自主課題研究レポート

偏角関数を用いた曲線の研究

008 岩淵 勇樹

担当教員: 佐藤 卓治教員

2006 年 1 月 27 日

1　偏角関数とは

直交座標平面上に与えられた任意の連続な曲線について、次のような関数を定義する。

定義:　$\theta(l)$ を曲線の基準点 (原点にあるとする) からの長さ l での接線の偏角 (x 軸に対する角度 (rad)) とし、これを偏角関数と呼ぶ。ただし、基準点が端点でない場合は一方の開曲線を $+l$ 方向、もう一方の開曲線を $-l$ 方向と適当に定め、それぞれの接線方向について偏角を求めるものとする。

このとき、元の曲線は次のようにして表される。

$$y = \int_0^t \sin\theta(l)dl$$
$$x = \int_0^t \cos\theta(l)dl$$
$$(-L_1 \leq t \leq L_2)$$

なお、曲線の長さを L とおくと、 $L = L_1 + L_2$ となる。また、閉曲線の場合は $L_1 = L_2 = \infty$ としてもよい。

1.1　性質

- $\theta'(l)$ は曲線の基準点からの長さ l での曲率を示す
- $\theta(l) + \theta$ ⇔ 曲線の θ (rad) 回転
- $\theta\left(\frac{l}{k}\right)$ ⇔ 曲線の k 倍拡大
- $\theta(l)$ が奇関数 ⇔ 曲線は線対称図形
- $\theta(l)$ が偶関数 ⇔ 曲線は点対称図形
- $\theta(l)$ が周期関数 ⇔ 曲線は繰り返しパターン (または閉曲線)

2　離散データとしての適用

上述のように定義された偏角関数は、代数的な処理を施すと原始関数で表現できない式がしばしば現れるため、標本化して取り扱うと便利である。標本化間隔を Δl とおいたとき、元の曲線は次のように復元される。

$$y = \sum_{k=0}^{t/\Delta l} \sin\theta(k\Delta l)\Delta l$$
$$x = \sum_{k=0}^{t/\Delta l} \cos\theta(k\Delta l)\Delta l$$
$$(-L_1 \leq t \leq L_2)$$

しかし当然のことながら、これは元の曲線を完全に復元するものではなく、 Δl を大きくとることによって復元できる曲線は粗くなる。なお、この表現は曲線を長さ Δl の線分の集合で近似したものに等しい。

3　偏角関数を応用した再帰曲線の描画

再帰曲線とは、元となる曲線に、ある決まった平行移動・拡大・回転といった幾何学変換を複数回施した曲線である。また、その変換を無限回繰り返した曲線は、その一部が全体と相似関係にある (=自己相似的な) 図形となり、これはフラクタルと呼ばれる。この節では、再帰曲線を偏角関数で表すことによって、その性質や再帰曲線描画への応用を考える。

3.1　コッホ曲線の描画

一般的に再帰曲線は端点の位置を一定に保って描かれるが、ここでは便宜上、一辺の長さを 1 と定める（ここで考える再帰曲線は、長さの等しい線分が連結された曲線に限る）。それによって、偏角関数を $\theta(l) = t_{\lceil i \rceil}$ とあらわせるような数列 $\{t_1, t_2, \ldots, t_k\}$ として表現できる。

