KETCindy 関数リファレンス

KETCindy Project Team

2019 年 9 月 17 日

- 第 3.2 版 -

目次

1 平面の図形とグラフ 3
- 1.1 設定・定義 . . . 3
  - 1.1.1 環境設定 . . . 3
  - 1.1.2 描画設定・定義 . . . 5
- 1.2 描画 . . . 12
  - 1.2.1 書式とオプション . . . 12
  - 1.2.2 点・線分・直線 . . . 13
  - 1.2.3 曲線 . . . 27
  - 1.2.4 関数のグラフ . . . 39
  - 1.2.5 文字 . . . 47
  - 1.2.6 マーキング . . . 49
- 1.3 プロットデータの操作 . . . 53
- 1.4 計算 . . . 73
- 1.5 値の取得と入出力 . . . 74
- 1.6 作表 . . . 77
- 1.7 その他 . . . 84

2 他の数式処理ソフトなどとの連携 92
- 2.1 R との連携 . . . 92
- 2.2 Maxima との連携 . . . 100
- 2.3 Risa/Asir との連携 . . . 110
- 2.4 FriCAS(Axiom) との連携 . . . 110
- 2.5 MeshLab との連携 . . . 111
- 2.6 表計算ソフトとの連携 . . . 115

3 **アニメーション PDF** 119
3.1 概要 . . . 119
3.2 関数 . . . 120
3.3 制作例 . . . 121

4 KeT スライド 124
4.1 概要と制作手順 . . . 124
4.2 コンテンツファイル . . . 125
4.3 関数 . . . 131

5 KETCindy3D 134
5.1 概要 . . . 134
5.2 設定 . . . 135
5.3 描画 . . . 137
5.3.1 点・線 . . . 137
5.3.2 多面体 . . . 141
5.3.3 曲面 . . . 147
5.4 プロットデータの操作 . . . 155
5.5 その他 . . . 165

6 KeTJS 174
6.1 CindyJS と KeTJS . . . 174
6.2 KeTJS の動作環境 . . . 174
6.3 KeTJS の設定 . . . 176
6.4 KeTJS のコマンド . . . 177

7 **付録** 179
7.1 用語解説 . . . 179
7.2 Cinderella の作図ツール . . . 179
7.3 色名とカラーコード一覧 . . . 181
7.4 点の作図についての比較表 . . . 182

8 **関数一覧** 183

1　平面の図形とグラフ

1.1　設定・定義

1.1.1　環境設定

関数　Ketinit(options)

機能　KETCindy を初期化する。平面図形では Draw スロットに，空間図形では Initialization スロットの冒頭に記述する必要がある。

説明　option 作業サブフォルダ

通常は不要で，`Ketinit()` だけでよい。

関数　Initglist(), Setglist(), Addglist()

機能　ketlib スロットで作られる描画データを描画リストに追加する。

説明　Implicitplot,Hatchdata など実行時間のかかるコマンドを figures スロットにおくと，その都度実行されてしまう。それを避けるため ketlib スロットにおいたときに用いる。

```
Initglist(); // ketlibスロットで
Implicitplot(''1'',fun,rng);
Setglist();

Ketinit(); // figuresスロットで
Addglist();
```

関数　Setfiles(filename)

機能　出力するファイル名の設定

説明　出力する Tex のファイル名を指定する。

出力するファイル名は 初期設定では，作図している Cinderella のファイル名。

たとえば，triangle.cdy で作図して出力すると，triangle.tex ができる。

これに対し，triangle.cdy で作図しているときに，grav.tex で出力したい場合は

```
Setfiles("grav");
```

とすると，grav.tex ができる。

関数　Setparent(filename)

機能　Parent ボタンで出力するファイル名の設定

説明　Figpdf() を使って Parent ボタンで出力する Tex のファイル名を指定する。

Parent ボタンで出力するファイル名は 初期設定がないので，指定する必要がある。

たとえば，triangle.cdy で作図しているときに，図サイズの grav.pdf を作る場合，

```
Setparent("grav");
```
とすると，図の TeX ファイル triangle.tex と PDF を作る grav.tex ができ，ここから grav.pdf ができる。

関数 Changework(パス名)
機能 作業ディレクトリを指定（変更）する
説明 作業ディレクトリは，初期設定では，現在作図しているファイルのあるフォルダ（ディレクトリ）の fig フォルダである。これを変更する。

関数 Addpackage(パッケージ名)
機能 TeX のパッケージを追加する
説明 プレビュー用の TeX ソースにパッケージを追加する。

【例】emath パッケージを追加する。
```
Addpackage("emath");
```
または
```
Addpackage(["emath"]);
```
により，プレビュー用の TeX のプリアンブルに
```
\usepackage{emath}
```
が追加されて，emath のコマンドが利用できる。

注）初期設定では，次のパッケージを利用している。

```
ketpic, ketlayer, amsmath, amssymb, graphicx, color
```

関数 Usegraphics("pict2e")
機能 TeX のグラフィクスパッケージを"pict2e" に変更する
説明 デフォルトのパッケージは "tpic" であるが，これを"pict2e" に変更する。

⇒ 関数一覧

1.1.2 描画設定・定義

関数 Addax(0 または 1)

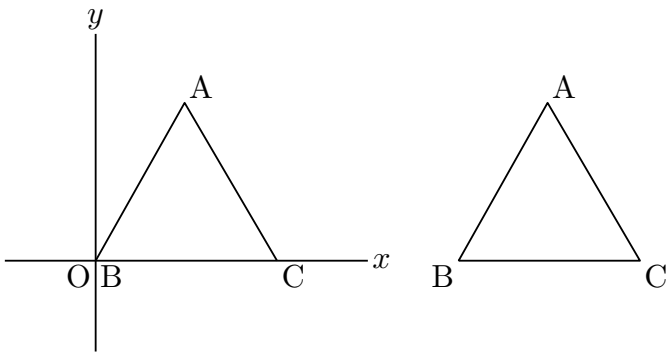
機能 座標軸の表示

説明 引数が 1 のとき座標軸を描き，0 のとき描かない。初期設定は 1 で，座標軸を描かない場合のみ `Addax(0)` とすればよい。

【例】三角形を描く

左図が 初期設定（座標軸表示）　Addax(0) をつけると右図になる。

```
Listplot([B,A,C]);
Letter([A,"ne","A",B,"se","B",C,"se","C"]);
```



関数 Setax()

機能 座標軸の書式を設定する。

説明 Cinderella の描画面には反映されない。（座標軸は描かれない）

引数はリストで与え，要素は順番に

1. 軸の形状（直線は ”l” ，矢印は ”a”） 初期設定は直線
   矢印の大きさの倍率を指定するときは，”a0.5”のようにする。
   また，矢印のスタイルは Setarrow で指定する。
2. 横軸名 初期設定は `"x"`
3. 横軸名の位置 初期設定は `"e"`
4. 縦軸名 初期設定は `"y"`
5. 縦軸名の位置 初期設定は `"n"`
6. 原点名 初期設定は `"O"`（文字として書かれる）
7. 原点名の位置 初期設定は `"sw"`
8. 線種
9. 線の色
10. ラベルの色

それぞれダブルクォートでくくる。色は，色名が使える。”red” など。

10 の引数のうち n 番目だけを指定する場合は，[n,”内容”] で指定できる。
また，後方は省略できる。

【例】座標軸の先端を矢印にし，原点の北西に O を書く。

```
Setax(["a","","","","","","nw"]);
```

【例】原点の北西に O を書く。

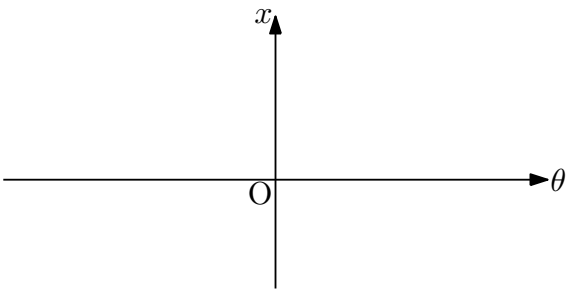
```
Setax([7,"nw"]);
```

【例】軸を赤の点線にする。

```
Setax(["","","","","","","","do","red"]);
```

【例】先端を矢印にし，横軸を θ，縦軸を x にして矢じりの左側に書く。

```
Setax(["a","\theta","","x","w"]);
```



関数 Drwxy(), Drwxy(options)

機能 指定する手順で座標軸を描く

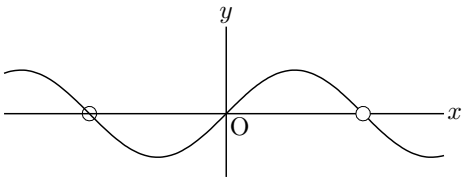
説明 座標軸は 初期設定では最後に描かれるが，座標軸上に白抜きの点を表示するなど，先に描くことが必要な場合に用いる。描画面には座標軸は表示されない。

options は次のリストである。

[”Origin=”, ”Xrng=”, ”Yrng=”]

【例】点 $(-\pi,\ 0)$ と $(\pi,\ 0)$ を白抜きの点で表示する。

```
Setax([7,"se"]);
Setpt(5);
Pointdata("1",[[-pi,0]],["Inside=0"]);
Drwxy();
Plotdata("1","sin(x)","x",["dr","Num=200"]);
Pointdata("2",[[pi,0]],["Inside=0"]);
```



このスクリプトでは， `Pointdata("1",[[-pi,0]],["Inside=0"]);` を実行したのち座標軸を描き，次に，$y = \sin x$ のグラフを描いてから，再び `["Inside=0"]);` にして実行するので，点 $(-\pi,0)$ の上を座標軸が通り，点 $(\pi,0)$ は座標軸とグラフの上を通るので白抜きになる。

⇒ 関数一覧

関数　Definecolor(色名 , 定義のリスト)

機能　色名を定義する

説明　ユーザー命名の色名を定義する。定義リストは RGB または CMYK のリスト
各色０～１の範囲で指定する。定義した色名は，Setcolor(color,options) で使うことができる。
なお，KETCindy では，68 色を色名で使うことができる。色の名称はカラーコード一覧 参照。

【例】暗い紫色を darkmaz の名称で定義して使う。

```
Definecolor("darkmaz",[0.8,0,0.8]);
Setcolor("darkmaz");
```

関数　Setcolor(color,options)

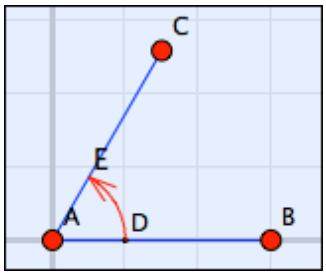
機能　描画色の設定

説明　引数 color はカラーコードまたは色の名称。
カラーコードは RGB または CMYK をリストで与える。各色０～１。
色の名称はカラーコード一覧 の 68 色が指定できる。

【例】Cinderella の描画ツールと CindyScript で線分 AB,AC を 60°の角をなすように描いておき，点 D と E を弧の両端になるように設定して

```
Setcolor([1,0,0]);
Circledata([A,D],["Rng=[0,pi/3]"]);
Arrowhead(E,[-1,0.8],[2,1]);
```

を実行すると，矢じりつきの弧を赤で表示することができる。
１行目は，`Setcolor("red");` でもよい。座標軸を描く場合は，このあと `Setcolor("black")` で黒に戻しておかないと，座標軸が赤で表示されてしまうので要注意。



⇒ 関数一覧

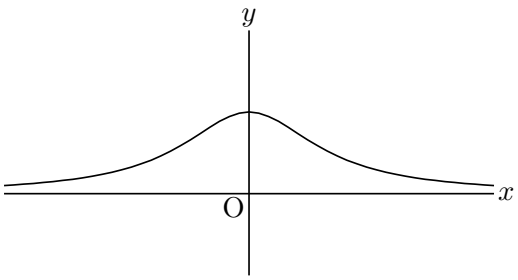
関数　Deffun(関数名 , 定義のリスト)

機能　関数を定義する

説明　関数定義は，CindyScript の関数定義 f(x):=式 でもできるが，Deffun() を使うことにより，R でこの関数を利用することができる。目的に応じて使い分けるとよい。
式のリストには if 文を用いた場合分けの関数式を記述することもできる。

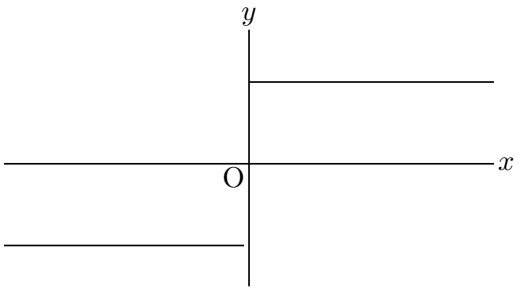
【例】$f(x)=\dfrac{1}{x^2+1}$ を定義し，グラフを描く。

```
Deffun("f(x)",["regional(y)","y=1/(x^2+1)","y"]);
Plotdata("1","f(x)","x");
```



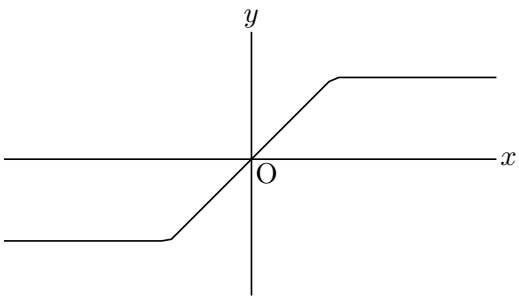
【例】$f(x)=\begin{cases}1 & (x\geq 0)\\ -1 & (x<0)\end{cases}$ を定義してグラフを描く。

```
Deffun("f(x)",["regional(y)","if(x>=0,y=1,y=-1)","y"]);
Plotdata("1","f(x)","x",["Dis=1","Num=100"];
```



if 文はネストすることができる。

```
Deffun("f(x)",["regional y","if(x>1,y=1,if(x>-1,y=x,y=-1))","y"]);
```



関数　Defvar(文字列)

機能　変数を定義する

説明　変数の定義を R と共有する。

【例】`Defvar("const=3");`

複数の変数を定義するときはリストにする。

【例】`Defvar([“ a” ,3,” b” ,1]);`

関数　Fontsize(記号)

機能　フォントサイズを設定する

説明　次に Fontsize() を実行するまで有効

記号は，”t” , ”ss” , ”f”, ”s” , ”n” , ”la”,”La”, ”LA”, ”h” , ”H”

【例】作図ツールの「点を加える」で，A〜G の点をとっておく。小さい方からいくつか表示する。

```
Pointdata("1",[A,B,C,D,E,F,G],["Size=2"]);
Fontsize("t"); Letter([A,"s2","A"]);
Fontsize("ss"); Letter([B,"s2","B"]);
Fontsize("s"); Letter([C,"s2","C"]);
Fontsize("la"); Letter([D,"s2","D"]);
Fontsize("La"); Letter([E,"s2","E"]);
Fontsize("h"); Letter([F,"s2","F"]);
Fontsize("H"); Letter([G,"s2","G"]);
```

関数　Ptsize(n) , Setpt(n)

機能　表示する点の大きさを設定する。

説明　Ptsize() と Setpt() は同じである。 初期設定は 1

全体の点の大きさを設定する。点の大きさを個々に変えたい場合は，size オプションを用いる。

【例】1 から 4 までの点の大きさ

あらかじめ，Cinderella の作図ツールで点 A,B,C,D を作図しておく。

```
Pointdata("1",A,["Size=1"]);
Pointdata("2",B,["Size=2"]);
Pointdata("3",C,["Size=3"]);
Pointdata("4",D,["Size=4"]);
```

⇒ 関数一覧

関数　Setarrow(size,angle,position,cut,segstyle)

機能　矢線の形状の設定をする。

説明　引数は，順に，矢じりの大きさ, 開き角, 矢じり位置, 切り込み位置, 線種と表示色
それぞれの意味と値は，Arrowdata() を参照のこと。

関数　Setmarklen(数)

機能　座標軸の目盛の長さを設定する

説明　Htickmark() , Vtickmark() で座標軸に目盛を入れるとき，その長さを設定する。

関数　Setorigin(座標)

機能　描画する座標軸の原点を設定（移動）する。座標系は変化しない。

説明　描画する座標軸の原点を引数の座標とする。座標は点の識別名でもよい。

【例】原点を (3,2) として座標軸を描く。

```
Setorigin([3,2]);
```

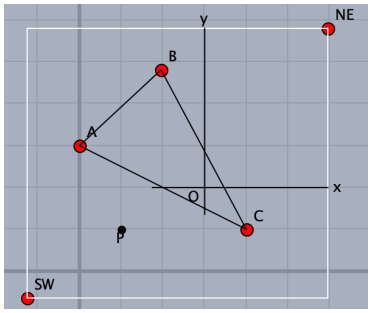
原点を点 A の位置にして座標軸を描く。

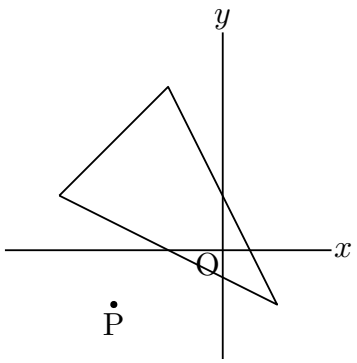
```
Setorigin(A);
```

【例】原点は (3,2) に移動するが，スクリプトではもとの座標系を使う。

```
Setorigin([3,2]);
Listplot([A,B,C,A]);
Pointdata("1",[1,1],["Size=3"]);
Letter([[1,1],"s2","P"]);
```

左が実行時の Cinderella の画面，右が TEX の結果。





関数　Setpen(数)

機能　線の太さの初期値を設定する

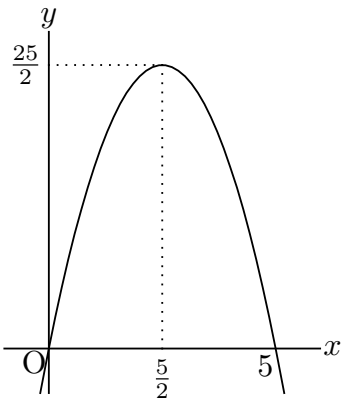
`Listplot()` などの描画関数のオプション `dr` で，個々の太さは指定できる。

関数 Setscaling(倍率)

機能 縦方向の倍率を設定する。倍率は実数またはリスト。実数の場合は縦方向，リストの場合は [横方向，縦方向] の指定となる。

説明 2 次関数の応用問題などでは，グラフが縦に大きくなる場合があり，y 軸方向のスケーリングを変えたいことがよくある。次のスクリプトは，$f(x) = -x^2 + 10x$ のグラフを縦軸方向を半分にして描くものである。

```
Setscaling(0.5);
A.xy=[0,25/4];
B.xy=[5/2,25/4];
C.xy=[5/2,0];
Listplot([A,B],["do"]);
Listplot([C,B],["do"]);
Plotdata("1","-2*x^2+10*x","x");
Letter([[5,0],"s2w","5",[0,25/2],"w2",
"$\frac{25}{2}$",C,"s4","$\frac{5}{2}$"]);
```



ここで，点 A,B の座標が

```
A.xy=[0,25/4];
B.xy=[5/2,25/4];
```

となっていることに注意されたい。y 座標をあらかじめ半分にしている。すなわち，Cinderella で作図した幾何要素に対しては Setscaling は無効である。これは，Putpoint 関数を用いて点の位置を決めても同じである。

たとえば，次のスクリプトでは，Cinderella の画面上では 2 本の線分が点 B でつながるが，書き出された TEX の図では離れてしまう。

```
Setscaling(0.5);
Putpoint("A",[0,2]);
Putpoint("B",[2,2]);
Listplot([A,B]);
Listplot("1",[[0,0],[2,2]]);
```

関数 Setunitlen(文字列)

機能 単位長を設定する。 初期設定は 1cm。

この関数は，スクリプトの初めの方に書くのがよい。

【例】Setunitlen(”8mm”)

関数 Setwindow(x の範囲, y の範囲)

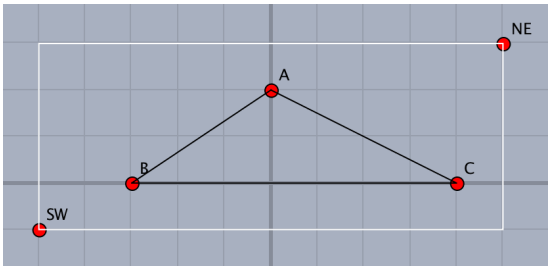
機能 出力する描画領域を設定する。

説明 出力する描画領域は，通常は 2 点 SW と NE を対角とする矩形領域である。

この 2 点をドラッグすることによりビジュアルに描画領域を決められる。

しかし，これとは別に出力範囲を設定したい場合にこの関数を用いる。
また，表を作成したときは，表の範囲が出力範囲として優先される（Tabledata() を実行したとき）ので，表外に図を描いた場合は，最後にこの関数で出力範囲を指定して書き出す。

【例】`Setwindow([-5,5],[-1,3]);`



⇒ 関数一覧

1.2 描画

1.2.1 書式とオプション

描画関数は曲線などを作図する関数である。
基本的な書式は次の通り。

関数名 (name , 点リストなど , options);

name は，プロットデータの名称で，関数ごとに決められた頭部のあとに付けられる。たとえば，線分を描く Listplot() でできるプロットデータは，頭部が”sg”であり，name を”1”とすれば，”sg1” という名称のプロットデータができる。name 指定は不要の場合もあり，その場合は KETCindy が自動的に名称を作成する。なお，name に演算記号は使えないので，番号として負の数は使えない。

点リストなどには，点の座標，点の識別名，複数の点のリスト，複数の点を示す文字列などがあり，関数によって異なる。点は Cinderella で作図した幾何要素の点を利用できる。

options は，線種・表示する文字列・解像度・出力の有無などを指定するオプション群。

線種はつぎの 4 通り。 初期設定は実線。

”dr, n”	太さ n の実線で描く。
”da,m,n”	破線を描く。 m は破線の長さ，n は破線の間隔 (m,n は省略可) m,n オプションは Cinderella の描画面には反映されない。
”id,m,n”	ギャップからはじまる破線を描く。
”do,m,n”	点線で描く。

m は点の間隔，n は太さ (m,n は省略可)

描画色指定は，RGB または CMYK のリストで指定するか，色名を用いる。

【例】`"Color=[0,0.7,0]"` で暗い緑になる。

出力の有無は

”notex” Cinderella 画面上の図形を出力しない

”nodisp” Cinderella 画面上にも出力しない

”nodisp”は画面上にも，Rへのデータにも出力されないが，プロットデータは作成されるので，プロットデータだけを利用したい場合に有効である。

【例】`pdata=Circledata([A,B],["nodisp"]);`

として，後にプロットデータ pdata を利用する。

その他，次のようなオプションがある。

”Size=n” 点の大きさ，線の太さの指定

”Num=n”曲線の場合の分割数（プロットデータの個数 +1）

1.2.2 点・線分・直線

関数 Pointdata(name , 点リスト , options)

機能 点のデータを作成する。

説明 与えられた座標の点データを作成する。オプションは”Size=”,”Color=”,”Inside”。

Inside オプションは，点の内部についての指定。

0：白抜き

0 から 1 まで：濃度

-1：塗らない

カラーコードまたは色名：その色で塗る

【例】

(1) 座標指定で 2 つの点データを作る。

```
Pointdata("1",[[1,2],[-2,3]]);
```

(2) 作図した点 A,B について，点データを作る。

```
Pointdata("1",[A,B]);
```

A,B が作図されていない場合は作成されない。

Cinderella の描画面上では既存の点 A,B に黒の点が重なって表示される。

(3) A の位置に大きさ 4 で点を作る。

```
Pointdata("1",A,["size=4"]);
```

(4) 点データを作り，TeX にオプション 0（白抜き）で描く

```
Pointdata("1",[A,B],["Inside=0"]);
```

(5) 点データを作るが，TeX には出力しない

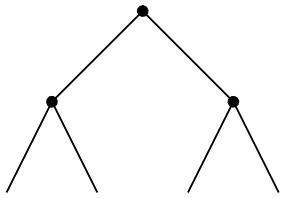
```
Pointdata("1",[[3,4],[5,6]],["notex"]);
```

(6) 点データを作るが，TeX には出力せず画面上にも表示しない。

```
Pointdata("1",[[3,4],[5,6]],["nodisp"]);
```

(7) 節点を明示した木を描く

```
Ptsize(3);
Pointdata("1",[[1,2],[3,4],[5,2]]);
Listplot("1",[[0,0],[1,2],[3,4],[5,2],[4,0]]);
Listplot("2",[[1,2],[2,0]]);
Listplot("3",[[5,2],[6,0]]);
```



注) 幾何点の有無など，付録の「点の作図についての比較表」を参照のこと。

関数 Putpoint(点名 , 座標 1 , 座標 2)

機能 点を作る

説明 識別名が点名の点を，既存でなければ座標 1 に作る。既存ならば座標 2 に移動する。Tex には出力されない。

【例】点 A を作る。

(1,1) に固定点 A を作る。 この点は動かすことができない。

```
Putpoint("A",[1,1]);
```

(1,1) に自由点を作るには次のようにする。

```
Putpoint("A",[1,1],[A.x,A.y]);
```

この点は座標 2 の効果により，自由点となり，ドラッグして動かすことができる。

注) 点名は半角アルファベットとする。数字や漢字でも Cinderella では点ができるが，R でエラーとなる。

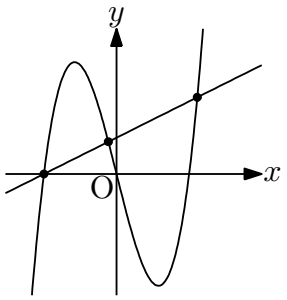
関数 Putintersect(点名 , PD1 , PD2 , [No])

機能 2 曲線の交点を作る

説明 PD1,PD2 は 2 曲線のプロットデータ名。作成される点は幾何点。
描画範囲に交点が 1 つだけのとき，第 4 引数がなくても交点が作られる。
描画範囲に 2 つ以上の交点がある場合，第 4 引数を省略するとコンソールに交点の座標のリストと，「Choose point number 」というガイドが表示される。そこで，引数の No として，その番号を指定すると，その点が作られる。この関数で作成されるのは幾

何点だけなので，TEX の図に点として明示するためには Pointdata() で書き出す。
次の例は，3 次曲線と直線の交点を 3 つとも取ったものである。

```
Plotdata("1","x^3-4*x","x",["Num=200"]);
Plotdata("2","1/2*x+1","x");
Putintersect("P","gr1","gr2",1);
Putintersect("Q","gr1","gr2",2);
Putintersect("R","gr1","gr2",3);
Pointdata("1",[P,Q,R],["size=4"]);
```



交点が存在しない場合は，「No intersect point」がコンソールに表示される。

関数 Putoncurve(点の名前, プロットデータ, options)

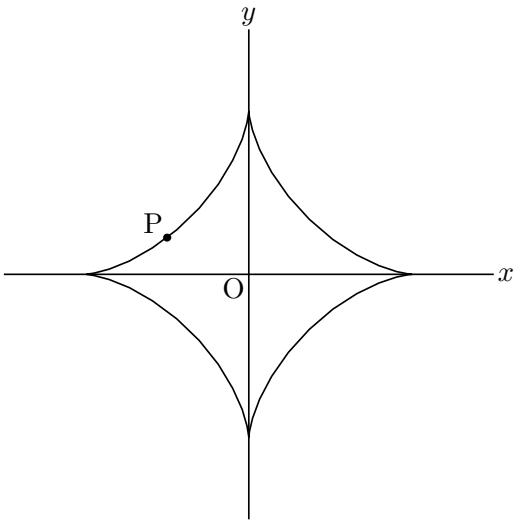
機能 曲線上に点を乗せる。

説明 点が存在しない場合は新たに作る。すでにその点が存在する場合は，その点の x 座標を使う。初期値の x 座標の 初期設定は 0。
options は，x 座標の範囲をリストで与える。

【例】アステロイド上の動点 P をとる。

```
Paramplot("1","[2*cos(t)^3,2*sin(t)^3]","t=[0,2*pi]");
Putoncurve("P","gp1",[-1,1]);
```

点 P がアステロイド上にでき，この点はドラッグするとアステロイド上を $-1 \leq x \leq 1$ の範囲で動かすことができる。ただし，-1,1 の付近は y 座標の判断の関係でぴったりはいかない。



⇒ 関数一覧

関数 Putonline(点名 , 座標 1 , 座標 2)

機能　直線上に点を作る

説明　座標１，座標 2 を通る直線上に点名の点を作る。できた点は直線に対してインシデントとなる。

【例】点 A, B を通る直線上に点 P をとる。

```
Putonline("P",A,B);
```

関数　Putonseg(点名 , 座標１ , 座標 2)

機能　線分上に点を作る

説明　座標１，座標 2 を端点とする線分上に点名の点を作る。できた点は線分に対してインシデントとなる。指定した点がすでに存在する場合は動かさない。

【例】

線分 AB 上に点 C をとる。

```
Putonseg("C",A,B);
```

点 (-1,0),(2,2) を通る線分上に点 C をとる。

```
Putonseg("C",[[-1,0],[2,2]]);
```

関数　Reflectpoint(点，対称点または対称軸)

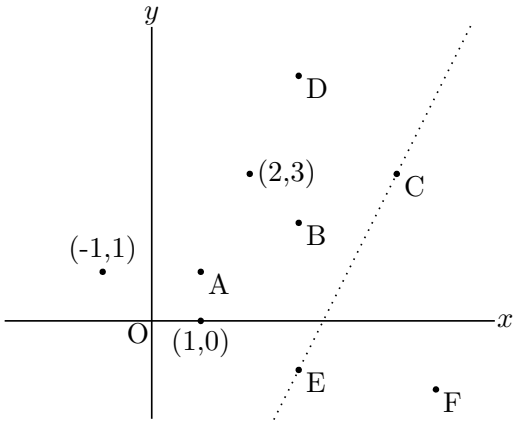
機能　点の鏡映の座標を返す。

説明　点を指定された点または軸に関して対称移動した点の座標を返す。対称軸は [点 1, 点 2] で指定

【例】点 A〜F を作図しておき，C〜F を A の鏡映の位置に配置する。

C は B に関して A と対称な点
D は点 (2,3) に関して A と対称な点
E は点 (1,0) に関して (-1,1) と対称な点
F は直線 CE に関して A と対称な点

```
C.xy=Reflectpoint(A,B);
D.xy=Reflectpoint(A,[[2,3]]);
E.xy=Reflectpoint([-1,1],[[1,0]]);
F.xy=Reflectpoint(A,[C,E]);
Lineplot([C,E],["do"]);
```



注）鏡映は Cinderella の作図ツールでも作成することができる。場合によっては Cinderella で作図する方が簡明である。

関数　Rotatepoint(点，角度 , 中心)

機能　点の位置を回転する

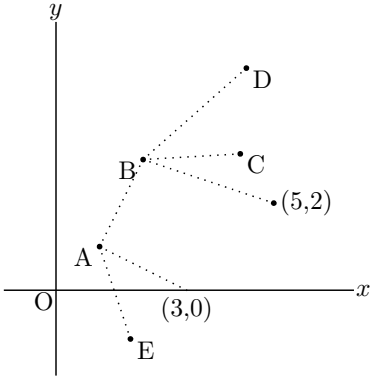
説明　点を，中心で示された点の周りに回転した座標を返す。角度は弧度法で与える

【例】点 A〜E は作図しておき，C〜E をそれぞれの位置に配置する。

点 C は A を，Bに関して $\frac{2}{3}\pi$ だけ回転した点

点 D は点 (5,2) を，Bに関して $\frac{\pi}{3}$ だけ回転した点

点 E は点 (3,0) を A に関して $-\frac{\pi}{4}$ だけ回転した点




```
C.xy=Rotatepoint(A,2*pi/3,B);
D.xy=Rotatepoint((5,2),pi/3,B);
E.xy=Rotatepoint([3,0],-pi/4,A);
```

注）図の点線は位置関係を示すためのもの。

点名や座標は，実際には Letter() 関数で記述する。

関数　Scalepoint(点，比率ベクトル，中心)

機能　点の位置の拡大・縮小を行う

説明　点を，指定された中心を原点とする座標系で，比率ベクトルの分だけ拡大・縮小した位置の座標を返す。

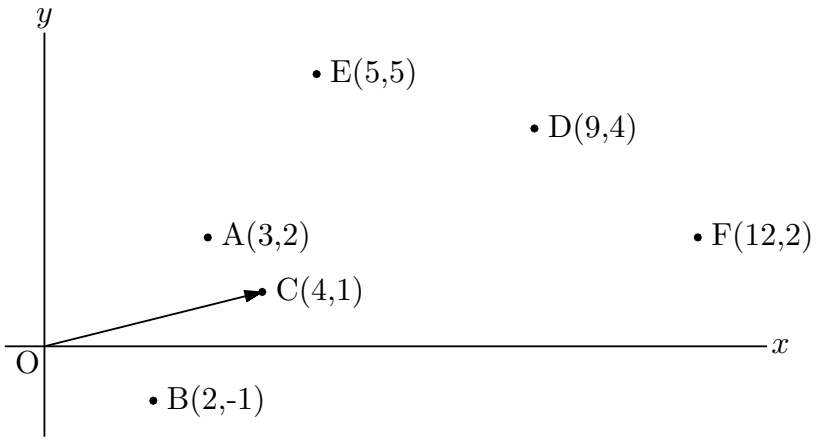
【例】点 A〜F は作図ツールで適当な位置にとっておく。

点 D を，点 A を原点を中心に横に 3 倍，縦に 2 倍した位置に置く。

点 E を，点 A を点 B を中心に横に 3 倍，縦に 2 倍した位置に置く。

点 F を，点 A を原点を中心にベクトル $\overrightarrow{OC}$ で示された比率の位置に置く。

```
D.xy=Scalepoint(A,[3,2],[0,0]);
E.xy=Scalepoint(A,[3,2],B);
F.xy=Scalepoint(A,C.xy,[0,0]);
Arrowdata("1",[[0,0],C]);
Pointdata("1",[A,B,C,D,E,F],["size=2"]);
Letter([A,"e2","A("+A.x+","+A.y+")"]);
Letter([B,"e2","B("+B.x+","+B.y+")"]);
Letter([C,"e2","C("+C.x+","+C.y+")"]);
Letter([D,"e2","D("+D.x+","+D.y+")"]);
Letter([E,"e2","E("+E.x+","+E.y+")"]);
Letter([F,"e2","F("+F.x+","+F.y+")"]);
```



点 A,B,C をドラッグすると，インタラクティブに D,E,F の位置が変わる。

関数　Translatepoint(点 , 移動ベクトル)

機能　点を平行移動する

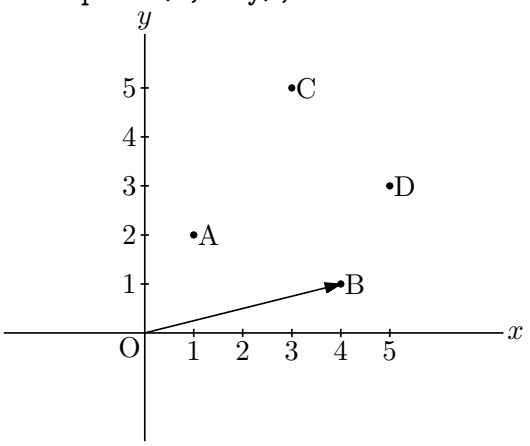
説明　点を移動ベクトルで示された分だけ平行移動した点の座標を返す

【例】点 A〜D は作図しておく。

点 C を点 A を x 軸方向に 2 , y 軸方向に 3 だけ平行移動した点にする。

点 D を点 A をベクトル $\overrightarrow{OB}$ だけ平行移動した点にする。

```
C.xy=Translatepoint(A,[2,3]);
D.xy=Translatepoint(A,B.xy);
```



⇒ 関数一覧

関数　Arrowdata(name,[始点 , 終点] , options)

機能　2 点間を結ぶ矢線を描く。

説明　name は，座標を数値で与えるときに必要。幾何要素の識別名で与えるときはなくてもよい。

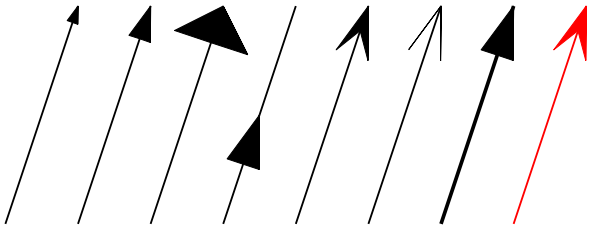
options は矢じりの形状などの指定。

[矢じりの大きさ, 開き角, 矢じり位置, 切り込み位置, 線種, 線の表示色, トリ

ミング]
のリストで与える。矢じりの大きさから切り込み位置までは数で，後方のオプションを指定するときは，前方も書く。
開き角は 60 分法で与える。ただし，° はつけない。 5 未満の時は 18°の倍数指定とする。
矢じり位置は，線分の長さを 1 とした始点からの距離。
切り込み位置は，0 のとき三角形の矢じり，1 のとき線のみの矢じりとなる。
ただし，Cinderella の画面上には全ては反映されない。たとえば，太さ指定をしても画面上では太さは同じ。
トリミング :”Cutend=m” または ”Cutend=[m,n]” で，右辺が数のときは両端を m だけカットする。リストのときは始点を m, 終点を n だけカットする。m が負のときは延長する。
これらの設定は，Setarrow() で初期設定を変更することができる。

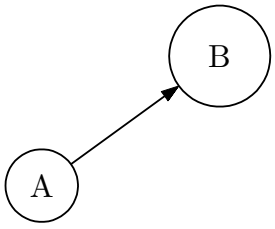
【例】オプションの設定とその結果を示す。

```
Arrowdata("1",[A,B]);
Arrowdata("2",[[1,0],[2,3]],[2]);
Arrowdata("3",[[2,0],[3,3]],[3,45]);
Arrowdata("4",[[3,0],[4,3]],[3,1,0.5]);
Arrowdata("5",[[4,0],[5,3]],[3,1,1,0.5]);
Arrowdata("6",[[5,0],[6,3]],[3,1,1,1]);
Arrowdata("7",[[6,0],[7,3]],[3,"dr,2"]);
Arrowdata("8",[[7,0],[8,3]],[3,1,1,0.5,"Color=red"]);
```



