МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ СИСТЕМ І ПРОЦЕСІВ

Лекція 5. Особливості програмування фракталів та динамічних систем

Київ – 2015

I Сохранение графики и анимация в Maxima

Как в формате `gnuplot`, так и в `xmaxima` полученное изображение можно сохранить нажатием правой клавиши мыши на нём и выбирая соответствующий пункт меню. Заметим лишь, что изображения, получаемые при помощи команд `wxplot2d` и `wxdraw2d` имеют маленькое разрешение при таком сохранении.

Заметим, что при выводе графики посредственно в рабочую область командами `wxplot` и `wxdraw` опция перенаправления на терминал не работает, потому что этими командами графика выводится на рабочую область программы.

Для сохранения графики можно воспользоваться следующими командами:

пример

```
draw(terminal=png,                                    /* выбор типа файла */
   file_name="c:\\temp\\picture",  /* имя целевого файла, без расширения! */
   gr2d(explicit(sin(x),x,0,1)))$                     /* график,с опциями */
```

что эквивалентно коду:

пример

```
draw(terminal=png,
   file_name="c:\\temp\\picture",
   explicit(sin(x),x,0,1))$
```

Команды `gr2d` и `gr3d` задают соответственно дву- и трёхмерные сцены для отрисовки их командой `draw`. Команда `draw` может отрисовать сразу несколько сцен. Для отрисовки одной сцены связка команд `draw`–`grd2` эквивалентна команде `draw2d` (то же самое для `gr3d`).

В качестве параметров для сохранения графики можно использовать параметр `dimensions=[`*xpixels*`,`*ypixsels*`]`, указывающий на разрешение целевого файла *xpixels* × *ypixels*. Наиболее распространенные терминалы графических файлов: `png`, `eps`, `jpg`, `eps_color`, `pdf`, `gif`, `animated_gif`, `svg`.

Еще один совет: когда сохраняете много изображений, желательно делать подписи на них (какая функция, какие параметры использовались). Это можно сделать опциями `title` и `key`. При этом удобно использовать команду `string(t)` — она конвертирует числовое значение `t` в строчное. Пример (русские буквы не всегда получается использовать):

пример

```
t:%pi$
draw2d(title="График функции sin(x+t)",
key=concat("Значение параметра t=",string(t)),
explicit(sin(x+t),x,0,1))$
```

Анимацию (видео или анимированное изображение формата GIF) можно создать из серии уже сохранённых изображений. Мы разберем, как это можно сделать с помощью бесплатной программы работы с графикой из командной строки Image Magick [13e]. Вызов ее из командной строки (чтобы не запутаться, обозначим пробел символом ␣):

пример

```
convert␣-delay␣10␣c:\\temp\\*.png␣c:\\temp\\anim.gif)$
```

- `convert` — команда Image Magick
- `-delay 10` — задержка между кадрами
- `c:\\temp\\*.png` — кадры для сборки: все файлы png с таким адресом
- `c:\\temp\\anim.gif` — файл-результат

пример

```
for t:1 thru 20 do
   draw2d(terminal=png,
          file_name=concat("c:\\temp\\ga",add_zeroes(t)),
          explicit(sin(x+2*%pi/20*t),x,0,2*%pi)
          )$     /* конвертация с помощью Image Magick: */
system("convert -delay 10 c:\\temp\\*.png c:\\anim.gif")$
```

Здесь команда `system("`*cmd*`")` вызывает команду "*cmd*" в командной строке операционной системы. Команда `concat(`x, y`)` присоединяет элемент y справа к элементу x (команда `sconcat` возвращает результат в виде строки). Команда `add_zeroes(`t`)`, где t — строка или число, конвертируемое в строку, — возвращает строку из 10 символов, дополняя строку t слева нулями (например `add_zeroes(abc)`$\mapsto$ 0000000*abc*), а если длина строки t превосходит 9, то возвращается сама строка t. Вместо расширения AVI можно указать MPG — тогда получится файл видео.

Сделаем анимированное изображение семейства графиков функций

$$f(x,t)=\sqrt{\cos x}\cdot\cos\left((200|t|+25)x\right)+\sqrt{|x|}-0.7(4-x^x)^{0.01},\qquad x\in[-1.8,1.8],$$

при разных значениях параметра $t\in[-1,1]$. Сохраним его в файл `C:\\temp\\anim.gif`:

пример

```
load(draw)$
f(x,t):=sqrt(cos(x))*cos((abs(t)+25)*x)+sqrt(abs(x))-0.7*(4-x*x)^0.01$
g:makelist(gr2d(explicit(sin(f(x,t)),x,-1.8,1.8),noframe),t,-1,1,0.1)$
draw(terminal=animated_gif, /* тип терминала графики: анимированный GIF */
     delay=20,              /* Задержка между кадрами в миллисекундах */
     file_name="C:\\temp\\anim",     /* проверьте, существует ли папка! */
     g)$     /* массив графических объектов - графиков семейства f(x,t) */
```

Здесь пользовательская опция `noframe` введена в комментарии на с. 9. Имя файла `anim` должно быть без расширения: оно дополняется выбором терминала выше.