3.1.1　コッホ曲線の描画方法（一般的な方法）

第 0 段階　水平な線分を第 0 段階とする。

第 1 段階　第 0 段階の線分を 3 等分し、真ん中の線分を一辺とする正三角形を描き、下の辺を消去する。

第 n 段階　第 $n-1$ 段階の曲線における各辺を第 0 段階とみなし、それぞれについて第 1 段階における処理を施す。

3.1.2　コッホ曲線の偏角関数

第 n 段階での曲線を I_n 、それを 4 等分した曲線をそれぞれ $I_{n1}, I_{n2}, I_{n3}, I_{n4}$ とあらわすと、I_{n1}, I_{n4} はそれぞれ I_{n-1} と同一であり、また、 I_{n2} は I_{n-1} を $\frac{\pi}{3}$ 回転したもの、 I_{n3} は I_{n-1} を $-\frac{\pi}{3}$ 回転したものとなる。すなわち、第 1.1 節で述べた性質より、 I_{n2} 部分における偏角は I_{n-1} 部分における偏角に $\frac{\pi}{3}$ 加算したもの、 I_{n3} 部分における偏角は I_{n-1} 部分における偏角に $-\frac{\pi}{3}$ 加算したものであることがわかる。（図 1 参照）

これより、コッホ曲線の第 n 段階における曲線の偏角関数は、次のような数列 t_k^n によって再帰的に求められる。

$$t_k^1 = \left\{0, \frac{\pi}{3}, -\frac{\pi}{3}, 0\right\}$$

$$t_k^n = t_{\lceil \frac{k}{4} \rceil}^{n-1} + t_{(k \bmod 4)+1}^1 \, (n \geq 2)$$

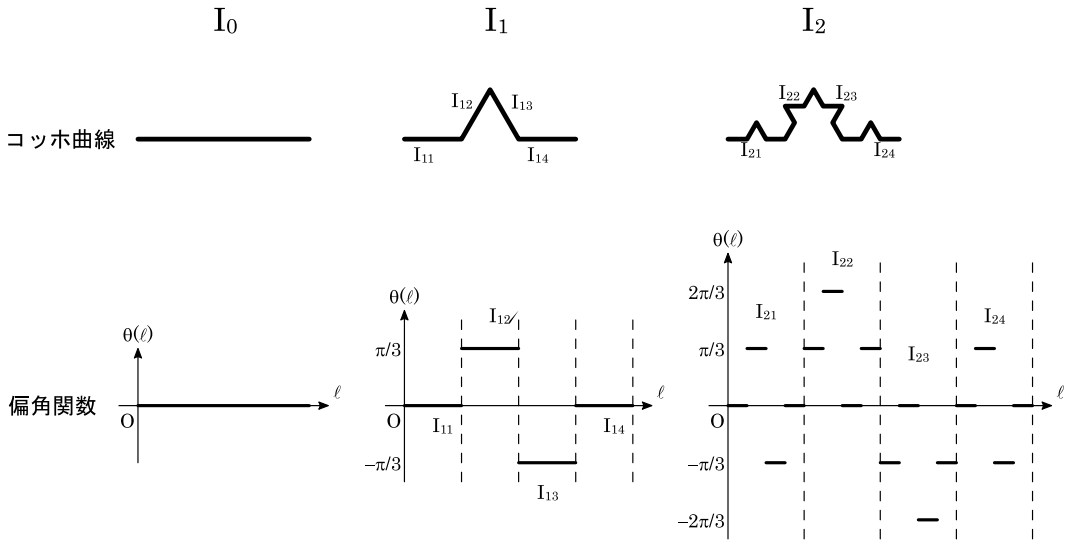


図 1: 各段階におけるコッホ曲線とその偏角関数

3.2 他の再帰曲線の描画

上述のコッホ曲線描画方法を拡張し、第 1 段階をジェネレーターとしてさまざまな数列を与えると、多種多様な曲線が得られる。その Mathematica によるプログラムと実行結果を以下に示す。

In[1]:=

```
ジェネレーターの定義;
コッホ曲線;
genekoch = {0, π/3, -π/3, 0};
クロス;
genecross = {0, π/2, 0, -π/2, 0};
レヴィのC曲線;
genec = {0, π/2};
ペアノ曲線の一種;
genepeano = {0, π/2, 0, -π/2, π, -π/2, 0, π/2, 0};
名称不明;
gene1 = {0, π/2, 0, -π/2, -π/2, 0, π/2, 0};
ミンコフスキー曲線;
geneminkowski = {0, π/2, 0};
シェルピンスキー曲線;
genesierpinski = {0, 2/3 π, 0, -2/3 π, 0};
名称不明(2次元);
gene2 = {π/3, -π/3, -π/3, π/3};
名称不明(2次元);
gene3 = {π/6, -π/2, π/6};
```

