

モノクロ・リカレンス by gnuplot with image

2023/06/19, 29 山田 泰司 <taiji@aihara.co.jp>

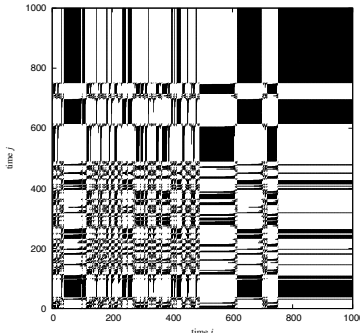
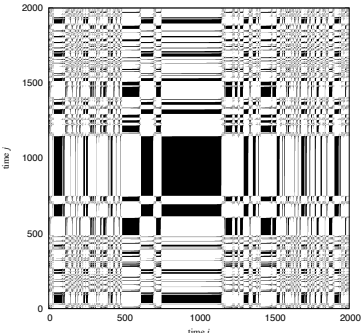
はじめに

gnuplot でモノクロ・リカレンスプロットを書くとき、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 — vbifurcation | ラスタ形式 — pbifurcation |
|-------------------|---|--|
| 時系列長 | 1000 | 2000 |
| EPS ファイルのサイズ (kB) | 3904 | 244 |
| PDF ファイルのサイズ (kB) | 6208 | 128 |
| SVG ファイルのサイズ (kB) | 18112 | 236 |
| 挿入図 (修正ベルヌーイ写像) |  |  |

このように EPS 内ラスタ形式では 2^2 倍の画素数でもファイルサイズが 1 割にも満たない。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は座標 (i, j) 昇順で普通に印字すればよく、近傍のみ印字したベクタ形式のファイルよりもサイズが大きくなる。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。

```
double map(const double theta0)
{
    // modified Bernoulli map
    return !(theta0 > 0.5) ?
        theta0 + pow(2, B-1)*(1 - 2*epsilon)*pow(theta0, B) + epsilon :
        theta0 - pow(2, B-1)*(1 - 2*epsilon)*pow(1-theta0, B) - epsilon;
}

int main(int argc, char *argv[])
{
    : // 中略
    switch (t_output_format) {
    case pixels_output_format:
    { // plot with image 向けの出力
        std::vector<double> V(n_iteration, 0);
        std::vector<std::vector<double>> > M(n_iteration,
            std::vector<double>(n_iteration, 0));
        theta0 = theta;
        for (int i = 0; i < n_transient; ++i)
            theta0 = map(theta0);
        for (int i = 0; i < n_iteration; ++i) {
            theta1 = map(theta0);
            V[i] = theta1;
            for (int j = 0; j < i; ++j) {
                const double d = std::sqrt((V[i] - V[j])*(V[i] - V[j]));
                M[i][j] = M[j][i] = d;
            }
            theta0 = theta1;
        }
    }
}
```

```
        for (int i = 0; i < n_iteration; ++i)
            for (int j = 0; j < n_iteration; ++j)
                std::cout << i << '\t' << j << '\t' <<
                    (M[i][j] < threshold ? 1 : 0) << std::endl;
    }
    break;
default:
    { // plot with dots 向けの出力
        std::vector<double> V(n_iteration, 0);
        theta0 = theta;
        for (int i = 0; i < n_transient; ++i)
            theta0 = map(theta0);
        for (int i = 0; i < n_iteration; ++i) {
            theta1 = map(theta0);
            V[i] = theta1;
            for (int j = 0; j < i; ++j) {
                const double d = std::sqrt((V[i] - V[j])*(V[i] - V[j]));
                if (d < threshold) {
                    std::cout << i << '\t' << j << '\t' << 1 << std::endl;
                    std::cout << j << '\t' << i << '\t' << 1 << std::endl;
                }
            }
            theta0 = theta1;
        }
    }
    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/recurrences-mono-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 time {/Times-Italic i}}';
set ylabel '{/Times-Roman time {/Times-Italic j}}';
set size square;
set mono;
plot 'filename.dat' with dots 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 5 倍になる。

```
perl -pe "s/\\Pnt {.*} def/\\Pnt {stroke gsave 0 setlinecap 5 setlinewidth M 5 0 V stroke grestore} def/" -i '$@'
```

SVG ファイルの編集

```
set terminal svg;
set palette gray negative;
unset colorbox;
set xlabel '{/Times-Roman time {/:Italic i}}';
set ylabel '{/Times-Roman time {/:Italic j}}';
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 0.5 倍で、塗りつぶされる。

```
perl -pe "s/<circle id='gpDot' r='[^']*' (.*)\\/><rect id='gpDot' width='0.5' height='0.5' \\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、なんとかできるのである。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。