【例】２つの円を矢線で結ぶ。

```
Circledata("1",[A,A.xy+[0.5,0]]);
Circledata("2",[B,B.xy+[0.7,0]]);
Arrowdata([A,B],["Cutend=[0.5,0.7]"]);
Letter([A,"c","A",B,"c","B"]);
```



Cinderella の作図ツールで２点 AB をとっておく。
円 A,B の半径が同じ（たとえば 0.5）であれば, `Arrowdata([A,B],["Cutend=0.5"]);` でよい。

関数 Setarrow([arrowsize,angle,position,cut,linestyle])

機能 Arrowdata,Arrowhead で描く矢印のスタイルを設定する。

説明 arrowsize,angle,position,cut,linestyle は，順に大きさ (1)，開き角 (18)，位置 (1)，切り込み (0)，矢線スタイル (”dr”) である。（カッコ内はデフォルト値）

関数 Arrowhead(点 , 方向 , options) , Arrowhead(点 , プロットデータ,options)

機能 点に矢じりだけを描く

説明 指定された位置に，指定された方向を向いた矢じりだけを描く。
点は座標または幾何要素名。方向は原点から見て座標 [a,b] の方向。
options は [大きさ，矢じりの開き角，形状と位置] のリスト。
矢じりの開き角は 60 分法で片側半分の角。
形状は，”f” ：塗りつぶしの三角形（ 初期設定）または” l ” ： ラインのみ。
ただし，塗りつぶし矢じりは画面上では塗りつぶしにならない。
位置は，”t”（ 初期設定）または ”c” , ”b”

”t” は矢じりの先端が終点に一致
”c” は三角形の中心が終点と一致
”b” は終点が矢じりの底辺にのる。

プロットデータを指定したときは，曲線上の点に矢じりをつける。
曲線には向きがあり，それによって矢じりの向きが決まる。” Invert(PD) ” とすると反対向きの矢じりになる。
曲線の向きとは，曲線を描くときの順序で，プロットデータの順序でもある。invert() はこのプロットデータを逆順にするものである。

【例】

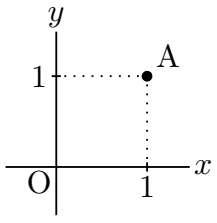
点 A が右図の位置のとき

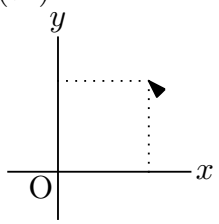
(ア) Arrowhead(A,[-1,1]);

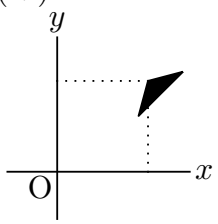
(イ) Arrowhead([1,1],[-1,1],[2,60]);

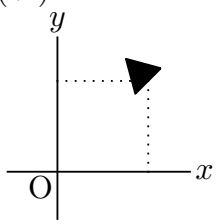
(ウ) Arrowhead(A,[-1,1],[2,30,”b”]);

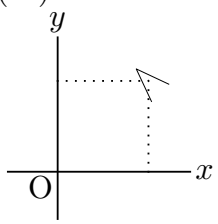
(エ) Arrowhead([1,1],[-1,1],[2,20,”lc”]);



(ア)


(イ)


(ウ)


(エ)


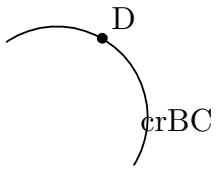
曲線 crBC 上の点 D が右図のようなとき

(オ) Arrowhead(D,”crBC”);

(カ) Arrowhead(D,”crBC”,[2]);

(キ) Arrowhead(D,”crBC”,[2,30,”l”]);

(ク) Arrowhead(D,”Invert(crBC)”);



(オ) (カ) (キ) (ク)

関数 Lineplot（name , 2 点のリスト , options)

機能 2 点のリストで示された点を結ぶ直線を描く。

説明 2 点のリストは座標または幾何要素の名前で与える。

options は次の通り。

線種”dr, n”, ”da,m,n” , ”do,m,n”

”+”半直線を描く。

”dr” , ”da” , ”do” と ”+” はリストにして両方指定することができる。

点のリストが，座標ではなく幾何要素名のリストの場合は，name は省略できる。

いくつか例を示す。

各座標を結ぶ直線を引く

```
Lineplot("1",[[0,0],[1,2]])
```

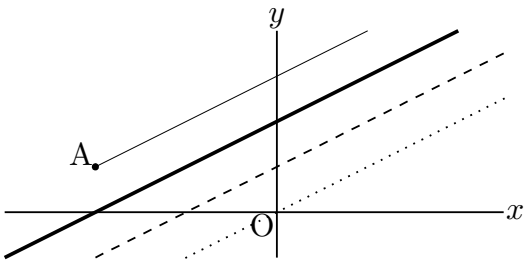
Cinderella の描画ツールで 2 点 A,B をとっておき，直線 AB を引く

```
Lineplot([A,B]);
```

option の働きの例

`Lineplot([A,B],["dr,0.5","+"]);`	A を端点とする半直線を引く
`Lineplot([C,D],["dr,2"]);`	直線 CD を太さ 2 で描く
`Lineplot([E,F],["da"]);`	直線 EF を破線で描く
`Lineplot([G,H],["do"]);`	直線 GH を点線で描く

結果は，次図左上から。



⇒ 関数一覧

関数　Listplot（name , 点のリスト , options)

機能　点のリストで示された点を結ぶ。

説明　点のリストは座標または幾何要素名のリストで与える。点が，座標ではなく幾何要素名の場合は，name は省略可

プロットデータの名前は，”sg” に引数の name を付加したものとなる。

options は次の通り。

線種 ”dr, n”, ”da,m,n” , ”do,m,n”

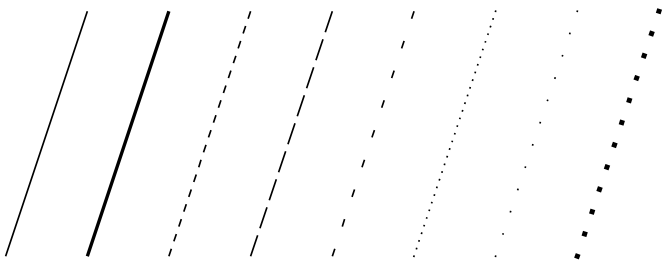
トリミング :”Cutend=m” または ”Cutend=[m,n]”

数のときは両端を m だけカットする。リストのときは始点を m, 終点を n だけカットする。m が負のときは延長する。

options の使用例

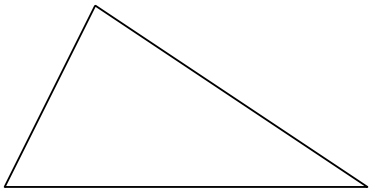
`Listplot([A,B]);`	線分 AB を描く。太さは 初期設定。
`Listplot([C,D],["dr,2"]);`	線分 CD を描く。太さ 2
`Listplot([E,F],["da"]);`	線分 EF を破線で描く
`Listplot([G,H],["da,3,1"]);`	線分 GH を破線で描く。線を長く
`Listplot([K,L],["da,1,3"]);`	線分 KL を破線で描く。間隔を空ける
`Listplot([M,N],["do"]);`	線分 MN を点線で描く。
`Listplot([O,P],["do,3"]);`	線分 OP を点線で描く。間隔を空ける
`Listplot([Q,R],["do,3,3"]);`	線分 QR を点線で描く。間隔を空けて太く

結果は次図左から。



【例】三角形を描く。
Cinderella の作図ツールで三角形 ABC を描いておく。あるいは，単に 3 点 A,B,C をとるだけでもよい。

```
Addax(0);
Listplot([A,B,C,A]);
```

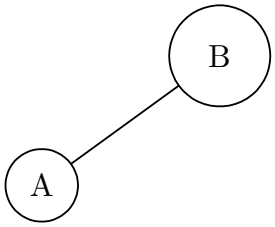


点の位置は座標で指定してもよい。
その場合は name が必要。

```
Listplot("1",[[0,0],[2,0],[1,2],[0,0]]);
```

【例】２つの円を線分で結ぶ。

```
Circledata("1",[A,A.xy+[0.5,0]]);
Circledata("2",[B,B.xy+[0.7,0]]);
Listplot([A,B],["Cutend=[0.5,0.7]"]);
Letter([A,"c","A",B,"c","B"]);
```



Cinderella の作図ツールで２点 AB をとっておく。
円 A,B の半径が同じであれば，`Listplot([A,B],["Cutend=0.5"]);` でよい。

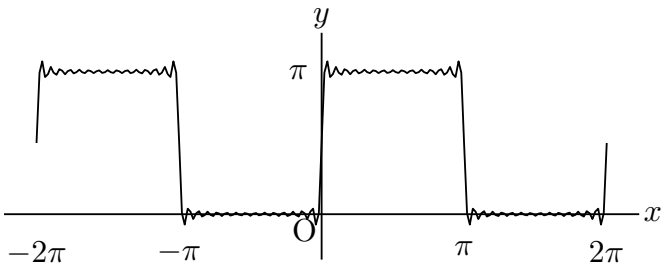
プロットデータは点の座標のリストである。したがって，プロットデータを自作して Listplot() で表示することができる。

【例】有限フーリエ級数展開

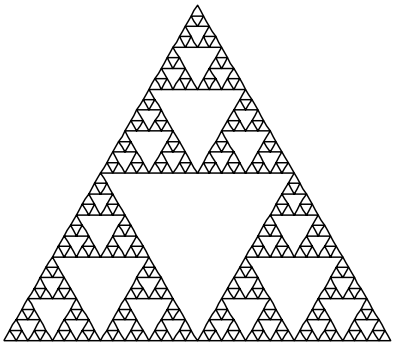
$$\frac{\pi}{2}+\sum_{n=0}^{30}\frac{1-(-1)^n}{n}\sin nx$$

次のように Cindyscript で関数を定義し，プロットデータ pd を作って引数に渡す。

```
f(x):=(
s=pi/2;
repeat(30,n,s=s+(1-(-1)|^n)/n*sin(n*x));
);
pd=apply(0..200,t,
x=-2*pi+t*4*pi/200;
[x,f(x)];
);
Listplot("1",pd);
Expr([[-2*pi,-0.5],"s","-2\pi",[-pi,-0.5],"s","-\pi",[pi,-0.5],"s",
"\pi",[2*pi,-0.5],"s","2\pi",[0,pi],"w2","\pi"]);
```



リストの長さには制限がある。たとえば，タートルグラフィクスを用いたシェルピンスキーのギャスケットでは 200 くらいずつのリストに分割する。



⇒ 関数一覧

関数　Mksegments()

機能　すべての幾何線分の PD を作成

説明　Cinderella の「線分を加える」ツールで描いたすべての線分をそのままプロットデータとする。たとえば，線分 AB を作ると，プロットデータ sgAB が作成される。その後，インスペクタで点 B の識別名を変更（たとえば Q に）すると，プロットデータ名も変更される。線分はすでに描かれていてもよい。

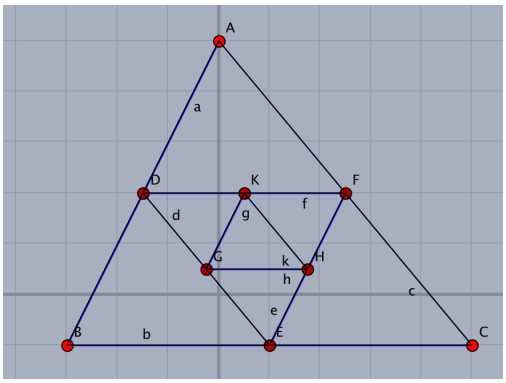
【例】等比数列の例題

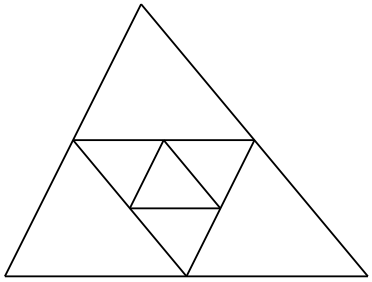
三角形の各辺の中点を結んでできる三角形を次々に作っていく，等比数列の図を描く。

まず「線分を加える」ツールで三角形 ABC を描く。

「中点を加える」ツールで各辺の中点を取り，「線分を加える」ツールで中点を結ぶ。

これを繰り返す。`Mksegments()` を書いておけば，`Listplot([A,B,C]` などを書かなくても，作図ができた時点で，図のデータができる。





関数　Framedata(name , リスト,options)

機能　矩形を描く

説明　リストの形は 2 通り。

その 1：[中心 , 横 , 縦] で，矩形を描く。横，縦は中心からの距離。

その 2：2 点のリスト。点が座標でなく名称のときは name は省略できる。

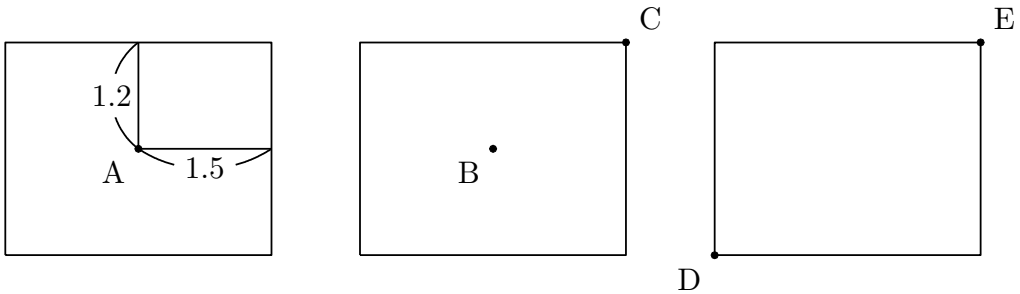
点の座標は点の名前でもよい。点を座標で与える場合は name は省略できない。

リストを省略した場合は，描画範囲と同一の矩形を描く。

その 2 のタイプでは，option として，”center” または ”corner” がある。”center” のときは，中心と対角点（ 初期設定），”corner” のときは 2 点を対角点として解釈する。

以下にいくつか例を示す

`Framedata("1");`	描画範囲 (SW,NE) と同一の矩形を描く
`Framedata("2",[[0,0],2,2]);`	原点を中心とする縦横幅 4 の正方形を描く
`Framedata("3",[A,1.5,1.2]);`	点 A を中心とする横 3，縦 2.4 の矩形を描く。(図左)
`Framedata([B,C]);`	点 B を中心, 点 C を頂点とする矩形を描く。(図中央)
`Framedata([D,E],["corner"]);`	点 D,E を対角点とする矩形を描く。(図右)



矩形の角を丸めたい場合は，Framedata() ではなく，Ovaldata() を使うとよい。

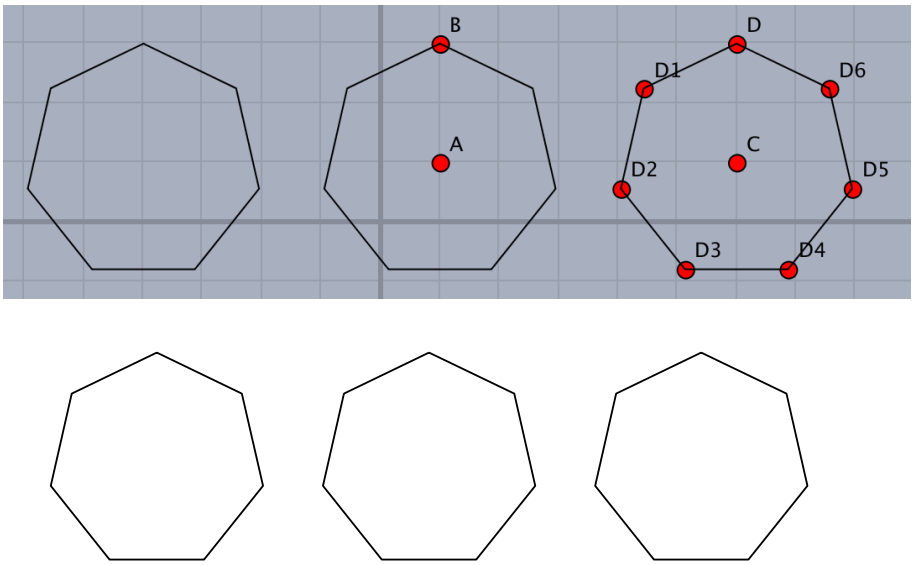
関数　Polygonplot(name , 点リスト , 整数,options)

機能　2 点を半径とする円に内接する正多角形を描く。

説明 点リストを [A,B] とすると，A を中心とする半径 AB の円周上に点をとって正多角形を描く。ただし円は描かない。A,B は座標でもよい。
点リストが座標ではなく作図してある点の名称のとき，オプションに”Geo=y” をつけると，頂点の幾何点を作る。幾何点の名称は B に番号を付けたものとなる。整数でない数を指定した場合は，きちんと閉じない折れ線が描かれる。

【例】点リストと option の違いによる作図と，TeX の図を示す。

```
Addax(0);
Polygonplot("1",[[-4,1],[-4,3]],7);
Polygonplot("2",[A,B],7);
Polygonplot("3",[C,D],7,["Geo=y"]);
```



円に内接する形でなく，与えられた線分 AB を 1 辺とする正多角形を描くには次のようにする。
線分 AB は，Cinderella の作図ツールなどで描かれているものとする。ただし，線分でなく，両端の点が与えられているだけでもよい。Cindyscript で点 A,B が複素平面上にあるものとして，多角形の頂点の位置を計算する。

【例】AB を 1 辺とする正五角形を描く。

```
n=5;
pti=[complex(A),complex(B)];
th=2*pi/n;
repeat(n-2,s,
z1=pti_s;
z2=pti_(s+1);
z=z2+(z2-z1)*(cos(th)+i*sin(th));
pti=append(pti,z);
);
```

```
pt=apply(pti,gauss(#));
pt=append(pt,A.xy);
Listplot("1",pt);
```

pti は，各頂点に対応する複素数のリスト，pt が各頂点の座標のリストである。

⇒ 関数一覧

1.2.3 曲線

関数 Bezier(名前，節点リスト，制御点リスト，[オプション])

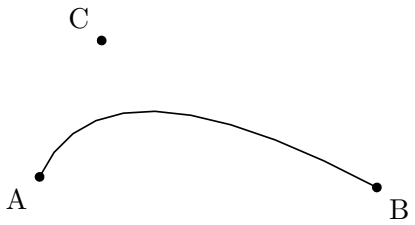
機能 ベジェ曲線を描く

説明 制御点は，各区間に対して，3 次の場合 2 個，2 次の場合 1 個のリストで与える。
オプションは
”Num=n”: 節点間の分割数（分点数 −1）を指定できる。 ベジェ曲線とスプライト曲線の関数は節点間が短い場合が多いので初期設定は 10 になっている。Plotdata() などと違い，大きい数（200 など）を指定すると，全体の分割数が増大して描画時間がかかるようになってしまうので注意。

【例】

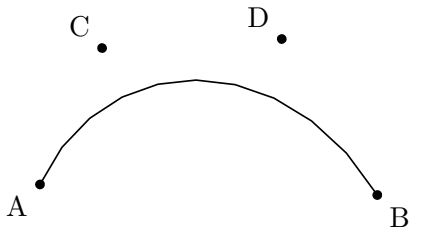
2 次ベジェ曲線

```
Bezier("1",[A,B],[C]);
```



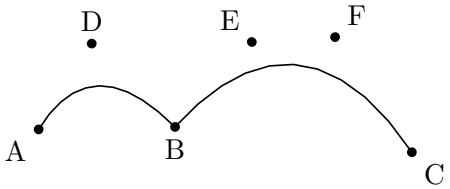
3 次ベジェ曲線

```
Bezier("2",[A,B],[C,D]);
```



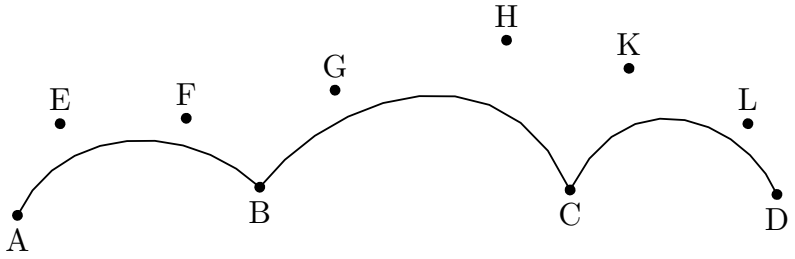
節点を増やす。2 次と 3 次。

```
Bezier("3",[A,B,C],[[D],[E,F]]);
```



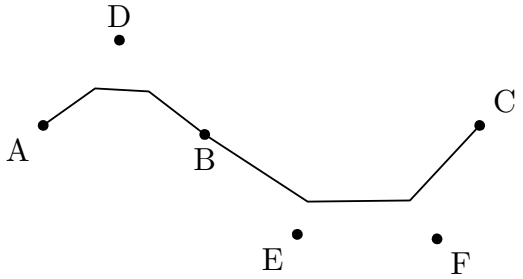
全て同じ次数の場合，次のようにしてもよい。

```
Bezier("4",[A,B,C,D],[E,F,G,H,K,L] );
```

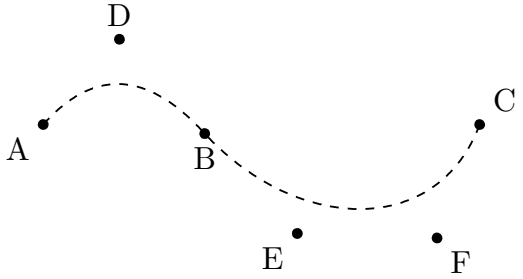


オプションの例

```
Bezier("5",[A,B,C],[[D],[E,F]],["Num=3"]);
```

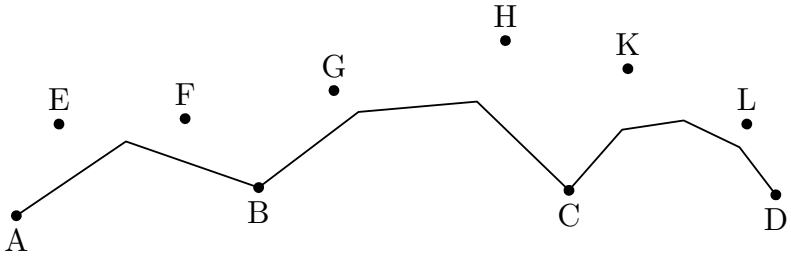



```
Bezier("6",[A,B,C],[[D],[E,F]],["Num=40","da"]);
```



Num を（ベクトルとして）区間ごとに与えることもできる。

```
Bezier("1",[A,B,C,D],[E,F,G,H,K,L],["Num=[2,3,4]"]);
```



関数 Beziersmooth(名前，節点リスト，[オプション])

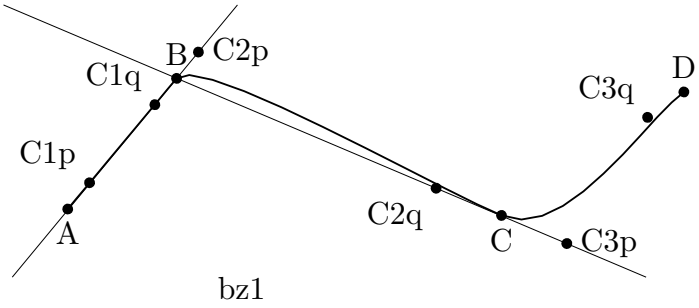
機能 節点間を 3 次ベジェ曲線でスムーズに結んだ曲線を描く

説明 節点をはさむ制御点は 1 直線上にとる（したがって，1 つは半自由点で，直線上しか動けない)。制御点は自動的に配置される。その後，節点や制御点を動かして，描きた

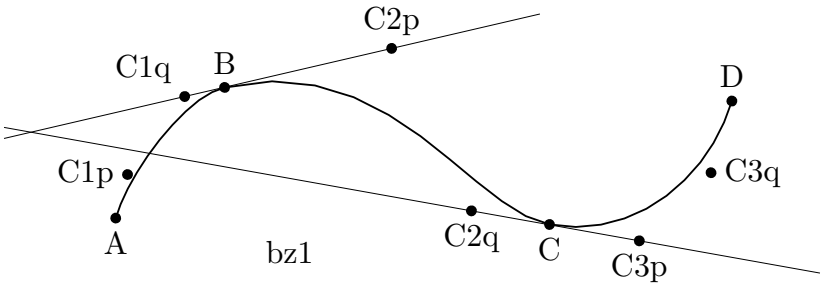
いものにする。

【例】

```
Beziersmooth("1",[A,B,C,D]);
```



その後，節点や制御点を動かして，描きたいものにする。ただし，C2p は C1q と B を通る直線上しか動けない。C3p は C2q と C を通る直線上しか動けない。



関数 Beziersym(名前，節点リスト，[オプション])

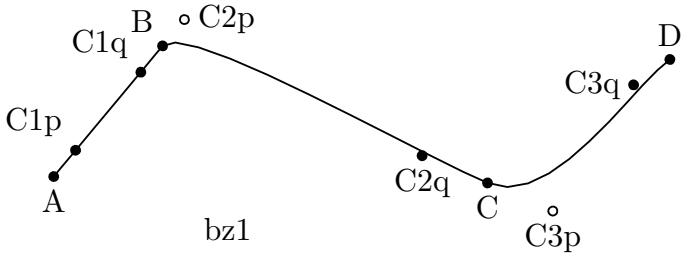
機能 節点間を 3 次ベジェ曲線でスムーズに結んだ曲線を描く

説明 節点をはさむ制御点は節点に関し対称（片方は表示されず，動かせない)。制御点は自動的に配置される。その後，節点や制御点を動かして描きたいものにする。

【例】

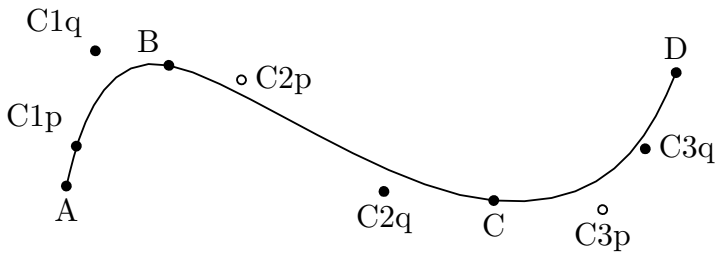
```
Beziersym("1",[A,B,C,D]);
```

C2p と C3p は表示されない



その後，節点や制御点を動かして，描きたいものにする。

C2p と C3p は表示されず，動かせない。



⇒ 関数一覧

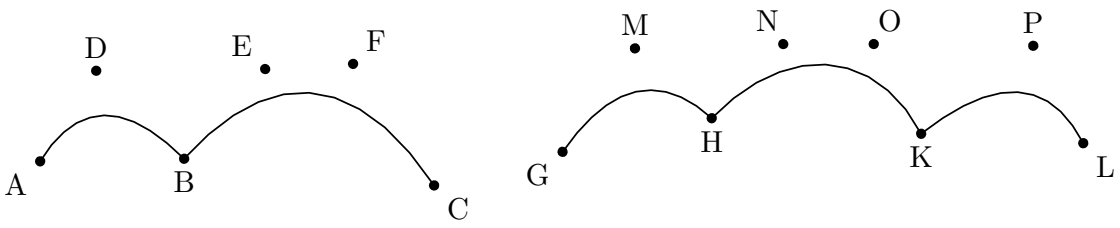
関数　Mkbeziercrv(名前, [節点リスト, 制御点リスト] のリスト,options)

機能　複数のベジェ曲線を描く

説明　[節点リスト, 制御点リスト] が 1 つの場合は，Bezier() と同じ。

【例】ベジェ曲線を 2 つ描く。

```
Mkbeziercrv("5",[[[A,B,C],[[D],[E,F]]],[[G,H,K,L],[[M],[N,O],[P]]]]);
```



関数　Mkbezierptcrv(節点リスト ptlist，[オプション])

機能　ベジェ曲線を描く

説明　制御点は，自動的に配置される。

複数の場合は [ptlist1, ptlist2....]

名前は，A から順に自動的につける。

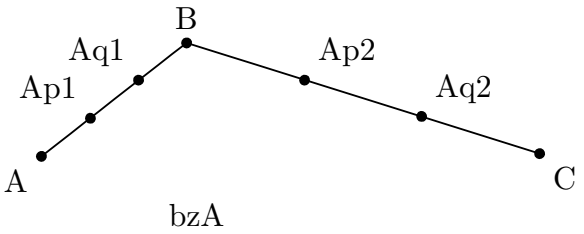
オプション

`"Deg=..."` 次数指定ができる。（初期設定は 3 次）

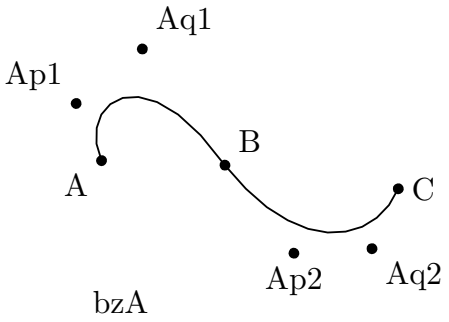
`"Num=..."` 各区間の区間数（分点数 −1）を指定できる。（初期設定は 10）

【例】

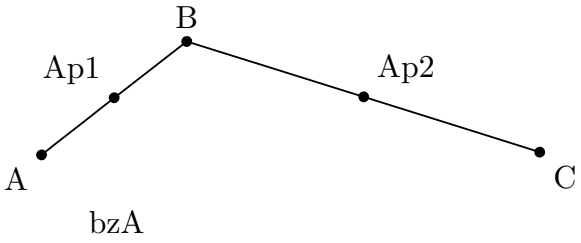
```
Mkbezierptcrv([A,B,C]);
```



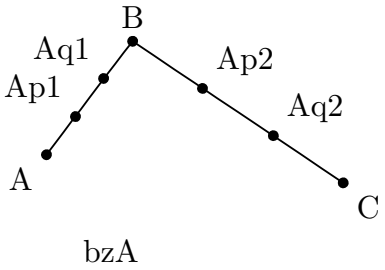
その後，節点や制御点を動かして，
描きたいものにする。

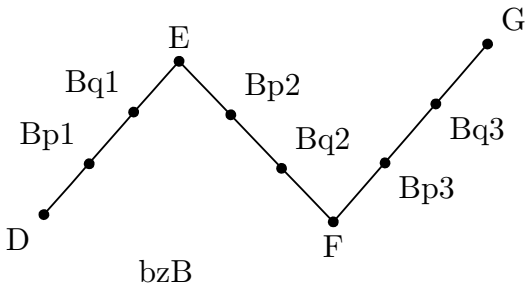


`Mkbezierptcrv([A,B,C],["Deg=2"]);`
`Deg=2` とすると 2 次になる。
制御点は各区間に 1 個ずつできる。



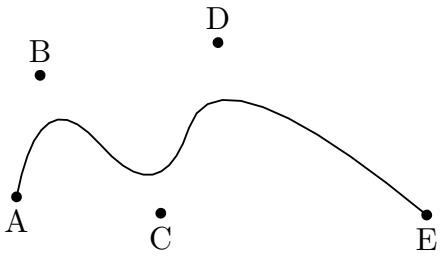
複数の場合は [ptlist1, ptlist2....]
`Mkbezierptcrv([[A,B,C],[D,E,F,G]]);`



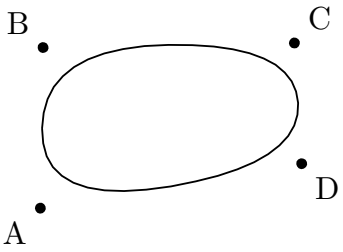


関数 Bspline(名前，制御点リスト，[オプション])
機能 2 次 B-spline 曲線を描く
説明 節点は自動的に計算され，表示されない

【例】`Bspline("1",[A,B,C,D,E])`
これは， `Bezier("1",[A,(B+C)/2,(C+D)/2,E],[B,C,D])` と同じ。曲線の名前が bz1 ではなく bzb1 となる。
通常の B-spline 曲線の端の制御点の代わりに，端点を動かせるようにしている。



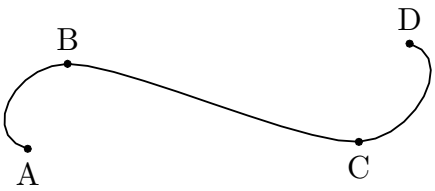
【例】`Bspline("1",[A,B,C,D,A]);`
リストの最初と最後が同じ場合は閉曲線になる。
`Bezier("1",[(D+A)/2,(A+B)/2,(B+C)/2,(C+D)/2,(D+A)/2],[A,B,C,D]);`
と同じ。



参照：Ospline：大島のスプラインを描く

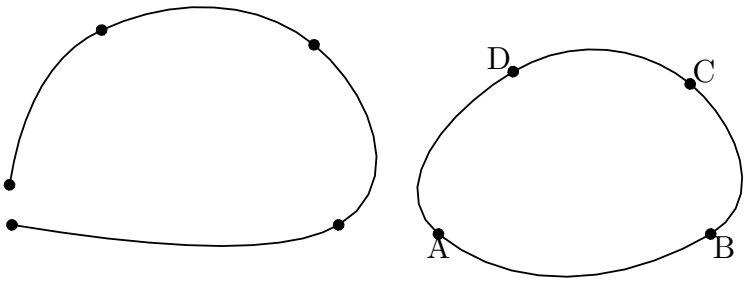
関数 CRspline(名前，節点リスト，[オプション])
機能 単独の Catmull-Rom スプライン曲線を描く
説明 自由点は，節点のみで，制御点は節点から作られ移動はできない。
オプションに，通常のオプションのほか，次が使える。
`size ->n`:画面上での線の太さを指定する。

【例】`CRspline("1",[A,B,C,D]);`



関数 Ospline(名前，制御点リスト，[オプション])
機能 大島の spline 曲線を描く
説明 制御点を通るスプライン曲線を描く
リストの最初と最後が同じ場合は閉曲線になる。

【例】`Ospline("1",[A,B,C,D,E]);Ospline("1",[A,B,C,D,A]);`



スプライン曲線については次も参照されたい：Bspline：Bスプラインを描く

関数　Circledata(name, リスト,options)

機能　円または多角形を描く。

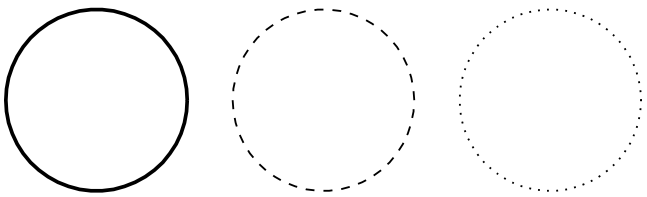
説明　中心の点と，円周上の 1 点（か半径），または 3 点をリストで与えて円を描く。
中心と円周上の点を，座標ではなく幾何要素名で指定する場合は name は省略可。
options は以下のものをリストで与える。省略した場合は実線で円が描かれる。

”Rng=[θ_1, θ_2]”	角 θ_1 から θ_2 の範囲の弧を描く。角は弧度法で与える。
”Num=分割数”	円を描くときの分割数。値が小さい場合は多角形になる。
線種	”dr, n”, ”da,m,n” , ”do,m,n”

【例】いろいろな円を描く。

原点中心，半径 2 の円	`Circledata("1",[[0,0],[2,0]]);`
	(`[[0,0],2]` でもよい)
A 中心，半径 AB の円	`Circledata([A,B]);`
A 中心，半径 2 の円	`Circledata("1",[A,A+[2,0]]);`
	(`[A,2]` でもよい)
3 点 A,B,C を通る円	`Circledata([A,B,C]);`

下図左より，オプションに ”dr,2” , ”da” , ”do” をつけた場合。



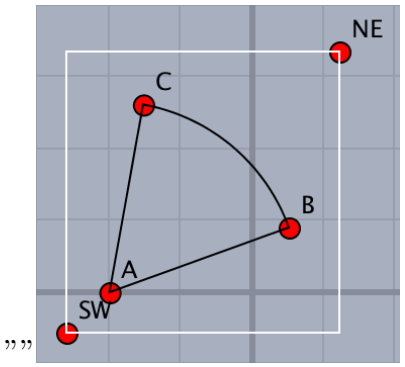
`Circledata([A,B,C]);` で，3 点 A,B,C を通る円を描いたとき，できた円の中心は `Pointdata("1",[crABCcenter]);` で作図できる。

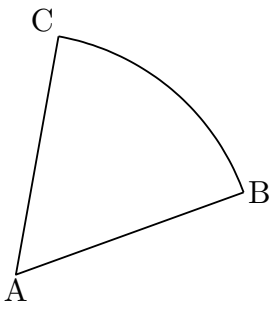
【例】A 中心，半径 AB，中心角 60°の弧を描く。

```
Circledata([A,B],["Rng=[0,pi/3]"]);
```

【例】A 中心，半径 AB，中心角 60°の扇型を描く。点　A,B,C を適当に取っておく。

```
th=arctan2(B-A);
C.xy=Rotatepoint(B,pi/3,A);
Circledata([A,B],[Assign("Rng=[th,th+pi/3]","th",th)]);
Listplot([B,A,C]);
Letter([A,"s","A",B,"e","B",C,"nw","C"]);
```





1 行目は，AB が x 軸となす角を arctan2 関数 によって求めている。

【例】弧を太く描く

```
Circledata([C,D],["dr,3","Rng=[0,pi/3]"]);
```

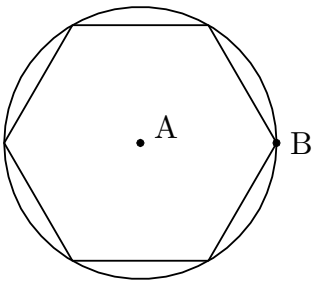
円は N が大きな値の正 N 多角形として描いている。option の [”Num=数値”] によってその細かさを指定できる。N の値が小さければ正多角形が描けることになる。

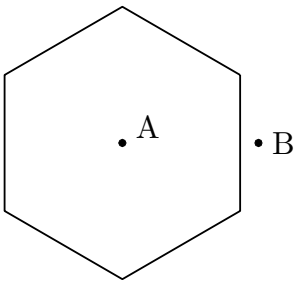
【例】A 中心，半径 AB の円と，その円に内接する正六角形

```
Circledata("1",[A,B]);
Circledata("2",[A,B],["Num=6"]);
```

