, которое мы будем просто называть «фрактал».

Основное содержание пособия близко по изложению книге Р.М. Кроновера [1] и охватывает широко известные алгоритмы построения фрактальных множеств. Каждая глава снабжена списком Задач. В Приложении даются сведения об использовании готовых специализированных пакетов для изучения фракталов и динамических систем.

Первое множество, связанное с областью хаотической динамики, построим в Maxima прямо на этой странице[1]:

```
load(dynamics)$
orbits(x^2+c,0,50,200,[c,-2,0.25],[style,dots],[nticks,1000]);
```

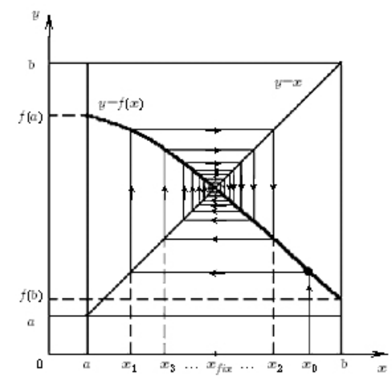


Это изображение, напоминающее заснеженные горные вершины, — бифуркационная диаграмма (или диаграмма орбит), представляющая собой иллюстрацию возникновения хаоса методом удвоения периода. С этим графиком связана знаменитая универсальная константа Фейгенбаума.

Рекомендуемая литература:
официальная страница Maxima: [3e],
установочный файл Maxima: [4e],
руководства пользователя Maxima: [8e], [9e], [6e].

[1]Здесь и далее мы не будем ставить точки после формул кода в конце предложений, так как это несомненно вызовет ошибку при их компиляции.

Глава 1



НЕПОДВИЖНЫЕ ТОЧКИ, ОРБИТЫ, БИФУРКАЦИИ

Земле мы, конечно, припишем вид куба: ведь из всех четырёх родов наиболее неподвижна и пригодна к образованию тел именно земля, а потому ей необходимо иметь самые устойчивые основания. ***Платон***

Хаотизация поведения большого числа нелинейных систем количественно описывается универсальными числами $\delta = 4,6692016\ldots$ и $\alpha = 2,502907875\ldots$
Митчелл Фейгенбаум

В этой главе мы изучаем качественное поведение некоторых дискретных динамических систем, знакомясь с различными видами их представления: график орбиты, паутинная и бифуркационная диаграмма, — а также с такими понятиями как: периодические точки (и их характер), точки бифуркации, константа Фейгенбаума, унимодальные отображения и «переход к хаосу с помощью удвоения периода».

1.1 Паутинные диаграммы

Неподвижной точкой отображения (функции) $f\colon \mathbb{R} \to \mathbb{R}$ называется точка $x_f \in \mathbb{R}$, для которой $f(x_f) = x_f$. Неподвижные точки отображений имеют большое значение в различных областях математики. Например, известный метод Ньютона нахождения нулей функции опирается именно на теорию неподвижной точки.

Орбитой точки $x \in \mathbb{R}$ называется последовательность точек

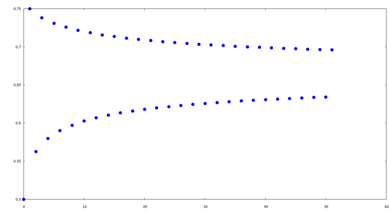
$$x, f(x), f(f(x)), \ldots, f^{(n)}(x), \ldots \tag{1.1}$$

Орбиту точки можно изобразить на графике как последовательность точек $(0, x_0), (1, x_1), \ldots, (n, x_n)$, где $x_{k+1} = f(x_k)$. Последовательность $\{x_k\}$ можно задать так:

пример
```
f(x):=3*x*(1-x)$
x[0]:0.5$
for i:0 thru 50 do x[i+1]:f(x[i])$
```

Отобразим данную последовательность на графике:

пример
```
orb:makelist([i,x[i]],i,0,50))$
plot2d([discrete,orb],[style,points])$
```



Комментарий. Последнюю строчку можно заменить на
```
load(draw)$
draw2d(point_type=filled_circle,points(orb))$
```

Гораздо проще для аналогичного представления орбит использовать команду `evolution`[1] пакета `dynamics`:

пример
```
load(dynamics)$
evolution(3*x*(1-x),0.5,50)$
```

Неподвижная точка x_f называется *притягивающей (аттрактором)*[2], если существует её окрестность $U(x_f)$, для любой точки $x \in U(x_f)$ которой $\lim\limits_{n\to\infty} f^{(n)}(x) = x_f$. Если функция f дифференцируема, то в притягивающей неподвижной точке $f(x_f) = x_f, |f'(x_f)| < 1$.

Неподвижная точка называется *отталкивающей (репеллером)*, если для любой точки $x \in \breve{U}(x_f)$ любой её проколотой окрестности $f^{(n)}(x) \not\to x_f$. Если функция f дифференцируема, то в притягивающей неподвижной точке $f(x_f) = x_f, |f'(x_f)| > 1$.

В Maxima неподвижную точку отображения $f(x)$ на отрезке $[a, b]$ можно искать командой `find_root(f(x)-x,a,b)`, в которой возможно указать точность вычисления:
```
find_root(f(x)-x,a,b,abserr=0.0001,relerr=0.00001)
```

Пакет `newton1` позволяет находить неподвижные точки в окрестности данной точки x_0 методом Ньютона с точностью `eps` (то есть $f(x) \approx x$ с погрешностью не больше `eps`):
```
load (newton1)$
newton(f(x)-x,x,x_0,eps)
```

Сосчитаем модуль производной функции $f(x)$ для нахождения типа неподвижной точки $x_f = a$:
```
at(abs(diff(f(x),x)),[x=a])
```

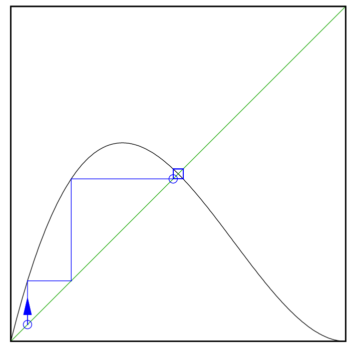
Комментарий.

[1] Подробнее об этой команде см. Приложение, с. 39.

[2] Неподвижная точка — частный случай точки периода $p \in \mathbb{N}$, см. часть I, параграф 4.1.

`diff(`f`(x),x)` — возвращает производную функции f по переменной `x`;
`diff(`f`(x),x,k)` — возвращает производную `k`-го порядка функции f по переменной `x`;
`at(`$g(x)$`,[x=a])` — возвращает значение выражения $g(x)$ в точке `x=a`.

Неподвижные точки находятся методом итераций $f^{(n)}(x_0) \to x_f$. Процесс сходимости (или же, наоборот, расходимости, а также более сложные типы поведения данной динамической системы[3]) удобно изображать графически с помощью «паутинных диаграмм». *Паутинная диаграмма* — это отображение в одной системе координат графика функции $f(x)$, диагонали $y = x$ и представляющей итерационный процесс ломаной линии с вершинами в точках



$$(x_0, x_0), (x_0, x_1), (x_1, x_1), (x_1, x_2), \ldots,$$

$$x_n = f(x_{n-1}), n = 1, 2, \ldots$$

Алгоритм 1.1: **Построение паутинной диаграммы**

Вход: $f(z)$; ✎ *функция, для которой ищется неподвижная точка*
$[a, b]$; ✎ *область определения функции $f(x)$*
x_0; ✎ *начальная точка для итераций из отрезка $[a, b]$*
$level$; ✎ *количество итераций отображения $f(x)$*

Выход: паутинная диаграмма: изображение ломаной итераций, графика функции и диагонали $y = x$.

1: $path = \{x_0, x_0\}$; ✎ *инициализация массива точек-вершин ломаной*
2: **повторять** $level$ **раз**
3: $y_0 = f(x_0)$;
4: $path = path \cup \{\{x_0, y_0\}, \{y_0, y_0\}\}$;
5: $x_0 = y_0$;
6: отобразить ломаную с вершинами из массива $path$, график функции $f(x)$ и диагональ $y = x$ в одной системе координат на интервале $[a, b]$.

Программируем на Maxima (листинг № 1.1):

Построение паутинной диаграммы

```
load(draw)$
[a,b]: [0,1]$                         /* левая и правая границы: a<=x<=b */
x0: 0.05$                             /* начальная точка */
level: 10$                            /* число итераций */
f(x):=4*x(1-x)$                       /* исходная фунция */

path:[[x0,x0]]$                       /* массив вершин ломаной */
```

[3]Т.е. процесс $x_n = f(x_{n-1}), n = 1, 2, \ldots$, где начальное приближение x_0 играет роль управляющего параметра.

```
thru level do(                                      /* построение ломаной path */
    y0: f(x0),
    path: append(path,[[x0,y0],[y0,y0]]),
    x0: y0
)$
draw2d(
    color=black,
    explicit(f(x),x,a,b),                              /* отображаем функцию f */
    color="#23AB0F",
    explicit(x,x,a,b),                             /* отображаем диагональ y=x */
    color=blue,
                               /* отображаем концевые точки ломаной окружностями: */
    point_type=circle,point_size=1, points([path[1],last(path)]),
    point_type=dot,points_joined=true,points(path),       /* отображаем ломаную */
          /* отображаем вектор направления итераций вдоль первого ребра ломаной */
    head_length = 0.05, head_angle=13, vector(path[1],0.6*(path[2]-path[1])),
    user_preamble="set size ratio -1"
)$
```

Комментарий.

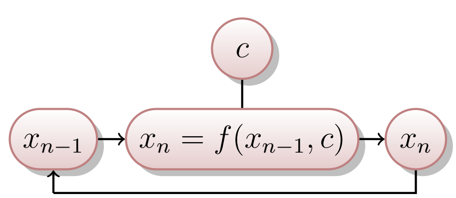
> Замечания об использованных командах:
>
> `color=black` — определение цвета для графики (например: blue, red, green, magenta, black, cyan). В пакете `draw` есть 83 именных цвета (см. в справке по Maxima), другие цвета можно определять в шестнадцатеричной системе счисления в формате $\#RRGGBB$: `color="#23AB0F"`;
>
> `explicit(`$f(x)$`,x,a,b)` — график явно заданной функции $f(x)$, $x \in [\mathtt{a}, \mathtt{b}]$;

1.2 Бифуркационная диаграмма

В математике, особенно при изучении динамических систем, под понятием *бифуркационная диаграмма*[4] подразумевают изображение на рисунке смены возможных динамических режимов системы (равновесных состояний, неподвижных точек, периодических орбит и пр.) при изменении значения бифуркационного параметра.

Рассмотрим динамическую систему с параметром $c \in C$:

$$\begin{gathered} f\colon U \times C \to U, \quad U, C \subset \mathbb{R}, \\ x_n = f(x_{n-1}, c), \quad n = 1, 2, \ldots, \quad x_0 \in U. \end{gathered} \tag{1.2}$$

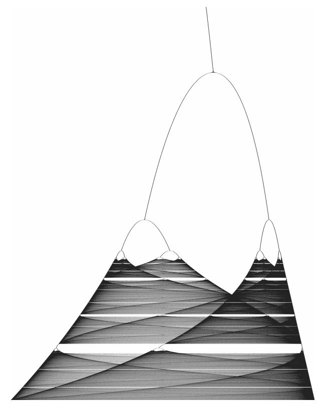


Требуется изобразить зависимость наличия стабильных периодических точек отображения f от изменения параметра c. Будем отмечать по оси ординат параметр c, а по оси абсцисс — соответствующие этому значению параметра стабильные периодические точки.

[4]Иногда её называют диаграммой орбит.

Находить аналитически[5] и исследовать на стабильность характер периодических точек достаточно трудоемко: нужно решать уравнения $f^{(n)}(x,c) = x$, $n = 0, 1, \ldots$, а затем исследовать $\left|\frac{d}{dx} f^{(n)}(x,c)\right|$.

Наиболее эффективным и простым способом построения такой диаграммы является следующий. Разбив область U на сетку значений, для каждого из них запустим итерационный процесс (1.2) (с произвольной, но не отталкивающей начальной точкой) и будем отрисовывать N точек орбиты, откинув первые $skip$ значений[6]. При сравнительно большом $skip$ наша орбита притянется к стабильной периодической орбите. Тогда на рисунке будет видно, группируются ли точки в кластеры, соответствующие стабильным (притягивающим) периодическим точкам (маловероятно появление на графике нестабильных периодических точек, почему?). Таким образом, вне хаотических областей параметра c ветви бифуркационной диаграммы состоят из притягивающих точек x периода 2^n ($n \in \mathbb{N}$): $|\frac{d}{dx} f^{(2^n)}(x,c)| < 1$.



Алгоритм 1.2: **Построение бифуркационной диаграммы**

Вход: $f(x,c)$; ✎ *итерируемая функция*
$[c_1, c_2]$; ✎ *область изменения параметра с*
$prec$; ✎ *количество точек для деления отрезка* $[c_1, c_2]$
N; ✎ *количество отображаемых точек при каждом с*
$skip$; ✎ *количество пропущенных начальных точек при каждом с*
x_0; ✎ *произвольная точка из области определения функции* $f(x)$

Выход: изображение бифуркационной диаграммы.

$path = \emptyset$; ✎ *инициализация массива точек графика*
$c = c_1$; ✎ *значение с будет меняться от* c_1 *до* c_2
повторять $prec$ **раз**
  $c = c + \frac{c_2 - c_1}{prec}$;
  повторять $skip$ **раз**
    $x_0 = f(x_0, c)$;
  повторять N **раз**
    $x_0 = f(x_0, c)$;
    $path = path \cup \{x_0, c\}$
отобразить точки из массива $path$ на дисплее.

Программируем на Maxima (листинг № 1.2):

Построение бифуркационной диаграммы (для $f(x) = cx(1-x)$)

```
load(draw)$
[c1,c2]: [-2,-0.7]$          /* левая и правая границы: c1<=c<=c2 */
f(x):=c*x*(1-x)$                            /* исходная фунция */

```

[5]Например, для отображения $f(x) = 4x(1-x)$ есть простая формула $x_n = \sin^2(2^n \pi\theta)$ ($\theta = \frac{1}{\pi}\sin^{-1}\sqrt{x_0}$) [2, с. 181]; а для отображения $f(x) = 2x(1-x)$ — формула $x_n = \frac{1}{2} - \frac{1}{2}(1-2x_0)^{2^n}$.

[6]В известной книге [3] для создания иллюстраций используются значения $skip = 5000$ и $N = 120$.

```
prec: 150$                                              /* число точек деления c */
N:120$                                     /* число отображаемых точек при каждом c */
skip:30$                               /* число пропущенных первых точек при каждом c */

x0: 0.4$                                                      /* начальная точка */
path:[]$                                                   /* массив точек графика */
c:c1$

thru prec do(
   c:c+(c2-c1)/prec,
   thru skip do x0: f(x0),                           /* пропускаем первые skip точек */
   thru N do(                               /* добавляем в массив последующие N точек */
      x0: f(x0),
      path: endcons([x0,c],path)
   )
)$

draw2d(point_type=dot,points(path),user_preamble="set size ratio -1")$
```

Самым известным и первым примером динамической системы, обладающей загадочной бифуркационной диаграммой[7], является логистическое[8] отображение

$$f\colon [0,1] \ni x \mapsto cx(1-x) \in [0,1], \qquad c \in [0,4]. \tag{1.3}$$

Его часто приводят в пример того, как из очень простых нелинейных уравнений может возникать сложное, хаотическое поведение. Это отображение популяризовал биолог Роберт Мэй (1976), хотя впервые его аналог был рассмотрен Пьером Франсуа Ферхюльстом[9] (1838). Динамическая модель $x_n = cx_{n-1}(1-x_{n-1})$ описывает численность биологической популяции в дискретные моменты времени: $x_n \in [0,1]$ представляет относительный размер популяции в год n, множитель c — это скорость роста популяции (с учетом её репродукции и гибели в результате перенаселения и нехватки ресурсов).

