

La increíble belleza escondida en los Sistemas Dinámicos Discretos

Jefferson E. King Dávalos

Renato Leriche Vázquez

Departamento de Matemáticas, Facultad de Ciencias, UNAM

In [2]: `using Revise, Pkg`

In [3]: `sdd_path = "D:\\Software\\Dev\\SDD\\SDD"`

Out[3]: `"D:\\Software\\Dev\\SDD\\SDD"`

In [4]: `Pkg.develop(path=sdd_path*"Core.jl")`

```
Resolving package versions...
No Changes to `D:\Software\System\.julia\environments\v1.6\Project.toml`
No Changes to `D:\Software\System\.julia\environments\v1.6\Manifest.toml`
```

In [5]: `Pkg.develop(path=sdd_path*"Geometry.jl")`

```
Resolving package versions...
No Changes to `D:\Software\System\.julia\environments\v1.6\Project.toml`
No Changes to `D:\Software\System\.julia\environments\v1.6\Manifest.toml`
```

In [6]: `Pkg.develop(path=sdd_path*"Graphics.jl")`

```
Resolving package versions...
No Changes to `D:\Software\System\.julia\environments\v1.6\Project.toml`
No Changes to `D:\Software\System\.julia\environments\v1.6\Manifest.toml`
```

In [7]: `Pkg.develop(path=sdd_path*"jl")`

```
Resolving package versions...
No Changes to `D:\Software\System\.julia\environments\v1.6\Project.toml`
No Changes to `D:\Software\System\.julia\environments\v1.6\Manifest.toml`
```

In [8]: `Pkg.develop(path=sdd_path*"KleinianGroups.jl")`

```
Resolving package versions...
No Changes to `D:\Software\System\.julia\environments\v1.6\Project.toml`
No Changes to `D:\Software\System\.julia\environments\v1.6\Manifest.toml`
```

In [9]: `Pkg.develop(path=sdd_path*"IFS.jl")`

```
Resolving package versions...
No Changes to `D:\Software\System\.julia\environments\v1.6\Project.toml`
No Changes to `D:\Software\System\.julia\environments\v1.6\Manifest.toml`
```

In [10]: `using Colors, ColorSchemes`

In [11]: `using SDDCore, SDDGeometry, SDD, SDDIFS, SDDKleinianGroups`

In [12]: `import SDDGraphics
Gr = SDDGraphics
SDDGraphics.backend(:luxor)`

Out[12]: `:luxor`

In [13]: `using Interact`

¿Resumen, introducción, índice?

Definiciones básicas

$f : X \rightarrow X$, X espacio métrico y $x \in X$.

n -ésima iterada de f :

$$f^n = f \circ f^{n-1}$$

con $f^0 = Id$.

Órbita de x bajo f :

$$o(x, f) = \{f^n(x)\}_{n=0}^{\infty} = \{x, f(x), f^2(x), \dots\}$$

Conjunto de puntos atrapados:

$$\mathcal{K}(f) = \{x \in X \mid o(x, f) \text{ es acotada}\}$$

Gráficas de iteradas

Dada $f : \mathbb{R} \rightarrow \mathbb{R}$ podemos graficar f^n con la instrucción `plot`.

Ejemplo

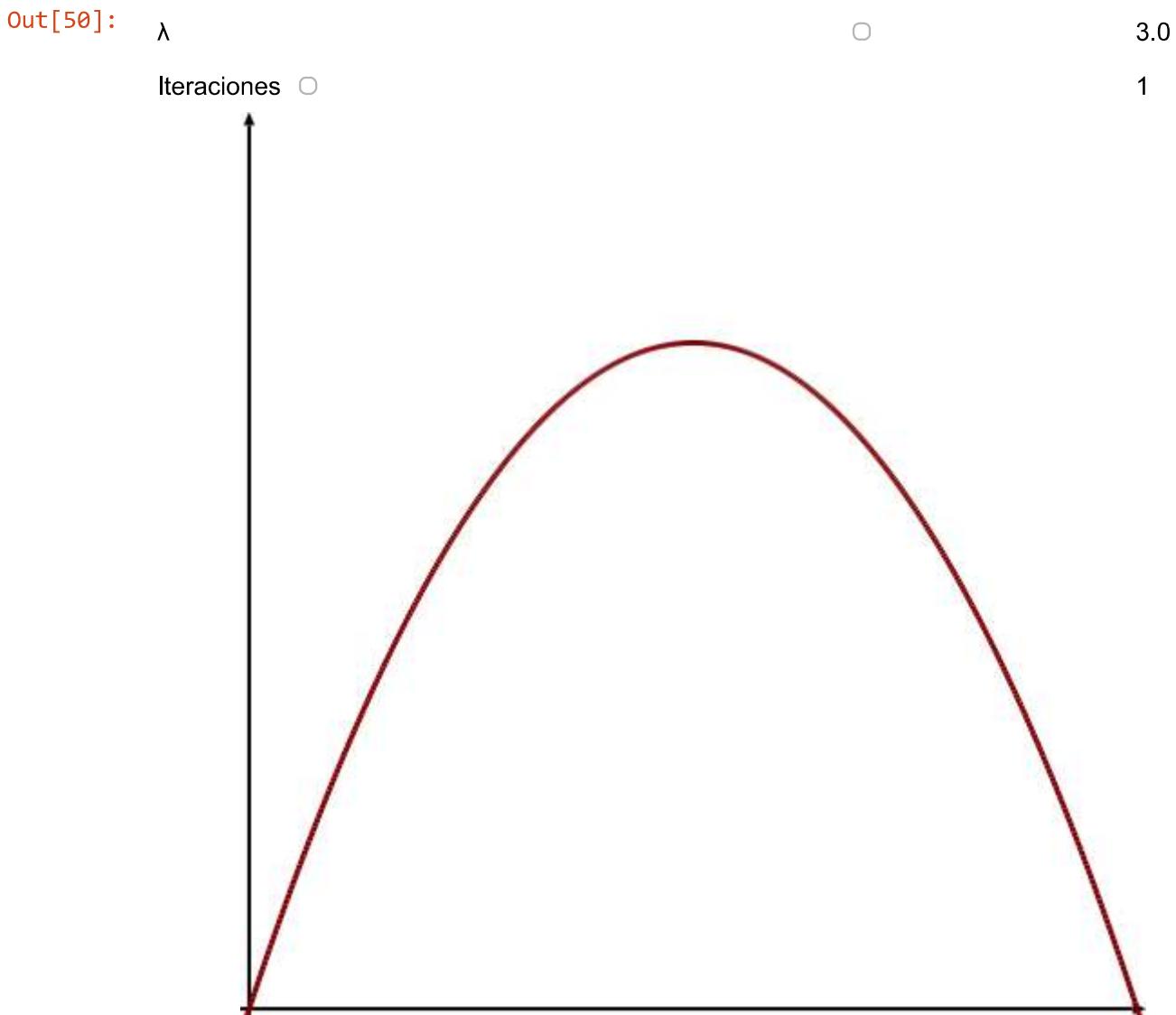
$L_\lambda(x) = \lambda x(1 - x)$, la familia logística, con $\lambda \in \mathbb{R}$.

In [14]: `L(λ, x)=λ*x*(1-x)`

Out[14]: `L (generic function with 1 method)`

```
In [49]: Gr.backend(:luxor)
Gr.configure(rect=(-0.01,1.01,-0.01,1.01), canvas=(500,500), linewidth=3, axes=true)
```

```
In [50]: @manipulate for λ=slider(0.1:0.05:4.1, value=3.0, label="λ"), n = slider(1:6, val=1)
    plot(x -> L(λ, x), iterations=n)
end
```



Análisis gráfico de órbitas

Dada $f : \mathbb{R} \rightarrow \mathbb{R}$ podemos graficar el análisis gráfico de la órbita de un punto con la instrucción `graphicalanalysis`.

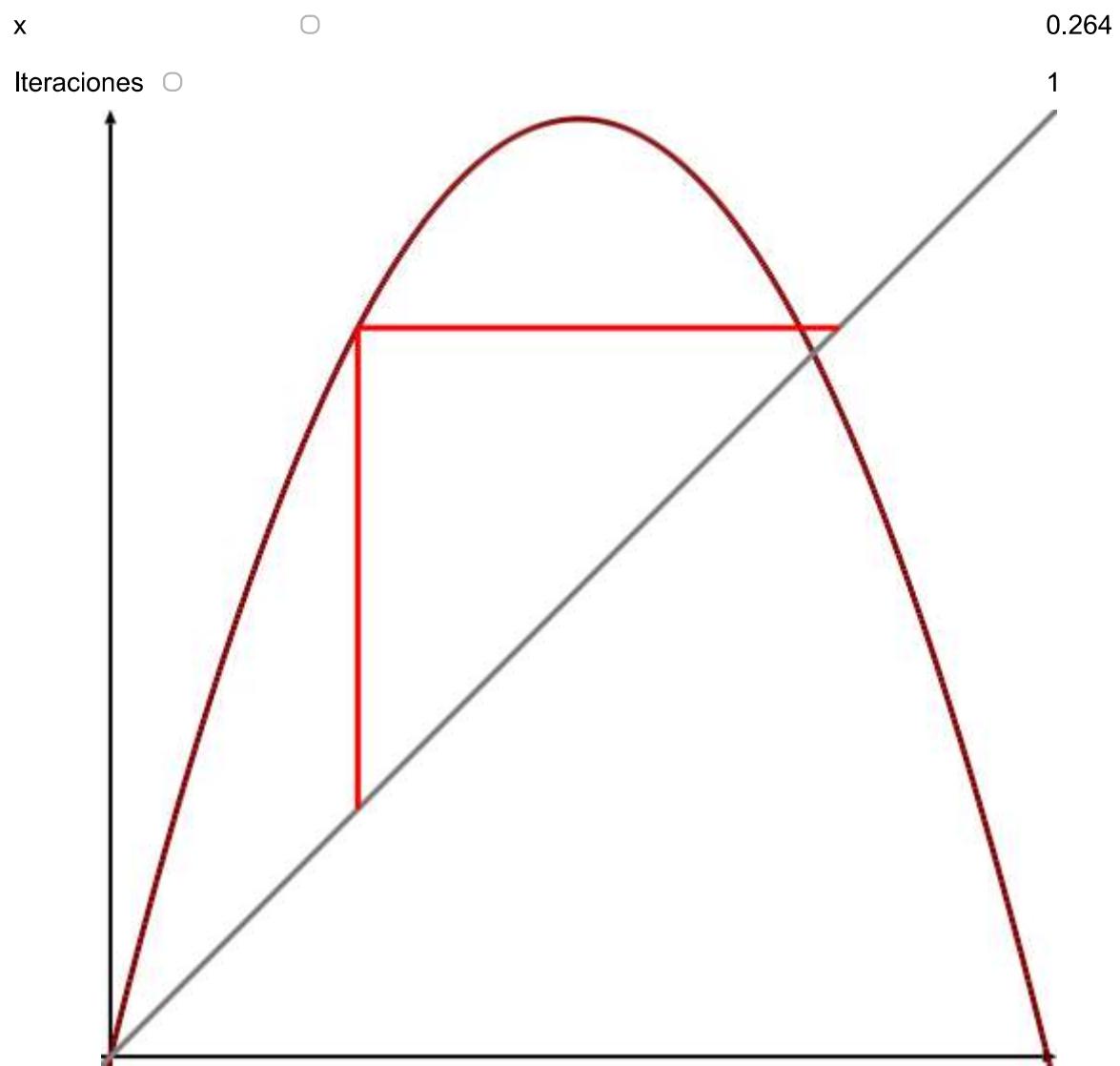
Ejemplo

$$L(x) = 4x(1 - x).$$

In [31]:

```
@manipulate for x=slider(0.0:0.001:1.0, value=0.0, label="x"), n=slider(1:100, va
    graphicalanalysis(x -> L(4,x), x, iterations=n)
end
```

Out[31]:



Puntos fijos, periódicos y pre-periódicos

Punto fijo de f : x tal que $f(x) = x$.

Conjunto de puntos fijos de f :

$$\text{Fix}(f) = \{x \in X \mid f(x) = x\}$$

Punto periódico de f de periodo n : x tal que $f^n(x) = x$ y $f^m(x) \neq x$ para todo $m < n$.

Conjunto de puntos periódicos de f de periodo n :

$$\text{Per}_n(f) = \{x \in X \mid x \text{ es de periodo } n\}$$

Conjunto de puntos periódicos de f :

$$\text{Per}(f) = \{x \in X \mid x \text{ es periódico}\}$$

Punto pre-periódico de f : x tal que $f^n(x)$ es periódico pero $f^m(x)$ no lo es para toda $m < n$.

Ejemplo

La familia de las funciones tiene, con $\lambda \in \mathbb{R}$:

$$T_\lambda(x) = \begin{cases} \lambda x & x \leq \frac{1}{2} \\ \lambda - \lambda x & x > \frac{1}{2} \end{cases}$$

In [28]: `Tent(λ,x) = x ≤ 0.5 ? 2x : 2-2x`

Out[28]: Tent (generic function with 1 method)

Caso $\lambda = 2$.

In [30]: `@manipulate for x=slider(0.0:0.001:1.0, value=0.0, label="x"), n=slider(1:100, va
graphicalanalysis(x -> Tent(2,x), x, iterations=n)
end`

Out[30]:

Puntos atractores y repulsores

Punto fijo atractor: Para cualquier vecindad \mathcal{N}_x de x existe otra vecindad $\mathcal{N}'_x \subset \mathcal{N}_x$ tal que para todo $y \in \mathcal{N}'_x$, $f^n(y) \rightarrow x$ cuando $n \rightarrow \infty$.

Punto fijo repulsor: Existe una vecindad \mathcal{N}_x tal que para todo $y \in \mathcal{N}_x - \{x\}$ existe $N > 0$ tal que $f^N(y) \notin \mathcal{N}_x$.