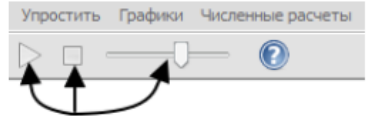
Еще один вариант осуществления анимации возможен в оболочке wxMaxima:

пример

```
with_slider_draw(t,makelist(i,i,1,20),noframe,explicit(f(x,t),x,-1.8,1.8))$
```

t — параметр кадров; опции; кадры графики

После того, как wxMaxima закончила вычисления и отобразила рисунок, выделите этот рисунок мышью — и в верхней панели инструментов активируются три элемента для управления анимацией.



II Обзор пакета `fractals`

С помощью пакета `fractals` автора José Ramírez Labrador можно строить известные фракталы:

- треугольник Серпинского, фракталы «Дерево», «Папоротник»;
- множество Мандельброта и множества Жюлиа;
- снежинки Коха;
- отображения Пеано: кривые Серпинского и Гильберта.

Данный пакет обладает ограниченными возможностями. Однако рекомендуется изучить его исходный код, находящийся в файле «`fractals.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.

☆ `sierpinskiale(`n`)` — возвращает массив из координат $n-1$ случайной точки, принадлежащей треугольнику Серпинского, получающемуся рандомизированным Алгоритмом (см. секцию 2.2, с. 21). Функции этой СИФ:

$$f_1\begin{bmatrix}x\\y\end{bmatrix}=0.5\begin{bmatrix}x\\y\end{bmatrix},\ f_2\begin{bmatrix}x\\y\end{bmatrix}=0.5\begin{bmatrix}x\\y\end{bmatrix}+\begin{bmatrix}1\\0\end{bmatrix},\ f_3\begin{bmatrix}x\\y\end{bmatrix}=0.5\begin{bmatrix}x\\y\end{bmatrix}+\begin{bmatrix}0\\1\end{bmatrix}.$$

пример

```
load(fractals)$
plot2d([discrete,sierpinskiale(10000)], [style,dots])$
```

☆ `treefale(`n`)` — возвращает массив из координат $n-1$ случайной точки, принадлежащей древовидному аттрактору СИФ, получающемуся рандомизированным алгоритмом. Функции этой СИФ:

$$f_1\begin{bmatrix}x\\y\end{bmatrix}=\tfrac{3}{4}\begin{bmatrix}0\\y\end{bmatrix},\ f_2\begin{bmatrix}x\\y\end{bmatrix}=\tfrac{3}{4}\begin{bmatrix}\cos\frac{\pi}{6} & -\sin\frac{\pi}{6}\\ \sin\frac{\pi}{6} & \cos\frac{\pi}{6}\end{bmatrix}\cdot\begin{bmatrix}x\\y\end{bmatrix}+\begin{bmatrix}0\\1\end{bmatrix},\ f_3\begin{bmatrix}x\\y\end{bmatrix}=\tfrac{3}{4}\begin{bmatrix}\cos\frac{\pi}{6} & \sin\frac{\pi}{6}\\ -\sin\frac{\pi}{6} & \cos\frac{\pi}{6}\end{bmatrix}\cdot\begin{bmatrix}x\\y\end{bmatrix}+\begin{bmatrix}0\\1\end{bmatrix}.$$

☆ `fernfale(`n`)` — возвращает массив из координат $n-1$ случайной точки, принадлежащей древовидному аттрактору СИФ, получающемуся рандомизированным алгоритмом (здесь вероятности выбора функции зависят от коэффициентов сжатия). Функции этой СИФ:

$$f_1\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}0\\0.16y\end{bmatrix},\ f_2\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}0.85 & 0.04\\-0.04 & 0.85\end{bmatrix}\cdot\begin{bmatrix}x\\y\end{bmatrix}+\begin{bmatrix}0\\1.6\end{bmatrix},\ f_3\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}0.2 & -0.26\\0.23 & 0.22\end{bmatrix}\cdot\begin{bmatrix}x\\y\end{bmatrix}+\begin{bmatrix}0\\1.6\end{bmatrix},$$

$$f_4\begin{bmatrix}x\\y\end{bmatrix}=\begin{bmatrix}-0.15 & 0.28\\0.26 & 0.24\end{bmatrix}\cdot\begin{bmatrix}x\\y\end{bmatrix}+\begin{bmatrix}0\\0.44\end{bmatrix}.$$

☆ `mandelbrot_set(`x, y`)` — возвращает число $n \leq 29$ итераций, для которого $|f_{x+y\mathbf{i}}^{(n)}(0)| \geq 100$, либо число 30, если за 29 итераций орбита точки 0 осталась внутри круга $|z| < 100$. Напомним, множеству Мандельброта $\mathcal{M}$ принадлежат точки $c = x + y\mathbf{i}$, для которых орбита нуля ограничена (см. секцию 4.2, с. 41).