1.3 Универсальность Фейгенбаума

Митчелл Фейгенбаум[10] [8, 9] описал механизм «получения хаоса с помощью удвоения периода» и заметил, что он возникает не только при итерациях логистического отображения, но и для других двузначных отображений интервала в себя, таких как $x^2 + c$, $c \sin \pi x$, $cx^2 \sin \pi x$.

---

[7] Бифуркационная диаграмма для логистического отображения называется диаграммой Фейгенбаума.

[8] Его также называют логистической параболой, а также отображением Ферхюльста. П.Ф. Ферхюльст ввел название *логистическая функция* для решение уравнения $\dot f(t) = cf(t)(1-f(t))$. Иногда логистическим отображением называют отображение $f(x) = 1 - cx^2$, обладающее схожими свойствами.

[9] На самом деле, логистическое уравнение было рассмотрено Бенджамином Гомпертцем в 1825 году.

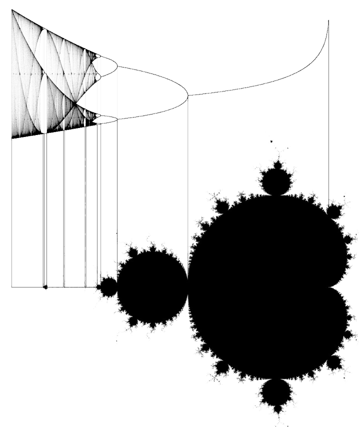
[10] Одновременно и независимо от него это наблюдение и его объяснение сделали П. Кулле и Ч. Трессер [4, 5]. Одно из строгих обоснований такого поведения было получено с применением компьютерных вычислений О. Лэнфордом [6]. Обзор этой темы и доказательство универсальности см. также в статье М. Любича [7].

Обозначим через $c_0, c_1, \ldots$ точки бифуркации на бифуркационной диаграмме, то есть точки c_n, в которых итерации $f(x,c)$ сменяют притягивающую орбиту периода 2^{n-1} на притягивающую орбиту периода 2^n.

М. Фейгенбаум заметил, что для вышеупомянутых отображений существует конечный предел[11] c_∞, $c_n \to c_\infty$, иногда называемый *точкой Фейгенбаума*. При этом при $c \to c_\infty - 0$ происходит удвоение периода, а участок $c > c_\infty$ называется областью хаоса. Он также заметил, что отношение длин последовательных интервалов между точками бифуркаций имеет «универсальный» предел[12], называемый *константой Фейгенбаума*[13]:

$$\delta = \lim_{n\to\infty} \frac{c_n - c_{n-1}}{c_{n+1} - c_n} = 4{,}66920160910299067185320382046620161725818557747$$
$$576863274565134300413433021131473713868974402394801381716\ldots$$

Этот предел характеризует скорость перехода динамической системы, испытывающей удвоение периода, к хаотическому поведению.



Оказалось, что одинаковое поведение: области удвоения на диаграмме орбит, один и тот же предел δ, «хаотическое» поведение — имеют многие унимодальные (одногорбые) отображения интервала в себя, и константа δ помогает предсказывать наступление «хаоса». Известна связь константы Фейгенбаума и множества Мандельброта: диаметры больших окружностей на вещественной оси равны периодам удвоения c_n для отображения $f(x,c) = c - x^2$, и δ совпадает с пределом отношения этих убывающих диаметров.

В настоящее время не известны необходимые и достаточные критерии, позволяющие утверждать, что произвольное семейство отображений $f(x,c)$ обладает вышеупомянутыми свойствами. Однако частичные ответы, среди каких функций искать такие отображения, дают следующая теорема Давида Зингера и условия соответствующих теорем из [10, 8].

Д. Зингер [11] впервые использовал *производную Шварца* для исследования динамики отображений отрезков и обратил внимание на важность условия отрицательного *шварциана*

$$Sf(x) \equiv \frac{f'''(x)}{f'(x)} - \frac{3}{2}\left(\frac{f''(x)}{f'(x)}\right)^2.$$

Сосчитаем его в Maxima:

```
Sf(x):=block([u],
      at(diff(f(u),u,3)/diff(f(u),u)-1.5*(diff(f(u),u,3)/diff(f(u),u,2))^2,
         [u=x])
           )$
```

[11] Свой для каждого отображения f.

[12] Единый для многих таких отображений f.

[13] См. [10e], [11e]. Ср. со значением из книги [1, гл. 6, с. 162], точным лишь до порядка 10^{-4}.

Комментарий.

`block([`$x_1, \ldots, x_n$`],`$f_1, \ldots, f_k$`)` — выполняет команды $f_1, \ldots, f_k$ и возвращает значение последнего выражения f_k, считая $x_1, \ldots, x_n$ локальными переменными. Заметим, что желательно вводить локальные переменные при определении функций. Подумайте, почему?

Теорема 1.1 (Д. Зингер, см. [11], [2, Appendix D]). *Пусть* $f\colon [0,1] \to [0,1]$, $f \in C^3$, *причём*

- $f(0) = f(1) = 0$;
- *существует единственная критическая точка* $x_0 \in [0,1]$: $f'(x_0) = 0$;
- $Sf(x) < 0 \qquad \forall x \in [0,1]$.

Тогда в $(0,1)$ *существует не более одной стабильной периодической орбиты для* f. *Если такая орбита существует, то ее точки — предельные для орбиты критической точки* x_0.

Итак, динамические системы, обладающие описанными универсальными свойствами диаграмм удвоения периода, следует искать среди унимодальных (одногорбых) отображений $f(x,c)$ (т.е. $f\ :\ [a,b] \ni x \mapsto f(x,c) \in [a,b]$ — непрерывно, имеет единственный максимум в точке $x_0 \in (a,b)$ и $f(a) = f(b)$). Кроме того, следует обратить внимание на отрицательность шварциана.

Каким образом можно вычислять точки бифуркации c_n? Аналитическое нахождение c_n связано с вычислением критических точек отображений $f^{(n)}(x,c)$, что в большинстве случаев практически невыполнимо. Экспериментально эти значения можно найти при построении бифуркационной диаграммы. Однако наша цель — подсчитать константу δ, и это можно сделать без нахождения c_n. Рассмотрим *прямой* способ вычисления константы δ [1, гл. 6]. Нам понадобится следующая лемма.

Лемма 1.1. *Пусть* $x_0 = x_0(c)$ *—единственная критическая точка отображения* $f(x,c)$ *и* x *— сверхпритягивающая точка периода* n. *Тогда* x *принадлежит орбите* x_0. *В частности,* x_0 *— тоже сверхпритягивающая точка периода* n.

⟦ Доказательство. Пусть x — сверхпритягивающая точка периода n, тогда

$$0 = \frac{d}{dx} f^{(n)}(x,c) = f'_x(f^{(n-1)}(x,c),c) \cdot f'_x(f^{(n-2)}(x,c),c) \cdot \ldots \cdot f'_x(x,c),$$

откуда, ввиду единственности критической точки x_0, определяемой условием $f'_x(x_0,c) = 0$, получаем, что при некотором k $f^{(k)}(x,c) = x_0$, откуда x принадлежит орбите x_0.

Можно легко показать, что *если одна точка орбиты периода* n *является сверхпритягивающей, притягивающей или отталкивающей, то этим же свойством обладают все точки орбиты*, в частности, x_0 — сверхпритягивающая точка периода n. ⟧

Между каждой парой точек бифуркации c_n и c_{n+1} существует точка c_n^*, обладающая сверхпритягивающей орбитой[14] периода 2^n. Искать c_n^* проще, чем c_n. Для этого надо

[14]Т.е. среди всех притягивающих точек x при $c \in (c_n, c_{n+1})$ найдется значение c_n^* такое, что точка x — сверхпритягивающая периода 2^n (в силу Леммы 1.1 при этом значении c_n^* можно положить $x = x_0$). Это утверждение, как и формула (1.4), требуют доказательства.

решить систему уравнений:

$$\begin{cases} f'_x(x_0, c_n^*) = 0, & (x_0 \text{ — сверхпритягивающая точка отображения } f(x, c_n^*)) \\ f^{(2^n)}(x_0, c_n^*) = x_0 & (x_0 \text{ — точка периода } 2^n \text{ для отображения } f(x, c_n^*)). \end{cases}$$

Можно доказать, что

$$\delta = \lim_{n\to\infty} \frac{c_n^* - c_{n-1}^*}{c_{n+1}^* - c_n^*}, \qquad (1.4)$$

что и является прямым способом нахождения δ (в отличии от способа ренормализации[15]).

Алгоритм 1.3: **Нахождение константы Фейгенбаума (в общих чертах)**

Вход: $f(x, c)$; ✎ *функции с параметром c, для которых ищется константа Фейгенбаума*
n; ✎ *количество точек c_i^* для подсчёта предела δ*
Выход: массив значений $\{\delta_i\}_{i=1}^n$, $\delta \approx \delta_n$.

1: x_0; ✎ *находим единственную критическую точку для $f(x)$: $f'_x(x_0, c) = 0$*
2: $\{c_0^*, c_1^*, \delta_1\} = \{0, 1, 3.2\}$; ✎ *начальные приближения для вычислений*
3: **для** $i = 2, \ldots, n$
4: c_i^*; ✎ *находим корень c уравнения $f^{(2^i)}(x_0, c) = x_0$. Для этого можно воспользоваться методом Ньютона с начальным приближением $c = c_{i-1}^* + \frac{c_{i-1}^* - c_{i-2}^*}{\delta_{i-1}}$. Поскольку у функции $f^{(2^i)}$ имеется много плотно расположенных нулей c, угадывание значения начального приближения с помощью приближённого значения $\delta \approx \delta_{i-1}$ существенно влияет на точность вычислений*
5: $\delta_i = \frac{c_{i-1}^* - c_{i-2}^*}{c_i^* - c_{i-1}^*}$;

В приведённом Алгоритме 1.3 особого внимания заслуживает строка 4. Для решения уравнения $f^{(2^i)}(x_0, c) = x_0$ относительно переменной c можно воспользоваться средствами Maxima[16]. Здесь сложность решения заключается в том, что степень[17] уравнения экспоненциально растет, например, при $f(x, c) = c - x^2$ и $i = 4$ ($x_0 = 0$) получаем уравнение

$$c-(c-(c-(c-(c-(c-(c-(c-(c-(c-(c-(c-(c-(c-(c-c^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2)^2 = 0.$$

Чтобы взять ситуацию в свои руки, опишем, как можно эффективно реализовать метод Ньютона в данном случае. Введем две последовательности функций (или выражений):

$$\text{для } k \geq 1 \qquad B_k(c) = f(B_{k-1}(c), c), \qquad B_0(c) = x_0;$$
$$D_k(c) = B_k(c)'_c, \quad D_0(c) = 0.$$

Тогда

$$B_k(c) = f^{(k)}(x_0, c) \qquad \text{и}$$
$$D_k(c) = B_k(c)'_c = f'_x(B_{k-1}(c), c) D_{k-1}(c) + f'_c(B_{k-1}(c), c).$$

[15] Этот способ прост, но не является эффективным, в виду экспоненциально растущего объёма вычислений. Более эффективный способ использует теорию ренормализации, см. [12, 13].

[16] Напомним, в Maxima есть команды `newton` и `find_root`, см. с. 6.

[17] В случае, если f — полином.

Запишем метод Ньютона для нашего уравнения $B_{2^i}(c) - x_0 = 0$:

$$\tilde{c}_n = \tilde{c}_{n-1} - \frac{B_{2^i}(\tilde{c}_{n-1}) - x_0}{(B_{2^i}(\tilde{c}_{n-1}) - x_0)'_c} = \tilde{c}_{n-1} - \frac{B_{2^i}(\tilde{c}_{n-1}) - x_0}{{B_{2^i}}'_c(\tilde{c}_{n-1})} = \tilde{c}_{n-1} - \frac{B_{2^i}(\tilde{c}_{n-1}) - x_0}{D_{2^i}(\tilde{c}_{n-1})},$$

$$\tilde{c}_0 \text{ — начальное приближение корня}, n = 1, 2, \ldots$$

Таким образом, для вычисления корня c_i^* не нужно решать уравнение степени 2^i, а достаточно итеративно вычислять функции $B_k(c)$ и $D_k(c)$ и, затем, значения этих функций в точках c_n.

Алгоритм 1.4: **Нахождение константы Фейгенбаума (подробнее)**

Вход: $f(x, c)$; *✎ функции с параметром с, для которых ищется константа Фейгенбаума*
ε; *✎ точность промежуточных вычислений*
n; *✎ количество точек c_i^* для подсчёта предела δ*
Выход: массив значений $\{\delta_i\}_{i=1}^n$, $\delta \approx \delta_n$.

x_0; *✎ находим единственную критическую точку для $f(x)$: $f'_x(x_0, c) = 0$*
$\{c_0^*, c_1^*, \delta_1\} = \{0, 1, 3.2\}$; *✎ начальные приближения для вычислений*
$B(c) = f(f(x_0))$ *✎ $B(c) = B_2(c)$*
$D(c) = f'_x(f(x_0), c) f'_c(x_0, c) + f'_c(f(x_0), c)$ *✎ $D(c) = D_2(c)$*
для $i = 2, \ldots, n$
  $c = c_{i-1}^* + \frac{c_{i-1}^* - c_{i-2}^*}{\delta_{i-1}}$; *✎ начальное приближение для метода Ньютона*
  повторять 2^{i-1} **раз**
    $D(c) = f'_x(B(c), c) D(c) + f'_c(B(c), c)$; . *✎ цикл вычисляет $B_{2^i}(c)$ и $D_{2^i}(c)$ по предыдущим значениям $B_{2^{i-1}}(c)$ и $D_{2^{i-1}}(c)$*
    $B(c) = f(B(c), c)$;
  $C_{old} = 0$;
  повторять
    $C_{old} = C$; *✎ в этом цикле ищем корень $B_{2^i}(C) = x_0$ методом Ньютона*
    $C = C - \frac{B(C) - x_0}{D(C)}$; *✎ итерации методом Ньютона: $\tilde{c}_n = \tilde{c}_{n-1} - \frac{B_{2^i}(\tilde{c}_{n-1}) - x_0}{{B_{2^i}}'_c(\tilde{c}_{n-1})}$*
  пока не $\frac{|C - C_{old}|}{C} < \varepsilon$
  $c_i^* = C$;
  $\delta_i = \frac{c_{i-1}^* - c_{i-2}^*}{c_i^* - c_{i-1}^*}$;

Программируем на Maxima (листинг № 1.3):

Нахождение константы Фейгенбаума

```
f(x):=c-x^2$                                    /* изучаемая функция */
x0:0$                       /* единственная критическая точка функции f */
[c[0],c[1],d[1]]:[0,1,3.2]$              /* начальные приближенные значения */

B:f(f(x0))$                                  /* начальные значения для B и D */
D:1-2*f(x0)$
n:8$                         /* количество точек для подсчета предела d */
eps:10^-8$                             /* точность промежуточных вычислений */
```

```
9
10  for i:2 thru n do
11      (
12      C:c[i-1]+(c[i-1]-c[i-2])/d[i-1], /* начальное приближение для метода Ньютона */
13      thru 2^(i-1) do                                       /* итерируем B и D */
14          (
15          D:1-2*B*D,
16          B:f(B)
17          ),
18      C_old:0,
19      unless abs(C-C_old)/C<eps do                  /* запускаем метод Ньютона */
20          (
21          C_old:C,
22          C:C-at((B-x0)/D,[c=C])
23          ),
24      c[i]:C,
25      d[i]:(c[i-1]-c[i-2])/(c[i]-c[i-1]),
26      print(d[i])                           /* выводим на экран значение d[j] */
27      )$
```

Комментарий.

