2002 年度 卒業研究レポート

2, 3 次元の力学系

伊藤　聡一郎

２００３年2月２８日

目 次

第 1 章　はじめに **2**

第 2 章　流れと微分方程式 **3**

2.1　流れの定義 3

2.2　軌道、周期軌道、不動点 4

2.3　再帰的、自己回帰的 4

2.4　ポアンカレ写像 5

第 3 章　平面上の流れ **6**

3.1　定理 1 6

3.2　定理 2 6

3.3　ω 極限集合 7

3.4　定理 3 7

3.5　定理 4 7

第 4 章　ポアンカレ・ベンディクソンの定理 **8**

第 5 章　プログラムと数値実験 **9**

5.1　例 1 9

5.2　例 2 14

5.3　例 3 19

5.4　ファンデルポール方程式 23

5.5　ダフィング方程式 28

5.6　ローレンツ方程式 33

第1章　はじめに

この卒業研究では、2 , 3 次元の常微分方程式の力学系について研究しました。

2 次元においては、主に高橋 [1] を参考にして , 微分方程式の定める流れの話に始まり、平面上の微分方程式の典型的な振る舞いを述べている、ポアンカレ・ベンディクソンの定理の証明をしました。また、数値実験においては、極限周期軌道 (limit cycle) を持つような、微分方程式について実験しました。

3 次元においては、ローレンツアトラクターとよばれる解軌道が見られるローレンツ方程式の数値実験を研究しました。3 次元の力学系については、ポアンカレ・ベンディクソンの定理を適用することができず、実際、ローレンツ方程式では極限周期解というものは現れません。ローレンツ方程式の数値実験に見られるように、二つの不動点の周りを行ったり来たりする、きわめて複雑な解曲線を得ることができます。解は、初期値により “原理的に” 完全に決定されているにもかかわらず、実際には予測不可能です。ほんの少しだけ違う初期値から出発すると、未来の状態ははなはだしく違ったものになるというのが、カオスの複雑性でもあり、おもしろさであると、実感いたしました。

第2章　流れと微分方程式

2.1　流れの定義

実 n 空間 $\mathbf{R}^2$ 上の連続なベクトル場を $a(x)$ とし、以下では常微分方程式

$$\frac{dx}{dt} = a(x) \tag{2.1}$$

に対して、初期値問題の解の一意性が成立し、かつ爆発が起こらないと仮定する。したがって、$\mathbf{R}^n$ の点 x_0 に対して、初期条件

$$x(0) = x_0 \tag{2.2}$$

をみたす解の時刻 t における値 $x(t)$ を対応させる写像 T_t が、すべての実数 t に対して得られる。ここで、解 $x(t) = T_t x_0$ の一意性を仮定しているので、次のような性質がわかる

$$T_{t+s} = T_t T_s,\ T_0 = Id(\text{ 恒等写像 }). \tag{2.3}$$

実際、

$$T_t(T_s x_0) = T_t x(s) = x(t+s) = T_{t+s} x_0$$

であった。また、(2.3) より、自動的に T_t は可逆な写像となり、逆写像は、

$$T_t^{-1} = T_{-t} \tag{2.4}$$

によって与えられることがわかる。ベクトル場 $a(x)$ の連続性を仮定しているので、径数に関する連続性から、

$$\text{写像 } (t,x) \in \mathbf{R} \times \mathbf{R}^n \mapsto T_t x \in \mathbf{R}^n \text{ は連続} \tag{2.5}$$

である。(2.3) ~ (2.5) をみたす写像の族 $T_t,\ -\infty<t<\infty$ を流れと呼ぶ。

微分方程式 (2.1) の定める流れ T_t に対しては、(2.1) を書き直してみれば、

$$\frac{d}{dx} T_t x = a(T_t x),\ -\infty<t<\infty \tag{2.6}$$

が x をとめるごとに成立している。逆に、x をとめるごとに $T_t x$ が t について微分可能であるような流れ T_t は、あるベクトル場 $a(x)$ に対する微分方程式 (2.1) の定める流れとなる。実際

$$\frac{d}{dt} T_t x \mid_{t=0} = a(x)$$

とおいて，ベクトル場 $a(x)$ を定めれば，性質 (2.3) より

$$\frac{d}{dt} T_t x = \frac{d}{ds} T_{t+s} x \mid_{s=0} = \frac{d}{ds} T_s(T_t x) \mid_{s=0} = a(T_t x)$$

が得られる。したがって、(2.6) が x をとめるごとに成立している。

命題 **1** (径数に関する連続性) D を $\boldsymbol{R}^{n+1}$ の領域、A を $\boldsymbol{R}^m$ の領域とし、$f(t,x,\alpha)$ は、

$$\tilde{D} = \{(t,x,\alpha); (t,x) \in D,\ \alpha \in A\}$$

で定義された $\boldsymbol{R}^n$ 値連続関数とし、次の初期値問題を考える。

$$\frac{dx}{dt} = f(t,x,\alpha),\ x(t_0) = x_0. \tag{2.7}$$

ベクトル場 $f(t,x,\alpha)$ は係数 α と独立なリプシッツ条件

$$\mid f(t,x,\alpha) - f(t,x,\alpha) \mid \leq L \mid x - y \mid \quad (L \text{ は } \alpha \text{ によらない定数})$$

が、$(t,x) \in D,\ (t,y) \in D,\ \alpha \in A$ のとき成立するものとする。このとき *(2.7)* の局所解を $x(t) = \xi(t,\alpha)$ と書けば、$\xi(t,\alpha)$ は存在している限り (t,α) の連続関数である。

(注：この定理は、t が有限の範囲についての主張で、$t \to \infty$ での挙動の連続性まで保証するものではない)

2.2 軌道、周期軌道、不動点

点 $x \in \mathbf{R}^n$ に対して、集合 $\{T_t x;\ -\infty<t<\infty\}$ を x の軌道という。$\tau>0,\ T_\tau x = x$ であれば、点 x は周期 τ の周期点であるとよび、x の軌道は周期 τ の周期軌道であるという。$T_\tau x = x$ となるような τ の最小値を、x の最小周期という。とくに、$T_t x \equiv x\ (t \in \mathbf{R})$ となる点 x は、不動点であるという。

2.3 再帰的、自己回帰的

流れ $T_t,\ -\infty<t<\infty$ に対して、点 x が (正の向きに) 再帰的であるとは、x の任意の近傍 U に対して、$\lim_{n\to\infty} t_n = +\infty$ となる数列 $(t_n)_{n\geq 0}$ がとれて、

$$T_{-t_n}(U) \cap U = \{y \in U;\ T_{t_n} y \in U\} \neq \emptyset$$

が成立することをいう。
例 1 ．不動点 x は再帰的である。$U \ni x$ ならば、つねに $T_{-t}U \cap U \ni x$ であるから。
例 2 ．周期点 x は再帰的である。周期を τ とすれば、$U \ni x$ のとき、

$$T_{n\tau}(U) \cap U \neq \emptyset \quad (n \in \mathbf{Z}).$$

例 3 ．点 x に対して、$\lim_{n\to\infty} T_n$ となる数列 $(t_n)_{n\geq 0}$ がとれて、

$$\lim_{n\to\infty} T_{t_n} x = x$$

が成立するとき、正の向きに自己回帰的であるとよぶことにする。不動点や周期点は自己回帰的な点の例である。一般に自己回帰的な点は再帰的である。実際、x が自己回帰的ならば、x の任

意の近傍Uに対して、あるn_0以上の任意のnについて$T_{t_n}x \in U$、したがって、$T_{-t_n}(U) \cap U \neq \emptyset$である。

例４．$\mathbf{R}^n$の有界集合K上の流れが体積を保つならば、集合Kのすべての点は再帰的である（ポアンカレの再帰性定理）。

証明）Kの点xが再帰的でないと仮定しよう。すると、xの近傍Uと正数τがとれて、$t \geq \tau$のとき、$T_{-t}(U) \cap U = \emptyset$でなければならない。とくに、$T_{-n\tau}(U) \cap U = \emptyset$ $(n = 1, 2, \cdots)$. さらに、相異なる整数n, mに対して$T_{n\tau}(U) \cap T_{m\tau}(U) = \emptyset$となる。実際、$n>m$とすれば、

$$T_{n\tau}(U) \cap T_{m\tau}(U) = T_{n\tau}(U \cap T_{(-n-m)\tau}(U)) \neq \emptyset.$$

したがって、

$$U_0 = U,\ U_1 = T_\tau(U),\ \cdots,\ U_n = T_{n\tau}(U),\ \cdots$$

は互いに交わらないKの部分集合であり、T_τは体積を保つので、U_nの体積はすべてU_0の体積に等しいことになる。ところで、U_0の体積は正であるから、kの体積は無限大でなければならない。これは、Kがコンパクトであったことに矛盾。ゆえに非再帰点は存在しない。

2.4　ポアンカレ写像

連続なベクトル場$a(x)$から定まる、$\mathbf{R}^n$の流れ$T_t,\ -\infty<t<\infty$を考える。超局面Sが、流れT_tの切断面であるとは、S上の各点において、ベクトル場$a(x)$とSが横断的に交わることをいう。

点xを通る切断Sがあるとき、もし、$t>0$で、$T_t x$がS上に戻ってくるならば、そのような最小のtを帰還時間と呼び、写像$x \longmapsto T_t x$を、帰還写像または、ポアンカレ写像と呼ぶ。

点xを通る切断Sがあり、$T_t x \in S$となる正数tが存在するとき、点xはポアンカレ切断を持つという。もちろん、点xが不動点ならば、ベクトル$a(x)$は0であり、ポアンカレ切断は持たない。

第3章　平面上の流れ

3.1　定理 1

定理 1 平面上の微分方程式の定める流れに対して、ポアンカレ切断を持つような再帰的な点は周期点である。

証明) 再帰的な点 x がポアンカレ切断 S を持つとする。平面上で考えているので、S は曲線弧である。t を点 x の S への帰還時間として、$y = T_t x$ とする。軌道の一部 $T_s x, \quad 0 \leq s \leq t$ に、S の弧 $\widehat{yx}$ をつなげて、得られる閉曲線 γ を考えよう。γ は閉曲線であるから、平面を 2 つの領域、内部と外部に分ける。(ジョルダンの定理) その内部を D としよう。以下、$y \neq x$ と仮定する。さて、S は切断であったから、弧 $\widehat{yx}$ 上でベクトル場 a は

① D に対してつねに内向き

② D に対してつねに外向き

のいずれかをみたす。
①γ の残りの部分は軌道 $T_t x$ の一部ゆえ、流れは D の中に閉じ込められ、D から出ることができない。したがって、$T_s x,\ s \geq t$ は D 内にとどまる。よって、x の近くの点は二度と x の近くに戻れないことになり、点 x の再帰性に反する。
②流れは D から出るのみで、入ることができず、やはり点 x の再帰性に反する。ゆえに、$y = T_t x$ であるので、点 x は周期点である。