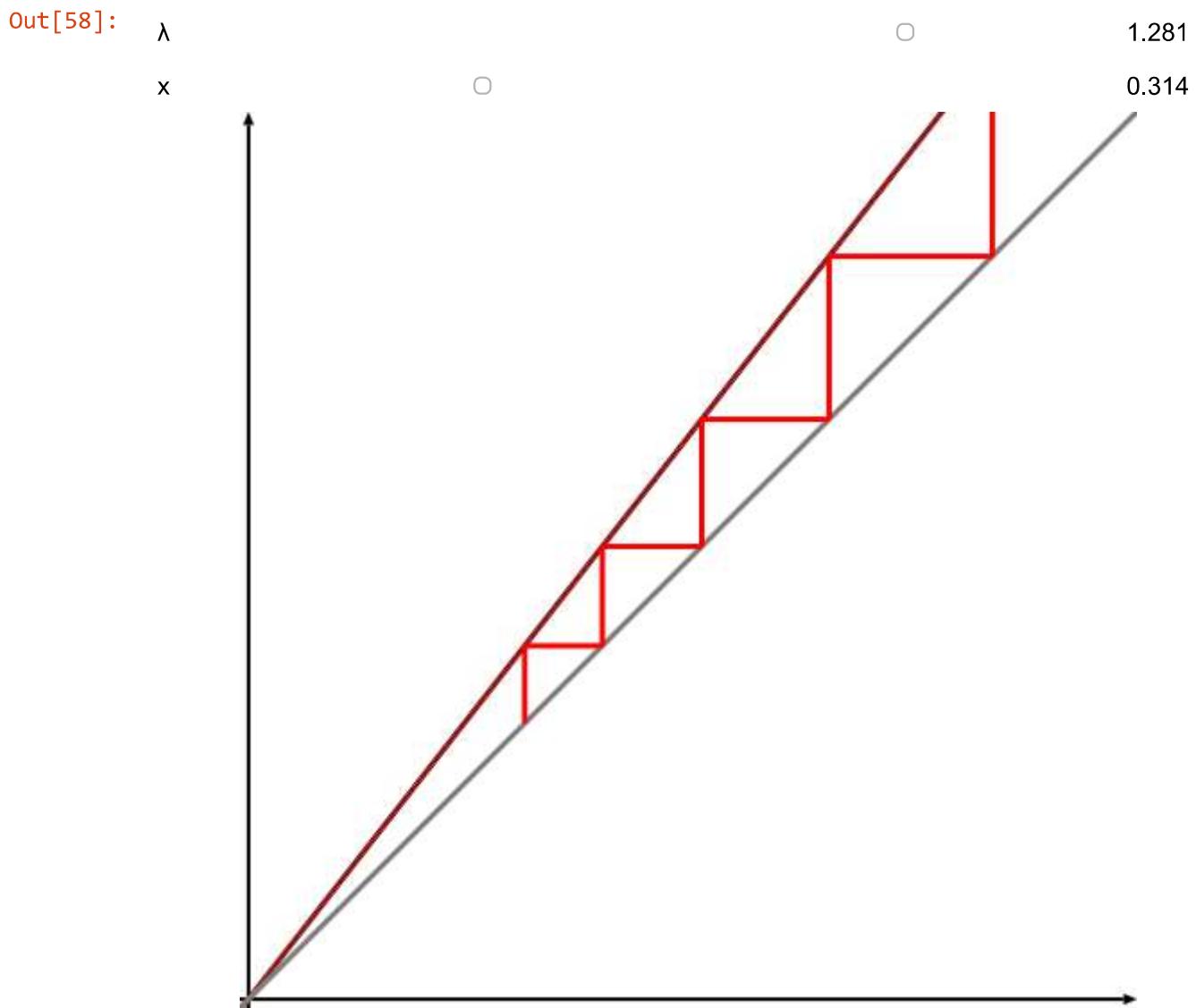
Cuenca de atracción de un punto fijo atractor x :

$$W^s(x) = \{y \in X \mid f^n(y) \rightarrow x \text{ cuando } n \rightarrow \infty\}$$

Ejemplo

$$x \mapsto \lambda x$$

In [58]: `manipulate for λ=slider(0.5:0.001:1.5, value=1.0, label="λ"), x=slider(0.0:0.001:nd
graphicalanalysis(t -> λ*t, x, iterations=20)`



Ejemplo

$$f(\beta, x, y) = (x^2 + y^2 + \beta)(\cos(\theta)x - \sin(\theta)y, \cos(\theta)y + \sin(\theta)x)$$

In [44]: `θ=2π/15
hopf(β,x,y)=(x^2+y^2+β).*(cos(θ)*x-sin(θ)*y,cos(θ)*y+sin(θ)*x)`

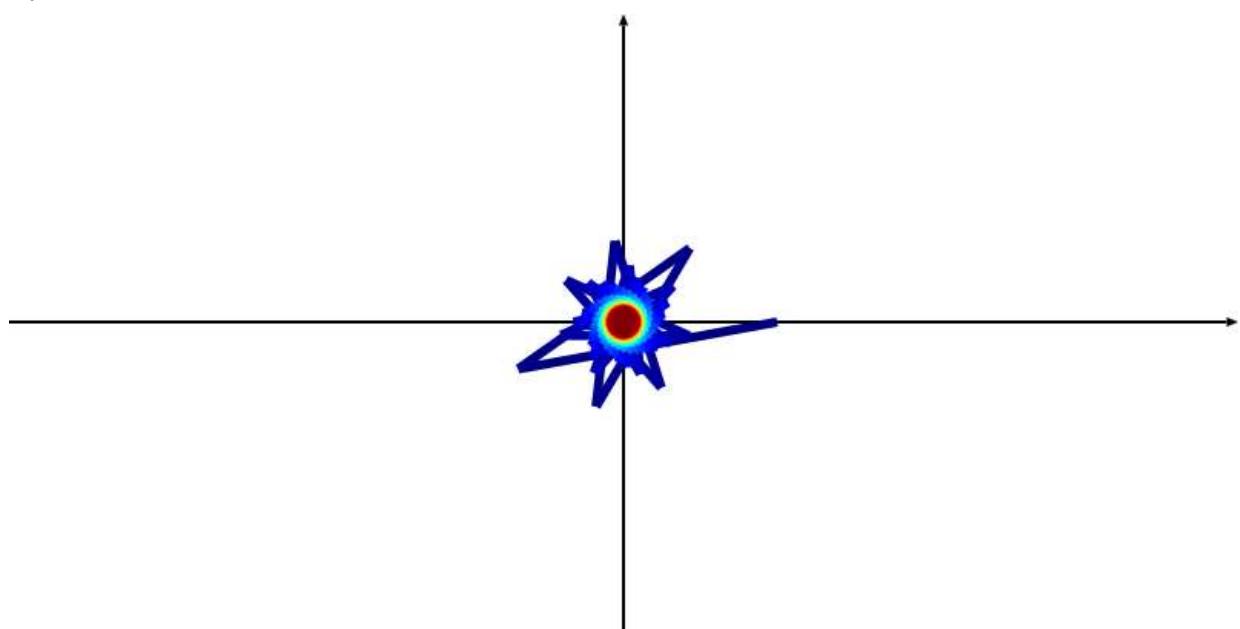
Out[44]: `hopf (generic function with 1 method)`

In [45]: `Gr.backend(:luxor)
Gr.configure(rect=(-2,2,-1,1), canvas=(800,400), linewidth=6, colormap=:jet)`

In [47]:

```
@manipulate for β=-10:0.1:10
    drawpointorbitpathR2((x,y) -> hopf(β,x,y), 0.5, 0.0, iterations=200)
end
```

Out[47]:



Caos

f sensible a las condiciones iniciales: Existe una constante r_0 tal que para todo x , para cada \mathcal{N}_x se cumple que existen $y \in \mathcal{N}_x$ y $N \geq 0$ tales que $d(f^N(x), f^N(y)) \geq r_0$.

f topológicamente transitiva: Existe x tal que $\overline{o(x, f)} = X$.

f topológicamente mezclante: Para cualesquiera abiertos no vacíos $U, V \subset X$ existe $N \geq 0$ tal que $f^N(U) \cap V \neq \emptyset$.

f caótica: Topológicamente mezclante y $\overline{\text{Per}(f)} = X$.

Teorema: Si f es caótica, entonces es sensible a las condiciones iniciales.

In []:

```
fdiforbits(x1,x2,n) =
```

Conjugación topológica

$f : X \rightarrow X$ y $g : Y \rightarrow Y$ continuas en espacios métricos.

f conjugada con g: Existe $h : X \rightarrow Y$ homeomorfismo tal que $h \circ f = g \circ h$.

Teorema: Si f es conjugada con g mediante h , entonces $x \in \text{Per}_n(f)$ si y sólo si $h(x) \in \text{Per}_n(g)$.

Teorema: Si f es caótica y f es conjugada con g , entonces g es caótica.

Funciones reales diferenciables

$f : \mathbb{R} \rightarrow \mathbb{R}$ diferenciable.

Teorema: x punto fijo de f .

- Si $|f'(x)| < 1$, entonces x es atractor.
- Si $|f'(x)| > 1$, entonces x es repulsor.

Punto fijo neutro: Si no es atractor ni repulsor.

Funciones en los complejos

$f : \hat{\mathbb{C}} \rightarrow \hat{\mathbb{C}}$ meromorfa.

Teorema: z punto fijo de f .

- Si $|f'(z)| < 1$, entonces z es atractor.
- Si $|f'(z)| > 1$, entonces z es repulsor.
- Si $|f'(z)| = 1$, entonces z es neutro.

Conjunto de Fatou:

$$\mathcal{F}(f) = \{z \in \hat{\mathbb{C}} \mid \{f^n\}_{n \in \mathbb{N}} \text{ es normal}\}$$

Conjunto de Julia:

$$\mathcal{J}(f) = \hat{\mathbb{C}} - \mathcal{F}(f)$$

Teorema: $f(\mathcal{F}(f)) = \mathcal{F}(f)$ y $f(\mathcal{J}(f)) = \mathcal{J}(f)$.

Teorema: $\mathcal{F}(f)$ es abierto y $\mathcal{J}(f)$ es cerrado.

Teorema:

$$\mathcal{J}(f) = \overline{\{z \in \text{Per}(f) \mid z \text{ es repulsor}\}}$$

Teorema: Con $z \in \mathcal{J}(f)$

$$\mathcal{J}(f) = \overline{\bigcup_{n=0}^{\infty} f^{-1}(z)}$$

Teorema: f es caótica en $\mathcal{J}(f)$.

La familia cuadrática compleja

La familia cuadrática estándar:

$$\{q_c : \mathbb{C} \rightarrow \mathbb{C} \mid q_c(z) = z^2 + c\}_{c \in \mathbb{C}}$$

Conjunto de Mandelbrot:

$$\mathcal{M} = \{c \in \mathbb{C} \mid o(0, q_c) \text{ es acotada}\}$$

Teorema: $\mathcal{M} = \{c \in \mathbb{C} \mid \mathcal{J}(q_c) \text{ es conexo}\}$.

Teorema: \mathcal{M} es conexo.

q_c **estructuralmente estable** (en el espacio de parámetros): Existe una vecindad \mathcal{N}_c de c tal que para toda $c' \in \mathcal{N}_c$, $q_{c'}$ es conjugada topológicamente con q_c .

Teorema: El conjunto de parámetros tales que q_c es estructuralmente estable, es denso en \mathbb{C} .

q_c **hiperbólica**: El punto crítico 0 está en una cuenca de atracción.

Teorema: Si q_c es hiperbólica, entonces $\mathcal{F}(q_c)$ está formado sólo por cuencas de atracción.

Teorema: Si q_c es hiperbólica, entonces es estructuralmente estable.

Conjetura: $\mathbb{C} - \partial\mathcal{M} = \{c \in \mathbb{C} \mid q_c \text{ es hiperbólica}\}$ (y entonces también $\mathbb{C} = \{c \in \mathbb{C} \mid q_c \text{ es hiperbólica}\}$).

Funciones racionales complejas

Función racional: $R : \hat{\mathbb{C}} \rightarrow \hat{\mathbb{C}}$ tal que $R = \frac{P}{Q}$, donde P y Q son polinomios con de una variable, con coeficientes en \mathbb{C} y sin raíces en común.

$R : \hat{\mathbb{C}} \rightarrow \hat{\mathbb{C}}$ racional.

Grado de R : $\deg(R) = \max\{\deg(P), \deg(Q)\}$.

Teorema: R con grado d tiene, a lo más, $2d - 2$ puntos críticos.

Teorema: Toda cuenca de atracción de R tiene un punto crítico.

Corolario: R con grado d tiene, a lo más, $2d - 2$ órbitas atractoras distintas.

R **hiperbólica**: Todos los puntos críticos de R están en la cuenca de atracción de algún punto periódico atractor.

Transformaciones de Möbius

Transformación de Möbius: $T : \hat{\mathbb{C}} \rightarrow \hat{\mathbb{C}}$ de la forma

$$T(z) = \frac{az + b}{cz + d}$$

donde $a, b, c, d \in \mathbb{C}$ son tales que $ad - bc \neq 0$.

Teorema: Las transformaciones de Möbius son los automorfismos conformes de $\hat{\mathbb{C}}$, forman un grupo isomorfo a $PSL(2, \mathbb{C})$, son holomorfas en toda $\hat{\mathbb{C}}$, son transitivas en la familia de todos los círculos de $\hat{\mathbb{C}}$ y tienen sólo 1 o 2 puntos fijos (sin considerar el caso Id).

T **parabólica**: Sólo tiene un punto fijo.

T elíptica: Tiene dos puntos fijos neutros.

T loxodrómica: Tiene dos puntos fijos, uno atractador y otro repulsor.

Grupos de transformaciones de Möbius

$$\Gamma < PSL(2, \mathbb{C})$$

Órbita de z bajo Γ :

$$\Gamma z = \bigcup_{T \in \Gamma} \{T(z)\}$$

Punto límite z de Γ : Existe w y transformaciones distintas $T_n \in \Gamma$ tales que $T_n(w) \rightarrow z$ cuando $n \rightarrow \infty$.

Conjunto límite de Γ :

$$\Lambda(\Gamma) = \{z \in \hat{\mathbb{C}} \mid z \text{ es punto límite de } \Gamma\}$$

Conjunto ordinario de Γ :

$$\Omega(\Gamma) = \hat{\mathbb{C}} - \Lambda(\Gamma)$$

Teorema: $\Lambda(\Gamma)$ y $\Omega(\Gamma)$ son invariantes bajo Γ .