Замечания об использованных командах:
строчки 20–26 реализуют метод Ньютона для поиска корней уравнения `B=x0`, их можно заменить одной командой:
`c[i]:newton(B-x0,c,C,eps),`
убрав за ненадобностью строчку 17 (напомним, для этого надо подключить пакет `load(newton1)`).
`unless` *cond* `do` *smth* — цикл: пока выполняется условие *cond* выполнять действие *smth*.

Задача 1.1 Придумайте пример функции одновременно с притягивающей и отталкивающей неподвижными точками.

Задача 1.2 Пусть $f(x) = cx(1-x)$, $x \in [0,1]$. Изобразите $f^{(2)}(x)$ и найдите графически точки периода 2 для функции f, а также её неподвижные точки. Являются они притягивающими или отталкивающими для $f^{(2)}$? Изобразите $f^{(4)}(x)$ и найдите графически точки периодов 1, 2, 4 для функции f. Заметьте, как нестабильные неподвижные точки порождают пары стабильных точек периода 2.

Задача 1.3 Изобразить бифуркационную диаграмму возмущённой логистической функции $f(x) = cx(1-x)(1 + a\sin bx)$, $a = -0.015$, $b = 31.6$ ([2, Appendix D]). Что происходит с производной Шварца для данной функции?

Задача 1.4 Изобразить бифуркационную диаграмму для функций $x^2 + c$, $c\sin\pi x$, $cx^2\sin\pi x$ на соответствующих множествах.

Задача 1.5 Придумать функцию (и построить её бифуркационную диаграмму), отличную от рассмотренных выше, удовлетворяющую условиям теоремы Зингера.

Задача 1.6 Модифицируйте Листинг 1.2 так, чтобы можно было отрисовывать бифуркационную диаграмму в произвольной области $[a, b] \times [c, d] \subset U \times C$ (чтобы рассматривать части диаграммы при меньшем масштабе). Продемонстрируйте, что бифуркационная диаграмма обладает свойством самоподобия (например, для логистического отображения рассмотрев область $[0.53, 0.6] \times [-1.86, -1.82]$).

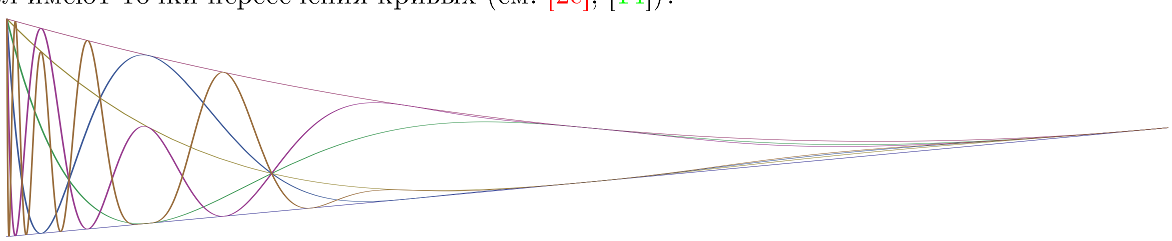
Задача 1.7 Придумайте пример функции с двумя притягивающими неподвижными точками. Найдите точку x_Γ, разделяющую области притяжения этих двух неподвижных точек. Продемонстрируйте, как меняется поведение орбиты точек при малых отклонениях от точки x_Γ.

Задача 1.8 Постройте модификацию бифуркационной диаграммы, рассмотренную Чипом Россом (см. [2e], [14]).

Задача 1.9 Озвучьте бифуркационную диаграмму и переход к хаосу: время композиции — это параметр c, каждая стабильная периодическая точка при данном c озвучивается своим музыкальным тоном. Таким образом зазвучит каждая ветвь диаграммы.

Задача 1.10 Покажите, что если $Sf(x) < 0$, то и $Sf^{(n)}(x) < 0$ при любом $n \in \mathbb{N}$.

Задача 1.11 Пусть $f_c(x) = x^2 + c$. Постройте Q-кривые $y(c) = f^{(n)}(0)$ при $n = 0, 1, 2, 3, 4, 5, 6, 7$ на одном графике, $c \in [-2, 0]$. Что напоминает получившийся рисунок, какой смысл имеют точки пересечения кривых (см. [2e], [14])?



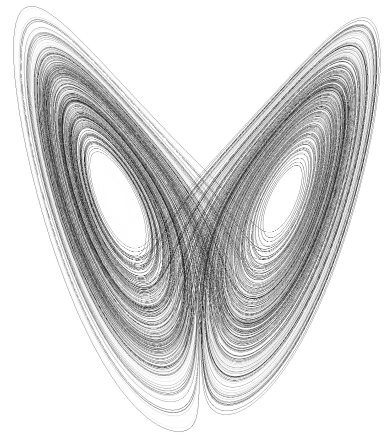
Задача 1.12 Изучите и напишите в Maxima алгоритм получения константы Фейгенбаума с помощью метода ренормализации, описанного в [12, 13].

Задача 1.13 Предложите алгоритм нахождения значений точек бифуркации c_n.

Задача 1.14 Докажите утверждение, выделенное курсивом в Лемме 1.1.

Рекомендуемая литература: [1, гл. 6], [2, гл. 4, Appendix D], [15], [14], [12], [16], [17], [18], [13], [19]; [2e], [11e].

Глава 2



ХАОТИЧЕСКАЯ ДИНАМИКА. ДЕТЕРМИНИРОВАННЫЙ ХАОС

Математика может открыть определённую последовательность даже в хаосе.

Гертруда Стайн

Предсказывать довольно сложно, особенно будущее.

Нильс Бор

Простейшая дискретная динамическая система — это множество с действующей на нём трансформацией: $f\colon X \to X$. Под действием этой трансформации каждая точка множества образует свою орбиту — последовательность точек

$$x, \quad f(x), \quad f^2(x) \equiv f(f(x)), \quad \ldots, \quad f^{(n)}(x) \equiv \underbrace{f(f \ldots (f}_{n \text{ раз}}(x)) \ldots), \quad \ldots$$

Для выделенной начальной точки $x_0 \in X$ этот процесс можно записать как последовательность

$$x_0, \quad x_1 = f(x_0), \quad x_2 = f(x_1), \quad \ldots, \quad x_n = f(x_{n-1}), \quad \ldots$$

Определённая таким образом динамическая система является детерминированной: зная начальную точку, мы всегда можем определить её следующее положение, применив в к ней трансформацию f.

Задачей теории динамических систем является описание характера эволюции системы во времени: изучение предельных траекторий, их устойчивости, а также типичности самой системы.

Динамические системы описывают многих естественные процессы: от численности популяций до процесса представления вещественного числа в двоичной системе счисления.

2.1 Детерминированный хаос. Теорема Такенса

2.1.1 Детерминированная хаотичность

Многие детерминистические динамические системы обнаруживают кажущееся хаотическое поведение. Обратно, по такому хаотическому поведению системы можно предположить, что оно на самом деле не случайно, а задается вполне определённым механизмом.

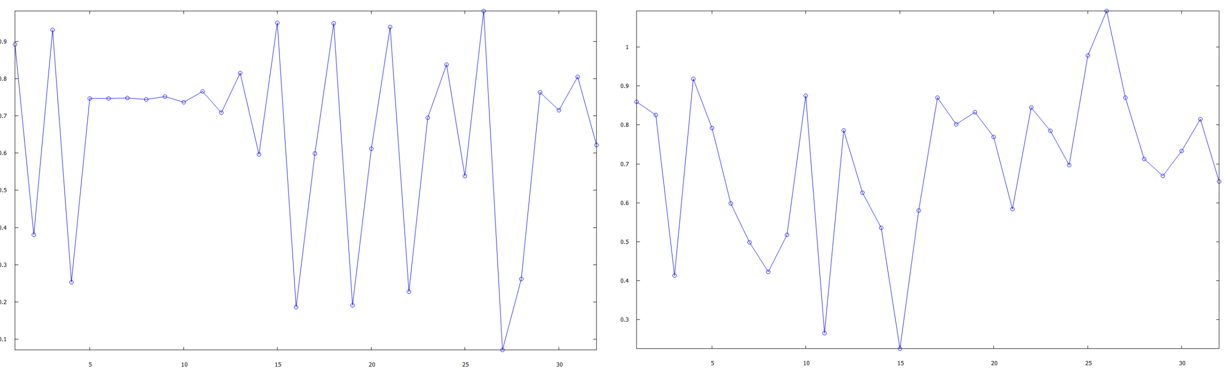
Классическим примером здесь является логистическая парабола — отображение $f\colon [0,1] \to [0,1]$:

$$f(x) = \alpha x(1-x), \qquad \alpha \in [2,4].$$

Рассмотрим вполне детерминированную последовательность значений[1] $x_n = f(x_{n-1})$, пусть $x_0 = 0.892$, $\alpha = 3.95$. Отобразим эти точки[2]:

пример

```
load(dynamics)$
evolution(3.95*x*(1-x),0.892,31,[style,linespoints])$
```



Найдем среднее m и дисперсию s данного набора значений $\{x_k = f(x_{k-1})\}$ и сгенерируем случайную последовательность randomseq с нормальным распределением $N(\mathtt{m},\mathtt{s})$:

пример

```
f(x):=3*x*(1-x)$
x[0]:0.5$
for i:0 thru 31 do x[i+1]:f(x[i])$
[m,s]:[mean(listarray(x)),var(listarray(x))]$
randomseq:random_normal(m,s,32)$
```

Комментарий.

`listarray(x)` — в данном случае возвращает списком элементы недекларированного массива {`x[0]`,…,`x[n]`} (а попросту, набора не связанных друг с другом переменных);

`mean(x)` — среднее выборки, представленной списком `x`;

`var(x)` — дисперсия выборки, представленной списком `x`;

`random_normal(m,s)` — псевдослучайная величина, распределенная нормально со средним `m` и дисперсией `s`;

[1] Пример взят из книги [20, с. 121].

[2] Воспользуемся уже обсуждавшейся командой `evolution`, с параметрами графики `[style,linespoints]` (соединять «жирные» точки ломаной), подробнее см. с. 39.

> `random_normal(m,s,n)` — псевдослучайный набор из `n` чисел, распределённых нормально со средним `m` и дисперсией `s`

Возьмём для определённости следующий массив данных и отобразим его:

пример

```
randomseq:[0.860,0.826,0.413,0.919,0.792,0.599,0.499,0.423,0.518,0.875,0.265,
0.786,0.626,0.537,0.226,0.581,0.870,0.802,0.833,0.769,0.584,0.845,0.785,
0.697,0.979,1.093,0.871,0.713,0.669,0.734,0.815,0.656]$
plot2d([discrete,randomseq])$
```

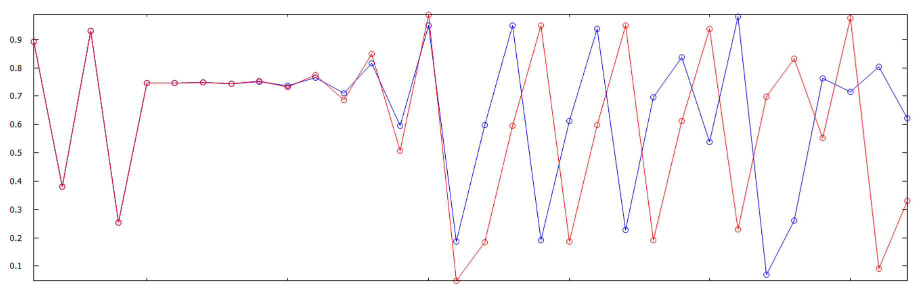
Можете ли вы отличить, какой из графиков выше порожден случайным процессом, а какой — детерминированным?

2.1.2 Существенная зависимость от начальных условий

Проиллюстрируем ещё одно характерное для данного случая свойство, называемое *существенной зависимостью от начальных условий*. Изобразим на одном графике две траектории для логистического отображения: при $x_0 = 0.892$ и $x_0 = 0.89201$ ($\alpha = 3.95$ в обоих случаях).

пример

```
load(draw)$
f(x):=3.95*x*(1-x)$
x[0]:0.892$
orb:append([[0,x[0]]],makelist([i+1,x[i+1]:f(x[i])],i,0,31))$
x[0]:0.89201$
orb2:append([[0,x[0]]],makelist([i+1,x[i+1]:f(x[i])],i,0,31))$
draw2d(point_type=circle,points_joined=true,
       points(orb),color=red,points(orb2))$
```



Комментарий.

> Здесь для удобства мы воспользовались командой `draw2d` из пакета `draw`, описанной в первой части пособия. Кроме того, здесь создание последовательностей $\{x_i\}$ и $orb = \{i, x_i\}$ происходит в одной строчке.

Легко заметить, что небольшое возмущение начального значения влечёт существенное изменение траектории.

2.1.3 Бифуркации, катастрофы в качественном поведении

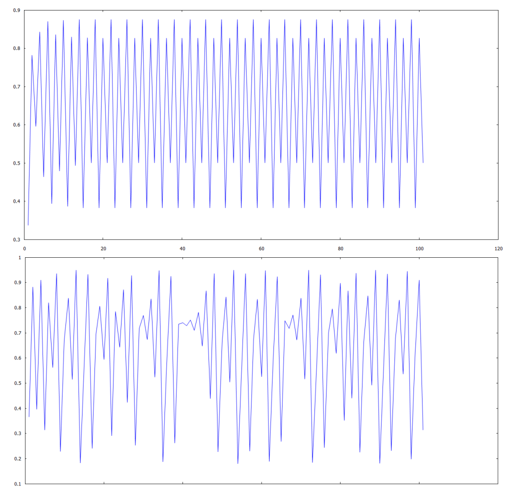
На примере нашего логистического отображения покажем ещё одно интересное свойство — качественное изменение поведения системы при небольших изменениях управляющего параметра. Если рассматривать α как параметр системы (аналогично параметру c в предыдущей главе), то можно заметить, например, что в окрестности точки $\alpha = 3.5$ системы $f(x, \alpha)$ имеют предельный цикл периода 4, в то время как в окрестности $\alpha = 3.7$ обнаруживается хаотическое поведение.

Изобразим это на двух графиках: пусть $x_0 = 0.892$ в обоих случаях.

