Chapter 10. グラフィックスと計算（その１）

１０－１．グラフィックスにおける曲線の表し方

１０－１－１．直線や曲線の表し方

直線や曲線の表し方は、今まで１つしか習ったことがないかも知れません。たとえば、y = ax + bみたいな形多項式は理解できるでしょう。しかし、これは、直線や曲線の表し方のの仕方の１つでしかありません。ここでグラフィックスを始めるにあたり、直線や曲線の表し方には、いくつかの方法があるということを紹介します。ここでは３種類の表し方の形式を紹介します。というのは、グラフィックスでは、これらの形式のいずれかを使って直線や曲線を描画していくことが多いからです。

★陽関数形式
陽関数形式（Explicit Function Form）での標記の仕方は、２次元の場合は、ｘかｙのどちらかが、もう一方の他方の座標値から計算されるという標記を取ります。これは、今まで習ってきた直線や曲線の表し方です。一般的には、関数fを用いて以下のように表わします。

$$y = f(x)$$　　ここでf(x)は、xを用いた何らかの式を示す関数

たとえば、y = ax + b による直線以外に、２次方程式で表わされる $y = ax^2 + bx + c$なども含まれます。

★陰関数形式
陰関数形式（Implicit Function Form）では、ｘとｙの両方の座標値を用いて、それらの関係を表わす式で表現されます。一般的には、関数gを用いて次のように表わされます

$$f(x, y) = 0$$　　ここでf(x, y) は、xとyを用いた何らかの式を示す関数

たとえば、円の方程式を考えてみますと、円の半径をrと置きますと、$r^2 = x^2 + y^2$ の関係があります。これは、ピタゴラスの定理とも呼ばれています。さて、この関係を使って、陰関数形式で、円を記述しますと以下のようになります。

$$x^2 + y^2 - r^2 = 0$$

★パラメータ形式
パラメータ形式（ParametricForm）は、ｘの座標とｙの座標を、まったく別の変数ｔなどを使って表わすものです。ここで使われる変数ｔは、一般にはパラメータあるいは媒介変数と呼ばれます。多くの場合、ｔの取りうる値は、０から１までで、ある曲線などを表わすときに、ｔ＝０の場合は、一方の端の点の座標を求めることができますし、ｔ＝１の場合は、もう一方の端の点を求めることができます。それぞれの座標は、変数ｔに対する関数として次のように表現されます。

$$x = f(t)$$　　ここでf(ｔ)は、ｔを用いてx座標値を求める式を示す関数
$$y = g(t)$$　　ここでg(ｔ) は、tを用いてy座標値を求める式を示す関数

たとえば、先ほどの円の多項式ですが、媒介変数tを用いると次のように表現することができます。ただし、以下の式で、変数tの取り得る範囲は、0 <= t && t <= 2 * π になります。すこしJava風な記述にしました。

$$x = r * \cos(t)$$
$$y = r * \sin(t)$$

基本的な直線や２次式の曲線（放物線と呼ばれます）、円などについて、陽関数・陰関数・パラメータ形式で表わした場合、どのように記述されるか示してみましょう。

	陽関数形式	陰関数形式	パラメータ形式
直線	$y = ax + b$	$y - ax - b = 0$	$x = t$ $y = at + c$
放物線	$y = ax^2 + bx + c$	$y - ax^2 - bx - c = 0$	$x = t$ $y = at^2 + bt + c$
円	$y = \sqrt{}(r^2 - x^2)$	$x^2 + y^2 - r^2 = 0$	$x = r * \cos(t)$ $y = r * \sin(t)$

なお、上記の放物線は、２次曲線の限られた記述になっています。一般的な２次曲線（Quadric curve）の陰関数形式での標記は、次のようになります。円についても、この式で表わすことができることに注意してください。

$$ax^2 + by^2 + cxy + dx + ey + f = 0$$

このような標記で表わされる２次曲線は、円錐曲線（Conics）とも呼ばれますが、２次曲線は、楕円（Ellipse）・放物線（Parabola）・双曲線（Hyperbola）のいずれかになります。上記に表わした正円は、楕円の一部として考えられます。これらに関するよく用いられる記述を以下に記しておきます。これは、覚える必要はありませんが、３つの種類があることだけ覚えておいて下さい。

楕円　陰関数形式　$x^2 / a^2 + y^2 / b^2 = 1$　ただし、a>0 && b>0
　　パラメータ形式　$x = a \cos \theta$, $y = b \sin \theta$