Teorema: $\Lambda(\Gamma)$ es cerrado y $\Omega(\Gamma)$ es abierto.

Grupos kleinianos

Γ discontinuo: $\Omega(\Gamma) \neq \emptyset$.

Γ kleiniano (ó discreto): Γ no tiene puntos de acumulación en $PSL(2, \mathbb{C})$.

Teorema: Si Γ es discontinuo, entonces es discreto.

Γ kleiniano elemental: $\Lambda(\Gamma)$ formado sólo por 0, 1 o 2 puntos.

Teorema: Si Γ es kleiniano no elemental, entonces $\Lambda(\Gamma)$ está formado por un número infinito de puntos.

Teorema: Si Γ es kleiniano no elemental, entonces

$$\Lambda(\Gamma) = \overline{\bigcup_{\gamma \in \Gamma \text{ loxodrómica}} \text{Fix}(\gamma)}$$

(Para lo siguiente se requieren definiciones complicadas con extensiones de Poincaré, cocientes de grupos, 3-variedades o 3-orbifolios, volúmenes hiperbólicos, representaciones de grupos, espacios de grupos, etc...)

Teorema: El conjunto de grupos kleinianos hiperbólicos, es denso en el espacio de grupos kleinianos.

Teorema: Un grupo kleiniano estructuralmente estable, es hiperbólico.

Conjetura: El conjunto de grupos kleinianos estructuralmente estables, es denso en el espacio de grupos kleinianos.)

(Respaldo)

Presentaremos los avances de desarrollo de un sistema de software, formado por un ecosistema de bibliotecas y una aplicación,

creado para ser útil en la investigación y docencia en el área de sistemas dinámicos discretos.

Este trabajo es parcialmente financiado por el **Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT)** de la UNAM

en el *Proyecto PAPIIT-UNAM IT103720* con título

"Sistema de Software para Investigación en Sistemas Dinámicos Discretos"

Equipo de trabajo:

- Responsable: **Dr. Jefferson E. King Dávalos**.
- *Project Manager*: **Renato Leriche Vázquez** (tesista de Doctorado del Dr. Guillermo Sienra).
- *Senior Software Developer*: **Luis Muñiz Valledor** (tesista de Licenciatura del Dr. Marco Montes de Oca).
- *Software Developer*: **José D. Blancas Camarena** (tesista de Licenciatura de Renato Leriche).
- *Software Developer*: **Edgar A. Rodríguez Lucio** (tesista de Licenciatura de Renato Leriche).
- *Software Developer*: **Fernando Santana Plascencia** (tesista de Licenciatura del Dr. Jefferson King).
- *Software Developer*: **Ian X. Belaustegui** (servicio social).
- Colaborador: **Dr. Guillermo F.J. Sienra Loera**.
- Colaborador: **Dr. Marco A. Montes de Oca Balderas**.

(Todos del Departamento de Matemáticas, Facultad de Ciencias, UNAM.)

Bibliotecas de software

Una *biblioteca de software* proporciona un conjunto de instrucciones (sobre un lenguaje de programación) para construir soluciones a problemas en cierta área.

Las principales bibliotecas de software creadas en este proyecto son:

- **SDD**: Para la teoría de iteración de funciones en \mathbb{R} , \mathbb{R}^2 y \mathbb{C} .

- **SDDIFS**: Para la teoría de sistemas de funciones iteradas.

- **SDDKleinianGroups**: Para la teoría de grupos Kleinianos.

Además, de las bibliotecas de software complementarias:

- **SDDCore**. Instrucciones en común a todas las bibliotecas.
- **SDDGraphics**. Elementos básicos para visualización.
- **SDDGeometry**. Construcción y manejo de objetos y transformaciones geométricas.

SDD es el nombre de nuestro software y significa "Sistemas Dinámicos Discretos".

El software para este proyecto fué desarrollado utilizando el lenguaje de programación **Julia**.

Se eligió este **Julia** por ser un lenguaje de programación moderno, especialmente diseñado para cómputo científico y que combina simplicidad con eficiencia.

En esta presentación haremos programación en vivo, pues estamos usando el ambiente **Jupyter** que permite usar **Julia** de manera fácil e interactiva.

Haremos recuento de algunas de las posibilidades que ofrecen las bibliotecas y mostraremos en vivo las imágenes generadas.

La biblioteca **SDD**

In [14]: `using SDD`

Gráficas de iteradas

Dada $f : \mathbb{R} \rightarrow \mathbb{R}$ podemos graficar f^n con la instrucción `plot`.

Ejemplo

$L(x) = 4x(1 - x)$, la función logística.

$L_\lambda(x) = \lambda x(1 - x)$, la familia logística.

In [16]: `L4(x) = 4x*(1-x)`

Out[16]: L4 (generic function with 1 method)

In [13]: `L(λ,x)=λ*x*(1-x)`

Out[13]: L (generic function with 1 method)

In [14]: `Gr.backend(:luxor)
Gr.configure(rect=(-0.01,1.01,-0.01,1.01), canvas=(500,500), linewidth=3, axes=true)
plot(L4, iterations=1)`

UndefVarError: L4 not defined

Stacktrace:

- [1] top-level scope
 @ In[14]:3
- [2] eval
 @ .\boot.jl:360 [inlined]
- [3] include_string(mapexpr::typeof(REPL.softscope), mod::Module, code::String,
filename::String)
 @ Base .\loading.jl:1116

In [15]: `@manipulate for λ = 0.1:0.05:4.1, n = 1:6
 plot(x -> L(λ, x), iterations=n)
end`

Out[15]:

Análisis gráfico de órbitas

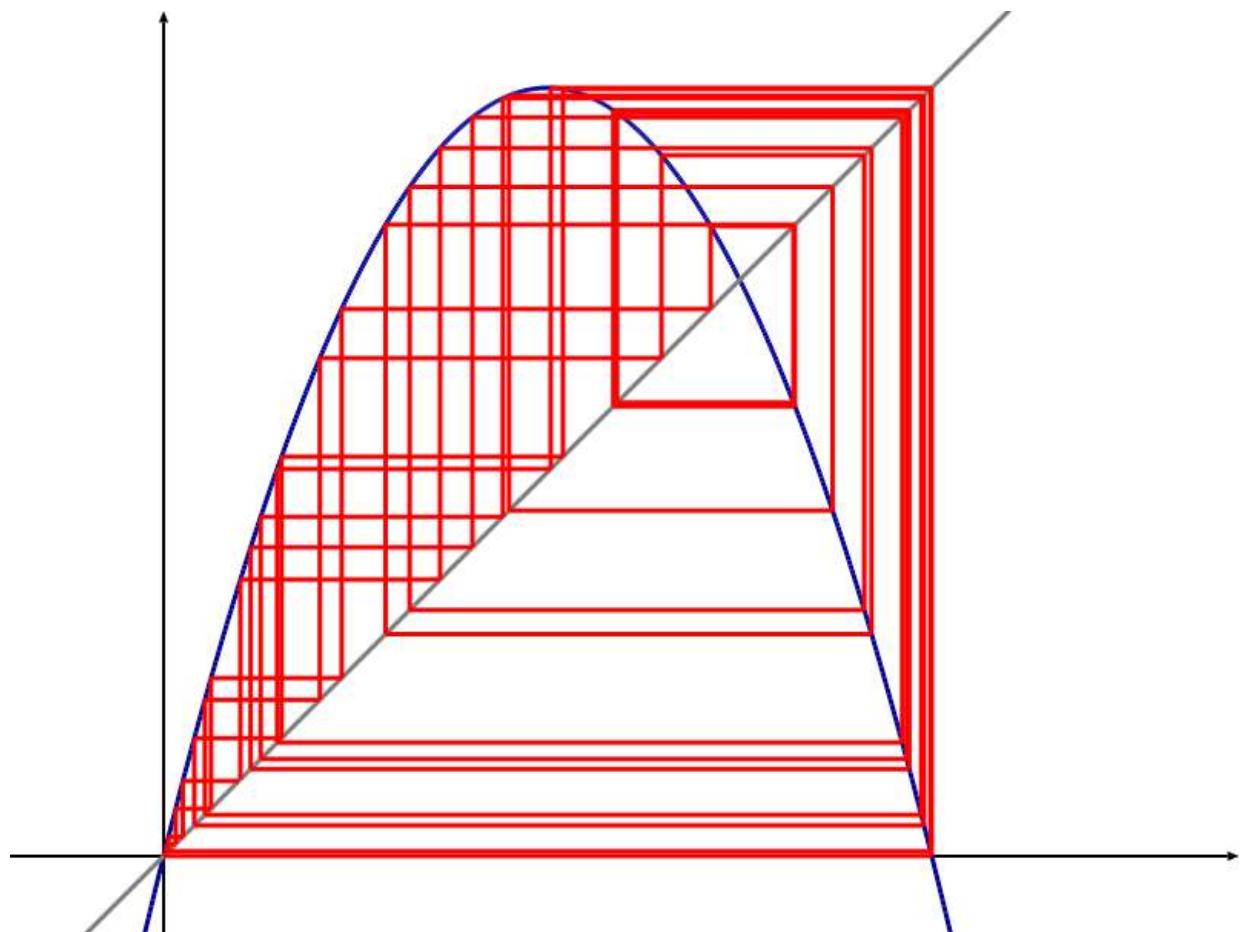
Dada $f : \mathbb{R} \rightarrow \mathbb{R}$ podemos graficar el análisis gráfico de la órbita de un punto con la instrucción `graphicalanalysis`.

Ejemplo

$$L(x) = 4x(1 - x).$$

In [31]: `graphicalanalysis(L4, 0.1, iterations=40)`

Out[31]:



Diagramas de órbitas atractoras (de bifurcación)

Dada la familia $f_\lambda : \mathbb{R} \rightarrow \mathbb{R}$ podemos graficar el diagrama de órbitas atractoras (de bifurcación) con la instrucción `orbitsdiagram`.

Ejemplo

La familia logística $L_\lambda(x) = \lambda x(1 - x)$.

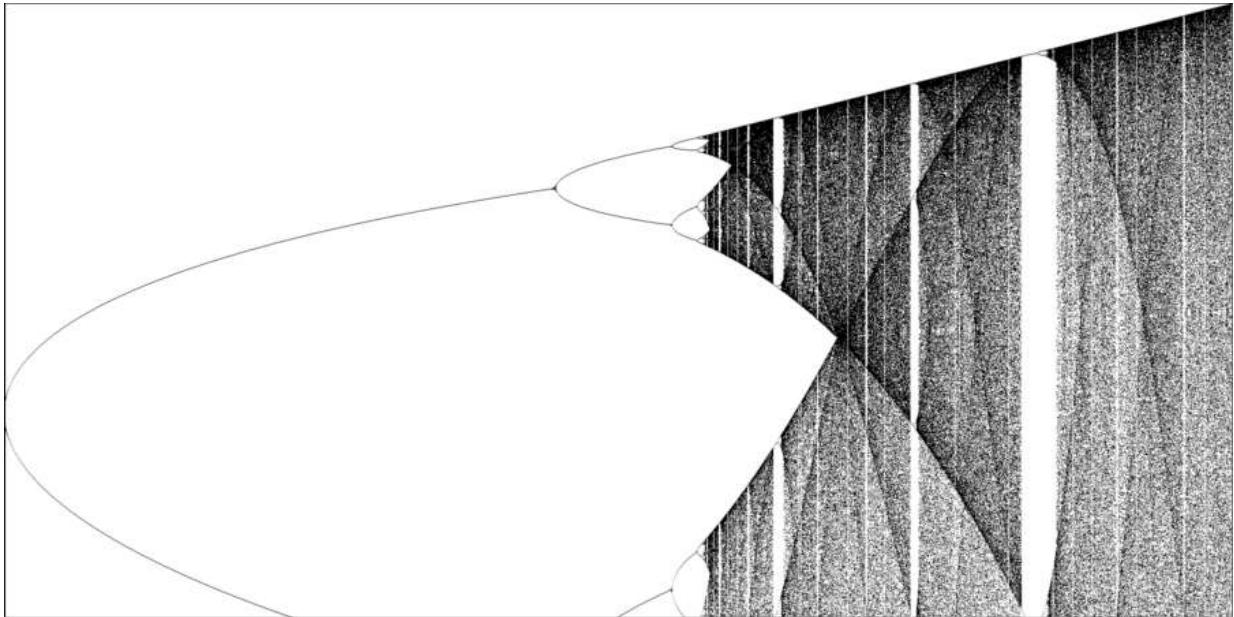
In [32]: $L(\lambda, x) = \lambda * x * (1 - x)$

Out[32]: L (generic function with 1 method)

In [33]:

```
Gr.backend(:images)
Gr.configure(rect=(3,4,0.5,1), canvas=(1800,900))
draworbitsdiagram(L, 0.5, preiterations=500, iterations=1000)
```

Out[33]:



In []:

Órbitas de puntos

Dada una función en \mathbb{R}^2 o \mathbb{C} podemos dibujar la órbita de un punto con `drawpointorbitR2` o `drawpointorbitC`, la trayectoria de la órbita de un punto con `drawpointorbitpathR2` o `drawpointorbitpathC`, y la órbita de varios puntos con `drawpointssetorbitR2` o `drawpointssetorbitC`.