Эту функцию можно использовать для раскрашивания областей по скоростям убегания точек, например, так:

пример

```
load(fractals)$
plot3d(mandelbrot_set, [x, -2.5, 1], [y, -1.5, 1.5],
[gnuplot_preamble, "set view map"],[grid, 150, 150])$
```

Здесь опция `grid` указывает на то, что область изменения координат $[-2.5, 1] \times [-1.5, 1.5]$ должна быть разделена на 150×150 точек (x, y), в которых будет вычисляться функция `mandelbrot_set(x,y)`. Опция `gnuplot_preamble` используется для интерпретации значения функции `mandelbrot_set(x,y)` как цвета точки `(x,y)` (попробуйте убрать эту опцию).

☆ `julia_set(`x, y`)` — возвращает число $n \leq 29$ итераций, для которого $|f_c{}^{(n)}(x+y\mathbf{i})| \geq 100$, либо число 30, если за 29 итераций орбита точки $x + y\mathbf{i}$ осталась внутри круга $|z| < 100$. Здесь число c по умолчанию задается переменной `julia_parameter:%i`. Напомним, заполняющему множеству Жюлиа принадлежат точки $x+y\mathbf{i}$, орбита которых ограничена (см. секцию 4.1, с. 33). Применять эту команду можно так:

пример

```
load(fractals)$
plot3d(julia_set, [x, -2, 1], [y, -1.5, 1.5],
[gnuplot_preamble, "set view map"],[grid, 150, 150])$
```

Автор данного пакета советует опробовать этот код на следующих числах `julia_parameter`: $-0.745 + 0.113002\mathbf{i}$, $-0.39054 - 0.58679\mathbf{i}$, $-0.15652 + 1.03225\mathbf{i}$, $-0.194 + 0.6557\mathbf{i}$, $0.011031 - 0.67037\mathbf{i}$.

☆ `julia_sin(`x, y`)` — то же, что и предыдущая команда, только для функции $f(z) = c\sin z$. Автор рекомендует параметр `julia_parameter:1+0.1*%i$`.

☆ `snowmap(`*vert*, *iter*`)` — возвращает массив вершин снежинки Коха после *iter* итераций, построенной на сторонах ломаной линий, заданной массивом вершин *vert* в виде комплексных чисел. Если первая и последняя вершины совпадают, то ломаная линия замкнута. Результирующее множество расположено слева от ломаной линии, считая в направлении от начала к ее концу. Для построения используется рекурсивный алгоритм. Пример использования (подумайте, в чем разница):

пример

```
plot2d([discrete,snowmap([1,exp(%i*%pi*2/3),exp(-%i*%pi*2/3),1],4)])$
```

☆ `hilbertmap(`*iter*`)` — возвращает массив вершин непрерывной кривой Гильберта, плотно заполняющей плоскую область, после *iter* итераций.

пример

```
plot2d([discrete,hilbertmap(4)])$
```

☆ `sierpinskimap(`*iter*`)` — возвращает массив вершин непрерывной кривой Серпинского, плотно заполняющей плоскую область, после *iter* итераций.

пример

```
plot2d([discrete,sierpinskimap(4)])$
```

III 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.

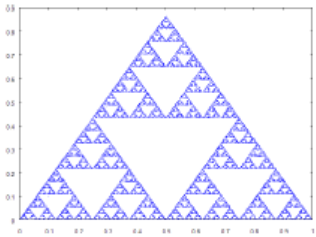
Напомним, для использования команд данного пакета нужно его загрузить:

```
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.

☆ `chaosgame([[`x_1`,`y_1`],...,[`x_m`,`y_m`]],[`x_0`,`y_0`],b,n,options)` — реализует «игру в Хаос»: отмечается точка (x_0, y_0),

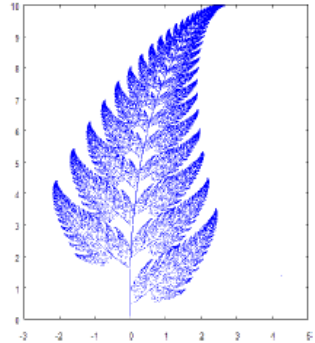
```
chaosgame([[0,0],[1,0],[0.5,sqrt(3)/2]],
          [0.1,0.1],1/2,30000,[style,dots])$
```



☆ `ifs([`r_1`,...,`r_m`],[`A_1`,...,`A_m`],[[`x_1`,`y_1`],...,[`x_m`,`y_m`]],[`x_0`,`y_0`],n,options)` — реализует построение аттрактора системы итерированных функций (x_0, y_0),