放物線　陽関数形式　$y^2 = 4ax$　これは、$x = 1/ 4a * y^2$

双曲線　陰関数形式　$x^2 / a^2 + y^2 / b^2 = 1$　ただし、a>0 && b>0
　　パラメータ形式　$x = a \cosh t$, $y = b \sinh t$　ただし、$\cosh t = (e^t + e^{-t}) / 2$
　　$\sinh t = (e^t - e^{-t}) / 2$

１０－１－２．グラフィックスの位置を整える

直線や２次曲線など、多項式で表わされる線を表示しようと思うとき、Java特有の座標に考慮しなければいけません。ウィンドウの左上が、(0,0)の原点になっており、右下に向けて、x座標もy座標も大きくなっていることです。このため、ウィンドウの中央に原点を移動するために、座標変換を行なわなければなりません。この原点移動を、式として記述してみましょう。ウィンドウ上の座標をwx,wyとし、原点の座標の移動分をdx,dyとおきますと、本来のx,yから次のように移動することになります。下記の式で-yとしたのは、y座標の方向が逆になっているからです。

▼原点の移動

$$wx = x + dx$$
$$wy = -y + dy$$

次に、ウィンドウのサイズが決まっていますので、そのサイズに合わせて、直線や曲線を拡大・縮小させなければなりません。この拡大・縮小率をx座標方向をa、y座標方向をbとしますと、最終的に以下の式が得られます。

▼原点の移動、および拡大・縮小率を考慮

$$wx = a * x + dx$$
$$wy = - b * y + dy$$

１０－２．グラフを描くアプレット

１０－２－１．垂直線で２次多項式の曲線を描く

放物線で一番簡単な、$y = x^2$ という懐かしい響きのするグラフを書いてみましょう。まずは、垂直線を並べて、その輪郭として描くことを考えてみます。アプレットの幅を300ピクセルと高さを200ピクセルと置き、原点のウィンドウ上の座標を(150,150)と決めてみます。さきほどの原点の移動の式を使うと以下のようになります。

$$wx = x + 150;\quad wy = -y + 150;$$

垂直線なので、１番目と３番目のパラメータが同じ値になります。ここでは、ウィンドウ上のy座標が0から引かれる垂直線を考えてみます。陽関数形式として、xを移動させる形にしてみます。なお、１つずつ移動させると、べったりとした描画になってしまうので、垂直線であるとわからせるために、xを４つずつの間隔で移動させてみます。 x の移動範囲は、-100から100まで移動させます。変数y, wx, wyは、予め宣言しておくようにしました。

```
public void paint (Graphics g ) {
        int  y, wx, wy ;
        for  (  int  x=-100;  x <=100;  x += 4 ) {
                y = x * x ;
                wx = x + 150;
                wy = -y + 150;
                g.drawLine( wx,  0,  wx,  wy);
        }
}
```