пример

```
load(dynamics)$
evolution(3.5*x*(1-x),0.892,100)$
```

```
evolution(3.8*x*(1-x),0.892,100)$
```



2.1.4 Теорема Такенса о вложении

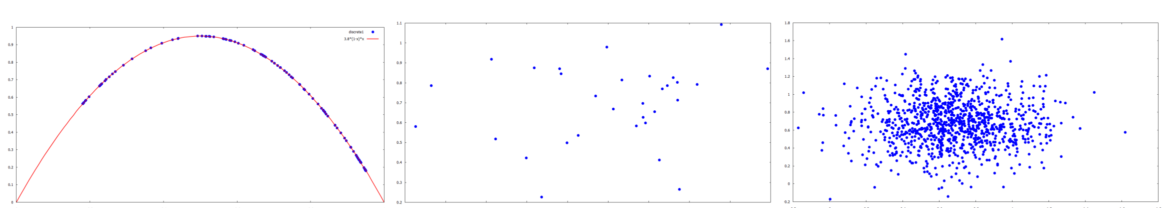
Одним из способов проверки, является ли данный набор данных не случайным, а порождённым некоторой неизвестной зависимостью, например так:

$$x_n = F(x_{n-1}, x_{n-2}, \ldots, x_{n-k}), \qquad \text{где } k \text{ «не очень большое»},$$

— является теорема Такенса о вложении.

Вложим одномерную последовательность $\{x_1, \ldots, x_n, \ldots\}$ в плоскость в виде последовательности:

$$\{(x_1, x_2), (x_2, x_3), \ldots, (x_{n-1}, x_n), (x_n, x_{n+1}), \ldots\}.$$



2.2 Фрактальные аттракторы динамических систем

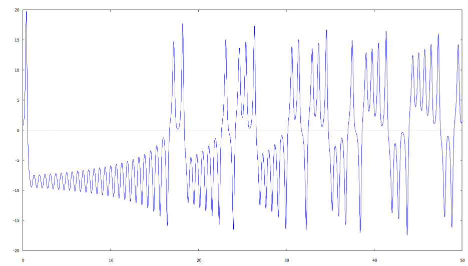
2.2.1 Аттрактор Лоренца

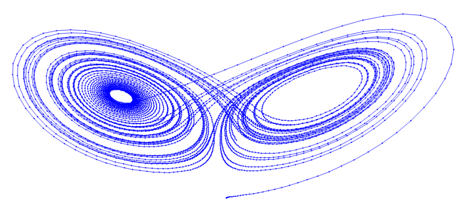
$$\begin{cases} \dot{x} = \sigma(y - x), \\ \dot{y} = x(r - z) - y, \\ \dot{z} = xy - bz. \end{cases}$$

пример

```
load(dynamics)$
[sigma,r,b]:[10,28,8/3]$
eq:[sigma*(y-x),x*(r-z)-y,x*y-b*z]$
init:[1.0,0,0]$
t_range:[t,0,50,0.01]$
sol:rk(eq,[x,y,z],init,t_range)$
len:length(sol)$
t:makelist(sol[k][1],k,1,len)$
x:makelist(sol[k][2],k,1,len)$
y:makelist(sol[k][3],k,1,len)$
z:makelist(sol[k][4],k,1,len)$
plot2d([discrete,t,x])$
```

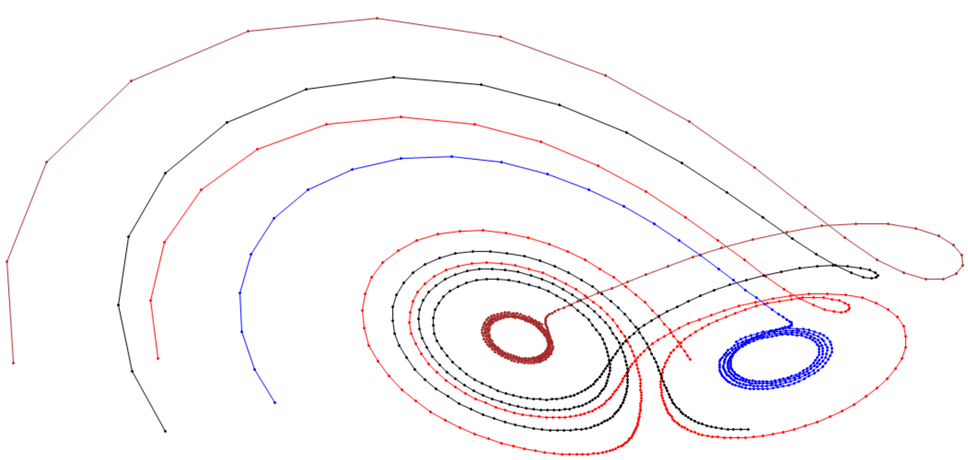
```
load(draw)$
draw3d(point_size=0.2,points_joined=true,
    point_type=filled_circle,points(x,y,z))$
```



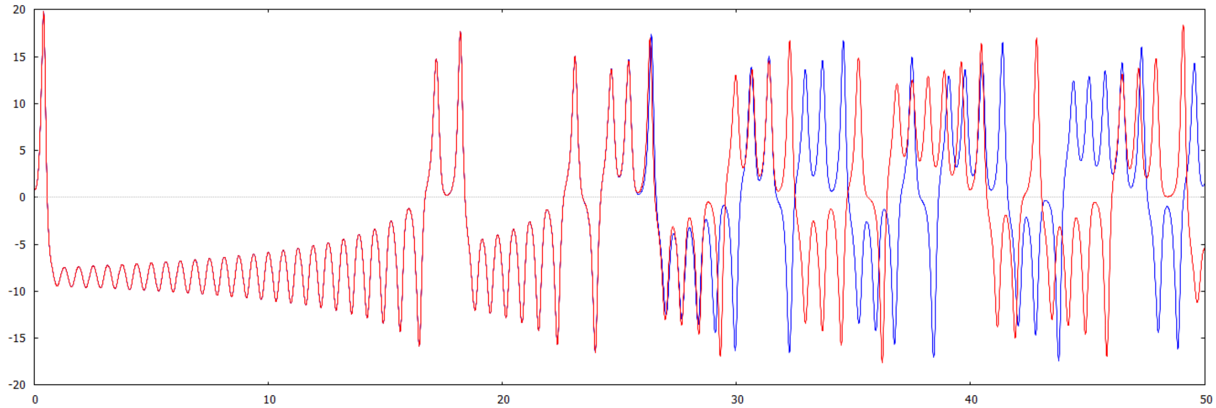


Отметим, что области графика $x(t)$, лежащие выше оси $x = 0$, соответствуют виткам в правой части «крыла бабочки» аттрактора Лоренца, а нижележащие области — виткам в левой её части. Обдумайте связь между поведением $x(t)$ и движением точки по аттрактору в фазовом пространстве. Очевидно, такую же связь можно усмотреть, если построить графики $y(t)$ и $z(t)$.

Одно из свойств аттрактора Лоренца как странного аттрактора: он является аттрактором, то есть, к нему «притягиваются» другие траектории в фазовом пространстве. Все траектории отличаются начальными значениями $r_0 = \{x(0), y(0), z(0)\} = \{x_0, y_0, z_0\}$ задачи Коши.



Другим свойством является существенная зависимость от начальных условий. Выберем одну из координат в фазовом пространстве: пусть это будет x. Изобразим два решения $x(t)$ задачи Коши: при $r_0 = \{1, 0, 0\}$ и $r_0 = \{1.001, 0, 0\}$ на одном рисунке.



При близких начальных условиях различия в траекториях проявляются экспоненциально быстро, поведение систем разительно отличается: там где синий график выше оси $x = 0$, а красный — ниже, — точки фазового пространства находятся в разных «крыльях бабочки», а значит, например, движение цилиндров в эксперименте Лоренца происходит в противоположных направлениях, и его трудно предсказать при неточности в начальном значении.

2.2.2 Аттрактор Энона

Задача 2.1 Покажите, что при различных начальных значениях траектории притягиваются к аттрактору Лоренца, как это изображено на с. 21.

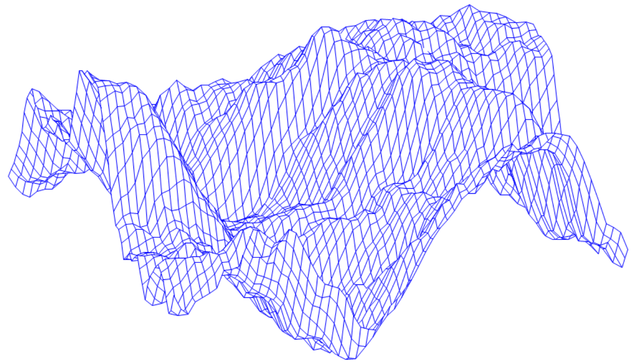
Задача 2.2 Покажите, что для аттрактора Лоренца при близких начальных значениях траектории быстро разбегаются, как это изображено на с. 22 и показано на с. 19 для логистического отображения. Для этого достаточно показать быстрое расхождение одной из компонент $x(t)$, $y(t)$ или $z(t)$ при малом отличии в r_0.

Задача 2.3 Постройте графики $y(t)$ и $z(t)$ для аттрактора Лоренца, как это сделано для графика $x(t)$ на странице с. 21. Пронаблюдайте связь между поведением графиков и характером движения точки в фазовом пространстве относительно аттрактора.

Задача 2.4 Придумайте, как можно показать, что аттрактор Лоренца локально устроен как произведение канторова множества на интервал, то есть его сечение Пуанкаре (например, плоскостью $z = 0$) — канторово множество.

Задача 2.5 Исследуйте аттрактор Рёсслера по аналогии с задачами 2.1–2.4.

Рекомендуемая литература: [20]



Глава 3

ФРАКТАЛЬНОЕ БРОУНОВСКОЕ ДВИЖЕНИЕ

Gin a body meet a body flyin' through the air.
Gin a body hit a body, Will it fly? And where?
Ilka impact has its measure, Ne'er a ane hae I,
Yet a' the lads they measure me,
Or, at least, they try.
Gin a body meet a body Altogether free,
How they travel afterwards We do not always see.
Ilka problem has its method By analytics high;
For me, I ken na ane o' them,
But what the waur am I?

Джеймс Клерк Максвелл
(подражая Роберту Бёрнсу)

Броуновское движение представляет собой пример естественного фрактала с фрактальной размерностью $d = 1.5$ (Мандельброт и Ван Несс, 1968). Впервые его наблюдал шотландский ботаник Роберт Броун[1] в 1827 году: он заметил непрерывное беспорядочное движение взвешенных в жидкости маленьких частиц (пыльцы), но ошибочно приписал причину движения самим частицам ([22]). Только в 1905 году Альберт Эйнштейн[2] и вслед за ним в 1906 году Мариан Смолуховский объяснили это движение хаотическими соударениями с молекулами окружающей среды ([21]). В 1908–1913 годах Жан Батист Перрен поставил ряд опытов ([23]), подтвердивших выводы Эйнштейна и Смолуховского. И, наконец, в 1923 год Норберт Винер построил первую математическую модель броуновского движения ([24, 25]). Альтернативные подходы были предложены Андреем Николаевичем Колмогоровым в 1933 году и Полем Леви в 1948 году ([26], [27], все три подхода см. в [28]).

[1]Это традиционный вариант русского написания фамилии Brown, хотя правильнее переводить *Браун*, как это сделано в [21].

[2]Правильнее *Айнштайн*, но традиция — сильная штука!

В этой главе мы построим кривые классического броуновского движения, а также кривые и поверхности фрактального броуновского движения (рассмотренные Б. Мандельбротом и В. Нессом [29]), пригодные для моделирования рельефа поверхности и других естественных процессов.

3.0.3 Классическое броуновское движение

Рассмотрим случайный процесс (случайную величину) $X(t)$, заданную на отрезке $[0, T]$.

Определение 3.1. *Случайный процесс $X(t)$ называется одномерным броуновским движением (или винеровским процессом) на интервале $[0, T]$, если он обладает следующими свойствами[3]:*

- $X(0) = 0$ *почти наверное и* $X(t)$ *— почти наверное непрерывная функция на* $[0, T]$*;*
- $X(t)$ *— процесс с независимыми приращениями[4];*
- $X(t)$ *— процесс с приращениями, распределёнными нормально[5].*

Отметим следующие свойства броуновского движения:

- $X(t)$ почти наверное нигде не дифференцируем (см. [29]);
- $X(t)$ — марковский процесс (не обладает памятью), т.е. если известна величина $X(t)$, то при $t_1 < t < t_2$ величины $X(t_1)$ и $X(t_2)$ независимы;
- Фрактальная размерность графика $X(t)$ равна 1.5;
- Приращение $X(t)$ обладает свойством статистического самоподобия[6]: для любого $r > 0$
$$X(t + \triangle t) \triangleq \frac{1}{\sqrt{r}}(X(t + r\triangle t) - X(t)).$$
- Стационарность приращений: дисперсия приращения зависит только от разности моментов времени:
$$D(X(t_2) - X(t_1)) = \sigma^2 |t_2 - t_1|. \tag{3.1}$$
- Математическое ожидание приращения равно
$$E(|X(t_2) - X(t_1)|) = \sqrt{2/\pi}\sigma\sqrt{|t_2 - t_1|}.$$

Для моделирования броуновского движения можно воспользоваться разными алгоритмами. Мы рассмотрим три из них.

Например, проще всего реализовать дискретную реализацию броуновского движения, рассмотрев последовательность $x_0 = 0$, $x_{n+1} = x_n + g_n$, где g_n — случайная величина, имеющая нормальное распределение (например, $N(0, 1)$).

[3] Определённый таким образом процесс является нормальным, или гауссовским, а также марковским.

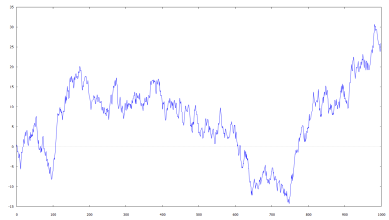
[4] При любых фиксированных $t_1 < \ldots < t_N$ случайные величины $\triangle_1 = X(t_2) - X(t_1), \ldots, \triangle_{N-1} = X(t_N) - X(t_{N-1})$ являются независимыми.

[5] Случайная величина $\triangle X = X(t_1) - X(t_2)$ имеет нормальное распределение с параметрами $N(0, \sigma^2(t_2 - t_1))$ для любых $t_2 > t_1$, а величина σ^2 — постоянная для данного процесса.

[6] Здесь символ $\triangleq$ означает одинаковое распределение двух случайных величин.

пример

```
load(distrib)$
N:1000$
X[0]:0$
for i:1 thru N do X[i]:X[i-1]+random_normal(0,1)$
plot2d([discrete,makelist(i,i,0,N),listarray(X)])$
```



3.0.4 Алгоритм срединных смещений

Метод *случайного срединного смещения* основан на работах Н. Винера ([24, 25]), он более сложен, чем метод из предыдущего параграфа, однако используется для конструктивного доказательства существования броуновского движения, а также для построения *фрактальной интерполяции* (когда необходимо, чтобы кривая проходила через заданные точки интерполяции). Метод также может быть обобщен на случай n-мерных броуновских движений.

