数理解析研究所講究録
第1951巻 2015年 200-208

KETpicによる作図プログラミング書法の確立

木更津工業高等専門学校 基礎学系　山下　哲 (Satoshi Yamashita)
Faculty of Fundamental Research,
National Institute of Technology, Kisarazu College
工学院大学 基礎・教養教育部門　北原　清志 (Kiyoshi Kitahara)
Division of Liberal Arts,
Kogakuin University
長野工業高等専門学校 一般科　前田　善文 (Yoshifumi Maeda)
Faculty of General Education,
National Institute of Technology, Nagano College
群馬工業高等専門学校 一般教科（自然科学）　碓氷　久 (Hisashi Usui)
General Education (Natural Science),
National Institute of Technology, Gunma College
明治大学 総合数理学部　阿原　一志 (Kazushi Ahara)
School of Interdisciplinary Mathematical Sciences,
Meiji University
芝浦工業大学 工学部　牧下　英世 (Hideyo Makishita)
College of Engineering,
Shibaura Institute of Technology
東邦大学 理学部　高遠　節夫 (Setsuo Takato)
Faculty of Science,
Toho University

1　はじめに

KETpicは，LaTeXで図入り教材を作成するために，数式処理システム（Computer Algebra System，以下CAS）のマクロパッケージとして2006年に開発された[1]．使用できるCASはMaple，Mathematica，Maxima，Matlab，Scilab，Rであるが，現在，全機能を装備しているのはScilab版であり，Scialb版KETpicについてはマニュアル本が2011年に出版された[2]．KETpicによる図入りLaTeX文書の作成方法は，CASで作成した正確な線画データを挿図用LaTeXファイルへpicture環境とtpic specialsコードで出力し，文書用LaTeXファイルに¥inputコマンドで挿入するというものである（図1参照）．挿図に画像ファイルではなく，LaTeXファイル（テキストファイル）を使用するため，ファイルサイズが極めて小さく，ファイルのやり取りが容易である．KETpicの描画方法は，線画を基本とし，空間図形も稜線画法やスケルトン法（立体交差で奥にある線を消して奥行きを表現する方法）を駆使して描画できる（図2参照）．KETpicコマ

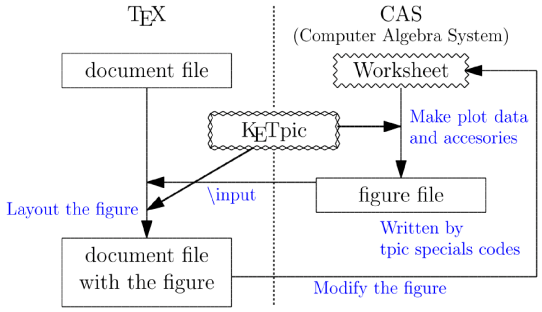

図 1. KETpic の利用方法

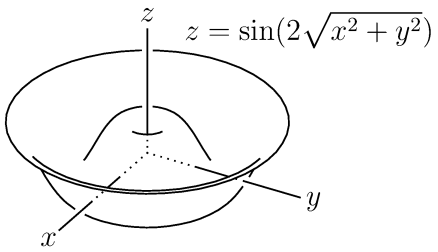

図 2. 空間図形の描画

ンドもシンボリックであるため，初心者でもプログラムを見るだけでどのように作図しているか理解できる．この性質を Symbolic Thinking と名付け，2011 年以降，作図におけるその重要性を提唱してきた [3]．また，描画機能だけでなく，作表機能，LaTeX マクロ作成機能，ページレイアウト機能も 2011 年に装備され，図入り教材を作成するための LaTeX 総合支援ツールとなった [4]．

このように KETpic の装備が拡張され，オリジナル図入り教材を作成する KETpic ユーザーが増えてきた．ユーザーが増えるにつれて，作図プログラムを共有する動きが始まったが，他のユーザーが作成した作図プログラムが転用しにくいという欠点が明らかになった．とくに初心者は独自の書き方でプログラミングするため，プログラムの読みやすさに注意を払わない．そこで，KETpic による作図プログラミング書法を確立する必要が生じ，読みやすい作図プログラミング書法の要件を 2013 年から調査してきた [5]．本論文では，この調査結果を報告し，KETpic による作図プログラミング書法の要件を確立する．