ここで，同じ [A,B] を使うため，name を付与して区別する必要がある。(下図左)
また，頂点の位置を変えるのであれば，Rng= オプションを使う。(下図右)

```
Circledata("2",[A,B],["Num=6","Rng=[pi/6,13/6*pi]"]);
```





関数 Mkcircles()

機能 すべての幾何円の PD を作成

説明 Cinderella の「円を加える」ツール（3 種類いずれでも）で描いたすべての円をそのままプロットデータとする。たとえば，中心 A，円周上の点を B とした円を作ると，プロットデータ crAB が作成される。その後，インスペクタで点 B の識別名を変更（たとえば Q に）すると，プロットデータ名も変更される。円はすでに描かれていてもよい。

⇒ 関数一覧

関数 Ellipseplot(name, 点リスト , 定義域, options)

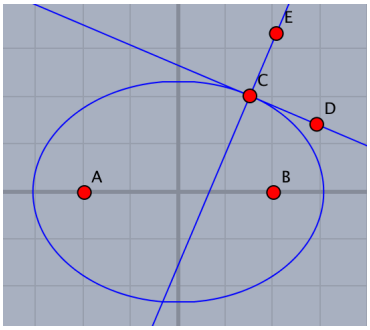
機能 焦点と通る点を与えて楕円を描く。

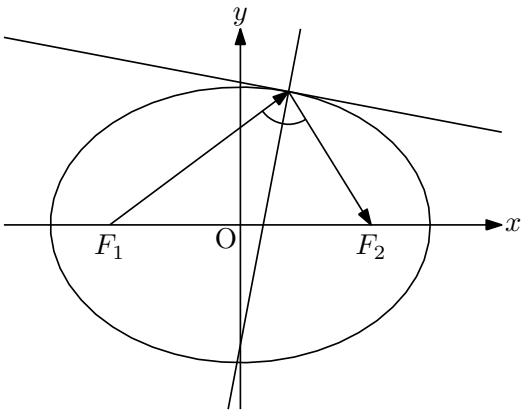
説明 点リストで 2 つの焦点と通る点を与える。点は Cinderella の幾何点が使える。
また，通る点のかわりに，焦点からの距離の和を実数で与えることもできる。
実際には，媒介変数表示 $x = a\cos\theta, y = b\sin\theta$ を，回転・平行移動して描いている。
定義域はこのときの t の定義域で，省略も可能。省略したときの初期値は [-5,5]

【例】点 A,B を焦点とする楕円を描く。
`Ellipseplot("1",[A,B,C]);` 点 C を通る楕円を描く。
`Ellipseplot("1",[A,B,4]);` 焦点からの距離の和が 4 である楕円を描く。
`Ellipseplot("1",[A,B,C],"[0,pi]");` 楕円の半分を描く。

【例】Cinderella の作図ツールを使う
作図ツールに，焦点と通る点で楕円を描くもの，点の極線を描くツールがある。（モードメニュー / 直線 / 点の極線）これを利用すると，楕円上にとった点をインシデントにできるので，インタラクティブに図を変更することができる。この Cinderella の作図機能と合わせて，一方の焦点から出た光が楕円上で反射して他方の焦点に至る，という図を次のようにして描くことができる。
まず，3 つの点，焦点 A,B と通る点 C を適当な位置に作図する。次に「焦点と通る点で決まる楕円」ツールを選び，点 A,B,C を順に指定すると，楕円が描かれる。
モードメニューの「直線」から「点の極線」を選び，点 C と楕円を順に指定すると接線が引かれる。
「垂線を加える」ツールを用いて，点 C で垂線，すなわち法線を引く。（下図）
「点を加える」ツールを用いて，接線，法線上に適当に点を取る。(D,E となったとする)
次のスクリプトを書いて実行すると，楕円に関して入射角と反射角が等しくなるように光が反射する様子を図にすることができる。

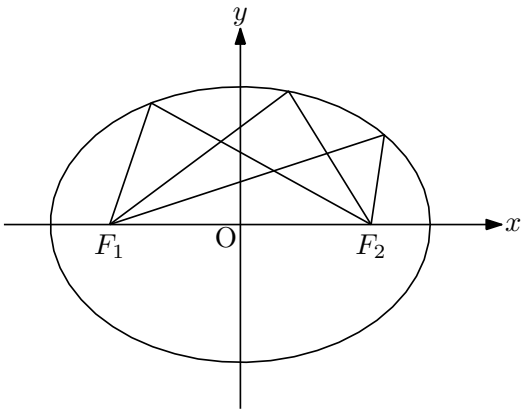
```
Ellipseplot("1",[A,B,C]);
Lineplot([C,D]);
Lineplot([C,E]);
Arrowdata([A,C]);
Arrowdata([C,B]);
Anglemark([A,C,B]);
Expr([A,"s2","F_1",B,"s2","F_2"]);
```





また，接線，法線を描かず，この楕円上に点 D,E,・・をとり（個数は任意）次のスクリプトを書けば，何本かの光線が一方の焦点を出て他方の焦点に集まる様子を描くことができる。

```
Ellipseplot("1",[A,B,C]);
Listplot([A,C,B]);
Listplot([A,D,B]);
Listplot([A,E,B]);
Expr([A,"s2","F_1",B,"s2","F_2"]);
```



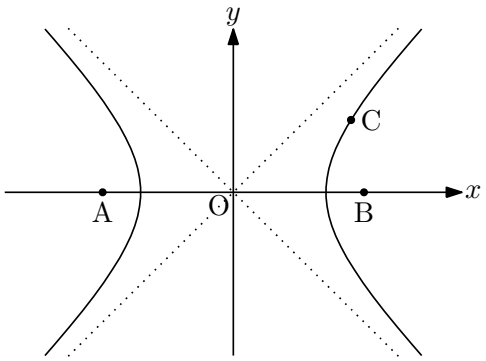
関数　Hyperbolaplot(name, 点リスト , 定義域, options)

機能　焦点と通る点を与えて双曲線を描く。

説明　点リストで 2 つの焦点と通る点を与える。点は Cinderella の幾何点が使える。
また，通る点のかわりに，焦点からの距離の差を実数で与えることもできる。

実際には，ハイパボリック関数を用いた媒介変数表示 $x = \cosh t, y = \sinh t$ を回転・平行移動している。
option として，”Asy=線種”を与えると，漸近線を指定した線種で表示する。 初期設定では漸近線は非表示。

【例】点 A,B を焦点とする双曲線を描く。
`Hyperbolaplot("1",[A,B,C]);` 点 C を通る双曲線を描く。
`Hyperbolaplot("1",[A,B,2]);` 焦点からの距離の差が 2 の双曲線を描く。
`Hyperbolaplot("1",[A,B,C],["Asy=do"]);` 漸近線を点線で描く。



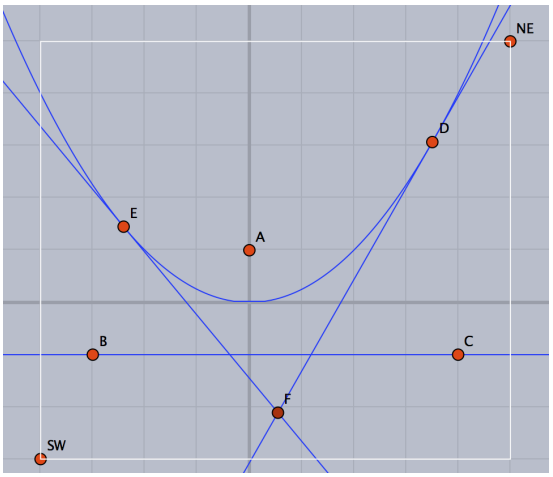
⇒ 関数一覧

関数 Parabolaplot(name，点リスト , 定義域, options)
機能 点リスト [A,B,C] で示された焦点，準線で決まる放物線を描く。
説明 焦点 A と準線 BC で決定する放物線を描く。
実際には，2 次関数 $y = x^2$ のグラフを回転・平行移動して描いており，定義域は，$y = x^2$ での定義域と考えてよい。定義域は省略することもできる。省略したときの初期値は [-5,5]

【例】点 A を焦点，直線 BC を準線とする放物線を描く
`Parabolaplot("1",[A,B,C]);`
定義域を $-4 \leq x \leq 4$ とする。
`Parabolaplot("1",[A,B,C],"[-4,4]");`
点 (0,1) を焦点，直線 $y = -1$ を準線とする放物線を描く
`Parabolaplot("1",[[0,1],[-1,-1],[1,-1]]);`

【例】放物線上の 2 点で引かれた接線と放物線で囲まれた領域を斜線で描く。
Cinderella の作図ツールに，焦点と準線で放物線を描くものがある。また，点の極線を描くツールがある。（モードメニュー / 直線 / 点の極線）これを利用すると，放物

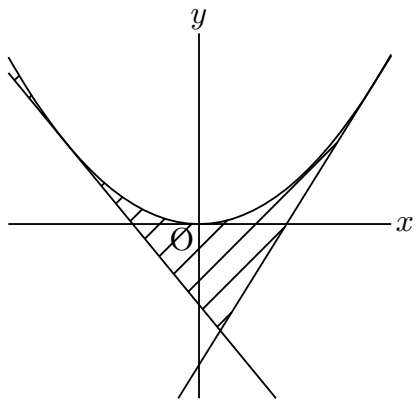
線上にとった点をインシデントにできるので，インタラクティブに図を変更することができる。この Cinderella の作図機能と合わせて，次の手順で図を描く。
まず，焦点 A(0,1) と準線 $y=-1$：BC を作図する。次に「焦点と準線で決まる放物線」ツールを選び，点 A と直線 BC を指定すると，放物線が描かれる。方程式では $y=\dfrac{1}{4}x^2$ の放物線である。
次に，放物線上に点 D,E をとる。Cinderella の作図機能を用いているので，この 2 点は放物線上だけを動かすことができる。（インシデント）
モードメニューの「直線」から「点の極線」を選び，点 D と放物線，点 E と放物線を順に指定すると接線が引かれる。その交点に点を取る。



以上で作図ができたので，次のスクリプトを書いて実行する。

```
Parabolaplot("1",[A,B,C]);
Lineplot([D,F]);
Lineplot([E,F]);
Hatchdata("1",["iii"],[["gr1para","s"],["lnEF","n"],["lnDF","n"]]);
```

これで，次図ができる。このあと，文字などは適当に追加する。



なお，Cinderella の作図ツールで放物線を描かず，焦点 A と準線上の点 B,C だけを用意して，次のスクリプトで描くこともできる。

```
Parabolaplot("1",[A,B,C]);
Putoncurve("D","gr1para");
Putoncurve("E","gr1para");
Tangentplot("1","gr1para","x="+D.x);
Tangentplot("2","gr1para","x="+E.x);
Hatchdata("1",["iii"],[["gr1para","s"],["lntn1","n"],["lntn2","n"]]);
```

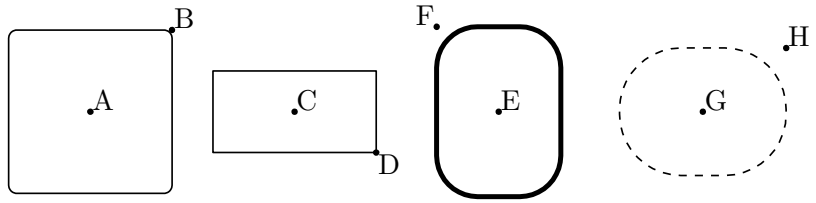
関数 Ovaldata(name, 点リスト,options)

機能 角を丸くした矩形を描く

説明 中心と対角の 1 点を指定し，角を丸くした矩形を描く
options は，角の落とし具合と線種など。 初期設定は 0.2

【例】いくつかの例を示す。

```
Ovaldata("1", [A,B]);
Ovaldata("2", [C,D],[0]);
Ovaldata("3", [E,F],[1,"dr,3"]);
Ovaldata("4", [G,H],[1.5,"da"]);
```



1.2.4 関数のグラフ

関数 Plotdata(name , 式 , 変数と定義域 , options)

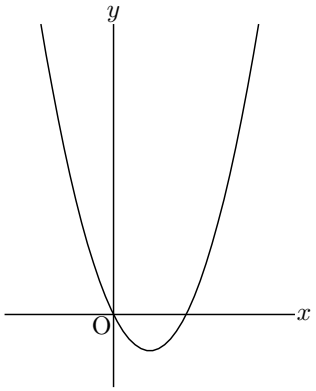
機能 関数のグラフを描く。プロットデータの名前は，gr

説明 式で表された関数のグラフを，指定された定義域で描く。
式，定義域は ” ” でくくって文字列とする。定義域は x=に続いてリストで指定。
options は次の通り。

線種	”dr, n”, ”da,m,n” , ”do,m,n”
”Num=数値”	描画時の分割数
”Dis=数値”	値が指定数値以上ジャンプする場合は不連続点とみなす。
”Exc=数値リスト	リストで示された点は除外する。
”Exc=関数”	関数の零点は除外する。
”Color=RGB”	色指定。RGB は CMYK でもよい。

【例】2 次関数 $f(x) = x^2 - 2x$ のグラフを定義域指定なしで描く。

```
Plotdata("1","x^2-2*x","x");
```

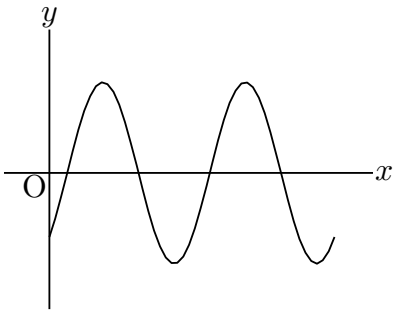



```
Plotdata("1","x^2-2*x","x",["Color=[1,0,0]"]);
```

とすると赤で描かれる。

【例】三角関数 $2\sin\left(2x - \dfrac{\pi}{4}\right)$ のグラフを，定義域 $0 \leq x \leq 2\pi$ で描く。

```
Plotdata("3","2*sin(2*x-pi/4)","x=[0,2*pi]");
```

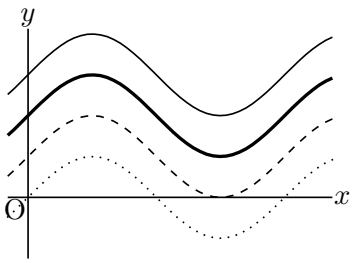


CindyScript では，plot(式 , 定義域); で描くが， KETCindy を用いるときは，CindyScript の plot 関数のかわりに，この Plotdata を使えばよい。
軸に数字を入れるのであれば，Letter() を用いる。

options の使用例

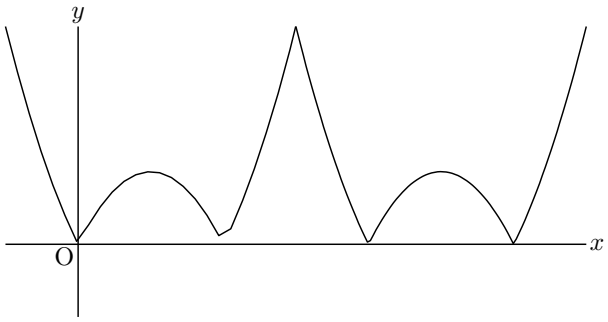
`Plotdata("1","sin(x)+3","x");`	初期設定
`Plotdata("2","sin(x)+2","x",["dr,2"]);`	同じく，太さ 2 で描く
`Plotdata("3","sin(x)+1","x",["da"]);`	同じく，破線で描く
`Plotdata("4","sin(x)","x",["do"]);`	同じく，点線で描く

結果は次図上から。



Num=分割数の指定

グラフの描画は，区間を分割して関数値をとり，各点を結ぶという通常の方法によっている。N の指定はこの分割数の指定である。 初期設定は 50。思うような結果が得られない場合はこの値を大きく指定するとよい。下図左は 初期設定，右は Num=200。



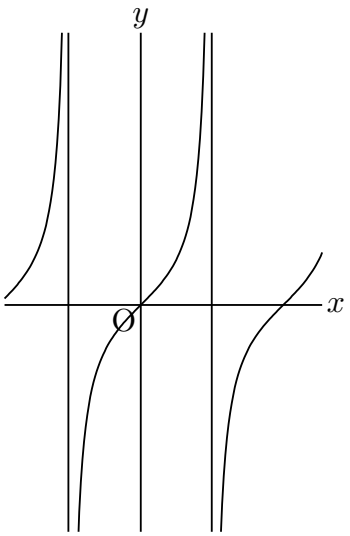
不連続点の指定

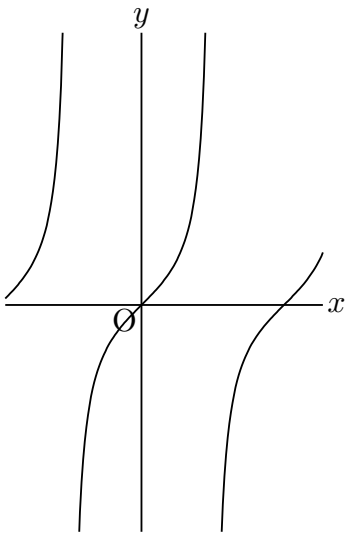
Dis オプションにより，値がジャンプする不連続点を線で結ばないようにする。Num オプションと合わせて使うと効果が上がる。

【例】$f(x)$ =tan$x$ のグラフは，そのままではあたかも漸近線が描かれたようになるが，これは，不連続点の前後をそのまま結んでいるためである。（下図左）

```
Plotdata("1","tan(x)","x",["Num=200","Dis=50"]);
```

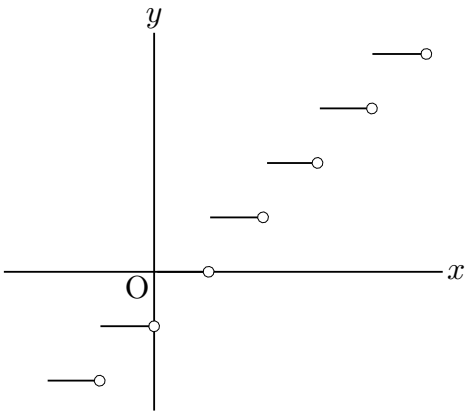
のように，”Dis” オプションを使えば余分な線が描かれなくなる。（下図右）





【例】ガウス記号 $[x]$ で表される関数（床関数：floor()）のグラフ。

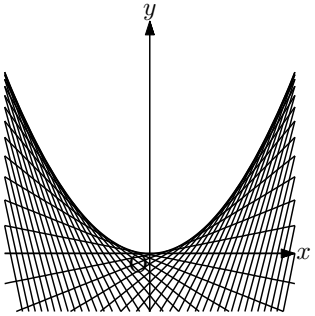
```
Plotdata("1","floor(x)","x",["Num=100","Dis=0.9"]);
Drwxy();
repeat(7,s,start -> -2,
Pointdata(text(s+3),[s+1,s],["Inside=0","Size=3"]);
);
```



なお，ここで，Pointdata() の name を `text(s+3)` としているのは，s が -2 から始まるので，負の数が使えない name を 1 から始まるようにするためである。

関数に文字係数がついており，文字係数の値を変化させながらグラフを描くには，Assign を使うか，Defvar で変数を定義する。

【例】直線 $y = bx - b^2$ の係数 b を変化させて描き，包絡線をうかびあがらせる。




```
repeat(50,t,
 cb=t/5-5;
 Plotdata(text(t),Assign("b*x-b^2","b",cb),"x");
);
```

または

```
Defvar("b");
repeat(50,t,
 b=t/5-5;
 Plotdata(text(t),"b*x-b^2","x");
);
```

関数 Implicitplot(name, 式,x の定義域,y の定義域, options)

機能 陰関数のグラフを描く。

説明 陰関数の式を与えてグラフを描く。式，定義域とも文字列。

options は，”r”,”m”,”Wait=n” が指定できる。Wait の初期値は 10。

”r”,”m”に関しては，オプションなしのとき

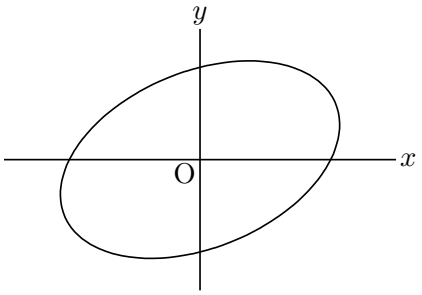
i) データファイルがなければ，新しく作る

ii) データファイルが既にあればそれを読み込む

”m”のとき，強制的にデータファイルを作り直す。
”r” のとき，すでにあるデータファイルを読み込む。

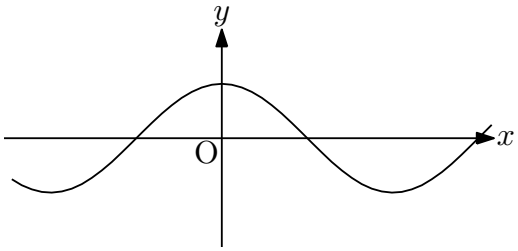
【例】楕円を描く。

```
Implicitplot("1","x^2-x*y+2*y^2=4","x=[-3,3]","y=[-2,2]");
```

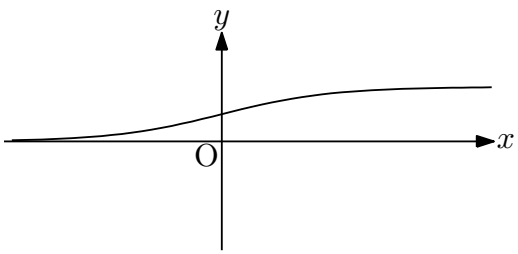


関数 Deqplot(name, 式，変数名，初期値，options)
機能 微分方程式の解曲線を描く
説明 微分方程式と初期値を与えて解曲線を描く。

【例】$y'' = -y$ で，初期値が $x = 0$ のとき $y = 1, y' = 0$ の解曲線
`Deqplot("1","y''=-y","x",0, [1,0]);`
注） 微分記号のプライムは，シングルクォートまたはバッククォート。



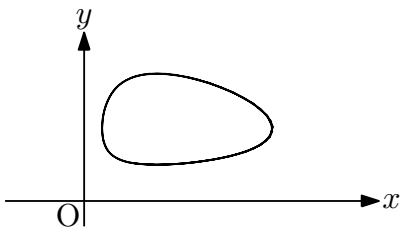
【例】$y' = y * (1 - y)$ で，$x = 0$ のとき，$y = 0.5$ の解曲線
`Deqplot("2","y'=y*(1-y)","x",0, 0.5,["Num=100"]);`



【例】$[x, y]' = [x(1 - y), 0.3y(x - 1)]$ で，変数は t，$t = 0$（区間の左端）のときの x, y

の値が 1 と 0.5 であるときの解曲線

```
Deqplot("3","[x,y]'=[x*(1-y),0.3*y*(x-1)]","t=[0,20]",
[1,0.5],["Num=200"]);
```



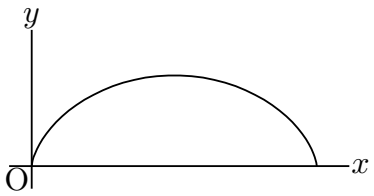
関数 Paramplot(name , 式 , 変数と定義域,options)

機能 媒介変数表示の曲線を描く。

説明 式は””でくくった媒介変数表示のリストで与える。
定義域も ” ” でくくって文字列とし，t=に続いてリストで指定する。
options は線種が有効

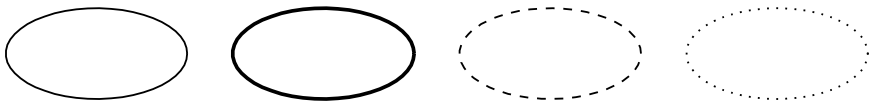
【例】サイクロイド曲線を描く。

```
Paramplot("1","[t-sin(t),1-cos(t)]","t=[0,2*pi]");
```



【例】options の使用例。左から， 初期設定，太線，破線，点線の楕円

```
Paramplot("1","[2*cos(t)-5,sin(t)]","t=[0,2*pi]");
Paramplot("2","[2*cos(t),sin(t)]","t=[0,2*pi]",["dr,2"]);
Paramplot("3","[2*cos(t)+5,sin(t)]","t=[0,2*pi]",["da"]);
Paramplot("4","[2*cos(t)+10,sin(t)]","t=[0,2*pi]",["do"]);
```

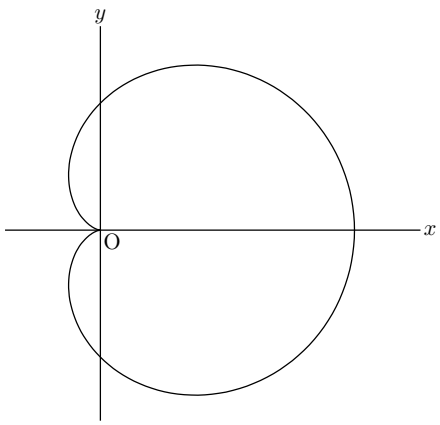


関数 Polarplot(name , 式 , 変数と定義域,options)

機能 極座標表示 $r = f(\theta)$ の曲線を描く。

【例】カージオイド曲線を描く。

```
Polarplot("1","2*(1+cos(t))","t=[0,2*pi]",["Num=200"]);
```



関数　Periodfun(定義式, 周期,options)

機能　周期関数のグラフを描く。戻り値は Maxima 形式の式と period のリスト。

説明　周期関数の式を定義してグラフを描く。定義式は，関数式（文字列），区間，分割数のリスト。

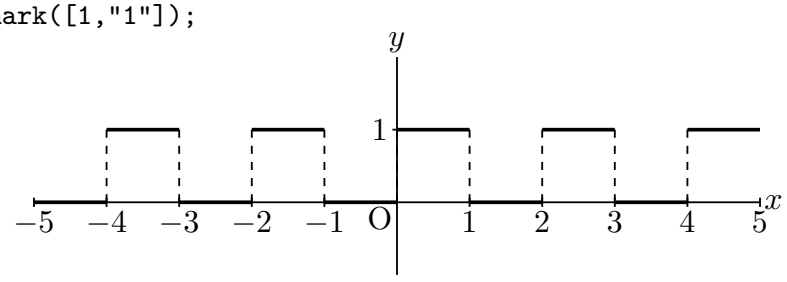
この関数固有のオプションは ”Con=” で，不連続点を線で結ぶか否かと，その時の色。初期設定は破線。結ばない場合は ”Con=n”，色指定は線種に続いてコンマで区切って指定する。たとえば，”Con=do,Color=red”。

周期（描画回数）は，数またはリストで指定する。周期が m のとき，2m+1 周期分描かれる。

注意）関数は左右対称な定義域 [-a,a] で定義すること。

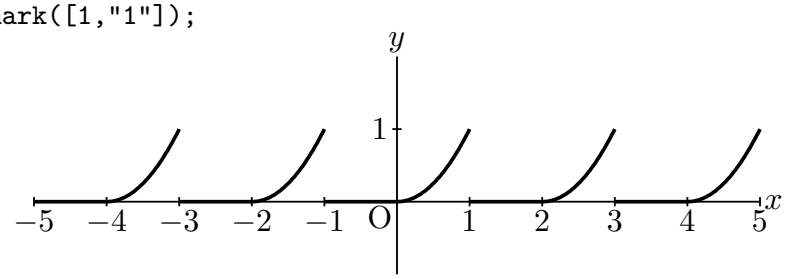
【例】矩形関数のグラフ

```
defL=["0",[-1,0],1,"1",[0,1],1];
Periodfun(defL, 2,["dr,2"]);
memori=apply(-5..5,x,[x,text(x)]);
memori=flatten(remove(memori,[[0,"0"]]));
Htickmark(memori);
Vtickmark([1,"1"]);
```



一方を放物線にした場合

```
defL=["0",[-1,0],1,"x^2",[0,1],50];
Periodfun(defL, 2, ["Con=n","dr,2"]);
memori=apply(-5..5,x,[x,text(x)]);
memori=flatten(remove(memori,[[0,"0"]]));
Htickmark(memori);
Vtickmark([1,"1"]);
```



関数　Fourierseries(name, 係数, 周期, 項数)

機能　フーリエ級数のグラフを描く。

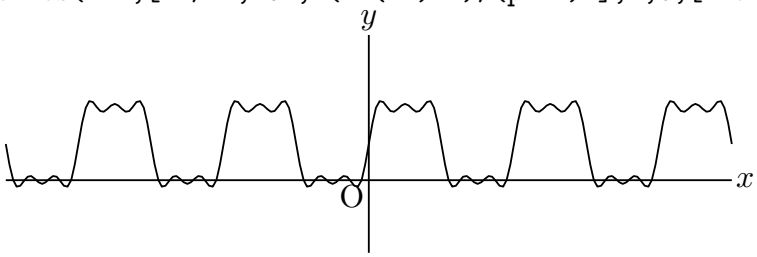
説明　$a_0 + \sum_{n=1}^{\infty}(a_n \cos nx + b_n \sin nx)$ を描く。係数は a_0, a_n, b_n のリストで，それぞれの要素は文字列とする。

この関数固有のオプションは ”Con=y/n” で，不連続点を破線で結ぶか否か。 初期設定は y。

矩形波のフーリエ近似のような場合は，オプションとして，[”Num=200”]（100 以上の値を指定）をつけるとよい。

【例】矩形波のフーリエ近似

```
Fourierseries("1",["1/2","0","(1-(-1)^n)/(pi*n)"],2,6,["Num=200"]);
```



関数　Tangentplot(name , PD , 位置 , options)

機能　接線を描く。プロットデータの名前は，lntn

説明　曲線 PD の指定した位置での接線を描く。位置は ”x=n” で指定する。

使用例は Parabolaplot の例を参照。

⇒ 関数一覧

1.2.5 文字

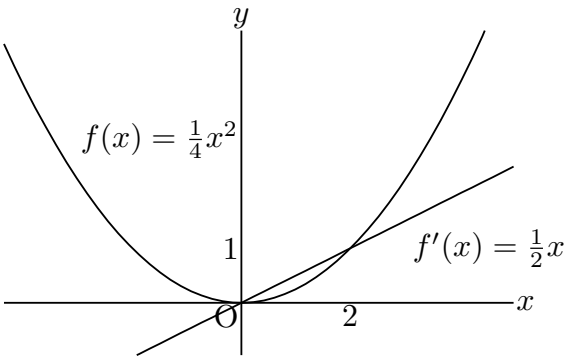
関数 Expr([座標 , 位置 , 文字列],option)

機能 TEX 記法の文字列を与えて数式を書く。

説明 Letter で文字列の前後に$ $をおくのと同じ。
導関数の記号は, ′（シングルクォート）を用いる。
複数の箇所に文字を書く場合は，Letter() と同様，引数をリストにして与える。
option は フォントサイズで，[”size=32”] のように指定する。

【例】$f(x)=\dfrac{1}{4}x^2$ とその導関数 $f'(x)=\dfrac{1}{2}x$ の式，軸上に必要な数を入れる。

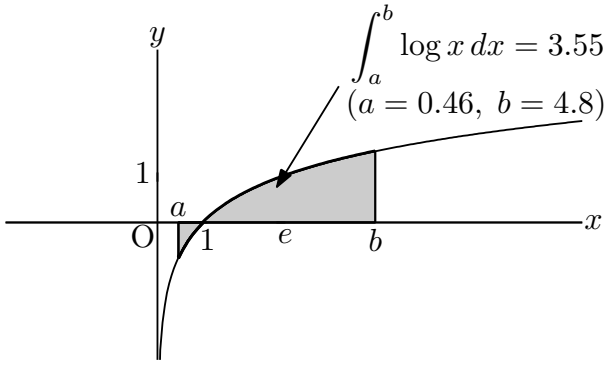
```
Expr([[-3,3],"e","f(x)=\frac{1}{4} x^2",[3,1.5],"s2e2",
"f‘ (x)=\frac{1}{2}x",[2,0],"s","2",[0,1],"w","1"]);
```



【例】対数関数の定積分の記号および積分値を図に書き込む。

```
Expr([Q+[0.2,0],"ne","\displaystyle \int_a^b \log x\,dx="+
text(L.x*(log(L.x)-1)-G.x*(log(G.x)-1)) ]);
```

`L.x*(log(L.x)-1)-G.x*(log(G.x)-1)` は，点 L,G(図の a,b) をドラッグして積分範囲を決めるようにしているので，そこから計算した値。
矢線は `Arrowdata(Q,P);` で表示している。矢線の始点が Q



関数　Exprrot(座標 , 向き , 方向，文字列,option)

機能　TEX 記法の文字列を与えて傾いた数式を書く。

説明　「座標」の位置に，指定された向きで数式を書く。

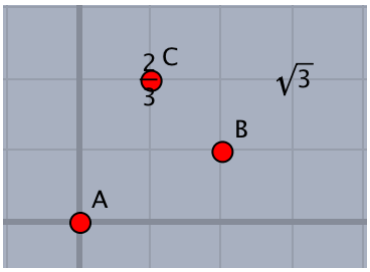
向きはベクトルで与える。

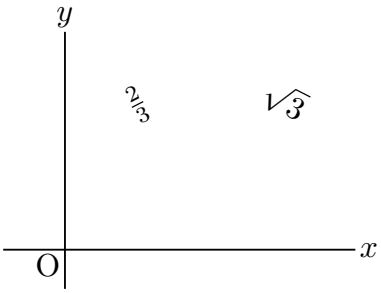
方向は t (ベクトルと同じ向き) と n（ベクトルと垂直な向き） の両方を数字つきで与える。ただし，方向はオプション。

座標，向きとも，Cinderella で作図した幾何点を用いることができる。

option は フォントサイズで，[”size=32”] のように指定する。

```
Exprrot([3,2],[2,-1],"t0n1","\sqrt{3}");
Exprrot(C,B-A,"\frac{2}{3}");
```





⇒ 関数一覧

関数　Letter([位置, 方向, 文字列],option)

機能　文字列を表示する

説明　「位置（座標)」と方向で指定された場所に文字を書き込む。

位置（座標）は点の名前で指定することもできる。

場所は上下左右を東西南北で表し， n/s/w/e/c の方向で表す。c は中央。

指定位置からの距離を，数値で与えることもでき，e2, e3 は e より少し離して置く。

複数の文字列をリストの形にして渡すことができる。

注）導関数の記号 ′ は，数式モード（$ ではさむ）で ′（シングルクォート）を用いる。

option は フォントサイズで，[”size=32”] のように指定する。

文字列が一つのときは，

Letter(位置, 方向, 文字列,option)

にできる。

【例】

座標 (2,1) の南東にＰを表示

```
Letter([[2,1] ,"se","P"]);
```

点Ｃを中央としてＣを表示

```
Letter([C ,"c", "C"]);
```

点 A の南西に A，E の南に数式を表示

```
Letter([A,"sw","A",E,"s","$ f(x)=\frac{1}{4} x^2 $"]);
```

関数　Letterrot(座標, 方向ベクトル, 移動量, 文字列,option)

機能　文字列を回転して表示する

説明　座標で示された位置に，方向ベクトルで指定された向きに回転して文字を書き込む。
第 3 引数は微小移動量で，略すこともできる。
option は フォントサイズで，[”size=32”] のように指定する。

```
Letterrot(C,B-A,"t2n5","AB");
```

移動量を略して

```
Letterrot(C,B-A,"AB");
```

とすることもできる。この場合は，微小な移動はされない。

⇒ 関数一覧

1.2.6 マーキング

関数　Anglemark(name, 点リスト , options)

機能　点リストで示された角に弧の形状の角の印をつける。

説明　Listplot() などと同様，点リストが点名の場合は name は省略できる。弧を描かず文字だけを入れる場合は options に ”nodisp” を指定する。
options は次の通り。
数値角の印の大きさ。 初期設定は 1
線種”dr, n”, ”da,m,n” , ”do,m,n”
”Expr=文字” または ”Letter=文字” : 文字を入れる
”Expr=位置 , 文字”: 位置を指定して文字を入れる。位置は頂点からの距離。

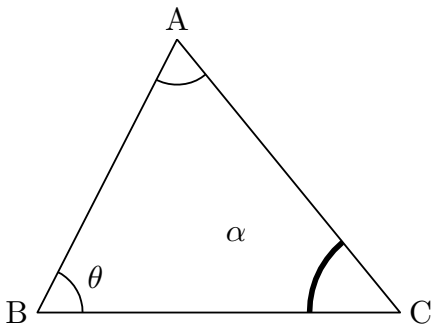
【例】三角形の内角に印をいれ，文字を書き込む。（下図左）

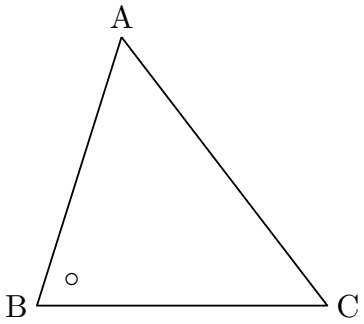
```
Listplot([A,B,C,A]);
Letter([A,"n1","A",B,"w1","B",C,"e1","C"]);
Anglemark([B,A,C]);
Anglemark([C,B,A],["Expr=\theta"]);
Anglemark([A,C,B],[2,"dr,3","Expr=2,\alpha"]);
```

【例】三角形の内角に弧を描かず，○ だけ書き込む。（下図右）

```
Listplot([A,B,C,A]);
Letter([A,"n1","A",B,"w1","B",C,"e1","C"]);
```

```
Anglemark([C,B,A],["Expr=\circ","nodisp"]);
```





※角の印には平行四辺形の形状のものもある。Paramark() を参照のこと。

関数　Paramark(name, 点リスト , options)

機能　点リストで示された角に平行四辺形の形状の角の印をつける。

説明　Listplot() などと同様，点リストが点名の場合は name は省略できる。

options は次の通り。

数値角の印の大きさ。 初期設定は 1

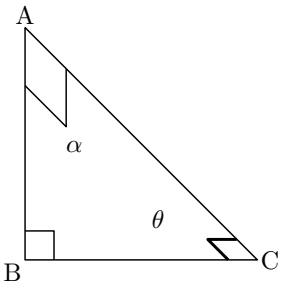
線種”dr, n”, ”da,m,n” , ”do,m,n”

”Expr=文字”: 文字を入れる

”Expr=位置 , 文字”: 位置を指定して文字を入れる。位置は頂点からの距離。

【例】三角形の内角に印をいれ，文字を書き込む。

```
Listplot([A,B,C,A]);
Paramark([A,B,C]);
Paramark([C,A,B],[3,"Expr=\alpha"]);
Paramark([B,C,A],["dr,2","Expr=2,\theta"]);
```