Алгоритм случайного срединного смещения вычисляет значения $X(t)$ в диадических рациональных точках вида $\frac{k}{2^n} \in [0,1]$. Последовательно вычисляются значения в середине отрезка $[0,1]$, затем в серединах отрезков $[0,\frac{1}{2}]$ и $[\frac{1}{2},1]$ и т. д.. На каждом шаге итерации должен выполняться закон дисперсии для приращений (3.1) в вычисленных точках. Параметр σ определяет масштаб по вертикальной оси, не влияя на фрактальную размерность графика.

Алгоритм 3.1: **Броуновское движение методом срединного смещения (1)**

Вход: N; ✎ *число шагов алгоритма, при этом всего 2^N+1 точек интерполяции*
σ; ✎ *параметр вертикального масштаба, коэффициент дисперсии*
Выход: массив значений $\{X(\frac{k}{2^N})\}_{k=0}^{2^N}$. ✎ *реализация броуновского движения $X(t)$ на дискретном множестве точек вида $t_k = \frac{k}{2^N}$, $k=\overline{0,2^N}$*

1: $X(0)=0$;
2: $X(1)=\sigma g$ ✎ *g — случайная величина, распределенная нормально с параметрами $N(0,1)$*
3: Шаг 1:
$$X\left(\tfrac{1}{2}\right)=\tfrac{1}{2}\left(X(0)+X(1)\right)+\tfrac{1}{2}\sigma g;$$
4: Шаг 2:
$$X\left(\tfrac{1}{4}\right)=\tfrac{1}{2}\left(X(0)+X\left(\tfrac{1}{2}\right)\right)+\tfrac{1}{2^{3/2}}\sigma g;$$
$$X\left(\tfrac{3}{4}\right)=\tfrac{1}{2}\left(X\left(\tfrac{1}{2}\right)+X(1)\right)+\tfrac{1}{2^{3/2}}\sigma g;$$
$\vdots$
5: Шаг N:
$$X\left(\tfrac{1}{2^N}\right)=\tfrac{1}{2}\left(X(0)+X\left(\tfrac{1}{2^{N-1}}\right)\right)+\tfrac{1}{2^{(N+1)/2}}\sigma g;$$
$\vdots$
$$X\left(\frac{2^N-1}{2^N}\right)=\frac{1}{2}\left(X\left(\frac{2^{N-1}-1}{2^{N-1}}\right)+X(1)\right)+\frac{1}{2^{(N+1)/2}}\sigma g;$$

Заметим, что точки $t_k=\frac{k}{2^N}$ можно последовательно занумеровать номерами k. При этом если точка имеет вид $\frac{a}{2^b}$, то её номер $k=a2^{N-b}$. Укажем алгоритм, в котором точки t_k пронумерованы эффективно.

Алгоритм 3.2: **Броуновское движение методом срединного смещения (2)**

Вход: N; ✎ *число шагов алгоритма, при этом всего 2^N+1 точек интерполяции*
σ; ✎ *параметр вертикального масштаба, коэффициент дисперсии*
Выход: массив значений $\{X(k)\}_{k=0}^{2^N}$. ✎ *реализация броуновского движения $X(t)$ на дискретном множестве точек вида $t_k=\frac{k}{2^N}$, $k=\overline{0,2^N}$*

$X(0)=0$;
$X(1)=\sigma g$ ✎ *g — случайная величина, распределенная нормально с параметрами $N(0,1)$*
для $j=1,\dots,N$
  для $i=1,\dots,2^{N-1}$
    $X\left((2i-1)2^{N-j}\right)=X\left((i-1)2^{N-j+1}\right)+X\left(i2^{N-j+1}\right)+\frac{1}{2^{(j+1)/2}}\sigma g$; ✎ *это соответствует формуле $X\left(\frac{2i-1}{2^j}\right)=X\left(\frac{i-1}{2^{j-1}}\right)+X\left(\frac{i}{2^{j-1}}\right)+\frac{1}{2^{(j+1)/2}}\sigma g$*

Программируем на Maxima (листинг № 3.1):

Броуновское движение методом срединного смещения

```
1   load(distrib)$
2   N:10$
3   s:2$
4   [X[0],X[2^N]]:[0,s*random_normal(0,1)]$
5
6   for j:1 thru N do
7      for i:1 thru 2^(j-1) do
8         X[(2*i-1)*2^(N-j)]:float((X[(i-1)*2^(N-j+1)]+X[i*2^(N-j+1)])/2+
9                                s*random_normal(0,1)/2^((j+1)/2))$
10
11  plot2d([discrete,makelist(k/2^N,k,0,2^N),listarray(X)]);
```

Комментарий.

В строчке 8 используется команда `float`, чтобы хранить в памяти результат, а не последовательность действий (подумайте над этим).

3.0.5 Фрактальное броуновское движение

Фрактальное броуновское движение (ФБД) уже не является марковским процессом, а обладает некоторой «памятью» (Б. Мандельброт и В. Несс, [29]). Кроме того, вводя параметр $0<H<1$ можно получить одномерное ФБД размерности $d=2-H$ и двумерное ФБД размерности $d=3-H$. Заметим, что классическое броуновское движение получается как частный случай при $H=0.5$. Для аппроксимации ФБД нет простого метода, вроде суммирования нормальных случайных величин, как в случае классического броуновского движения. Для аппроксимации ФБД наиболее удобно использовать преобразования Фурье.

Рассмотрим случайный процесс (случайную величину) $X(t)$, заданную на отрезке $[0,T]$.

Определение 3.2. *Случайный процесс $X(t)$ называется одномерным*[7] *фрактальным броуновским движением на интервале $[0, T]$, если он обладает следующими свойствами*[8]*:*

- $X(0) = 0$ *почти наверное и* $X(t)$ *— почти наверное непрерывная функция на* $[0, T]$*;*
- $X(t)$ *— процесс с приращениями, распределенными нормально*[9]*.*

Отметим следующие свойства фрактального броуновского движения:

- $X(t)$ почти наверное нигде не дифференцируем (см. [29]);
- Фрактальная размерность графика $X(t)$ равна $2 - H$;
- Процесс $x(t)$ не обладает свойством независимости приращений;
- Приращение $X(t)$ обладает свойством статистического самоподобия[10]: для любого $r > 0$
$$X(t + \triangle t) \triangleq \frac{1}{r^H}(X(t + r\triangle t) - X(t)).$$
- Стационарность приращений: дисперсия приращения зависит только от разности моментов времени:
$$D(X(t_2) - X(t_1)) = \sigma^2 |t_2 - t_1|^{2H}. \tag{3.2}$$
- Математическое ожидание приращения равно
$$E(|X(t_2) - X(t_1)|) = \sqrt{2/\pi}\sigma |t_2 - t_1|^H.$$

3.0.6 Метод Фурье-фильтрации для построения ФБД

Метод Фурье-фильтрации для аппроксимации ФБД заключается в следующем. Нам понадобится

Теорема 3.1 ([1])**.** *Если $X(t)$ — ФБД с параметром H, то его спектральная плотность*
$$S(f) \propto \frac{1}{f^{2H+1}}.$$

Идея метода состоит в следующем. Строится преобразование Фурье для искомого ФБД в частотной области, задавая случайные фазы и подбирая амплитуды, удовлетворяющие свойству из Теоремы 3.1. Затем получаем ФБД во временной области с помощью обратного преобразования Фурье.

Будем моделировать дискретный аналог ФБД, то есть наша цель — получить величины $\{X_n\}_{n=0}^{N-1}$, аппроксимирующие ФБД в точках n. Воспользуемся формулой дискретного преобразования Фурье
$$\hat{X}_n = \sum_{k=0}^{N-1} X_k e^{-2\pi kn/N}$$

[7] Не путайте с фрактальной размерностью его графика!

[8] Определённый таким образом процесс является нормальным, или гауссовским, но не марковским.

[9] Случайная величина $\triangle X = X(t_1) - X(t_2)$ имеет нормальное распределение с параметрами $N(0, \sigma^2(t_2 - t_1)^{2H})$ для любых $t_2 > t_1$, а величины σ^2 и H — постоянные для данного процесса.

[10] Здесь символ $\triangleq$ означает одинаковое распределение двух случайных величин.

и обратного дискретного преобразования Фурье

$$X_n = \sum_{k=0}^{N-1} \hat{X}_k e^{2\pi kn/N}.$$

Далее будем рассматривать только чётные значения N, а для применения метода *быстрого дискретного преобразования Фурье* нужно, чтобы $N = 2^M$, $M \in \mathbb{N}$. Метод быстрого дискретного преобразования Фурье реализован во многих системах компьютерной алгебры, в том числе и в Maxima. Он позволяет сократить вычисления в $\frac{2N}{\log_2 N}$ раз.

Для того, чтобы получающиеся величины X_n были вещественными, мы наложим условие сопряжённой симметрии:

$$\hat{X}_0, \hat{X}_{N/2} \in \mathbb{R}, \quad \hat{X}_n = \hat{X}_{N-n}, \quad n = 1, \ldots, N/2 - 1. \tag{3.3}$$

Фильтрация относится к той части моделирования, когда мы заставляем коэффициенты преобразования Фурье удовлетворять степенному закону из Теоремы 3.1:

$$|\hat{X}_n|^2 \propto \frac{1}{n^{2H+1}}, \quad n = 1, \ldots, N/2.$$

Для этого мы возьмём

$$\hat{X}_n = \frac{g e^{2\pi iu}}{n^{H+0.5}},$$

где g — независимые значения нормально распределённой случайной величины с параметрами $N(0,1)$, а u — независимые значения равномерно распределённой на отрезке $[0,1]$ случайной величины. Оставшиеся коэффициенты вычислим из соотношений (3.3).

Для вычисления искомой аппроксимации ФБД $\{X_n\}_{n=0}^{N-1}$ применим обратное дискретное преобразование Фурье к набору $\{\hat{X}_n\}_{n=0}^{N-1}$.

Алгоритм 3.3: **Кривая ФБД методом Фурье-фильтрации**

Вход: $H \in (0,1)$; ✎ *параметр ФБД, размерность графика равна $d = 2 - H$*
$N = 2^M, M \in \mathbb{N}$; ✎ *параметр, определяющий количество точек дискретизации ФБД*
Выход: массив значений $\{X_n\}_{n=0}^{N-1}$. ✎ *дискретная аппроксимация ФБД в последовательные моменты времени n*

$\hat{X}_0 = g$;
для $j = 1, \ldots, N/2 - 1$
  $\hat{X}_j = \frac{g e^{2\pi iu}}{j^{H+0.5}}$;
$\hat{X}_{N/2} = \frac{g\cos(2\pi iu)}{(N/2)^{H+0.5}}$; ✎ *Здесь cos — вещественная часть комплексной экспоненты e*
для $j = N/2 + 1, \ldots, N - 1$
  $\hat{X}_j = \overline{\hat{X}_{N-j}}$;
$X = \text{ОДПФ}(\hat{X})$. ✎ *Вектор $X = \{X_0, \ldots, X_{N-1}\}$ получается обратным дискретным преобразованием Фурье из вектора $\hat{X} = \{\hat{X}_0, \ldots, \hat{X}_{N-1}\}$.*

Программируем на Maxima (листинг № 3.2):

Кривая ФБД методом Фурье-фильтрации

```
load(distrib)$
load(fft)$
N:2^4$
H:1$

X[0]:random_normal(0,1)$
for j:1 thru N/2-1 do X[j]:random_normal(0,1)*
                        exp(2*%pi*%i*random_continuous_uniform(0,1))/j^(H+0.5)$
X[N/2]:random_normal(0,1)*cos(2*%pi*random_continuous_uniform(0,1))/(N/2)^(H+0.5)$
for j:N/2+1 thru N-1 do X[j]:conjugate(X[N-j])$

Y:inverse_fft(rectform(ev(listarray(X),numer)))$

plot2d([discrete,makelist(i,i,0,N-1),realpart(Y)])$
```

Комментарий.

`fft` — пакет для работы с быстрым дискретным преобразованием Фурье. Для применения команд этого пакета длина вектора-аргумента должна быть степенью 2. Вектор-аргумент может быть комплекснозначным, но все числа должны быть записаны в алгебраической форме.

`fft(x)` — возвращает массив (или список), являющийся быстрым дискретным преобразованием Фурье от массива (или списка) `x`;

`inverse_fft(x)` — возвращает массив (или список), являющийся быстрым обратным дискретным преобразованием Фурье от массива (или списка) `x`.

`random_continuous_uniform(a,b)` — псевдослучайная величина, распределенная равномерно на отрезке [a, b], команда пакета `distrib`;

`conjugate(x)` — возвращает комплексно-сопряжённое к комплексному числу `x`;

Зачем нужны команды `rectform` и `ev`?

`rectform(x)` — приводит комплексное число `x` к алгебраической форме $a + ib$; без приведения числа к алгебраической форме нельзя пользоваться командой `fft`;

`ev(x,numer)` — в данном случае вычисляет значение `x` с флагом `numer` (численно).

Рассмотрим пример, чтобы понять, зачем нам нужны команды `rectform` и `ev`:

пример

```
fpprintprec:14$
a:-0.1234*%e^(0.1234*%e*i*%pi);
float(a);
ev(a,numer);
rectform(a);
rectform(ev(a,numer));
```

```
(%o2) -0.1234 %e^(0.1234 %i π)
(%o3) -0.1234 2.718281828459^(0.1234 %i π)
(%o4) -0.1234 (0.378034692412 %i+0.925791429715)
(%o5) -0.1234 %i sin(0.1234 π)-0.1234 cos(0.1234 π)
(%o6) -0.0466494810436 %i-0.114242662427
```

`fpprintprec:x` — константа, ограничивающая количество выводимых на экран знаков в десятичном представлении всех чисел в программе; она не изменяет сами числа и не влияет на выполнение программы.

Зачем использована команда `realpart`, ведь теоретически должны получаться вещественные значения при соблюдении условия сопряжённой симметрии (3.3)? При подсчёте обратного преобразования Фурье с конечной точностью заданных значений получаются погрешности, в данном случае порядка $10^{-16}i$.