2 KETpic プログラムの特徴

本節では，図 2 を作成する KETpic プログラム例を挙げて，KETpic プログラムの特徴について言及する．Scilab 版 KETpic を用いて作成した図 2 の作図プログラムは以下の通りである．

```
Ketlib=lib("c:/work/ketpicsciL5");
cd("c:/work");
Ketinit();
Fname="surface.tex";

Setangle(60,30);
Fd=list("z=sin(2*sqrt(abs(x^2+y^2)))","x=R*cos(T)",...
        "y=R*sin(T)","R=[0,4]","T=[0,2*%pi]","e");
S=Sfbdparadata(Fd);
Ax=Xyzax3data("x=[0,5]","y=[0,5]","z=[0,4]");
Axo=Crvsfparadata(Ax,S,Fd);
```

```
12  Axi=CrvsfHiddenData();
13
14  Setwindow([-5,5],[-2.5,4]);
15  Ps=Skeletonparadata(S,list(Axo));
16  Paxo=Projpara(Axo);
17  Paxi=Projpara(Axi)
18  Windisp(Ps,Paxo,"c")
19
20  Openfile(Fname,"0.5cm");
21  Drwline(Ps,Paxo);
22  Dottedline(Paxi);
23  Xyzaxparaname(Ax);
24  Expr([1,4],"se","z=\sin(2\sqrt{x^2+y^2})");
25  Closefile("0");
```

上記プログラムは大きく3つの部分で構成されている．第1の部分は第1行目から第4行目までで**プリアンブル部**と呼ばれている．プリアンブル部では，作図プログラムに必要な環境を設定する．第1行目でKETpicライブラリをScilabへ読み込み，第2行目で作業用フォルダworkを指定し，第3行目でKETpicを初期化し，第4行目で作成する図ファイルの名前を定義する．

第2の部分は第6行目から第18行目までで**プロットデータ生成部**と呼ばれている．空間図形の作図ではさらに2つに分かれ，第6行目から第12行目までを**3Dデータ生成部**，第14行目から第18行目までを**投影2Dデータ生成部**という．平面図形の作図ではプロットデータ生成部が2つに分かれることはない．3Dデータ生成部では，図2の空間図形の3Dプロットデータを生成する．6行目は投影方向（視点の位置）を定義する．7行目から8行目で曲面の関数Fdを定義し，9行目で曲面の稜線の3DプロットデータSを生成する．第10行目でxyz座標軸の3DプロットデータAxを生成し，第11行目で曲面外の座標軸データの部分Axoを抽出し，第12行目で曲面に隠された座標軸データの部分Axiを抽出する．投影2Dデータ生成部では，上記で生成した3Dプロットデータを第6行目で定義した方向に投影する．第14行目で投影面上の描画範囲を定義する．第15行目で曲面の稜線Sに曲面外の座標軸データAxoとのスケルトン法を用いて投影2DデータPsを生成し，第16行目で曲面外の座標軸データの投影2DデータPaxo，第17行目で曲面に隠された座標軸データの投影2DデータPaxiをそれぞれ生成する．第18行目で生成した投影2DデータPs, PaxoをScilabの画面に表示する．

第3の部分は第20行目から最後までで**図ファイル書き出し部**と呼ばれている．プロットデータ生成部で生成された2Dプロットデータを描画コードに変換し，図ファイルに書き出す．第20行目では，第4行目で定義された図ファイルを開き，単位長を0.5cmとする．第21行目で曲面の稜線データPsと曲面外の座標軸データPaxoを実線で書き，第22行目で曲面に隠された座標軸データPaxiを点線で書く．第23行目で座標軸の名前x, y, zを書き入れ，第24行目で2変数関数名$z=\sin(2\sqrt{x^2+y^2})$を点$(1,4)$の南東方向に書く．第25行目で2D座標軸を書かずに図ファイルを閉じる．

以上の説明からわかるように，KETpicプログラムに使用されるコマンドはいずれもシンボリックであるため，各行を見るだけでどの部分を作図しているか初心者でも理解

できる．そのため，数学的な作図手順に従ってプログラムを書くことが重要である．また，プログラムの構成が見えるようにするために，第5, 13, 19行目に空白行を挿入しブロック化している．このような書き方の工夫がKETpicプログラムを読みやすくしている．次節では，KETpicプログラムを読みやすくするための作図プログラミング書法の要件を紹介する．