命題 2 (ジョルダンの定理) 定理 1 におけるジョルダンの定理とは、平面 $\boldsymbol{R}^2$ 内の $Jordan$ 閉曲線 J は $\boldsymbol{R}^2$ を内、外 2 つの領域に分ける、ということである。詳しく言えば、' $\boldsymbol{R}^2 \setminus J$ はちょうど 2 つの領域 $G_1,\ G_2$ の直和 $G_1 + G_2$ となり、J は $G_1,\ G_2$ の共通の境界となる ' 。なおこの時 p を J の任意の 1 点とすれば、p を一端とする $Jordan$ 弧で p 以外の部分は $G_i\ (i = 1, 2)$ に含まれるものが存在する、ということである。

3.2　定理 2

定理 2 平面上の C^1 級の流れに対して，自己回帰的な点は、不動点または周期点である。

証明) 不動点でない自己回帰的な点を x とすれば、$a(x) \neq 0$ である。そこで、$a(x)$ と直交する、点 x 中心の線分 S の長さを十分短く取れば、S の各点で，S とベクトル場 a は、直角に近い角度で交わる。ゆえに S は切断である。さらに、x は自己回帰的であるから、点 x の任

意の近傍 V に対して、$T_\tau x \in V$ となる、正数 τ がとれる。S を十分短くとり、δ を十分小さくとって

$$U = \{T_s y;\ y \in S,\ -\delta \leq s \leq \delta\} \tag{3.1}$$

とおけば、U は点 x の近傍であり、$U \subset V$ となる。x が自己回帰的なので、$T_t x \in U$ となる $t{>}s$ が存在する。このとき、$y \in S$ と $|\ s\ | \leq \delta$ に対して、$T_t x = T_s y$。したがって、$y = T_{t-s} x \in S$ さらに $t - s{>}0$ なので、S は点 x のポアンカレ切断であることがわかる。したがって、定理 1 より、x は周期点である。よって定理 2 は証明された。

備考　(3.1) の形にかける集合を、流れ箱と呼ぶ。

3.3　ω 極限集合

流れ、T_t による点 x の正の半軌道 $T_t x,\ t{>}0$ の集積点全体の作る集合を点 x の ω 極限集合とよび、$L_\omega(x)$ であらわす。ここで、点 y が $T_t x,\ t{>}0$ の集積点であるとは、$\lim t_n = +\infty$ となる適当な数列 $(t_n)_{n\geq 0}$ に対して、y が $T_{t_n} x$ の極限となることを言う。時間を反転して、負の半軌道 $T_{t_n} x,\ t{<}0$ から、同様に定義される集合を、点 x の α 極限集合とよぶ。

3.4　定理 3

定理 3 ある点の ω 極限集合に属する任意の点は再帰的である。

証明) $y \in L_\omega(x)$ とする。$\lim t_n = +\infty$ となるある数列 $(t_n)_{n\geq 0}$ に対して $y = \lim_{n\to\infty} T_{t_n} x$ であるから、y の任意の近傍 U に対して、ある番号 n_0 がとれて、$n \geq n_0$ ならば、$T_{t_n} x \in U$ である。よって、$S_n = t_n - t_{n_0}$ とおけば、$\lim_{n\to\infty} s_n = +\infty$ かつ $T_{s_n}(T_{t_{n_0}}) \in U$、よって、$T_{-s_n}(U) \cap U \neq \infty$ ゆえに、y は再帰点である。

3.5　定理 4

定理 4 ある点 x の ω 極限集合 $L_\omega(x)$ に属する点 y をとる。もし、点 y の ω 極限集合 $L_\omega(y)$ が、不動点でない点 z を含むならば、点 y はポアンカレ切断をもつ再帰的な点であり、したがって、y は周期点である。

証明) 点 z は不動点でないから、点 z を通る切断 S を作ることができる。$z \in L_\omega(y)$ ゆえ、

$$\lim_{n\to\infty} T_{t_n} y = z, \quad \lim_{n\to\infty} T_n = \infty$$

となる数列 (t_n) がとれる。よって S から流れ箱を定理 2 と同様に作れば、

$$T_s y \in S, \quad T_t y \in S, \quad s{<}t$$

となる、s, t が無数に存在する。ゆえに、S は点 $T_s y$ のポアンカレ切断となる。一方、y が再帰的であるから、$T_s y$ もそうである。ゆえに、$T_s y$ は周期点、したがって、y 自身も周期点である。

◎ この定理で得られたように、ある点の軌道 $T_t x$ が $t \to \infty$ で近づいていくような周期軌道を極限周期軌道 (limit cycle) という。

第4章　ポアンカレ・ベンディクソンの定理

定理 5 (ポアンカレベンディクソンの定理)
◎ 平面上の微分方程式の定める流れを考える。このとき、ある点の ω 極限集合がコンパクトで、不動点を含まないならば、この ω 極限集合は周期軌道である。　$\cdots$①
言い換え $\Rightarrow$
◎ 平面上の微分方程式の定める流れに対して、ある有解閉領域で閉じ込めが生じており、かつこの領域に不動点が全くなければ、この領域内に周期軌道が存在する。　$\cdots$②

★つまり、平面上のベクトル場は典型的には源点 (ソース) 、沈点 (シンク) 、鞍点 (サドル) 、極限周期軌道 (リミットサイクル) の 4 つの運動以外にはないということを述べている。
証明) ②の仮定をみたす有界閉領域 D の任意の点 x をとるとき $L_\omega(x)$ は明らかに有界であり、かつ閉集合である。また仮定から、$L_\omega(x)$ は不動点を含まない。ゆえに、①を示せば十分である。さて、点 x の ω 極限集合 $L_\omega(x)$ がコンパクトであり、不動点を全く含まないとする。$y \in L_\omega(x)$ とすると、定義より、$L_\omega(y)$ は $L_\omega(x)$ に含まれる空でない集合である。さらに、$z \in L_\omega(y)$ ならば、点 z は仮定より不動点でない。ゆえに、定理 3 より、y は周期点である。よって、$L_\omega(x)$ が点 y の軌道そのものであることを示せば証明は終わる。
　点 y 自身は不動点でないので、y を通るポアンカレ切断 S をとることができる。$y \in L_\omega(x)$ であったから、$\lim t_n = +\infty,\ \lim T_{t_n}x = y$ となる数列 $(t_n)_{n\geq 0}$ がとれる。したがって、$T_{s_0} \in S$ となる正数 s_0 が存在し、$s{>}s_0$ で s に戻る時刻を順に $s_1,\ s_2,\ \cdots$ とすれば、$\lim_{n\to\infty} T_{s_n}x = y$ である。よって、y の周期を t とすれば、$T_t y = y$ であるから、

$$\lim_{n\to\infty} T_{t+s_n}x = \lim_{n\to\infty} T_t(T_{s_n}x) = T_t y = y.$$

ゆえに、$T_s x, s \geq s_0$ は曲線 $C_n : T_s x, s_n \leq s \leq s_{n+1}$ の和集合であり、各曲線 C_n は

$$C_n : T_s(T_{s_n}x),\ 0 \leq s \leq s_{n+1} - s_n (\leq t)$$

と書けるので、$n \to \infty$ で、y の軌道に近づく。したがって、$L_\omega(x)$ は y の軌道に含まれる。よって、y の軌道と一致する。

第5章　プログラムと数値実験

5.1　例 1

$$\begin{cases} \dfrac{dx}{dt} = y \\ \dfrac{dy}{dt} = -6x - y - 3x^2 \end{cases}$$