```
a1:matrix([0.85,0.04],[-0.04,0.85])$
a2:matrix([0.2,-0.26],[0.23,0.22])$
a3:matrix([-0.15,0.28],[0.26,0.24])$
a4:matrix([0,0],[0,0.16])$
p1:[0,1.6]$
p2:[0,1.6]$
p3:[0,0.44]$
p4:[0,0]$
r: [85,92,99,100]$
ifs(r,[a1,a2,a3,a4],[p1,p2,p3,p4],
          [5,0],50000,[style,dots])$
```

пример



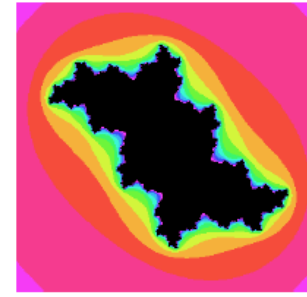
☆ `julia(`x`,`y`,options)` — создаёт изображение множества Жюлиа с параметром $c = x + iy$, $x, y \in \mathbb{R}$, и сохраняет в файле в формате XPM в директорию установки Maxima (см. ниже опцию `filename`).

Точки вне множества Жюлиа окрашиваются в различные цвета в зависимости от количества итераций, после которых орбита выходит из круга радиуса 2. Максимальное количество итераций определяется опцией `levels`. Если после заданного количества итераций орбита точки остается внутри круга радиуса 2, точка окрашивается в цвет, заданный опцией `color`. Все цвета, в которые раскрашиваются точки вне множества Жюлиа, имеют одинаковую насыщенность (*saturation*) и яркость (*value*), но разные углы оттенка (*hue*), равномерно распределённые между *hue* и *hue* + *huerange*.

Доступные опции, дополнительные к опциям пакета `plot2d`, опишем в конце секции.

пример

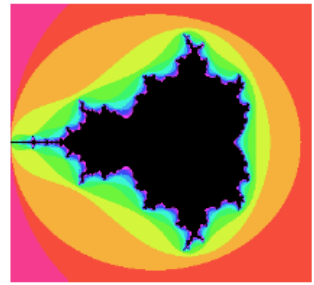
```
julia(-0.11,0.65569999)$
```



☆ `mandelbrot(options)` — создаёт изображение множества Мандельброта и сохраняет в файле в формате XPM. Окрашивание точек происходит по тому же правилу, что и в команде `julia`.

пример

```
mandelbrot()$
```



Дополнительные опции команд `julia` и `mandelbrot`:

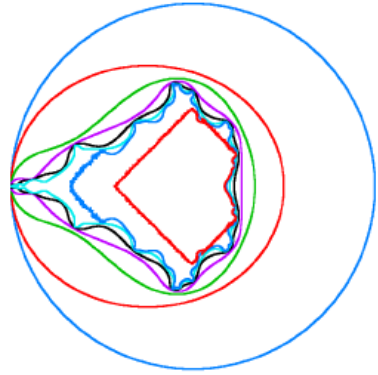
- `size` — задаёт размер изображения в пикселах. При этом `size=a` (или `[size,a]`) задает квадратное изображение `a`×`a`, а `[size,a,b]`. Если `a`≠`b`, изображение будет выглядеть искаженным. По умолчанию размеры 400×400.
- `levels` — определяет максимальное число итераций, это число также равно количеству используемых цветов для раскраски точек вне множества. Значение по умолчанию 12. Большие значения требуют большего времени вычислений.
- `huerange` — задает область углов оттенка (*hue*) для раскраски точек вне множества. Значение по умолчанию 360 (это означает, что цвета будут всех оттенков). Если это значение больше 360, область углов оттенка повторяется. Если значение меньше нуля, то углы оттенка будут убывать с количеством итераций.
- `hue` — задаёт значение оттенка в градусах для первого цвета для окраски точек вне множеств. Значение по умолчанию 300, что отвечает пурпурному цвету (*magenta*). Стандартные значения для других цветов: 0 (красный цвет), 45 (оранжевый), 60 (желтый), 120 (зелёный), 180 (голубой, *cyan*), 240 (синий).
- `saturation` — задаёт насыщенность цвета точек вне множества (в пределах от 0 до 1). По умолчанию 0.46.

- `value` — задаёт яркость цвета точек вне множества (в пределах от 0 до 1). По умолчанию 0.96.
- `color` — задаёт значение цвета точек множества тремя параметрами: `[color,`*hue*`,`*sturation*`,`*value*`]` (оттенок, насыщенность, яркость). По умолчанию все параметры равны 0, что соответствует чёрному цвету.
- `center` — задает координаты точки $c = x+iy$ комплексной плоскости, приходящейся на центр изображения: `[center,`x`,`y`]`. Значение по умолчанию $x = y = 0$ (начало координат находится в середине изображения).
- `radius` — задаёт радиус наибольшей окружности, умещающейся в изображаемой квадратной области (определяет масштаб изображения). По умолчанию 2.
- `filename` — задаёт имя файла изображения без расширения. По умолчанию это имя `"julia"` или `"mandelbrot"` и файл сохраняется в директорию, в которой установлена Maxima. Если необходимо сохранять файлы в другой директории, следует указать полное имя файла с путём, но без расширения.

IV Дополнительные примеры листингов программ

Здесь собраны листинги программ разных авторов (в основном, отсюда [10e],[27e],[28e]). Мы не останавливаемся на подробном обсуждении реализации этих алгоритмов. Предполагается, что необходимые сведения читатель найдет в интернете. Рекомендуется изучить приведенные листинги, запустить их в Maxima, изучить неизвестные вам команды и опции и подумать, почему автор именно так реализовал свой алгоритм. Попробуйте изменить параметры программ, чтобы лучше их понять.

☆ **Вычисление и отображение лемнискат множества Мандельброта (авторы алгоритма на Maxima A. Majewski, J. Villate и A. Vodopivec), см. [24e].**



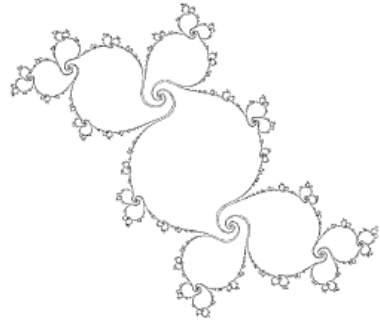
Программа выводит результат в файл `lemniscates.png` в папке программы Maxima.

Лемнискаты множества Мандельброта