3 KETpicによる作図プログラムの調査

2010年以降，KETpicプログラムが他の作図ツールプログラムよりも読みやすく，プログラミングよりも作図そのものに集中できることを提唱してきた．つまり，

「数学的な作図手順に従ってコマンドを実行する際に，図の全体像を認識しながら作図の質的な改良に集中できる」

という思考を**Symbolic Thinking**と名付けた．Symbolic Thinkingは作図プログラミングで重要な意味をもち，KETpicを使用することで実現されることを私たちは主張してきた．では，Symbolic Thinkingを実現するような良いKETpicプログラムとはどのようなものであるべきか．それは，他のユーザーに読みやすいようシンボリックに表記されていることである．そこで，私たちはKETpicによる多くの作図プログラムを調査することにより，ノンシンボリックな表記を見つけ出し，KETpicによる作図プログラミング書法の要件を列挙した．本節では，2013年に実施したこの調査について報告する．

調査対象は104個のKETpicによる作図プログラムであり，その内訳は，プログラミング初心者の数学教育者1名による51個とプログラマーである数学教育者1名による53個である．その結果，列挙された作図プログラミング書法の要件は以下の9つである．

基本5要件

(1) コマンドの適切な配置

(2) 変数やプロットデータの適切な名前

(3) CASの計算機能の利用

(4) KETpicコマンドの適切な使用

(5) プログラムのブロック化

応用4要件

(6) 基準点を用いた文字や数式の適切な配置

(7) リスト構造の適切な使用

(8) for文の利用

(9) ローカル変数の使用

以下，上記要件のいくつかについてプログラム例を示しながら説明する．

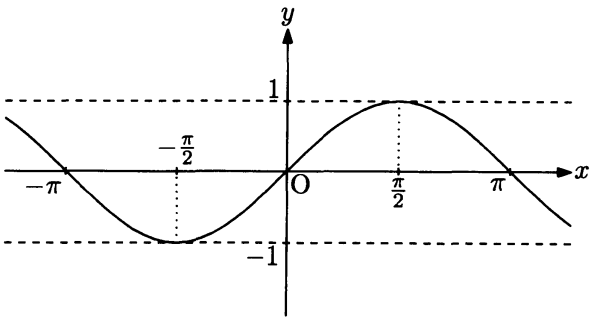

図 3. $y = \sin x$ のグラフ (1)

(1) コマンドの適切な配置

図 3 のプログラム

```
 6 G=Plotdata("sin(x)","x=[-4,4]");
...
11 Setwindow([-4,4],[-2,2]);
12 Windisp(G,UpL,LoL,CiL1,CiL2,"c")
```

において，ノンシンボリックな表記は，第 11 行目の画面範囲を定義するコマンド Setwindow である．画面範囲はプロットデータ生成部の最初に定義すべきであるから，シンボリックな表記に修正すると次のようになる．

```
 6 Setwindow([-4,4],[-2,2]);
 7 G=Plotdata("sin(x)","x=[-4,4]");
...
12 Windisp(G,UpL,LoL,CiL1,CiL2,"c")
```

(2) 変数やプロットデータの適切な名前

図 3 のプログラム

```
 8 L1=Listplot([Xmin(),1],[Xmax(),1]);
 9 L2=Listplot([Xmin(),-1],[Xmax(),-1]);
10 L3=Listplot([-%pi/2,0],[-%pi/2,-1]);
11 L4=Listplot([%pi/2,0],[%pi/2,1]);
12 Windisp(G,L1,L2,L3,L4,"c")
```

において，ノンシンボリックな表記は，4 本の線のプロットデータの名前 L1, L2, L3, L4 である．プロットデータの名前は，どの図形のプロットデータかわかるように上限線 (Upper Line) UpL, 下限線 (Lower Line) LoL, 座標指示線 (Coordinate Indicator Line) CiL1, CiL2 と名付けるべきであるから，シンボリックな表記に修正すると次のようになる．

```
 8 UpL=Listplot([Xmin(),1],[Xmax(),1]);
 9 LoL=Listplot([Xmin(),-1],[Xmax(),-1]);
```

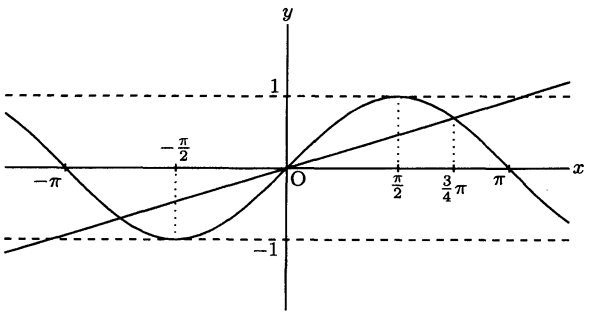

図 4. $y = \sin x$ のグラフ (2)

```
10 CiL1=Listplot([-%pi/2,0],[-%pi/2,-1]);
11 CiL2=Listplot([%pi/2,0],[%pi/2,1]);
12 Windisp(G,UpL,LoL,CiL1,CiL2,"c")
```