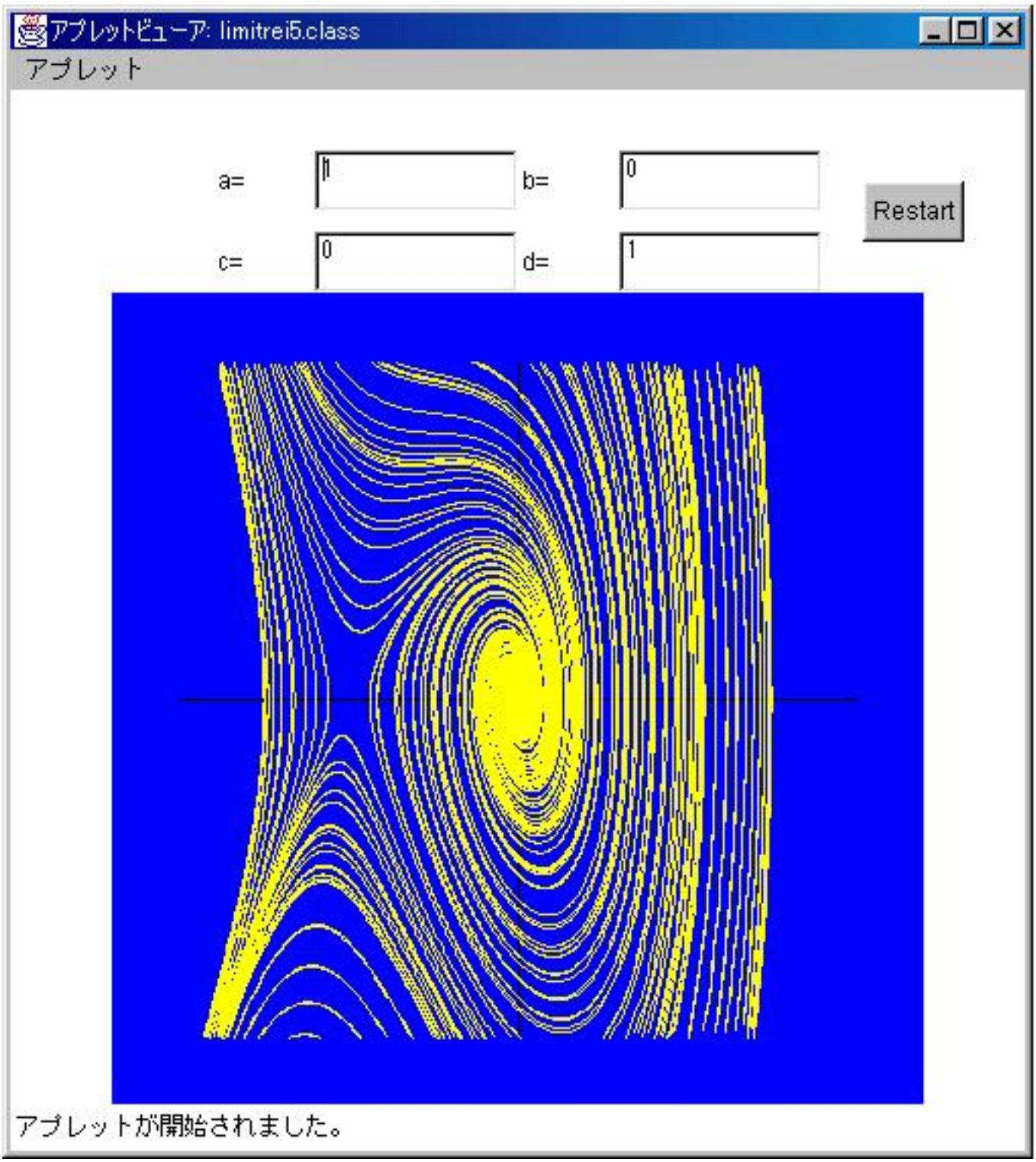
この微分方程式の不動点は $(x, y) = (0, 0),\ (-2, 0)$ である。また、この結果の描画範囲は $-4 \leq x,\ y \leq 4$ である。この微分方程式は、笠原 [2] を参考にした。

```
 1
 2  // <APPLET code="limitrei5.class" width=500 height=500> </APPLET>
 3  // limitrei5
 4
 5  import java.applet.*;
 6  import java.awt.*;
 7  import java.awt.event.*;
 8
 9  class GraphCanvas extends Canvas {
10
11    static final boolean DEBUG = false; // true;
12    private static final String message = "graph of a function with 1 variable";
13    // 問題に取って基本的なパラメーター
14    // 係数行列
15    private double a, b, c, d;
16    // 描画範囲
17    private double x_max =   4.0;
18    private double x_min = - 4.0;
19    private double x_margin = (x_max - x_min) / 10;
20    private double y_max =   4.0;
21    private double y_min = - 4.0;
22    private double y_margin = (y_max - y_min) / 10;
23
24    // 座標系の変換のためのパラメーター
25    private int CanvasX = 400, CanvasY = 400;
26    private double ratiox, ratioy, X0, Y0;
27
28    // コンストラクター
29    public GraphCanvas() {
30      super();
31    }
32    public GraphCanvas(int cx, int cy) {
33      super();
34      CanvasX = cx; CanvasY = cy;
35    }
36
```

```
37    public void compute(double A, double B, double C, double D) {
38      a = A; b = B; c = C; d = D;
39      repaint();
40    }
41
42    private boolean IsIn(double x, double y) {
43      return (x_min <= x && x <= x_max && y_min <= y && y <= y_max);
44    }
45    // 座標変換の準備
46    private void space(double x0, double y0, double x1, double y1) {
47      X0 = x0; Y0 = y0;
48      ratiox = CanvasX / (x1 - x0);
49      ratioy = CanvasY / (y1 - y0);
50    }
51    // ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
52    private int wx(double x) {
53      return (int)(ratiox * (x - X0));
54    }
55    // ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
56    private int wy(double y) {
57      return CanvasY - (int)(ratioy * (y - Y0));
58    }
59    // 力学系の右辺 f=(fx, fy)
60    private double fx(double x, double y) {
61      return y;
62    }
63    private double fy(double x, double y) {
64      return -6*x-y-3*x*x;
65    }
66    // x[], y[] の内容をグラフにする
67    private void drawGraph(Graphics g, double x0, double y0, double T) {
68      double h = 0.01 / Math.sqrt(a * a + b * b + c * c + d * d);
69      if (T < 0.0)
70        h = - h;
71      int iter = (int)Math.rint(Math.abs(T / h));
72      double x = x0;
73      double y = y0;
74      double new_x, new_y;
75      for (int i = 0; i <= iter; i++) {
76        double k1x = h * fx(x, y);
77        double k1y = h * fy(x, y);
78        double k2x = h * fx(x + k1x / 2, y + k1y / 2);
79        double k2y = h * fy(x + k1x / 2, y + k1y / 2);
80        double k3x = h * fx(x + k2x / 2, y + k2y / 2);
81        double k3y = h * fy(x + k2x / 2, y + k2y / 2);
82        double k4x = h * fx(x + k3x, y + k3y);
83        double k4y = h * fy(x + k3x, y + k3y);
84        new_x = x + (k1x + 2 * k2x + 2 * k3x + k4x) / 6;
85        new_y = y + (k1y + 2 * k2y + 2 * k3y + k4y) / 6;
86        if (IsIn(x, y) && IsIn(new_x, new_y))
87            g.drawLine(wx(x), wy(y), wx(new_x), wy(new_y));
88        x = new_x; y = new_y;
89      }
90    }
91
92    public void paint(Graphics g) {
93      //
94      space(x_min - x_margin, y_min - y_margin,
```

```
95            x_max + x_margin, y_max + y_margin);
96      //
97      setBackground(Color.blue);
98      //
99      g.setColor(Color.black);
100     g.drawLine(wx(x_min), wy(0.0), wx(x_max), wy(0.0));
101     g.drawLine(wx(0.0), wy(y_min), wx(0.0), wy(y_max));
102     //
103     g.setColor(Color.yellow);
104     int n = 15;
105     double dt = 2 * Math.PI / n;
106     double Time = 10.0;
107     for (int i = 0; i < n; i++) {
108       double t = i * dt;
109       drawGraph(g, Math.cos(t), Math.sin(t),   Time);
110       drawGraph(g, Math.cos(t), Math.sin(t), - Time);
111     }
112     n=80; dt= 2 * Math.PI /n;
113     for (int i = 0; i < n; i++) {
114      double t = i * dt, r=3;
115       drawGraph(g,r* Math.cos(t),r*Math.sin(t),   Time);
116       drawGraph(g,r* Math.cos(t), r*Math.sin(t), - Time);
117     }
118   }
119 }
120
121 public class limitrei5 extends Applet implements ActionListener {
122
123   private int N = 20;
124   private double lambda = 0.5;
125   private double Tmax = 0.5;
126   // ユーザーとのインターフェイス（パラメーターの入力）
127   private Label label_a, label_b, label_c, label_d;
128   private TextField input_a, input_b, input_c, input_d;
129   private double a, b, c, d;
130   private Button startB;
131   //
132   private GraphCanvas gc;
133
134      private void ReadFields() {
135     a = Double.valueOf(input_a.getText()).doubleValue();
136     b = Double.valueOf(input_b.getText()).doubleValue();
137     c = Double.valueOf(input_c.getText()).doubleValue();
138     d = Double.valueOf(input_d.getText()).doubleValue();
139   }
140   // 準備（変数の用意と座標系の初期化など）
141   public void init() {
142     // ナル・レイアウト
143     setLayout(null);
144     // a, b, c, d を入力するためのテキスト・フィールド
145      add(label_a = new Label("a=")); label_a.setBounds(100, 30, 40, 30);
146     add(label_b = new Label("b=")); label_b.setBounds(250, 30, 40, 30);
147     add(label_c = new Label("c=")); label_c.setBounds(100, 70, 40, 30);
148     add(label_d = new Label("d=")); label_d.setBounds(250, 70, 40, 30);
149     add(input_a = new TextField("" + 1)); input_a.setBounds(150, 30, 100, 30);
150     add(input_b = new TextField("" + 0)); input_b.setBounds(300, 30, 100, 30);
151     add(input_c = new TextField("" + 0)); input_c.setBounds(150, 70, 100, 30);
152     add(input_d = new TextField("" + 1)); input_d.setBounds(300, 70, 100, 30);
```

```
153       // 再計算ボタン
154       startB = new Button("Restart");
155       add(startB);
156       startB.setBounds(420, 45, 50, 30);
157       startB.addActionListener(this);
158
159       // キャンバス
160       gc = new GraphCanvas();
161       add(gc);
162       gc.setBounds(50, 100, 400, 400);
163       ReadFields();
164       gc.compute(a, b, c, d);
165     }
166
167     // ボタンを押されたら、テキスト・フィールドの内容を読み取って、再描画
168     public void actionPerformed(ActionEvent e) {
169       if (e.getSource() == startB) {
170         ReadFields();
171         gc.compute(a, b, c, d);
172       }
173     }
174   }
175
```


アプレットビューア: limitrei5.class
アプレット
a=
1
b=
0
Restart
c=
0
d=
1
アプレットが開始されました。

5.2 例 2

$$\begin{cases} \dfrac{dx}{dt} = y \\ \dfrac{dy}{dt} = -x + dy - y(x^2 + y^2 - a^2) \end{cases}$$

