

分岐図 by gnuplot with image

2023/06/19, 20, 25, 30 山田 泰司 <taiji@aihara.co.jp>

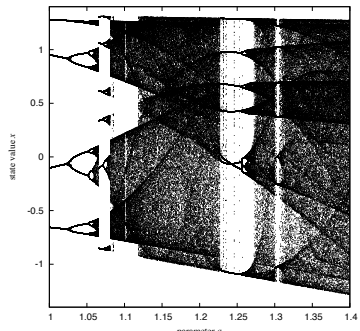
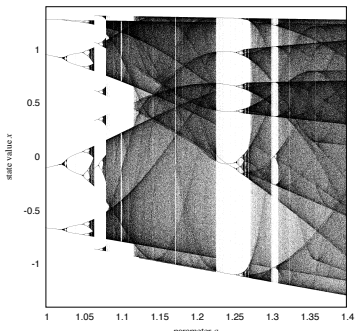
はじめに

gnuplot で分岐図を書くとき、普通に plot コマンドを用いるとファイルサイズが大きくなりがちで、ゆえに解像度を上げることが困難になる。本稿ではスケーラブル・ベクタ形式 (EPS, PDF, SVG) のなかにラスタ形式の画像を埋め込む plot with image コマンドによる方法を紹介する。

ファイルサイズの比較

まずは結論から、EPS, PDF, SVG 形式のファイルサイズの比較を示す。

表 1 内部形式によるファイルサイズの比較

\ 分岐図の内部形式	ベクタ形式 — vbirfucation	ラスタ形式 — pbifurcation
パラメータ毎の点数	500	2000
パラメータの分解能	500	2000
EPS ファイルのサイズ (kB)	3436	492
PDF ファイルのサイズ (kB)	4160	320
SVG ファイルのサイズ (kB)	15800	900
挿入図 (エノン写像)		

このように EPS 内ラスタ形式では 4^2 倍の画素数でもファイルサイズが 1 割程度に収まる。よって、ラスタ形式であればさらに解像度を上げることが容易である。ちなみに、SVG は cairo ライブラリを gnuplot 内で利用しており、重量級の図には適さないと思われる。

また、gnuplot にて `set terminal eps` とすると cairo ライブラリを利用した EPS となり (epscairo)、設計上、これも効率的な PostScript とはなっておらず、重量級の図には適さない。よって、`set terminal postscript eps level3` とするとよい¹。

データファイルの作成方法

しかし、ラスタ形式では生成しておくデータ形式にひと工夫必要であり、ベクタ形式のファイルよりも往々にしてサイズが大きくなる。そして、分岐図の場合、プログラム内でヒストグラム計算が必要となる。詳しくは文献 [2] のソースコードで事足りるが、以下に一部を紹介する。

```
std::vector<double> map(const std::vector<double> v0)
{
    // Hénon map
    std::vector<double> v1(v0.size());
    x1 = y + 1 - a*x*x;
    y1 = b*x;
    return v1;
}
int main(int argc, char *argv[])
{
    : // 中略

    switch (t_output_format) {
    case pixels_output_format: // plot with image 向けの出力
        for (int i = 0; i < n_bifurcation; ++i) {
            std::vector<size_t> bins(n_resolution, 0);
            v0 = parameters.v;
            a = a0 + (a1 - a0)*i/n_bifurcation;
            for (int j = 0; j < n_transient; ++j)
                v0 = map(v0);
            for (int j = 0; j < n_iteration; ++j) {
                v1 = map(v0);
                if (!(v1[0] < lb || ub < v1[0]))
                    ++bins[size_t((v1[0] - lb)*n_resolution/(ub - lb))];
                v0 = v1;
            }
            for (size_t j = 0; j < size_t(n_resolution); ++j)
                std::cout << a << '\t' << (ub - lb)*j/n_resolution + lb <<
                    '\t' << (bins[j] ? 1 : 0) << std::endl;
        }
    }

    : // 略
}
```

```
break;
default: // plot with dots 向けの出力
    for (int i = 0; i < n_bifurcation; ++i) {
        v0 = parameters.v;
        a = a0 + (a1 - a0)*i/n_bifurcation;
        for (int j = 0; j < n_transient; ++j)
            v0 = map(v0);
        for (int j = 0; j < n_iteration; ++j) {
            v1 = map(v0);
            std::cout << a << '\t' << v1[0] << std::endl;
            v0 = v1;
        }
    }
    break;
}

: // 略
```

参考文献

- [1] T. Williams & C. Kelley, “gnuplot 5.4 — An Interactive Plotting Program,” http://www.gnuplot.info/docs_5.4/gnuplot-ja.pdf, 2022.
- [2] T. Yamada, <https://www.aihara.co.jp/~taiji/lecture/bifurcations-gnuplot-20230619.tar.xz>, 2023.

¹ level3 としないと PostScript level1 となり内部のイメージデータが圧縮されないので、ベクタ形式の方式と同程度のファイルサイズとなる。

付録 出力ファイルの編集

特にベクタ形式にて plot with dots のスタイルを gnuplot では、筆者が調べた限りでは調整できないと思われる。

そこで、EPS 出力形式、及び、SVG 出力形式にて、丸ドットを角ドット及びサイズ調整するための perl ワンライナーを以下に紹介する

EPS ファイルの編集

```
set terminal postscript eps enhanced level3;
set palette gray negative;
unset colorbox;
set xlabel '{/Times-Roman parameter {/Times-Italic a}}';
set ylabel '{/Times-Roman state value {/Times-Italic x}}';
set size square;
set xrange [0:1]; set yrange [-1:1]; set zrange [0:1];
set mono;
plot 'filename.dat' with image notitle;
```

このように生成した EPS 出力ファイルにて、以下のようにすると丸ドットが角ドットになり、サイズが 10 倍になる。

```
perl -pe "s/\\Pnt {.*} def/\\Pnt {stroke gsave 0 setlinecap 10 setlinewidth M 10 0 V stroke grestore} def/" -i '$@'
```

SVG ファイルの編集

```
set terminal svg;
set xlabel '{/Times-Roman parameter {/:Italic a}}';
set ylabel '{/Times-Roman state value {/:Italic x}}';
set size square;
set mono;
plot 'filename.dat' with dots notitle;
```

このように生成した SVG 出力ファイルにて、以下のようにすると丸ドットが角ドットになり、サイズは 1 倍で、塗りつぶされる。

```
perl -pe "s/<circle id='gpDot' r='[^']*' (.*)\\/><rect id='gpDot' width='1' height='1' \\1 fill='currentColor'\\/>/" -i '$@'
```

ここで「\$@」は makefile 中のターゲットファイルである。いずれにしても、ページ記述言語 PostScript 3 及び W3C SVG 1.1 はオープン規格であるので、なんとかなるのである。一方で、PDF ファイルは内部でオブジェクト毎に圧縮されているので、以上のような編集は困難である。