Ejemplo

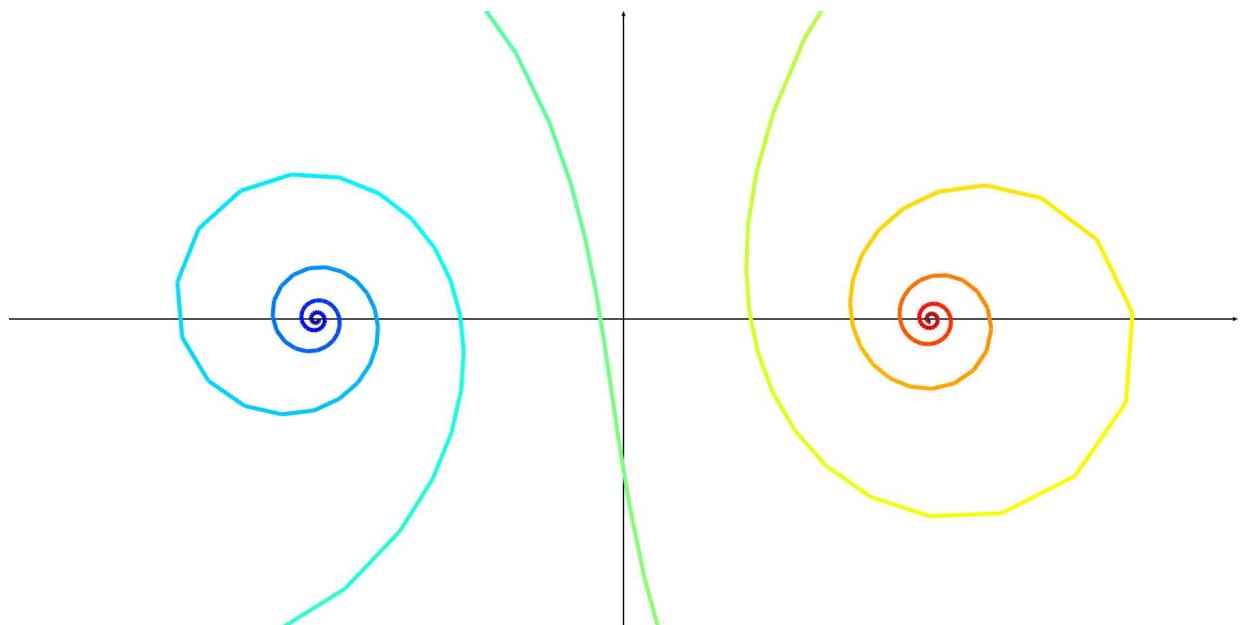
$S(z) = \frac{(1+6i)z+1}{z+1+6i}$, una transformación de Möbius loxodrómica.

In [34]: $S(z) = ((1+6im)*z+1)/(z+1+6im)$

Out[34]: S (generic function with 1 method)

```
In [35]: Gr.backend(:luxor)
Gr.configure(rect=(-2,2,-1,1), linewidth=6, colormap=:jet)
drawpointorbitpathC(S, -0.99, iterations=200)
```

Out[35]:



In []:

Ejemplo

$H(x, y) = (1.4x + 0.3y - x^2, x)$, una transformación de Hénon.

```
In [36]: H(x, y) = (1.4 + 0.3y - x^2, x)
```

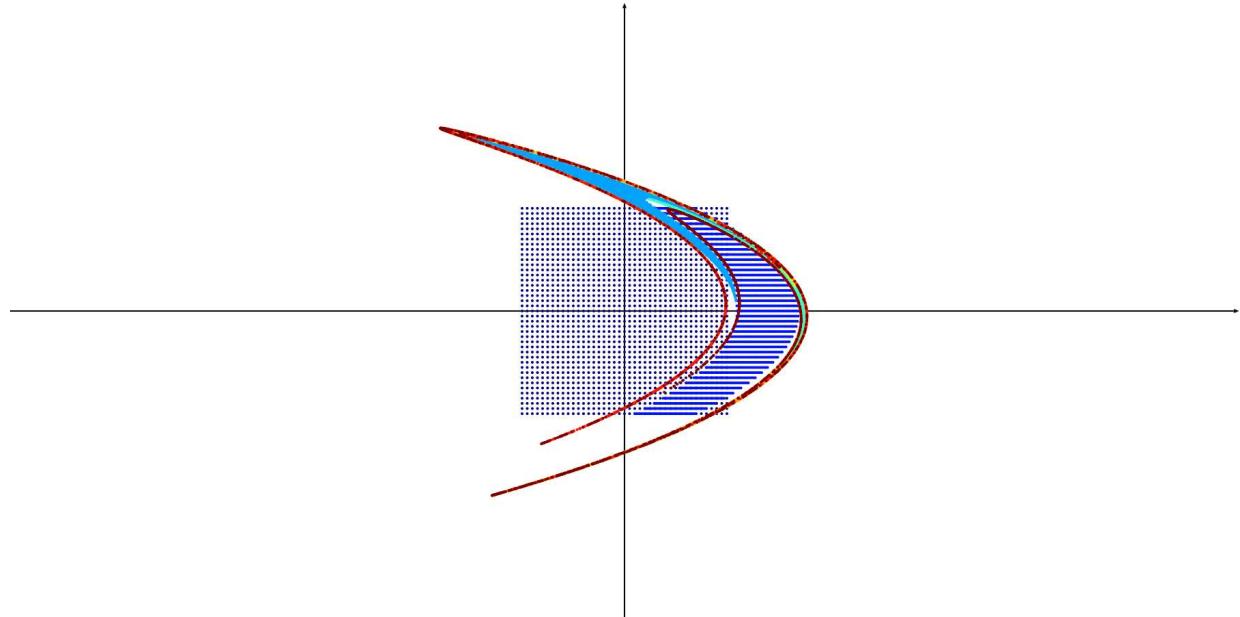
Out[36]: H (generic function with 1 method)

```
In [37]: square = [ (x,y) for x in -1:0.05:1 for y in -1:0.05:1 ];
```

In [38]:

```
Gr.configure(rect=(-6,6,-3,3), pointsize=2)
drawpointssetorbitR2(H, square, iterations=8)
```

Out[38]:



In []:

Conjuntos de puntos atrapados

Dada una función en \mathbb{R}^2 o \mathbb{C} podemos dibujar el conjunto de puntos atrapados con `drawtrappedpointsR2` o `drawtrappedpointsC`.

Ejemplo

$$F_1(x, y) = (x^2 - y^2 - x, 2xy + y), \text{ función en } \mathbb{R}^2.$$

In [39]:

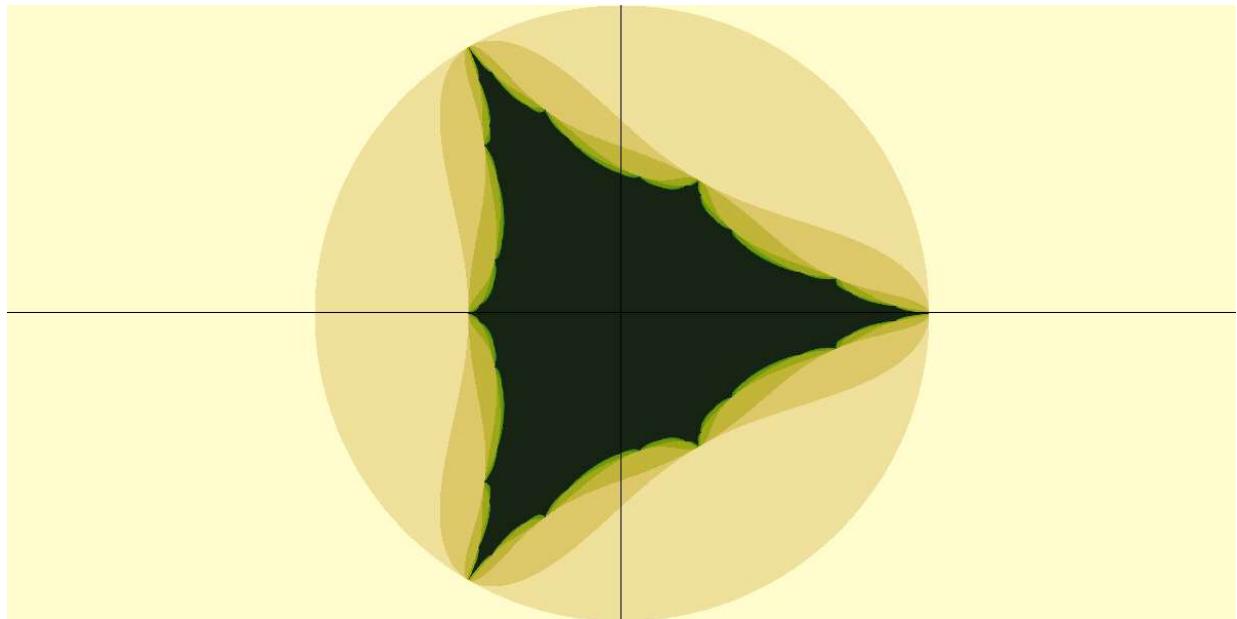
```
F1(x,y)=(x^2-y^2-x,2x*y+y)
```

Out[39]:

F1 (generic function with 1 method)

```
In [40]: Gr.backend(:images)  
Gr.configure(rect=(-4,4,-2,2), canvas=(1200,600), colormap=:speed)  
drawtrappedpointsR2(F1, maxiterations = 12)
```

Out[40]:



In []:

Ejemplo

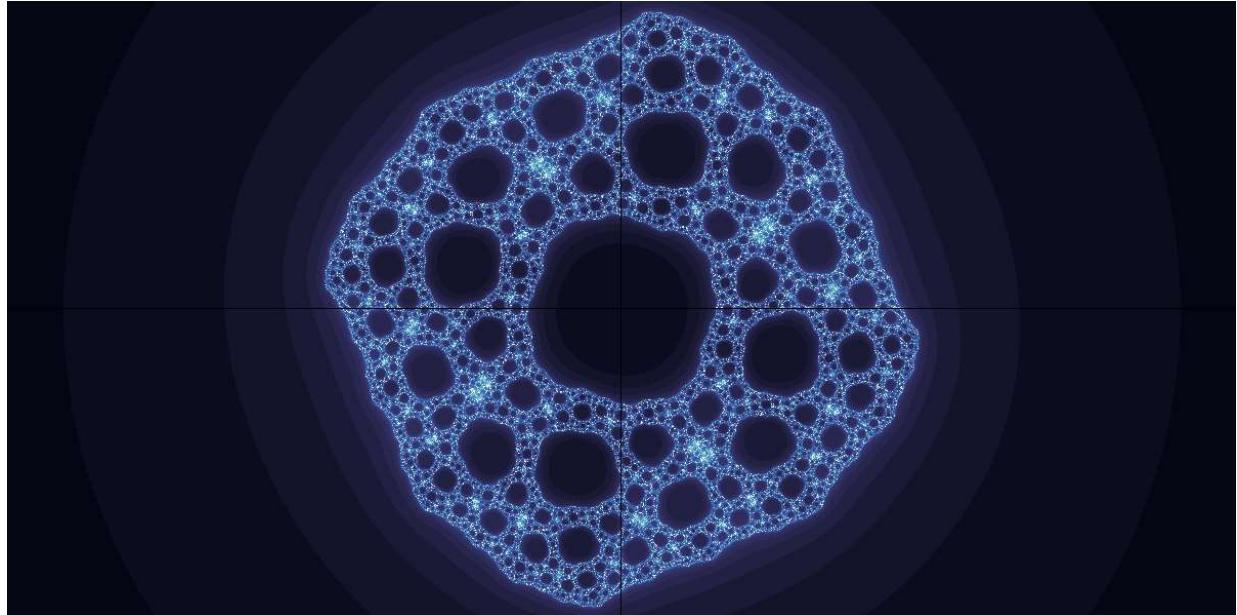
$R(z) = \frac{z^3 - 0.05 + 0.1i}{z^2}$, una función racional en \mathbb{C} .

```
In [41]: R(z)=(z^4-0.05+0.1im)/(z^2)
```

Out[41]: R (generic function with 1 method)

In [42]: `Gr.configure(rect=(-2.2,2.2,-1.1,1.1), colormap=:ice)
drawtrappedpointsC(R, maxiterations = 32)`

Out[42]:



In []:

Ejemplo

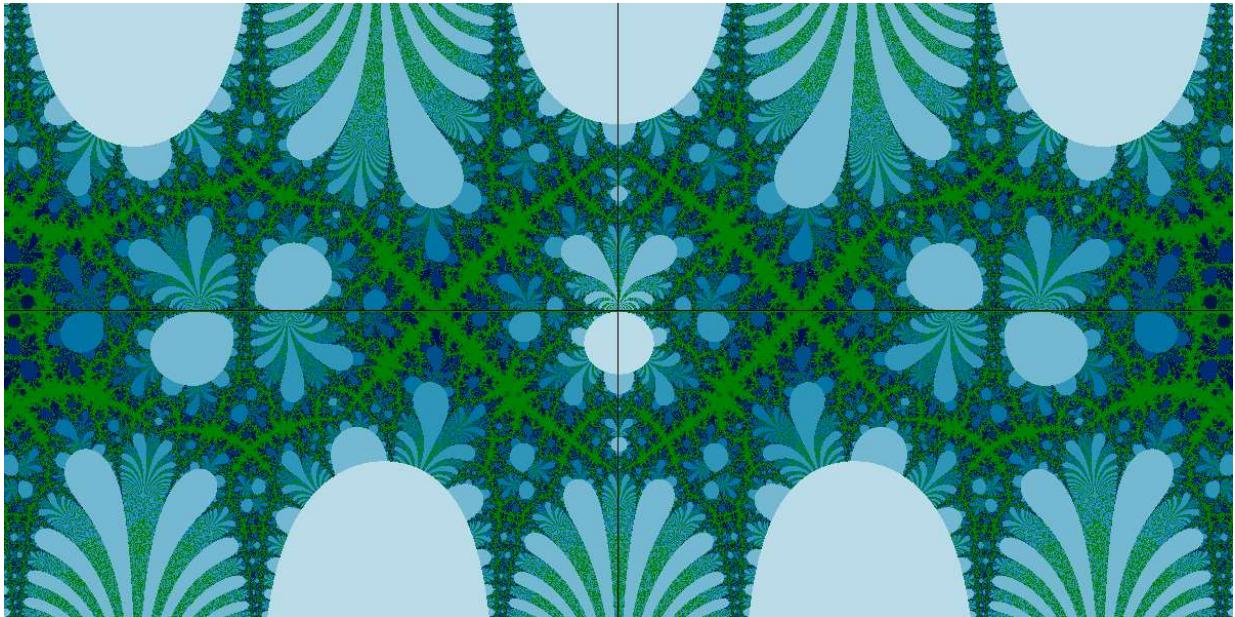
$m(z) = \sin(z) + \frac{4}{z}$, una función meromorfa trascendente.