```
load(implicit_plot)$
c: x+%i*y$
ER:2$                                        /* Радиус убегания, должен быть >=2 */
f[n](c) := if n=1 then c else (f[n-1](c)^2 + c)$
ip_grid:[100,100]$          /* устанавливаем разрешение первичной сетки узлов для
                                        implicit_plot, по умолчанию: [50, 50] */
ip_grid_in:[15,15]$         /* устанавливаем разрешение вторичной сетки узлов для
                                        implicit_plot, по умолчанию: [5, 5] */
my_preamble: "set zeroaxis; set title 'Boundaries of level sets of escape time
              of Mandelbrot set'; set xlabel 'Re(c)'; set ylabel 'Im(c)'"$
implicit_plot(makelist(abs(ev(f[n](c)))=ER,n,1,7), [x,-2.5,2.5],[y,-2.5,2.5],
             [gnuplot_preamble, my_preamble],[gnuplot_term,"png size 1000,1000"],
             [gnuplot_out_file, "lemniscates.png"])$
```

Заметим, что программе нужно много времени для вычислений. Если вы не видите результат, уменьшите количество выводимых лемнискат (параметр n). Приведенная программа не может корректно строить лемнискаты №7—8 из-за вычислительных трудностей при работе с числами с плавающей точкой (floating point).

☆ Построение множества Жюлиа с помощью модифицированного метода обратных итераций MIIM (автор алгоритма на Maxima A. Majewski), см. [25e].



Программа выводит результат в файл `miim.png` в папке программы Maxima. Модифицированный метод обратных итераций: останавливаем обратные итерации, если пиксел был посещён больше, чем `HitLimit` раз.

Множество Жюлиа, алгоритм MIIM

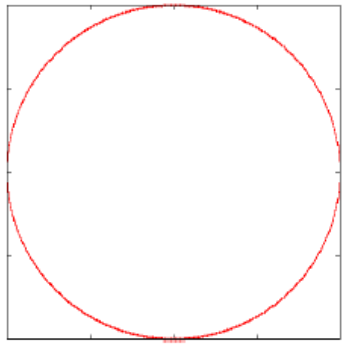
```
c:-0.11+0.65569999*%i$                                    /* параметр множества Жюлиа */
HitLimit:14$       /* пропорционален степени детализации и затраченному времени */
[iXmax,iYmax]:[500,500]$   /* разрешение, пропорционально детализации и времени */
                   /* размер рисунка будет : ширина:iXmax-0+1; высота:iYmax-0+1 */
start:elapsed_run_time ();
NumberOfPoints:0;
f(z,c):=z*z+c;
finverseplus(z,c):=sqrt(z-c);
finverseminus(z,c):=-sqrt(z-c);
[zxMin,zxMax]:[-2.0,2.0]$          /* определяем область комплексной плоскости, */
[zyMin,zyMax]:[-2.0,2.0]$                        /* в которой ищем множество Жюлиа */
PixelWidth:(zxMax-zxMin)/iXmax$
PixelHeight:(zyMax-zyMin)/iYmax$
array(Hits,iXmax,iYmax);  /* 2D массив, считающий количество посещений пикселов */
                                                 /* решётки точками нашей орбиты */
    /* Hits>0 означает, что точка принадлежит орбите. Вначале таких точек нет: */
for iX:0 thru iXmax step 1 do
```

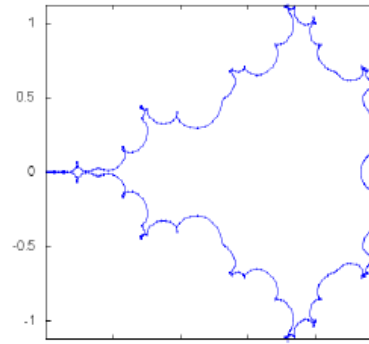
```
    for iY:0 thru iYmax step 1 do Hits[iX,iY]:0$
                        /* считаем неподвижные точки отображения f(z,c): z=f(z,c) */
fixed:float(rectform(solve([z*z+c=z],[z])));
                                        /* выделяем из решения отталкивающую точку */
if (abs(2*rhs(fixed[1]))<1)
    then block(beta:rhs(fixed[1]),alfa:rhs(fixed[2]))
    else block(alfa:rhs(fixed[1]),beta:rhs(fixed[2]))$
                            /* выбираем отталкивающую точку в качестве начальной */
```

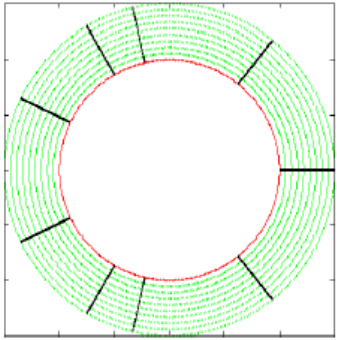
```
stack:[beta]$                                          /* сохраняем beta в стеке */
                          /* создаём 2 списка точек и копируем beta в эти списки */
xx:makelist (realpart(beta), i, 1, 1)$                          /* список Re(z) */
yy:makelist (imagpart(beta), i, 1, 1)$                          /* список Im(z) */
NumberOfPoints:1$
                                                    /* обратные итерации beta: */
block(
    loop,
    z:last(stack),
    stack:delete(z,stack),
    z:finverseplus(z,c),          /* обратная итерация - первый прообраз (корень) */
    iX:fix((realpart(z)-zxMin)/PixelWidth),    /* переводим из мировых координат */
    iY:fix((imagpart(z)-zyMin)/PixelHeight),                    /* в экранные */
    hit:Hits[iX,iY],
    if hit<HitLimit
       then block(
           Hits[iX,iY]:hit+1,
           stack:endcons(z,stack),           /* добавляем z в конец списка stack */
           if hit=0 then block(xx:cons(realpart(z),xx),yy:cons(imagpart(z),yy)),
           NumberOfPoints:NumberOfPoints+1
           ),
    z:-z,                         /* обратная итерация - второй прообраз (корень) */
```

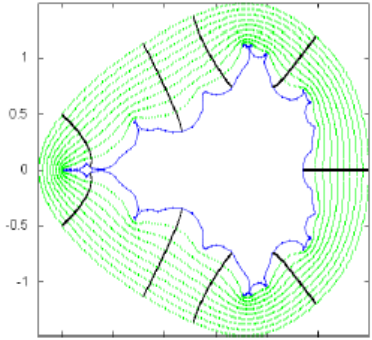
```
    iX:fix((realpart(z)-zxMin)/PixelWidth),   /* переводим из мировых координат */
    iY:fix((imagpart(z)-zyMin)/PixelHeight),                    /* в экранные */
    hit:Hits[iX,iY],
    if hit<HitLimit
       then block(
          Hits[iX,iY]:hit+1,
          stack:endcons(z,stack),          /* добавляем z в конец списка stack */
          if hit=0 then block(xx:cons(realpart(z),xx),yy:cons(imagpart(z),yy)),
          NumberOfPoints:NumberOfPoints+1
          ),
    if is(not emptyp(stack)) then go(loop)
)$
stop:elapsed_run_time()$
t:fix(stop-start);