Для построения аппроксимации двумерного фрактального броуновского движения методом Фурье-фильтрации используются те же идеи, что и в одномерном случае[11]. Вместо $\hat{X}_n$ используются $\hat{X}_{k,j}$, $k,j=\overline{0,N-1}$, условие Теоремы 3.1 примет вид ([30]):

$$|\hat{X}_{k,j}|^2 \propto \frac{1}{(k^2+j^2)^{H+1}}, \quad n,k=1,\ldots,N/2,$$

мы возьмем[12]

$$\hat{X}_{k,j} = \frac{ge^{2\pi i u}}{(k^2+j^2)^{H/2+0.5}}, \quad n,k=1,\ldots,N/2.$$

Запишем обратное дискретное преобразование Фурье: для $m,n=\overline{0,N-1}$

$$\begin{gathered} X_{m,n} = \sum_{k=0}^{N-1}\sum_{j=0}^{N-1}\hat{X}_{k,j}e^{-2\pi i\frac{kn+jm}{N}} = \hat{X}_{0,0} + \sum_{k=1}^{N-1}\hat{X}_{k,0}e^{-2\pi i\frac{kn}{N}} + \sum_{j=1}^{N-1}\hat{X}_{0,j}e^{-2\pi i\frac{jm}{N}} + \\ + \sum_{k=1}^{N/2}\sum_{j=1}^{N/2}\hat{X}_{k,j}e^{-2\pi i\frac{kn+jm}{N}} + \sum_{k=\frac{N}{2}+1}^{N-1}\sum_{j=\frac{N}{2}+1}^{N-1}(\ldots) + \sum_{k=1}^{N/2}\sum_{j=\frac{N}{2}+1}^{N-1}(\ldots) + \sum_{k=\frac{N}{2}+1}^{N-1}\sum_{j=1}^{N/2}(\ldots). \quad (3.4) \end{gathered}$$

Из формулы (3.4) следует, что для вещественности всех величин $X_{m,n}$ достаточно выполнения следующих условий сопряжённой симметрии:

$$\begin{aligned} \hat{X}_{N-k,N-j} &= \overline{\hat{X}_{k,j}}, & k,j &= \overline{1,N/2}; & \hat{X}_{N/2,N/2} &\in \mathbb{R}, & (3.5)\\ \hat{X}_{k,N-j} &= \overline{\hat{X}_{N-k,j}}, & k,j &= \overline{1,N/2-1}; & \hat{X}_{0,0} &\in \mathbb{R}, & (3.6)\\ \hat{X}_{0,N-j} &= \overline{\hat{X}_{0,j}}, & j &= \overline{1,N/2}; & \hat{X}_{0,N/2} &\in \mathbb{R}, & (3.7)\\ \hat{X}_{N-k,0} &= \overline{\hat{X}_{k,0}}, & k &= \overline{1,N/2}; & \hat{X}_{N/2,0} &\in \mathbb{R}. & (3.8) \end{aligned}$$

(очевидно, условия (3.7)–(3.8) обеспечивают вещественность первых двух сумм, а условия (3.5)–(3.6) — оставшихся четырёх сумм).

[11]Здесь одно и два измерения — это размерность аргумента процесса X: $X(t)$ или $X(t_1,t_2)$, а не фрактальная размерность их графиков.

[12]Мы строим аппроксимацию ФБД на квадратной решётке узлов, однако приведённый здесь алгоритм можно легко распространить на прямоугольную решётку $2^{M_1}\times 2^{M_2}$, в этом случае формулы будут более громоздкими.

Алгоритм 3.4: **Поверхность ФБД методом Фурье-фильтрации**

Вход: $H \in (0,1)$; ✎ *параметр ФБД, размерность графика равна $d = 3 - H$*
$N = 2^M, M \in \mathbb{N}$; ✎ *параметр, определяющий количество точек ФБД по каждому из двух измерений*

Выход: массив значений $\{X_{n,k}\}_{n,k=0}^{N-1}$. ✎ *дискретная аппроксимация ФБД на решётке узлов*

для $j,k = \overline{1, N/2}$
 $\hat{X}_{j,k} = \frac{g e^{2\pi i u}}{(j^2+k^2)^{H/2+0.5}}$;
 $\hat{X}_{N-j,N-k} = \overline{\hat{X}_{j,k}}$;
для $k = \overline{1, N/2-1}$
 $\hat{X}_{0,k} = \frac{g e^{2\pi i u}}{(k^2)^{H/2+0.5}}$;
 $\hat{X}_{k,0} = \frac{g e^{2\pi i u}}{(k^2)^{H/2+0.5}}$;
 $\hat{X}_{0,N-k} = \overline{\hat{X}_{0,k}}$;
 $\hat{X}_{N-k,0} = \overline{\hat{X}_{k,0}}$;
для $j,k = \overline{1, N/2-1}$
 $\hat{X}_{N-j,k} = \frac{g e^{2\pi i u}}{((N-j)^2+k^2)^{H/2+0.5}}$;
 $\hat{X}_{j,N-k} = \overline{\hat{X}_{N-j,k}}$;
$\hat{X}_{0,0} = 0$;
$\hat{X}_{N/2,0} = \frac{g\cos(2\pi u)}{((N/2)^2)^{H/2+0.5}}$;
$\hat{X}_{0,N/2} = \frac{g\cos(2\pi u)}{((N/2)^2)^{H/2+0.5}}$;
$\hat{X}_{N/2,N/2} = \frac{g\cos(2\pi u)}{(2(N/2)^2)^{H/2+0.5}}$;
$X = \text{ОДПФ}(\hat{X})$; ✎ *Обратное дискретное преобразование Фурье матрицы $\hat{X} = \{\hat{X}_{j,k}\}_{j,k=0}^{N-1}$*

Программируем на Maxima (листинг № 3.3):

Поверхность ФБД методом Фурье-фильтрации

```
load(distrib)$
load(fft)$
load(draw)$
N:2^4$
H:1$

for j:1 thru N/2 do
   for k:1 thru N/2 do
      (Y[j,k]:random_normal(0,1)*exp(2*%pi*%i*random_continuous_uniform(0,1))/
            (j^2+k^2)^(H/2+0.5),
      Y[N-j,N-k]:conjugate(Y[j,k])
      )$
for k:1 thru N/2-1 do
   (Y[0,k]:random_normal(0,1)*exp(2*%pi*%i*random_continuous_uniform(0,1))/
         (k^2)^(H/2+0.5),
```

```
16        Y[k,0]:random_normal(0,1)*exp(2*%pi*%i*random_continuous_uniform(0,1))/
17               (k^2)^(H/2+0.5),
18        Y[0,N-k]:conjugate(Y[0,k]),
19        Y[N-k,0]:conjugate(Y[k,0])
20       )$
21   for j:1 thru N/2-1 do
22       for k:1 thru N/2-1 do
23           (Y[N-j,k]:random_normal(0,1)*exp(2*%pi*%i*random_continuous_uniform(0,1))/
24                     ((N-j)^2+k^2)^(H/2+0.5),
25           Y[j,N-k]:conjugate(Y[N-j,k])
26           )$
27   Y[0,0]:0$
28   Y[N/2,0]:random_normal(0,1)*cos(2*%pi*random_continuous_uniform(0,1))/
29            ((N/2)^2)^(H/2+0.5)$
30   Y[0,N/2]:random_normal(0,1)*cos(2*%pi*random_continuous_uniform(0,1))/
31            ((N/2)^2)^(H/2+0.5)$
32   Y[N/2,N/2]:random_normal(0,1)*cos(2*%pi*random_continuous_uniform(0,1))/
33              (2*(N/2)^2)^(H/2+0.5)$
34
35   X:map(inverse_fft,rectform(genmatrix(Y,N-1,N-1,0,0)))$
36   Z:realpart(map(inverse_fft,transpose(X)))$
37
38   draw3d(elevation_grid(Z,0,0,1,1),surface_hide = true);
```

Комментарий.

`genmatrix(A,`j_1`,`j_2`,`i_1`,`i_2`)` — возвращает матрицу, соответствующую массиву A, причём элемент `A[`i_1,,j_1`]` является левым верхним элементом матрицы, а элемент `A[`i_2,,j_2`]` — нижним правым (аналог команды `listarray` для списков); если пропустить аргументы i_1 и j_1, то по умолчанию считается $i_1 = j_1 = 1$;

`transpose(x)` — возвращает транспонированную к `x` матрицу;

Заметим, что в силу формулы

$$X_{m,n} = \sum_{k=0}^{N-1}\sum_{j=0}^{N-1} \hat{X}_{k,j} e^{-2\pi i \frac{kn+jm}{N}} = \sum_{k=0}^{N-1} e^{-2\pi i \frac{kn}{N}} \sum_{j=0}^{N-1} \hat{X}_{k,j} e^{-2\pi i \frac{jm}{N}} = \sum_{k=0}^{N-1} \widehat{\hat{X}_k} e^{-2\pi i \frac{kn}{N}}$$

двумерное дискретное преобразование Фурье и обратное к нему можно получить, используя одномерные преобразования Фурье следующим образом: сначала применяем преобразование Фурье к каждой строчке матрицы $\hat{X}$, а затем — к каждому столбцу полученной матрицы. Эта схема реализована в нашем алгоритме в строках 35-36;

`elevation_grid(Z,`x_0`,`y_0`,`$\triangle x$`,`$\triangle y$ — команда, используемая внутри команды `draw3d`, которая рисует график поверхности, аппликаты точек которой заданы элементами матрицы `Z` $_{m,n}$, а абсциссы и ординаты точек — индексами матрицы. А именно, индексы матрицы отображаются на плоскость следующим образом: $_{1,1} \leftrightarrow (x_0, y_0 + \triangle y)$, $_{1,n} \leftrightarrow (x_0 + \triangle x, y_0 + \triangle y)$, $_{m,1} \leftrightarrow (x_0, y_0)$ и $_{m,n} \leftrightarrow (x_0 + \triangle x, y_0)$, задавая отображение всех индексов на прямоугольную сетку узлов на плоскости.

Эта команда поддерживает следующие опции, указываемые в команде `draw3d`:

`surface_hide = true` — отрисовывает только видимые части поверхности;

`line_type = dots` — тип линии сетки на поверхности: `solid` (сплошная, по умолчанию) и `dots` (точечная);

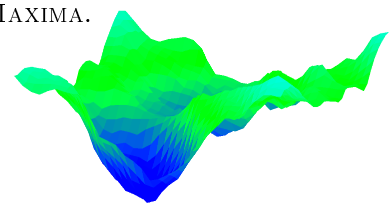
`line_width = x` — ширина линии сетки на поверхности (x> 0, по умолчанию 1);

`color = red` — задаёт цвет графика;

`enhanced3d(f(x,y,z),x,y,z)` — задает режим раскраски поверхности: x,y,z — явное указание переменных функции f, реализующей цвет точки (x,y,z) поверхности; кроме того, для этой опции можно (но не обязательно!) указать палитру цветов (по умолчанию она «марсианская»):

`palette=[`α`,`β`,`γ`]` — палитра, задаваемая тремя числами α, β, γ; заметим, её задание отличается для команды `plot3d`, описанной в справке по Maxima.

Попробуйте заменить строчку 38 на следующий код:



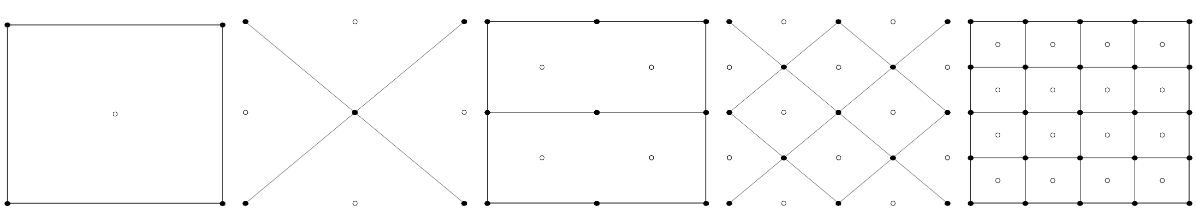
пример

```
draw3d(palette=[0,30,25],enhanced3d=[z,x,y,z],
surface_hide=true, elevation_grid(m,0,0,1,1));
```

И напоследок приведем алгоритм, позволяющий строить аппроксимацию поверхности ФБД с помощью метода срединного смещения. Этот алгоритм работает сравнительно быстро и поэтому широко применяется, однако его реализация не полностью удовлетворяет определению ФБД.

Воспользуемся аналогией с одномерным методом срединных смещений для аппроксимации броуновского движения. Определим значения $X(t_1, t_2)$ в вершинах большого квадрата (см. схему ниже) и вычислим значение в его центре как среднее арифметическое значений его вершин плюс случайное смещение. Рассмотрим квадраты (треугольники, если их центры лежат на границе исходного квадрата), стороны которых в $\sqrt{2}$ раз меньше и параллельны диагоналям исходного квадрата, определим значение в них как среднее арифметическое значений окружающих их вершин плюс случайные смещения.

Повторим два этих шага необходимое число раз. Таким образом, на каждом шаге сначала вычисляются значения в центрах «обычных» квадратов, а затем — в центрах «диагональных» квадратов.



Алгоритм 3.5: **Поверхность ФБД методом срединного смещения**

Вход: $H \in (0,1)$; ✎ *параметр ФБД, размерность графика равна $d = 3 - H$*
σ; ✎ *параметр, определяющий вертикальный масштаб*
$N = 2^M, M \in \mathbb{N}$; ✎ *параметр, определяющий количество точек ФБД по каждому из двух измерений*
Выход: массив значений $\{X_{n,k}\}_{n,k=0}^{N}$. ✎ *дискретная аппроксимация ФБД на решётке узлов*

$X_{0,0} = X_{0,N} = X_{N,0} = X_{N,N} = 0$; ✎ *инициализация значений в вершинах большого квадрата*
$r = \sigma$;
$D = N$; ✎ *инициализация шагов решётки*
$d = N/2$;
повторять M **раз**
  $r = r/2^{H/2}$; ✎ *меняем масштаб*
  Считаем значения в центрах квадратов как среднее арифметическое их вершин плюс случайное приращение:
  для $i, j = d, \dots, N-2$ с шагом D
    $X_{i,j} = (X_{i+d,j+d} + X_{i+d,j-d} + X_{i-d,j+d} + X_{i-d,j-d})/4 + rg$; ✎ *здесь и делее g — случайная величина, распределенная нормально с параметрами $N(0,1)$*
  $r = r/2^{H/2}$; ✎ *меняем масштаб*
  Считаем значения на границе большого квадрата как среднее значение трёх окружающих узлов плюс случайное приращение:
  для $i = d, \dots, N-d$ с шагом D
    $X_{i,0} = (X_{i+d,0} + X_{i-d,0} + X_{i,d})/3 + rg$;
    $X_{i,N} = (X_{i+d,N} + X_{i-d,N} + X_{i,N-d})/3 + rg$;
    $X_{0,i} = (X_{0,i+d} + X_{0,i-d} + X_{d,i})/3 + rg$;
    $X_{N,i} = (X_{N,i+d} + X_{N,i-d} + X_{N-d,i})/3 + rg$;
  Считаем значения в центрах диагональных квадратов как среднее значение в вершинах этих квадратов плюс случайное приращение:
  для $i = d, \dots, N-d$ с шагом D
    для $j = D, \dots, N-D$ с шагом D
      $X_{i,j} = (X_{i,j+d} + X_{i,j-d} + X_{i+d,j} + X_{i-d,j})/4 + rg$;
  для $i = D, \dots, N-D$ с шагом D
    для $j = d, \dots, N-d$ с шагом D
      $X_{i,j} = (X_{i,j+d} + X_{i,j-d} + X_{i+d,j} + X_{i-d,j})/4 + rg$;
  $D = D/2$;
  $d = d/2$;

Программируем на Maxima (листинг № 3.4):

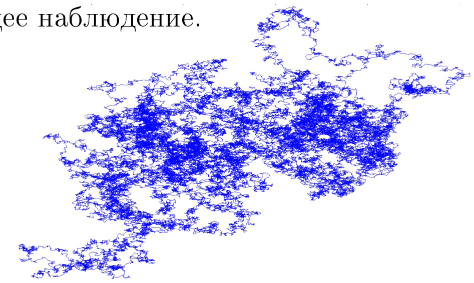
Поверхность ФБД методом срединного смещения

```
load(distrib)$
load(draw)$
H:1$                                   /* параметр размерности поверхности */
s:2$                                   /* параметр вертикального масштаба */
M:6$                                   /* число шагов алгоритма */

X[0,0]:0$                /* инициализация значений в точках первого квадрата */
X[0,N]:0$
X[N,0]:0$
X[N,N]:0$
N:2^M$
r:s$
D:N$                                   /* инициализация шага решётки */
d:N/2$

thru M do
   (
   r:r/2^(H/2),
   for i:d thru N-d step D do
      for j:d thru N-d step D do
         X[i,j]:float((X[i+d,j+d]+X[i+d,j-d]+X[i-d,j+d]+X[i-d,j-d])/4+
                r*random_normal(0,1)),
   r:r/2^(H/2),
   for i:d thru N-d step D do
      (
      X[i,0]:(X[i+d,0]+X[i-d,0]+X[i,d])/3+r*random_normal(0,1),
      X[i,N]:(X[i+d,N]+X[i-d,N]+X[i,N-d])/3+r*random_normal(0,1),
      X[0,i]:(X[0,i+d]+X[0,i-d]+X[d,i])/3+r*random_normal(0,1),
      X[N,i]:(X[N,i+d]+X[N,i-d]+X[N-d,i])/3+r*random_normal(0,1)
      ),
   for i:d thru N-d step D do
      for j:D thru N-D step D do
         X[i,j]:(X[i,j+d]+X[i,j-d]+X[i+d,j]+X[i-d,j])/4+r*random_normal(0,1),
   for i:D thru N-D step D do
      for j:d thru N-d step D do
         X[i,j]:(X[i,j+d]+X[i,j-d]+X[i+d,j]+X[i-d,j])/4+r*random_normal(0,1),

   D:D/2,
   d:d/2
   )$
Y:genmatrix(X,N,N,0,0)$
draw3d(elevation_grid(Y,0,0,1,1),surface_hide = true);
```

Комментарий.