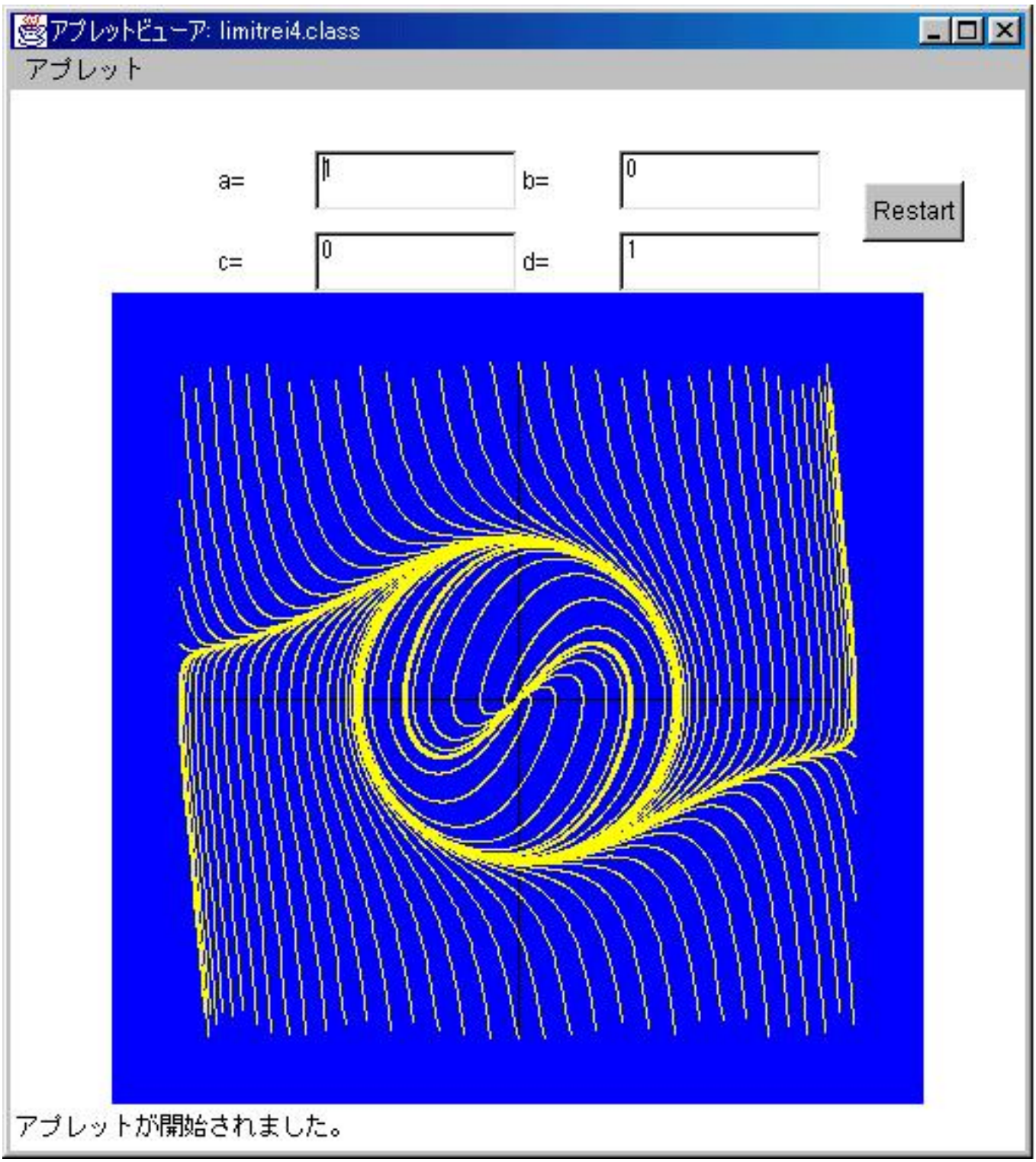
この微分方程式の不動点は $(x, y) = (0, 0)$ であり、パラメータ $a,\ d$ は正定数とする。また、この結果の描画範囲は $-3 \leq x,\ y \leq 3$ である。この微分方程式は、笠原 [2] を参考にした。

```
 1
 2  // <APPLET code="limitrei4.class" width=500 height=500> </APPLET>
 3  // limitrei4
 4
 5  import java.applet.*;
 6  import java.awt.*;
 7  import java.awt.event.*;
 8
 9  class GraphCanvas extends Canvas {
10
11    static final boolean DEBUG = false; // true;
12    private static final String message = "graph of a function with 1 variable";
13    // 問題に取って基本的なパラメーター
14    // 係数行列
15    private double a, b, c, d;
16    // 描画範囲
17    private double x_max =   3.0;
18    private double x_min = - 3.0;
19    private double x_margin = (x_max - x_min) / 10;
20    private double y_max =   3.0;
21    private double y_min = - 3.0;
22    private double y_margin = (y_max - y_min) / 10;
23
24    // 座標系の変換のためのパラメーター
25    private int CanvasX = 400, CanvasY = 400;
26    private double ratiox, ratioy, X0, Y0;
27
28    // コンストラクター
29    public GraphCanvas() {
30      super();
31    }
32    public GraphCanvas(int cx, int cy) {
33      super();
34      CanvasX = cx; CanvasY = cy;
35    }
36
37    public void compute(double A, double B, double C, double D) {
38      a = A; b = B; c = C; d = D;
39      repaint();
40    }
41
42    private boolean IsIn(double x, double y) {
43      return (x_min <= x && x <= x_max && y_min <= y && y <= y_max);
44    }
45    // 座標変換の準備
46    private void space(double x0, double y0, double x1, double y1) {
```

```
47      X0 = x0; Y0 = y0;
48      ratiox = CanvasX / (x1 - x0);
49      ratioy = CanvasY / (y1 - y0);
50    }
51    // ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
52    private int wx(double x) {
53      return (int)(ratiox * (x - X0));
54    }
55    // ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
56    private int wy(double y) {
57      return CanvasY - (int)(ratioy * (y - Y0));
58    }
59    // 力学系の右辺 f=(fx, fy)
60    private double fx(double x, double y) {
61      return y;
62    }
63    private double fy(double x, double y) {
64      return -x+d*y-y*(x*x+y*y-a*a);
65    }
66    // x[], y[] の内容をグラフにする
67    private void drawGraph(Graphics g, double x0, double y0, double T) {
68      double h = 0.01 / Math.sqrt(a * a + b * b + c * c + d * d);
69      if (T < 0.0)
70        h = - h;
71      int iter = (int)Math.rint(Math.abs(T / h));
72      double x = x0;
73      double y = y0;
74      double new_x, new_y;
75      for (int i = 0; i <= iter; i++) {
76        double k1x = h * fx(x, y);
77        double k1y = h * fy(x, y);
78        double k2x = h * fx(x + k1x / 2, y + k1y / 2);
79        double k2y = h * fy(x + k1x / 2, y + k1y / 2);
80        double k3x = h * fx(x + k2x / 2, y + k2y / 2);
81        double k3y = h * fy(x + k2x / 2, y + k2y / 2);
82        double k4x = h * fx(x + k3x, y + k3y);
83        double k4y = h * fy(x + k3x, y + k3y);
84        new_x = x + (k1x + 2 * k2x + 2 * k3x + k4x) / 6;
85        new_y = y + (k1y + 2 * k2y + 2 * k3y + k4y) / 6;
86        if (IsIn(x, y) && IsIn(new_x, new_y))
87            g.drawLine(wx(x), wy(y), wx(new_x), wy(new_y));
88        x = new_x; y = new_y;
89      }
90    }
91
92    public void paint(Graphics g) {
93      //
94      space(x_min - x_margin, y_min - y_margin,
95            x_max + x_margin, y_max + y_margin);
96      //
97      setBackground(Color.blue);
98      //
99      g.setColor(Color.black);
100     g.drawLine(wx(x_min), wy(0.0), wx(x_max), wy(0.0));
101     g.drawLine(wx(0.0), wy(y_min), wx(0.0), wy(y_max));
102     //
103     g.setColor(Color.yellow);
104     int n = 18;
```

```
105      double dt = 2 * Math.PI / n;
106      double Time = 10.0;
107      for (int i = 0; i < n; i++) {
108        double t = i * dt;
109        drawGraph(g, Math.cos(t), Math.sin(t),   Time);
110        drawGraph(g, Math.cos(t), Math.sin(t), - Time);
111      }
112      n=80; dt= 2 * Math.PI /n;
113      for (int i = 0; i < n; i++) {
114       double t = i * dt, r=3;
115        drawGraph(g,r* Math.cos(t),r*Math.sin(t),   Time);
116        drawGraph(g,r* Math.cos(t), r*Math.sin(t), - Time);
117      }
118    }
119  }
120
121  public class limitrei4 extends Applet implements ActionListener {
122
123    private int N = 20;
124    private double lambda = 0.5;
125    private double Tmax = 0.5;
126    // ユーザーとのインターフェイス（パラメーターの入力）
127    private Label label_a, label_b, label_c, label_d;
128    private TextField input_a, input_b, input_c, input_d;
129    private double a, b, c, d;
130    private Button startB;
131    //
132    private GraphCanvas gc;
133
134    private void ReadFields() {
135      a = Double.valueOf(input_a.getText()).doubleValue();
136      b = Double.valueOf(input_b.getText()).doubleValue();
137      c = Double.valueOf(input_c.getText()).doubleValue();
138      d = Double.valueOf(input_d.getText()).doubleValue();
139    }
140    // 準備（変数の用意と座標系の初期化など）
141    public void init() {
142      // ナル・レイアウト
143      setLayout(null);
144      // a, b, c, d を入力するためのテキスト・フィールド
145      add(label_a = new Label("a=")); label_a.setBounds(100, 30, 40, 30);
146      add(label_b = new Label("b=")); label_b.setBounds(250, 30, 40, 30);
147      add(label_c = new Label("c=")); label_c.setBounds(100, 70, 40, 30);
148      add(label_d = new Label("d=")); label_d.setBounds(250, 70, 40, 30);
149      add(input_a = new TextField("" + 1)); input_a.setBounds(150, 30, 100, 30);
150      add(input_b = new TextField("" + 0)); input_b.setBounds(300, 30, 100, 30);
151      add(input_c = new TextField("" + 0)); input_c.setBounds(150, 70, 100, 30);
152      add(input_d = new TextField("" + 1)); input_d.setBounds(300, 70, 100, 30);
153      // 再計算ボタン
154      startB = new Button("Restart");
155      add(startB);
156      startB.setBounds(420, 45, 50, 30);
157      startB.addActionListener(this);
158
159      // キャンバス
160      gc = new GraphCanvas();
161      add(gc);
162      gc.setBounds(50, 100, 400, 400);
```

```
163       ReadFields();
164       gc.compute(a, b, c, d);
165     }
166
167     // ボタンを押されたら、テキスト・フィールドの内容を読み取って、再描画
168     public void actionPerformed(ActionEvent e) {
169       if (e.getSource() == startB) {
170         ReadFields();
171         gc.compute(a, b, c, d);
172       }
173     }
174   }
175
```


アプレットビューア: limitrei4.class
アプレット
a=
1
b=
0
Restart
c=
0
d=
1
アプレットが開始されました。

5.3 例 3

$$\begin{cases} \dfrac{dx}{dt} = -x - y + x(x^2 + 2y^2) \\ \dfrac{dx}{dt} = x - y + y(x^2 + 2y^2) \end{cases}$$