load(draw)$      /* отображаем обратные орбиты точки beta с помощью пакета draw */
draw2d(file_name="miim",terminal='png,dimensions=[iXmax,iYmax],
      yrange=[zyMin,zyMax],xrange=[zxMin,zyMax],
      title="Julia set in dynamical plane for f(z,c):=z*z+c drawn using MIIM",
      key=concat("c=",string(c),". There are ",string(NumberOfPoints)," points"),
      label([concat("HitLimit=",string(HitLimit)," Time=",string(t)),-1.2,-1.8]),
      xlabel="Z.re ", ylabel="Z.im",point_type=dot,point_size=8,color=black,
      points(xx,yy)
)$
```

☆ **Построение отображения окружности на множество Мандельброта (автор алгоритма на Maxima A. Majewski, R.Fateman, G. A. Edgar), см. [26e].**








1
0.5
0
-0.5
-1

Как показали А. Дуади и Дж. Хаббард, существует конформное преобразование, отображающее в расширенной комплексной плоскости $\hat{\mathbb{C}}$ внешность множества Мандельброта $\mathcal{M}$ на внешность единичного диска $\overline{\mathbb{D}}$ (функция Л. Бетхера) $\Phi_M \colon \hat{\mathbb{C}} \setminus \mathcal{M} \to \hat{\mathbb{C}} \setminus \overline{\mathbb{D}}$.

Обратное к нему отображение $\Psi_M \colon \hat{\mathbb{C}} \setminus \overline{\mathbb{D}} \to \hat{\mathbb{C}} \setminus \mathcal{M}$ называется отображением И. Юнграйза, точнее, алгоритмом Юнграйза, предложившего конструктивное построение этой функции (см. [19, 20, 21]):

$$\Psi_M(\omega) = \omega + \sum_{m=0}^{\infty} b_m \omega^{-m} = \omega - \frac{1}{2} + \frac{1}{8\omega} - \frac{1}{4\omega^2} + \frac{15}{128\omega^3} + \dots$$

Отображение Юнгрaйза для множества Мандельброта