In [43]: `m(z) = sin(z) + 4/z`

Out[43]: `m (generic function with 1 method)`

```
In [44]: Gr.configure(rect=(-8,8,-4,4), colormapinv=:ocean)
drawtrappedpointsC(m, hasescaped = z -> imag(z) > 4, maxiterations = 12)
```

Out[44]:



In []:

Ejemplo

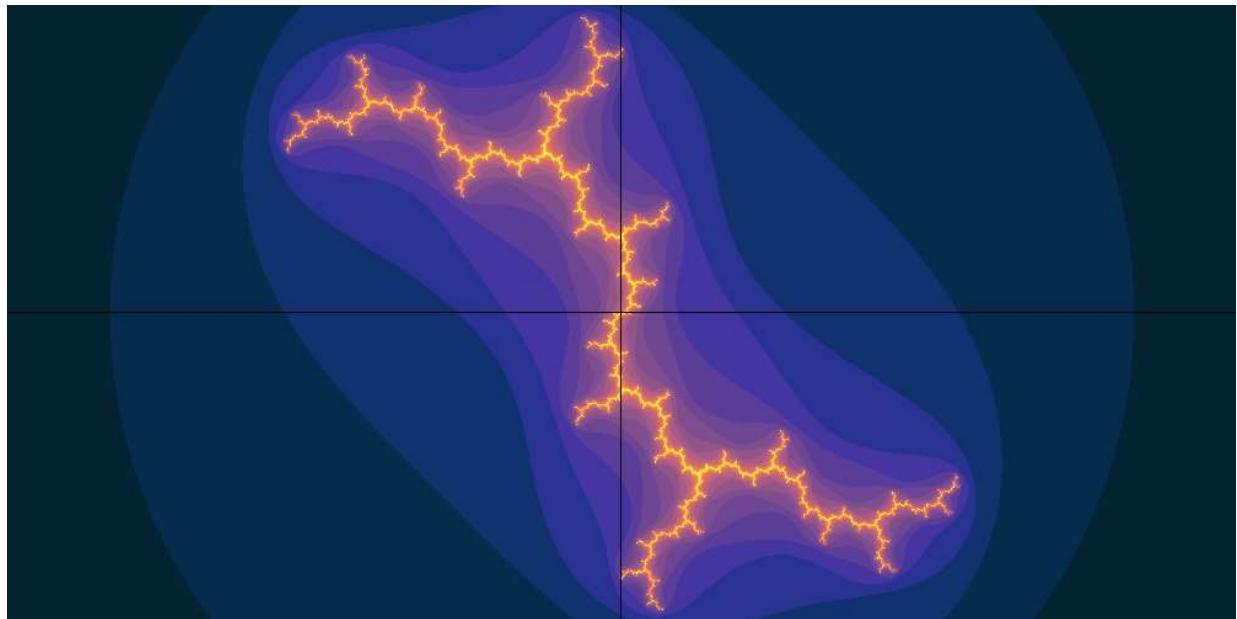
La familia cuadrática $q_c(z) = z^2 + c$ en \mathbb{C} .

```
In [17]: q(c,z) = z^2 + c
```

Out[17]: q (generic function with 1 method)

```
In [19]: Gr.backend(:images)
Gr.configure(rect=(-2.4,2.4,-1.2,1.2), canvas=(1000,500), colormap=:thermal)
drawtrappedpointsC(z -> q(im,z), maxiterations = 20)
```

Out[19]:



```
In [20]: @manipulate for x=-2.0:0.1:2.0, y=-2.0:0.1:2.0, n=1:32
    drawtrappedpointsC(z -> q(complex(x,y),z), maxiterations = n)
end
```

Out[20]:

Conjuntos de Mandelbrot

Dada una familia de funciones en \mathbb{R}^2 o \mathbb{C} podemos dibujar el conjunto de Mandelbrot con `drawmandelbrotR2` o `drawmandelbrotC`.

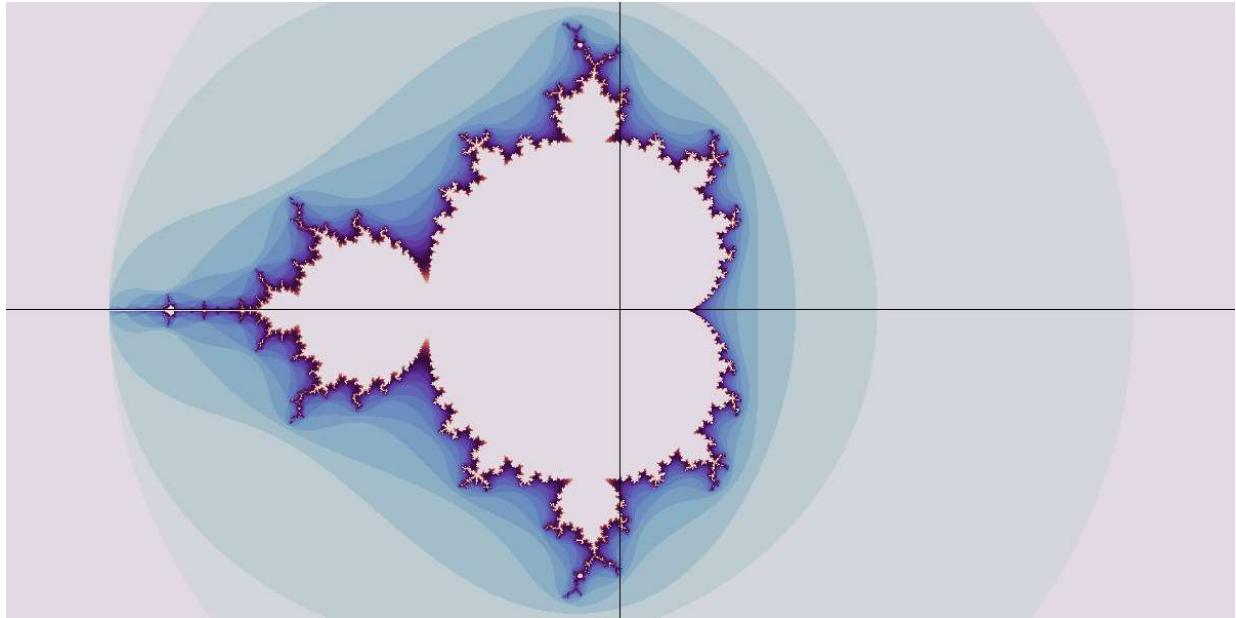
Ejemplo

La familia cuadrática $q_c(z) = z^2 + c$ en \mathbb{C} .

In [49]:

```
Gr.configure(rect=(-2.4,2.4,-1.2,1.2), colormap=:twilight)
drawmandelbrotC(q, maxiterations = 30)
```

Out[49]:

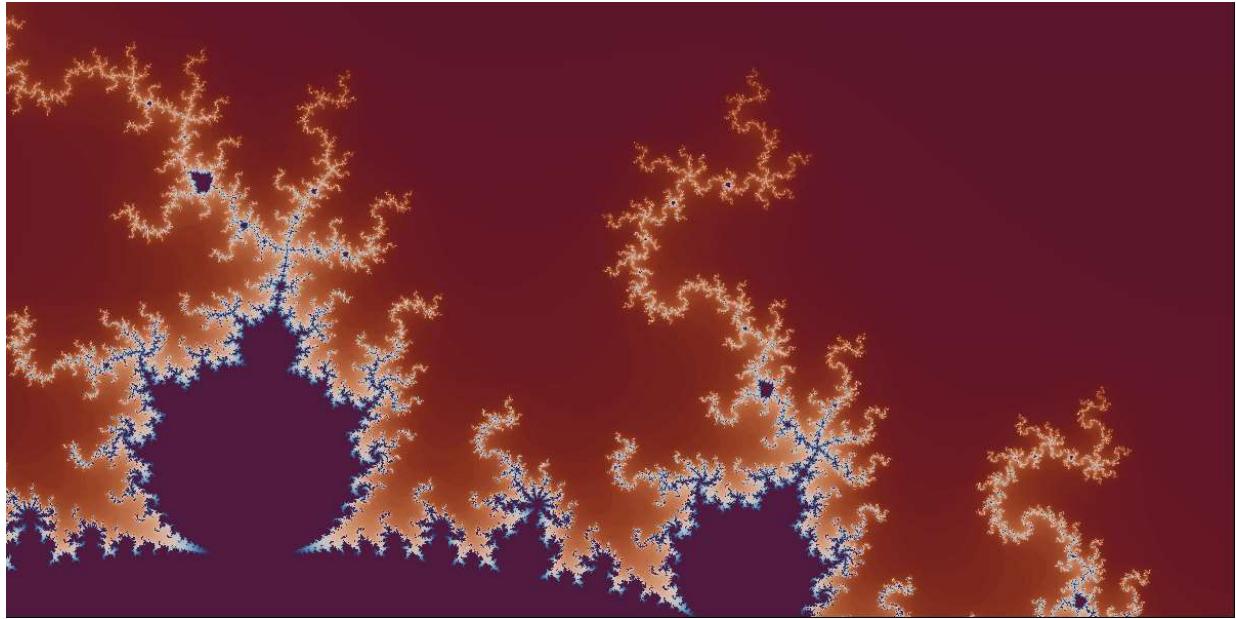


In []:

In [188]:

```
Gr.backend(:images)
Gr.configure(rect=(-1.04,-0.84,0.24,0.34), canvas=(1200,600), colormapinv=:vik0)
drawmandelbrotC(q, maxiterations = 120)
```

Out[188]:



In []:

Ejemplo

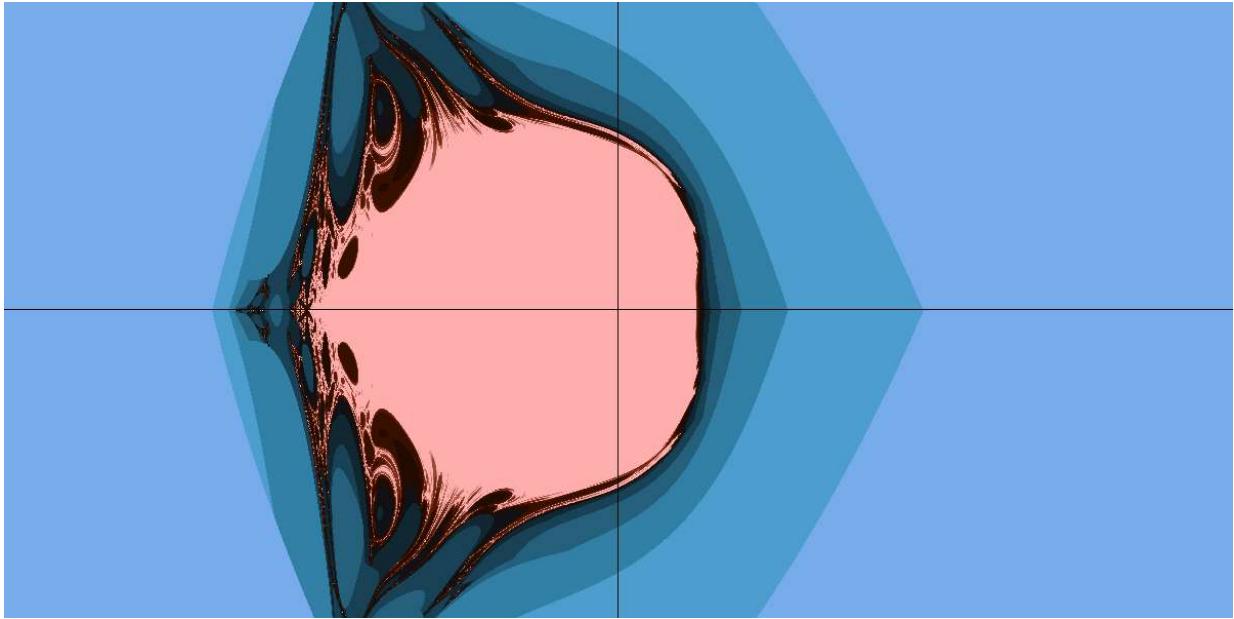
$F_{a,b}(x, y) = (x^2 - y^2 + ax + by, 2xy + -ay + bx)$, una familia en \mathbb{R}^2 .

In [50]: `F(a,b,x,y)=(x^2-y^2+a*x+b*y,2x*y-a*y+b*x)`

Out[50]: F (generic function with 1 method)

In [51]: `Gr.configure(rect=(-4,4,-2,2), colormap=:berlin)
drawmandelbrotR2(F, 0.5, hasescaped = (x,y) -> abs(x)+abs(y) > 4, maxiterations =`

Out[51]:



In []:

Cuencas de atracción

Dada una función en \mathbb{R}^2 o \mathbb{C} y una lista de puntos periódicos atractores o parabólicos, podemos dibujar sus cuencas de atracción con `drawbasinsR2` o `drawbasinsC`.

Ejemplo

$N(z) = z - \frac{1-z^3}{-3z^2}$, el método de Newton-Raphson para $f(z) = 1 - z^3$.