(3) CAS の計算機能の利用

図 4 のプログラム

```
12 L=Lineplot([0,0],[3/4*%pi,1/sqrt(2)]);
13 CiL3=Listplot([3/4*%pi,0],[3/4*%pi,1/sqrt(2)]);
14 Windisp(G,UpL,LoL,CiL1,CiL2,L,CiL3,"c")
```

において，ノンシンボリックな表記は，点の座標 `[3/4*%pi,1/sqrt(2)]`, `[3/4*%pi,0]` である．関数の値は CAS で計算できるから，シンボリックな表記に修正すると次のようになる．

```
12 X1=3/4*%pi;
13 P=[X1,sin(X1)];
14 L=Lineplot([0,0],P);
15 CiL3=Listplot([X1,0],P);
16 Windisp(G,UpL,LoL,CiL1,CiL2,L,CiL3,"c")
```

(6) 基準点を用いた文字や数式の適切な配置

図 3 のプログラム

```
14 Openfile(Fname,'1cm');
...
20 Htickmark(-%pi,"sw","-\pi",-%pi/2,"n","-\frac{\pi}{2}",...
    %pi/2,"\frac{\pi}{2}",%pi,"sw",'\pi');
21 Vtickmark(-1,"sw","-1",1,"nw","1");
22 Closefile("1");
```

において，第 20 行目と第 21 行目の座標軸上の目盛は基準点を用いた数値の適切な配置である．例えば，x 軸上の目盛 $-\pi$ の配置は第 20 行目の `Htickmark(-%pi,"sw","-\pi")` で定義される．第 1 引数 `-%pi` で x 軸上の基準点 $-\pi$ を指定し，第 2 引数 `"sw"` で南西

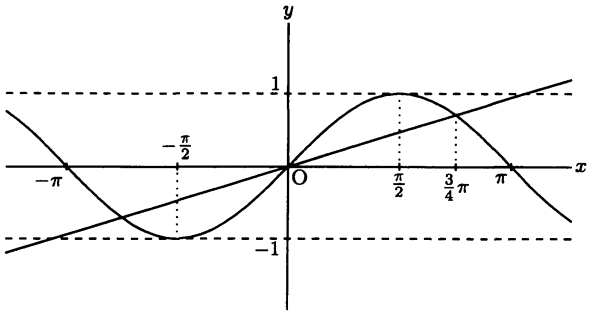

図 5. $y = \sin x$ のグラフ (3)

の方向を指定し，第3引数"-\pi") で基準点から指定された方向に書き出す数値 $-\pi$ を指定する．このように，基準点を用いると，文字や数式を適切に配置できる．

(7) リスト構造の適切な使用

(8) for 文の利用

要件 (7) と (8) は併用されることが多いから，同時に説明する．図5のプログラム

```
12 XL=[%pi/4,%pi/2,3/4*%pi];
13 LL=list();
14 CiLL=list();
15 for I=1:3
16   P=[XL(I),sin(XL(I))];
17   L=Lineplot([0,0],P);
18   CiL=Listplot([XL(I),0],P);
19   LL($+1)=L;
20   CiLL($+1)=CiL;
21 end
22 Windisp(G,UpL,LoL,CiL1,CiL2,LL(1),LL(2),LL(3),...
    CiLL(1),CiLL(2),CiLL(3),"c")
```

において，第13行目と第14行目および第19行目と第20行目がリスト構造の適切な使用である．例えば，第13行目で定義される空のリスト LL には，第17行目で定義される線のプロットデータ L が第19行目で格納される．このように，空のリストに使用するデータをまとめることができる．

第15行目から第21行目までが for 文であり，同じ作業を繰り返すことにより，3本の直線と座標指示線を同時に定義できる．第15行目で for 文で使用するローカル変数 I に1を代入し，第16行目で曲線上の点，第17行目で直線，第18行目で座標指示線をそれぞれ定義する．第19行目で LL に第1番目の直線のプロットデータを格納し，第20行目で CiLL に第1番目の座標指示線を格納する．I に1を加えて第16行目から第20行目までを繰り返す．I に3を代入して第16行目から第20行目までを実行すると，第21行目に移り，for 文を終了する．このように，for 文を用いると，同じ作業を繰り返し実行できる．

表1. ノンシンボリックな表記をもつプログラムの調査結果

requirements	beginner		programmer	
	bad/total	rate (%)	bad/total	rate (%)
(1) Arrange commands	26/51	51.0	9/53	17.0
(2) Suitable names	31/51	64.7	20/53	37.7
(3) Calculation of CAS	14/39	38.9	14/27	51.9
(4) KETpic commnads	37/51	72.6	29/53	54.7
(5) Readable blocks	19/51	37.3	0/53	0.0
(6) Reference points	32/51	62.8	11/50	22.0
(7) List structure	1/1	100.0	1/2	50.0
(8) for syntax	1/1	100.0	0/2	0.0
(9) Local variables	35/47	74.5	17/32	53.1

(9) ローカル変数の使用
図4のプログラム

```
12 X1=3/4*%pi;
13 P=[X1,sin(X1)];
14 L=Lineplot([0,0],P);
15 CiL3=Listplot([X1,0],P);
```

において，第12行目と第13行目がローカル変数の使用である．ローカル変数 X1, P を用いることで，第14行目と第15行目のプロットデータの定義が読みやすくなる．

以上のように，9つの要件についてノンシンボリックな表記があるプログラムを確認し，上記の表1を作成した．ただし，「bad/total」欄の数字はプログラムの個数を表し，total の数字は要件を満たすべきプログラムの個数である．表1からわかった事実をまとめると，以下にようになる．

- 数学教育者にとって，要件 (3)「CAS の計算機能の利用」を満たすことはそれほど難しくない．
- 要件 (4)「KETpic コマンドの適切な使用」の割合が高いことから，2名の数学教育者はいずれも KETpic について学習不足である．
- 要件 (5)「プログラムのブロック化」を満たすことは，プログラミング初心者でもたやすい．
- 応用4要件 (6)〜(9) を満たすことは，プログラミング初心者にとって難しい．
- プログラマーでさえ4要件を満たさないことから，プログラマーが読みやすいプログラムを書くとは限らない．

以上のことから，KETpic による作図プログラミング書法の確立が必須であるといえる．

4 まとめと今後の課題

第1節でKETpicおよびその利用方法（図1）を紹介し，第2節で図2の作図プログラム例を示してKETpicによる作図プログラムの特徴を紹介した．第3節では，2名の数学教育者が作成した104個のKETpicによる作図プログラムの中からノンシンボリックな表記を調査し，KETpicによる作図プログラミング書法の9つの要件を見出した．調査結果から，プログラミング初心者には応用4要件を満たすことは難しいことや，2名の数学教育者がKETpicについて学習不足であることなどが判明した．とくに，KETpicの学習不足については，KETpicマニュアルを端から端まで通読することが必須である（We should read KETpic manual from cover to cover）．

2014年9月以降，動的幾何システム（Interactive Geometry System，以下IGS）であるCinderellaを利用したKETCindyを開発している[6]．KETCindyは，CinderellaのCindyScriptを利用してScilabの作図実行ファイルを作成するというシステムである．作成されるScilabの作図プログラムは第3節に挙げた9つの要件を満たしており，KETpicによる作図プログラムの雛形といえる．つまり，KETpicによる作図プログラミング書法の雛形はKETCindyという形で実現された．

以上のことから，今後の課題は以下のようになる．

- KETpic+TEXによる図入り教材作成支援ポータルサイトを構築し，KETpicユーザーに役立つマニュアル・作図プログラミング書法のガイド・挿図教材事例集などを掲載する．なお，海外の研究者も閲覧できるよう英語版も作成する予定である．
- 図入り教材の作成について，数学教育者にインタビューやアンケートを実施し，KETpicの改良に役立てる．

参考文献

[1] 山下哲，関口昌由，高遠節夫：「Mapleによる図形描画用TEXファイルの作成について」，日本数学教育学会高専・大学部会論文誌，Vol.13, No1, pp.31–40, 2006.

[2] CASTEX応用研究会編：『KETpicで楽々TEXグラフ』，イーテキスト研究所，2011.

[3] S.Yamashita, S.Takato:「Making Materials and Mathematical Understanding Based on Symbolic Thinking」，京都大学数理解析研究所講究録1735「数式処理と教育」，pp.173–180, 2011.

[4] 山下哲，高遠節夫：「KETpicによる教材作成とSymbolic Thinking」，京都大学数理解析研究所講究録1780「数学ソフトウェアと教育」，pp.72–82, 2012.

[5] 山下哲，北原清志，前田善文，碓氷久，阿原一志，高遠節夫：「KETpicによる作図プログラミング書法について」，京都大学数理解析研究所講究録1909「数学ソフトウェアとその効果的教育利用に関する研究」，pp.1–7, 2014.

[6] KETCindyのwebページ：　https://sites.google.com/site/ketcindy/