※角の印には弧の形状のものもある。Anglemark() を参照のこと。

関数　Bowdata(name, 点リスト , options)

機能　弓形を描く

説明　点リストで与えられた 2 点を結ぶ弓形を描く。Listplot() などと同様，点リストが点名の場合は name は省略できる。

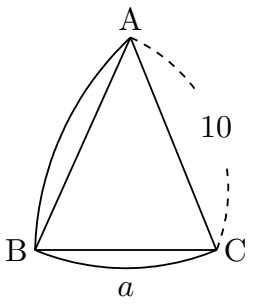
2 点を反時計回りに回る方向に弓形を描く。

options は，[曲がり , 空白サイズ, 文字, 線種]

曲がりは弧の曲がり具合の指定。 初期設定は 1
空白サイズは中央にあける空白の大きさ
文字は，”Expr=文字”
また，”Expr=微小移動 , 文字”で位置を指定して文字を入れる。
微小移動は tn
t は線分方向の微小移動。移動量は数字をつける。正負が可。
n は線分と垂直方向の微小移動

【例】三角形 ABC の各辺に弓形マークをつけ記号を入れる。

```
Listplot([A,B,C,A]);
Letter([A,"n1","A",B,"w1","B",C,"e1","C"]);
Bowdata([A,B]);
Bowdata([B,C],[1,"Expr=t0n3,a"]);
Bowdata([C,A],[2,1.2,"Expr=10","da"]);
```

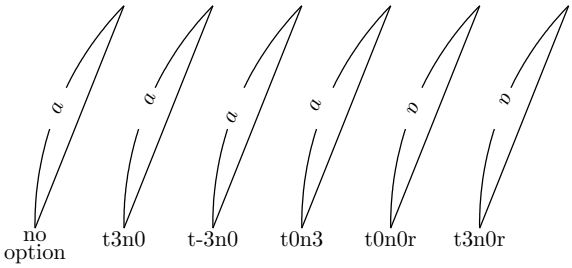


これに加え，文字を回転して表示する方法がある。ただし，Cinderella の画面には反映されない。文字をを回転するには次のように書く。

”Exprrot=微小移動 , 文字”

微小移動の最後に r をつけると，上下反転する。
以下にいくつか例を示す。

```
Bowdata([B,A],[1,1,"Exprrot=a"]);
Bowdata([D,C],[1,1,"Exprrot=t3n0,a"]);
Bowdata([F,E],[1,1,"Exprrot=t-3n0,a"]);
Bowdata([H,G],[1,1,"Exprrot=t0n3,a"]);
Bowdata([L,K],[1,1,"Exprrot=t0n0r,a"]);
Bowdata([N,M],[1,1,"Exprrot=t3n0r,a"]);
```



関数 Drawsegmark(name, リスト,options）または Segmark(name, リスト,options）

機能 線分に印をつける

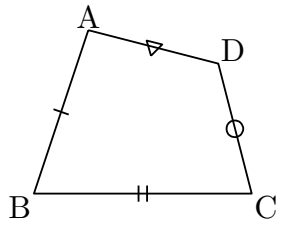
説明 リストで与えられた 2 点を端点とする線分に印をつける。印には 4 種類がある。
options は

Type=n ：印の種類 n=1〜4

Width　：二本線のときの線の幅

【例】四角形 ABCD を描き線分に印をつける。

```
Listplot([A,B,C,D,A]);
Segmark("1",[A,B],["Type=1"]);
Segmark("2",[B,C],["Type=2","Width=1.5"]);
Segmark("3",[C,D],["Type=3"]);
Segmark("4",[D,A],["Type=4"]);
```

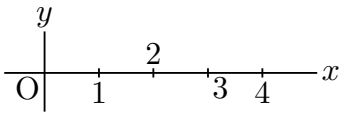


関数　Htickmark([横座標 , 方向 , 文字])

機能　横軸に目盛と文字を書く。

説明　引数は位置（横座標），方向，文字。複数点の情報を [ ] 内にまとめて記入できる。方向を省略すると ”s1”になる。微調整は描画面には反映されないので，PDF にして確認する。目盛の長さは Setmarklen() で設定できる。

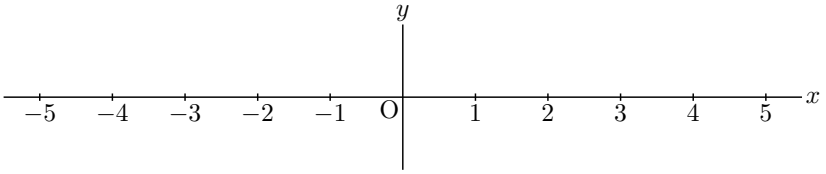
【例】 方向指定の例：`Htickmark([1,"1",2,"n1","2",3,"se","3",4,"4"]);`



【例】-5 から 5 までの目盛を打つ。Cindyscript のリスト処理を使って，次のように引数のリストを作って渡す。

```
memori=apply(-5..5,x,[x,text(x)]);
memori=flatten(remove(memori,[[0,"0"]]));
Htickmark(memori);
```

１行目，apply のカッコ内の -5..5 でリスト [-5,-4,-3,-2,-1,0,1,2,3,4,5] ができる。それを用いて，apply で [数, 数の文字] からなるリストができる。text(x) は x を文字にする関数。2 行目で，このリストから，[0,”0”] を除き，リストを平滑化する。 結果は次のようになる。



関数　Vtickmark([横座標 , 方向 , 文字])

機能　縦軸に目盛と文字を書く。

説明　Htickmark と同様。縦軸に目盛を書く。方向を省略すると ”w1”になる。

【例】点 (0, 1), (0, 2) の西側に 1, 2 を表示する。

```
Vtickmark([1,"1",2,"2"]);
```

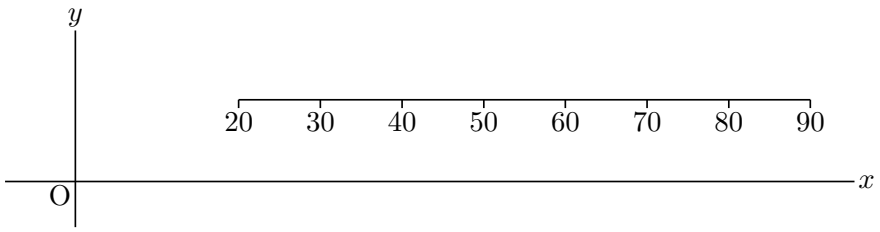
関数　Rulerscale(始点 , 横軸目盛 , 縦軸目盛)

機能　目盛を打つ

説明　始点の位置を縦横の起点として目盛りを打つ。目盛はリストで与える。[”r”,a,b,c,d] の形式では, a から b まで c 間隔で, 倍率 d の目盛を打つ。[”f”,n1,”str”,n2,”str”,・・] の形式では, n と”str”がセットで, n の位置に”str”を書く。ただし, 位置は Cinderella の描画面の原点を 0 とする。
Listplot() とともに用いると，座標軸とは異なる線分に目盛を打つことができる。
Framedata() とともに用いると矩形に目盛を打つことができる。

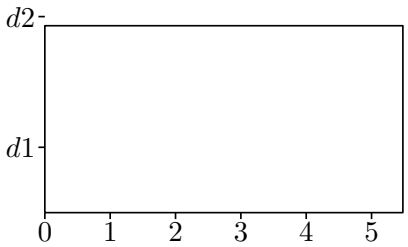
【例】x 軸上の (2,1) から (9,1) まで線分を引き，1 目盛を 10 として目盛を打つ。

```
Listplot("1",[[2,1],[9,1]]);
Rulerscale([2,1],["r",2,9,1,10],[]);
```



【例】A を原点に置いた矩形枠を描き，横に 0,1,2,3,4,5，縦に d1，d2 の目盛を打つ。

```
Framedata("1",[A,B],["corner"]);
Rulerscale(A,["r",0,5,1],["f",1,"d1",3,"d2"]);
```



⇒ 関数一覧

1.3 プロットデータの操作

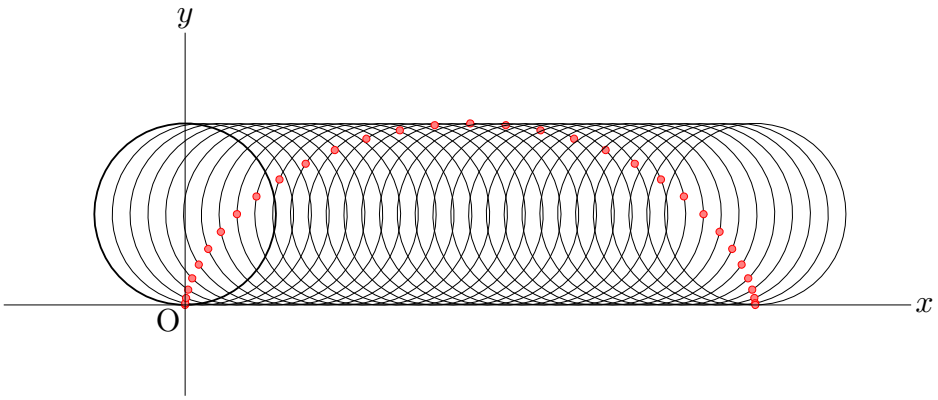
関数　Drawfigures(or Drwfigs)(name , データリスト, オプションリスト)

機能　複数のプロットデータのスタイル（オプション）をリストで与えて描画する

説明 複数のプロットデータをまとめて扱う。たとえば，円と，円周上の点の 2 つのプロットデータをまとめて扱えば，平行移動や回転で，それらのプロットデータをまとめて平行移動や回転ができる。

【例】サイクロイドの図を描く。

```
opcr=["dr"];
oppt=["Size=2","Color=red"];
Circledata("1",[[0,1],[0,0]],opcr);
Pointdata("1",[0,0],oppt);
ad1=["cr1","pt1"];
dt=2*pi/32;
opcr=["dr,0.3"];
nn=32;
forall(1..nn,
  t=dt*#;
  Rotatedata(2,ad1,-t,[[0,1],"nodisp"]);
  Translatedata(2,"rt2",[t,0],["nodisp"]);
  Drawfigures(text(#),["tr2_1","tr2_2"],[opcr,oppt]);
);
```



最初に，まとめて平行移動や回転をするデータをリスト化しておく。また，各データのオプションをリストとして与えていることに注意。こうすることで，サイズや色などのスタイルを元のスタイルに合わせることができる。

⇒ 関数一覧

関数 Changestyle(PD リスト, options)

機能 描画オプションを変更する

説明 複数の図形の描画オプションを一括して変更する。

【例】線分 AB，円 AB の線を破線にして TEX に書き出さないようにする。

```
Changestyle(["sgAB","crAB"],["da","notex"]);
```

関数　Invert(PD)

機能　プロットデータを逆順にする

関数　Joincrvs(name, プロットデータのリスト, options)

機能　隣接する曲線プロットデータ のリストを繋いで 1 本の曲線を作る。

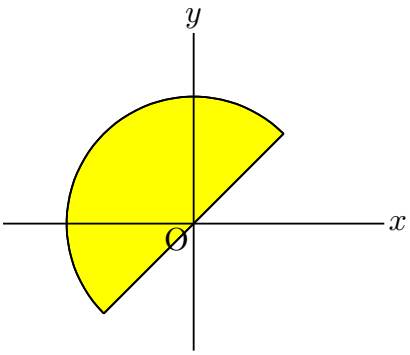
説明　曲線のリストは隣接する順番で指定する。

options は線種”dr, n”, ”da,m,n” , ”do,m,n”

【例】線分 $y = x$ $(-\sqrt{2} \leq x \leq \sqrt{2})$ と半円で得られる閉曲線を描いて黄色で塗る。

点 A は原点に，点 B は適当なところに作図しておく。

```
Plotdata("1","x","x=[-sqrt(2),sqrt(2)]");
B.xy=[sqrt(2),sqrt(2)];
Circledata("2",[A,B],["Rng=[pi/4,pi/4*5]"]);
Joincrvs("1",["gr1","cr2"]);
Shade(["join1"],["Color=yellow"]);
```



関数　Partcrv(name, A, B, プロットデータ, options)

機能　曲線プロットデータ上の点 A, B の間の部分曲線を描く。

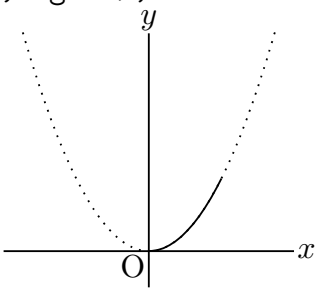
説明　2 点 A, B の順序は曲線の向きと同一であること。曲線の向きは，$y = f(x)$ のグラフでは x 座標が増加する向き。

options は線種”dr, n”, ”da,m,n” , ”do,m,n”

【例】放物線を点線で描き，一部を実線で描く。

`Plotdata("1", "x^2", "x", ["do"]);`（プロットデータの名前は gr1 となる）

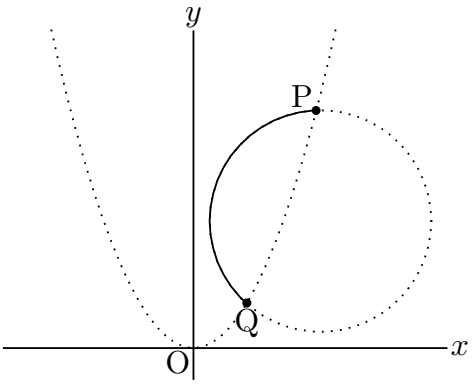
`Partcrv("1", [0,0], [1,1], "gr1");`



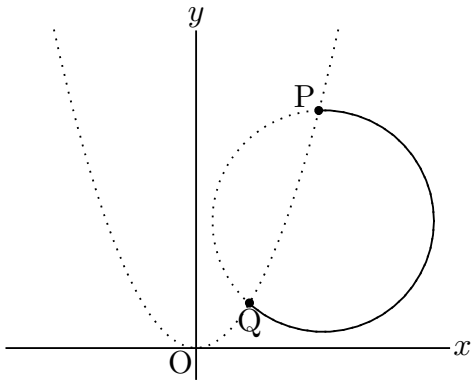
【例】円の一部を実線で描く。円のプロットデータは指定した円周上の点から反時計回りの順にできる。点 A は円の中心，B は円周上の点とする。点 P,Q は適当な位置に作図しておく。

```
Circledata([A,B], ["do"]);
Plotdata("1", "x^2", "x", ["do"]);
```

```
tmp=Intersectcrvs("crAB","gr1");
P.xy=tmp_1;
Q.xy=tmp_2;
Partcrv("1", P, Q, "crAB");
Partcrv("2", Q, P, "crAB");
```



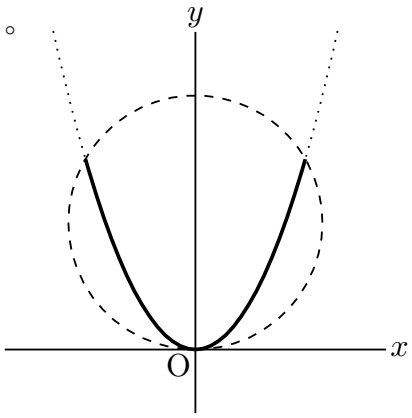
part1 の図



part2 の図

【例】放物線 $y = x^2$ が円で切り取られる部分を実線で描く。

```
Circledata("1",[[0,2],[0,0]],["da"]);
Plotdata("1","x^2","x",["do"]);
tmp=Intersectcrvs("cr1","gr1");
Partcrv("2",tmp_2,tmp_1,"gr1",["dr,2"]);
```



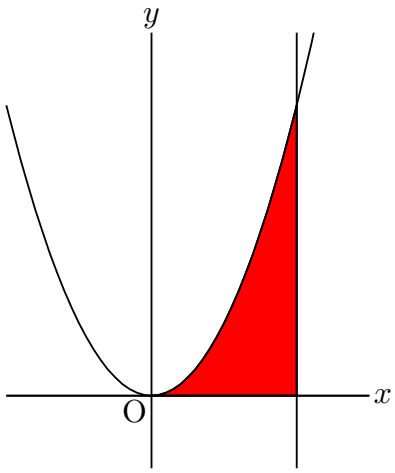
関数 Enclosing(name , PD リスト , [開始位置, 交点計算の許容限界 1, 2])

機能 複数の曲線から閉曲線を作る。

説明 開始位置は，最初と最後の曲線の交点が複数あるときに指定する。
開始点は近くに取ればよい。許容限界は，通常は指定しなくてよい。

【例】放物線と直線で囲まれる領域に色を塗るために Shade() を使う。

```
Plotdata("1","x^2","x");
Lineplot("1",[[0,0],[1,0]]);// axis x
Lineplot("2",[[2,0],[2,1]]);
Enclosing("1",["Invert(gr1)","ln1","ln2"],["nodisp"]);
Shade(["en1"],["Color=red"]);
```



注）閉曲線のとりかたでは，出発点を原点にした反時計回りまたは時計回りにすると
反時計回りで `Enclosing("1",["ln1","ln2","Invert(gr1)"]);`
時計回りで `Enclosing("1",["gr1","Invert(ln2)","Invert(ln1)"]);`

⇒ 関数一覧

関数 Hatchdata(name , 方向リスト , プロットデータ , options)

機能 閉曲線の内部に斜線を引く。

説明 引数は，曲線名，内部外部のパターンを与える”i”，”o” の文字列，閉曲線を与える曲線と領域の内部を定める方向のリストとオプション。

オプション（カッコ内はデフォルト値）

角度 (45)，間隔 (1)，`"Max=(20)"` 斜線の最大本数,

`"No=`**点リスト**`"` 点リストの点が選ばれているときは実行しない

`"File=y/m/n (n)"` データファイルを作るか

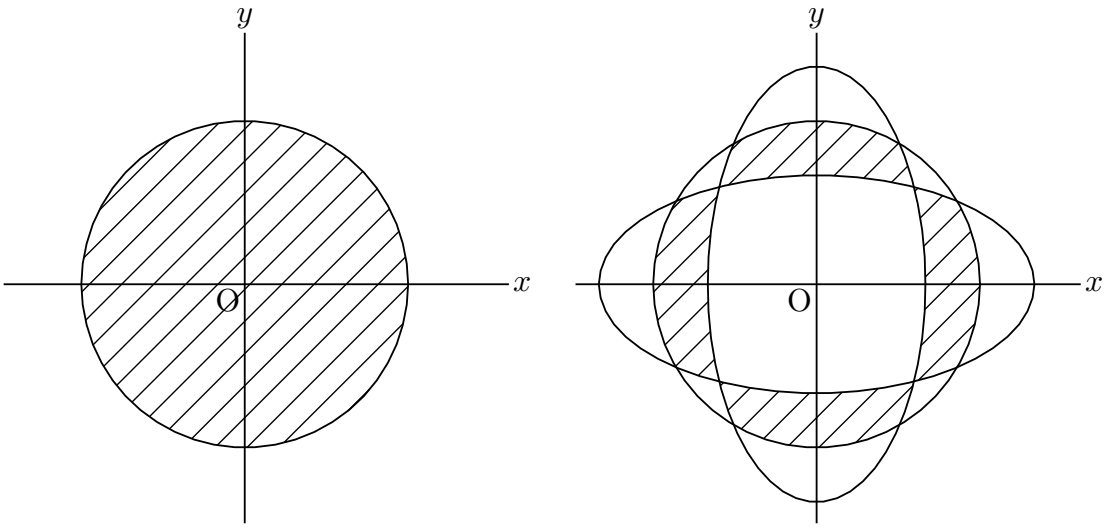
`"Check=`**点リスト**`"` 点リストの点が変更されていたら，ファイルを作り直す

【例】円の内部。(次図左)

```
Circledata([A,B],["dr"]);
Hatchdata("1",["i"],[["crAB"]],["dr,0.7"]);
```

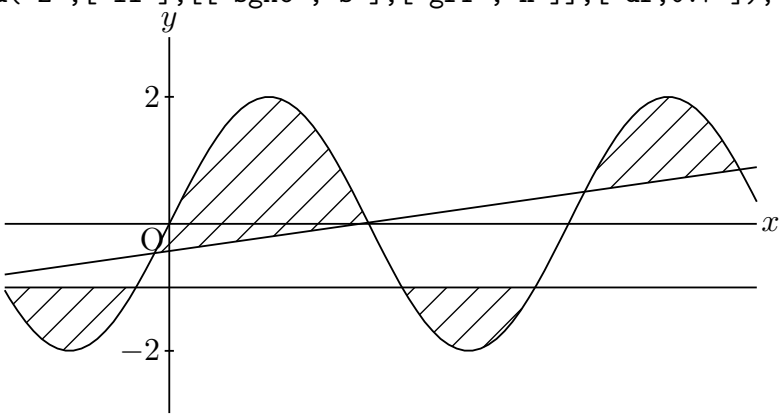
【例】3 つの閉曲線の内側・外側のパターンが同一である領域（次図右）

```
Circledata([A,B],["dr"]);
Paramplot("1","[4*cos(t),2*sin(t)]","t=[0,2*pi]");
Paramplot("2","[2*cos(t),4*sin(t)]","t=[0,2*pi]");
Hatchdata("1",["ioi"],[["crAB"],["gp1"],["gp2"]],["dr,0.7"]);
Hatchdata("2",["iio"],[["crAB"],["gp1"],["gp2"]],["dr,0.7"]);
```



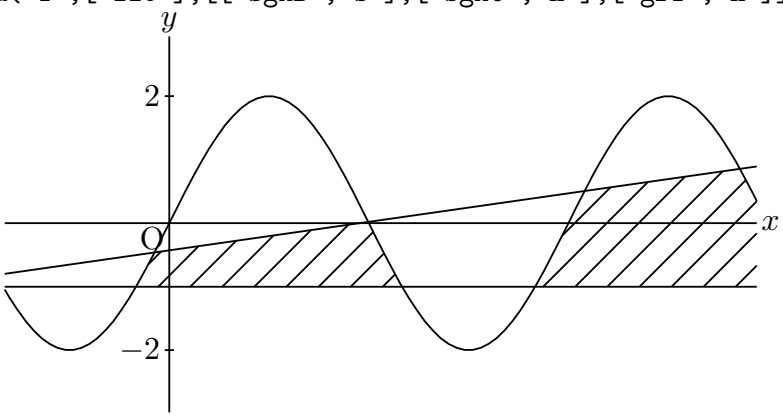
【例】複数の領域。

```
Plotdata("1","2*sin(x)","x=[-pi,3*pi]",["Num=100"]);
Listplot([A,B]);
Listplot([A,C]);
Hatchdata("1",["ii"],[["sgAB","n"],["gr1","s"]],["dr,0.7"]);
Hatchdata("2",["ii"],[["sgAC","s"],["gr1","n"]],["dr,0.7"]);
```



【例】複数の領域その 2。

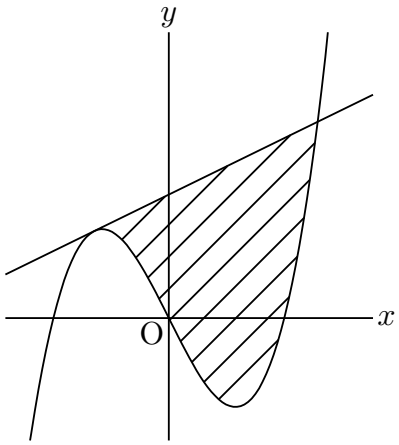
```
Plotdata("1","2*sin(x)","x=[-pi,3*pi]",["Num=100"]);
Listplot([A,B]);
Listplot([A,C]);
Hatchdata("1",["iio"],[["sgAB","s"],["sgAC","n"],["gr1","n"]]);
```



【例】3 次曲線と接線で囲まれた領域

点 A を原点付近に作図しておく。

```
Deffun("f(x)",["regional(y)","y=x^3-2*x","y"]);
Plotdata("1","f(x)","x",["Num=100"]);
Putoncurve("A","gr1");
coef=Derivative("f(x)","x",A.x);
Defvar(["coef",coef]);
Deffun("g(x)",["regional(y)","y=coef*(x-A.x)+A.y","y"]);
Plotdata("2","g(x)","x",["Num=1"]);
if(!Isptselected(A),
 Enclosing("1",["gr2","Invert(gr1)"],[A,"nodisp"]);
 Hatchdata("1",["i"],[["en1"]]);
);
```



点 A をドラッグして曲線上を動かすと,
`if(!Isptselected(A),・・・`の効果により,
その間は領域の斜線は引かれない。
点 A 以外の画面上の適当な位置をクリックして,
点 A が選択状態でなくなると斜線が引かれる。
引かれる斜線の向きや間隔を変えることもできる。
間隔は実数で指定できる。

【例】円の内部または円と直線で区切られた図形

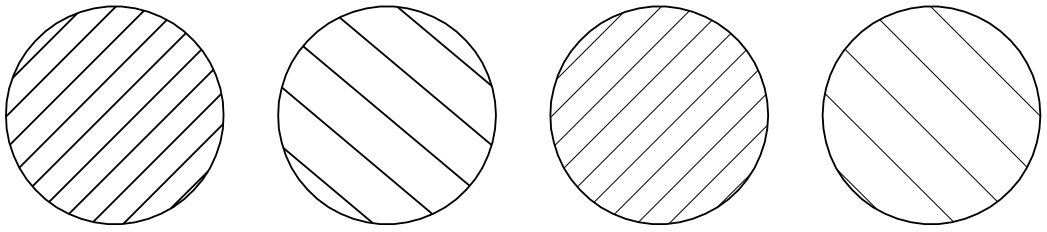
`Circledata([A,B]);` のプロットデータ crAB を用いて，下図左から

`Hatchdata("1",["i"],[["crAB"]]);` 円内に傾き 45° の斜線を引く

`Hatchdata("2",["i"],[["crAB"]],[-40,2]);` 傾き $-40°$，間隔を 2 倍に

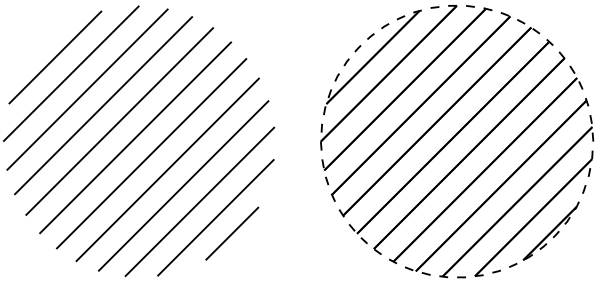
`Hatchdata("3",["i"],[["crAB"]],["dr,0.5"]);` 線の太さを 0.3 倍に

`Hatchdata("4",["i"],[["crAB"]],[-45,2,"dr,0.3"]);`



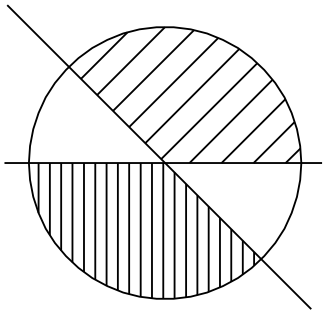
円のオプションに ”notex”をつけた場合と，破線で描いた場合。

```
Circledata([A,B],["notex"]);
Circledata([A,B],["da"]);
```



直線で分けられた領域を作り，対角の上下にハッチをかける。線を描き分ける。

```
Circledata([A,B]);
Lineplot("1",[A,B]);
Lineplot("2",[A,C]);
Hatchdata("1",["iii"],[["crAB"],["ln1","n"],["ln2","n"]]);
Hatchdata("2",["iii"],[["crAB"],["ln1","s"],["ln2","s"]],[90,0.5]);
```



⇒ 関数一覧

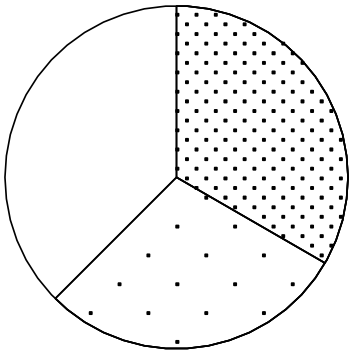
関数 Dotfilldata(name , 方向リスト , プロットデータ , options)

機能 領域を点で敷き詰める。

説明 R とデータの授受をおこなって描画する。書式は Hatchdata() と同様。
オプションは，ドットの密度で 0.1〜0.8 程度。 初期設定は 0.3。

【例】円グラフ

Partcrv() と Enclosing() で閉曲線を作って点を敷き詰める。



```
r=3;
p0=r*[cos(pi/2),sin(pi/2)];
p1=r*[cos(-pi/6),sin(-pi/6)];
p2=r*[cos(-3*pi/4),sin(-3*pi/4)];
Circledata("1",[[0,0],[r,0]]);
Listplot("1",[[0,0],p0]);
Listplot("2",[[0,0],p1]);
Listplot("3",[[0,0],p2]);
Partcrv("1",p1,p0,"cr1");
Enclosing("1",["sg2","part1","Invert(sg1)"],[[0,0]]);
Partcrv("2",p2,p1,"cr1");
Enclosing("2",["sg3","part2","Invert(sg2)"],[[0,0]]);
```

```
Dotfilldata("1",["i"],[["en1"]]);
Dotfilldata("2",["i"],[["en2"]],[0.1]);
```

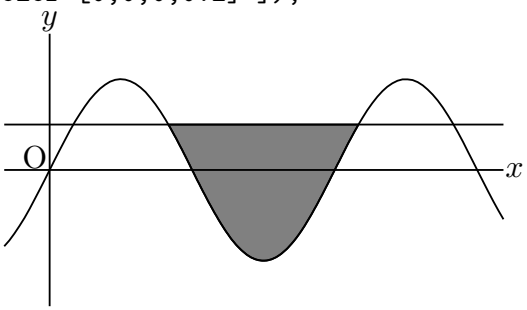
関数 Shade(("名前"), プロットデータのリスト , options)

機能 閉曲線で囲まれた領域を塗りつぶす。

説明 第１引数には，閉曲線を与える曲線分のプロットデータ名を並べる。
デフォルトでは，Joincrvs を使って閉曲線を作っている。ただし，プロットデータのリストに"Invert()"が入っていれば，Enclosing を使う。
option の Color は，Cinderella の画面上での描画色をリストで与える。濃さを指定したい場合は色名や RGB ではなく CMYK にする。
options には，他に，次のものがある。
・Enclosing を使うかどうか："Enc=y/n"（初期値は y）
・Enclosing のときの開始点，描画色
・描画領域のトリミング：Trim=y/n （初期値は n ）
複数の Shade を使うときは，名前をつける。

【例】$y = 2\sin x$ のグラフと直線 $y = 1$ とで囲まれた部分に黒 0.2 の濃さで色を塗る。

```
Setax([7,"nw"]);
Plotdata("1","2*sin(x)","x",["Num=100"]);
Lineplot("1",[[0,1],[1,1]]);
Enclosing("1",["ln1","Invert(gr1)"],[[2,1],"nodisp"]);
Shade(["en1"],["Color=[0,0,0,0.2]"]);
```



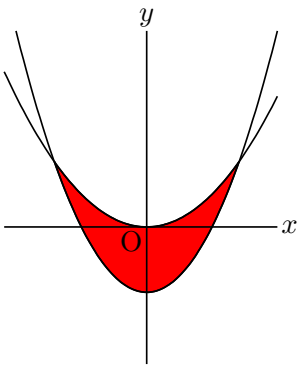
正弦曲線と直線の交点は簡単に計算できるので，次のように Partcrv() で部分曲線を求め，Enclosing で閉曲線を求めずに Shade を使ってもよい。

```
Plotdata("1","2*sin(x)","x",["Num=100"]);
Lineplot("1",[[0,1],[1,1]]);
Partcrv("1",[5*pi/6,1],[13*pi/6,1],"gr1");
Shade(["ln1","Invert(part1)"],[[2.5,1],"Color=0.2*[1,0,0,1]"]);
```

【例】2 つの放物線で囲まれた部分を赤で塗る。

```
Plotdata("1","x^2-1","x=[-3,3]");
Plotdata("2","x^2/2","x=[-3,3]");
Shade(["gr2","Invert(gr1)"],[[-1.5,1],"Color=[1,0,0]","alpha->0.4"]);
```

ここで，alpha->0.4 は画面上の色濃度指定。



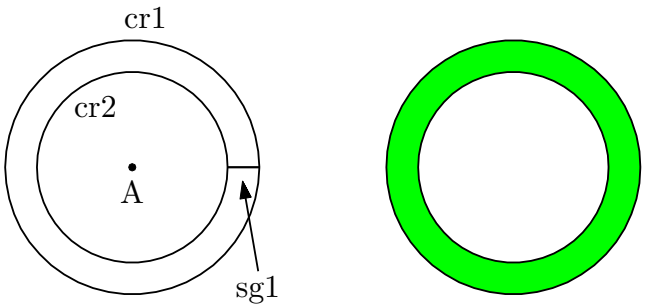
【例】描画領域（NE,SW の矩形領域）からはみ出した部分は表示しないようにする。

```
Plotdata("1","x^2","x");
Partcrv("1",[-1,1],[3,9],"gr1");
Listplot("1",[[3,9],[-1,1]]);
Joincrvs("1",["sg1","part1"]);
Shade(["join1"],["Color=[0,0,0.2,0]","Trim=y"]);
```

SW,NE を動かしてみると，白枠内だけが色塗りされる。

【例】同心円をリング状に塗る。
下図右のように，同心円をリング状に塗るが，円 2 つだけでは閉曲線はできない。そこで，左図のように，円の描き始めを線分で結んで閉曲線を作る。このとき，向きを考えて，Joincrvs で結ぶように，”Enc=n” をつける。線分は非表示にしたいので，"nodisp"オプションをつけておく。なお，点 A を適当な位置に作図しておく。

```
r1=2;
r2=1.5;
Circledata("1",[A,A+[r1,0]]);
Circledata("2",[A,A+[r2,0]]);
Listplot("1",[A+[r1,0],A+[r2,0]],["nodisp"]);
Shade(["cr1","sg1","Invert(cr2)","Invert(sg1)"],["Enc=n","Color=green"]);
```



その他，Joincrvs() の例も参照のこと

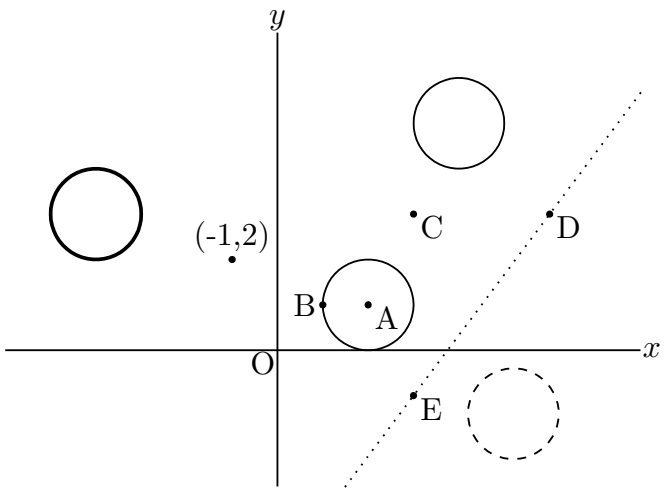
関数　Reflectdata(name , プロットデータ , 対称点または対称軸,options)

機能　プロットデータの鏡映を作成

説明　プロットデータを指定された点または軸に関して対称移動する。
対称点は座標または，点の識別名。ただし，対称点を座標で示すときは要素がひとつのリストにする。
対称軸はリスト [点 1, 点 2] で指定。

【例】中心 A , 半径 AB の円を描き，そのプロットデータを用いて鏡映を描く。
点 C に関して対称な円を実線
点 (-1,2) に関して対称な円を太い実線
直線 DE に関して対称な円を破線

```
Circledata([A,B]);
Reflectdata("1","crAB",[C]);
Reflectdata("2","crAB",[[-1,2]],["dr,2"]);
Reflectdata("3","crAB",[D,E],["da"]);
```



関数　Rotatedata(name , プロットデータ , 角度 , [中心 , options])

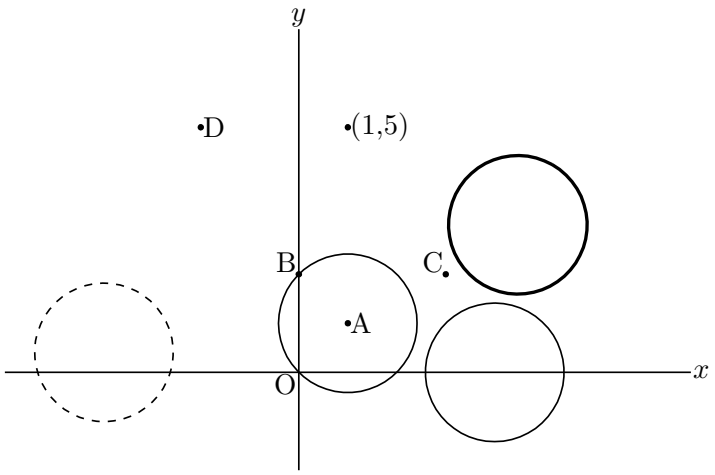
機能　プロットデータの位置を回転する

説明　図形を，中心で示された点の周りに回転する。角度は弧度法で与える
中心と options はまとめてリストで与える。

【例】中心 A , 半径 AB の円を描き，次のように回転して線種を変えて描く。

点 C を中心に $\frac{\pi}{2}$，点 (1,5) を中心に $\frac{\pi}{3}$ ， 点 D を中心に $-\frac{\pi}{3}$

```
Circledata([A,B]);
Rotatedata("1","crAB",pi/2,[C]);
Rotatedata("2","crAB",pi/3,[[1,5],"dr,2"]);
Rotatedata("3","crAB",-pi/3,[D,"da"]);
```