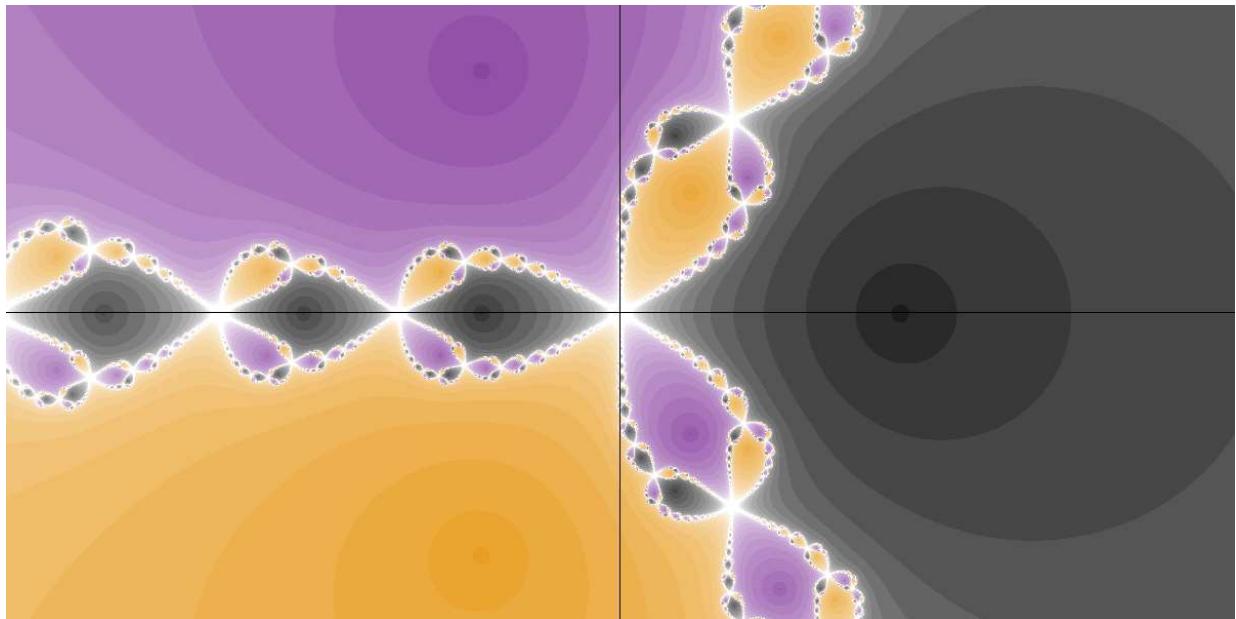
In [52]: `N(z)=z+(1-z^3)/(3z^2)`

Out[52]: N (generic function with 1 method)

In [53]:

```
Gr.configure(rect=(-2.2,2.2,-1.1,1.1), colormap=:CMRmap)
drawbasinsC(N, [1,exp(2pi*im/3),exp(4pi*im/3)], maxiterations=18)
```

Out[53]:



In []:

Preimágenes

Dada una función en \mathbb{R}^2 o \mathbb{C} , podemos dibujar los mapas de preimágenes de las n -ésimas iteradas con `drawpreimagesR2` o `drawpreimagesC`.

Ejemplo

$R_{herman}(z) = e^{2\pi(0.6151732...i)} \frac{z^2(z-4)}{1-4z}$, una función racional con anillo de Herman.

In [54]:

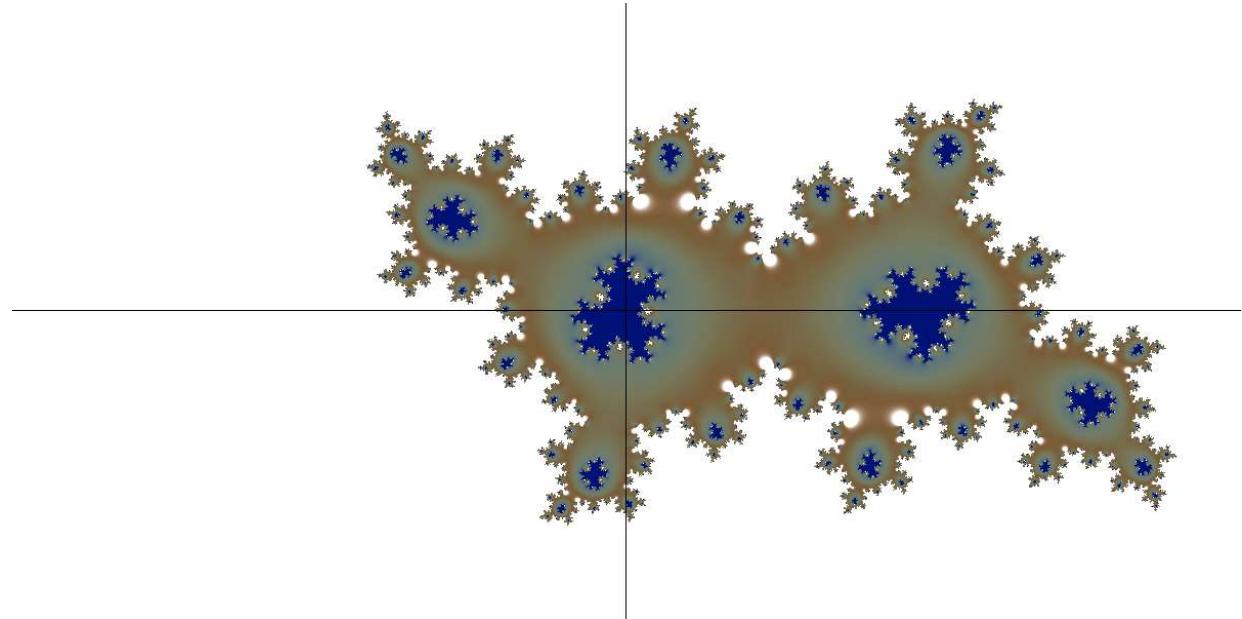
```
Herman(z:Number) = exp(2pi*0.6151732*im)*(z^2)*(z-4)/(1-4z)
```

Out[54]:

```
Herman (generic function with 1 method)
```

In [55]: `Gr.configure(rect=(-8,8,-4,4), colormap=:darkterrain, cf=RadialCF(0,0,innerradius=1, outerradius=4), drawpreimageC(Herman, iterations=24))`

Out[55]:



In []:

Ejemplo

Con la familia cuadrática $q_c(z) = z^2 + c$ en \mathbb{C} , puntos de periodo 3 con $f(z) = q_c^{36}(z) - z$.

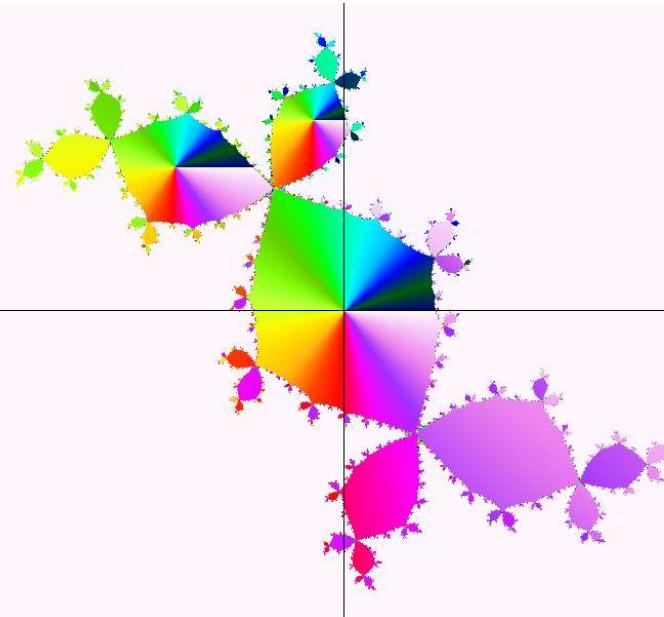
In [56]: `qrabbit = z -> q(-0.122656 + 0.744864im, z)`
`qrabbit36 = SDDCore.compose(qrabbit, 36)`
`f36(z::Number) = qrabbit36(z) - z`

Out[56]: `f36 (generic function with 1 method)`

In [57]:

```
Gr.configure(rect=(-2.4,2.4,-1.2,1.2), colormap=:gist_ncar, cf=AngleCF())
drawpreimageC(f36, iterations=1)
```

Out[57]:



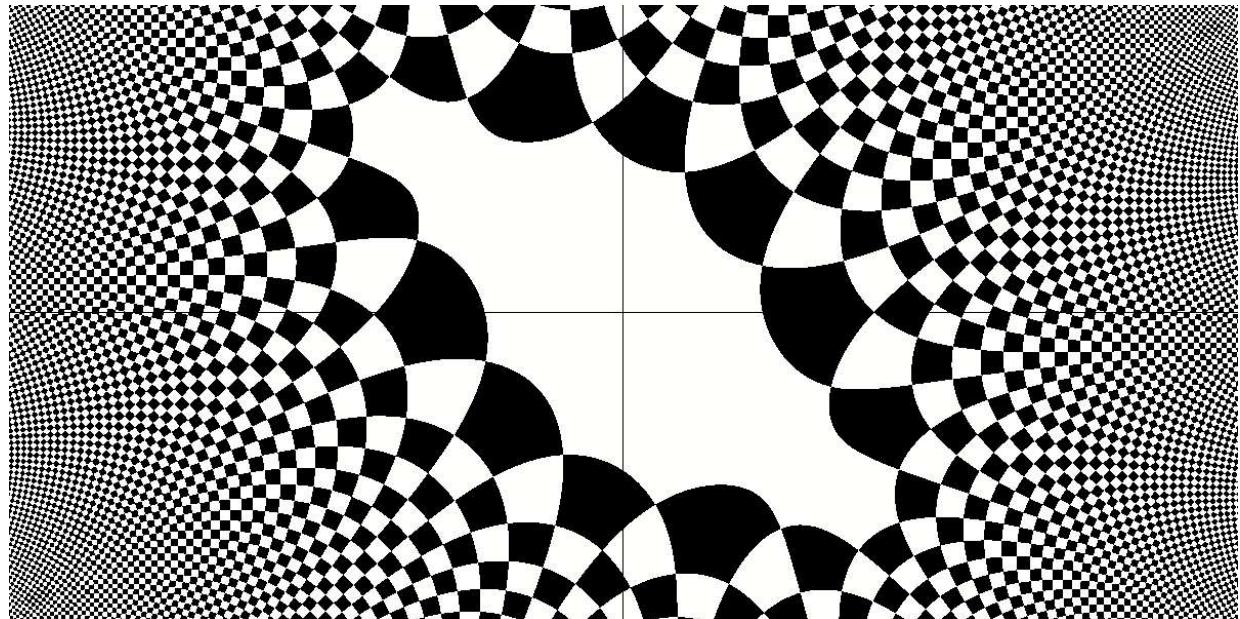
In []:

Ejemplo

Preimagen de un tablero de ajedrez bajo q_c^2 .

In [58]: `Gr.configure(rect=(-2.4,2.4,-1.2,1.2), colormap=:grays, cf=ChessCF(0,0,1,1))
drawpreimageC(qrabbit, iterations=2)`

Out[58]:



In []:

La biblioteca **SDDIFS**

In []: `using SDDIFS`

Atractores

Dado un *sistema de funciones iteradas*, podemos dibujar su atractor con la instrucción `drawattractorC`.

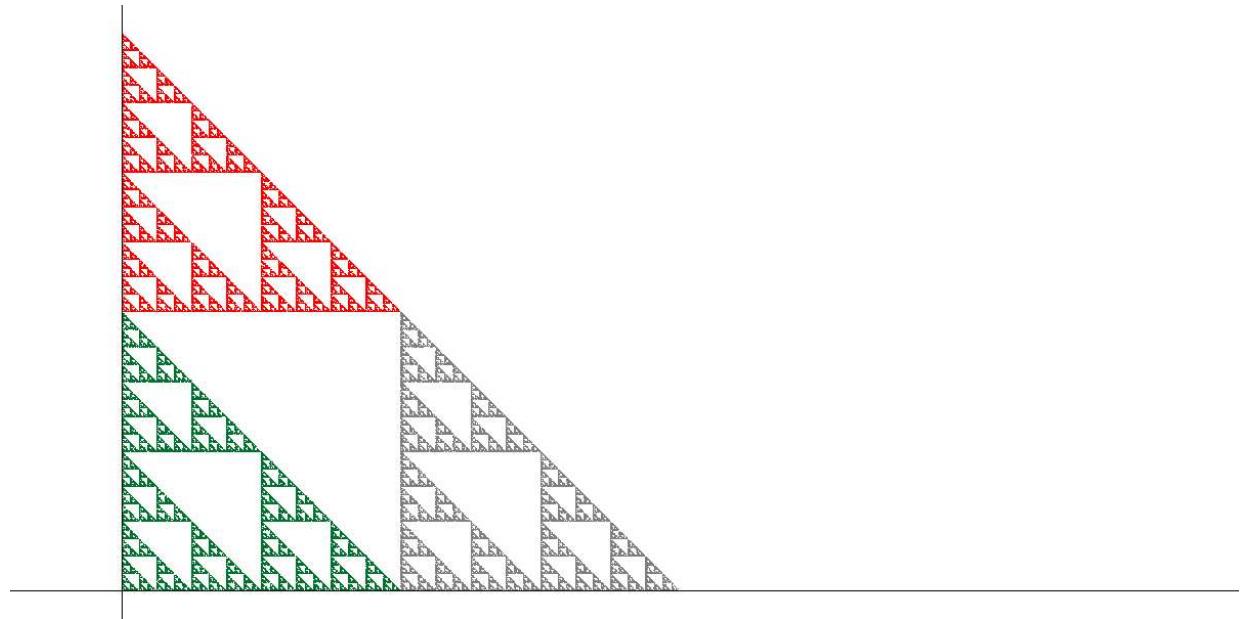
Ejemplo

Un triángulo de Sierpinsky como atractor de $\{z \mapsto \frac{1}{2}z, z \mapsto \frac{1}{2}z + 1, z \mapsto \frac{1}{2}z + i\}$.

```
In [59]: sierpinsky = [LinearTransformation(0.5), AffineTransformation(0.5,1), AffineTrans
```

```
In [60]: Gr.configure(rect=(-0.4,4,-0.1,2.1), colormap=:flag_ae)
drawattractorC(sierpinsky, iterations=100000)
```