二乗するとあまりにも数が大きくなるので、すぐ表示領域からy座標がはみ出してしまいます。あまり輪郭線が２次方程式の放物線っぽくありません。

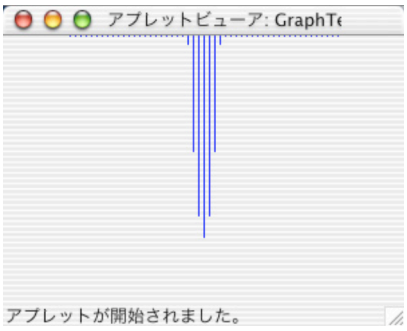


図10-1　すぐに、はみ出た垂直線

そこで、y方向の圧縮率として二乗した数を50ぐらいで割って、横方向の大きさを圧縮してみましょう。wyの計算の部分を以下のようにしてみました。

$$wy = -y / 50 + 150;$$

この場合は、輪郭がだいぶ昔教科書で見たような形になってきました。しかし、少しはみ出ている部分があります。これをなんとかしたいものです。

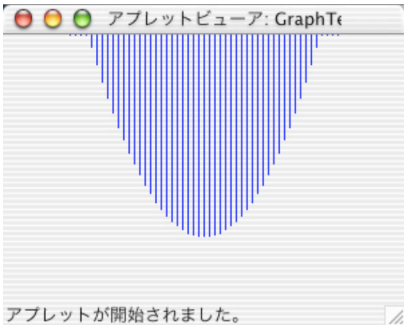


図10-2　ま〜だ、はみ出している垂直線

先ほどの例では、x座標が150になる前に輪郭線の曲線の部分が終わってしまっています。これを避けるために、-150や+150の場合についてもう少し精密に考えてみましょう。xの値が100のときは、圧縮を掛けないとなると、$wy = -x*x+150 = -100*100+150 = -10000+150 = -9850$となって、途方もなくアプレットの描画範囲からはみ出してしまいます。縦方向の圧縮率を計算するときは、-100や+100のときに２乗した結果がちょうど150になるようにすれば、$wy = -150+150 = 0$ となって描画範囲からはみ出ません。そのための圧縮率は、150/10000すなわち、0.015ということになります。よって、最終的な２次関数は、$wy=150-x^2*0.015$ということになります。これを終点のy座標として指定してみます。今回は、途中の変数を全部省略して、いきなりdrawLineのパラメータの部分に記述してみました。

```
public void paint (Graphics g ) {
        for  (  int  x=-100;  x <=100;  x += 4 ) {
                g.drawLine( x + 150,  0,  x + 150,   150  -  (int)(x * x * 0.015) );
        }
}
```

整数に戻すために、型変換をしていることに注意してください。これによって描かれる図は、次のような感じになります。

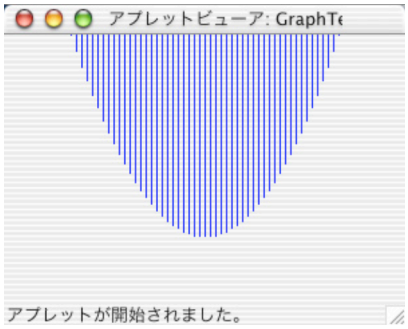


図10-3　２次曲線の輪郭線を持つ垂直線

drawLineメソッドを使って、繰返しを用いて図形を描くにはどのようにすれば良いかを考えてきました。このような簡単なメソッドでも、与えるパラメータの式を工夫することによって、いろいろな図形を描くことができるのです。アプレットの描画領域に収めるためには、圧縮をしたり、一定の数を足したりして、パラメータの式をうまく設計しないといけません。圧縮率や最大値などは、どの範囲を描きたいかによって自分で工夫してみてください。ちなみに、縦線ではなくて、始点をまったく１つの点にしてしまえば（すなわち、g.drawLine(50, 0, x + 150, 150 - (int)(x * x *0.015));にすれば）、こんな渋い図形も描けます。

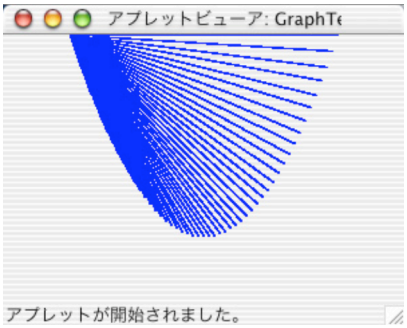


図10-4　始点を一点にした場合

１０－２－２．折れ線で近似して描く

垂直線の輪郭線として曲線を描いてきましたが、昔の数学の教科書のように、曲線そのものを描画するようにしたいものです。線は、点の動きとして表わされますから、ｘを変えながら、ｙ座標の位置を求め、座標の決まった各点を描いていけばいいように思えます。ところが、Javaでは点を描くというメソッドがありません。また一般的にも点で描くということは行なわれていません。一般的は、線で曲線を描くことが行なわれています。つまり、折れ線で曲線を近似しようということになります。

線を描くにはdrawLineメソッドを使うのですが、始点と終点の２つの点の座標が必要でした。２つの点をどのように決めましょうか？１つの点は上の計算で求めるとして、もう１つの点は前に計算した点の座標を覚えておくことにします。１つ前の点から、現在の点まで線を引くようにします。そこで座標を計算して線を描いた後に、

```
lastx = x;
lasty = y;
```

として、常に一つ前の点の座標を別の変数に覚えておけば、そこから、現在までの点まで線を引くことができるでしょう。

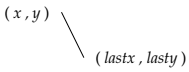


図10-5　前に求めた点から今回求めた点まで線を引く

パラメータ形式で表わされる曲線のことを、パラメトリック曲線（Parametric Curve）と呼びますが、この曲線を、以上のような形でｎ本の折れ線で近似するためには、つぎのプログラムの断片で描くことになります。ここで、ｘ＝ｆ(ｔ)やｙ=ｇ(ｔ)とは、ｔを使った何らかの式で、ｘやｙが描かれるということを示しています。

▼パラメトリック曲線の描画方法

```
double delta = 1.0 / n;
double       t = 0.0;
int    lastx = f( 0 ), lasty = g( 0 );
for ( int  i=1; i <= n;  i++ ) {
        t += delta;
        int  x = f( t ),   y = g( t );
        g.drawLine( lastx, lasty, x, y );
        lastx = x;
        lasty = y;
}
```

この方法で、先ほどの放物線を折れ線で表示してみましょう。ここでは、ウィンドウの座標への変換も考慮ｓると、関数ｆとｇは、次のように記述できるのではないでしょうか。

```
f( t ) =  t * 200  - 100  + 150          // xの変化範囲が0を中心とする-100から100
                                         // の200の範囲なので、+150は座標変換用
```

g(t) = - (t*200 -100) * (t*200-100) * 0.015 + 150

実際のアプレットのプログラムとして、記述してみましょう。たとえば、折れ線の数であるnを100とします。型変換も含めると次のように記述できます。paintメソッドの部分だけを記述しました。

```
public void paint( Graphic g ) {
        int     n = 100;
        double  delta = 1.0 / n;
        double  t = 0.0;
        int     lastx =  50;                    // 0 + 200 - 100 + 150の結果
        int     lasty = 0;                      // - ( 0*200-100)*(0*200-100)*0.015+150の結果
        for ( int i=1; i <= n; i++ ) {
                t += delta;
                int     x = (int)( t * 200) - 100 + 150;
                int     y = -(int)((t*200-100) *(t*200-100) *0.015)+ 150;
                g.drawLine( lastx, lasty, x, y );
                lastx = x;
                lasty = y;
        }
}
```

実行結果を以下に示します。この実行結果では、ｘ座標、ｙ座標を青色で、放物線を赤色で表示するようにしました。

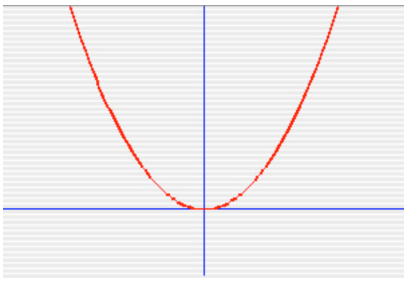
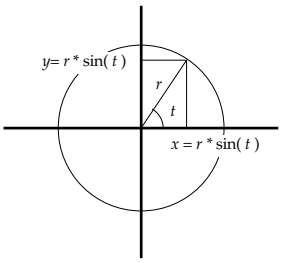


図10-6 放物線を表わしたアプレット

１０－２－３．円・楕円の描画

同じようにパラメトリック曲線の描画手法を用いて、三角関数のcosとsinを使ってx、y座標を計算していけば、円が書けるはずです。以下の式でrは半径、tは中心の角度とします。

$x = r * \cos(t)$
$y = r * \sin(t)$



↑誤植：$x = r * \cos(t)$の間違い

図12-1　円と三角関数

ところが、Mathクラスでは面倒なことに、角度はradian形式という角度体系を用いています。0から360度というのはdegree形式なので、それをradian形式に変換する必要があります。代表的な角度を以下に書き記してみましょう。

degree		radian	degree		radian
0	→	0	180	→	π
45	→	π / 4	270	→	3π / 2
90	→	π / 2	360	→	2π

こんな感じの対応を思い出して戴けましたか？公式を使って変換すると、実数のdouble型を用いて、Javaでは次のように式を表すことができます。たとえば、下の例では35度を変換して求めようとします。

```
int   degree  = 35;
double radian =  degree / 180.0 * Math.PI;                    //   = Math.toRadians(  degree );
```

なお上記の注のように、この変換をするための関数として、Math.toRadians(　)と、Math.toDegrees( )がJava2から用意されています。ここまでわかれば、あとはxおよびy座標を、radian角度と三角関数で求めることはできます。いま、半径80ドットぐらいを目安に考えましょう。しかしながら、アプレットのｘ座標ｙ座標は、0からしか始まっていませんし、しかもｙ座標は逆になっています。そこで、中心を（100,100）の座標に置くように考えて、両座標にいつでも100を加えてやります。という訳で、これらを考慮すると、ウィンドウ上のx座標、ｙ座標は次のような形で求めることになります。

```
x =  cos( radian ) * 80 + 100;
y = -sin( radian ) * 80 + 100;
```