⇒ 関数一覧

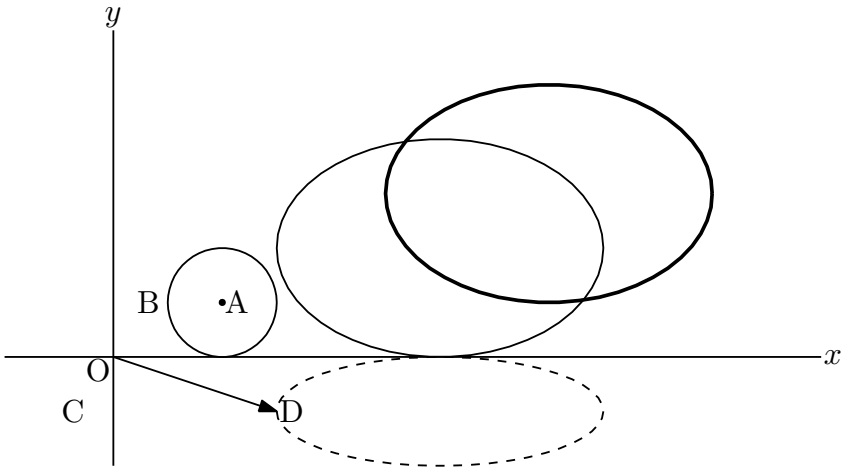
関数　Scaledata(name , プロットデータ，x 方向比率 , y 方向比率 , [中心 , options])

機能　図形の位置を拡大・縮小する

説明　図形の位置をプロットデータを用いて指定された比率で拡大・縮小する
比率は [x 方向比率 , y 方向比率] のリストで与えてもよい。
中心と options はまとめてリストで与える。options は線種

【例】点 A(2,1)，B(1,1)，C(-1,-1)，D(3,-1) を作図しておく。
A を中心とする半径 AB の円のプロットデータを作り，
原点中心に x 軸方向に 3，y 軸方向に 2 拡大する。
C を中心に x 軸方向に 3，y 軸方向に 2 拡大し，実線で太く描く。
原点中心にベクトル $\overrightarrow{\mathrm{OD}}$ だけ拡大し，破線で描く。

```
Circledata([A,B]);
Scaledata("1","crAB",3,2,[[0,0]]);
Scaledata("2","crAB",3,2,[C,"dr,2"]);
Scaledata("3","crAB",[D.x,D.y],[[0,0],"da"]);
```



関数　Translatedata(name , プロットデータ , 移動ベクトル , options)

機能　プロットデータを平行移動する

説明　プロットデータを移動ベクトルで示された分だけ平行移動する。

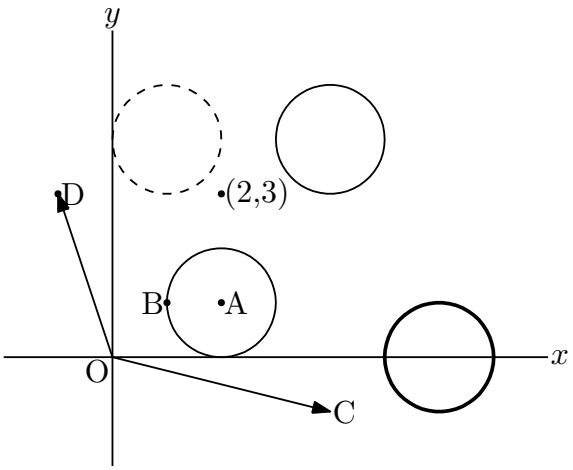
【例】点 A,B,C,D を作図ツールでとっておく。

`Circledata([A,B]);` でできる円（crAB）を x 軸方向に 2，y 軸方向に 3 だけ平行移動して実線で描く。

ベクトル $\overrightarrow{\mathrm{OC}}$ だけ平行移動し，実線で太く描く。

ベクトル $\overrightarrow{\mathrm{OD}}$ だけ平行移動し，破線で描く。

```
Circledata([A,B]);
Translatedata("1","crAB",[2,3]);
Translatedata("2","crAB",C,["dr,2"]);
Translatedata("3","crAB",D,["da"]);
```



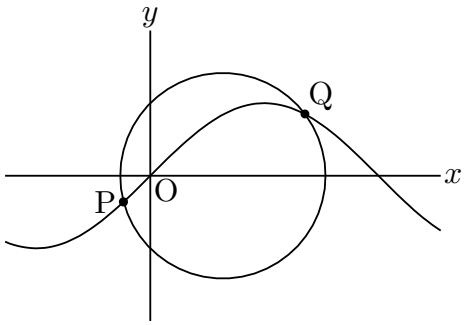
関数　Intersectcrvs(プロットデータ 1, プロットデータ 2)

機能　2 曲線の交点リストを取得する。

説明　オプションとして，共有点があるかどうかを判断するための限界値があるが，通常は使わない。

【例】円と曲線の交点を P,Q とする。

```
Plotdata("1", "sin(x)", "x", ["Num=100"]);
Circledata([A, B]);
tmp=Intersectcrvs("gr1","crAB");
P.xy=tmp_1;
Q.xy=tmp_2;
```



この関数は，交点のデータのリストを返すので，tmp = [[-0.37, -0.36], [2.13, 0.85]] のように値が返ってくる。交点の順序は PD1, PD2 の順序と曲線の向きによって決まる。曲線の向きは，$y = f(x)$ のグラフでは x 座標が増加する向きで，パラメーター表示曲線ではパラメータの増加する向き。また，PD1 上から探し始めて PD2 との交点を拾っていく。

交点がひとつの場合も tmp = [[2.45, 0.63]] と 2 重のリストに入っているので，点として取出すには `P.xy=tmp_1;` とする。

注）交点の算出は，数式処理によるのではなく，プロットデータからの数値探索のアルゴリズムによっている。

関数　IntersectcrvsPp(プロットデータ 1, プロットデータ 2)

機能　2 曲線の交点のパラメータリストを取得する。

説明　2 曲線の交点の座標とパラメータのリストを返す。

Intersectcrvs() との違いは，パラメータがあるかどうかである。

【例】放物線と直線の交点のパラメータを求める。

2 点 A(-1,1),B(2.4) を作図しておく。

```
Plotdata("1","x^2","x");
Lineplot([A,B]);
p1=Intersectcrvs("gr1","lnAB");
p2=IntersectcrvsPp("gr1","lnAB");
println("p1="+p1);
println("p2="+p2);
```

とすると，コンソールには

```
p1=[[-1,1],[2,4]]
p2=[[[-1,1],17.68,1],[[2,4],42.66,1]]
```

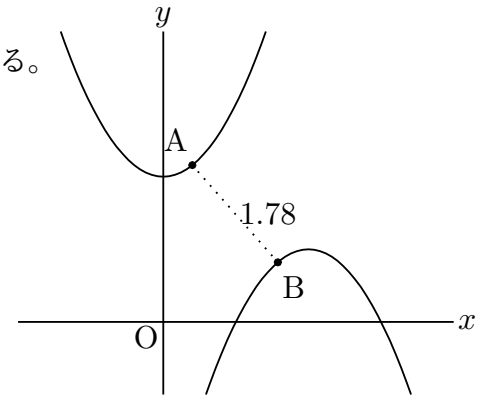
と表示される。

関数　Nearestpt(PD1，PD2)

機能　2 曲線に対し，最も近い点とそのパラメータ，距離のリストを返す

説明　戻り値は，それぞれの曲線上の点の座標とプロットデータ中の位置，その距離からなるリスト。

【例】2 つの放物線上の点の最短距離とその位置を求める。
点 A,B を作図ツールでとっておく。




```
Plotdata("1", "x^2+2", "x=[-2,2]");
Plotdata("2", "-(x-2)^2+1","x=[0,4]");
plist=Nearestpt("gr1","gr2");
A.xy=plist_1;
B.xy=plist_3;
Listplot([A,B],["do"]);
Pointdata("1",[A,B],["Size=2"]);
Letter([A,"n2w","A",B,"s2e","B",(A+B)/2,"e",text(plist_5)]);
```

ここで plist に代入されたリストは次のようになっている。

[[0.4,2.16],31,[1.58,0.82],20.73,1.78]

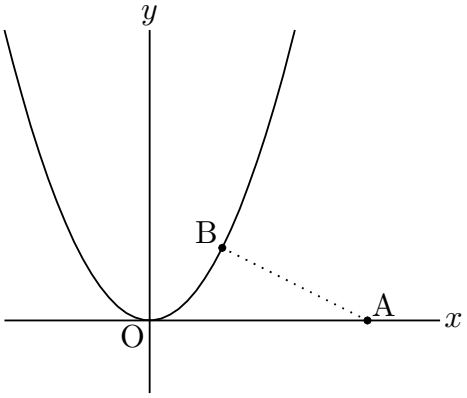
なお，距離 1.78 は小数点以下第 3 位を四捨五入して表示されている。

関数　Nearestptcrv(座標, プロットデータ)

機能　点に最も近いプロットデータの点を求める

説明　第 1 引数の座標に最も近い曲線プロットデータ上の点の座標を返す。

【例】点 (3,0) に最も近い $y = x^2$ 上の点を求める。
点 A は (3,0) に，点 B は適当な位置に作図しておき，次のスクリプトを実行すると点 B が求める点となる。




```
Plotdata("1", "x^2", "x");
B.xy=Nearestptcrv(A.xy,"gr1");
Listplot([A,B],["do"]);
Ptsize(2);
Pointdata("1",[A,B]);
Letter([A,"ne","A",B,"nw","B"]);
```

注）第 1 引数は座標なので，A ではなく A.xy としなければならない。

⇒ 関数一覧

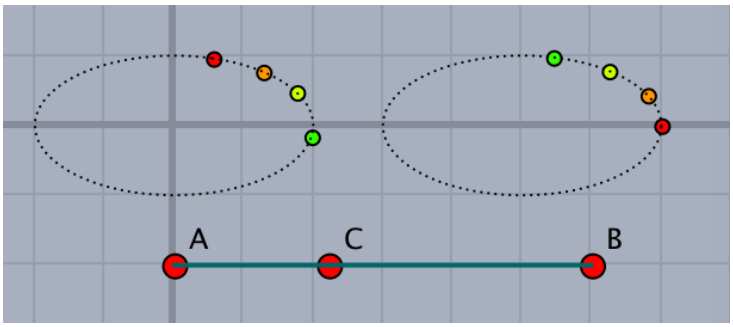
関数 Numptcrv (プロットデータ)

機能 プロットデータの個数を返す

説明 Cindyscript で length(PD) とするのと同じ

【例】 Implicit() と Paramplot() でそれぞれ楕円を描いたときのプロットデータの順序を比較する。ただし，TeX には書き出さない。

```
Slider("A-C-B",[0,-2],[6,-2]);
Implicitplot("1","x^2+4*y^2=4","x=[-2,2]","y=[-2,2]",["do"]);
Paramplot("1","[2*cos(t)+5,sin(t)]","t=[0,2*pi]",["do","Num=140"]);
println([Numptcrv(imp1),Numptcrv(gp1)]);
n=floor(C.x*2);
repeat(n,s,start->0,
t=s*10+1;
draw(imp1_t,color->hue(s/10));
draw(gp1_t,color->hue(s/10));
);
```



4 行目で，2 つのプロットデータの個数が同じであることを確かめている。
スライダを動かすと，10 個おきのプロットデータに対応する点が描かれる。

関数 Paramoncrv(点の座標, 曲線の名前)

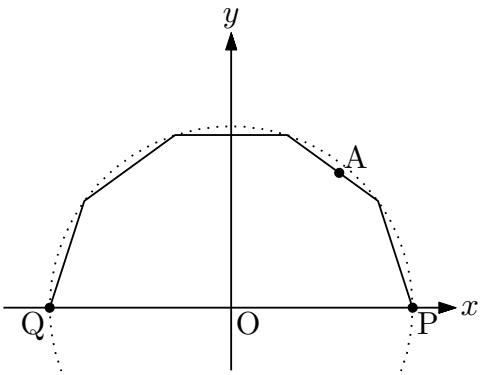
機能 曲線上の点のパラメータ値を返す。

説明 曲線は折れ線として描かれ，曲線上の各点はこの折れ線の節点を基準としたパラメータ値を持つ。パラメータ値は整数部分が節点の番号，小数部分が節間の位置を表す。

【例】図のような点 P から Q に至る円周上の 5 等分点を節点とする折れ線 cr1 において，n 番目の線分上の点は $n \leq t \leq n+1$ の範囲のパラメータ値を持つ。
たとえば，図の点 A は 2 番目の線分上にあり，この値は

```
println(Paramoncrv(A.xy,"cr1"));
```

によってコンソールに表示される。(たとえば 2.45)。
点 A の位置を動かすとパラメータ値は変わる。



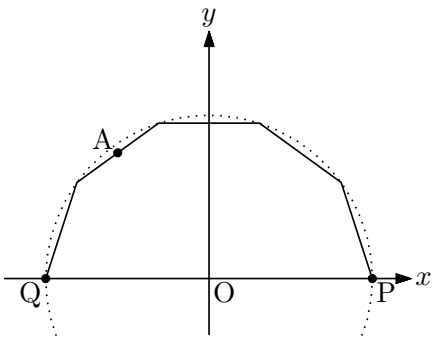
⇒ 関数一覧

関数 Pointoncrv(点のパラメータ値, PD)
機能 曲線上のパラメータ値を持つ点の座標を返す。
説明 曲線（折れ線）上の節点を基準としたパラメータ値により点の位置が定まる。

【例】図のような点 P から Q に至る半円周上の 5 等分点を節点とする折れ線 cr1 において，パラメータ値 4.5 を持つ点 A は 4 番目の線分の中点である。したがって

```
Circledata("0",[[0,0],[2,0]],["do"]);
Circledata("1",[[0,0],[2,0]],["Num=5","Rng=[0,pi]"]);
tmp=Pointoncurve(4.5,"cr1");
Pointdata("1",tmp,["Size=3"]);
Letter([tmp,"nw","A",[2,0],"se","P",[-2,0],"sw","Q"]);
```

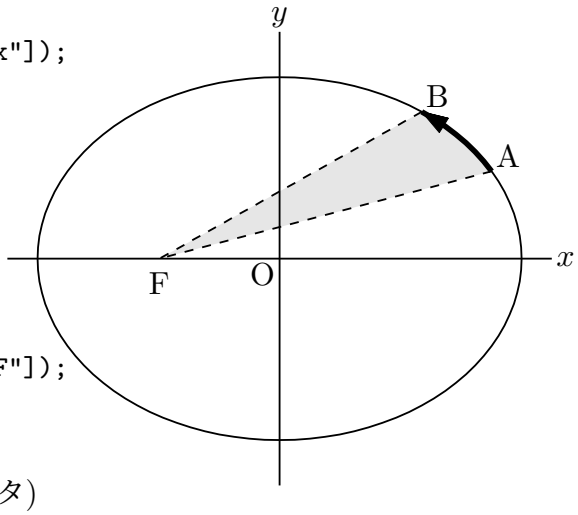
によって，点 A を中点に置くことができる。



関数 Ptcrv(n, プロットデータ)
機能 曲線プロットデータの n 番目の節点を返す
説明 Cindyscript の PD_n と同じ

【例】楕円上の点で分割する。あからじめ必要な点を作図しておく。

```
Circledata([O,P],["do","Num=100","notex"]);
Scaledata("1","crOP",4/3,1);
F.xy=[-sqrt(7),0];
A=Ptcrv(9,sc1);
B=Ptcrv(16,sc1);
Listplot("1",[A,F,B],["da"]);
Partcrv("1",A,B,"sc1",["dr,3"]);
Shade(["part1","sg1"],0.1);
Arrowhead(B,"sc1",[1.5]);
Letter([A,"ne","A",B,"ne","B",F,"s2","F"]);
```



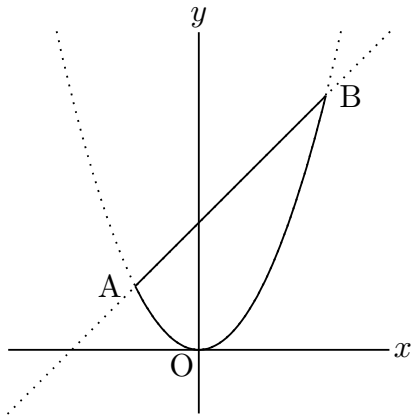
関数 Ptstart(プロットデータ) , Ptend(プロットデータ)

機能 プロットデータの最初の点，最後の点を取得する。

説明 プロットデータの最初の点，最後の点の座標を返す。

【例】定義域を限定したグラフの両端の点を取得し線分 AB を引く。

```
Deffun("f(x)",["regional(y)","y=x^2","y"]);
Plotdata("1","f(x)","x",["do"]);
Plotdata("2","f(x)","x=[-1,2]");
Lineplot("1",[Ptstart(gr2),Ptend(gr2)],["do"]);
Listplot("1",[Ptstart(gr2),Ptend(gr2)]);
Letter([A,"w2","A",B,"e2","B"]);
```



関数 ReadOutData(ファイル名)

機能 外部データをプロットデータとして読み込む

説明 C や R などで作成した KeTCindy 形式のデータファイルを読み込む。
引数を省略した場合は，Fhead で定義したファイル名のテキストファイルから読み込む。

ファイル名にはコンマで区切ってパスを与えることができる。たとえば,

```
ReadOutData("/datafolder","file.txt");
```

KeTCindy 形式のデータとは

変数名//
start//（リストの始まり）
[, ,], …. //（個々のデータ２か３次元）
…
end//（リストの終わり）
start//（次のリストの始まり）
…
end//
変数名//
start//
…
end//

の形式のテキストファイル。

関数 Readcsv(path,filename,option)

機能 csv ファイルを読む。

説明 csv ファイルを読みこむ。戻り値は読み込んだデータのリスト。
第１引数の path は，ファイルを作業フォルダ（ 初期設定は fig ）に置いた場合は省略することができる。そうでない場合は，フルパスで指定する。たとえば，”/Users/Hoge/Desktop”
option は，”Flat=” で，”Flat=y” の場合は，読み込んだデータをリスト化したときに平滑化（１次元のリスト）にする。 初期設定は ”Flat=n”
【例】次のような CSV ファイル sample.csv を読み込むとする。

```
12,14,15,18,13
9,13,17,21
```

つまり，２行分のデータである。

```
data=Readcsv("sample.csv");
```

とすると,

```
data=[[12,14,15,18,13],[9,13,17,21]]
```

となる。
したがって，１行目のデータだけ取り出したい場合は

```
dt1=data_1;
```

とする。

⇒ 関数一覧

関数 Readlines(path,filename,option)

機能 テキストファイルを 1 行ずつ読む。

説明 テキストファイルを 1 行ずつ読みこむ。戻り値は読み込んだ文字列のリスト。
第 1 引数の path は，ファイルを作業フォルダ（ 初期設定は fig ）に置いた場合は省略することができる。そうでない場合は，フルパスで指定する。たとえば，”/Users/Hoge/Desktop”

関数 WriteOutData(ファイル名,PD リスト)

機能 外部データに書き出す

説明 プロットデータを KeTCindy 形式のデータファイルに書き出す。出力先の 初期設定は作業フォルダ。
【例】　放物線と円のプロットデータを書き出す。

```
Plotdata("1", "x^2","x");
Circledata("1",[[0,0],[1,0]]);
WriteOutData("figdata.txt",["gr1",gr1,"cr1",cr1]);
```

書き出されたファイルの中身は次のようになっている。

```
gr1//
start//
[[-2.68843,7.22765],[-2.51807,6.34067],・・,[-2.00698,4.02798]]//
[[-1.83662,3.37318],[-1.66626,2.77642],・・,[-1.15518,1.33443]]//
    以下，同様にプロットデータが続く
[[5.82965,33.98479]]//
end//
cr1//
start//
[[1,0],[0.99211,0.12533],[0.96858,0.24869],・・,[0.80902,0.58779]]//
    以下，同様にプロットデータが続く
[[0.87631,-0.48175],[0.92978,-0.36812],・・,[1,0]]//
end////
```

関数 Extractdata(データ名, 属性)

機能 ReadOutData() で読み込んだデータに属性をつける。

説明 ReadOutData() で読み込んだデータには，線種などの属性がついていないので，そのままでは表示されない。そこで，この関数により属性をつけて表示する。

```
ReadOutData("figdata.txt");
Extractdata("gr1",["da"]);
```

⇒ 関数一覧

1.4 計算

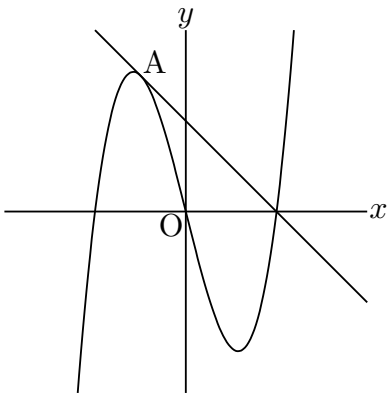
関数 Derivative(関数式 , 変数 , 値)

機能 関数の微分係数を求める

説明 関数式で与えられた関数の,「変数＝値」における微分係数を求める。
値は，点の座標を用いることができる。点 A の x 座標であれば， A.x とする。

【例】3 次曲線上の点 A で接線を引く。点 A,B は作図ツールで適当にとっておく。

```
Deffun("f(x)",["regional(y)","y=x^3-4*x","y"]);
coef=Derivative("f(x)","x",A.x);
A.y=f(A.x);
B.y=coef*(B.x-A.x)+A.y;
Plotdata("1","f(x)","x",["Num=200"]);
Lineplot([A,B]);
Letter([A,"ne","A"]);
```



なお，曲線のプロットデータを用いて，微分係数を求めることもできる。
書式は，Derivative(PD , 値) で，次のように使う。（上の例と同じ図ができる）

```
Deffun("f(x)",["regional(y)","y=x^3-4*x","y"]);
Plotdata("1","f(x)","x",["Num=200"]);
coef=Derivative("gr1","x="+A.x);
A.y=f(A.x);
B.y=coef*(B.x-A.x)+A.y;
Lineplot([A,B]);
Letter([A,"ne","A"]);
```

また，曲線の接線については，Tangentplot も参照されたい。

⇒ 関数一覧

関数　integrate(関数式 , 変数 = 範囲 , options)

関数　integrate(PD , 範囲 , options)

機能　関数式またはプロットデータで与えられた関数（データ）の数値積分の値を求める。

説明　options は次の通り。

”Rule=s”：シンプソン法による。 初期設定は大島ベジェ公式。

”Num=数値”：分割数の指定。初期値は 100

【例】$f(x) = x^3 - 2x^2 + 2$ について，0 から 3 までの定積分の値を求める。

```
f(x):=x^3-2*x^2+2;
val=Integrate("f(x)","x=[0,3]");
 println(val);//8.25が表示される
```

【例】上の例と同じ関数をプロットデータで指定する。

```
plotdata("1","x^3-2*x^2+2","x");
println(Integrate("gr1",[0,3]));
```

数値積分ではなく，数式処理として定積分の値を求める場合は，Maxima を利用する。CalcbyM を参照。

関数　Inversefun(関数 , 範囲 , 値)

機能　関数の逆関数値を求める

説明　関数は文字列で，関数式もしくは定義された関数名とする。

指定された範囲の中で逆関数値を求める。存在しない場合は一方の端点を戻り値とし，コンソールに「not found」と表示される。

数式処理ではなく数値探索のアルゴリズムを使っているので，単調関数でない場合は範囲をできるだけ狭くとるとよい。値が複数ある場合は，小さいほうが返される。

【例】`x=Inversefun("sin(x)","x=[0,pi/2]",0.5);`

実行すると $x = 0.5236$ となる。

⇒ 関数一覧

1.5　値の取得と入出力

計算値やプロットデータの値を取得したり，R 用とのデータのやりとりをする。

関数　Asin(実数）,Acos(実数)

機能　逆三角関数の値を求める。

説明 CindyScript の組み込み関数に，arcsin(x),arccos(x) があるが，x の絶対値が 1 より大きい場合は虚数を返す。このことがRではエラーになるので，計算誤差により 1 よりわずかに大きくなる場合のために用意した関数。

関数 Sqr(実数)
機能 平方根を求める。
説明 CindyScript の組み込み関数に，sqrt(x) があるが，x が負の場合は虚数を返す。これに対し，x が負の場合は 0 を返すようにした関数。計算誤差により 1 よりわずかに大きくなる場合のために用意した。

関数 BBdata(ファイル名,option)
機能 画像ファイルのサイズを求める
説明 TeX 文書において，inputgraphics コマンドで画像を貼り込むときの BB サイズを求める。TeX 処理系の extractbb を用いて画像ファイルから BB データを作り，テキストファイルとして作業ディレクトリに書き出す。これを読んで，コンソールに ingludegarphics のコマンドを書き出す。これをそのままコピーすればよい。 なお，bb の値は整数値ではなく，高精細の値を小数点以下 2 桁に四捨五入して示される。画像ファイルは，PDF に限らず，PNG，JPG などでもよい。
option は，幅または高さの指定。
”w=40mm” で width=40mm が，”h=40mm”で height=40mm が付加される。

【例】

```
10 BBdata("ellipsecindy.pdf");
11 BBdata("circle.png",["w=40mm"]);
12
```

```
\includegraphics[bb=0.00 0.00 272.01 240.01]{ellipsecindy.pdf}
\includegraphics[bb=0.00 0.00 306.02 219.01,width=40mm]{circle.png}
```

⇒ 関数一覧

関数 Cindyname()
機能 作図中のファイル名を取得する。
説明 たとえば，現在作図しているファイル名が「polygon.cdy」のとき，”polygon” を返す。

関数 Crossprod(リスト，リスト)
機能 2 つのベクトルの外積を求める。
説明 Cindyscript の組み込み関数 cross(リスト，リスト) と同じ。

【例】Crossprod([1,0,0],[1,1,1]);
結果は [0,-1,1]

関数 Dotprod(リスト，リスト)
機能 2 つのベクトルの内積を求める。
説明 Cindyscript では，積の演算で内積が求められる。

【例】Dotprod([1,2,3],[1,-1,1]);
結果は 2
[1,2,3]*[1,-1,1] でも同じ結果を得る。

関数 Findarea(プロットデータ)
機能 プロットデータで囲まれる部分の面積を求める。
説明 閉曲線をなすプロットデータで囲まれる部分の面積を求める。大島のベジェ公式を用いている。

【例】楕円の面積を求めて表示する。

```
Paramplot("1","[3*cos(t),2*sin(t)]","t=[0,2*pi]");
area=Findarea("gp1");
println(Sprintf(area,6));
```

コンソールに面積 18.849536 が表示される。

関数 Findlength(プロットデータ)
機能 プロットデータの曲線の長さを求める。
説明 プロットデータが描く曲線の長さを求める。大島のベジェ公式を用いている。

【例】円周の長さを求めて表示する。

```
Circledata("1",[[0,0],[2,0]]);
len=Findlength("cr1");
println(Sprintf(len,6));
```

コンソールに 12.558097 が表示される。

⇒ 関数一覧

1.6 作表

関数 Tabledata(横データ，縦データ, 除外線 , options)

機能 表の枠を作成し，表のデータ list を返す

説明 Cinderella の描画面上に左下を原点とする表を作成する。

除外線がない場合は空リストを指定する。（必須）

options のうち，Tabledata に特有なものは

ラベルのスキップ幅，”Setwindow=n”，”Geo=n”，”Move=[0,0]”

である。

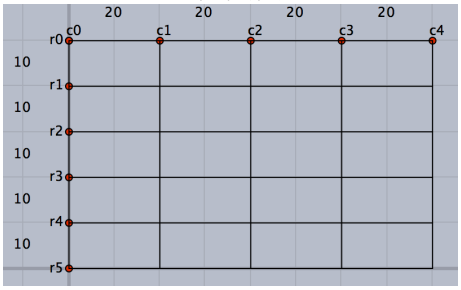
”Setwindow=n” をつけると，NE,SW による出力領域指定が有効になり，NE,SW をドラッグして出力領域を変更できる。（初期状態は，表の右上と左下）つけない場合は，表の部分だけが出力される。

縦横データは，次のように，間隔の幅で与える。ただし，幅は Cinderella の描画面の 0.1 を単位とする。

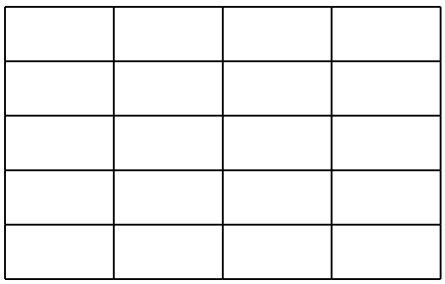
```
Yoko=[20,20,20,20];
Tate=[10,10,10,10,10];
Tabledata(Yoko,Tate,[]);
```

”Geo=y”とすると，作成された表には，行，列の制御点がつく。画面上では，横罫線の番号 r0,r1,・・・縦罫線の番号 c0,c1,・・・と見ることもできる。また，縦幅，横幅が数字で示される。ただし，これらは TEX には出力されない。また，作表は Cinderella の描画面上では座標平面上に置かれるが，TEX への出力は座標平面上には置かないことが多いので，座標軸は非表示としている。

描画面



TeX



”Geo=y”の場合は，表のサイズ・行幅・列幅は，作成後にそれぞれの制御点をドラッグすることにより任意に変えることができる。

除外線は，除外するセルの罫線を，r と c で位置指定する。

横罫線の場合，横罫線の番号，範囲（から，まで）

縦罫線の場合，縦罫線の番号，範囲（から，まで）

とする。

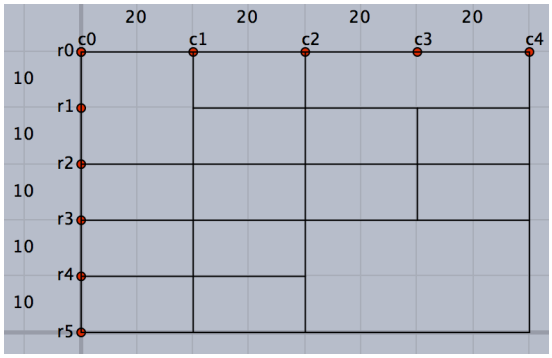
ラベルのスキップ値は，いくつおきにラベルをつけるかの設定である。0 とするとラベルが表示されない。ただし，ラベルは Cinderella の画面上だけの問題。

”Move=点の座標”とすると，指定された点が左下になるように平行移動される。

【例】4 つの罫線を非表示にする

```
Rmv=["r1c0c1","c3r0r1","c3r3r5","r4c2c4"];
Yoko=[20,20,20,20];
Tate=[10,10,10,10,10];
Tabledata(Yoko,Tate,Rmv);
```

で，次の表ができる。



＜補足＞

”Geo=y”の場合，制御点 r0,r1,・・・,c0,c1,・・・がなければ新しく作り，すでに存在する場合はそのままとする。したがって，一度表を作成したのち，行数・列数を修正して作り直す場合は，一度既存の点を消去する必要がある。そのためには，「すべての点を選択する」ツールをクリックして点を消去するのがよい。クリックすると，消去後すぐに新規作成される。(誤って「すべての要素を選択する」を選ばないこと)

他の点が描画されている場合は，表の部分だけドラッグで選択するか，表示メニューの「式による表示」で一覧表を出して，制御点を選択して消去する。

”Geo=n”（デフォルト）の場合，幾何点を生成しない。幾何点を作成しないメリットは，スクリプトだけで全体の縦横幅を変更できること。デメリットはインタラクティブな微調整ができないこと。

【例】１つおきにスキップして，r1,r3,c1,c2 を非表示とする。

```
Yoko=[20,20,20,20];
Tate=[10,10,10,10,10];
Tabledatalight(Yoko,Tate,[],[2]);
```

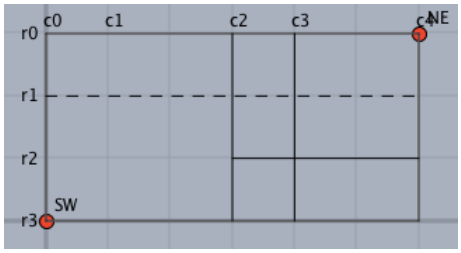
関数　Changetablestyle(罫線リスト, 変更オプション)

機能　Table の罫線の描画オプションを変更

説明　罫線の部分的に指定して描画オプションを変更できる。

【例】

```
Tabledatalight([10,20,10,20],[10,10,10],[]);
Changetablestyle(["r1c0c4"],["da"]);
Changetablestyle(["r2c0c2","c1r0r3"],["nodisp"]);
```



複数の表を描くこともできる。

⇒ 関数一覧

関数　Findcell(列番号, 行番号)

機能　セルの情報 list（中心，横幅／ 2，縦幅／ 2）を返す

説明　列番号，行番号は左上のセルを 1 列 1 行として数える。

【例】`Tabledata(Tate,Yoko,[]);`

```
println(Findcell(2,1));
```

とすると，2 列 1 行のセルの中心の座標と横幅の半分，縦幅の半分の値がリストとしてコンソールに表示される。

関数　Putcell (列番号, 行番号, 位置, 文字データ)

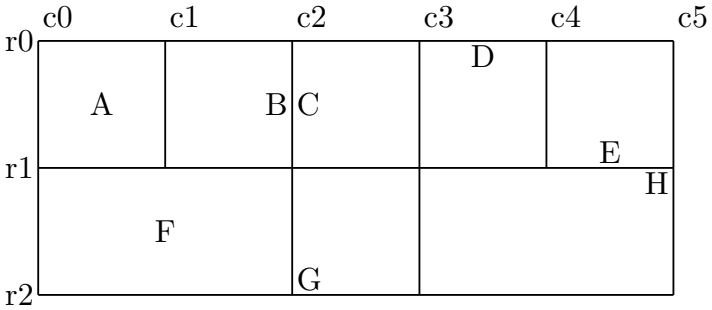
機能　セルに文字列を入れる

説明　複数のセルにまたぐ位置指定の場合，列番号，行番号は，セル左上と右下の制御点の名称で指定する。

位置は c, r, l, t, b （中央 center , 右 right , 左 left , 上 top , 下 bottom ）

位置の例を以下に示す。

```
yoko=[20,20,20,20,20];
tate=[20,20];
Tabledata(yoko,tate,["c1r1r2","c4r1r2"]);
Putcell(1,1,"c","A");
Putcell(2,1,"r","B");
Putcell(3,1,"l","C");
Putcell(4,1,"t","D");
Putcell(5,1,"b","E");
Putcell("c0r1","c2r2","c","F");
Putcell("c2r1","c3r2","lb","G");
Putcell("c3r1","c5r2","rt","H");
```



※数式の場合は，**Putcellexpr** を用いる。

⇒ 関数一覧

関数　Putcol (列番号, 文字位置，文字列リスト)
機能　１列に順に文字を書き入れる
説明　列番号で指定した列に，第１行から順に文字列リストの文字を書き入れる
数の場合はダブルクォートでくくらなくてもよい。
セルを飛ばす場合は，ヌル文字列 "" を書く。

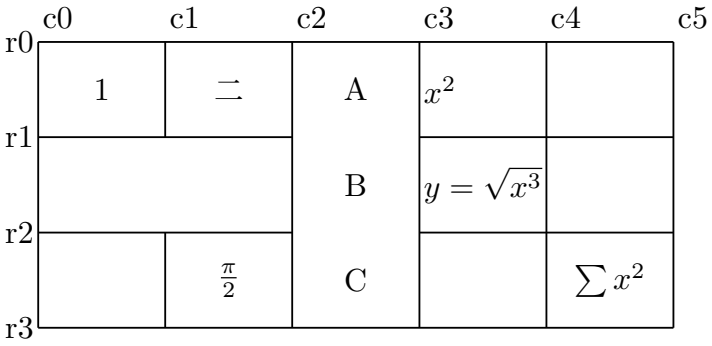
関数　Putcolexpr (列番号, 文字位置，文字列リスト)
機能　１列に順に文字を書き入れる
説明　文字列に TEX 書式を使うことができる

関数　Putrow (行番号, 文字位置，文字列リスト)
機能　１行に順に文字を書き入れる
説明　行番号で指定した行に，第１列から順に文字列リストの文字を書き入れる。

関数　Putrowexpr (行番号, 文字位置，文字列リスト)
機能　１行に順に文字を書き入れる

説明　文字列に TEX 書式を使うことができる
文字を入れる例を示す。

```
Tate=[20,20,20,20,20];
Yoko=[15,15,15];
Tabledata(Tate,Yoko,["c1r1r2","r1c2c3","r2c2c3"]);
Putcol(3,"c",["A","B","C"]);
Putcolexpr(4,"l",["x^2","y=\sqrt{x^3}"]);
Putrow(1,"c",[1,"二"]);
Putrowexpr(3,"c",["","\frac{\pi}{2}","","","\sum{x^2}"]);
```

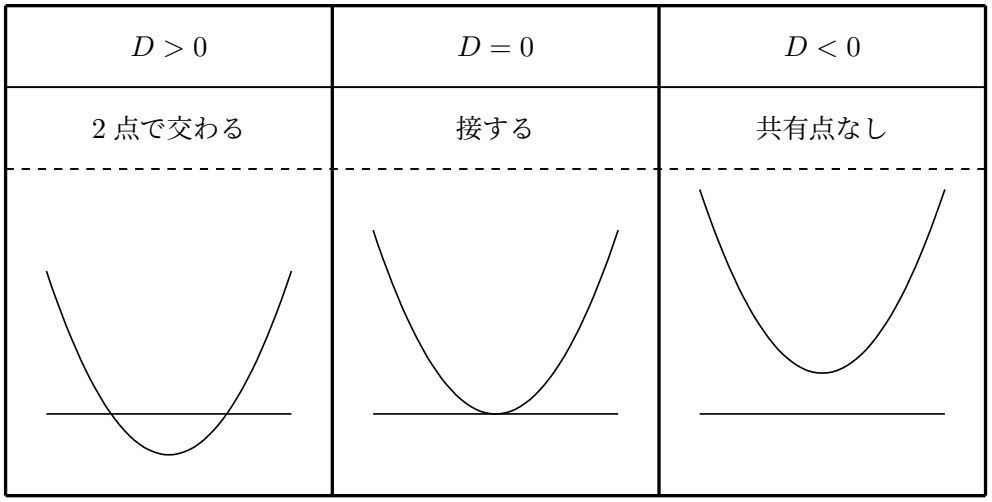


※ r0,c0,・・は画面に表示される番号。
グラフや文を入れた表の作成例
Putcolexpr(),Putrowexpr() では，数式だけでなく，一般の TEX の文を入れることができる。また，グラフの位置を適当に合わせて描画することにより，表のセルの中にグラフを入れることができる。