In[20]:=

```
ジェネレーターの選択と偏角関数列の定義;
Clear[f]
generator = genekoch
glen = Length[generator]
f[list_] := Flatten[Table[list + list[[k]], {k, 1, glen}]]
```

Out[22]=

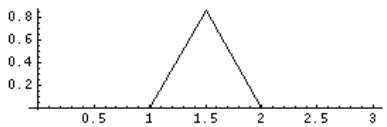
$$\left\{0, \frac{\pi}{3}, -\frac{\pi}{3}, 0\right\}$$

Out[23]=

4

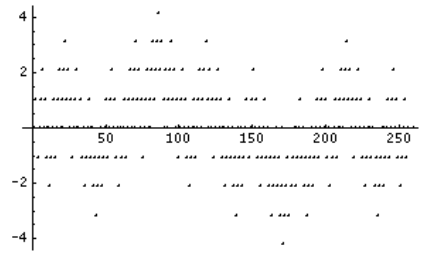
In[25]:=

```
ジェネレーターを座標列に変換してプロット;
Clear[gpx, gpy]
gpx = Table[0, {glen + 1}];
gpy = Table[0, {glen + 1}];
For[i = 1, i ≤ glen, i++,
 gpx[[i + 1]] = N[gpx[[i]] + Cos[generator[[i]]]];
 gpy[[i + 1]] = N[gpy[[i]] + Sin[generator[[i]]]]
]
ListPlot[Transpose[{gpx, gpy}], AspectRatio → Automatic, PlotJoined → True];
```



In[31]:=

```
偏角関数列の生成およびプロット；
repeat = 3;
inclinationlist = Nest[f, generator, repeat];
ListPlot[%];
```



In[35]:=

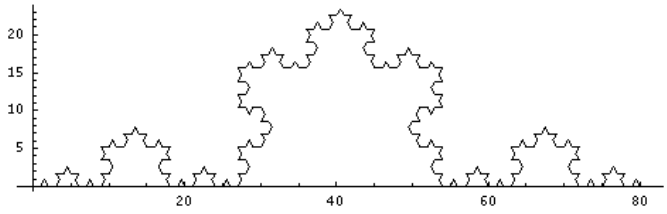
```
偏角関数列の座標列への変換;
Clear[px, py]
ilen = Length[inclinationlist]
px = Table[0, {ilen + 1}];
py = Table[0, {ilen + 1}];
For[i = 1, i ≤ ilen, i++,
 px[[i + 1]] = N[px[[i]] + Cos[inclinationlist[[i]]];
 py[[i + 1]] = N[py[[i]] + Sin[inclinationlist[[i]]]
]
```

Out[37]=

256

In[41]:=

```
生成したフラクタルのプロット；
ListPlot[Transpose[{px, py}], AspectRatio → Automatic, PlotJoined → True];
```