なお、cosとsinは、Mathクラスのメソッドですし、最終的には実数で計算していたものを整数に戻してやらなければならないので、Java上では、次のように書くことになります。

```
x = (int) (Math.cos( radian ) * 80 + 100);
y = (int) (-Math.sin( radian ) * 80 + 100);
```

パラメトリック曲線を描く手法で描いていきます。ただし、今回は、パラメータ変数の変化範囲は、0から360になります。最初は、上の式にradian=0を代入しますと(180,100)になりますので、最初はそこから始めましょう。

```
lastx= 180;                 //  100ずらしてあります。すなわち80+100
lasty= 100;                 //   100ずらしてあります。すなわち0+100
```

という訳で、繰返しを使って角度は、0度から360度まで動かしてみて、線を描くようなアプレットを作ってみましょう。

```
import  java.awt.*;
import java.applet.*;
public class MyCircle extends Applet {
        public void paint( Graphics gc ) {
                double radian;
                int  x, y, lastx, lasty;

                g.drawLine( 0, 100, 200, 100 );      //  横軸
                g.drawLine( 100, 0, 100, 200 );      //  縦軸
                lastx= 180;
                lasty= 100;
                for (int i=1; i<=360; i++ ) {
                        radian = Math.toRadians(  i  );
                        x = (int) (Math.cos( radian ) * 80 )+ 100;
                        y = (int) (-Math.sin( radian ) * 80) + 100;
                        g.drawLine( lastx, lasty, x, y );
                        lastx = x;
                        lasty = y;
                }
        }
}
```

１０－２－４．リサージュ図形を描く

調子に乗ってきたところで、オシロスコープなんてものがありまして、あれって一体何ものかよくわからないけど、いろいろ測定する機械らしくて、いろんな図形がでるらしいのですが、理系以外の人にはあまり興味ないものかも知れません。そういうオシロスコープで、面白いのがリサージュ図形なのですが、これも理系以外の人にはあまりお馴染でありません。

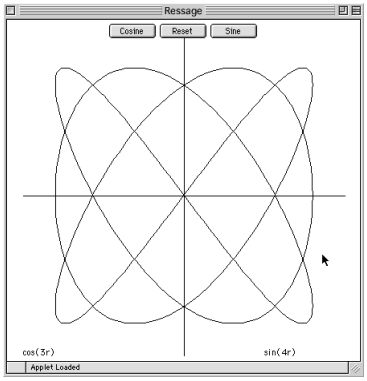


図10-7　リサージュ図形

この図形の特徴は、円を描くときの三角関数の角度の進み具合を変えてやればすぐ描けることです。円だったらcosで0から360度まで、sinでも0から360度まで行儀良く変化させていきましたが、cosで0から720度まで変えてしまったらどうなるでしょう？えっ、cosだけ２回転もさせちゃったら、まずいでしょう？まあ、やってみましょう。

```
x = (int) (Math.cos( radian*2 ) * 80 + 100);          // 2倍2倍！
```

たとえば、上のようにx座標を求める式で、cos関数の角度を２倍するだけで、変な図形が出てきます。これをリサージュ図形と言うらしいのです。これ繰返しを使って何倍にもできるようにしちゃえ！って感じで次のアプレットを作ってみました。インスタンス変数zoomに倍率をいれておき、どんどん倍率が＋１されていくのが特徴です。４倍まで変わります。倍率が変わるたびに、色を変えるようにしました。

```
import  java.awt.*;
import java.applet.*;

public class Ressage extends Applet {
        public void paint( Graphics g ) {
                double radian;
                int  x, y, lastx, lasty;

                g.drawLine( 0, 100, 200, 100 );   g.drawLine( 100, 0, 100, 200 );
                for (  int          zoom =1; zoom <= 4;  zoom ++ ) {
                        g.setColor(  (zoom == 1 ) ? Color.blue : ( zoom == 2 )  ? Color.red :
                                ( zoom == 3 ) ? Color.orange : Color.green );
                        lastx= 180;   lasty= 100;
                        for (int i=0; i<=360; i++ ) {
                                radian = Math.toRadians( i );
                                x = (int) (Math.cos( radian*zoom ) * 80 + 100);
                                y = (int) (- Math.sin( radian ) * 80 + 100);
                                g.drawLine( lastx, lasty, x, y );
                                lastx = x;   lasty = y;
                        }
                }
        }

}
```

１０－２－５．螺旋（らせん）を描く

螺旋は、円と似ているのですが、円の半径が角度が変化していくについて変化していきます。この変化について、一定の割合で変化していくものと、指数的に変化していくものがあり、ここではその２つについて簡単に描画してみることにしましょう。

★代数螺旋（アルキメデス螺旋：Archimedean Spiral）
蚊取り線香ような螺旋は、アルキメデスの螺旋と呼ばれています。角度θと半径ｒの割合が一定の定数ｃになるような螺旋です。ｃ＝ｒ／θとおきますと、各座標は、次のような形になります。

$x = c * \theta * \cos(\theta)$, $y = c * \theta * \sin(\theta)$

この螺旋をアプレットで描画してみましょう。半径を100としまして、360度で半径の大きさになるようにｃの値を設定します。paintメソッドの部分だけを記述してあります。ｃを求めるのに、一番大きな角度である360を半径で割っておき、それに角度を掛けるようにしてあります。

```
public void paint( Graphics g ) {
        double radian;
        int  x, y, lastx, lasty;

        lastx= 100;   lasty= 100;
        double   c =  100 /  360.0;
        for (int i=0; i<=360; i++ ) {
                radian = Math.toRadians( i );
                x = (int) (Math.cos( radian ) * c *  i   + 100);
                y = (int) (- Math.sin( radian ) * c *  i  + 100);
                g.drawLine( lastx, lasty, x, y );
                lastx = x;   lasty = y;
        }
}
```

★対数螺旋（Logarithimic spiral）
角度と半径が指数の関係にある螺旋で、対数螺旋と呼ばれます。巻き貝の螺旋がこのような螺旋の構造になっています。$r = a^{c\theta}$とおきますと、各座標は次のようになります。

$x = a^{c\theta} \cos \theta$, $y = a^{c\theta} \sin \theta$

このアプレットでは、３回転ぐらいさせてみましょう。上の式から考えると、a=e（自然対数）とおきますと対数を使って、ｃ＝log(ｒ)／θとなります。paintメソッドだけ記述しますと、以下のようになります。なお、θは、ラジアン体系を使っています。

```
public void paint( Graphics g ) {
        double radian;
        int  x, y, lastx, lasty;

        lastx= 100;   lasty= 100;
        double   c =  Math.log(  100 ) /  ( 3 * 2 * Math.PI);
        for (int i=0; i<=360 * 3; i++ ) {
                radian = Math.toRadians(  i );
                x = (int) (Math.cos( radian ) * Math.pow(  Math.E,  c * radian)  + 100);
                y = (int) (- Math.sin( radian ) * Math.pow(  Math.E,  c  *  radian ) + 100);
                g.drawLine( lastx, lasty, x, y );
                lastx = x;   lasty = y;
        }
}
```

１０−３. 課題

課題10-1.
一次関数と二次関数について、自分でアプレットを作成し、drawLineの４つのパラメータをfor文と共に変化させて、効果を確かめなさい。drawRectや、drawOval、あるいはdrawArcなども用いて、パラメータにfor文と共に変化させるような式を記述して、その効果を確かめてみなさい。

課題10-2.
次のような二次関数を用いて、水平線から構成される図を描画するような、アプレットを作りなさい。クラス名は、Leafにしなさい。均等に色分けされた色は次の６色を使いなさい：上から緑(Color.green)、黄(Color.yellow)、橙(Color.orange)、赤(Color.red)、シアン(Color.cyan)、青(Color.blue)。

問題の図

ヒント：葉っぱ全体を包む正方形の一辺のサイズをWとすると、葉っぱの左端は、y^2/Wで、右端はW-(W-y)^2/Wで計算できます。横線であることに注意しなさい。

10-3.
通常のサインカーブ、コサインカーブを描くようなアプレットを作成してみなさい。また、垂直線を間隔を開けて並べて、その輪郭から、サインカーブを描くようなアプレットを作成してみなさい。クラス名は、SineTesterで。

10-4.
リサージュ図形のアプレットを拡張して、Sineの方の倍率も変えることができるように変更しなさい。

10-5.
双曲螺旋（Hyperbolic spiral）は、c = rθの関係があり、x = (c / θ) * cos θで、y = (c / θ) * sin θになります。これを何回転もさせるようなアプレットを作りなさい。クラス名は、HyperbolicSpiralで。

10-6.
リチュースと呼ばれる、代数螺旋は何回転もさせると以下のような美しい螺旋を描きます。$r^2 = a^2 / \theta$です。これを描くアプレットを作成しなさい。クラス名は、Lituus（巻杖のラテン語）で。