МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ СИСТЕМ І ПРОЦЕСІВ

Лекція 6. Введення в моделювання динамічних систем. Нерухомі точки, точки біфуркації.

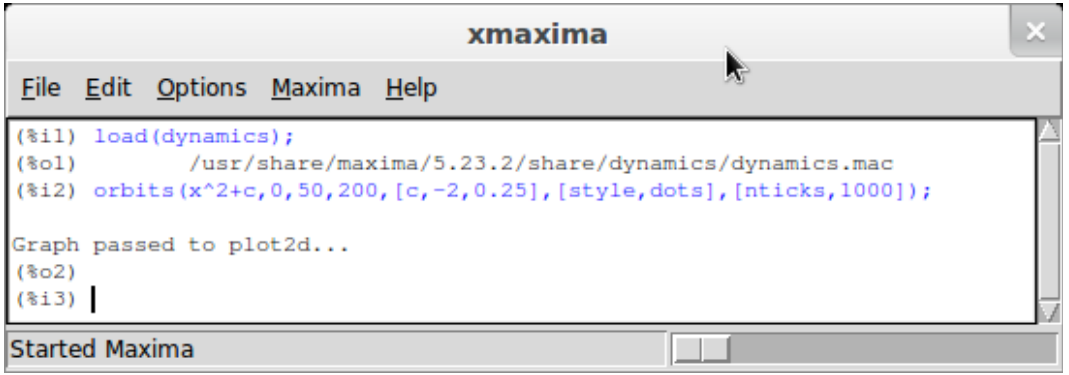
Київ – 2015

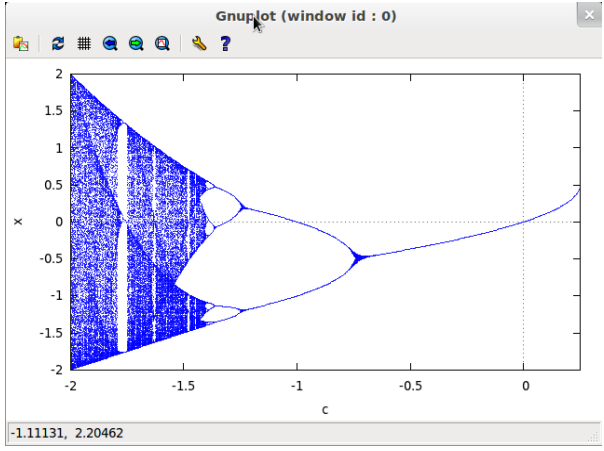
Во второй части нашего курса будут изложены методы построения паутинных и бифуркационных диаграмм, а также диаграмм орбит, построение странных аттракторов, кривых и поверхностей фрактального броуновского движения. Содержание этой части сфокусировано на динамических аспектах фрактальной геометрии, или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

В качестве среды моделирования выбрана система компьютерной алгебры Maxima.

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

```
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.

В этой лекции мы рассмотрим 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])$
```

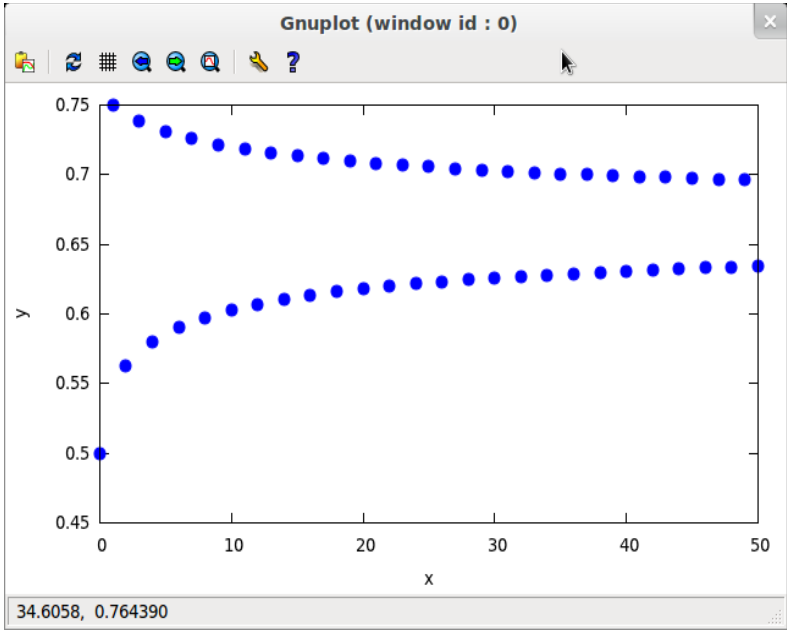
Отобразим данную последовательность на графике:

пример

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

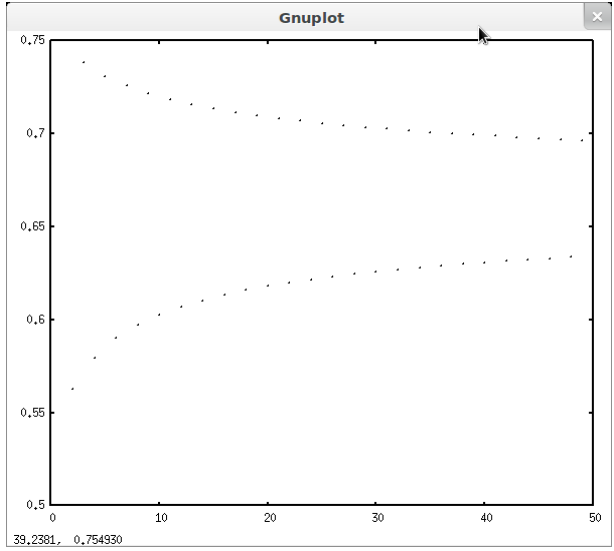
File Edit Options Maxima Help

```
(%i3) f(x):=3*x*(1-x);
(%o3)                         f(x) := 3 x (1 - x)
(%i4) x[0]:0.5;
(%o4)                                  0.5
(%i5) for i:0 thru 50 do x[i+1]:f(x[i]);
(%o5)                                  done
(%i6) orb:makelist([i,x[i]],i,0,50));
incorrect syntax: Too many )'s
st([i,x[i]],i,0,50))
                  ^
(%i6) orb:makelist([i,x[i]],i,0,50);
(%o6) [[0, 0.5], [1, 0.75], [2, 0.5625], [3, 0.73828125],
[4, .5796661376953125], [5, .7309599195141345],
[6, .5899725467340735], [7, .7257148225025549],
[8, .5971584567079203], [9, .7216807028704055],
[10, .6025729979246489], [11, .7184363402902498],
[12, .6068566957218066], [13, .7157449397382518],
[14, .6103623629320143], [15, 0.713460446544187],
[16, .6133039132834688], [17, .7114866697039566],
[18, .6158201656125887], [19, .7097570677124175],
[20, .6180059176340649], [21, .7082238102100269],
[22, .6199285345848561], [23, 0.706851439776987],
[24, .6216374455865626], [25, .7056129954935281],
[26, .6231698882525353], [27, .7044875358835738],
[28, .6245545430047923], [29, .7034584974506021],
[30, .6258139194454301], [31, .7025125730213365],
[32, .6269659733048335], [33, 0.701638924868269],
[34, .6280252319339057], [35, .7008286199648089],
```

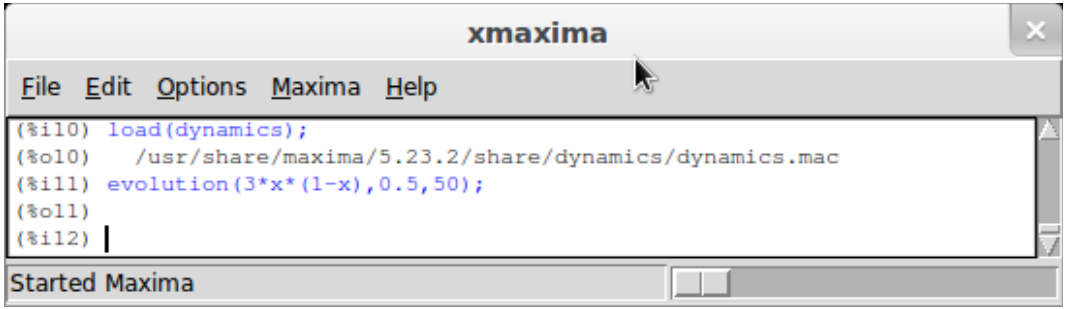

Gnuplot (window id : 0)
0.75
0.7
0.65
0.6
0.55
0.5
0.45
y
0
10
20
30
40
50
x
34.6058, 0.764390

Комментарий. Последнюю строчку можно заменить на

```
load(draw)$
draw2d(point_type=filled_circle,points(orb))$
```



Гораздо проще для аналогичного представления орбит использовать команду evolution пакета dynamics:

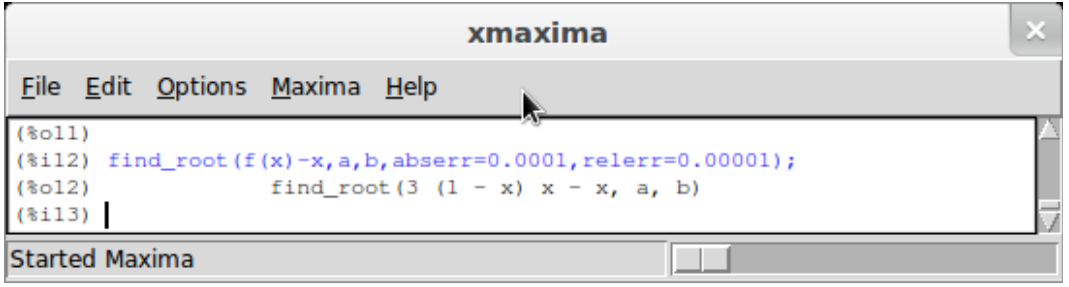


Неподвижная точка x_f называется *притягивающей (аттрактором)*[2], если существует её окрестность $U(x_f)$, для любой точки $x \in U(x_f)$ которой $\lim\limits_{n\to\infty} f^{(n)}(x) = x_f$. Если функция f дифференцируема, то в притягивающей неподвижной точке $f(x_f) = x_f, |f'(x_f)| < 1$.

Неподвижная точка называется *отталкивающей (репеллером)*, если для любой точки $x \in \mathring{U}(x_f)$ 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)`



* Проколотой окрестностью точки называется окрестность точки, из которой исключена эта точка.

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)
```

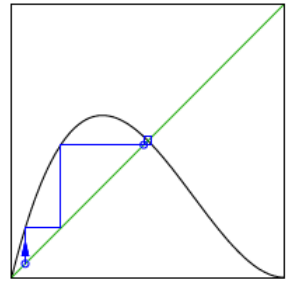
Сосчитаем модуль производной функции $f(x)$ для нахождения типа неподвижной точки $x_f = a$:

```
at(abs(diff(f(x),x)),[x=a])
```

`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 $[a, b]$.

Построение паутинной диаграммы

```
1  load(draw)$
2  [a,b]: [0,1]$                          /* левая и правая границы: a<=x<=b */
3  x0: 0.05$                                              /* начальная точка */
4  level: 10$                                             /* число итераций */
5  f(x):=4*x(1-x)$                                        /* исходная фунция */
6
7  path:[[x0,x0]]$                                        /* массив вершин ломаной */
```