> Здесь цикл `for` используется с дополнительной опцией `step`, означающей шаг инкремента цикла. Кроме того, подумайте, с какой целью используется команда `float` в строчке 22.

Задача 3.1 Пусть $X(t)$ и $Y(t)$ — независимые одномерные броуновские движения. Изобразите путь $f(t) = (X(t), Y(t))$ на графике. Этот путь имеет фрактальную размерность $d = 2$ почти наверное. Сделайте соответствующее наблюдение.



Постройте также путь $f(t) = (X(t), Y(t), Z(t))$ в $\mathbb{R}^3$, где $Z(t)$ — ещё одна реализация броуновского движения. Что можно сказать о фрактальной размерности такого пути?

Задача 3.2 В алгоритмах 3.1–3.2 использовался случайный сдвиг на σg. Будут ли изменения, если заменить этот сдвиг на $g(0, \sigma)$ — случайную величину, распределенную нормально с параметрами $N(0, \sigma)$?

Задача 3.3 Реализуйте алгоритм для двумерного броуновского движения методом случайного срединного смещения. См. указание в [1, гл. 9, с. 268].

Задача 3.4 Реализуйте алгоритм построения аппроксимации ФБД с помощью ковариационной матрицы. Алгоритм состоит в следующем. Необходимо получить аппроксимацию ФБД $X(t)$ в точках $t_1, \dots, t_n$. Для этого:

• Находим матрицу $\Gamma = (r(t_i, t_j))_{i,j=1}^n$, где $r(t, s) = \frac{1}{2}\left(s^{2H} + t^{2H} - |t - s|^{2H}\right)$;

• Находим $\Sigma = \sqrt{\Gamma}$ (т.е. $\Sigma^2 = \Gamma$) — соответствует матрице стандартного отклонения, ассоциированной с ковариационной матрицей Γ;

• Пусть $v = \{g_1, \dots, g_n\}$ – вектор, составленный из n нормальных случайных величин с параметрами $N(0, 1)$;

• Тогда $X = \Sigma v$.

Как вычислить матрицу Σ? Для этого можно использовать разложение Холецкого[13] (попробуйте!). Здесь же укажем другой способ (посредством собственных значений матрицы Γ):

• Поскольку Γ — симметричная положительно-определённая матрица, то её собственные значения λ_i вещественнны и $\lambda_i \geq 0$, $i = \overline{1, n}$;

• Пусть $\Lambda = Diag(\lambda_i)$, $\Lambda_{i,j} = \lambda_i \delta_{ij}$, — диагональная матрица собственных значений, и пусть $\Lambda^{1/2} = Diag(\lambda_i^{1/2})$ — диагональная матрица с элементами $\Lambda_{ij}^{1/2} = \lambda_i^{1/2} \delta_{ij}$ (δ_{ij} — символ Кронекера);

• Пусть P — матрица, столбцами которой являются собственные векторы v_i, отвечающих

[13] Андре-Луи Холе́цкий (1875-1918) — французский математик. Речь идет о представлении симметричной положительно-определённой матрицы A в виде $A = LL^T$, где L — нижняя треугольная матрица со строго положительными элементами диагонали. Такое разложение существует, единственно и алгоритмически несложно.

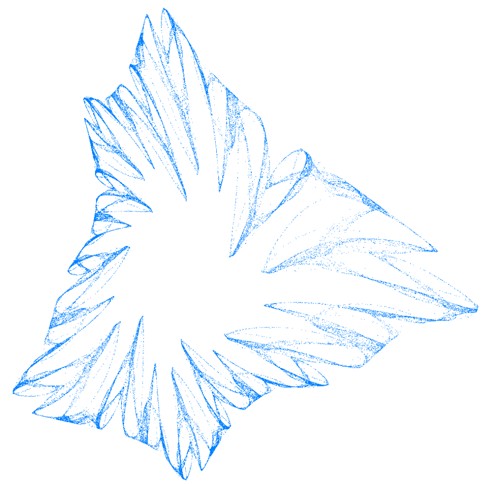
собственных значениям λ_i. Заметим, что матрица P обратима ввиду линейной независимости собственных векторов.
• Тогда $\Sigma = P\Lambda^{1/2}P^{-1}$ (так как $\Gamma = P\Lambda P^{-1}$).

Задача 3.5 Реализуйте алгоритм построения аппроксимации ФБД с помощью гипергеометрической функции Гаусса. Алгоритм состоит в следующем.
Если $X(s)$ — классическое броуновское движение, то ФБД с параметром H можно получить так:
$X_H(t) = \int_0^t K_H(t,s)\,\mathrm{d}B(s)$,
где $K_H(t,s) = \frac{(t-s)^{H-\frac{1}{2}}}{\Gamma(H+\frac{1}{2})}\,{}_2F_1\left(H-\frac{1}{2}, \frac{1}{2}-H, H+\frac{1}{2}, 1-\frac{t}{s}\right)$, Γ — гамма-функция Эйлера. и ${}_2F_1$ — гипергеометрическая функция (ряд) Гаусса:
$${}_2F_1(a,b,c,z) = 1 + \sum_{k=1}^{\infty}\left[\prod_{n=0}^{k-1}\frac{(a+n)(b+n)}{(1+n)(c+n)}\right]z^k = 1 + \frac{ab}{c}\frac{z}{1!} + \frac{a(a+1)b(b+1)}{c(c+1)}\frac{z^2}{2!} + \ldots =$$
$$= \frac{\Gamma(c)}{\Gamma(b)\Gamma(c-b)}\int_0^1 t^{b-1}(1-t)^{c-b-1}(1-tz)^{-1}\,\mathrm{d}t.$$
Предположим, мы хотим получить аппроксимацию ФБД в точках $0 = t_0 < t_1 < \ldots < t_n = T$, тогда:
• Пусть $\delta B_j = g_j\sqrt{T/n}$, $j = \overline{0,n}$ (g_j — независимые нормально распределённые случайные величины с параметрами $N(0,1)$), — приращения броуновского движения на интервале $[0,T]$.
• Тогда для каждого t_j, $j = \overline{0,n}$, подсчитаем
$$X_H(t_j) = \frac{n}{T}\sum_{i=0}^{j-1}\int_{t_i}^{t_{i+1}} K_H(t_j,s)\,\mathrm{d}s\,\delta B_i.$$
Для подсчёта гамма-функции Эйлера можно воспользоваться командой
`gamma(z)` — возвращает значение гамма-функции Эйлера $\Gamma(z) = \int_0^\infty t^{z-1}e^{-t}\,\mathrm{d}t$.
Для подсчёта интеграла можно использовать метод квадратур Гаусса (или численные методы нахождения интегралов в Maxima). Для подсчёта гипергеометрической функции можно считать предел ряда или использовать встроенную в Maxima функцию
`hypergeometric([a,b],[c],z)` — возвращает значение гипергеометрической функции Гаусса ${}_2F_1(a,b,c,z)$.

Задача 3.6 Подумайте, что изменится, если в Алгоритме 3.4 аппроксимации поверхности ФБД методом Фурье-фильтрации вместо квадратной решётки узлов взять прямоугольную.

Рекомендуемая литература: [20], [1, гл. 9]

ПРИЛОЖЕНИЕ



Математика существует не для того, чтобы навязывать кому-либо тяжёлую работу. Наоборот, она существует только для удовольствия. Для удовольствия тех, кто любит анализировать то, что он делает, или может сделать, или то, что уже сделал в надежде сделать это ещё лучше.

Роберт Брингхерст

В этой главе собраны дополнительные сведения, полезные для дальнейшего и более глубокого изучения применения программы Maxima для моделирования фракталов: использование пакета `dynamics`.

I Обзор пакета `dynamics`

С помощью пакета `dynamics` можно строить различные графические представления динамических систем и фракталов:

- паутинная диаграмма;
- бифуркационная диаграмма;
- эволюция орбиты одно- и двумерного отображений;
- «игра в хаос»;
- система итерированных функций, заданная аффинными преобразованиями;
- множества Жюлиа, Мандельброта;

а также, в нём реализован метод Рунге-Кутты 4го порядка для решения систем дифференциальных уравнений.

Данный пакет обладает ограниченными возможностями. Однако рекомендуется изучить его исходный код, находящийся в файле «`dynamics.mac`». Параметры всех команд этого пакета, приведённых ниже, можно изменить непосредственно в тексте этого пакета или скопировать соответствующий кусок кода в свою рабочую область и изменить его. Рассмотрим функции этого пакета.

Напомним, для использования команд данного пакета нужно его загрузить:

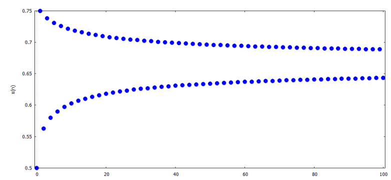
```
load(dynamics)$
```

Для вывода графики команды пакета **dynamics** используют команду **plot2d**, поэтому все опции *options* этой команды можно передавать в команды пакета **dynamics**, например, можно менять графические интерфейсы, различные стили графиков, цвета, менять масштаб осей на логарифмический и т.д. Смотрите список опций команды **plot2d** в справке по Maxima.

✰ **evolution**($F(x), x_0, n, options$) — графическое представление $n+1$ первых точек реккурентной последовательности $x_n = F(x_{n-1})$ с начальным членом x_0: отображение точек $(0, x_0), \ldots, (n, x_n)$. В качестве $F(x)$ может быть любое выражение от одной переменной.

пример

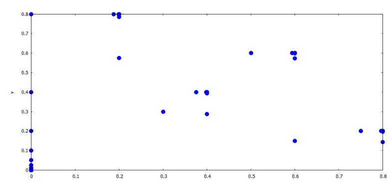
```
evolution(3*y*(1-y),0.5,100)$
```



✰ **evolution2d**(**[**$F(u,v), G(u,v)$**]**, **[**u,v**]**, **[**u_0, v_0**]**,$n, options$) — графическое представление $n+1$ первых точек (u_k, v_k) орбиты двумерной дискретной динамической системы: $u_n = F(u_{n-1}, v_{n-1})$, $v_n = G(u_{n-1}, v_{n-1})$ с начальной точкой (u_0, v_0). В качестве $F(u,v)$ и $G(u,v)$ могут быть любые выражения от двух переменных, эти переменные должны быть явно указаны в команде списком $[u, v]$.

пример

```
evolution2d([mod(2*u,1),0.5*(v+floor(2*u))],
            [u,v],[0.3,0.3],100)$
```



Комментарий. Рассмотренный пример — «преобразование пекаря». Напомним следующие команды:

floor(x) — «антье», целая часть x ($[x]$), также может вычислена с помощью команд:

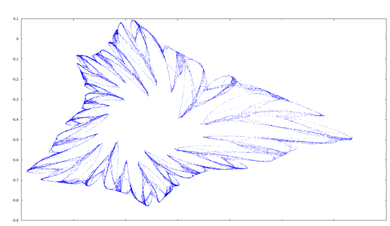
entier(x) — целая часть x,

fix(x) — целая часть x;

mod(x, y) — x по модулю y ($x - y[x/y]$).

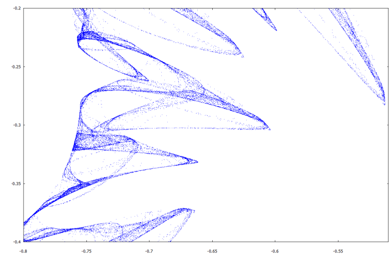
пример

```
f:0.6*x*(1+2*x)+0.8*y*(x-1)-y^2-0.9$
g:0.1*x*(1-6*x+4*y)+0.1*y*(1+9*y)-0.4$
evolution2d([f,g],[x,y],[-0.5,0],50000,
            [style,dots])$
```



пример

```
f:0.6*x*(1+2*x)+0.8*y*(x-1)-y^2-0.9$
g:0.1*x*(1-6*x+4*y)+0.1*y*(1+9*y)-0.4$
evolution2d([f,g],[x,y],[-0.5,0],300000,
   /* зум */ [x,-0.8,-0.6],[y,-0.4,-0.2],
                  [style,dots])$
```

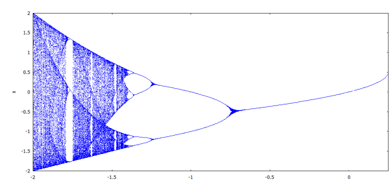


> Комментарий. Рассмотренный пример — динамическая система, приводящею к фрактальному аттрактору. Заметим, что дополнительными опциями `[x,-0.8,-0.6]`, `[y,-0.4,-0.2]`, передаваемыми команде `plot2d`, мы можем обрезать построенные изображения, тем самым увеличивая их фрагменты.

☆ `orbits(`$F(x,c), x_0, n_1, n_2,$`[`$c, c_0, c_1, step$`]`,*options*`)` — рисует бифуркационную диаграмму (диаграмму орбит) для однопараметрического семейства отображений $F(x,c)$ с параметром c. На оси абсцисс отмечается параметр $c \in [c_0, c_1]$, выбранный с шагом *step*. Для каждого значения c вдоль оси ординат отмечается последовательность $\{x_n = F(x_{n-1}, c)\}_{n=n_1+1}^{n_1+n_2+1}$ — кусок орбиты начальной точки x_0. Кроме опций команды `plot2d`, команда `orbits` принимает также опцию *pixels*, устанавливающую максимальное разрешение по вертикали.