また、上プログラムにあるジェネレータをいろいろ変更させて、図 2 から図 10 の曲線を得た。

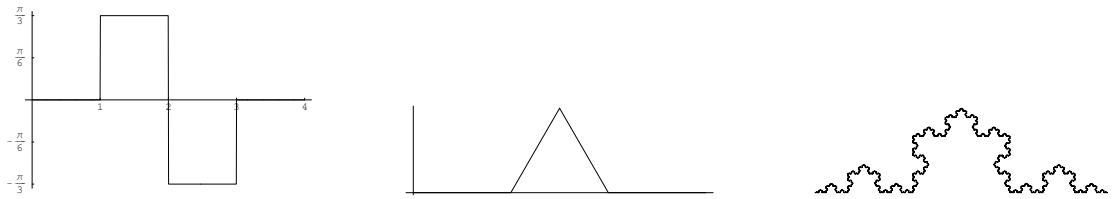


図 2: コッホ曲線

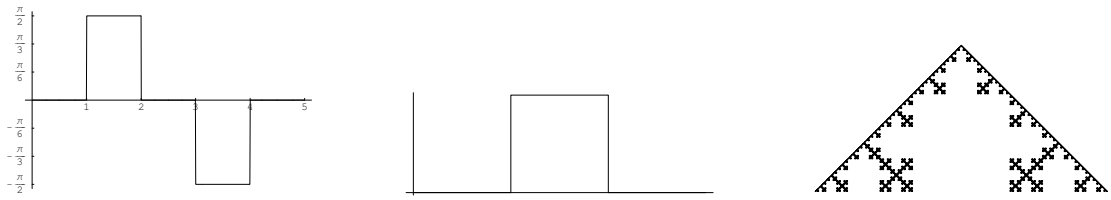


図 3: クロス

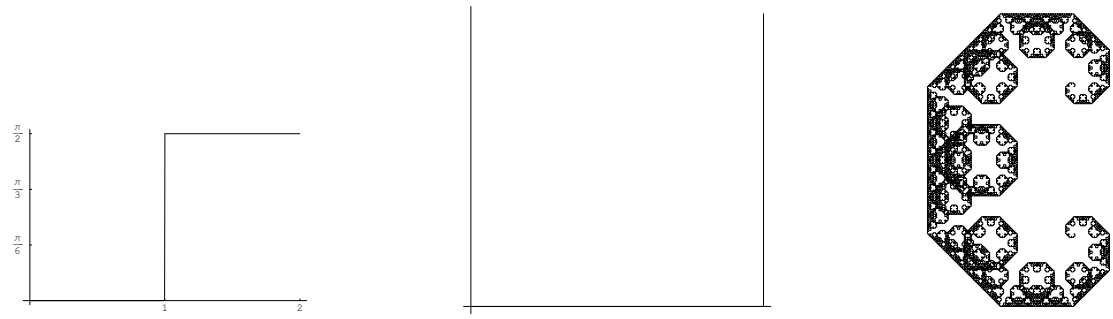


図 4: レヴィの C 曲線

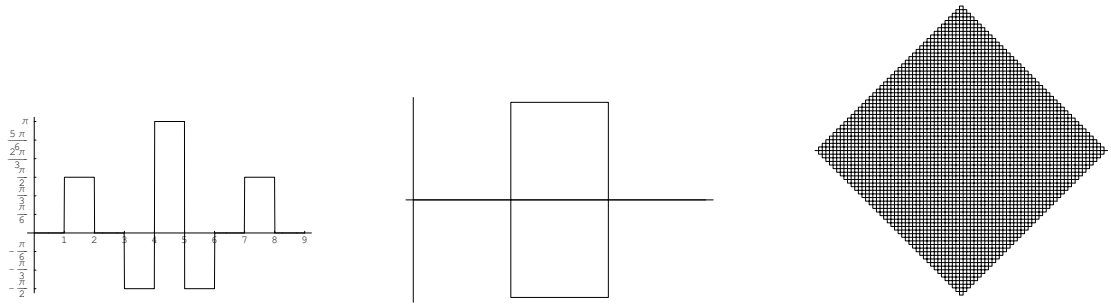


図 5: ペアノ曲線の一種

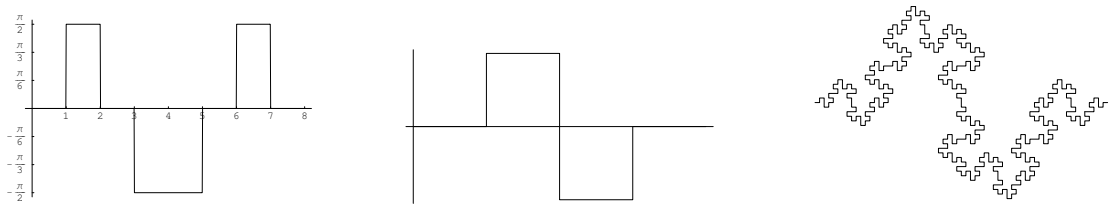


図 6: 一般呼称不明の曲線

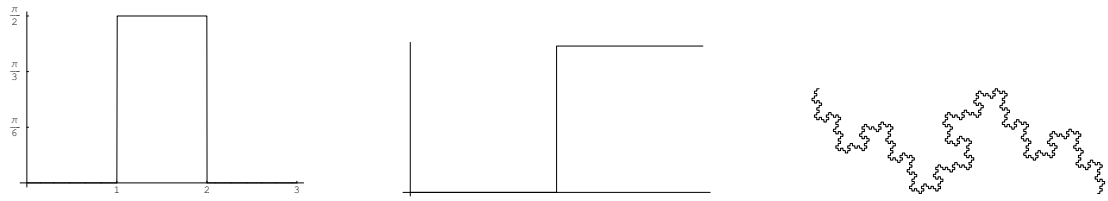


図 7: ミンコフスキー曲線

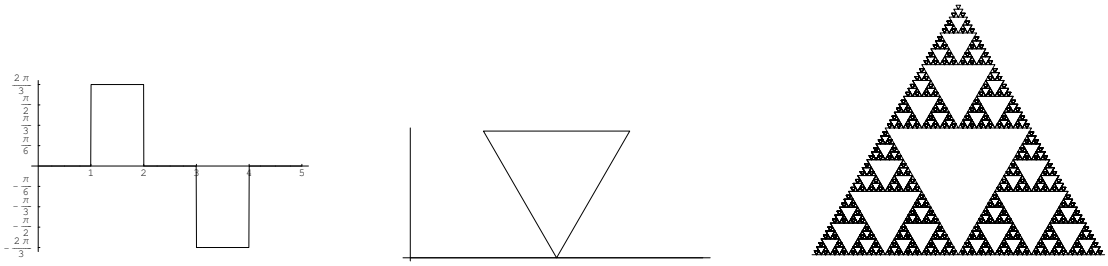


図 8: シェルピンスキー曲線

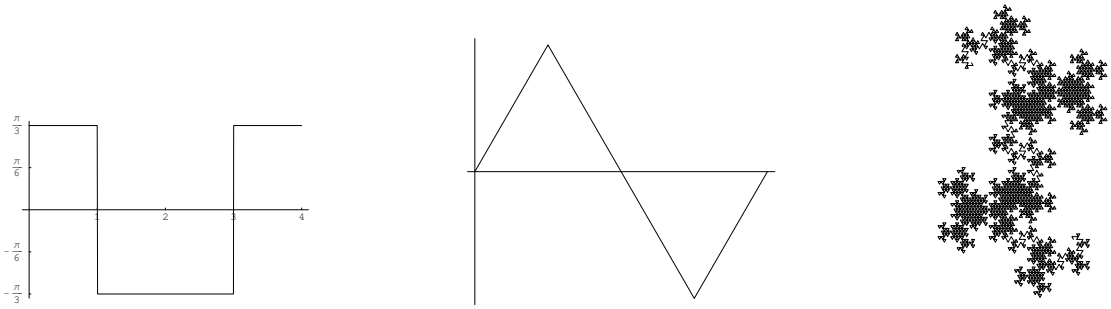


図 9: 一般呼称不明の 2 次元曲線

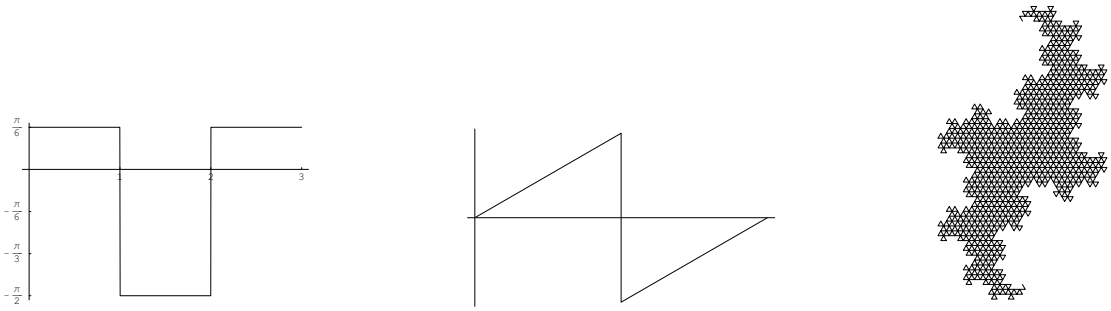


図 10: 一般呼称不明の 2 次元曲線