Out[60]:



```
In [ ]:
```

Ejemplo

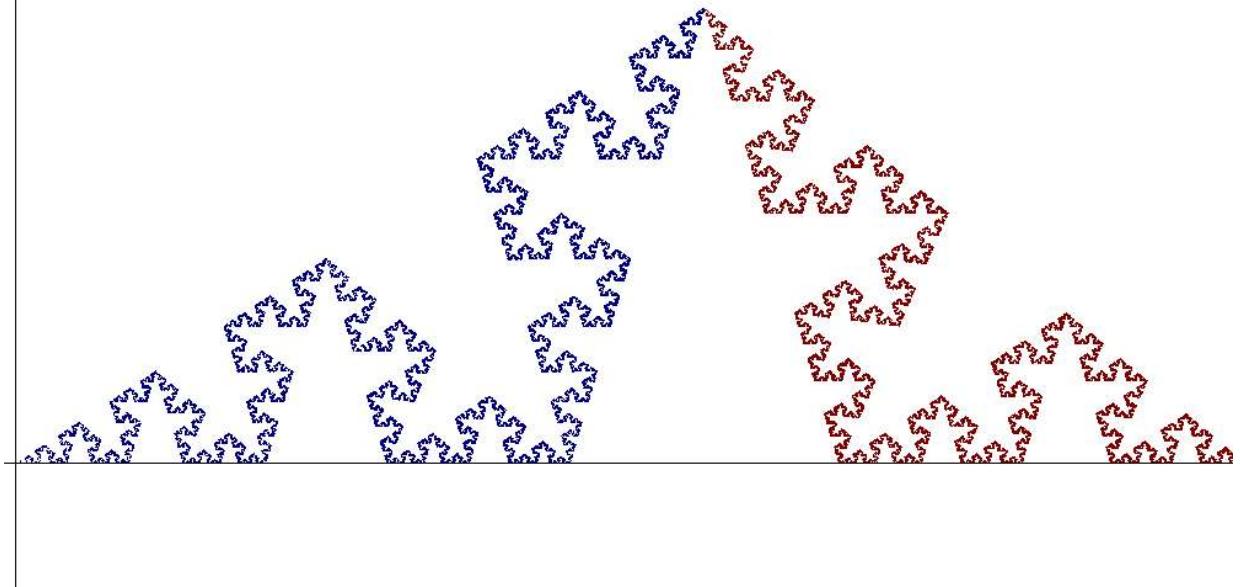
Una curva de Koch-Peano como atractador de $\{z \mapsto a\bar{z}, z \mapsto a + (1 - a)\bar{z}\}$.

```
In [37]: a = 0.56+0.37im
kochpeano=[z -> a*conj(z), z -> a+(1-a)*conj(z)];
```

In [38]:

```
Gr.configure(rect=(-0.01,0.99,-0.1,0.4), colormap=:jet)
drawattractorC(kochpeano, iterations=100000)
```

Out[38]:



In [39]:

```
}
```

syntax: unexpected "}"

Stacktrace:

```
[1] top-level scope
    @ In[39]:1
[2] eval
    @ .\boot.jl:360 [inlined]
[3] include_string(mapexpr::typeof(REPL.softscope), mod::Module, code::String,
filename::String)
    @ Base .\loading.jl:1116
```

La biblioteca *SDDKleinianGroups*

In []:

```
using SDDKleinianGroups
```

Órbitas de puntos

Podemos dibujar las órbitas de puntos o conjuntos de puntos con `drawpointorbit` y `drawpointssetorbit`.

Ejemplo

Carpeta de Apolonio con el grupo $\langle A_1, A_2 \rangle$ dado por

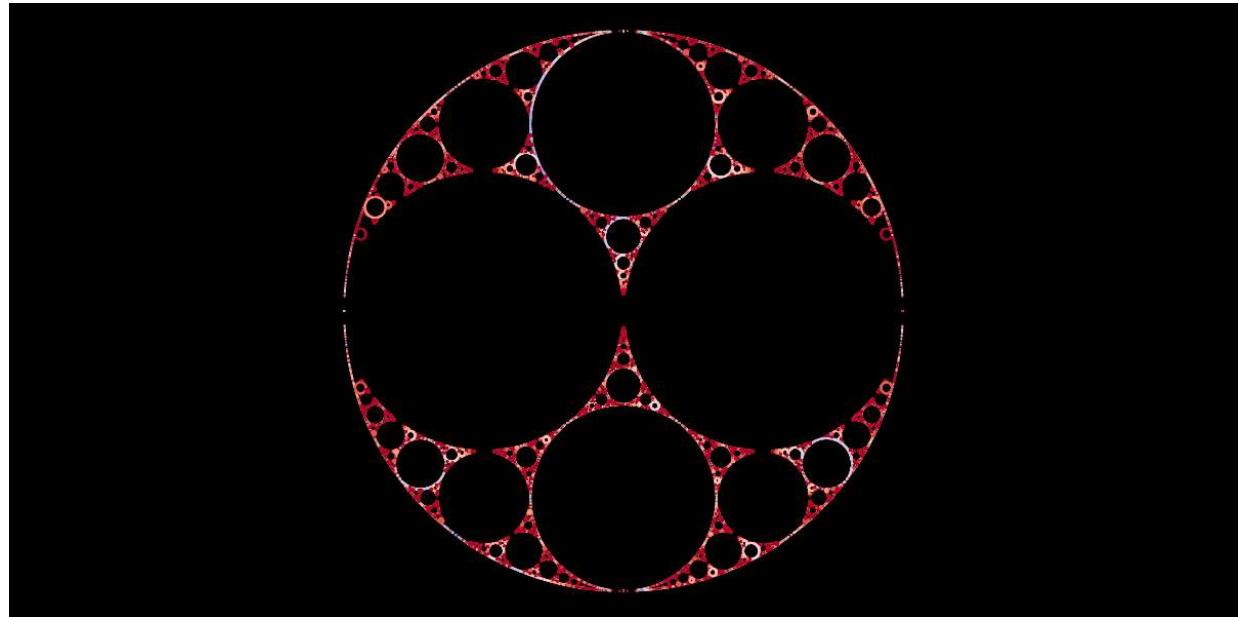
$$A_1(z) = \frac{z}{-2iz + 1}$$

$$A_2(z) = \frac{(1-i)z + 1}{z + 1 + i}$$

In [63]: `apollonian = [MobiusTransformation(1,0,-2im,1), MobiusTransformation(1-im,1,1,1+i)]`

In [190]: `Gr.backend(:luxor)
Gr.configure(rect=(-2.2,2.2,-1.1,1.1), pointsize=1, colormap=:coolwarm, bgcolor=:white)
drawpointssetorbit(apollonian, circlecomplexes(0,1), iterations=8)`

Out[190]:



In []:

Órbitas de círculos

Podemos dibujar también órbitas de círculos o conjuntos de círculos, con `drawcircleorbit` y `drawcirclesetorbit`.

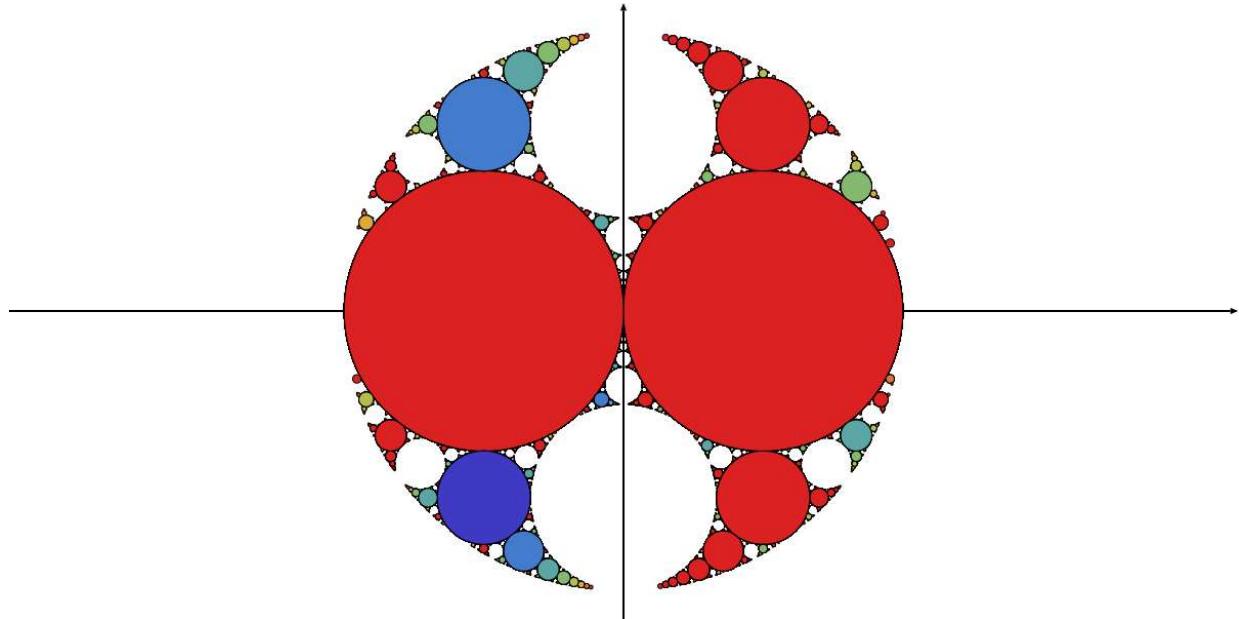
Ejemplo

Carpeta de Apolonio, con la órbita del círculo con centro en $-\frac{1}{2}$ y radio $\frac{1}{2}$.

In [64]:

```
Gr.backend(:luxor)
Gr.configure(rect=(-2.2,2.2,-1.1,1.1), style=:fillborder, colormap=:rainbow, lw=1)
drawcircleorbit(apollonian, Circle(-0.5,0.5), iterations=8)
```

Out[64]:



In []:

Ejemplo

Grupo de Schottky.

In [65]:

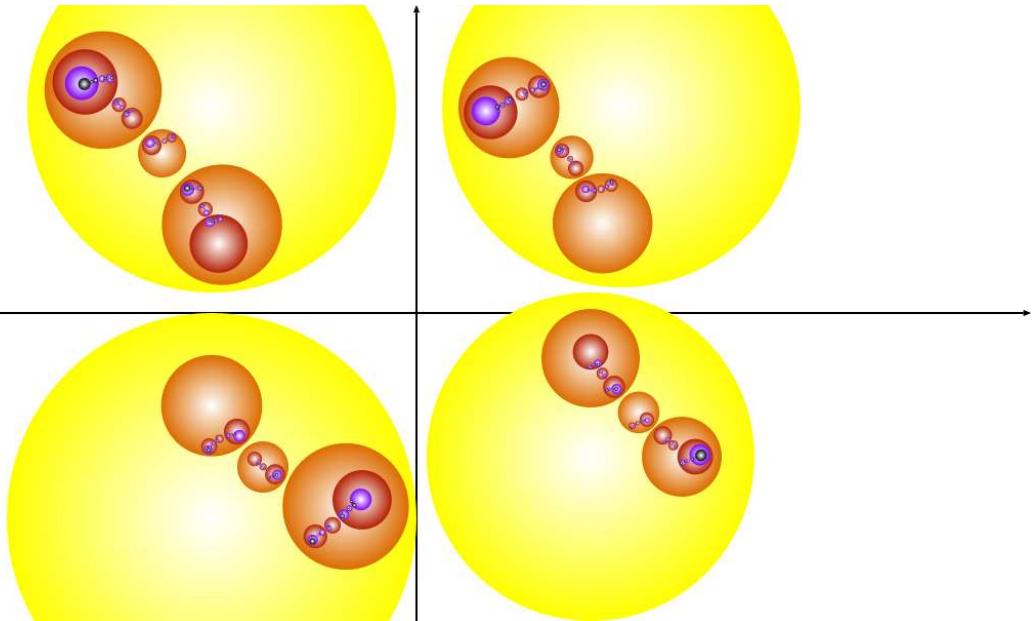
```
Da, Db, DA, DB = Circle(2.0+2im,1.75), Circle(1.7-1.4im,1.6), Circle(-2.0-2im,2.0)
schottkycircles = [Da, Db, DA, DB];

fa = MobiusTransformation(pickpoint(DA,0), pickpoint(DA,π/4), pickpoint(DA,π/2),
                         pickpoint(DA,3π/2), pickpoint(DA,5π/4), pickpoint(DA,π)) # DA to Da
fb = MobiusTransformation(pickpoint(DB,0), pickpoint(DB,π/2), pickpoint(DB,π),
                         pickpoint(Db,0), pickpoint(Db,3π/2), pickpoint(Db,π)) # DB to Db

schottkygenerators = [fa, fb];
```

In [66]: `Gr.configure(rect=(-6,6,-3,3), style=:pearl, colormapinv=:gnuplot)
drawcirclesetorbit(schottkygenerators, schottkycircles, iterations=4, traverse=:`

Out[66]:



In []:

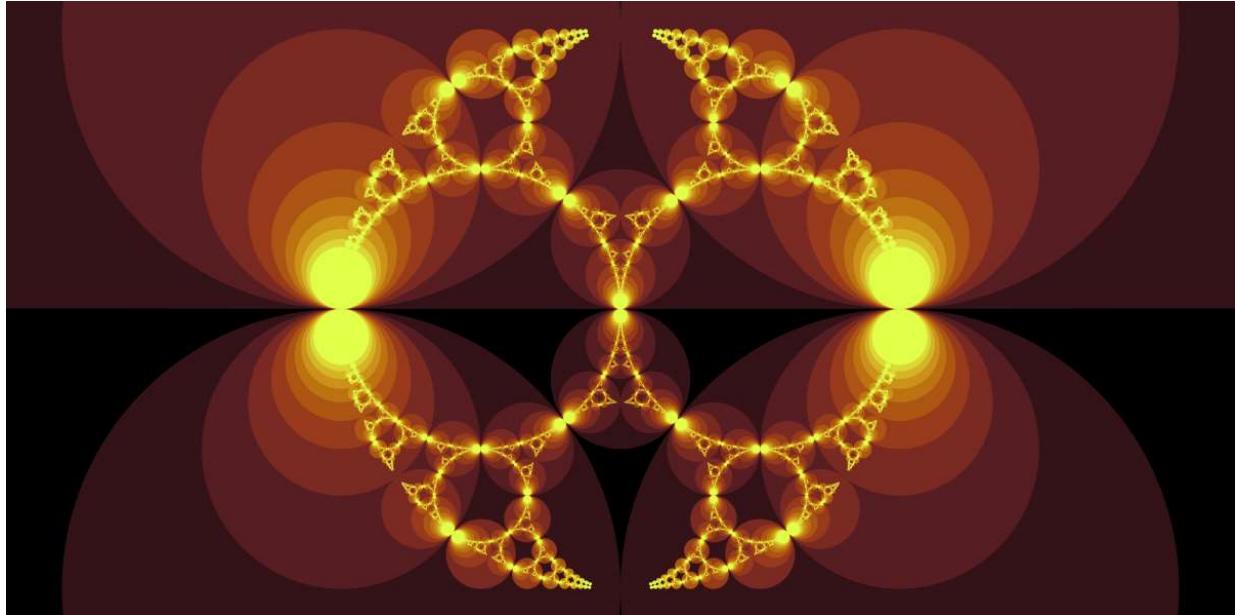
Ejemplo

La carpeta de Apolonio con otros círculos.