【例】2 次関数のグラフと 2 次方程式の判別式の関係
セルの中にグラフを描く例。実際には，セルの位置にグラフを描く。
x 軸を描くための点 A〜F は作図ツールでとっておく。
スクリプトを実行して表ができたら，制御点をドラッグしてサイズを調整し，点 A〜Fもドラッグして軸と放物線の共有状況を示すようにする。

```
Tate=[40,40,40];
Yoko=[20,20,20];
Tabledata(Tate,Yoko,[],["dr,2"]);
Changetablestyle(["r1c0c3"],["dr"]);
Changetablestyle(["r2c0c3"],["da"]);
Plotdata("1","(x-2)^2+0.5","x=[0.5,3.5]");
Plotdata("2","(x-6)^2+1","x=[4.5,7.5]");
Plotdata("3","(x-10)^2+1.5","x=[8.5,11.5]");
Listplot([A,B]);
Listplot([C,D]);
Listplot([E,F]);
Putrowexpr(1,"c",["D>0","D=0","D<0"]);
```

```
Putrow(2,"c",["2点で交わる","接する","共有点なし"]);
```



【例】増減表とグラフ

関数の増減表とグラフを１つの表の中に入れた例。

```
Tate=[6,6,10,6,10,6,40];
Yoko=[30,6,6,6];
Rmv=["c1r0r1","c2r0r1","c3r0r1","c4r0r1","c5r0r1", "r1c6c7",
"r2c6c7","r3c6c7"];
Tabledata(Tate,Yoko,Rmv,["dr"])
Tlistplot("23d",["c1r2","c2r3"]);
Tlistplot("23u",["c1r3","c2r2"]);
Putrowexpr(2,"c",["x",0,"\cdots","\tfrac{1}{4}","\cdots",4]);
Putrowexpr(3,"c",["y`","","-",0,"+"]);
Putrowexpr(4,"c",["y",0,"\searrow","-\tfrac{1}{4}","\nearrow",2]);
Putcell(1,1,"l2t2","{\small\begin{minipage}{44mm}$y=x-\sqrt{x}$\\$y`=
\dfrac{2\sqrt{x}-1}{2\sqrt{x}}=0$|より\vspace{1mm}\\\hspace*{2zw}$x=
\dfrac{1}{4}$\vspace{1mm}\\増減表は次のようになる\end{minipage}}" );
Plotdata("1","x-sqrt(x)","x=[0,3]",["do","notex"]);
Listplot("2",[[0,0],[3,0]],["do","notex"]);
Listplot("3",[[0,-0.5],[0,3]],["do","notex"]);
Translatedata("1","gr1",[4.9,1],["dr"]);
Translatedata("2","sg2",[4.9,1],["dr"]);
Translatedata("3","sg3",[4.9,1],["dr"]);
Letter(Ptend(tr2),"e1","\small{$x$}");
Letter(Ptend(tr3),"n1","\small{$y$}");
Letter(Ptstart(tr2),"w1","\small O");
Expr(Ptend(tr1),"nw-2","y=x-\sqrt{x}");
```

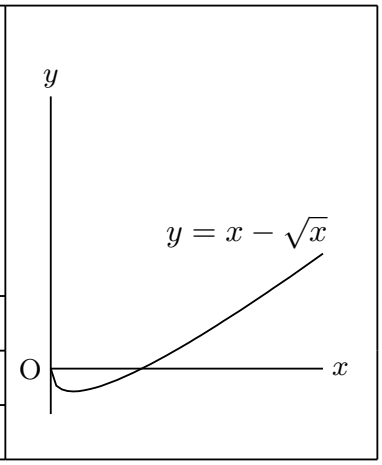
$y = x - \sqrt{x}$

$y' = \dfrac{2\sqrt{x} - 1}{2\sqrt{x}} = 0$ より

$x = \dfrac{1}{4}$

増減表は次のようになる.

x	0	$\cdots$	$\frac{1}{4}$	$\cdots$	4
y'		$-$	0	$+$	
y	0	$\searrow$	$-\frac{1}{4}$	$\nearrow$	2



【例】凹凸を含めた増減表

```
Tate=apply(1..8,20);
Yoko=[apply(1..4,10);
Tabledata(Tate,Yoko,[]);
Putrowexpr(1,c,["x","\cdots","-1","\cdots","0","\cdots","1","\cdots"]);
Putrowexpr(2,c,["y`","+","+","+","0","-","-","-"]);
Putrowexpr(3,c,["y``","+","0","-","-","-","0","+"]);
Putrowexpr(4,c,["y","\nelarrow","\frac{1}{\sqrt{e}}","\nerarrow",
"1","\serarrow","\frac{1}{\sqrt{e}}","\selarrow"]);
```

x	$\cdots$	-1	$\cdots$	0	$\cdots$	1	$\cdots$
y'	$+$	$+$	$+$	0	$-$	$-$	$-$
y''	$+$	0	$-$	$-$	$-$	0	$+$
y	$\nearrow$	$\frac{1}{\sqrt{e}}$	$\nearrow$	1	$\searrow$	$\frac{1}{\sqrt{e}}$	$\searrow$

ここで，凹凸を示す矢印は，ketpic.sty で定義されているものである。
nelarrow,nerarrow,selarrow,serarrow,NELarrow,NERarrow,SELarrow,SERrarrow がある。先頭の ne,se で北東・南東（右上・右下）次の r,l は回転の向き（r：right：反時計回り，l：left：時計回り）の矢印 (arrow) と覚えるとよい。直線系の矢印は NEarrow,SEarrrow。 少しずつ違うので試されたい。
なお，これらの矢印は CindyTeX にはないので，Cinderella の描画面には表示されない。

関数 Tgrid(セルラベル)
機能 表のセルの座標を返す

説明　指定されたセルの左上の座標を返す。実際には，セルラベルは罫線を示しているので，指定した罫線の交点（格子点）ということもできる。

関数　Tlistplot(セルラベル 1, セルラベル 2)
機能　指定された 2 つの格子点を線分で結ぶ
説明　セルに斜線を引くのに用いる。

【例】`Tlistplot(["c0r1","c1r2"]);`

⇒ 関数一覧

1.7 その他

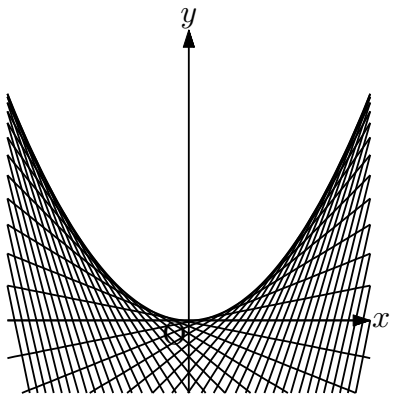
関数　Assign(文字列，文字，文字)
機能　文字列の中のある文字を他の文字で置き換える
説明　第 1 引数の文字列中の第 2 引数の文字を，第 3 引数の文字で置き換える。
第 3 引数が数値の場合，文字列に変換される。
第 2 引数と，第 3 引数をリストにして，複数の置き換えをすることができる。

【例】`a*x` を `1.3*x` とした文字列を返す。次のいずれも同じ結果になる。

```
Assign("x^2+a*x","a","1.3");
Assign("x^2+a*x","a",1.3);
```

【例】直線 $y = bx - b^2$ の係数 b を変化させて描き，包絡線をうかびあがらせる。

```
repeat(50,t,
cb=t/5-5;
Plotdata(text(t),Assign("b*x-b^2","b",cb),"x");
);
```



【例】文字で表された x と y の係数をまとめて数値で置き換える。

```
Assign("a*x^2+b*x",["a",1,"b",2]);
```

⇒ 関数一覧

関数 Colorcode(種別 1, 種別 2, カラーコード)

機能 種別 1 から種別 2 へカラーコードを変換する。戻り値は変換されたコード。

説明 種別は，”rgb”,”cmyk”,”hsv”のいずれか。

【例】変換例をいくつか示す。

RGB の [1,0,0] を CMYK に変換したコードをコンソールに表示する。

```
col=Colorcode("rgb","cmyk",[1,0,0]);
println(col);
```

CMYK の [0,1,1,0] を RGB に変換したコードをコンソールに表示する。

```
col=Colorcode("cmyk","rgb",[0,1,1,0]);
println(col);
```

RGB の [1,0,0] を HSV に変換したコードをコンソールに表示する。

```
col=Colorcode("rgb","hsv",[1,0,0]);
println(col);
```

関数 Dqq(文字列)

機能 文字列の前後に"をつける。

```
str="abc";
str2=Dqq(str);
println([str,str2]);
```

関数 Factorial(n)

機能 正の整数 n の階乗を計算する。

関数 Norm(ベクトル), Norm(ベクトル 1, ベクトル 2)

機能 ベクトル（2 つのベクトルの場合は差の大きさ）の大きさを計算する。

関数 Figpdf(option)

機能 出力枠サイズの PDF を作る。

説明 KETCindy では，通常，出力された fig.tex ファイルを閲覧する PDF を A4 サイズで

作成する。これに対し，Figpdf() を実行すると，出力サイズの PDF を作成する。閲覧用だけではなくワープロなどにに貼り込むときにそのまま使用できる。ただし，そのための親子プロセスを生成して実行するため，次の手続き (1)(2) が必要となる。

(1) Setparent(filename) で，出力する PDF 用のファイル名を設定する。
(2) 出力は，「Parent」のボタンを押す。

たとえば，fig.cdy で作図しているとき，

```
Setparent("pic");
```

とすると，fig.tex を表示した pic.pdf が作成される。pic.pdf が目的の PDF。
このファイル名は 作図している Cinderella のファイル名，または Setfiles() で指定したファイル名とは異なるものにする。
option は，マージン（余白）と平行移動量。指定しない場合は 初期設定値。
余白は，左右上下の順に 4 つの数をコンマで区切る。
平行移動量は，右方向，下方向をリストで与える。
余白指定と平行移動指定は同時に行うことができる。

【例】余白の設定

`Figpdf([5,5,10,10]);`	左右に 5mm，上下 10mm の余白
`Figpdf([[5,10]]);`	右に 5mm，下に 10mm 平行移動して表示
`Figpdf([5,8,10,10,[5,-5]]);`	左 5mm，右 8mm，上下 10mm の余白， 右に 5mm，上に 5mm 平行移動して表示

なお，座標軸を表示する場合，右側は最低 3mm の余白を設定しないと軸の文字が入らない。

⇒ 関数一覧

関数 Help(文字列)
機能 関数の使用例を取得する
説明 文字列で始まる関数の使用例をコンソールに表示する。

```
println(Help("L"));
```

のようにすると，コンソールに，次のように「L」で始まる関数の使用例が表示される。

```
Letter([C,"c","Graph of $f(x)$"]);
Letter([C,"c","xy"],["size->30"]);
```

文字を書き込む

```
Letterrot(C,B-A,"AB");
Letterrot(C,B-A,"t0n5","AB");
Letterrot(C,B-A,0,5,"AB");
 傾いた文字を書き込む
     ・・・・
```

関数 Helpkey(文字列)

機能 関数の使用例をキーワードで検索する

説明 文字列に与えたキーワードで関数の使用例を検索し，コンソールに表示する。

【例】`Helpkey("直線");` とすると，コンソールに次のように表示される。

```
IntersectsgpL("",[p1,p2],[p3,p4,p5],"draw");
IntersectsgpL("R","P-Q","A-B-C");
IntersectsgpL("R","P-Q","A-B-C","put");
 空間の直線と平面の交点
Lineplot("1",[[2,1],[3,3]]);
     ・・・・
```

関数 Indexall(str1,str2);

機能 文字列 str1 から str2 を検索しその位置をすべて返す

説明 Cindyscript の indexof() の拡張版。indexof() が最初に見つかった位置を返すのに対し，Indexall() は存在する位置をすべてリストにして返す。

【例】str=”abcabcabc” から ”b”を検索する。

`indexof(str,"b")` では，2 が返る。

`Indexall(str,"b")` では，[2,5,8] が返る。

関数 Isptselected(点名）または Ptselected(点名)

機能 点が選択されていれば true, そうでなければ false を返す。

説明 点名はリストで与える。引数はなしにすることも可能で，その場合はすべての点が対象。

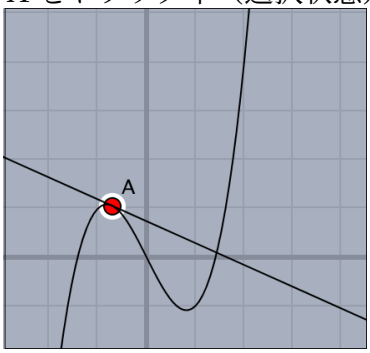
KeTCindy の関数の中には処理に時間がかかるものがある。その場合，点をドラッグするなど，画面上で操作をするとその都度再計算されるために，動きが非常に遅くなる。そこで，ドラッグする点をこの関数で指定すれば，ドラッグしている間は処理されないようにすることができる。

【例】点 A を原点近くにとっておき，次のスクリプトを実行する。

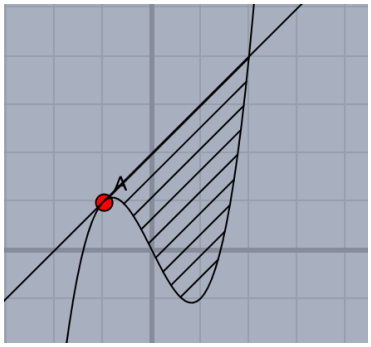
```
Deffun("f(x)",["regional(y)","y=x^3-2*x","y"]);
```

```
Plotdata("1","f(x)","x",["Num=100"]);
Putoncurve("A","gr1");
coef=Derivative("f(x)","x",A.x);
Defvar(["coef",coef]);
Deffun("g(x)",["regional(y)","y=coef*(x-A.x)+A.y","y"]);
Plotdata("2","g(x)","x",["Num=1"]);
if(!Ptselected(A),
Enclosing("1",["gr2","Invert(gr1)"],[A,"nodisp"]);
Hatchdata("1",["i"],[["en1"]]);
);
```

点 A をドラッグ中（選択状態）



点 A 以外をクリックして選択状態を解除



関数　Reparse(文字列か文字列のリスト)

機能　評価 (parse) してから実部をとる (re)。

説明　CindyJS では，実数の演算でも虚数の項が出ることがあり，その対応である。

【例】`str="(0-1)^2"; Reparse(str);`

注）CindyJS で format(parse(str),0) を実行すると `1-i*0` になる。

関数　Slider(名称，位置 1, 位置 2)

機能　スライダを作成する

説明　名称は ”A-C-B” の形で，端点を A,B，スライダ点を C としたスライダを作る。
端点 A,B の位置を，位置 1，位置 2 で指定する。単に”C”としたときは端点をとらない。スライダにより取得したい値は，点 C の座標（たとえば C.x）を利用する。
点 A,B,C はあらかじめ作図しておく必要はない。既にある場合はその点を使う。

【例】2 つのスライダを用意し，$y = a\sin(x - b)$ の a,b を変化させる。

```
Slider("A-C-B",[-5,-2],[5,-2]); // C is movable.
Slider("D-F-E",[-6,-2],[-6,2]); // F is movable.
Plotdata("1",Assign("y=a*sin(x-b)",["a",F.y,"b",C.x]),"x");
```

関数　Sprintf(実数, 長さ)
機能　小数点以下の長さを固定した文字列に変換
説明　実数を，小数点 n 位までの数とした文字列に変換する

【例】円周率
Sprintf(pi,2) は 3.14 を返す
Sprintf(pi,7) は 3.1415927 を返す
注）pi は Cindyscript の予約変数で，円周率を表す。

関数　Textformat(数，桁数)
機能　小数点以下の桁数を指定して数を文字列化する。
説明　第 1 引数は数のリストでもよい。数のリストの場合は，戻り値は，対応する数値を指定係数にした後，リストを文字列化する。Cindyscript の組み込み関数にも，format() という同様の関数があるが，format() は文字列のリストを返す。

【例】円周率を小数点以下 5 位までで文字列化する。

```
Textformat(pi,5);        format(pi,5);
```

戻り値は，いずれも `"3.14159"`

【例】第 1 引数がリストのときの，format() との戻り値の違い。

```
dt=[1/6,0.5];
Textformat(dt,4);  // 戻り値は ”[ 0.1667 , 0.5 ]”
format(dt,4);  // 戻り値は [ ”0.1667” , ”0.5” ]
```

関数　Texcom(TEX コード)
機能　TEX のコードを書き出す
説明　任意の TEX のコードを書き出す

関数 Windispg() または Windisp(データのリスト)

機能 定義されているプロットデータを Cinderella 画面に黒線で描く

説明 Windispg() は，スクリプトの最後に置くことで，出力される部分だけが黒で描かれるので，出力図を確認することができる。ただし，Letter() 関数で表示した点の名称などが Cinderella で作図したラベルと重なって表示されて見にくくなることもある。この関数を実行しなくても出力には影響しない。

Windisp(データのリスト) は，R から KETCindy 用に出力されたファイルを ReadOutData() 関数で読み込んだときに，必要なプロットデータ列だけを表示するのに用いる。

ReadOutData("filename.txt") でデータを読み込むと，そのデータに含まれるプロットデータ列が，コンソールに

Outdata of filename.txt：[Gfn,Gdfn,Gh]

のように表示される。

このうち，Gfn と Gh だけを表示するのであれば

```
Windispg([Gfn,Gh]);
```

とする。引数なしで

```
Windispg();
```

とすればすべてのプロットデータ列が表示される。

なお，いずれの場合も，作図したプロットデータも同時に表示される。

作図した図を全てではなく選択して表示する場合は，それらのプロットデータ名をリストにして引数とする。

たとえば，sg1, gr1, crAB が定義されているとき，

```
Windispg(["sg1","gr1"]);
```

とすれば，sg1,gr1 のみが表示される。

関数 Viewtex()

機能 TEX のソースファイルを書き出す。引数なし。

説明 グローバル変数 Fhead で定義したファイル名に "main" を付加した TEX のソースファイルとバッチファイル（Mac の場合はシェルファイル）を作成する。

関数 Workprocess()

機能 作図の経過を取得する

説明 作図ツールを用いた作図の経過を取得する。

```
println(Workproccess());
```

とすると，コンソールに作図手順が表示される。

関数 Op(n,list or str)

機能　リストまたは文字列から要素を抜き出す

説明　第 2 引数のリストまたは文字列の n 番目の要素（文字）を返す。
Cindyscript の アンダーバーの演算子 (list_n , str_n) と同様。

関数　Strsplit(文字列 , 文字)

機能　文字列を分解する。

説明　第 1 引数の文字列を第 2 引数の文字の位置で分解したリストを返す。

【例】文字 a で区切って分解する。

```
str="abcadeaf";
strL=Strsplit(str,"a"); //[" "  ," bc" ," de" ," f" ] を返す。
```

同様の関数に，Cindyscript の tokenize(文字列, 文字列) がある。tokenize() の第 2 引数は文字列や，文字のリストでもよい。

⇒ 関数一覧

関数　Fracform(数 , 分母のリストまたは最大値 [, 許容誤差の桁数 (5)])

機能　分母リストの 1 つを分母とする近似分数と誤差を返す。

説明　戻り値は，簡易 TeX-like 書式の文字列，誤差，分子，分母。

【例】`Fracform(0.33,[2,3]); => [fr(1,3),''err=0.0033'',1,3]`

⇒ 関数一覧

関数　Totexform(簡易 TeX-like 書式)

機能　TeX 書式の文字列を返す。

【例】`Totexform(fr(1,3)); => frac{1}{3}`

⇒ 関数一覧

関数　Tocindyform(簡易 TeX-like 書式)

機能　Cindy 書式の文字列を返す。

⇒ 関数一覧

2　他の数式処理ソフトなどとの連携

2.1　R との連携

R は主に統計解析のためのソフトウェアで，binorm（二項分布），pois（ポアソン），unif（一様分布），chisq（カイ 2 乗），f（F 分布），t（t 分布）など，多くの確率分布をサポートしている。

KETCindy では，kc.bat/sh によってコマンドを R に渡し，結果をテキストファイルで受け取る。このとき，R とのやりとりで，次のようなファイルが作業ディレクトリに作成される。

拡張子 r ：r 用のファイル

拡張子 dat，拡張子 txt：データファイル

このデータのやり取りに関する次のオプションがある。

オプションなしまたは,”　のとき

i) データファイルがなければ，新しく作る

ii) データファイルが既にあればそれを読み込む

”m”のとき，強制的にデータファイルを作り直す。

”r” のとき，すでにあるデータファイルを読み込む。

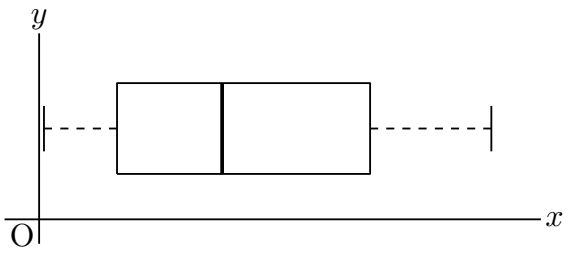
関数　Boxplot(名前, データ, 垂直位置, 箱の高さ,option)

機能　箱ひげ図を描く

説明　データは，リストで渡す場合とファイル名を渡してファイルから読み込む場合がある。データファイルは csv 形式とする。

【例】乱数で作成した 5 未満の実数のデータを箱ひげ図にする。

```
dt1=apply(1..100,5*random());
Boxplot("1",dt1,1,1/2);
```

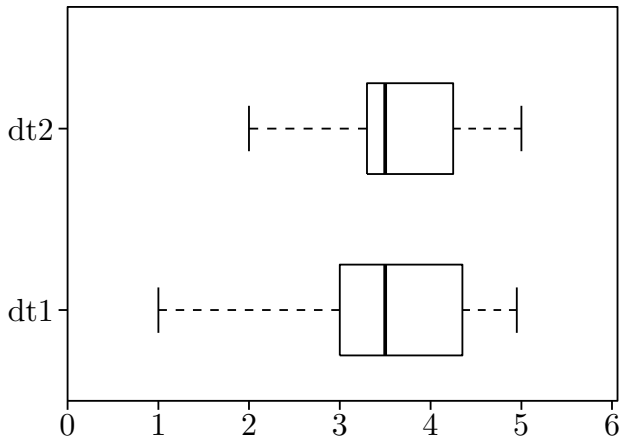


【例】外部ファイルとして用意したデータを読み込んで箱ひげ図にする。

```
Boxplot("2","datafile.csv",3,1/2);
```

複数列から成る csv ファイルを読み込むには，Readcsv を使う。csv ファイルは，作業フォルダ（ 初期設定は fig ）に入れておく。戻り値は読み込んだファイル。
データの値を画面に入るように調節するには，dt1/20 のようにしてリサイズする。
また，Framedata()，Rulerscale() を併用することで目盛を入れることができる。
Framedata() のために，表示領域の対角点 A,B を Cinderella の作図ツールで作図しておく。

```
data=Readcsv("datafile.csv");
dt1=apply(data,#_1);
dt2=apply(data,#_2);
Boxplot("1",dt1/20,1,1/2);
Boxplot("2",dt2/20,3,1/2);
Framedata("1",[A,B],["corner"]);
Rulerscale(A,["r",0,6,1],["f",1,"\mbox{dt1}",3,"\mbox{dt2}"]);
```



注）一度実行した後，データを書き直すと，図が更新されないので，”m” オプションをつけて Boxplot(”1”,dt1/20,1,1/2,[”m”]); とすると，図が更新される。データを書き出すときは，もう一度 ”m” オプションをはずして実行してから Figure ボタンを押す。これは，データの作成タイミングの関係。

⇒ 関数一覧

関数 Rfun(name, コマンド, 引数, option)

機能 R の 1 つのコマンドを実行して結果を返す

説明 バッチファイル kc.bat / シェルファイル kc.sh を利用して R とデータをやり取りし，計算結果を取得する。結果は，変数 R+name に入り，コンソールにも表示される。

【例】R を用いて標準正規分布から 10 個の乱数を発生し，戻り値から平均値と標準偏差を求めてコンソールに表示する。

```
Rfun("1","rnorm",[10]);
nx=length(R1);
mx=sum(R1)/nx;
sx=sqrt(R1*R1/nx-mx^2);
println("平均："+format(mx,4)+"標準偏差："+format(sx,4));
```

関数 CalcbyR(変数名, コマンド列，option)

機能 R のコマンドを実行して結果を返す

説明 バッチファイル kc.bat / シェルファイル kc.sh を利用して R とデータをやり取りし，計算結果を取得する。

コマンド列は，”戻り値=コマンド”，[引数] の 2 つをセットとして並べる。

最後の行の結果が戻り値として第 1 引数の変数名に代入される。”戻り値 1::戻り値 2・・”,[] の形（戻り値 1, 戻り値 2・・は各コマンドの戻り値）でコマンドを書くと，戻り値 1,・・のリストとなる。戻り値が一つの場合は実数。”=値”,[] の形の場合，「値」がそのまま戻り値となる。

【例】R を用いて N(50,5^2) から 10 個の乱数を発生し，平均と不偏分散も R で計算してその結果をコンソールに表示する。

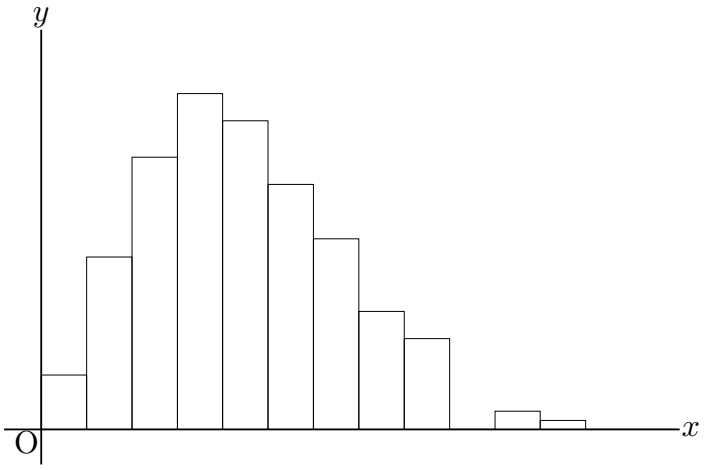
```
cmdL=[
"tmp1=rnorm",[10,50,5],
"tmp2=mean",["tmp1"],
"tmp3=var",["tmp1"],
"tmp1::tmp2::tmp3",[]
];
CalcbyR("rd",cmdL);
dt=rd_1;
mx=rd_2;
vx=rd_3;
println("データ："+dt);
println("平均："+format(mx,4)+"不偏分散："+format(vx,4));
```

CalcbyR() によって，データと平均，不偏分散からなるリストが作成されるので，mx に平均，vx に不偏分散を代入している。`rd_(-1)` は，リスト rd の末尾の要素。

【例】Rでポアソン分布から 200 個の乱数をとり，標本平均の分布の様子＝分散が小さくなって，正規分布に近づいている様子＝をヒストグラムで見る。分散はRで求めた不偏分散に (n-1)/n をかけて再計算してコンソールに表示する。

```
cmdL=[
"tmp1=rpois",[200,5],
"tmp2=mean",["tmp1"],
"tmp3=var",["tmp1"],
"=c(tmp2,tmp3,tmp1)",[]
];
```

```
CalcbyR("rd",cmdL);
dt=rd_(3..length(rd));
n=length(dt);
mx=rd_1;
vx=rd_2*(n-1)/n;
sx=sqrt(vx);
println(dt);
println(["m="+format(mx,4),"v="+format(vx,4)]);
Setscaling(1/5);
Histplot("1",dt,["Breaks=seq(0,14,1)","dr,0.5"]);
```



【例】ポアソン分布で乱数を 2000 個発生させ，10 個ずつの平均をRで計算し，ヒストグラムを作る。

```
cmdL=[
"tmp1=rpois",[2000,5],
"tmp2=c()",[],
"for(k in 1:200){",[],
"tmp=tmp1[(10*(k-1)+1):(10*k)]",[],
"tmp2=c(tmp2,mean(tmp))",[],
"}",[],
"=tmp2",[]
];
CalcbyR("rd2",cmdL);
Setscaling(1/10);
Histplot("2",rd2);
```

⇒ 関数一覧

関数 Histplot(name,data,option)

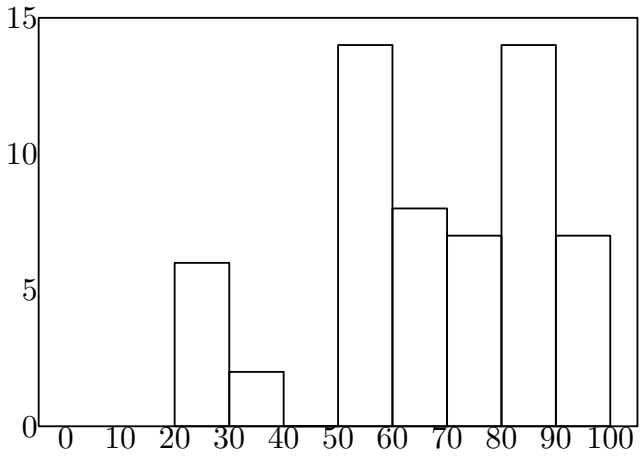
機能 R を利用してヒストグラムを描く

説明 data はリストにして作成するか，外部ファイルから Readcsv() で読み込む。戻り値は，階級境界値と，対応する度数のリスト。

階級境界値（ブレークポイント）は，自動的に設定される（スタージェスの公式による）が，オプションで，
"breaks=[0,10,20,30,40,50,60,70,80,90,100]"
などと指定することもできる。
この他のオプションは
”Den=yes/no”：密度の指定（初期値は no）
”Rel=yes/no”：相対度数にする/しない（初期値は no）

【例】csv ファイル（datafile.csv）を読み込み，ヒストグラムを作る。Framedata() と Rulerscale() を併用して，目盛付きの枠の中に表示する。表示枠の対角点 A,B は Cinderella の作図ツールで作図しておく。

```
Addax(0);
Setscaling(5);
Setunitlen("0.6mm");
data=Readcsv("datafile.csv");
Histplot("1",data,[""]);
Framedata("1",[A,B],["corner"]);
Rulerscale(A,["r",0,100,10],["r",0,15,5]);
```



2 行目と 3 行目は，データに合わせて縦方向を 5 倍にし，TeX の単位長を 0.6mm にしている。
Den,Rel オプションを yes にしたときは，Setscaling(100) くらいにするのがよい。
csv ファイルが複数のデータからなる場合は，
dt1=data_1; として，リストの第 1 要素を取得する。第 2 要素のヒストグラムであれば data_2 とする。

⇒ 関数一覧

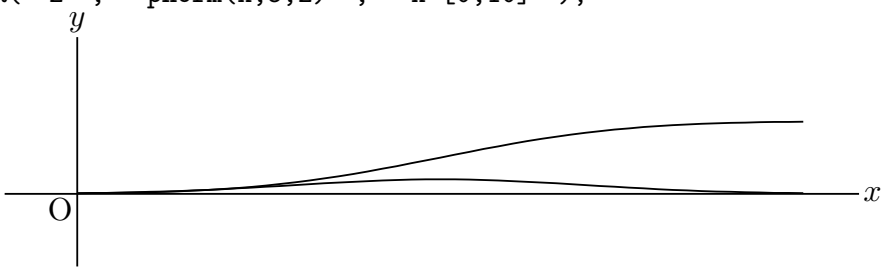
関数　PlotdataR(name, 式, 変数)

機能　R の関数のグラフを描く

説明　Cindyscript の組み込み関数にはない関数のグラフを R を利用して描く。

【例】平均 5, 標準偏差 2 の正規分布の密度関数と分布関数のグラフを描く。

```
PlotdataR(“ 1” , “ dnorm(x,5,2)” , ” x=[0,10]” );
PlotdataR(“ 2” , ” pnorm(x,5,2)” , ” x=[0,10]” );
```



【例】標準正規分布のグラフ上の点と x 軸を結んだ線分を描く。
点 A,B は Cinderella の作図ツールで作図しておき，点 A をグラフ上のおよその位置に置いてから実行する。

```
PlotdataR("1","dnorm(x)","x=[-5,5]");
Putoncurve("A","grR1",[-3,3]);
Putpoint("B",[A.x,0]);
Listplot("1",[A,B]);
```

2 行目の最後の引数の [-3,3] は，その範囲を動かすことを意味する。
A はグラフ上を動かすことができて，B はそれに伴って動く。ただし，少し動かす度に バッチ/シェル ファイルを実行するので，煩雑な場合は，Plotdata() の行をコメント化してから点 A を動かしたあと再実行するとよい。

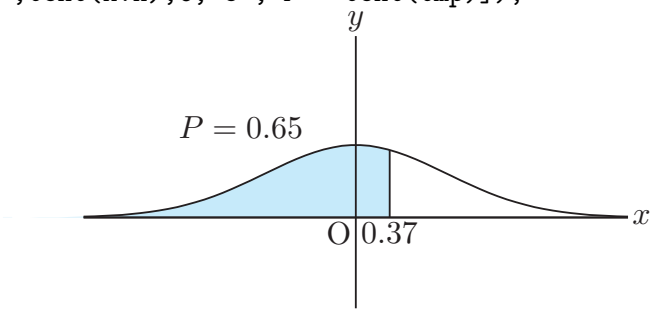
【例】上と同様で，x 軸上の点を自由点 A とし，曲線上に B を置く。

```
PlotdataR("1","dnorm(x)","x=[-5,5]");
PlotdataR("1","dnorm(x)","x=[-5,5]");
A.xy=[A.x,0];
Lineplot("1",[A,A+[0,1]],["nodisp"]);
Putintersect("B","grR1","ln1");
Listplot("1",[A,B]);
```

【例】前の例のグラフで，AB の左側に Shade をかけ，Shade の部分の面積を求める。
P の値を表示する位置に，Cinderella の作図ツールで点 C をとっておく。

```
PlotdataR("1","dnorm(x)","x=[-5,5]",["Num=100"]);
Putpoint("A",[0,0],[A.x,0]);
Lineplot("1",[A,A+[0,1]],["nodisp"]);
```

```
Putintersect("B","grR1","ln1");
Listplot("1",[A,B]);
Listplot("2",[[-5,0],[5,0]],"nodisp");
Enclosing("1",["Invert(grR1)","sg2","sg1"],[B,"notex"]);
Shade(["en1"],["Color=[0.2,0,0,0]"]);
tmp=0.5+Integrate("grR1",[0,A.x]);
Expr([A,"s",text(A.x),C,"e","P="+text(tmp)]);
```



⇒ 関数一覧

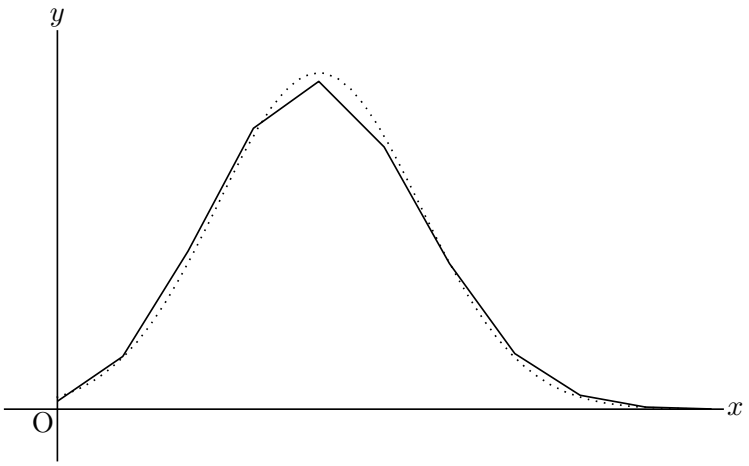
関数　PlotdiscR(name, 式, 変数)

機能　R を利用して離散型のグラフを描く

説明　dbinom （二項分布），dpois（ポアソン分布），dgeom（幾何分布）など離散型確率分布のグラフを描く。

【例】二項分布のグラフと正規分布のグラフを比較する。

```
Setscaling(20);
PlotdiscR("1","dbinom(k,10,0.4)","k=[0,10]");
PlotdataR("1","dnorm(x,10*0.4,sqrt(10*0.4*0.6))","x=[0,10]",["do"]);
```



【例】ポアソン分布および幾何分布のグラフ。

```
PlotdiscR("2","dpois(k,4)","k=[0,10]");
PlotdiscR("3","dgeom(k,0.3)","k=[0,10]");
```

⇒ 関数一覧

関数 Scatterplot(name,filename/datalist,option1,option2)

機能 2 次元データを読み込み，散布図を描く

説明 外部ファイル filename（csv 形式）を読み，散布図を描く。
外部ファイルの 2 次元データとは，次の形の csv ファイル。（行末は LF または CR）

```
2.3, 4.5 (LF)
3.2, 7 (LF)
2.0, 6.8 (LF)
```

datalist の場合は，次の形。

```
data=[[2.3,4.5],[3.2,7],[2.0,6.8], ••• ];
```

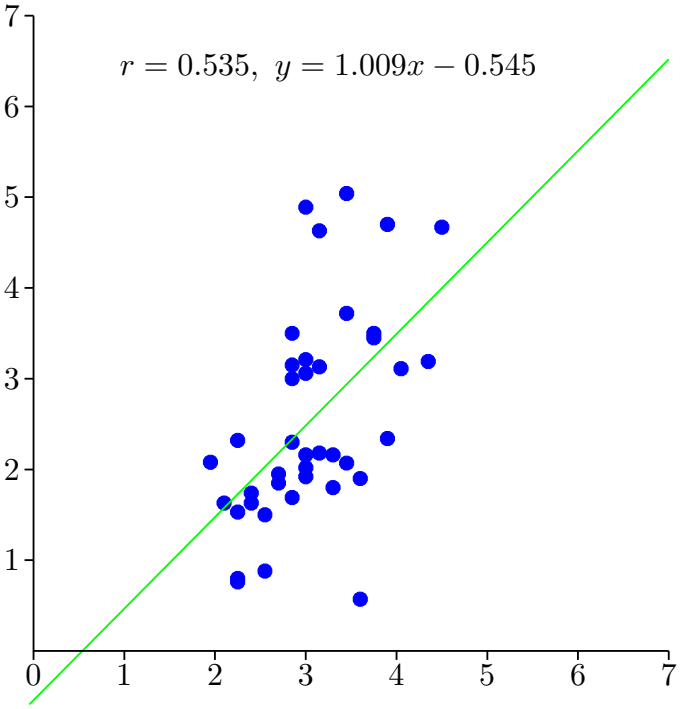
第 1 オプションは，回帰直線を描くかどうかと点のスタイル。
”Reg=no”：回帰直線を描くかどうか（yes/no) 初期値は yes
第 2 オプションは，相関係数と回帰直線の式を表示する位置と，回帰直線のスタイル。
位置は，幾何点の名称でもよい。