4　線画像の特徴抽出

この節では、ペンタブレットなどの入力装置から入力された曲線や、ディジタル画像から検出された曲線について解析する実践的なアルゴリズムを提案および実装する。

4.1　偏角関数への変換

4.1.1　入力される曲線

元となる曲線は、スプライン曲線やベジェ曲線によって記述されたものや、チェインコードによって記述されたものなどさまざまな形式が考えられるが、それらはその曲線中の点を細かく取って連結することによってほぼ再現できるため、曲線を座標列として入力する。

4.1.2　変換

入力される曲線は直線により近似されているものであるため、それを表す偏角関数は区間ごとに異なる一定値をもつような関数となる。なお、その区間は各座標間の距離であり、またその区間での値は 2 点を結ぶベクトルの偏角（ $-\pi \sim \pi$ とする）である。

つまり、n 点の座標列 $\{A_1, A_2, A_3, \ldots, A_n\}$ についてベクトル列 $\left\{\overrightarrow{A_1A_2}, \overrightarrow{A_2A_3}, \ldots, \overrightarrow{A_{n-1}A_n}\right\}$ を求め、それぞれ $\left\{|\overrightarrow{A_1A_2}|, |\overrightarrow{A_2A_3}|, \ldots, |\overrightarrow{A_{n-1}A_n}|\right\}$ の区間長と一定値 $\left\{\angle\overrightarrow{A_1A_2}, \angle\overrightarrow{A_2A_3}, \ldots, \angle\overrightarrow{A_{n-1}A_n}\right\}$ をもつ関数を求めればよい。

4.1.3　標本化、出力

第 2 節で述べたように、解析を容易にするために長さ軸を離散化することにする。また、今回は曲線の形状についての解析を主とするため、曲線の全長を揃えて出力することにする。出力は $-\pi \sim \pi$ の実数値をもつ N 個の点列となる。

4.1.4　実施内容

曲線の座標列は、ドロー系ツールで描いた曲線を、直線の集合に変換したうえで Adobe Illustrator (ai) 形式で書き出し、そのファイル内から該当する行を取り出すことによって、1 行中に x 座標と y 座標および制御文字 (m か L) がスペースに区切られたかたちで得られる。それを以下のようなプログラムに入力することによって、偏角関数が数列として出力される。なお、出力の点数は 256 点と定めた。