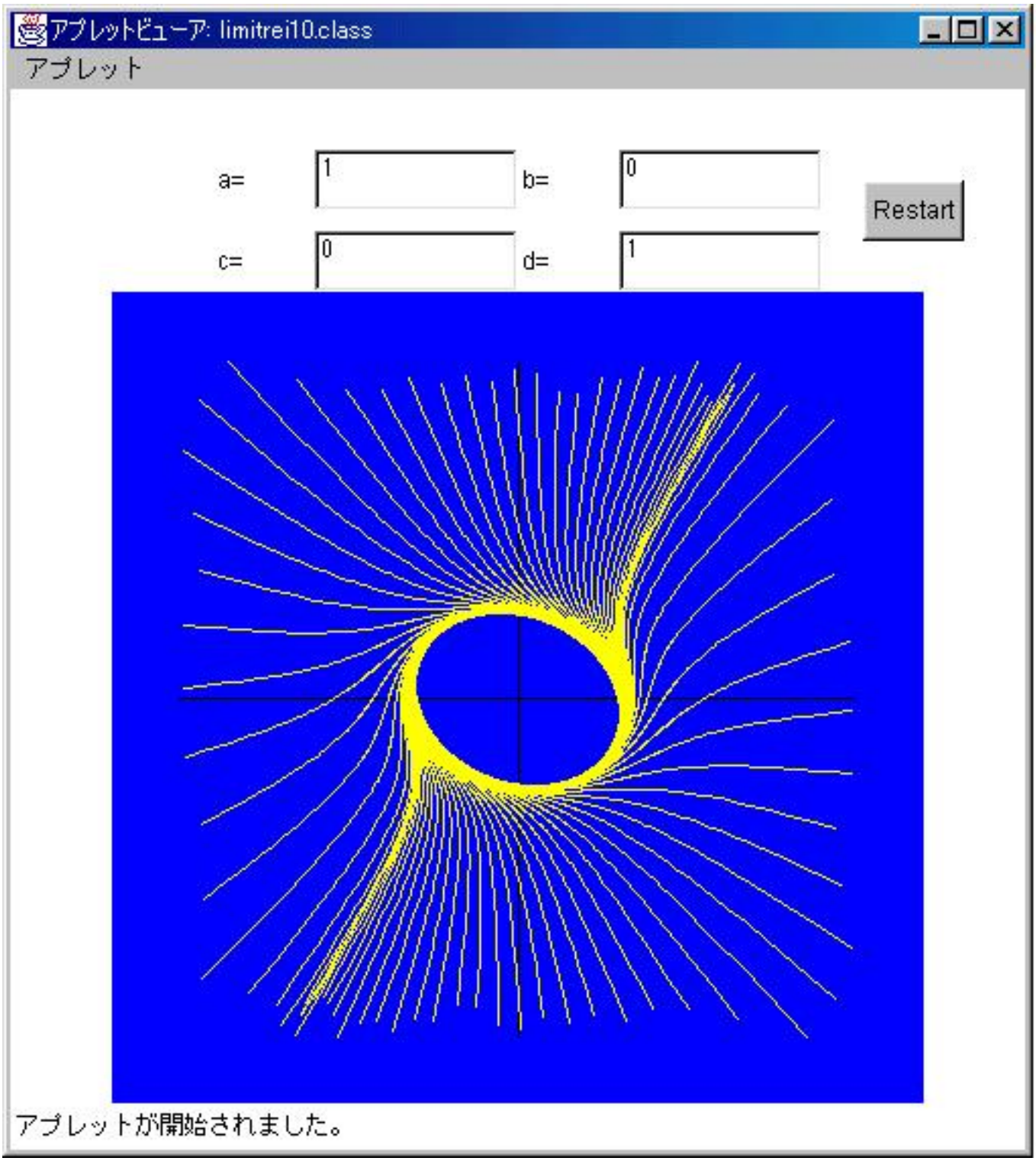
この微分方程式の不動点は $(x, y) = (0, 0)$ である。また、この結果の描画範囲は $-3 \leq x,\ y \leq 3$ である。この微分方程式は、笠原 [2] を参考にした。

```
1  // <APPLET code="limitrei10.class" width=500 height=500> </APPLET>
2
3  import java.applet.*;
4  import java.awt.*;
5  import java.awt.event.*;
6
7  class GraphCanvas extends Canvas {
8
9    static final boolean DEBUG = false; // true;
10   private static final String message = "graph of a function with 1 variable";
11   // 問題に取って基本的なパラメーター
12   // 係数行列
13   private double a, b, c, d;
14   // 描画範囲
15   private double x_max =   3.0;
16   private double x_min = - 3.0;
17   private double x_margin = (x_max - x_min) / 10;
18   private double y_max =   3.0;
19   private double y_min = - 3.0;
20   private double y_margin = (y_max - y_min) / 10;
21
22   // 座標系の変換のためのパラメーター
23   private int CanvasX = 400, CanvasY = 400;
24   private double ratiox, ratioy, X0, Y0;
25
26   // コンストラクター
27   public GraphCanvas() {
28     super();
29   }
30   public GraphCanvas(int cx, int cy) {
31     super();
32     CanvasX = cx; CanvasY = cy;
33   }
34
35   public void compute(double A, double B, double C, double D) {
36     a = A; b = B; c = C; d = D;
37     repaint();
38   }
39
40   private boolean IsIn(double x, double y) {
41     return (x_min <= x && x <= x_max && y_min <= y && y <= y_max);
42   }
43   // 座標変換の準備
44   private void space(double x0, double y0, double x1, double y1) {
45     X0 = x0; Y0 = y0;
46     ratiox = CanvasX / (x1 - x0);
```

```
47      ratioy = CanvasY / (y1 - y0);
48    }
49    // ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
50    private int wx(double x) {
51      return (int)(ratiox * (x - X0));
52    }
53    // ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
54    private int wy(double y) {
55      return CanvasY - (int)(ratioy * (y - Y0));
56    }
57    // 力学系の右辺 f=(fx, fy)
58    private double fx(double x, double y) {
59      return -x-y+x*(x*x+2*y*y);
60    }
61    private double fy(double x, double y) {
62      return x-y+y*(x*x+2*y*y);
63    }
64    // x[], y[] の内容をグラフにする
65    private void drawGraph(Graphics g, double x0, double y0, double T) {
66      double h = 0.01 / Math.sqrt(a * a + b * b + c * c + d * d);
67      if (T < 0.0)
68        h = - h;
69      int iter = (int)Math.rint(Math.abs(T / h));
70      double x = x0;
71      double y = y0;
72      double new_x, new_y;
73      for (int i = 0; i <= iter; i++) {
74        double k1x = h * fx(x, y);
75        double k1y = h * fy(x, y);
76        double k2x = h * fx(x + k1x / 2, y + k1y / 2);
77        double k2y = h * fy(x + k1x / 2, y + k1y / 2);
78        double k3x = h * fx(x + k2x / 2, y + k2y / 2);
79        double k3y = h * fy(x + k2x / 2, y + k2y / 2);
80        double k4x = h * fx(x + k3x, y + k3y);
81        double k4y = h * fy(x + k3x, y + k3y);
82        new_x = x + (k1x + 2 * k2x + 2 * k3x + k4x) / 6;
83        new_y = y + (k1y + 2 * k2y + 2 * k3y + k4y) / 6;
84        if (IsIn(x, y) && IsIn(new_x, new_y))
85            g.drawLine(wx(x), wy(y), wx(new_x), wy(new_y));
86        x = new_x; y = new_y;
87      }
88    }
89
90    public void paint(Graphics g) {
91      //
92      space(x_min - x_margin, y_min - y_margin,
93            x_max + x_margin, y_max + y_margin);
94      //
95      setBackground(Color.blue);
96      //
97      g.setColor(Color.black);
98      g.drawLine(wx(x_min), wy(0.0), wx(x_max), wy(0.0));
99      g.drawLine(wx(0.0), wy(y_min), wx(0.0), wy(y_max));
100     //
101     g.setColor(Color.yellow);
102     int n = 80;
103     double dt = 2 * Math.PI / n;
104     double Time = 10.0 / Math.sqrt(a * a + b * b + c * c + d * d);
```

```
105       for (int i = 0; i < n; i++) {
106         double t = i * dt;
107         drawGraph(g, Math.cos(t), Math.sin(t),   Time);
108         drawGraph(g, Math.cos(t), Math.sin(t), - Time);
109       }
110     }
111   }
112
113   public class limitrei10 extends Applet implements ActionListener {
114
115     private int N = 20;
116     private double lambda = 0.5;
117     private double Tmax = 0.5;
118     // ユーザーとのインターフェイス（パラメーターの入力）
119     private Label label_a, label_b, label_c, label_d;
120     private TextField input_a, input_b, input_c, input_d;
121     private double a, b, c, d;
122     private Button startB;
123     //
124     private GraphCanvas gc;
125
126     private void ReadFields() {
127       a = Double.valueOf(input_a.getText()).doubleValue();
128       b = Double.valueOf(input_b.getText()).doubleValue();
129       c = Double.valueOf(input_c.getText()).doubleValue();
130       d = Double.valueOf(input_d.getText()).doubleValue();
131     }
132     // 準備（変数の用意と座標系の初期化など）
133     public void init() {
134       // ナル・レイアウト
135       setLayout(null);
136       // a, b, c, d を入力するためのテキスト・フィールド
137       add(label_a = new Label("a=")); label_a.setBounds(100, 30, 40, 30);
138       add(label_b = new Label("b=")); label_b.setBounds(250, 30, 40, 30);
139       add(label_c = new Label("c=")); label_c.setBounds(100, 70, 40, 30);
140       add(label_d = new Label("d=")); label_d.setBounds(250, 70, 40, 30);
141       add(input_a = new TextField("" + 1)); input_a.setBounds(150, 30, 100, 30);
142       add(input_b = new TextField("" + 0)); input_b.setBounds(300, 30, 100, 30);
143       add(input_c = new TextField("" + 0)); input_c.setBounds(150, 70, 100, 30);
144       add(input_d = new TextField("" + 1)); input_d.setBounds(300, 70, 100, 30);
145       // 再計算ボタン
146       startB = new Button("Restart");
147       add(startB);
148       startB.setBounds(420, 45, 50, 30);
149       startB.addActionListener(this);
150
151       // キャンバス
152       gc = new GraphCanvas();
153       add(gc);
154       gc.setBounds(50, 100, 400, 400);
155       ReadFields();
156       gc.compute(a, b, c, d);
157     }
158
159     // ボタンを押されたら、テキスト・フィールドの内容を読み取って、再描画
160     public void actionPerformed(ActionEvent e) {
161       if (e.getSource() == startB) {
162         ReadFields();
```

```
163         gc.compute(a, b, c, d);
164       }
165     }
166   }
167
168
```



5.4 ファンデルポール方程式

$$\begin{cases} \dfrac{dx}{dt} = y \\ \dfrac{dy}{dt} = -x + a(1-x^2)y \end{cases}$$