```
8
9   thru level do(                                          /* построение ломаной path */
10      y0: f(x0),
11      path: append(path,[[x0,y0],[y0,y0]]),
12      x0: y0
13  )$
14  draw2d(
15      color=black,
16      explicit(f(x),x,a,b),                                  /* отображаем функцию f */
17      color="#23AB0F",
18      explicit(x,x,a,b),                                  /* отображаем диагональ y=x */
19      color=blue,
20                              /* отображаем концевые точки ломаной окружностями: */
21      point_type=circle,point_size=1, points([path[1],last(path)]),
22      point_type=dot,points_joined=true,points(path),        /* отображаем ломаную */
23            /* отображаем вектор направления итераций вдоль первого ребра ломаной */
24      head_length = 0.05, head_angle=13, vector(path[1],0.6*(path[2]-path[1])),
25      user_preamble="set size ratio -1"
26  )$
```

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"`;

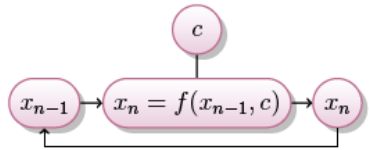
`explicit(`$f(x)$`,x,a,b)` — график явно заданной функции $f(x)$, $x \in$ [`a`, `b`];

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 $c \in C$:

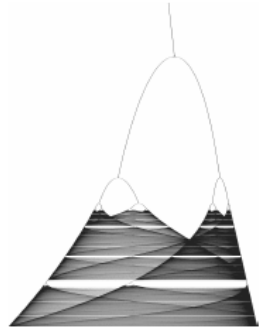
$$f : U \times C \to U, \quad U, C \subset \mathbb{R},$$
$$x_n = f(x_{n-1}, c), \quad n = 1, 2, \ldots, \quad x_0 \in U. \tag{1.2}$$



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 ($n \in \mathbb{N}$): $|\frac{d}{dx} f^{(2^n)}(x,c)| < 1$.

---

5 Например, для отображения $f(x) = 4x(1 - x)$ есть простая формула $x_n = \sin^2(2^n \pi\theta)$,

$(\theta = \frac{1}{\pi} \sin^{-1} \sqrt{x_0})$

Алгоритм 1.2: **Построение бифуркационной диаграммы**

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

Выход: изображение бифуркационной диаграммы.

1: $path = \emptyset$; ✎ *инициализация массива точек графика*
2: $c = c_1$; ✎ *значение c будет меняться от c_1 до c_2*
3: **повторять** $prec$ **раз**
4: $\quad c = c + \frac{c_2 - c_1}{prec}$;
5: $\quad$ **повторять** $skip$ **раз**
6: $\quad\quad x_0 = f(x_0, c)$;
7: $\quad$ **повторять** N **раз**
8: $\quad\quad x_0 = f(x_0, c)$;
9: $\quad\quad path = path \cup \{x_0, c\}$
10: 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 функция */

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

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

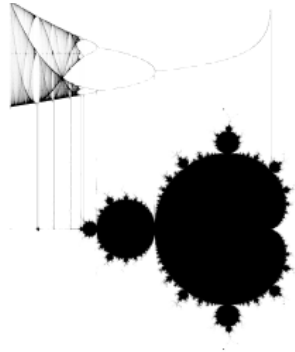
Митчелл Фейгенбаум[10] [8, 9] описал механизм «получения хаоса с помощью удвоения периода» и заметил, что он возникает не только при итерациях логистического отображения, но и для других двузначных отображений интервала в себя, таких как $x^2 + c$, $c \sin \pi x$, $cx^2 \sin \pi x$.

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]:

$$\delta = \lim_{n\to\infty} \frac{c_n - c_{n-1}}{c_{n+1} - c_n} = 4,669201609102990671853203820466201617258185577475768632745651343004134330211314737138689744023948013817 16\ldots$$

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.

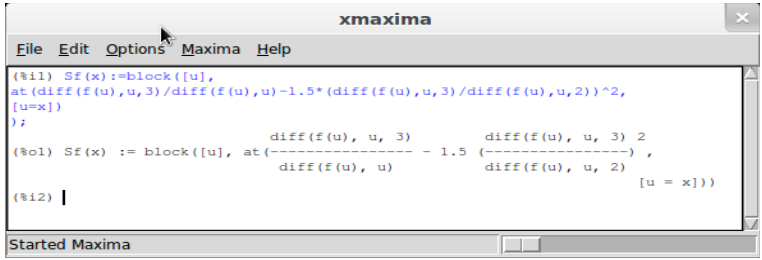
В настоящее время не известны необходимые и достаточные критерии, позволяющие утверждать, что произвольное семейство отображений $f(x, c)$ обладает вышеупомянутыми свойствами. Однако частичные ответы, среди каких функций искать такие отображения, дают следующая теорема Давида Зингера и условия соответствующих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



`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.*

Можно доказать, что

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

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

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

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

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

Тогда

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

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

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

$\tilde{c}_0$ — 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 c, 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

12: $C_{old} = C;$ ✎ *в этом цикле ищем корень* $B_{2i}(C) = x_0$ *методом Ньютона*

13: $C = C - \frac{B(C) - x_0}{D(C)};$ ✎ *итерации методом Ньютона:* $\tilde{c}_n = \tilde{c}_{n-1} - \frac{B_{2i}(\tilde{c}_{n-1}) - x_0}{B'_{2i_c}(\tilde{c}_{n-1})}$

14: **пока не** $\frac{|C - C_{old}|}{C} < \varepsilon$

15: $c_i^* = C;$

16: $\delta_i = \frac{c_{i-1}^* - c_{i-2}^*}{c_i^* - c_{i-1}^*};$

Нахождение константы Фейгенбаума

```
1  f(x):=c-x^2$                                   /* изучаемая функция */
2  x0:0$                                  /* единственная критическая точка функции f */
3  [c[0],c[1],d[1]]:[0,1,3.2]$                    /* начальные приближенные значения */
4
5  B:f(f(x0))$                                    /* начальные значения для B и D */
6  D:1-2*f(x0)$
7  n:8$                                  /* количество точек для подсчета предела d */
8  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      )$
```

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*.

ДОМАШНЄ ЗАВДАННЯ

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?