4.2　開曲線における解析・評価

ここでは、ペンタブレットからの入力などに想定される、始点と終点があらかじめ決まっている曲線について考える。

図 11: 教師曲線

図 12: 手書き曲線

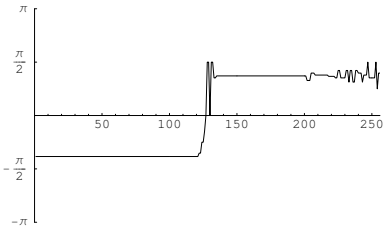


図 13: 教師曲線'v' の偏角関数

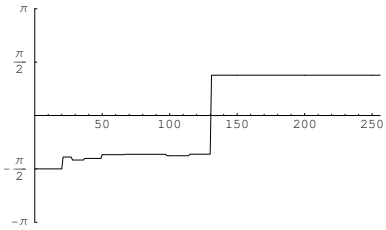


図 14: 手描き曲線'v' の偏角関数

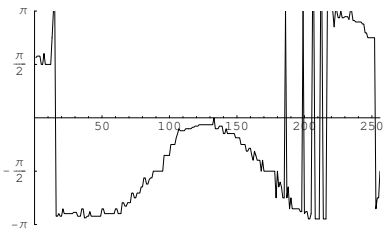


図 15: 教師曲線's' の偏角関数

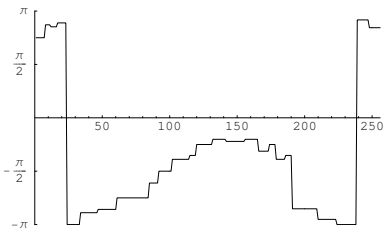


図 16: 手描き曲線's' の偏角関数

4.2.1 類似性の評価

ここでは、図 11 の教師曲線を基準の曲線として、図 12 の手書きの曲線が教師曲線と一致性が高いかどうかを調べる。

```
  len = Length[list];
  px = Table[0, {len + 1}];
  py = Table[0, {len + 1}];
  For[i = 1, i ≤ len, i++,
    px[[i + 1]] = N[px[[i]] + Cos[list[[i]] // n2r]];
    py[[i + 1]] = N[py[[i]] + Sin[list[[i]] // n2r]]
   ]
   ListPlot[Transpose[{px, py}], AspectRatio → Automatic, PlotJoined → True, Axes → None];
 }

対象とする文字のリスト
charalist = {"a", "b", "c", "d", "e", "g", "h", "n", "p", "q", "r", "s", "u", "v", "w", "z", "2", "3", "7", "9"}

偏角データの読み込み (教師データ)
cs[1] = imp["cs/a"];
cs[2] = imp["cs/b"];
cs[3] = imp["cs/c"];
...

偏角データの読み込み (手書きデータ)
fh[1] = imp["fh/a"];
fh[2] = imp["fh/b"];
fh[3] = imp["fh/c"];
...



2つのリストにおける誤差の2乗総和を求める
eval[list1_, list2_] := (Mod[list1 - list2, 0.5, -0.25]) ^2 // Total

教師 - 手書きデータのそれぞれの評価を数値で表示する
TableForm[Table[eval[fh[mm], cs[nn]], {mm, 1, 20}, {nn, 1, 20}]]

教師 - 手書きデータのそれぞれの評価を濃度図で表示する
ListDensityPlot[Table[eval[fh[mm], cs[nn]], {mm, 1, 20}, {nn, 1, 20}]];
```

一致性は数列の成分ごとに、正規化された角度の誤差を計算し、その数列全体での 2 乗和として評価する。この評価の値が低いほど類似性が高いと判別する。

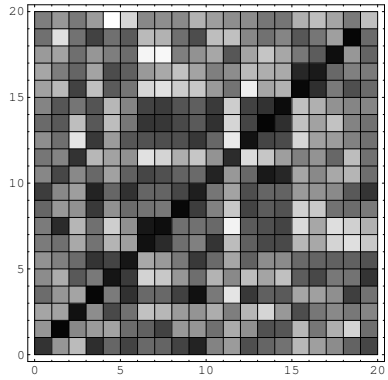


図 17: 教師曲線と手描き曲線の一致性

図 17 が、全教師曲線と全手描き曲線についてそれぞれ評価を行い、評価の値を濃度値として各座標に図 11 および図 12 に示した順にグリッド上に並べたものである。評価の値が低いほど黒くなるように描画しているが、対角線上、つまり、教師曲線と手描き曲線が一致する場合に評価の値が最も低くなっている。また、'h' と 'n'、'r' と 'v' など、人の目で見ても似ていると思える曲線同士でも類似性が高いという結果が見て取られる。これより、この方法による類似性の評価は信頼性が高いことがわかった。

4.2.2 対称性の評価

ここでは、図 11 の教師画像を用い、曲線が線対称や点対称といった性質をもつかを調べる。第 1.1 節で述べた、線対称曲線・点対称性曲線についての性質を利用して、上記のような式によって対称性を評価する。

```
 len = Length[list];
 px = Table[0, {len + 1}];
 py = Table[0, {len + 1}];
 For[i = 1, i ≤ len, i++,
   px[[i + 1]] = N[px[[i]] + Cos[list[[i]] // n2r]];
   py[[i + 1]] = N[py[[i]] + Sin[list[[i]] // n2r]]
  ]
  ListPlot[Transpose[{px, py}], AspectRatio → Automatic, PlotJoined → True, Axes → None];
}

対象とする文字のリスト
charalist = {"a", "b", "c", "d", "e", "g", "h", "n", "p", "q", "r", "s", "u", "v", "w", "z", "2", "3", "7", "9"}

偏角データの読み込み (教師データ)
cs[1] = imp["cs/a"];
cs[2] = imp["cs/b"];
cs[3] = imp["cs/c"];
...

偏角データの読み込み (手書きデータ)
fh[1] = imp["fh/a"];
fh[2] = imp["fh/b"];
fh[3] = imp["fh/c"];
...
```

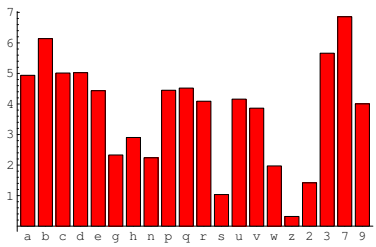


図 18: 点対称性の評価

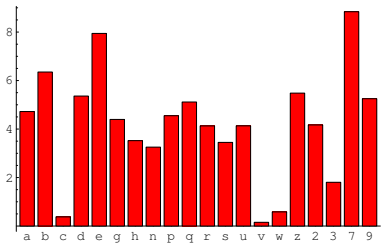


図 19: 線対称性の評価

点対称性の評価では、's'、'z' での評価の値がとりわけ低いことがわかる。また、線対称性の評価では、'c'、'v'、'w'、および'3' での評価の値がとりわけ低いことがわかる。これより、対称性の評価の信頼性が高いことがわかった。

5　考察

ディジタル写真などの保存形式として、ビットマップ (bmp)、JPEG、PNG などの”ラスタ系 (ビットマップ系、ペイント系)”と呼ばれる画像形式が一般的であるが、フォントにおける文字の形や手描きの CG などを保存する形式として”ベクタ系 (ドロー系)”と呼ばれる画像形式がある。今回研究した偏角関数は、微分が曲率であるが、曲率が高い部分ほど視覚は敏感、つまり情報量が高いということになる。その点で考えれば、偏角関数を利用することによってドロー系の画像形式における効率のよい圧縮方法を見出せるのではないかと考える。

また、第 4 節のように、偏角関数を解析することによってさまざまな特徴を知ることができるということがわかったため、これを文字や図形の判別や表情分析などに応用することも大いに可能ではないかと考える。今回は開曲線の特殊な状況での解析であったが、閉曲線や複数の曲線についての改良案も多数考えられ、更なる研究を行う価値はあると思われる。

しかし、偏角関数は単一曲線に対しては容易に考えられるが、複数の曲線に対しては位置関係の把握に弱い（たとえば曲線同士が近接しているかどうかを計算しにくい）ため、座標情報も活用するなどの工夫は随時必要であると考えられる。