【例】data.csv を読んで散布図を描き，回帰直線を引く。

```
Scatterplot("1","data.csv");
```

だけで描ける。オプションをつけた例は次。
点 A を相関係数と回帰直線の式を表示する点として Cinderella の作図ツールで取る。
点を青で大きさ 2 とし，回帰直線を緑で表示する。

```
Scatterplot("1","data.csv",["Size=4","Color=blue"],[A,"Color=green"]);
Listplot("1",[[0,7],[0,0],[7,0]]);
Rulerscale([0,0],["r",0,7,1],["r",1,7,1]);
```



⇒ 関数一覧

2.2 Maxima との連携

Maxima は数式処理ソフトで，KETCindy においては微積分の計算など，Cindyscript では不十分な点を補うことができる。

KETCindy では，kc.bat/sh によってコマンドを Maxima に渡し，結果をテキストファイルで受け取る。このとき，Maxima とのやりとりで，次のようなファイルが作業ディレクトリに作成される。

拡張子 max ：Maxima に渡すコマンドを記述したファイル

拡張子 txt：Maxima が出力したデータファイル

このデータのやり取りに関する次のオプションがある。

オプションなしまたは,”　のとき

i) データファイルがなければ，新しく作る

ii) データファイルが既にあればそれを読み込む

”m”のとき，強制的にデータファイルを作り直す。

”r” のとき，すでにあるデータファイルを読み込む。

このとき，ファイルの読み書きで不具合があると，数秒の後「==> file.txt not generated (5 s) 」のようなエラーメッセージがコンソールに表示される。このような場合は作業ディレクトリの設定などを確認していただきたい。この待ち時間については，Wait オプションで設定することもできる。

関数 CalcbyM(name，コマンド，option)

機能 Maxima のスクリプトを実行する

説明 第 2 引数は Maxima で実行するコマンドを記述したスクリプト。
コマンドと引数リストの繰り返しからなるスクリプトをリストとして作り，一度に実行する。
戻り値はない（未定義値)。実行結果は，コマンドリストの最後に記述した変数（引数は空リスト）の値が，name で指定された変数に代入される。複数の結果を戻すときは，:: で区切って記述するとリストにして name に代入される。

【例】$\sin x$ とその導関数を表示する。

```
cmdL=[
  "f:sin(x)", [],
  "df:diff",["sin(x)","x"],
  "f::df",[]
];
CalcbyM("fdf",cmdL);
println(fdf);
```

実行すると，結果の f と df のリストが変数 fdf に代入され，コンソールに，`[sin(x),cos(x)]` と表示される。

【例】2 次方程式 $x^2 - x - 4 = 0$ の解を求める。

```
cmdL=[
  "ans:solve",["x^2-x-4","x"],
  "ans",[]
];
CalcbyM("ans",cmdL);
println("ans="+ans);
```

コンソールには
`ans=[x = -(sqrt(17)-1)/2,x = (sqrt(17)+1)/2]`
が表示される。

応用例１：曲線の接線を引く

$f(x) = \dfrac{e^x + e^{-x}}{2}$ の，$x = a$ における接線の方程式を作る。

```
fx="(exp(x)+exp(-x))/2";
cmdL=[
  "df:diff",[fx,"x"],
```

```
    "c:ev",["df","x=a"],
    "b:ev",[fx,"x=a"],
    "eq:c*(x-a)+b",[],
    "eq",[]
];
CalcbyM("tn1",cmdL);
println(tn1);
```

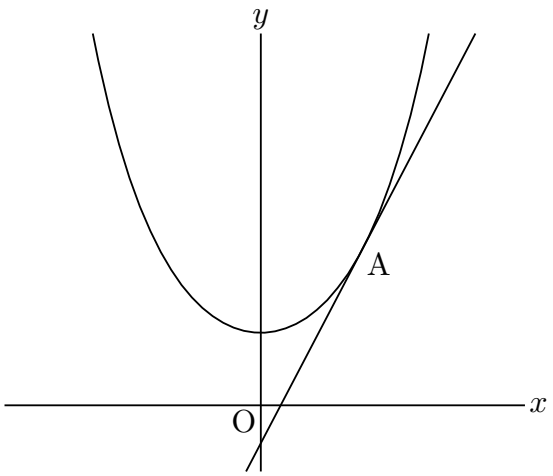
コンソールには

```
(%e^a-%e^-a)*(x-a))/2+(%e^a+%e^-a)/2
```

が表示される。
この，CalcbyM の戻り値 tn1 を用いて，曲線上の 1 点 A における接線のグラフを描く。以下のスクリプトを追加する。なお，点 A を Cinderella の作図ツールで適当なところにとっておく。

```
tn1=Assign(tn1,["%e^a","exp(a)","%e^-a","exp(-a)"]);
Plotdata("1",fx,"x");
Putoncurve("A","gr1");
tmp=Assign(tn1,["a",A.x]);
plotdata("2",tmp,"x",["Num=2"]);
```

1 行目では Maxima で作成した式を，Cindyscript でプロットできる式にしている。

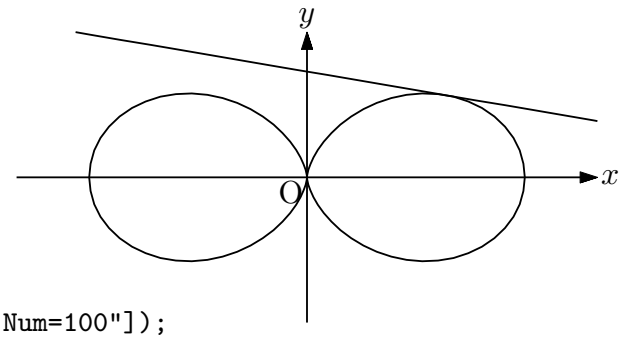


なお，接線の方程式を求めるだけであれば，Mxfun() を使うこともできる。Mxfun() の解説を参照のこと。

応用例 2：パラメトリックの場合の接線

媒介変数の値を決めるために，点 A を Cinderella の描画面の x 軸上にとっておき，その x 座標を媒介変数 t の値とする。スライダを作ってもよい。

```
fn="3*cos(t)^2*[cos(t),sin(t)]";
cmdL=[
  "f:",[fn],
  "df:diff",["f","t"],
  "df:trigsimp",["df"],
  "tn:f+s*df",[],
  "tn",[]
];
CalcbyM("tn2",cmdL);
Paramplot("1",fn,"t=[0,2*pi]",["Num=100"]);
gn=Assign(tn2,["t",A.x]);
Paramplot("2",gn,"s=[-3,3]");
```



cmdL で定義している Maxima のコマンド（trigsimp など）については，Maxima の解説書などを参照されたい。

【例】定積分の値を求める。

$\int_{-1}^{2}(-x^3+3x+2)dx$ の値を求める。結果は val で受け取り，Mxtex() に渡して，TeX 書式にして表示する。

```
cmdL=[
  "val:integrate",["-x^3+3*x+2,x,-1,2"],
  "val",[]
];
CalcbyM("val",cmdL);
Mxtex("1",val);
Expr([[2,2],"e","S="+tx1]);
```

関数 Mxbatch(ファイル名)

機能 Maxima のライブラリを使うバッチファイルを作成

説明 ketcindy/ketlib/maximaL にあるライブラリを用いるコマンドでバッチファイルを作成する。

ketcindy/ketlib/maximaL には, fourier_sec.max , matoperation.max , poincare.mac の 3 つのファイルがあり，関数を定義したライブラリが入っている。それぞれ次のような内容である。各ファイルをテキストエディタで開いて参照されたい。

fourier_sec.max ：フーリエ級数の計算を行う。

matoperation.max ： 行列の計算を行う。

poincare ： ハミルトニアンシステムに関する計算を行う。

【例】`cmd=Mxbatch("fourier_sec")`

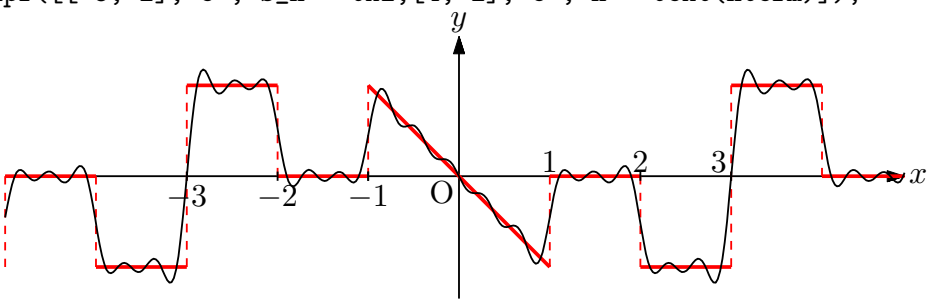
を実行すると，cmd に，たとえば

```
[batch,["/Applications/ketcindy/ketlib/maximaL/fourier_sec.max"]]
```

が代入される。（Mac/Windows および KeTCindy の設定によって異なる）

そこで，次のように利用する。

```
Setax(["a"]);
Slider("A-C-B",[-5.5,-1.5],[4.5,-1.5]);
defL=["1",[-3,-2],1,"0",[-2,-1],1,"-x",[-1,1],1,"0",[1,2],1,"-1",[2,3],1];
Drwxy();
tmp=Periodfun(defL,1,["dr,2","Color=red"]);
fun=tmp_1;
per=tmp_2;
Htickmark([1,"n","1",2,"n","2",3,"nw","3"]);
Htickmark([-1,"-1",-2,"-2",-3,"-3"]);
cmdL=Concat(Mxbatch("fourier_sec"),[
  "Ffun(x):="+fun,[],
  "c:fourier_sec_coeff",["Ffun(x)","x"],
  "c[1]::c[2]::c[3]",[]
]);
CalcbyM("ans",cmdL,[]);
nterm=round(4*(C.x-A.x));
Fourierseries("1",ans,per,nterm,["Num=400"]);
Mxtex("2",ans_3);
Expr([[-5,-2],"e","s_n="+tx2,[4,-2],"e","n="+text(nterm)]);
```



$$s_n = -\frac{2\left(\pi\, n\, \cos\left(\frac{2\,\pi\, n}{3}\right)+3\, \sin\left(\frac{\pi\, n}{3}\right)-\pi\, n\, \cos\left(\frac{\pi\, n}{3}\right)-\pi\, n\,(-1)^n\right)}{\pi^2\, n^2} \qquad n = 15$$

関数　Mxfun(name，式，リスト，option)

機能　Maxima の関数を実行する

説明　第 2 引数の「式」は Maxima の関数名。第 3 引数のリストは関数に渡す引数のリスト。戻り値は，第 1 引数の式に 1 つでも文字があると文字列となる。すべて数字（+,-, . を含む）の場合は 16 桁以下であれば数，それ以上の場合は文字列となる。また，戻り値は，変数 mx+name にも代入される。

オプションに ”Disp=no” をつけると，結果をコンソールに表示しない。

【例】10! を求める。

```
Mxfun("1","10!",[]);
```

を実行すると，コンソールに mx1 is 3628800 と表示される。この値は変数 mx1 に代入されているので,

```
Letter([[0,1],"e",mx1]);
```

とすれば Cinderella の描画面上に表示される。mx1 ではなく，戻り値を変数に代入して使うこともできる。

```
fact10=Mxfun("1","10!",[]);
Letter([[0,1],"e",fact10]);
```

【例】文字列の連結

文字列を引数とする場合, 例えば, Maxima の文字列を連結するコマンド concat では,

```
concat("a","b")
```

とするが，中にダブルクォートが入っているため，全体を文字列にすることができない。

このような場合は，第 2 引数を使って

```
Mxfun("1","concat",["a","b"])
```

とすればよい。

＜参考＞ CindyScript の unicode(code) 関数を用いてダブルクォートを表し，次のようにすることもできる。

```
dq=unicode("22");
comm="concat("+dq+"a"+dq+","+dq+"b"+dq+")";
Mxfun("1",comm,[]);
```

【例】$f(x) = \sin x$ を微分する

```
Mxfun("1", "diff",["sin(x)","x"])
```

とすると

```
diff(sin(x),x)
```

というコマンドを Maxima に渡して，戻り値を Cindy の変数 mx1 に代入する。

```
Mxfun("1", "diff(sin(x),x)",[]])
```

と，第 1 引数にまとめても同じ結果になる。ただし，この場合，第 2 引数は空リストとする。

＜参考＞ Cindyscript の微分との違い

Cindyscript でも微分はできる。たとえば,

```
f(x):=sin(x);
g(x):=d(f(#),x);
```

```
plot(g(#));
```

とすると，cos(x) のグラフが描かれる。
しかし，Cindyscript の微分が，微分の定義による数値計算であるのに対し，Maxima では数式処理として微分ができる。
その意味の違いは，次のスクリプトで確かめられる。

```
f(x):=sin(x);
g(x):=d(f(#),x);
println(g(x));
```

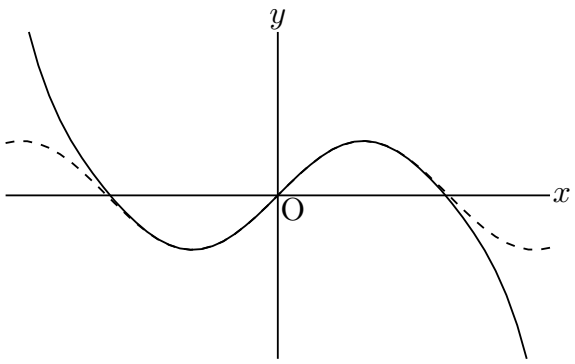
では，コンソールに表示されるのは未定義値（_ _ _） である。
一方，

```
Mxfun("1", "diff",["sin(x)","x"]);
println(mx1);
```

では，コンソールに `cos(x)` と表示される。
mx1 は文字列であるので，`Plotdata("1",mx1,"x")` でグラフが描ける。
また，Cindyscript の微分では，3 階か 4 階までの導関数が計算上の限度であるのに対し，Maxima なら何階でも微分ができるので，テイラー展開などで有利である。

【例】$\sin x$ の テイラー展開による近似のグラフを表示する。

```
Mxfun("1","taylor",["sin(x)","x",0,7],[""]);
Plotdata("1","sin(x)","x",["da"]);
Plotdata("2",mx1,"x");
```



なお，Mxtex() を用いれば，Mxfun() の結果の mx1 を TeX 書式にして表示できる。

```
Expr([[1,2],"e",Mxtex("1",mx1)]);
```

を追加すれば [1,2] の位置に式が表示される。

【例】接線の方程式を作る

$f(x)=\dfrac{e^x+e^{-x}}{2}$ の，$x=a$ における接線の方程式を作る。
関数式を文字列にしておき，Assign() を用いて変数 x を a に変えれば，$f(a)$ の式を作ることができる。導関数についても同様にする。

```
fx="(exp(x)+exp(-x))/2";
gx=Mxfun("1","diff",[fx,"x"]);
fa=Assign(fx,["x","a"]);
ga=Assign(gx,["x","a"]);
tf=ga+"*(x-a)+("+fa+")";
println(tf);
```

コンソールには

```
(%e^a-%e^-a)/2*(x-a)+((exp(a)+exp(-a))/2)
```

が表示される。
同様の例を CalcbyM() で例示しているので参照されたい。

関数 Mxtex(name, 式)

機能 式を TeX 書式にする

説明 第 2 引数の式は，直接書いた式もしくは Mxfun の戻り値。これを TeX の書式にする。戻り値は，変数 txname にも代入される。

【例】部分分数への分解

部分分数 $\dfrac{x^3}{(x+1)(x+2)}$ の分解を Maxima で行い，その結果を TeX 書式にして画面に表示する。画面に表示された結果はそのまま KETCindy で出力できる。

```
Mxfun("1","partfrac",["x^3/((x+1)*(x+2))","x"]);
Mxtex("1",mx1);
Expr([0,1],"e",tx1);
```

ここで，mx1，tx1 はそれぞれ Mxfun(”1”,・・) , Mxtex”1”,・・) の結果（戻り値）である。mx1，tx1 はコンソールにも表示され，tx1 は次のようになっている。

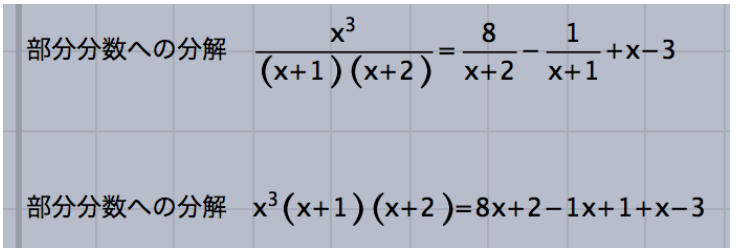
```
\frac{8}{x+2}-\frac{1}{x+1}+x-3
```

Cindyscript は TeX 書式をサポートしているのでこれで描画面に分数式が表示されるが，Tex の文書では，`\frac{}{}` ではなく，`\dfrac{}{}`を使うことが多い。そこで，

Assign() を用いて，”frac” を ”dfrac” に変えれば，そのまま Tex 文書で使える。ただし，Cindyscript は `\dfrac{}{}` をサポートしていないので，画面上では分数表記にならない。そのあたりの事情を次のスクリプトで示す。

```
fx="x^3/((x+1)*(x+2))";
pfx=Mxfun("1","partfrac",[fx,"x"]);
form=Mxtex("1",fx)+"="+Mxtex("2",pfx);
dform=Assign(form,["frac","dfrac"]);
Letter([0,5],"e","部分分数への分解$"+form+"$");
Letter([0,3],"e","部分分数への分解$"+dform+"$");
```

Cinderella の描画面では次のように表示される。



出力した TeX 挿入図では次のようになる。

Decomposition into partial fractions

$$\frac{x^3}{(x+1)\,(x+2)} = \frac{8}{x+2} - \frac{1}{x+1} + x - 3$$

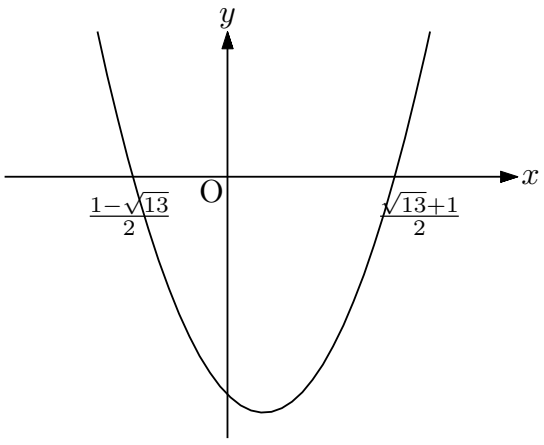
なお，文字列を置換するのに，`Assign(form,["frac","dfrac"])` ではなく，Cindyscript の文字列の関数 replace を用いて，

```
dform=replace(form,"frac","dfrac");
```

としてもよい。

【例】2 次関数のグラフを表示し，x 軸との交点の x 座標を表示する。

```
fx="x^2-x-3";
cmdL=[
"ans:solve",[fx,"x"],
"ans",[]
];
CalcbyM("ans",cmdL);
p1=indexof(ans,"[");
p2=indexof(ans,",");
p3=indexof(ans,"]");
s1=substring(ans,p1,p2-1);
```



```
s2=substring(ans,p2,p3-1);
s1=replace(s1,"x =","");
s2=replace(s2,"x =","");
Mxtex("1",s1);
Mxtex("2",s2);
Plotdata("1",fx,"x");
Expr([-2,-0.5],"e",tx1);
Expr([2,-0.5],"e",tx2);
```

ここで，CalcbyM("ans",cmdL); で得られる ans は，次のような文字列である。

"[x = -(sqrt(13)-1)/2,x = (sqrt(13)+1)/2] "

そこで，ここから 2 つの式だけを抽出する作業を行ったのち，Mxtex() で TeX の式を得ている。

さらに応用として，点 A を Cinderella の作図ツールで作図し，

```
if(A.y<0,
fx="(x-"+text(A.x)+")^2"+guess(A.y),
fx="(x-"+text(A.x)+")^2+"+guess(A.y);
);
```

とすると，点 A を頂点とする放物線と軸との交点の座標が描かれる。Maxima とのデータのやり取りをするためのタイムラグがあるが，インタラクティブに放物線の位置を変えることができる。

＜参考＞

2 次関数のような簡単な関数であれば，Cindyscript の roots() 関数を用いて 2 次方程式が解けるので，次のスクリプトでほぼ同じ動作をするものを作ることができる。「ほぼ」というのは点 A の位置によっては，guess() で解釈しきれないことがあるためである。Maxima を使えば数式処理で解を求めるので，A がどこにあってもきれいに表示できる。

```
fx="x^2-2*A.x*x+A.x^2+A.y";
cf=[A.x^2+A.y,-2*A.x,1];
sol=roots(cf);
s1=guess(sol_2);
s2=guess(sol_1);
Mxtex("1",s1);
Mxtex("2",s2);
Plotdata("1",fx,"x");
Expr([-2,-0.5],"e",tx1);
Expr([2,-0.5],"e",tx2);
```

⇒ 関数一覧

2.3 Risa/Asir との連携

関数 CalcbyA(name，コマンド，option)

機能 Risa/Asir のスクリプトを実行する

説明 第 2 引数は Risa/Asir で実行するコマンド。
コマンドと引数リストの繰り返しからなるリスト（例えば cmdL）を作って，一度に実行する。
戻り値はない。（未定義値）結果は，コマンドリストの最後に記述した変数（引数は空リスト）の値が name で指定された変数に代入される。複数の結果を戻すときは，:: で区切って記述するとリストにして代入される。

関数 Asirfun(name，式，リスト，option)

機能 Risa/Asir の関数を実行する

説明 第 2 引数の「式」は Risa/Asir の関数名。第 3 引数のリストは関数に渡す引数のリスト。
戻り値は，第 1 引数の式に 1 つでも文字があると文字列となる。すべて数字（+,-, . を含む）の場合は 16 桁以下であれば数，それ以上の場合は文字列となる。また，戻り値は，変数 asname にも代入される。
オプションに ”Disp=no” をつけると，結果をコンソールに表示しない。

⇒ 関数一覧

2.4 FriCAS(Axiom) との連携

関数 CalcbyF(name，コマンド，option)

機能 FriCAS のスクリプトを実行する

説明 第 2 引数は FriCAS で実行するコマンド。
コマンドと引数リストの繰り返しからなるリスト（例えば cmdL）を作って，一度に実行する。
戻り値はない。（未定義値）結果は，コマンドリストの最後に記述した変数（引数は空リスト）の値が name で指定された変数に代入される。複数の結果を戻すときは，:: で区切って記述するとリストにして代入される。

関数 Frfun(name，式，リスト，option)

機能 FriCAS の関数を実行する

説明 第 2 引数の「式」は FriCAS の関数名。第 3 引数のリストは関数に渡す引数のリスト。
戻り値は，第 1 引数の式に 1 つでも文字があると文字列となる。すべて数字（+,-, . を

含む）の場合は 16 桁以下であれば数，それ以上の場合は文字列となる。また，戻り値は，変数 friname にも代入される。
オプションに ”Disp=no” をつけると，結果をコンソールに表示しない。

⇒ 関数一覧

2.5 MeshLab との連携

MeshLab は，3D データ（obj データなど）を読み込んでレイトレーシングで表示・編集するソフトウェアである。レイトレーシングで 3D グラフィクスを描くには，Cinderella と親和性の高い Cindy3D を利用するのがよいが，MeshLab を使うメリットは 3D プリンタ用の STL ファイルを出力できることである。また，KETCindy で描いた 3D の図がレイトレーシングでどのようになるのかを見ることも比較的簡単にできる。

MeshLab との連携は，KETCindy から Obj 形式のデータを書き出すことで行う。Mkobj**() 関数でデータを作り，Mkviewobj() 関数で MeshLab を呼び出して表示を行う。

なお，Mkviewobj() 関数で MeshLab を呼び出して表示を行う場合，これを Draw スロットに書くと頻繁に呼び出しが行われるため非効率となる。そこで，if(1==0,・・・) で・・・の部分に MeshLab の呼び出し関係のスクリプトを書いて，実際に呼び出すときに if(1==1,・・・) とする方法と，呼び出し関係のスクリプトを関数化してボタンに割り当てる方法がある。ketcindy パッケージに含まれる sample にボタンをつけたものがある。

なお，3D であるので，Initialization スロットに
`Ketinit(); Ketinit3d();` を記述しておく。

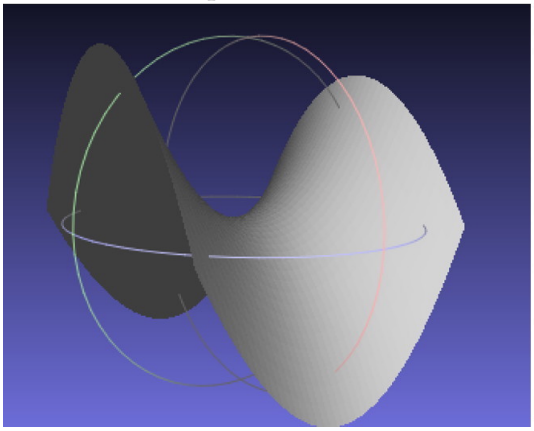
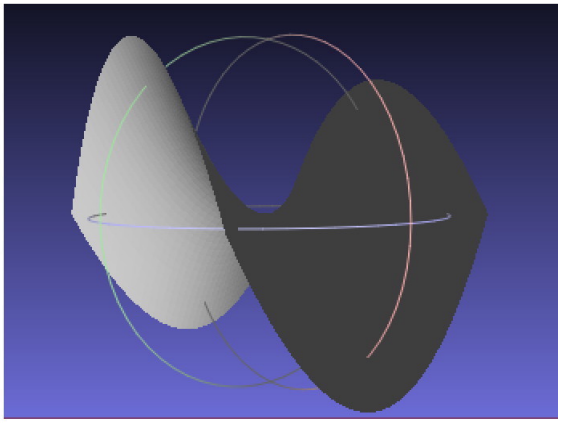
関数 Mkobjcmd(name，式，option)
機能 厚みを持たない曲面の obj ファイルのためのコマンドを作成する
説明 オプションは [分割数 1，分割数 2，表側の方向の指定]
表側の方向は，変数に対して，右手系の方向が”+”
作成されるデータは”oc”+name のファイル名の obj データである。この名称は，Mkviewobj() で用いる。（以下，Mkobj**() 関数では同様）

【例】：サドル面

```
fd=[ "z=x^2-y^2", "x=[-1,1]","y=[-1,1]"," "];
Sf3data("1",fd);
Windispg();
Mkobjcmd("1",fd,[40,40,"-"]);
Meshlab():=(
Mkviewobj("saddle",oc1, ["m","v"]);
);
```

このうち，`Sf3data("1",fd);` は Cinderella の画面に表示するためであって，なくてもよい。

次図で，左が option + の場合，右が - の場合である。

⇒ 関数一覧

関数 Mkobjcrvcmd(name，PD，option)

機能 空間曲線（直線）の obj ファイルのためのコマンドを作成

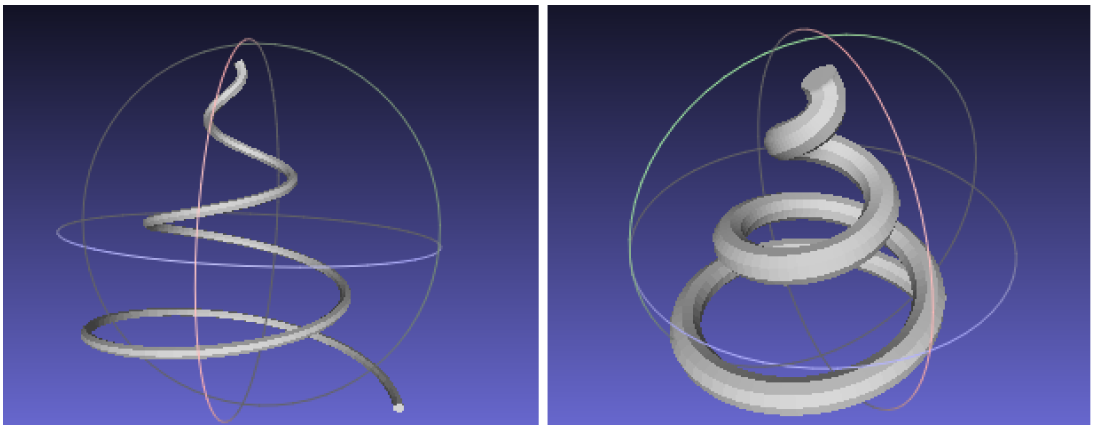
説明 オプションは [太さ，断面の形状（正多角形）の辺の数，断面の正面]

曲線は紐のようなもので表す。その断面は正多角形で， 初期設定は正 6 角形である。断面の正面は”xy”,”yz”,”zx”のいずれかで指定する。太くなった時に形状の差が現れる。

例太さ 0.03 で螺旋を描く

```
Spacecurve("1","[(6*pi-t)/(6*pi)*cos(t),(6*pi-t)/(6*pi)*sin(t),0.1*t]",
"t=[0,6*pi]",["Num=200"]);
Windispg();
Mkobjcrvcmd("1","sc3d1",[0.03]);
Meshlab():=(
Mkviewobj("spiral",oc1,["m","v"]);
);
```

`Mkobjcrvcmd("1","sc3d1",[0.1,8,"yz"]);` としたのが下図右。



⇒ 関数一覧

関数　Mkobjnrm(name，式)

機能　法線ベクトルのデータを作成

説明　式は曲面を表す式。これに対し，法線ベクトルを表す式を求める。

関数　Mkobjplatecmd(name，面データ，options)

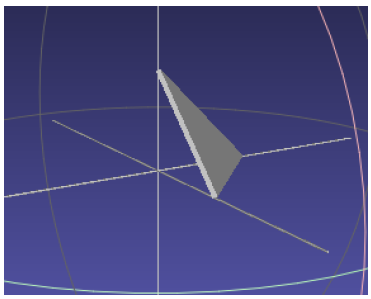
機能　面を描く

説明　面データを渡して面を描く。

options は，面の厚みの指定。厚みは中心線に対し，両側につけることができる。たとえば，[0.05] はプラス側に 0.05 の厚み，[0.05,-0.04] はマイナス側にも 0.04 の厚みをつける。

【例】三角形のプレートを描く

```
Xyzax3data("","x=[-5,5]","y=[-5,5]","z=[-5,5]");
p1=[2,0,0];
p2=[0,2,0];
p3=[0,0,2];
plane=[[p1,p2,p3],[[1,2,3]]];
Mkobjplatecmd("1",plane,[0.05]);
Mkobjcrvcmd("2","ax3d");
Mkviewobj("plane",Concatcmd([oc1,oc2]),["m","v"]);
```



関数　Mkobjpolycmd(name，PD，options)

機能　多面体を描く

説明　VertexEdgeFace() の戻り値を PD として渡して多面体を描く。

関数　Mkobjsymbcmd(PD, 実数，実数，ベクトル, ベクトル)

機能　文字等の obj データのためのコマンドを作成

説明　引数の PD を描く。第 2 引数は大きさ，第 3 引数は回転角，第 4 引数は正面方向のベクトル，第 5 引数は PD の中心の位置。

PD は，平面の描画コマンドによるプロットデータが使える。また，PD に半角アルファベットを文字として与えることができる。この場合，文字は n,p,q,r,t,x,y,z で，該当するフォントが data フォルダの fontF フォルダに用意されている。この中にないフォントは使えない。

関数　Mkobjthickcmd(name，式)

機能　厚みを持つ曲面の obj ファイルのためのコマンドを作成

説明　オプションは [分割数 1，分割数 2，厚み，表側の方向の指定，条件] 表側の方向は，変数に対して，右手系の方向が”+”。厚みを持つため，nsew のそれぞれについて，”+n+s-e-w” のように指定する。

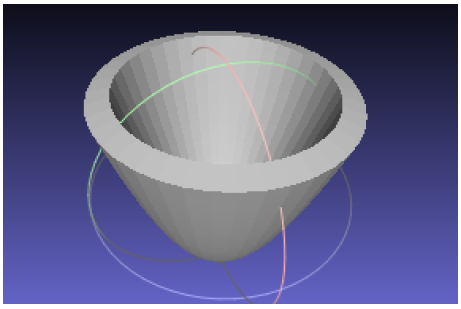
条件として，`"Assume(R>0)"` をつけると，R が 0 以下になるための不具合を回避できる。

また，”ratsimp” をつけると有理関数について，”trigsimp”をつけると三角関数について，処理を速くすることができる。

なお，この関数は Maxima を使うので，Maxima をインストールしていることが前提。

【例】回転放物線

```
fd=[
"z=(x^2+y^2)",
"x=R*cos(T)","y=R*sin(T)",
"R=[0,2]","T=[0,2*pi]","e"
];
Mkobjthickcmd("1",fd,[40,40,0.2,"+n+s-e-w+","assume(R>0)"]);
Mkviewobj("pala",oc1,["m","v","Wait=5"]);
```



関数　Mkviewobj(name，PD, options)

機能　obj ファイルを作成。option により MeshLab を立ち上げて表示する。

説明　第 2 引数に複数のプロットデータを与えるときは，Concatcmd() により 1 つにまとめる。オプションは

”m”または”make”	データを作る（指定しない場合もデータがなければ作る）
”v”または”view”	MeshLab を立ち上げて表示する
”W=n”	作成するための待ち時間。n秒。これを過ぎると終了する
”Unit=mm”	Setunitlen() と連動して 3 Dプリンタの数値の単位を mm で指定する 3D プリンターがインチで認識する場合は ”Unit=in” とする。

⇒ 関数一覧

2.6　表計算ソフトとの連携

表計算ソフトでは，複数のセルを選択してコピー（Windows では Crtl+ C ，Mac では Command+C）すると，セルの内容は tab 区切りのテキストデータとしてクリップボードにコピーされる。これを Cindyscript エディタにペーストすることで表計算ソフトのデータを KETCindy で利用できる。逆に，Cindyscript のコンソールへの出力を表計算ソフトのシートにコピーすることもできる。

また，表計算ソフトから書き出した CSV ファイルについても同様にして CSV 形式のデータを扱うことができる。

関数　Tab2list(str，option)

機能　str の内容をリストに変換する

説明　tab やコンマ区切りになっている文字列 str をリストに変換する。
option は，次の通り。
Blank=a：NULL のセルを a に置き換える。
Sep=b：セパレータ（区切り文字）を b とする。 初期設定は tab コード
次のような手順で表計算ソフトや CSV ファイルからデータを KETCindy に移すことができる。

(1) Cindyscript エディタで，適当な文字変数を用意する。
たとえば，data=””;

```
Fhead="template"; // write a head name
Texparent="";
Ketinit();

data="";

Windispg();
```

(2) 表計算ソフトで，適当な範囲を指定しクリップボードにコピーする。
Windows なら Ctrl+C，Mac なら Command+C

	A	B	C	D	E	F
1		A	T	G	C	
2	トリ結核菌	15.5	14.3	36.4	33.8	
3	大腸菌	24.7	23.6	26	25.7	
4	コムギ	27.4	27.1	22.7	22.8	
5	サケ	29.7	29.1	20.8	20.4	
6	ヒト	30.9	29.4	19.9	19.8	
7						

(3) data=””; のダブルクォートの間にペーストする。
最後の行は右図のように，” の前で改行されていてもよい。

```
data="  A   T   G   C
トリ結核菌  15.5  14.3  36.4  33.8
大腸菌 24.7  23.6  26  25.7
コムギ 27.4  27.1  22.7  22.8
サケ   29.7  29.1  20.8  20.4
ヒト   30.9  29.4  19.9  19.8";
```

```
data="  A   T   G   C
トリ結核菌  15.5  14.3  36.4  33.8
大腸菌 24.7  23.6  26  25.7
コムギ 27.4  27.1  22.7  22.8
サケ   29.7  29.1  20.8  20.4
ヒト   30.9  29.4  19.9  19.8
";
```

(4) この文字変数 data に対し，Tab2list(data) を実行すると，行列を表すリストが返される。
これを適当な変数に代入し，作表コマンドで表にするなど，目的に応じて利用する。
数値だけなら行列として計算もできる。

```
 5 data="   A    T    G    C
 6 トリ結核菌   15.5   14.3   36.4   33.8
 7 大腸菌  24.7   23.6   26   25.7
 8 コムギ  27.4   27.1   22.7   22.8
 9 サケ    29.7   29.1   20.8   20.4
10 ヒト    30.9   29.4   19.9   19.8";
11 dlist=Tab2list(data);
```

```
[[0,A ,T ,G ,C ],[トリ結核菌 ,15.5,14.3,36.4,33.8],[大腸菌 ,24.7,23.6,26,25.7],
[コムギ ,27.4,27.1,22.7,22.8],[サケ ,29.7,29.1,20.8,20.4],[ヒト ,
30.9,29.4,19.9,19]]
```

空文字のセル (NULL) が含まれる場合， 初期設定ではそのまま空文字になるが，アンケート処理などで無回答を 0 にしたいような場合は

```
dlist=Tab2list(data,["Blank=0"];
```

とする。

CSV ファイルから CSV 形式（コンマ区切り）のデータをコピーした場合は

```
dlist=Tab2list(data,["Sep=,"];
```

とする。

なお，文字列をセパレータで区切ってリスト化する Cindyscript の関数に tokenize() がある。上の例で，

```
dlist=tokenize(data,[unicode("000a"),unicode("0009")]);
```

とすると，改行コード (000a) と tab コード (0009) で切り分けてリスト化する。このとき，リストの各要素はつぎのようになる。

文字列→文字列

数値形式の文字→実数【例】 14 → 整数 1412.3 → 実数 12.3

計算式の形→文字列【例】 437-0023 → 437-0023 (文字列)

これに対し，Tab2list() では，計算式の形の文字列は数値と見なして計算結果を取得する。

【例】 437-0023 → 414 (数値)

したがって，郵便番号や日付 (28/12/5) のようなものは計算されてしまうので，tokenize() を用いるのがよい。なお，tokenize() の場合，空行は空リストになるので，最後の行でダブルクォートの前で改行されていると空リストが入る。

関数 Dispmat(list)