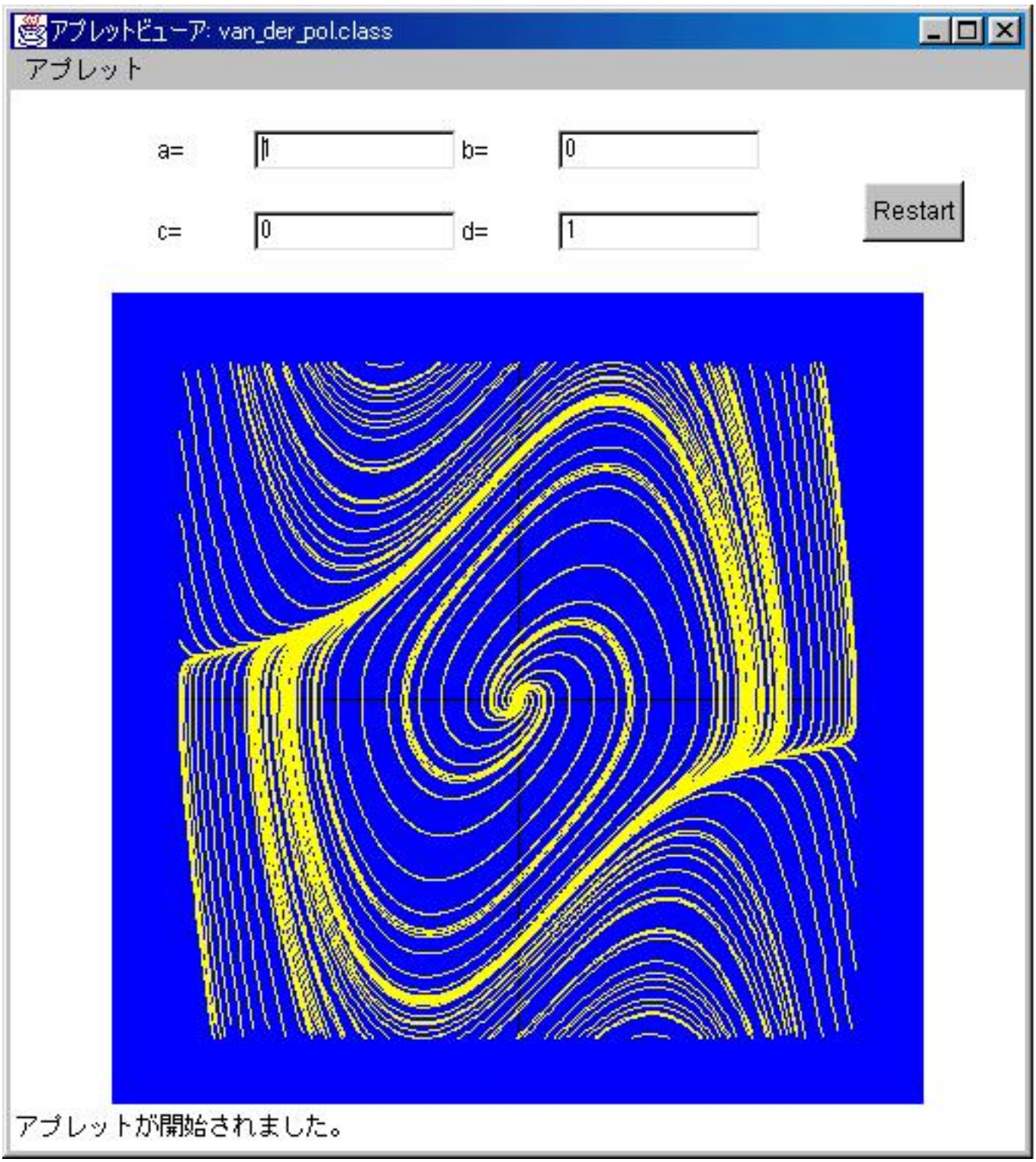
この微分方程式の不動点は $(x, y) = (0, 0)$ であり、パラメータ a は正定数とする。また、この結果の描画範囲は $-3 \leq x,\ y \leq 3$ である。この微分方程式は、高橋 [1] を参考にした。

```
 1
 2  // <APPLET code="van_der_pol.class" width=500 height=500> </APPLET>
 3  // van Der Pol
 4
 5  import java.applet.*;
 6  import java.awt.*;
 7  import java.awt.event.*;
 8
 9  class GraphCanvas extends Canvas {
10
11    static final boolean DEBUG = false; // true;
12    private static final String message = "graph of a function with 1 variable";
13    // 問題に取って基本的なパラメーター
14    // 係数行列
15    private double a, b, c, d;
16    // 描画範囲
17    private double x_max =   3.0;
18    private double x_min = - 3.0;
19    private double x_margin = (x_max - x_min) / 10;
20    private double y_max =   3.0;
21    private double y_min = - 3.0;
22    private double y_margin = (y_max - y_min) / 10;
23
24    // 座標系の変換のためのパラメーター
25    private int CanvasX = 400, CanvasY = 400;
26    private double ratiox, ratioy, X0, Y0;
27
28    // コンストラクター
29    public GraphCanvas() {
30      super();
31    }
32    public GraphCanvas(int cx, int cy) {
33      super();
34      CanvasX = cx; CanvasY = cy;
35    }
36
37    public void compute(double A, double B, double C, double D) {
38      a = A; b = B; c = C; d = D;
39      repaint();
40    }
41
42    private boolean IsIn(double x, double y) {
43      return (x_min <= x && x <= x_max && y_min <= y && y <= y_max);
44    }
45    // 座標変換の準備
46    private void space(double x0, double y0, double x1, double y1) {
```

```
47      X0 = x0; Y0 = y0;
48      ratiox = CanvasX / (x1 - x0);
49      ratioy = CanvasY / (y1 - y0);
50    }
51    // ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
52    private int wx(double x) {
53      return (int)(ratiox * (x - X0));
54    }
55    // ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
56    private int wy(double y) {
57      return CanvasY - (int)(ratioy * (y - Y0));
58    }
59    // 力学系の右辺 f=(fx, fy)
60    private double fx(double x, double y) {
61      return y;
62    }
63    private double fy(double x, double y) {
64      return -x +d*((1 - x*x)*y);
65    }
66    // x[], y[] の内容をグラフにする
67    private void drawGraph(Graphics g, double x0, double y0, double T) {
68      double h = 0.01 / Math.sqrt(a * a + b * b + c * c + d * d);
69      if (T < 0.0)
70        h = - h;
71      int iter = (int)Math.rint(Math.abs(T / h));
72      double x = x0;
73      double y = y0;
74      double new_x, new_y;
75      for (int i = 0; i <= iter; i++) {
76        double k1x = h * fx(x, y);
77        double k1y = h * fy(x, y);
78        double k2x = h * fx(x + k1x / 2, y + k1y / 2);
79        double k2y = h * fy(x + k1x / 2, y + k1y / 2);
80        double k3x = h * fx(x + k2x / 2, y + k2y / 2);
81        double k3y = h * fy(x + k2x / 2, y + k2y / 2);
82        double k4x = h * fx(x + k3x, y + k3y);
83        double k4y = h * fy(x + k3x, y + k3y);
84        new_x = x + (k1x + 2 * k2x + 2 * k3x + k4x) / 6;
85        new_y = y + (k1y + 2 * k2y + 2 * k3y + k4y) / 6;
86        if (IsIn(x, y) && IsIn(new_x, new_y))
87            g.drawLine(wx(x), wy(y), wx(new_x), wy(new_y));
88        x = new_x; y = new_y;
89      }
90    }
91
92    public void paint(Graphics g) {
93      //
94      space(x_min - x_margin, y_min - y_margin,
95            x_max + x_margin, y_max + y_margin);
96      //
97      setBackground(Color.blue);
98      //
99      g.setColor(Color.black);
100     g.drawLine(wx(x_min), wy(0.0), wx(x_max), wy(0.0));
101     g.drawLine(wx(0.0), wy(y_min), wx(0.0), wy(y_max));
102     //
103     g.setColor(Color.yellow);
104     int n = 18;
```

```
105      double dt = 2 * Math.PI / n;
106      double Time = 10.0;
107      for (int i = 0; i < n; i++) {
108        double t = i * dt;
109        drawGraph(g, Math.cos(t), Math.sin(t),   Time);
110        drawGraph(g, Math.cos(t), Math.sin(t), - Time);
111      }
112      n=80; dt= 2 * Math.PI /n;
113      for (int i = 0; i < n; i++) {
114       double t = i * dt, r=3;
115        drawGraph(g,r* Math.cos(t),r*Math.sin(t),   Time);
116        drawGraph(g,r* Math.cos(t), r*Math.sin(t), - Time);
117      }
118    }
119  }
120
121  public class van_der_pol extends Applet implements ActionListener {
122
123    private int N = 20;
124    private double lambda = 0.5;
125    private double Tmax = 0.5;
126    // ユーザーとのインターフェイス（パラメーターの入力）
127    private Label label_a, label_b, label_c, label_d;
128    private TextField input_a, input_b, input_c, input_d;
129    private double a, b, c, d;
130    private Button startB;
131    //
132    private GraphCanvas gc;
133
134    private void ReadFields() {
135      a = Double.valueOf(input_a.getText()).doubleValue();
136      b = Double.valueOf(input_b.getText()).doubleValue();
137      c = Double.valueOf(input_c.getText()).doubleValue();
138      d = Double.valueOf(input_d.getText()).doubleValue();
139    }
140    // 準備（変数の用意と座標系の初期化など）
141    public void init() {
142      // ナル・レイアウト
143      setLayout(null);
144      // a, b, c, d を入力するためのテキスト・フィールド
145      add(label_a = new Label("a=")); label_a.setBounds(70, 20, 40, 20);
146      add(label_b = new Label("b=")); label_b.setBounds(220, 20, 40, 20);
147      add(label_c = new Label("c=")); label_c.setBounds(70, 60, 40, 20);
148      add(label_d = new Label("d=")); label_d.setBounds(220, 60, 40, 20);
149      add(input_a = new TextField("" + 1)); input_a.setBounds(120, 20, 100, 20);
150      add(input_b = new TextField("" + 0)); input_b.setBounds(270, 20, 100, 20);
151      add(input_c = new TextField("" + 0)); input_c.setBounds(120, 60, 100, 20);
152      add(input_d = new TextField("" + 1)); input_d.setBounds(270, 60, 100, 20);
153      // 再計算ボタン
154      startB = new Button("Restart");
155      add(startB);
156      startB.setBounds(420, 45, 50, 30);
157      startB.addActionListener(this);
158
159      // キャンバス
160      gc = new GraphCanvas();
161      add(gc);
162      gc.setBounds(50, 100, 400, 400);
```

```
163       ReadFields();
164       gc.compute(a, b, c, d);
165     }
166
167     // ボタンを押されたら、テキスト・フィールドの内容を読み取って、再描画
168     public void actionPerformed(ActionEvent e) {
169       if (e.getSource() == startB) {
170         ReadFields();
171         gc.compute(a, b, c, d);
172       }
173     }
174   }
175
```


アプレットビューア: van_der_pol.class
アプレット
a=
1
b=
0
c=
0
d=
1
Restart
アプレットが開始されました。

5.5 ダフィング方程式

$$\begin{cases} \dfrac{dx}{dt} = y \\ \dfrac{dy}{dt} = x^3 - x - ay \end{cases}$$