пример

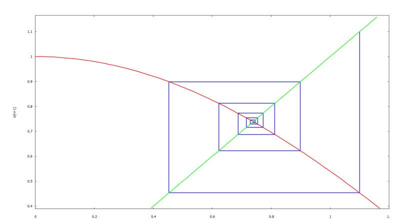
```
orbits(x^2+c,0,50,200,[c,-2,0.25],
/* опции графики */ [style,dots],[nticks,1000])$
```



☆ `staircase(`$F(x), x_0, n, options$`)` — рисует паутинную диаграмму[14] для последовательности $\{x_k = F(x_{k-1})\}_{k=1}^{n}$, заданной отображением F с начальной точкой x_0. См. с. 7.

пример

```
staircase(cos(x), 1.1, 11, [x, 0, 1.2])$
```



☆ `rk(`$F(y,x), y, y^0,$`[`$x, a, b, step$`])`

☆ `rk([`$F_1(y_1,\ldots,y_m),\ldots,F_m(y_1,\ldots,y_m,x)$`],[`$y_1,\ldots,y_m,x$`],[`$y_1^0,\ldots,y_m^0$`],[`$x,a,b,step$`])` — это два варианта одной команды соответственно для решения обыкновенного дифференциального уравнения первого порядка и для решения систем таких уравнений, используя метод Рунге-Кутты 4го порядка.

[14] Staircase — лестничный пролёт (англ.).

Комментарий. Напомним, в чем заключается этот метод. Рассмотрим задачу Коши: $y' = f(x, y)$, $y(x_0) = y_0$. Приближённое значение в последующих точках вычисляется по формуле:

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4),$$

где k_i вычисляются в 4 этапа:
$k_1 = hf(x_n, y_n)$, $k_2 = hf(x_n + \frac{h}{2}, y_n + \frac{k_1}{2})$, $k_3 = hf(x_n + \frac{h}{2}, y_n + \frac{k_2}{2})$, $k_4 = hf(x_n + h, y_n + k_3)$.
Этот метод имеет четвертый порядок точности: суммарная ошибка на интервале $[a, b]$ имеет порядок $O(h^4)$.

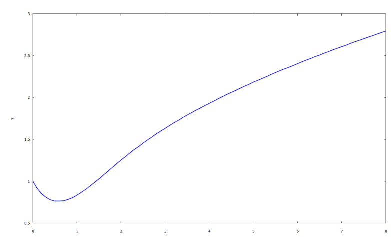
В первом варианте решается уравнение $y' = F(y, x)$. Здесь y — зависимая переменная, область изменения $[a, b]$ независимой переменной x разбивается на сетку узлов с шагом *step*. Начальное значение для y равно y^0. Заметим, что переменные не обязательно должны иметь имена y и x.

Во втором варианте решается система из m ОДУ $\{y'_k = F_k(y_1, \ldots, y_m, x)\}_{k=1}^n$ с m зависимыми переменными $[y_1, \ldots, y_m]$, вектором начальных значений $[y_1^0, \ldots, y_m^0]$ и одной независимой переменной $x \in [a, b]$, дискретизируемой с шагом *step*.

Результат вычислений выводится списком из r значений, полученных на каждой из $r = [(b - a)/step]$ итераций метода, переходящего от одного узла x к другому, если метод не прекратит свои вычисления раньше ввиду слишком большого значения одной из переменных $y_1, \ldots, y_m$. Каждое из r значений само является списком, состоящим из значений независимой переменной, и m значений переменных y_k.

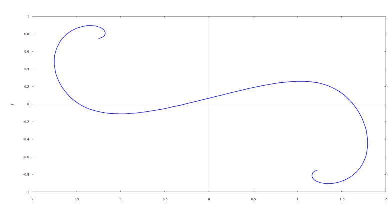
пример

```
sol:rk(t-x^2,x,1,[t,0,8,0.1])$
plot2d([discrete, sol])$
```



пример

```
[F,G]:[4-x^2-4*y^2,y^2-x^2+1]$
sol:rk([F,G],[x,y],[-1.25,0.75],[t,0,4,0.02])$
plot2d([discrete, args(submatrix(sol,1))])$
```



Комментарий. В первом примере решается уравнение $x'_t = t - x^2$ с начальным условием $x(0) = 1$, $t \in [0, 8]$ с шагом 0.1. Во втором примере решается система уравнений $x'_t = 4 - x^2 - 4y^2$, $y'_t = y^2 - x^2 + 1$ с начальными условиями $x(0) = -1.25$ и $y(0) = 0.75$, $t \in [0, 4]$ с шагом 0.02. В обоих случаях результат записывается в массив `sol` и отображается на графике.

Кроме того, напомним следующие команды:

`submatrix`$(r_1, \ldots, r_k, M, c_1, \ldots, c_s)$ — возвращает «подматрицу» матрицы M, полученную из неё выбрасыванием строк с номерами r_i и столбцов с номерами c_j. В случае данного примера решение `sol` состоит из списка, элементы которого — списки из трёх элементов вида [$t, x(t), y(t)$]. При этом координаты $(x(t), y(t))$ представляют точку искомой кривой. Необходимо выбросить первый элемент из каждого трёхэлементного списка, то есть первый столбец из матрицы, представляющей `sol`: `submatrix(sol,1)`.

`args(expr)` — возвращает списком аргументы верхнего уровня у выражения `expr`. В команду `plot2d` необходимо подавать *список* координат точек, а не *матрицу* (не смотря на то, что матрица в Maxima— это список своих строк). Поэтому здесь `args` — удобный способ перевода матрицы в список. Вообще, команда *args* универсальна и используется не только для этого.

Мы опускаем описание следующих команд:
✰`chaosgame`,
✰`ifs`,
✰`julia`,
✰`mandelbrot`,
— они были описаны в Приложении части I настоящего пособия.

РЕСУРСЫ ИНТЕРНЕТА

[1e] http://www.altlinux.org/images/0/0b/MaximaBook.pdf
Хорошее пособие по Maxima Е.А. Чичкарёва [31].

[2e] http://abacus.bates.edu/~sross/
Страница профессора Чипа Росса, посвященная бифуркационным диаграммам, диаграммам орбит и Q-кривым.

[3e] http://maxima.sourceforge.net/ru/
Официальная страница проекта Maxima на русском языке (однако, она редко обновляется, рекомендуется пользоваться английской версией страницы).

[4e] http://sourceforge.net/projects/maxima/files/
Здесь можно скачать установочный файл для Maxima.

[5e] http://maxima.sourceforge.net/ru/documentation.html
Подборка руководств по Maxima с официального сайта, в том числе и на русском языке.

[6e] http://andrejv.github.com/wxmaxima/help.html
Коллекция руководств по Maxima.

[7e] http://www.csulb.edu/~woollett/
Сайт профессора Edwin L. Woollett с очень полезной информацией о Maxima.

[8e] http://www.uneex.ru/static/MethodBooks_Maxima/Maxima.pdf
Пособие Н.А. Стахина «Основы работы с системой аналитических (символьных) вычислений Maxima» на русском языке.

[9e] http://www3.msiu.ru/~lao4/maxima.pdf
Методическое пособие с неофициального сайта МГИУ `www3.msiu.ru`.

[10e] http://oeis.org/A006890
Энциклопедия всевозможных констант, страница константы Фейгенбаума.

[11e] http://www.1-1-1.ir/pdf/feigenbaum-constant-to-1000-digits.pdf
Константа Фейгенбаума, сосчитанная до 1000 знаков, Hamid Naderi Yeganeh.

СПИСОК ЛИТЕРАТУРЫ

(После источников указаны страницы

настоящего пособия, на которых они упоминаются)

[1] *Кроновер, Р. М.* Фракталы и хаос в динамических системах. Основы теории / Р. М. Кроновер. — М. : Постмаркет, 2000. — 352 с. с. 4, с. 11, с. 12, с. 16, с. 27, с. 36, с. 37

[2] *Jackson, E. A.* Perspectives of Nonlinear Dynamics / E. A. Jackson. — New York : Cambridge University Press, 2008. — 496 p. с. 9, с. 12, с. 15, с. 16

[3] *Peitgen, H.-O.* Chaos and Fractals, New Frontiers of Science / H.-O. Peitgen, H. Jürgens, D. Saupe. — New York : Springer-Verlag, 1992. — 984 p. с. 9

[4] *Coullet, P.* Iterations d'endomorphismes et groupe de rénormalisation / P. Coullet, C. Tresser // *Comptes Rendus de l'Académie des Sciences.* — 1978. — Vol. 287A. — Pp. 577–580. с. 10

[5] *Coullet, P.* Itérations d'endomorphismes et groupe de rénormalisation / P. Coullet, C. Tresser // *Journal de Physique Colloques.* — 1978. — Vol. 39:C5. — Pp. 25–28. с. 10

[6] *Lanford III, O. E.* A computer-assisted proof of the feigenbaum conjectures / O. E. Lanford III // *Bull. Amer. Math. Soc.* — 1984. — Vol. 6:3. — Pp. 427–434. с. 10

[7] *Lyubich, M.* Feigenbaum-Coullet-Tresser universality and Milnor's hairiness conjecture / M. Lyubich // *Annals of Mathematics.* — 1999. — Vol. 149. — Pp. 319–420. с. 10

[8] *Feigenbaum, M.* Quantitative universality for a class of nonlinear transformations / M. Feigenbaum // *Journal of Statistical Physics.* — 1978. — Vol. 19, no. 1. — Pp. 25–52. с. 10, с. 11

[9] *Feigenbaum, M.* Universal behavior in nonlinear systems / M. Feigenbaum // *Los Alamos Science.* — 1980. — Pp. 4–27. с. 10

[10] *Metropolis, N.* On finite limit sets for transformations on the unit interval / N. Metropolis, M. L. Stein, P. R. Stein // *J. Combinatorial Theory.* — 1973. — Vol. 15(1):25. — Pp. 25–44. с. 11

[11] *Singer, D.* Stable orbits and bifurcations of the maps in the interval / D. Singer // *SIAM Journal on Applied Mathematics.* — 1978. — Sep. — Vol. 35, no. 2. — Pp. 260–267. с. 11, с. 12

[12] *Feigenbaum, M.* The universal metric properties of nonlinear transformations / M. Feigenbaum // *Journal of Statistical Physics.* — 1979. — Vol. 21, no. 6. — Pp. 669–706. с. 13, с. 16

[13] *Briggs, K. M.* A precise calculation of the feigenbaum constants / K. M. Briggs // *Mathematics of computation.* — 1991. — Vol. 57, no. 195. — Pp. 435–439. с. 13, с. 16

[14] *Ross, C.* The shadow-curves of the orbit diagram permeate the bifurcation diagram, too / C. Ross, M. Odell, S. Cremer // *International Journal of Bifurcation and Chaos.* — 2009. — Vol. 19, no. 9. — Pp. 3017–3031. с. 16

[15] *Глейк, Д.* Хаос. Создание новой науки / Д. Глейк. — Спб. : Амфора, 2001. — 398 с. с. 16

[16] *McGuire, J. B.* Asymptotic properties of sequences of iterates of nonlinear transformations / J. B. McGuire, C. J. Thompson // *Journal of Statistical Physics.* — 1982. — Vol. 27, no. 1. — Pp. 183–200. с. 16

[17] *Фейгенбаум, М.* Универсальность в поведении нелинейных систем / М. Фейгенбаум // *Успехи физических наук.* — 1983. — Октябрь. — Т. 141, № 2. — С. 343–374. с. 16

[18] *Вул, Е. Б.* Универсальность Фейгенбаума и термодинамический формализм / Е. Б. Вул, Я. Г. Синай, К. М. Ханин // *Успехи математических наук.* — 1984. — Т. 39, № 3 (237). — С. 3–37. с. 16

[19] *Briggs, K. M.* How to calculate the feigenbaum constants on your PC / K. M. Briggs // *Austral. Math. Soc. Gazette.* — 1989. — Vol. 16. — Pp. 89–92. с. 16

[20] *Liebovitch, L. S.* Fractals and Chaos Simplified for the Life Sciences / L. S. Liebovitch. — New York : Oxford University Press, 1998. — 288 p. с. 18, с. 22, с. 37

[21] Брауновское движение. А. Эйнштейн, М. Смолуховский: сб. ст. / Под ред. В. И. Давыдова. — М.–Л. : ОНТИ, 1936. — 608 с. — [пер. с нем. и франц., с доп. статьями Ю. А. Крутикова и Б.И. Давыдова]. `http://www.mirgorodsky.ru/mirgorodskiyal_statya/O_DVIGENII_VZVESHENNIH_V_POKOJASCHEISYA_JZIDKOSTI_CHASTITC_EINSHTEIN_1905.pdf`. с. 23

[22] *Brown, R.* A brief account of microscopical observations made in months of June, July, and August, 1827, on the particles contained in the pollen of plants; and on the general existence of active molecules in organic and inorganic bodies / R. Brown // *Edinburgh new Philosophical Journal.* — 1828. — Vol. 5. — Pp. 358–371. `http://sciweb.nybg.org/science2/pdfs/dws/Brownian.pdf`. с. 23

[23] *Перрен, Ж.* Атомы / Ж. Перрен. Современные проблемы естествознания. — М. : Госиздат, 1921. — 254 с. — [пер. с англ., 1916 г.]. http://www.archive.org/stream/atomsper00perruoft. с. 23

[24] *Wiener, N.* Differential space / N. Wiener // *J. Math. Phys.* — 1923. — Vol. 2. — Pp. 131–174. с. 23, с. 25

[25] *Wiener, N.* Un problème de probabilités dénombrables / N. Wiener // *Bull. Soc. Math. France.* — 1924. — Vol. 52. — Pp. 569–578. с. 23, с. 25

[26] *Kolmogorov, A. N.* Grundbegriffe der Wahrscheinlichkeitrechnung / A. N. Kolmogorov. — Berlin : Springer-Verlag, 1933. — Vol. 2:3 of *Ergebnisse der Mathematik.* — 62 p. с. 23

[27] *Lévy, P.* Processus Stochastiques et Mouvement Brownien / P. Lévy. — Paris : Gauthier-Villars, 1948. — 367 p. с. 23

[28] *Karatzas, I.* Brownian Motion and Stochastic Calculus (secon edition) / I. Karatzas, S. E. Shreve. — New York ; Berlin : Springer-Verlag, 1991. — 493 p. с. 23

[29] *Mandelbrot, B. B.* Fractional brownian motions, fractional noises and applications / B. B. Mandelbrot, J. W. V. Ness // *SIAM Review.* — 1968. — October. — Vol. 10, no. 4. — Pp. 422–437. с. 24, с. 26, с. 27

[30] *Peitgen, H.-O.* The Science of Fractal Images / H.-O. Peitgen, D. S. et al.; Ed. by H.-O. Peitgen, D. Saupe. — Tokyo ; New York : Springer-Verlag, 1988. — 312 p. с. 30

[31] *Чичкарёв, Е. А.* Компьютерная математика с Maxima: Руководство для школьников и студентов / Е. А. Чичкарёв. Библиотека ALT Linux. — М. : ALT Linux, 2012. — 384 с. http://www.altlinux.org/images/0/0b/MaximaBook.pdf. с. 43

[32] *Barnsley, M. F.* Fractals everywhere, second edition / M. F. Barnsley. — Boston : Academic Press, 1993. — 550 p.

[33] *Barnsley, M. F.* Superfractals / M. F. Barnsley. — Cambridge : Cambridge University Press, 2006. — 453 p.

УКАЗАТЕЛЬ КОМАНД

Замечания и пожелания можно отправлять автору по адресу Paul.Troshin@gmail.com