```
/* Используем симметрию вещественной оси, Psi_M функцию и алгоритм Юнгрaйза */
start:elapsed_run_time()$
jMax:50$          /* точность, пропорциональна детализации и времени вычислений */
iMax:300$                                    /* количество отрисовываемых точек */
iMaxBig:400$
                                  /* вычисление коэффициентов b функции Юнгрaйза */
betaF[n,m]:=block
    ([nnn:2^(n+1)-1],
    if m=0 then 1.0 else if ((n>0) and (m < nnn)) then 0.0 else
       (betaF[n+1,m]-sum(betaF[n,k]*betaF[n,m-k],k,nnn,m-nnn)-betaF[0,m-nnn])/2.0
    )$
b[m]:=betaF[0,m+1]$
wn[w,j]:= if j=0 then 1 else w*wn[w,j-1]$                          /* степени w^j */
Psi_M(w):=w + sum(b[j]/wn[w,j],j,0,jMax)$       /* функция Юнгрaйза; c=Psi_M(w) */
           /* показательная форма комплексного числа, t - доля дуги окружности */
                    /* возвращает точку на единичной окружности для доли угла t */
GiveCirclePoint(t):=R*%e^(%i*t*2*%pi)$
               /* возвращает точку внешнего луча с радиусом R и долей угла tRay */
GiveWRayPoint(R):=R*%e^(%i*tRay*2*%pi)$
NmbrOfCurves:9$
                              /* координаты на w-плоскости (а не на c-плоскости!) */
```

```
[R_min,R_max]:[1,1.5]$
dR:(R_max-R_min)/NmbrOfCurves$                                  /* для окружностей */
dRR:(R_max-R_min)/iMax$                                         /* для лучей */
                                                  /* f_0 плоскость = w-плоскость */
R:1$                                                  /* единичная окружность */
circle_angles:makelist(i/iMax,i,0,iMax/2)$
CirclePoints:map(GiveCirclePoint,circle_angles)$
                                                        /* внешние окружности */
circle_radii: makelist(R_min+i*dR,i,1,NmbrOfCurves)$
circle_angles:makelist(i/iMaxBig,i,0,iMaxBig/2)$
WCirclesPoints:[]$
for R in circle_radii do
    WCirclesPoints:append(WCirclesPoints,map(GiveCirclePoint,circle_angles))$
                                                                /* внешние w-лучи */
ray_radii:makelist(R_min+dRR*i,i,1,iMax)$
ray_angles:[0,1/3,1/7,2/7,3/7]$                  /* список углов < 1/2 для лучей */
WRaysPoints:[]$
for tRay in ray_angles do
    WRaysPoints:append(WRaysPoints,map(GiveWRayPoint,ray_radii))$
                                          /* плоскость параметра = c плоскость */
MPoints:map(Psi_M,CirclePoints)$                 /* точки множества Мандельброта */
CRaysPoints:map(Psi_M,WRaysPoints)$                          /* внешние z лучи */
Equipotentials:map(Psi_M,WCirclesPoints)$
                                       /* добавляем точки ниже вещественной оси */
```

```
for w in CirclePoints do CirclePoints:cons(conjugate(w),CirclePoints)$
for w in WRaysPoints do WRaysPoints:cons(conjugate(w),WRaysPoints)$
for w in WCirclesPoints do WCirclesPoints:cons(conjugate(w),WCirclesPoints)$
for c in MPoints do MPoints:cons(conjugate(c),MPoints)$
for c in CRaysPoints do CRaysPoints:cons(conjugate(c),CRaysPoints)$
for c in Equipotentials do Equipotentials:cons(conjugate(c),Equipotentials)$
                                                    /* затраченное время */
```

```
stop:elapsed_run_time ()$
time:fix(stop-start);
load(draw)$                                                    /* рисуем */
draw(file_name="jung", dimensions=[1000,500],
    terminal='png, columns=2,
    gr2d(title="unit circle with external rays & circles",
        point_type=filled_circle, points_joined=true,
        point_size=0.34, color=red,
        points(map(realpart, CirclePoints),map(imagpart, CirclePoints)),
        points_joined =false, color=green,
        points(map(realpart, WCirclesPoints),map(imagpart, WCirclesPoints)),
        color=black,
        points(map(realpart, WRaysPoints),map(imagpart, WRaysPoints))
        ),
    gr2d(title="Parameter plane : Image under Psi_M(w)",
        points_joined=true, point_type=filled_circle,
        point_size=0.34, color=blue,
        points(map(realpart, MPoints),map(imagpart, MPoints)),
        points_joined=false, color=green,
        points(map(realpart, Equipotentials),map(imagpart, Equipotentials)),
        color=black,
        points(map(realpart, CRaysPoints),map(imagpart, CRaysPoints))
        )
);
```

РЕСУРСИ ІНТЕРНЕТУ

[1e] http://maxima.sourceforge.net/ru/
Официальная страница проекта Maxima на русском языке (однако, она редко обновляется, рекомендуется пользоваться английской версией страницы).

[2e] http://sourceforge.net/projects/maxima/files/
Здесь можно скачать установочный файл для Maxima.

[3e] http://maxima.sourceforge.net/ru/documentation.html
Подборка руководств по Maxima с официального сайта, в том числе и на русском языке.

[4e] http://andrejv.github.com/wxmaxima/help.html
Коллекция руководств по Maxima.

[5e] http://www.uneex.ru/static/MethodBooks_Maxima/Maxima.pdf
Пособие Н.А. Стахина .Основы работы с системой аналитических (символьных) вычислений Maxima. на русском языке.

[6e] http://www3.msiu.ru/~lao4/maxima.pdf
Методическое пособие с неофициального сайта МГИУ www3.msiu.ru.

[7e] http://euler.us.es/~renato/clases/maxima/manualesPDF/MaximaByExamples/mbe5qdraw.pdf
Документация пакета Edwin L. Woollett qdraw.

[8e] http://www.csulb.edu/~woollett/
Сайт профессора Edwin L. Woollett с очень полезной информацией о Maxima.
[9e] http://www.csulb.edu/~woollett/mbe2plotfit.pdf
Туториал от профессора Edwin L. Woollett о выводе графике и работе с файлами (см. также другие его туториалы по предыдущей ссылке).
[10e] http://en.wikibooks.org/wiki/Maxima
Полезная информация о Maxima в открытой энциклопедии. Есть примеры программирования фракталов.
[11e] http://www.telefonica.net/web2/biomates/maxima/sound/index.html
Пакет работы со звуком sound.lisp.
[12e] http://maxima.cvs.sourceforge.net/viewvc/maxima/maxima/share/contrib/fractals/
Пакет для построения простейших фракталов fractals.mac. На этом же сайте есть много других пакетов для Maxima.
[13e] http://www.imagemagick.org/script/download.php
Отличная бесплатная программа Image Magick для работы с изображениями из командной строки.
[14e] http://www.biologie.uni-hamburg.de/b-online/e28_3/lsys.html
L-системы.

[15e] http://algorithmicbotany.org/
L-системы. Сайт профессора Пршемыслава Прузинкевича.
[16e] http://algorithmicbotany.org/papers/#abop
Книга Прузинкевича в открытом доступе.
[17e] http://mathworld.wolfram.com/LindenmayerSystem.html
Черпаем вдохновение в готовых демонстрациях с сайта mathworld.wolfram.com.
[18e] http://www.austromath.at/daten/maxima/zusatz/Graphics_with_Maxima.pdf
Хорошая документация по использованию графики в Maxima.
[19e] http://riotorto.users.sourceforge.net/gnuplot/animation/
Примеры создания анимации средствами Maxima.
[20e] http://riotorto.users.sourceforge.net/gnuplot/
Хорошая подборка примеров использования графики от создателя пакета draw Mario Rodr’ıguez Riotorto.
[21e] http://en.wikibooks.org/wiki/Fractals/Iterations_in_the_complex_plane/Julia_set
Алгоритмы построения и раскрашивания Множеств Жюлиа.
[22e] http://www.fraktal.republika.pl/iim.html
Реализации метода обратных итераций для построения множеств Жюлиа.
[23e] http://www.fraktal.republika.pl

Различные алгоритмы для изучения множеств Жюлиа, множества Мандельброта и др., а также полезные ссылки.

[24e] http://commons.wikimedia.org/wiki/File:Lemniscates5.png

[25e] http://commons.wikimedia.org/wiki/File:Miim.jpg

[26e] http://en.wikipedia.org/wiki/File:Jung50e.png

[27e] http://en.wikibooks.org/wiki/Fractals/Iterations_in_the_complex_plane/Julia_set#BSM.2FJ

Много различной информации об алгоритмах построения фрактальных множеств на комплексной плоскости.

[28e] http://en.wikibooks.org/wiki/Fractals/Computer_graphic_techniques/2D

Много различной информации об алгоритмах построения фрактальных множеств на комплексной плоскости.