In [67]: `specialcircles = [Circle(1000im,1000),Circle(-1-im,1), Circle(-0.25im,0.25), Circ`

In [68]: `Gr.configure(rect=(-2.2,2.2,-1.1,1.1), style=:fill, colormap=:solar, bgcolor=RGB(0.1,0.1,0.1), drawcircles=orbit(apollonian, specialcircles, iterations=9, traverse=:firstbreak))`

Out[68]:



In []:

Conjuntos límite

Podemos dibujar conjuntos límite, con varios algoritmos, utilizando `drawΛ`, `drawfixedpointsΛ` o `drawchaosgameΛ`.

Ejemplo

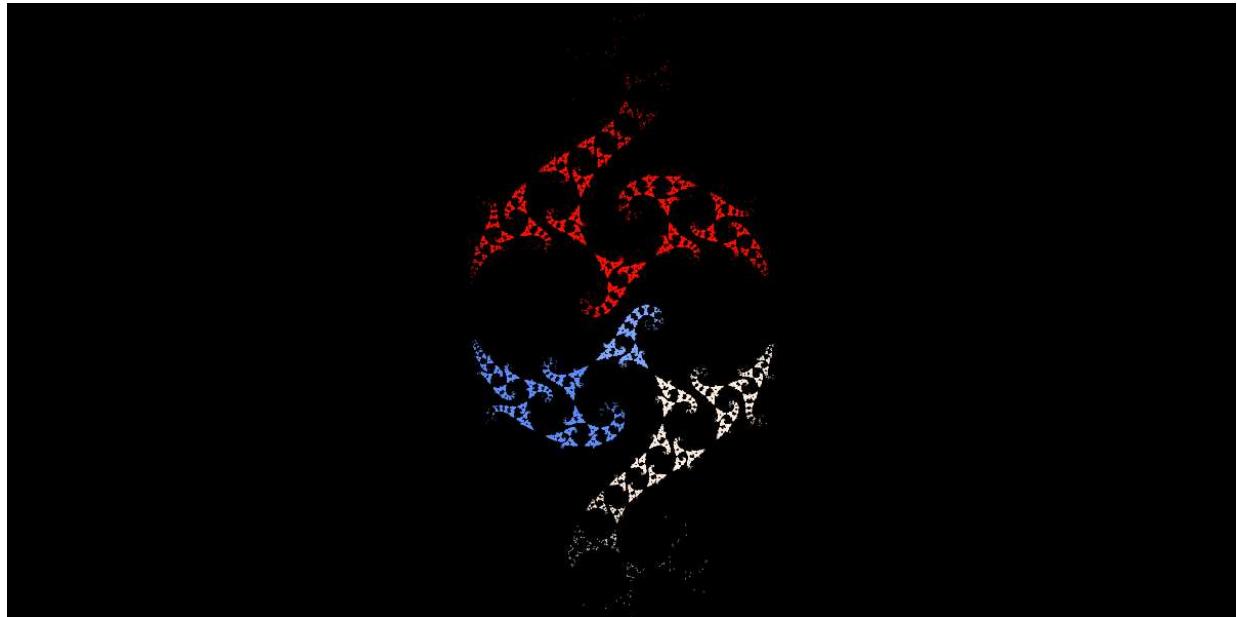
Un grupo quasi-Fuchsiano.

In [69]: `ta, tb = 1.87 + 0.1im, 1.87 - 0.1im
tab = (ta*tb + sqrt(complex(ta*ta*tb*tb-4*(ta*ta+tb*tb))))/2
z0 = (tab-2)*tb/(tab*tb-2ta+2im*tab)
a1, b1, c1 = ta/2, (ta*tab-2tb+4im)/((2tab+4)*z0), (ta*tab-2tb-4im)*z0/(2tab-4)
a2, b2, d2 = (tb-2im)/2, tb/2, (tb+2im)/2

quasifuchsian = [MobiusTransformation(a1,b1,c1,a1), MobiusTransformation(a2,b2,b2)]`

In [70]: `Gr.configure(rect=(-4,4,-2,2), pointsize=0.5, colormap=:blackbody, bgcolor=RGB(0, drawchaosgameΛ(quasifuchsian, iterations=200000)`

Out[70]:



In []:

Trabajo futuro

Estamos trabajando en lo siguiente, que debe estar finalizado a principios de Enero del 2022:

- En **SDD**
 - Diagramas de bifurcación de familias de funciones reales.
 - Implementaciones de más algoritmos de dibujo de cuencas de atracción.
 - Implementaciones de más algoritmos de dibujo de conjuntos de Mandelbrot.
 - Cálculo de exponentes de Lyapunov.

- En **SDDIFS**
 - Implementaciones de más algoritmos de dibujo de atractores.
 - Cálculo de dimensión fractal de atractores.

- En **SDDKleinianGroups**

- Implementaciones de más algoritmos de dibujo de conjuntos límite.
- Cálculo de dimensión fractal de conjuntos límite.

Y lo más importante:

- **Aplicación** de fácil uso para estudiantes, profesores e investigadores no programadores.

Todo el software que hemos desarrollado es libre y se puede descargar desde **GitHub**.

**github.com/Colectivo-SDD
(github.com/Colectivo-SDD)**

La versión 1.0 estable será publicada hasta Enero del 2022.

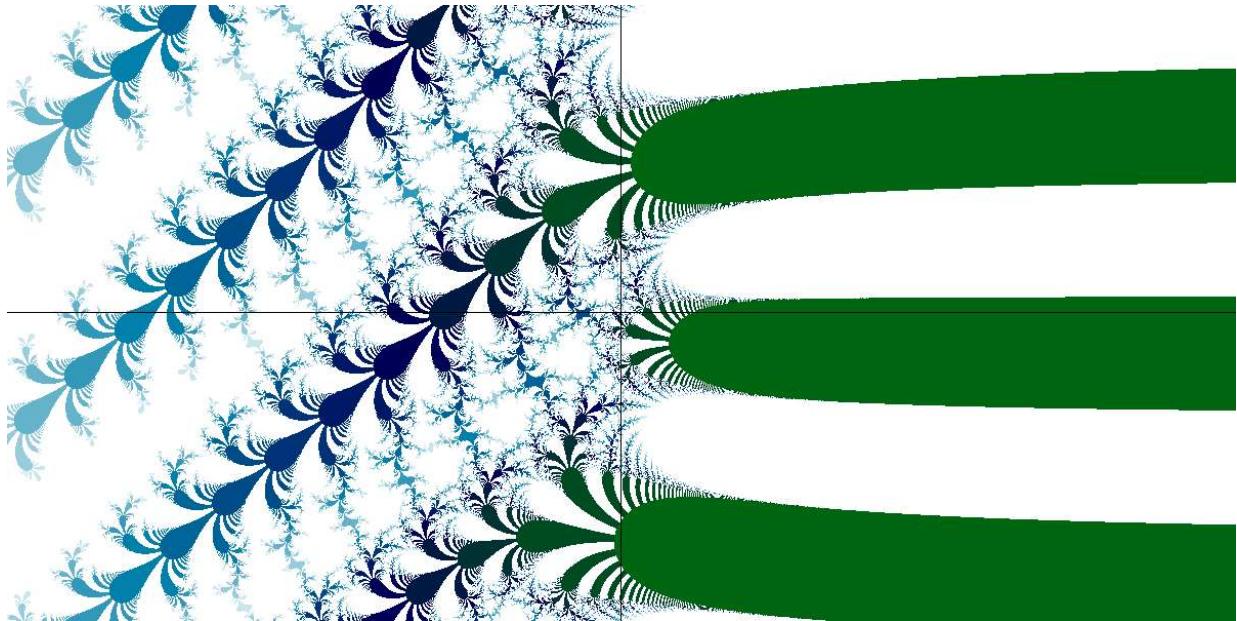
¡Gracias!

In [11]: `EW(λ::Number, z::Number) = λ*z*exp(z)`

Out[11]: EW (generic function with 1 method)

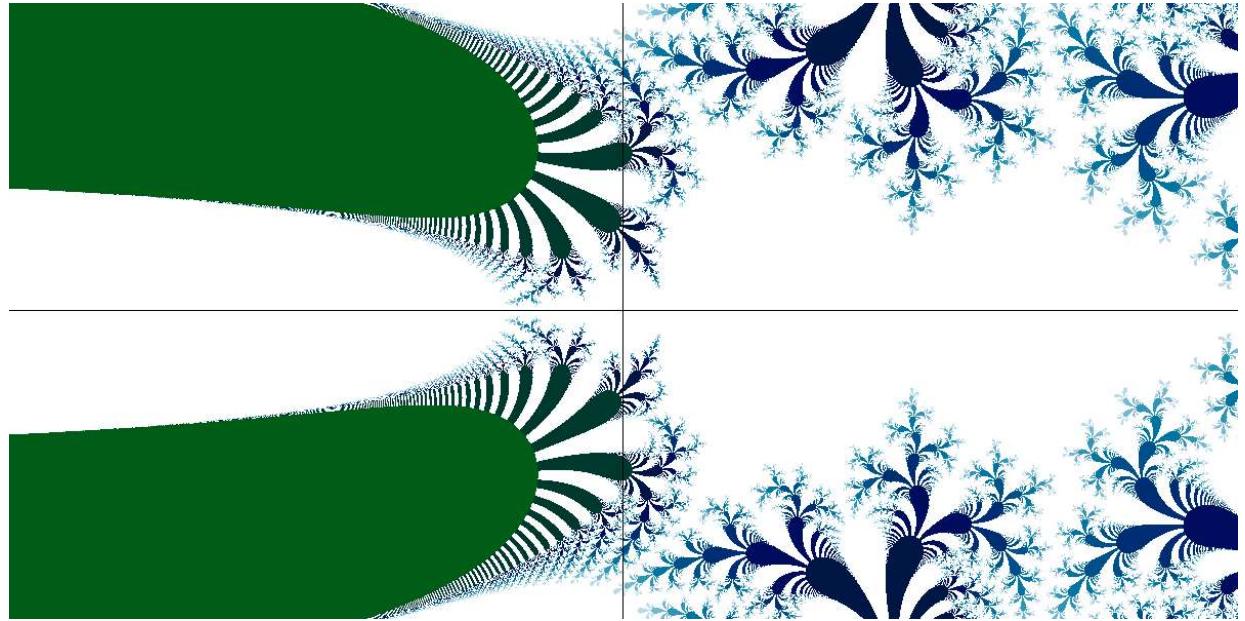
In [31]: `Gr.backend(:images)
Gr.configure(rect=(-16,16,-8,8), canvas=(1200,600), colormap=:ocean, axes=true)
drawtrappedpointsC(z -> EW(1.5+3im,z), hasescaped = z -> real(z) > 16, maxiterati`

Out[31]:



In [30]: `Gr.configure(rect=(-16,16,-8,8), canvas=(1200,600), colormap=:ocean, axes=true)
drawmandelbrotC(EW, -1, hasescaped = z -> real(z) > 16, maxiterations = 12)`

Out[30]:

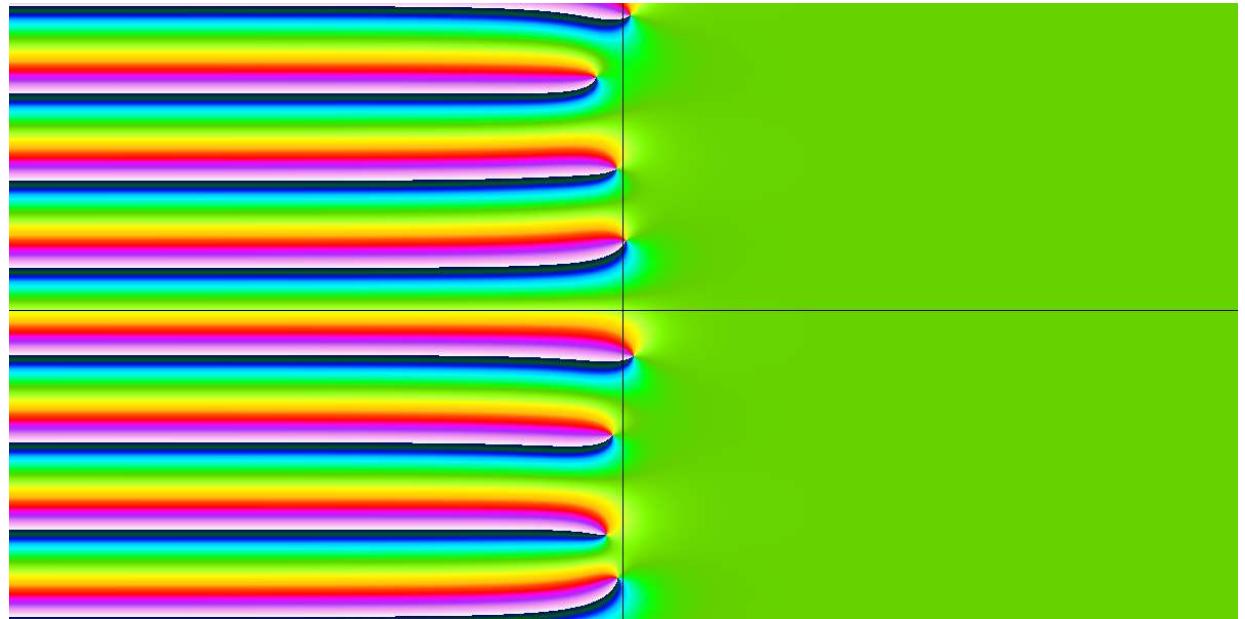


In [32]: `$\zeta_4(z::\text{Number}) = 1+1/(2^z)+1/(3^z)+1/(4^z)$`

Out[32]: `ζ_4 (generic function with 1 method)`

In [45]: `Gr.configure(rect=(-32,32,-16,16), colormap=:gist_ncar, cf=AngleCF(0,1))
drawpreimageC(ζ_4 , iterations=1)`

Out[45]:



In []:

