反復写像

分岐マップ

Bifurcation コマンドにより、反復写像の 2 次元画像を生成します。支配方程式におけるいくつかのパラメータ値の小さな変更が、展開中のシステム解で質的な変化を発生させる場合があり、これは分岐とも呼ばれます。たとえば、展開中の微分方程式のシステム解は、周期的状態とカオス的状態のあいだで切り替わる性質を持っています。

画像や分岐図が、こうした解の性質を分岐パラメータによる値の変化に合わせて描き出します。

```
> with(IterativeMaps):
  with(ImageTools):
```

ガウス写像はそのようなシステムの一例です。その分岐図は、以下の単一の実数変数の反復写像を使用して計算できます。

$$xnew = r + e^{k(x-1)^2}$$

この写像は、ときに「マウスマップ」とも呼ばれます。パラメータ値によっては、分岐図がマウスのように見えるためです。

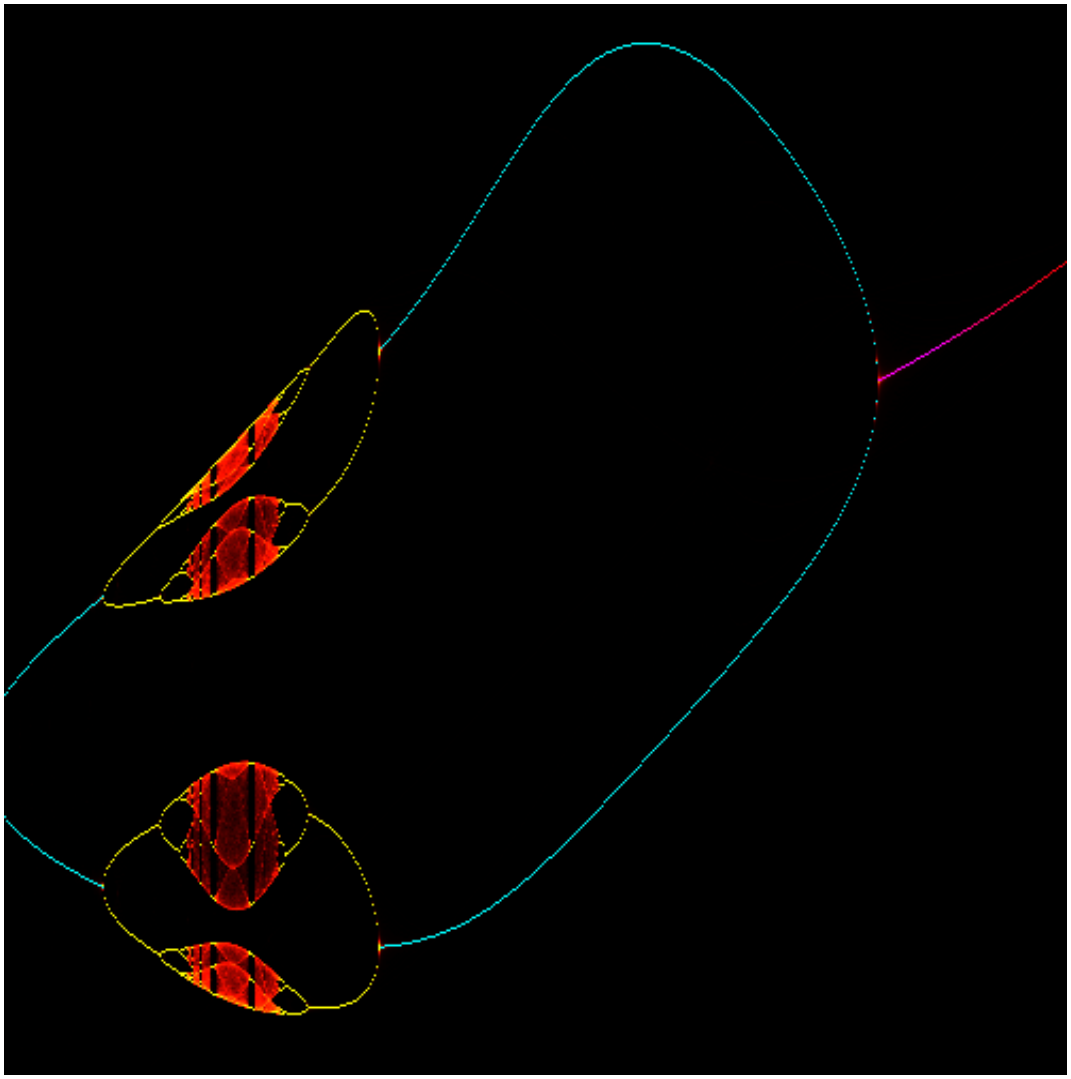
以下に生成された画像では、横方向が r に対応し、縦方向は x に対応しています。

```
> GaussMapImage := Bifurcation( [x], [r + e^(-4.8 (x-1)^2)], [0.5],
                                0.2, 1.7, xmin = 0.7, xmax = 2.1,
                                iterations = 10000 ):
```

```
> ArrayTools:-Dimensions( GaussMapImage )
```

$$[1..500, 1..500, 1..3] \quad \textbf{(1.1}$$

```
> ColouringProcedures:-HueToRGB(GaussMapImage):
  Embed(GaussMapImage)
```



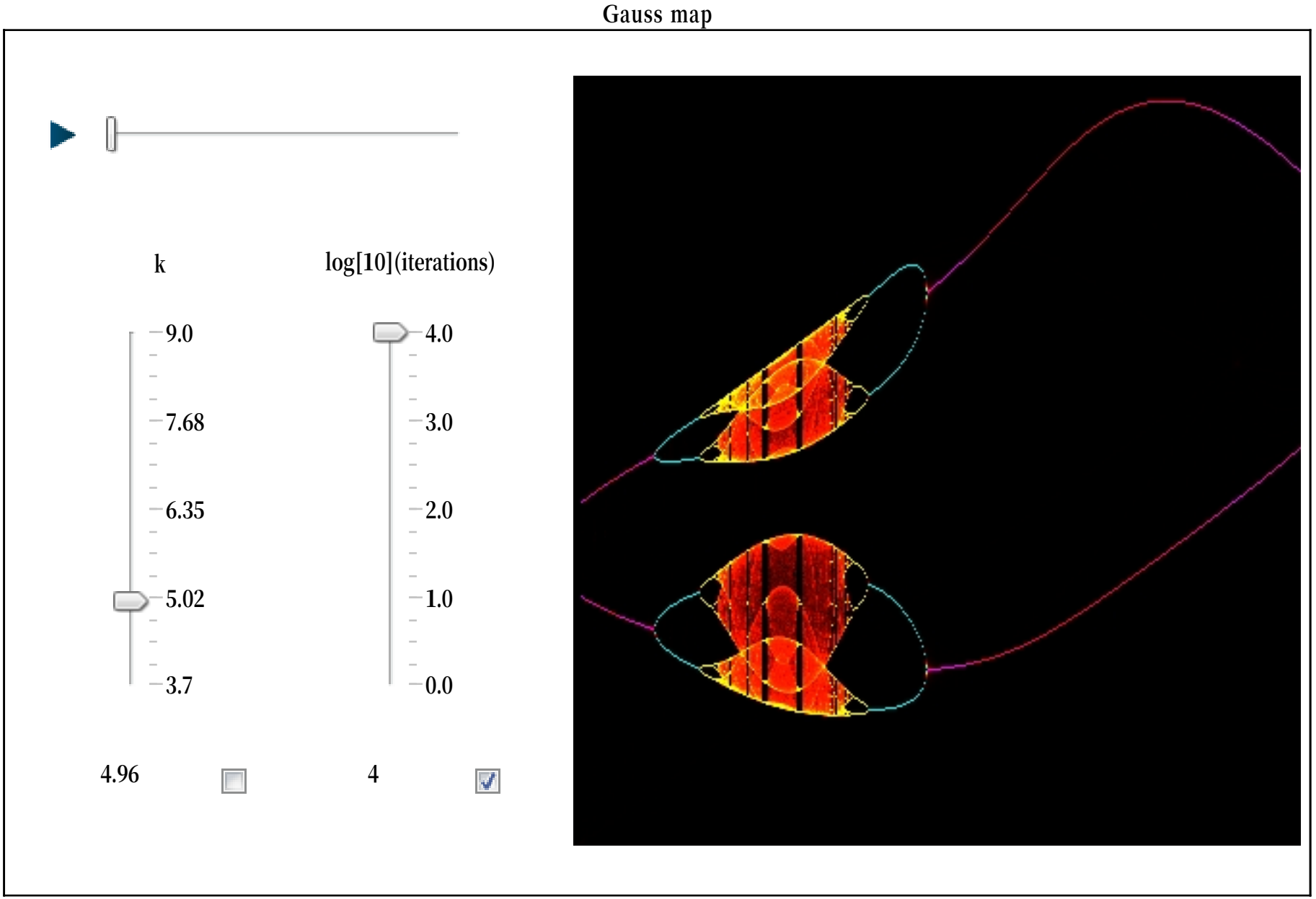
Explore コマンドは、分岐図を操作するためのアプリケーションを取得する場合に使用します。
これにより、反復数またはパラメータ値の変化に伴う反復プロセスの挙動を調査することが容易になります。

以下の appliable モジュールは、Bifurcation コマンドのインプレース計算機能の利点を活用するため、ローカルメンバーの img を使用します。以下の GaussMap のためのプロシージャは、パラメータ k および写像の反復数を指定する引数に基づいて画像を生成します。

```
GaussMap := module() local ModuleApply, img;
    ModuleApply := proc(k, logiter)
        if not type(img, Array) then
            img := Array(1..400, 1..400, 1..3,
                         datatype=float[8], order=C_order);
        else
            ArrayTools:-Fill( 0.0, img );
        end if;
        IterativeMaps:-Bifurcation( height = 400, width = 400,
                                    [x], [r+exp(-k*(x-1)^2)], [0.5],
                                    0.2, 1.0+0.5*(8.0-k)/(8.0-3.9),
                                    xmin = 0.5, xmax = 2.1,
                                    outputimage = img,
                                    iterations = 1+ceil(10^logiter) );
        ImageTools:-ColouringProcedures:-HueToRGB(img);
        return img;
    end proc:
end module:

```

```
> Explore(GaussMap(k, logiter),
                parameters = [[k = 3.7..9.0, animate = false], [logiter = 0.0..4.0, label = `log[10](iterations)`]],
                initialvalues = [k = 4.96, logiter = 4.0],
                placement = left, orientation = vertical,
                animate, title = "Gauss map")
```



アトラクタマップ

Attractor コマンドは、反復または展開する連立方程式の解の 2 次元画像を生成します。このコマンドは、システムの解が、ある特別な状態に達するのに十分近い状態となり、そして、その状態を維持するという特性を持つことからこの名前が付けられています。

たとえば、そのような特別な状態には特定の点、曲線、または面に近い状態などがあり、そのような状態を「アトラクタ」と呼びます。

レスラーアトラクタが、以下の一連の微分方程式の処理中に現れます。

$$\frac{d}{dt}x(t) = -y(t) - z(t)$$

$$\frac{d}{dt}y(t) = x(t) + a\,y(t)$$

$$\frac{d}{dt}z(t) = b + z(t)\,(x(t) - c)$$

このシステムの図は、固定ステップサイズ d t により効果的に解を近似する反復写像を使用して計算します。

一時的な変数 t x が変数のリストに追加され、変数値 x、y、および z が同時に反復します。

```
> dt := 0.0015 :
> xnew := x + dt ( -y - z) :
> ynew := y + dt (tx + a y) :
> znew := z + dt (b + z (tx - c)) :
```

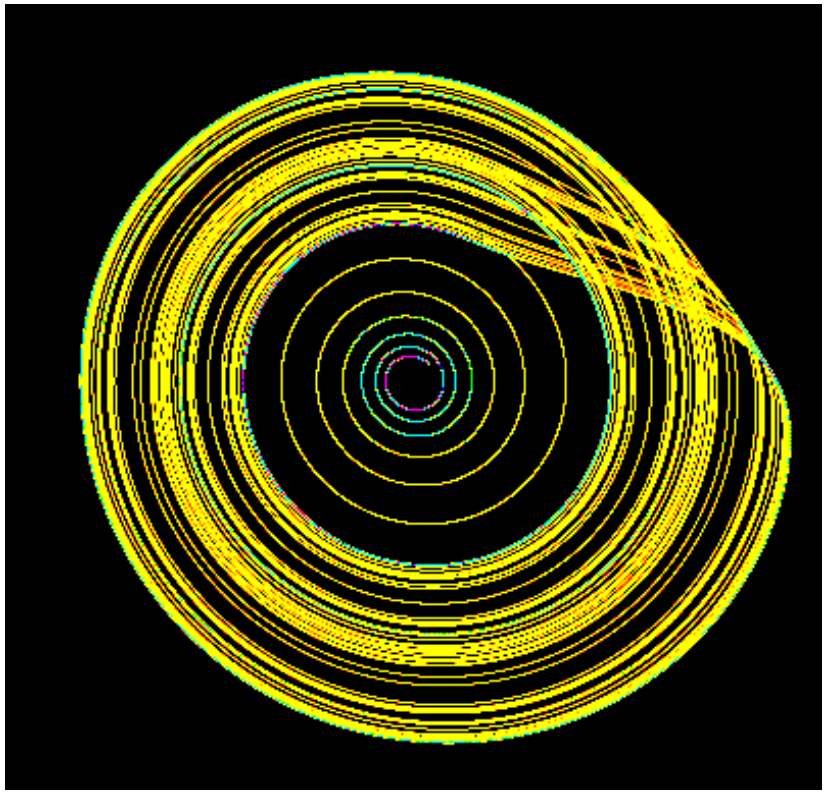
この反復写像の挙動は、パラメータ値 a、b、および c に依存しています。以下の値を使用すると、計算された軌道は固定アトラクタの周囲でカオス的な挙動を示します。パラメータ c が他の値を持つ場合、展開する解は周期性を持つようになります。

```
> a, b, c := 0.1, 0.1, 14.0 :
> RosslerImage, xrng, yrng := Attractor(height = 400, width = 400,
                                        [x, y, z, tx],
                                        [xnew, ynew, znew, x],
                                        [1, 1, 1, 1],
                                        xmin = -24, xmax = 24, ymin = -24, ymax = 22, fixview,
                                        iterations = 300000) :
> ArrayTools:-Dimensions( RosslerImage )
```

[1 ..400, 1 ..400, 1 ..3]

(2.1

```
> ColouringProcedures:-HueToRGB(RosslerImage):
  ImageTools:-Embed(RosslerImage):
```



上記の画像には、x および y の変数の空間が表示されています。斜め方向の表示は、2 つの付加的な変数と投影法を使用して取得できます。ここでは、最初の軸周りの回転を使用して投影が実行されています。固定表示幅を使用しています。

計算速度は、反復写像を適用する Maple コンパイラの内部使用によります。

```
> ang := -Pi/8:
  RosslerImage, xrng, yrng := CodeTools:-Usage(Attractor(height = 400, width = 400,
                                                         [xview, yview, x, y, z, tx],
```

$$\left[x,\ (28-y)\ \sin(ang)+z\sin\left(\frac{\pi}{2}-ang\right),\ xnew,\ ynew,\ znew,\ x\right],$$

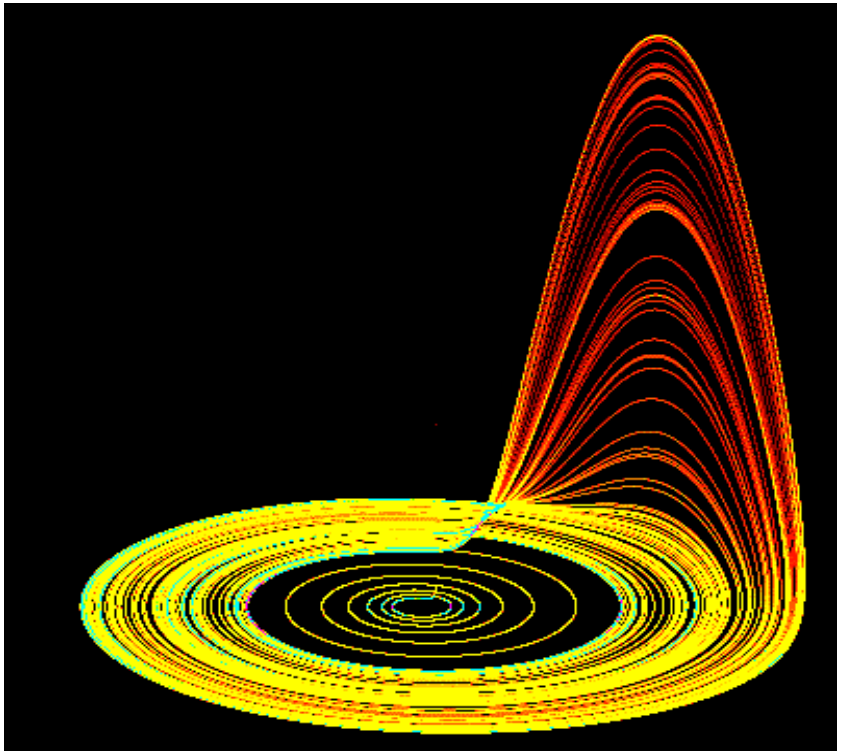
$$[1,\ 1,\ 1,\ 1,\ 1,\ 1],$$

$$xmin=-24,\ xmax=24,\ ymin=-20,\ ymax=28,\ fixview,$$

$$iterations=500000\Big)\Big):$$

memory used=3.69MiB, alloc change=3.66MiB, cpu time=30.00ms, real time=35.00ms, gc time=0ns

> *ColouringProcedures*:-*HueToRGB*(*RosslerImage*):
ImageTools:-*Embed*(*RosslerImage*):



Explore コマンドは、アトラクタ図を操作するためのアプリケーションを取得する場合に使用します。
これにより、反復数やパラメータを変化させたときの解の挙動を調査することが容易になります。傾斜角も操作可能になります。

以下の appliable モジュールは、Attractor コマンドのインプレース計算機能の利点を活用するため、ローカルメンバーの img を使用します。レスラーと呼ばれる以下のプロシージャは、パラメータ c、写像の反復数、および投影表示の傾斜角を指定する引数に基づいて画像を生成します。

```
Rossler := module() local ModuleApply, img;
    ModuleApply := proc(C, logiter, ang)
        local znew;
        znew := z + dt*(b + z*(tx - C));
        if not type(img, Array) then
            img := Array(1..400, 1..400, 1..3,
                         datatype=float[8], order=C_order);
        else
            ArrayTools:-Fill( 0.0, img );
        end if;
        IterativeMaps:-Attractor( height=400, width=400,
                                  [xview, yview, x, y, z, tx],
                                  [x, 1/2*((24-y)*sin(ang)+z*sin(Pi/2-ang)),
                                   xnew, ynew, znew, x],
                                  [1, 1, 1, 1, 1, 1],
                                  xmin=-24, xmax=24, ymin=-24, ymax=24, #fixview,
                                  outputimage = img,
                                  iterations = 1+ceil(10^logiter) );
        ImageTools:-ColouringProcedures:-HueToRGB(img);
        return img;
    end proc:
end module:

```

レスラープロシージャを Explore コマンドに引き渡すことで、引数の変化に対する便利なインターフェースを提供できます。

```
> Explore( Rossler(C, logiter, angle),
      parameters = [ [C = 4.0 ..14.0, animate, label = 'c', orientation = horizontal], [logiter = 4.0 ..6.7, label = `log[10](iterations)`], [angle = -π/2 ..0] ],
      initialvalues = [ logiter = 5.1, angle = -π/6 ],
      placement = left, orientation = vertical,
      animate = false, title = "Rossler attractor" )
```

Rossler attractor

log[10](iterations)

6.7
6.02
5.35
4.68
4.0

angle

0.0
-0.39
-0.79
-1.18
-1.57

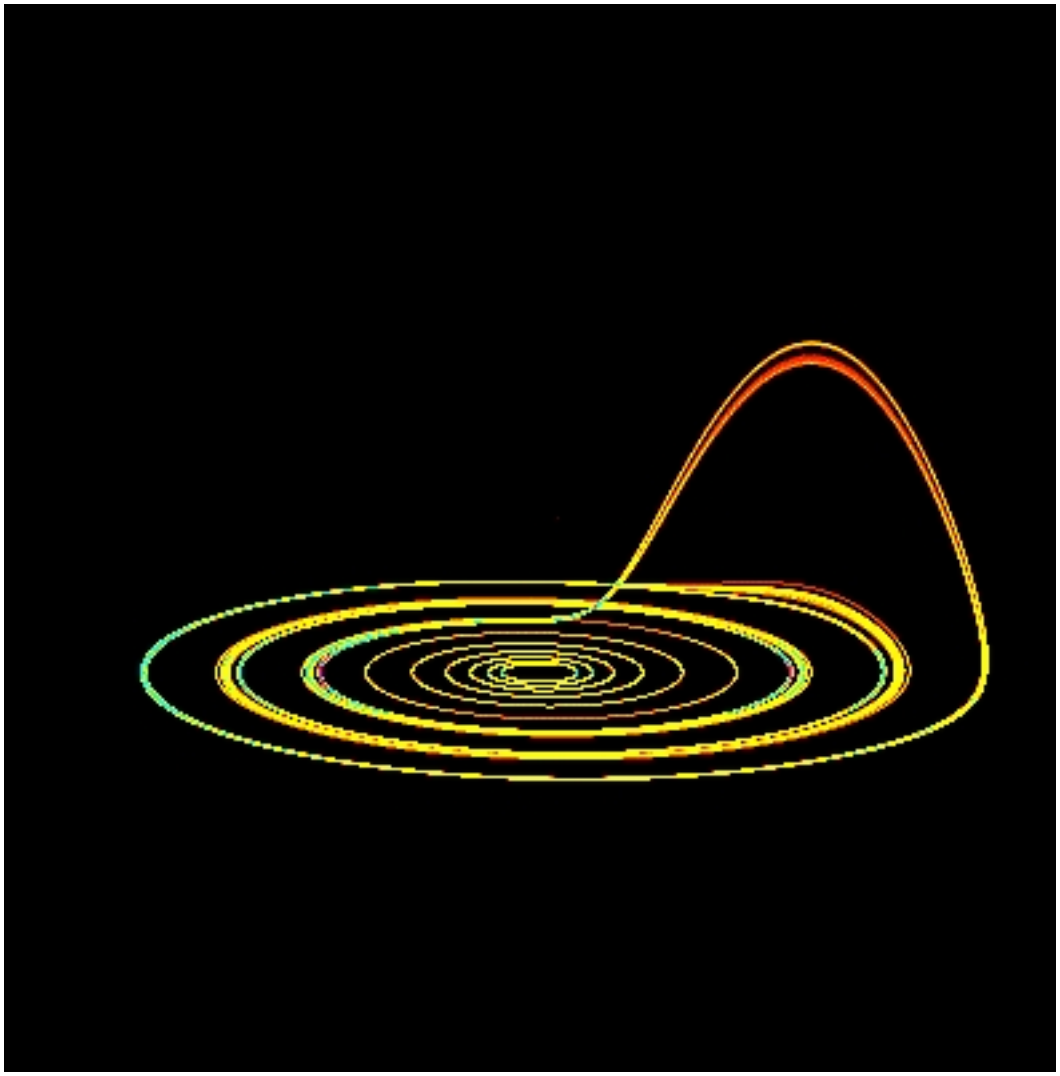
5.45

-0.524

c

4.0 6.5 9.0 11.5 14.0

12.7



エスケープマップ

Escape コマンドにより、複数の変数を含む反復写像からの画像を生成します。

生成される画像の次元は、写像で指定された最初の 2 つの変数の初期値に対応します。反復計算はいくつかの変数を含む場合があり、前回の値や (オプションで) 任意の付加的な変数を使用して 2 次元画像の値を更新することにより実行されます。

反復は、設定された反復の限界を超過するか、設定されたエスケープ条件が満たされるまで継続します。

こうした反復写像のよく知られた適用の 1 つは、フラクタルな性質を持つ領域における、境界線の近似を表す画像の生成です。

```
> expr := (zr + zi I)^3
```

$$expr := (zr + \mathrm{I}\,zi)^3 \tag{3.1}$$

```
> fzi := evalc(ℑ(expr))
```

$$fzi := -zi^3 + 3\,zi\,zr^2 \tag{3.2}$$

```
> fzr := evalc(ℜ(subs(zi = zit, expr)))
```

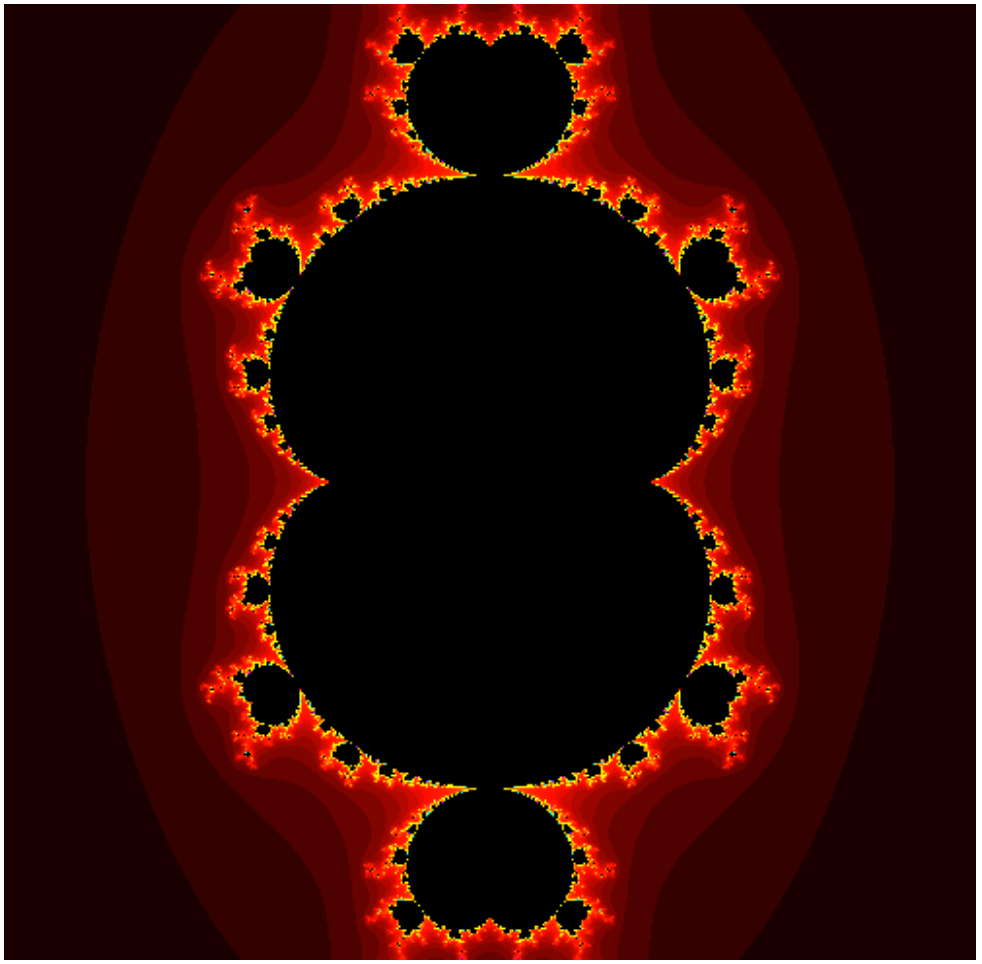
$$fzr := -3\,zit^2\,zr + zr^3 \tag{3.3}$$

```
> mandelbroid := Escape( [zi, zr, zit, zrsqr, zisqr],
                         [fzi + y, fzr + x, zi, zr^2, zi^2],
                         [y, x, y, x^2, y^2],
                         4 < zrsqr + zisqr,
                         -1.2, 1.2, -1.2, 1.2 ) :
```

```
> ArrayTools:-Dimensions( mandelbroid )
```

$$[1..500, 1..500, 1..3] \tag{3.4}$$

```
> ColouringProcedures:-HueToRGB(mandelbroid) :
  Embed(mandelbroid)
```



▶ Pages That Link to This Page