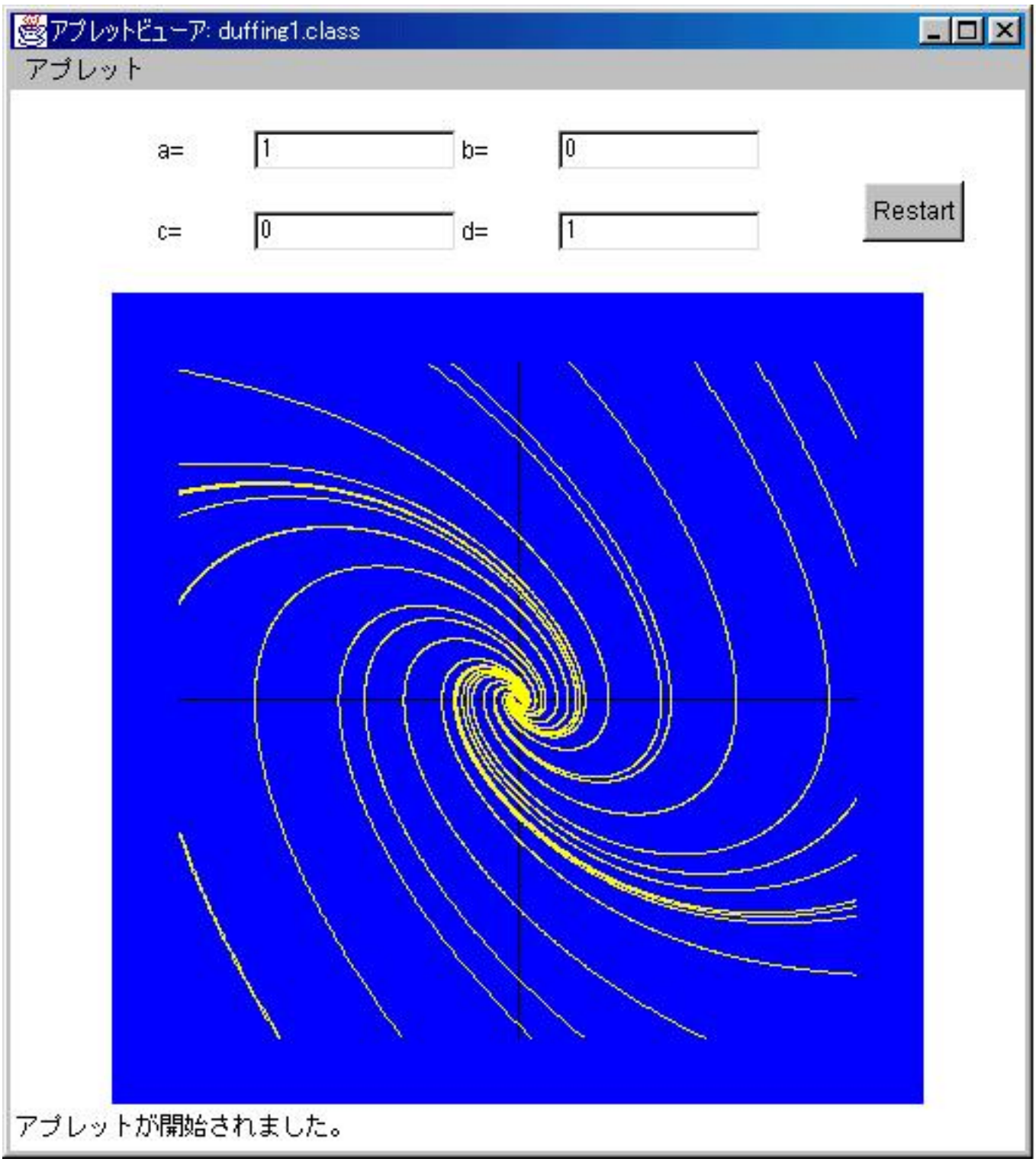
この微分方程式の不動点は $(x, y) = (0, 0),\ (1, 0),\ (-1, 0)$ であり、パラメータ a は正定数である。また、この結果の描画範囲は $-0.5 \leq x,\ y \leq 0.5$ である。この微分方程式は、高橋 [1] を参考にした。

```
 1
 2  // <APPLET code="duffing1.class" width=500 height=500> </APPLET>
 3  // Duffing
 4
 5  import java.applet.*;
 6  import java.awt.*;
 7  import java.awt.event.*;
 8
 9  class GraphCanvas extends Canvas {
10
11    static final boolean DEBUG = false; // true;
12    private static final String message = "graph of a function with 1 variable";
13    // 問題に取って基本的なパラメーター
14    // 係数行列
15    private double a, b, c, d;
16    // 描画範囲
17    private double x_max =   0.5;
18    private double x_min = - 0.5;
19    private double x_margin = (x_max - x_min) / 10;
20    private double y_max =   0.5;
21    private double y_min = - 0.5;
22    private double y_margin = (y_max - y_min) / 10;
23
24    // 座標系の変換のためのパラメーター
25    private int CanvasX = 400, CanvasY = 400;
26    private double ratiox, ratioy, X0, Y0;
27
28    // コンストラクター
29    public GraphCanvas() {
30      super();
31    }
32    public GraphCanvas(int cx, int cy) {
33      super();
34      CanvasX = cx; CanvasY = cy;
35    }
36
37    public void compute(double A, double B, double C, double D) {
38      a = A; b = B; c = C; d = D;
39      repaint();
40    }
41
42    private boolean IsIn(double x, double y) {
43      return (x_min <= x && x <= x_max && y_min <= y && y <= y_max);
44    }
45    // 座標変換の準備
```

```
46    private void space(double x0, double y0, double x1, double y1) {
47      X0 = x0; Y0 = y0;
48      ratiox = CanvasX / (x1 - x0);
49      ratioy = CanvasY / (y1 - y0);
50    }
51    // ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
52    private int wx(double x) {
53      return (int)(ratiox * (x - X0));
54    }
55    // ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
56    private int wy(double y) {
57      return CanvasY - (int)(ratioy * (y - Y0));
58    }
59    // 力学系の右辺 f=(fx, fy)
60    private double fx(double x, double y) {
61      return y;
62    }
63    private double fy(double x, double y) {
64      return x*x*x-x-a*y;
65    }
66    // x[], y[] の内容をグラフにする
67    private void drawGraph(Graphics g, double x0, double y0, double T) {
68      double h = 0.01 / Math.sqrt(a * a + b * b + c * c + d * d);
69      if (T < 0.0)
70        h = - h;
71      int iter = (int)Math.rint(Math.abs(T / h));
72      double x = x0;
73      double y = y0;
74      double new_x, new_y;
75      for (int i = 0; i <= iter; i++) {
76        double k1x = h * fx(x, y);
77        double k1y = h * fy(x, y);
78        double k2x = h * fx(x + k1x / 2, y + k1y / 2);
79        double k2y = h * fy(x + k1x / 2, y + k1y / 2);
80        double k3x = h * fx(x + k2x / 2, y + k2y / 2);
81        double k3y = h * fy(x + k2x / 2, y + k2y / 2);
82        double k4x = h * fx(x + k3x, y + k3y);
83        double k4y = h * fy(x + k3x, y + k3y);
84        new_x = x + (k1x + 2 * k2x + 2 * k3x + k4x) / 6;
85        new_y = y + (k1y + 2 * k2y + 2 * k3y + k4y) / 6;
86        if (IsIn(x, y) && IsIn(new_x, new_y))
87            g.drawLine(wx(x), wy(y), wx(new_x), wy(new_y));
88        x = new_x; y = new_y;
89      }
90    }
91
92    public void paint(Graphics g) {
93      //
94      space(x_min - x_margin, y_min - y_margin,
95            x_max + x_margin, y_max + y_margin);
96      //
97      setBackground(Color.blue);
98      //
99      g.setColor(Color.black);
100     g.drawLine(wx(x_min), wy(0.0), wx(x_max), wy(0.0));
101     g.drawLine(wx(0.0), wy(y_min), wx(0.0), wy(y_max));
102     //
103     g.setColor(Color.yellow);
```

```
104      int n = 25;
105      double dt = 2 * Math.PI / n;
106      double Time = 10.0;
107      for (int i = 0; i < n; i++) {
108        double t = i * dt;
109        drawGraph(g, Math.cos(t), Math.sin(t),   Time);
110        drawGraph(g, Math.cos(t), Math.sin(t), - Time);
111      }
112      n=80; dt= 2 * Math.PI /n;
113      for (int i = 0; i < n; i++) {
114       double t = i * dt, r=3;
115        drawGraph(g,r* Math.cos(t),r*Math.sin(t),   Time);
116        drawGraph(g,r* Math.cos(t), r*Math.sin(t), - Time);
117      }
118    }
119  }
120
121  public class duffing1 extends Applet implements ActionListener {
122
123    private int N = 20;
124    private double lambda = 0.5;
125    private double Tmax = 0.5;
126    // ユーザーとのインターフェイス（パラメーターの入力）
127    private Label label_a, label_b, label_c, label_d;
128    private TextField input_a, input_b, input_c, input_d;
129    private double a, b, c, d;
130    private Button startB;
131    //
132    private GraphCanvas gc;
133
134    private void ReadFields() {
135      a = Double.valueOf(input_a.getText()).doubleValue();
136      b = Double.valueOf(input_b.getText()).doubleValue();
137      c = Double.valueOf(input_c.getText()).doubleValue();
138      d = Double.valueOf(input_d.getText()).doubleValue();
139    }
140    // 準備（変数の用意と座標系の初期化など）
141    public void init() {
142      // ナル・レイアウト
143      setLayout(null);
144      // a, b, c, d を入力するためのテキスト・フィールド
145      add(label_a = new Label("a=")); label_a.setBounds(70, 20, 40, 20);
146      add(label_b = new Label("b=")); label_b.setBounds(220, 20, 40, 20);
147      add(label_c = new Label("c=")); label_c.setBounds(70, 60, 40, 20);
148      add(label_d = new Label("d=")); label_d.setBounds(220, 60, 40, 20);
149      add(input_a = new TextField("" + 1)); input_a.setBounds(120, 20, 100, 20);
150      add(input_b = new TextField("" + 0)); input_b.setBounds(270, 20, 100, 20);
151      add(input_c = new TextField("" + 0)); input_c.setBounds(120, 60, 100, 20);
152      add(input_d = new TextField("" + 1)); input_d.setBounds(270, 60, 100, 20);
153      // 再計算ボタン
154      startB = new Button("Restart");
155      add(startB);
156      startB.setBounds(420, 45, 50, 30);
157      startB.addActionListener(this);
158
159      // キャンバス
160      gc = new GraphCanvas();
161      add(gc);
```

```
162       gc.setBounds(50, 100, 400, 400);
163       ReadFields();
164       gc.compute(a, b, c, d);
165     }
166
167     // ボタンを押されたら、テキスト・フィールドの内容を読み取って、再描画
168     public void actionPerformed(ActionEvent e) {
169       if (e.getSource() == startB) {
170         ReadFields();
171         gc.compute(a, b, c, d);
172       }
173     }
174   }
175
```


アプレットビューア: duffing1.class
アプレット
a=
1
b=
0
c=
0
d=
1
Restart
アプレットが開始されました。

5.6 ローレンツ方程式

$$\begin{cases} \dfrac{dx}{dt} = -\sigma x + \sigma y \\ \dfrac{dy}{dt} = rx - y - xz \\ \dfrac{dz}{dt} = -bz + xy \end{cases}$$

この微分方程式の不動点は $(x, y, z) = (0, 0, 0),\ (x_0, x_0, z_0),\ (-x_0, -x_0, z_0)$ である。(ただし、$x_0 = \sqrt{b(r-1)},\ z_0 = r - 1$ とする) また、パラメータ $\sigma,\ r,\ b$ は正定数とする。

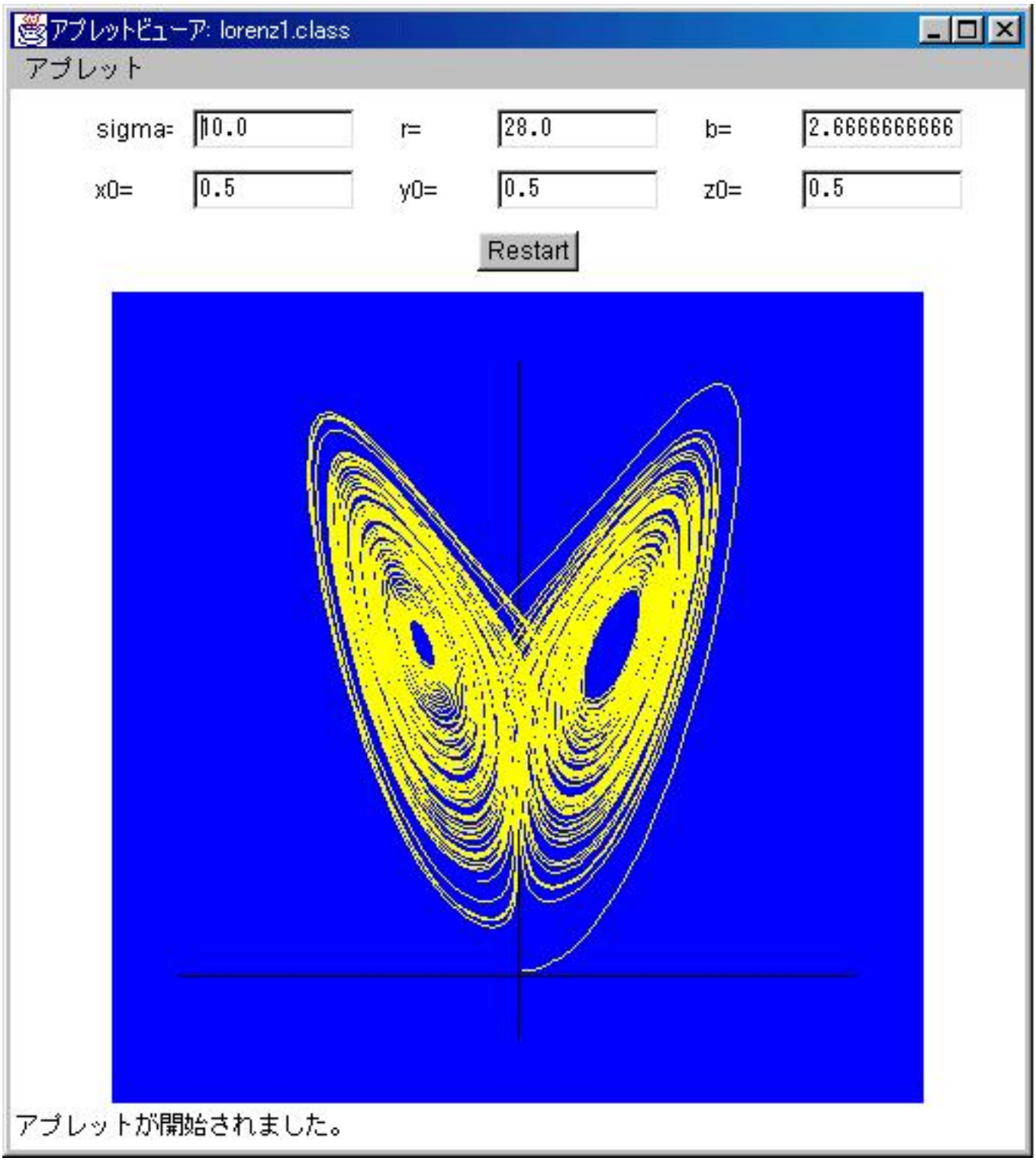
この結果は、$x - y$ 平面における解軌道であり、描画範囲は $-30 \leq x \leq 30,\ -5 \leq z \leq 50$ である。この微分方程式は、石村 [3] を参考にした。

```
 1  // <APPLET code="lorenz1.class" width=500 height=500> </APPLET>
 2  // lorenz
 3
 4  import java.applet.*;
 5  import java.awt.*;
 6  import java.awt.event.*;
 7
 8  class GraphCanvas extends Canvas {
 9
10      static final boolean DEBUG = false; // true;
11      private static final String message =
12          "graph of a function with 1 variable";
13      // 問題に取って基本的なパラメーター
14      // Lorenz 方程式の係数
15      private double sigma, r, b;
16      // 初期値
17      private double iv_x, iv_y, iv_z;
18      // 描画範囲
19      private double x_max =   30.0;
20      private double x_min = - 30.0;
21      private double x_margin = (x_max - x_min) / 10;
22      private double z_max =   50.0;
23      private double z_min = - 5.0;
24      private double z_margin = (z_max - z_min) / 10;
25
26      // 座標系の変換のためのパラメーター
27      private int CanvasX = 400, CanvasY = 400, CanvasZ = 400 ;
28      private double ratiox, ratioy, ratioz, X0, Y0, Z0;
29
30      // コンストラクター
31      public GraphCanvas() {
32          super();
33      }
34      public GraphCanvas(int cx, int cy, int cz) {
35          super();
36          CanvasX = cx; CanvasY = cy; CanvasZ = cz;
37      }
38
39      public void compute(double Sigma, double R, double B,
40                          double X0, double Y0, double Z0)
41      {
```

```
42          sigma = Sigma; r = R; b = B;
43          iv_x = X0; iv_y = Y0; iv_z = Z0;
44          repaint();
45      }
46
47      private boolean IsIn(double x, double z) {
48          return (x_min <= x && x <= x_max && z_min <= z && z <= z_max);
49      }
50      // 座標変換の準備
51      private void space(double x0, double z0, double x1, double z1) {
52          X0 = x0; Z0 = z0;
53          ratiox = CanvasX / (x1 - x0);
54          ratioz = CanvasZ / (z1 - z0);
55      }
56      // ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
57      private int wx(double x) {
58          return (int)(ratiox * (x - X0));
59      }
60      // ユーザー座標（ワールド座標系）をウィンドウ座標（デバイス座標系）
61      private int wz(double z) {
62          return CanvasZ - (int)(ratioz * (z - Z0));
63      }
64      // 力学系の右辺 f=(fx, fy, fz)
65      private double fx(double x, double y, double z) {
66          return - sigma * x + sigma * y;
67      }
68      private double fy(double x, double y, double z) {
69          return r*x - y - x * z;
70      }
71      private double fz(double x, double y, double z) {
72          return -b * z + x * y;
73      }
74      // x[], y[], z[] の内容をグラフにする
75      private void drawGraph(Graphics g,
76                             double x0, double y0, double z0, double T)
77      {
78          double h = 0.01;
79          if (T < 0.0)
80              h = - h;
81          int iter = (int)Math.rint(Math.abs(T / h));
82          double x = x0;
83          double y = y0;
84          double z = z0;
85          double new_x, new_y, new_z;
86          for (int i = 0; i <= iter; i++) {
87              double k1x = h * fx(x, y, z);
88              double k1y = h * fy(x, y, z);
89              double k1z = h * fz(x, y, z);
90              double k2x = h * fx(x + k1x / 2, y + k1y / 2, z + k1z / 2);
91              double k2y = h * fy(x + k1x / 2, y + k1y / 2, z + k1z / 2);
92              double k2z = h * fz(x + k1x / 2, y + k1y / 2, z + k1z / 2);
93              double k3x = h * fx(x + k2x / 2, y + k2y / 2, z + k2z / 2);
94              double k3y = h * fy(x + k2x / 2, y + k2y / 2, z + k2z / 2);
95              double k3z = h * fz(x + k2x / 2, y + k2y / 2, z + k2z / 2);
96              double k4x = h * fx(x + k3x, y + k3y, z + k3z);
97              double k4y = h * fy(x + k3x, y + k3y, z + k3z);
98              double k4z = h * fz(x + k3x, y + k3y, z + k3z);
99              new_x = x + (k1x + 2 * k2x + 2 * k3x + k4x) / 6;
```

```
100             new_y = y + (k1y + 2 * k2y + 2 * k3y + k4y) / 6;
101             new_z = z + (k1z + 2 * k2z + 2 * k3z + k4z) / 6;
102             if (IsIn(x, z) && IsIn(new_x, new_z))
103                 g.drawLine(wx(x), wz(z), wx(new_x), wz(new_z));
104             x = new_x; y = new_y; z = new_z;
105         }
106     }
107     public void paint(Graphics g) {
108         //
109         space(x_min - x_margin, z_min - z_margin,
110               x_max + x_margin, z_max + z_margin);
111         //
112         setBackground(Color.blue);
113         //
114         g.setColor(Color.black);
115         g.drawLine(wx(x_min), wz(0.0), wx(x_max), wz(0.0));
116         g.drawLine(wx(0.0), wz(0.0), wx(0.0), wz(0.0));
117         g.drawLine(wx(0.0), wz(z_min), wx(0.0), wz(z_max));
118         //
119         g.setColor(Color.yellow);
120         //
121         double Time = 100.0;
122         drawGraph(g, iv_x, iv_y, iv_z, Time);
123     }
124 }
125
126 public class lorenz1 extends Applet implements ActionListener {
127     private int N = 20;
128     private double lambda = 0.5;
129     private double Tmax = 0.5;
130     // ユーザーとのインターフェイス（パラメーターの入力）
131     private Label label_sigma, label_r, label_b;
132     private Label label_x0, label_y0, label_z0;
133     private TextField input_sigma, input_r, input_b;
134     private TextField input_x0, input_y0, input_z0;
135     private double sigma, r, b;
136     // 初期値
137     private double x0, y0, z0;
138     private Button startB;
139     //
140     private GraphCanvas gc;
141
142     private void ReadFields() {
143         sigma = Double.valueOf(input_sigma.getText()).doubleValue();
144         r = Double.valueOf(input_r.getText()).doubleValue();
145         b = Double.valueOf(input_b.getText()).doubleValue();
146         x0 = Double.valueOf(input_x0.getText()).doubleValue();
147         y0 = Double.valueOf(input_y0.getText()).doubleValue();
148         z0 = Double.valueOf(input_z0.getText()).doubleValue();
149     }
150     // 準備（変数の用意と座標系の初期化など）
151     public void init() {
152         // ナル・レイアウト
153         setLayout(null);
154         // sigma, r, b を入力するためのテキスト・フィールド
155         add(label_sigma = new Label("sigma="));
156         label_sigma.setBounds(40, 10, 40, 20);
157         add(input_sigma = new TextField("" + 10.0));
```

```
158         input_sigma.setBounds(90, 10, 80, 20);
159
160         add(label_r = new Label("r="));
161         label_r.setBounds(190, 10, 40, 20);
162         add(input_r = new TextField("" + 28.0));
163         input_r.setBounds(240, 10, 80, 20);
164
165         add(label_b = new Label("b="));
166         label_b.setBounds(340, 10, 40, 20);
167         add(input_b = new TextField("" + 8.0/3.0));
168         input_b.setBounds(390, 10, 80, 20);
169
170         // 初期値 x0, y0, z0 を入力するためのテキスト・フィールド
171         add(label_x0 = new Label("x0="));
172         label_x0.setBounds(40, 40, 40, 20);
173         add(input_x0 = new TextField("" + 0.5));
174         input_x0.setBounds(90, 40, 80, 20);
175
176         add(label_y0 = new Label("y0="));
177         label_y0.setBounds(190, 40, 40, 20);
178         add(input_y0 = new TextField("" + 0.5));
179         input_y0.setBounds(240, 40, 80, 20);
180
181         add(label_z0 = new Label("z0="));
182         label_z0.setBounds(340, 40, 40, 20);
183         add(input_z0 = new TextField("" + 0.5));
184         input_z0.setBounds(390, 40, 80, 20);
185
186         // 再計算ボタン
187         startB = new Button("Restart");
188         add(startB);
189         startB.setBounds(230, 70, 50, 20);
190         startB.addActionListener(this);
191
192         // キャンバス
193         gc = new GraphCanvas();
194         add(gc);
195         gc.setBounds(50, 100, 400, 400);
196         ReadFields();
197         gc.compute(sigma, r, b, x0, y0, z0);
198     }
199
200     // ボタンを押されたら、テキスト・フィールドの内容を読み取って、再描画
201     public void actionPerformed(ActionEvent e) {
202         if (e.getSource() == startB) {
203             ReadFields();
204             gc.compute(sigma, r, b, x0, y0, z0);
205         }
206     }
207 }
```


アプレットビューア: lorenz1.class
アプレット
sigma=
10.0
r=
28.0
b=
2.6666666666
x0=
0.5
y0=
0.5
z0=
0.5
Restart
アプレットが開始されました。

関連図書

[1] 高橋　陽一郎，微分方程式入門，東京大学出版会　（１９８８）

[2] 笠原　晧史，微分方程式の基礎，朝倉書店　（１９８２）

[3] 石村　直之，パワーアップ微分方程式，共立出版　（２００１）

[4] 丹波　敏雄，微分方程式と力学系の理論入門，遊星者（１９８８）

[5] 吉田　善章，非線形科学入門，岩波書店　（１９９８）

[6] 伊藤　秀一，常微分方程式と解析力学，共立出版　（１９９８）

[7] ニコラス・Ｂ・トゥフィラーロ，非線形動力学とカオス，アジソン・ウェスレイ・パブリッシャーズ・ジャパン（１９９４）

[8] マグローヒル，数学用語辞典，日刊工業新聞社　（２００１）