機能 リストを行列の形で tab 区切りにしてコンソールに表示する。

説明 行列を表すリスト （たとえば dlist) を引数として Dispmat(dlist) を実行すると，コンソールに行列型で内容が表示される。

実際には TAB 区切りの文字列。(println としなくても直接コンソールに表示される) これを表計算ソフトのシートにコピーする。

```
11 dlist=Tab2list(data);
12 Dispmat(dlist);
```

0	A	T	G	C	
トリ結核菌		15.5	14.3	36.4	33.8
大腸菌	24.7	23.6	26	25.7	
コムギ	27.4	27.1	22.7	22.8	
サケ	29.7	29.1	20.8	20.4	
ヒト	30.9	29.4	19.9	19	

関数 Writecsv(namelist, data, filename, option)

機能 data の内容を CSV ファイルに出力する

説明 ベクトルまたは行列となっている data を、filename のファイル名として CSV ファイルに書き出す。

option は，次の通り。（省略できる）

Col=nn：自然数 nn で指定した列数の CSV ファイルとして書き出す。

namelist は，CSV ファイルの 1 行目に追加される項目名。省略すると”C1,C2,...”という項目名が付く。

なお，列数の指定を省略すると data が行列の場合は、その列数を data がベクトルの場合は namelist の項目数を利用する。

⇒ 関数一覧

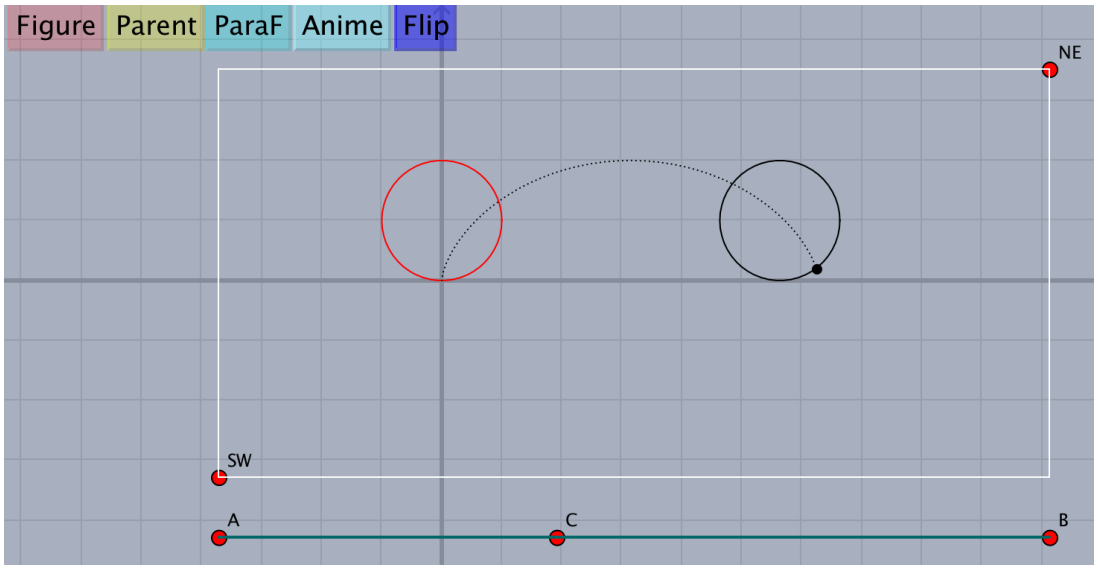
3　アニメーション PDF

3.1　概要

アニメーションのできる PDF を作る。

Cinderella の作図機能と Cindyscript を用いてアニメーションができるが，PDF にすることで Cinderella がなくても PDF ビュアーがあればアニメーションを実行できるので，プレゼンテーションや教材の受け渡しなどに便利である。

次の画面は，samples フォルダにある「s06animation」の「s0601cycloid」のものである。これをひな形として使うのがよい。すでにあるスライダなどが邪魔であれば，消去ボタンで消去するか，template2allbuttons.cdy をひな形として用いる。



画面上方のボタンには，次のようなスクリプトが割り当てられている。

Figure	： Viewtex();	現在の画面の PDF データを作る
Parent	：複数のスクリプト	Figpdf() を使うときに使用する。
ParaF	： Parafolder();	アニメーションのフレームデータを作る
Anime	： Mkanimation();	アニメーション PDF を作る
Flip	： Mkflipanime();	パラパラ動画 PDF を作る

アニメーション PDF を作るには，フレームを定義する関数を記述し，「ParaF」ボタンでフレームデータを作り，「Anime」または「Flip」ボタンで PDF を作成する。

Anime ボタンを押すと，`\usepackage[dvipdfmx]{animate}` でパッケージを読み込み，animateinline 環境のアニメーションを作る。

Flip ボタンを押すと，animateinline 環境ではなく，フレームに分割した PDF が生成される。

なお，アニメーション PDF でアニメーションを行うには Adobe Acrobat Reader など，アニメーションに対応した PDF リーダーが必要である。Windows の SumatraPDF，Mac の プレビューではアニメーションができない。

3.2 関数

関数 Setpara(fname,funcstr,range,options1,options2)

機能 アニメーションの設定をする

説明 fname は出力するファイル名，funcstr は定義した動画関数名，range は範囲
「Anime」ボタンを押すと，animate+fname.pdf が作られる。再生はコントローラか，画面クリックで行う。
「Flip」ボタンを押すと，flipanime+filename.pdf が作られる。再生（コマ送り）は，Acrobat Reader のページ送りボタンで行う。
options1 はアニメーションのデータを作るための設定。

m/r	データの作成 / 既存データがある場合の読み込み（ 初期設定は r ）
Div=n	フレーム数。初期値は 25。

options2 はアニメーションについての設定で，次の通り。

Frate=n	１秒間のフレーム数。初期値は 20。
Title=str	タイトル。指定しない場合は fname と同じ。
Scale=n	図の大きさの拡大率
opA=[option]	animateinline 環境のためのオプション 初期設定は `[loop,controls,buttonsize=3mm]`。 loop：繰り返し再生する。 controls：コントローラを表示する buttonsize：コントローラのサイズ palindrome：反転して繰り返し再生する step：コマ送りモードにする。コントローラもコマ送り仕様。 `"OpA=[controls,buttonsize=5mm]"` のように記述する。 + をつけると初期設定のものに追加することができる。 たとえば `"OpA=+step"` で `"OpA=[loop,controls,buttonsize=3mm,step]"` となる。 `"OpA=[]"` とすると，オプションなしとなり， 画面をクリックすると繰り返しなしで再生される。

記述例

```
Setpara("cycloid","mf(t)","t=[0,60]",["Div=60"],
   ["Frate=30","Title="サイクロイド","Scale=1.5","OpA=[controls]"]);
```

関数 Parafolder(funcstr,fname,range,options)

機能 アニメーションのフレームデータを作成する

説明 funcstr は動画関数名，fname は出力するフォルダ名，range は範囲
作業フォルダ (fig) 内に，フレームデータを格納した fname フォルダを作る。ひな形（s0601cycloid）にある ParaF ボタンに割り当てられており，通常はそのまま使えばよい。

関数 Mkanimation(path,folder)

機能 アニメーションの PDF を作る

説明 作業フォルダ (fig) 内に，フレームデータを格納した fname フォルダを作り，ここからアニメーションの PDF を作る。Setpara() で設定したファイル名を fname とすると，生成する TeX ファイルは，animatefname.tex （PDF 作成の TeX ファイル）と animfname.tex （動画データ）で，PDF の名称は，animatefname.pdf となる。
ひな形（s0601cycloid）にある Anime ボタンに割り当てられており，通常はあらためて設定せずそのまま使えばよい。

関数 Mkflipanime(path,folder)

機能 パラパラ動画の PDF を作る

説明 作業フォルダ (fig) 内に，フレームデータを格納した fname フォルダを作り，ここからパラパラ動画の PDF を作る。Setpara() で設定したファイル名を fname とすると，生成する TeX ファイルは，flipanimefname.tex （PDF 作成の TeX ファイル）で，PDF の名称は，flipanimefname.pdf となる。
ひな形にある Flip ボタンに割り当てられており，通常はあらためて設定せずそのまま使えばよい。

⇒ 関数一覧

3.3 制作例

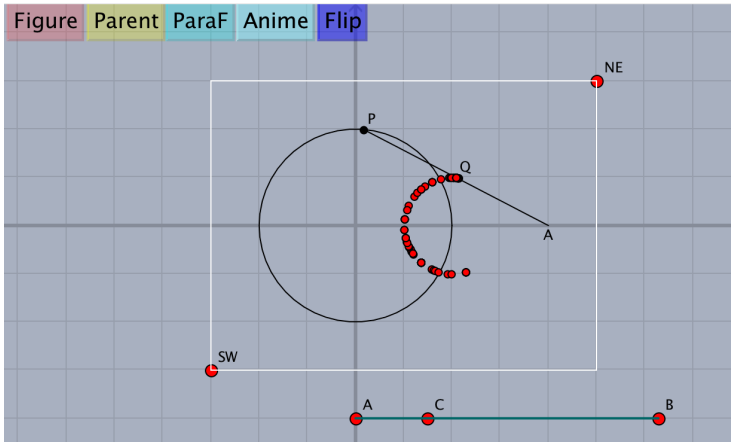
【例】定円上を動く点 P と，定点 A を結ぶ線分の中点を Q として動きを見る。

アニメーション（フレーム）を定義する関数は，時刻を t として， t における図を定義すると考える。時刻は単なる媒介変数であるので，t でなく s などでもよい。関数名は，たとえば mf(movie frame) とする。

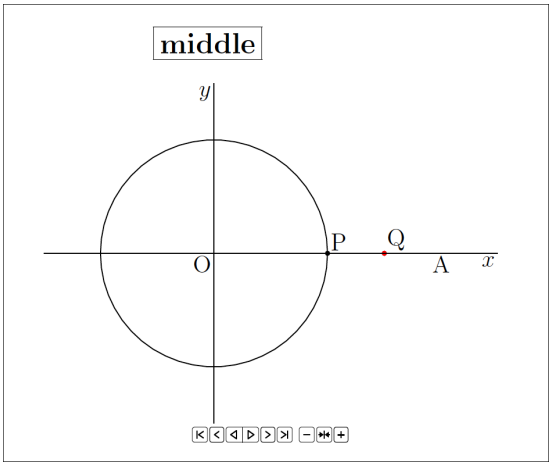
```
Setax(["","","sw","","sw"]);
Slider("A-C-B",[0,YMIN-1],[2*pi,YMIN-1]);
Circledata("1",[[0,0],[0,2]]);
mf(t):=(
  pt=2*[cos(t),sin(t)];
  mp=(pt+[4,0])/2;
  Listplot("1",[[4,0],pt]);
  Pointdata("1",[mp,pt],["Size=2"]);
  if(t==0,
    ptlist=[mp];
    ,
    ptlist=append(ptlist,mp);
  );
  Letter([[4,0],"s","A",pt,"en","P",mp,"ne","Q"]);
  Pointdata("2",ptlist,["Size=2","Color=red"]);
);
mf(C.x);
Setpara("middle","mf(t)","t=[0,4*pi]");
```

この例の場合，mf(C.x) を実行するとスライダを動かすことでインタラクティブに軌跡を表示できる。アニメーションを作る上では mf(C.x) やスライダはなくてもよい。

Cinderella の画面は次のようになる。



アニメーションを作成するときは //mf(C.x) とコメントアウトしてから 「ParaF」「Anime」 ボタンをクリックする。次の図は，でき上がった animatemiddle.pdf の始めの画面である。



また，次のようにオプションを指定すると，5 秒間のアニメーションとなる。

```
Setpara("middle","mf(t)","t=[0,4*pi]",["Div=30"],["Frate=6"]);
```

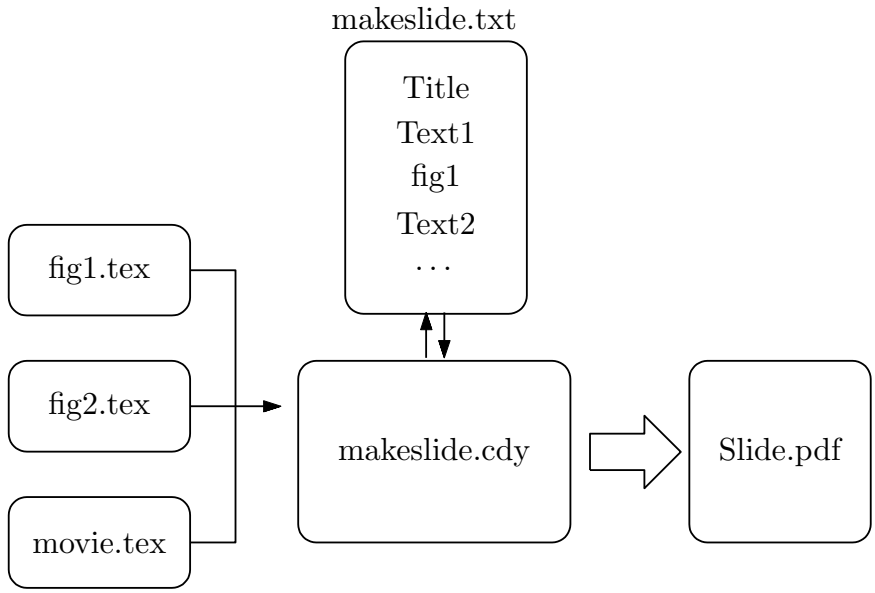
`["Div=150"],["Frate=30"]` とすると，やはり 5 秒間のアニメーションとなるが，1 秒間のフレーム数が多いため，なめらかな動きとなる。これは標準的なビデオのフレームレートである。ただし，ファイルサイズは約 5 倍となる。

⇒ 関数一覧

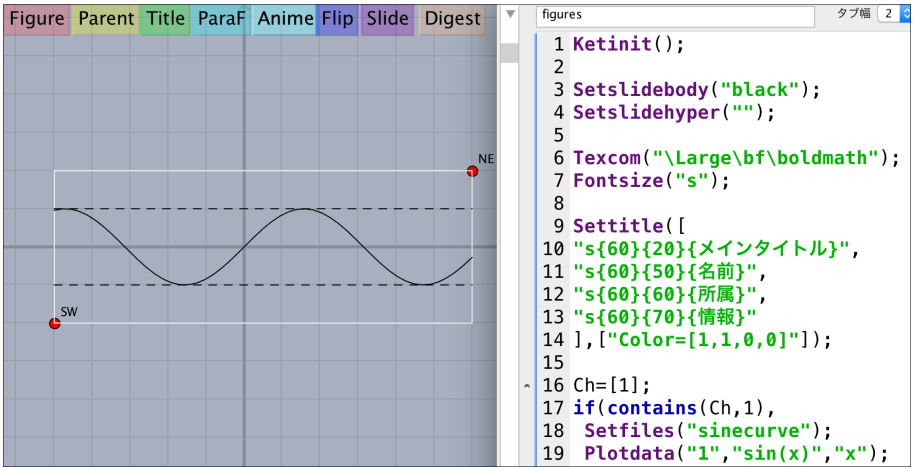
4　KeT スライド

4.1　概要と制作手順

KETCindy で作成した図とテキストを統合してプレゼンテーション用のスライド PDF を作成する。必要なファイルは，スライドの内容を記述したテキストファイル（ここではコンテンツファイルと呼ぶ）と，タイトルや図を作成し，コンテンツファイルと統合する KETCindy のファイルである。この 2 つのファイルは，拡張子が txt と cdy で，ファイル名は同一とする。たとえば，コンテンツファイルを makeslide.txt ，KETCindy ファイルを makeslide.cdy としたときの，制作イメージを次の図に示す。



makeslide.cdy



ひな形として，サンプルとして提供されている samples フォルダの中の s0701basic.cdy

をコピーし，適当にリネームして使うのがよいだろう。必要なボタンと，最低限のスクリプトが記述されている。

以下では，KETCindy のファイルを makeslide.cdy , コンテンツファイルを makeslide.txt として説明を進める。制作手順は次の通り。

(1) makeslide.cdy に `Settitle` コマンドでタイトルを書き，「Titile」ボタンで書き出す。

タイトルスライドが作業フォルダに，makeslide.txt が makeslide.cdy と同じフォルダに作成される。makeslide.txt がすでにある場合には上書きはされず，タイトルスライドだけが上書きされる。

(2) makeslide.txt に，スライドの各ページごとの内容を記述する。

(3) 必要な図やアニメーションのコードを書き，ボタンをクリックしてファイルを作る。

Figure：図を挿入するとき

ParaF：アニメーションやパラパラスライドを挿入するとき

(4) Slide ボタンでスライド PDF を作成する。

PDF と，関連する中間ファイルは，作業フォルダではなく，makeslide.cdy のあるフォルダに作成される。また，スライド PDF はアニメーションと同様，Windows，Mac では Adobe Reader で開く必要がある。Linux(Ubuntu) では Evince でプレゼンテーションができる。

4.2 コンテンツファイル

コンテンツファイルは次のような構成にする。

title::slide0//	タイトルスライド。title はコマンド。
main::三角比と三角関数//	セクション 1 のタイトル。main はコマンド。
直角三角形と三角比//	1 ページ目の表示内容。
・・・・//	
new::角の概念の拡張//	新しいページとタイトル。new はコマンド。
enumerate::[(1)]//	2 ページ目の表示内容。enumerate はコマンド。
・・・・//	
new::負の角//	新しいページとタイトル。
・・・・//	
main::三角関数のグラフ//	セクション 2 のタイトル。
$f(x)=\sin x$//	以下同様
new::振幅と周期//	
・・・・//	

・すべての行の末尾には必ず // をつける。

注）url の指定で // を用いるときは，|||| とすれば // に変換される。

・ページの内容は，コマンドでレイアウトなどを指定し，表示する文をテキストで書く。

コマンド

コマンドにおいて，各ブロックの引数の区切りは :: とし，各行の終わりには必ず//をつける。

【タイトルと壁紙】

タイトルスライドをつけるときは,

```
title::slide0//
```

を１行目に置く。タイトルスライドは makeslide.cdy で作る。slide0 は 初期設定のタイトルスライドのファイル名。このファイル名を変更 (たとえば ”start”) したときは，makeslide.cdy で，Settile() のオプションに ”Title=start” をつけて，ファイル名が一致するようにしておく。

タイトルスライドをつけないときはスライド名をつけないでおく。

```
title::// 
```

注）title コマンドは必須で，これを１行目に書かないとスライドは作成されない。

壁紙（背景）を表示するときは，タイトルコマンドに続けて壁紙ファイル名を書く。

```
title::slide0::wallpaper//
```

wallpaper は壁紙のファイル名。壁紙ファイルは TeX のファイルで，作業フォルダ（fig）に入れておく。

壁紙ファイルの一例

```
 {\color[cmyk]{0.6,0.2,0.8,0}\huge\rm\normalsize
\newpage
\begin{layer}{120}{0}
\lineseg{0}{2}{125}{0}
\lineseg{0}{88}{125}{0}
\putnotese{0}{90}{\ketcindy}
\end{layer}
}
```

【セクションタイトル】

main::**セクションタイトル名**//

セクションを分けないときはなくてもよい。

【新しいページ】

new(::**行下げ**)::**タイトル**((::**位置**)::**読み込みファイル**)//

例）new::[10]::**はじめに**::{50}{20}::figure//

読み込みファイルの表示サイズを変更するときは

```
new::[10]::はじめに::{50}{20}::figure,0.8//
```

のようにする。
読み込みファイルがなければ，figure は省略。

【箇条書き】

番号つき箇条書きは

```
enumerate//
```

で，enumerate 環境の始まりを示す。
番号にかっこをつけるなど，番号の形式を変えるには，

```
enumerate::[(1)]
```

のように，::で区切って形式を示す。 初期設定は，かっこなしの番号。
記号つき箇条書きは

```
itemize//
```

で itemize 環境の始まりを示す。記号は中黒。
enumerate，itemize のいずれも

```
item::文//
```

で item を記述する。
環境の終わりは，

```
end//
```

で示す。

【項目の順表示】

1 枚のスライド内で，項目を段階的に表示するときは，new の次の行に

```
%repeat=m(,para)//
```

を書く。m は段階数で，たとえば，そのスライドの内容を 3 段階で表示したい場合は

```
%repeat=3//
```

とする。実際には 3 枚のスライドが作られる。
para をつけると，右下にコントローラが表示され，前後に進めやすくなる。

```
%repeat=3,para//
```

段階的に表示したい行の先頭に，表示する順番を，2 番目以降から

```
%[2,-]::text//
%[3,-]::text//
```

のように書く。
番号指定を [-，3] とすると，3 番目まで表示する。
[1..3,5] とすると，1 番目から 3 番目までと 5 番目に表示する。(4 番目をスキップ)

【薄文字】

順表示したい項目の全体像を見せておいて，そのうちの現在までの項目を示すような場合に用いる。番号指定の前に thin をつけ，

```
%thin[2,-]::text//
%thin[3,-]::text//
```

のように書くと，現段階の項目よりあとは薄文字で表示される。

薄文字の濃さは，

```
\setthin{alpha}//
```

で指定できる。alpha は 0 から 1 までの数で， 初期設定は 0.1。

`\setthin{0}//` のとき，指定段階以降の項目は非表示になる。

初期設定の薄文字の濃さは，makeslide.cdy の Setslidebody の第 3 引数で設定できる。たとえば，

```
Setslidebody(["","",0.2]);
```

とする。

item とともに用いるときは，`%thin[n,-]` を先に書く。

【例】項目の順表示

図は， 3 番目の Java まで進んだところである。右下にコントローラがある。

```
new::プログラミング言語 //
%repeat=6,para//
\slidepage//
itemize//
item::Python//
%thin[2,-]::item::Ruby//
%thin[3,-]::item::Java//
%thin[4,-]::item::JavaScript//
%thin[5,-]::item::CindyScript//
%thin[6,-]::item::C//
end//
```

【図ファイルの順表示】

作業フォルダ（fig）の中にあるサブフォルダ（例えば subfig）の図ファイルすべてを順に表示する。

パラパラ動画のときに用いる。パラパラ動画では，ParaF ボタンをクリックすると，作業フォルダ (fig) の中に，動画のフレームファイルが入ったフォルダが作られる。そのフォルダのファイルをアルファベット順に表示する。もちろん，パラパラ動画以外のファイルの順表示に使ってもよい。

new の次に

%repeat=//

%para=subfig:{0}:s{60}{10}:input(:**倍率**)//

を書く。TeX のソースには,

layer{120}{0}, \putnotes{60}{10 **初期設定**put...}

の形で書き入れられる。

また，２行に分けずに

%repeat=,para=... と続けて書いてもよい。

【レイヤー】

ketlayer の\begin{layer}・・\end{layer} として書き出す。本来の記述より簡素になっている。

layer::{**範囲**}{0}//

で layer 環境の始まりを示す。layer 環境の終わりは

end//

レイヤーの中に作業フォルダ（fig）にある図を表示するには

putnote::**方向と位置**::**読み込みファイル**//

とする。

例） figure.tex を (30,10) の位置の南東（原点は左上）に表示する。

putnote::se{30}{10}::figure//

例） fig の中の figure.tex を 0.8 倍にして表示する。

putnote::se{30}{10}::figure,0.8//

例）includegraphics で figure.pdf を表示する。

putnote::se{30}{10}::include[width=5cm]::figure.pdf//

注）KeTpicStyle.pdf を参照。文字などは本来の書式を用いて次のように入れる。

\putnotee{30}{10}{**文字**}//

【テキストと動画】

コマンド以外のテキストはそのまま TEX に書き出される。行末には // をつける。

テキストを途中で改行するには，TEX の 強制改行マーク \\ をつける。

動画の場合は，動画の設定をして（アニメーションの節を参照）ファイルを作る。

パラパラ動画の場合は，ParaF ボタンをクリックすると，fig フォルダ内に動画のフレームファイルが入ったフォルダ（たとえば sincurve）が作られる。これを,

%repeat=//

%para=sincurve:{0}:s{60}{10}:input(:**倍率**)//

で表示する。

アニメーションの場合は，スライドのために，makeslide.cdy のスクリプトに，Addpackage(["[dvipdfmx]{animate}"]); を追加しておく。

Anime ボタンをクリックすると，パラパラアニメと同様に動画のフレームファイルが入ったフォルダが作られ，動画用の TEX ファイルができる。Setpara() で設定したファイル名が ”sincurve” のとき，動画用の TEX ファイルは ”animsincurve.tex” となるので，%repeat ではなく \input{fig/animsincurve}// で表示する。

【コメント行】

コマンドや文をコメントアウトするときは，%% とする。

【空白行】

空白行を入れたいときは，...// とする。

【タブ】

\Ltab{**長さ**}{**文１**}**文２**

とすると，文１が行頭から，文２が長さ分の字下げをした位置から表示される。長さは 20mm のように指定。

【ページ番号】

page 番号を表示するときは次のようにする。

main の場合は \slidepage[m]//

それ以外の場合は %repeat**の後に** \slidepage//

注）総ページ数を取得するため，TeX を２度コンパイルすることが必要である。そのため，Slide ボタンを２度クリックすると総ページ数が表示される。

mp3/mp4 ファイルの追加

makeslide.txt のタイトルコマンドの後に，以下を追加する。

```
title::slide0(::wallpaper)
::\usepackage{ketmedia}
::\usepackage[dvipdfmx]{media9}//
```

mp3 ファイルを追加するときは，\inputsound または\inputsoundclick を用いる。

\inputsountclik[90]{**フォルダ**/}{**ファイル**}

\inputsound は「自動再生」，\inputsoundclick は「クリック再生」である。最初の引数は，ボタンの水平位置（単位 mm）でデフォルトは 90 である．またフォルダには/をつける。

mp4 ファイルを追加するときは，\inputmovie を用いる。

\inputsountclik[90]{1}{0.4}{**フォルダ**/}{**ファイル**}

2 番目と 3 番目の引数は，幅と高さの\linewidth からの倍率である。

【余白などの編集】

スライドの上側余白として，初期状態で `\vspace*{18mm}` が設定される。次のページにはみ出してしまうような場合は，出力された tex ファイルで調整すればよい。

その他，適宜編集して，kc.command(bat/sh) を実行すれば，細かい部分を修正した PDF を作成することができる。

4.3 関数

makeslide.cdy で使う，KETCindy の関数。

関数 Setslidebody(bodycolor,bodystyle,density)
機能 全体の文字スタイルと薄文字の濃さ (0-1) の設定
説明 引数の意味は次の通り。

bodycolor	文字色 :初期設定は ”blue”
bodystyle	フォントタイプ :初期設定は `"\Large\bf\boldmath"`
density	薄文字の濃さ :初期設定は 0.1

引数がない場合や（ `Setslidebody()`）この関数を書かない場合は初期値が使われる。
ある引数だけを指定したい場合は，それより前は空にする。
【例】`Setslidebody(,"\large")`

関数 Setslidehyper(”dvipdfmx”,options)
機能 順表示でページ送りのコントローラを設定する。
説明 順表示しないときもこの関数の記述は必須と考えてよい。
パッケージ hyperref.sty を読み込み，かつ，options の値を与える。
`Setslidehyper();` は無効。初期設定で使う場合は `Setslidehyper("");` とする。
第 1 引数は hyperref.sty の第 1 パラメータ。初期設定は ”dvipdfmx”
options は [”cl=bool,lc=col,fc=col”,”Pos=[x,y]”,”Size=n”]

cl	colorlinks : リンクに色をつけるかどうか。true / false で指定。 初期設定は true
lc	linkcolor : コントローラの色。色名で指定。 初期設定は blue
fc	filecolor リンクの色。色名で指定。 初期設定は blue

以上 3 つは，セットで指定。一部初期値を使う場合は `lc=,` のように右辺を空にする。
Pos コントローラの位置。 初期設定は [125,73] （左上が原点）
Size コントローラの大きさ。 初期設定は 1

関数 Setslidemain([letterc,boxc,boxd,framec,xpos,size])
機能 メインスライド（セクション区切り）の設定

説明 引数の一部を 初期設定値とする場合は [,,,,3] や [,"red"] などとする。

letterc 文字の色。 初期設定は CMYK で [0.98,0.13,0,0.43]
boxc ボックスの色。 初期設定は [0,0.32,0.52,0]
frame フレームの色。 初期設定は [0,0.32,0.52,0]
xpos タイトルの水平位置。 初期設定は 62
size タイトルの倍率。 初期設定は 2

関数 Setslidepage([letterc,boxc,boxd,framec,shadowc,xpos,size])
機能 ページの設定をする。
説明 引数の一部を 初期設定値とする場合は [,,,,3] や [,"red"] などとする。

letterc 文字の色。 初期設定は CMYK で [0.98,0.13,0,0.43]
boxc ボックスの色。 初期設定は [0,0.32,0.52,0]
frame フレームの色。 初期設定は [0,0.32,0.52,0]
shadowc影の色。 初期設定は [0,0,0,0.5]
xpos タイトルの水平位置。 初期設定は 6
size タイトルの倍率。 初期設定は 1.3

関数 Setslidemargin([leftmarginchange,topmarginchange])
機能 スライドの左上マージンをデフォルトから変更する場合の値。
説明 Setslidemargin([+5,-10]);（横方向に +5mm，縦方向に −10mm 変更）

関数 Settitle(タイトルリスト,options)
機能 タイトルスライドを作る
説明 タイトルリストはコンマ区切りで位置と文を文字列で与える。例を参照。「Title」ボタンをクリックすると，コンテンツファイルがない場合は新たに作る。すでにある場合は上書きせずタイトルスライドのみを作る。
Settitle のオプションは次をリストで与える。

"Title=" タイトルスライドのファイル名。 初期設定は "slide0"。
"Layery=" タイトルの縦位置。 初期設定は 0。0 でないときは方眼を表示。
"Color=" 文字の色。 初期設定は blue。

【例】タイトルの設定

```
Settitle([
```

```
"s{60}{20}{メインタイトル}",
"s{60}{50}{名前}",
"s{60}{60}{所属}",
"s{60}{70}{情報}"
] ,
["Title=SlideA","Color=[1,1,0,0]"]);
```

出力する TEX ファイルに，\begin{layer} の設定をする。 s は，東西南北の s。
例のように，タイトルリストは改行すると見やすい。

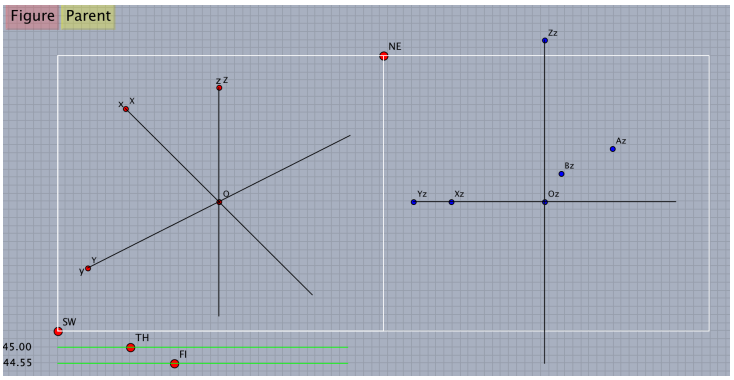
⇒ 関数一覧

5 KETCindy3D

5.1 概要

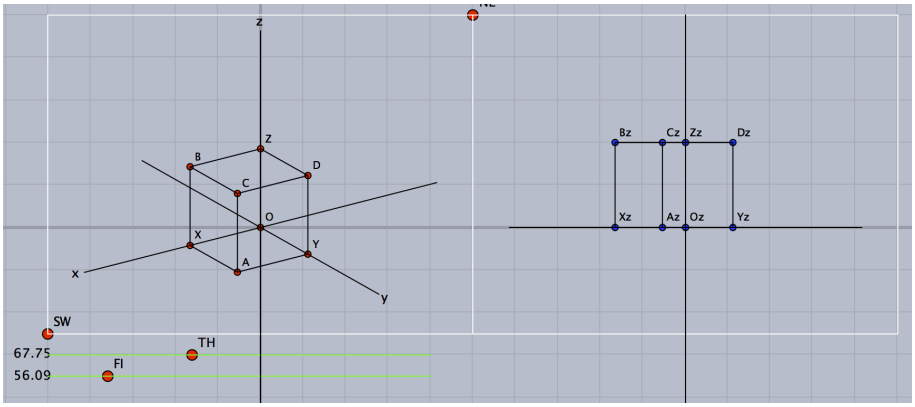
KETCindy3D の画面は次のように構成される。

Cinderella の描画面に，白の矩形で囲んだ領域が 2 つできる。NE,SW を対角とする左側の領域を主画面，右側の領域を副画面という。



主画面は平面の場合と同様，TeX に出力される範囲を示し，NE,SW の 2 点をドラッグすることにより変更できる。主画面の下方のスライダで視点が移動でき，主画面上では軸が回転する。副画面は，xy 平面上に視点を置いたものと考えればよい。

主画面上に Cinderella の作図ツールで点や線分を作図すると，副画面に対応する点が作図される。主画面上の点をドラッグすると x,y 座標を変更でき，副画面上の点をドラッグすると z 座標を変更できる。



KeTCindy3D では，線や面についての陰線処理を行う。陰線処理はC言語との連携により処理を速めている。C 言語を使う環境整備が必要であるが，現在はこれを標準としている。C 言語が使えない場合は R で計算する関数を用いることになるが，その場合はかなり時間がかかる。（場合にもよるが 2 分程度）

5.2 設定

関数 Ketinit3d()

機能 KeTCindy3D の使用宣言

説明 Cinderella の画面を 3 Dモードにする。

Cinderella の描画面に，視点移動のための 2 つのスライダを作る。スライダは初期位置が左端になる。スライダ TH で角 THETA を，スライダ FI で角 PHI を内部変数として定義する。

引数に 0 を入れて `Ketinit3d(0)` とすると，副画面を表示しない。

＜重要＞

この関数は Initialization スロットに置く。Ketinit() も，平面の場合と異なり Initialization スロットに置く。KeTCindy3D における変数の初期化などを行う，Start3d() は Draw スロットに書く。

関数 Setangle(TH,FI)

機能 回転角の指定

説明 スライダで設定できる回転角（視点の位置）TH と FI を度数法で指定する。たとえば，`Setangle(70,40)` とすると，TH,FI がその位置になる。スライダは固定されるので，再度スライダを有効にしたい場合は，コメント化して再実行する。

初期状態だけを決めたい場合は

```
if(!Ptselected(),Setangle(70,40));
```

または

```
if(!Isangle(),Setangle(70,40));
```

とすると，スライダは有効となり，スライダのいずれかの点を選択した状態であれば Figure ボタンも有効である。画面上のなにもないところをクリックして，点の選択状態を解除するともとに戻る。

回転角の取得については，回転角の取得を参照のこと。

関数 Start3d(option)

機能 3 Dの画面設定と空間点の認識

説明 副画面を作り，幾何点を 3 Dの点として認識する。この関数は必須で，Draw スロットの先頭に書く。

Cinderella の作図ツールで，点・線分を作図すると，内部関数の Ptseg3data() によってそれらを空間の点として認識し，副画面上に対応する点をとる。ただし，始めは z 座標を 0 とする。点の名前が A であれば，副画面上の点は Az となる。点をポイントして選択すると副画面の上に座標が表示される。

作図した点の名称をインスペクタで変更した場合，新しい名称に対応する点を副画面上に作成するが，以前の点は消えないので要注意。たとえば，点 A を作図した後，主画面上の点 A をインスペクタで点 D に変えた場合，副画面上に新たに Dz ができるが，以前の Az も残る。残った Az は，選択しておいて作図ツールの消去ボタン で消すことができる。

option に，除外点のリストを与えると，その点は空間点としない。（スライダで視点を移動しても位置は変わらない）

関数 Startsurf(options)

機能 曲面描画の初期化と定数の設定

説明 options で定数を設定する。定数としては，分割数，C のサイズ，誤差の限界を設定する。

options がないときは，以下の 初期設定を用いる。

[50,50],[1500,500,200],[0.01,0.1]

設定後に初期値にリセットするときは，文字列 ”reset” を引数に与える。

これにより，陰線処理をともなう面の描画の手順は，次のようになる。

(1)Startsurf(); で面描画の宣言をする。

(2) 描画関数でプロットデータを作る。

(3) ExeccmdC(); で，C 言語を用いてまとめて描画する。

⇒ 関数一覧

関数 Xyzax3data(name， x の範囲，y の範囲，z の範囲,options)

機能 座標軸を描く

説明 描画面に座標軸を描き，プロットデータ ax3d を作成する。name は空文字列でよい。

option は次の 2 つ。

矢じり：”an”：n は数字で矢じりの大きさ。n はなくてもよい。

原点 O：”Onesw”：nesw は微小位置。数字も付けられる。nesw をつけない場合の初期値は sw。

【例】 初期設定の座標軸

```
Xyzax3data("","x=[-5,5]","y=[-5,5]","z=[-5,5]");
```

矢じりをつける

```
Xyzax3data("","x=[-5,5]","y=[-5,5]","z=[-5,5]","a");
```

矢じりを大きくする

```
Xyzax3data("","x=[-5,5]","y=[-5,5]","z=[-5,5]",["a2"]);
```

原点の O を表示する。

```
Xyzax3data("","x=[-5,5]","y=[-5,5]","z=[-5,5]",["O"]);
```

原点の O の位置を調整して右上に表示する。やじりもつける。

```
Xyzax3data("","x=[-5,5]","y=[-5,5]","z=[-5,5]",["a","Oe2n2"]);
```

【注意】Putaxes3d() で点を取ると原点に点 O が作成される。この点名 O と表示が重複するのが煩わしい場合は，作図後にこの option をつけてから出力するとよい。

5.3 描画

5.3.1 点・線

関数 Pointdata3d(名前, 点リスト,options)

機能 点の 3D データと 2D データを作成し，画面と TEX に出力する。

説明 options は Pointdata() と同様。

【例】

```
Pointdata3d("1",[[1,1,1],[0,1,0]],["Size=2","Color=red"]);
```

関数 Putpoint3d(リスト,option)

機能 空間に幾何点を作図する

説明 点の名称と座標を与えて点を作図する。複数の点を一度に作図できる。
option は，”fix”(初期設定) または ”free”。リスト [”free”] にしてもよい。
”fix” では，固定点（ドラッグで移動できない点）とする。同じ名称の点がすでに存在する場合は，指定した位置に移動して固定点とする。
”free” では，自由点（ドラッグで移動できる）とする。同じ名称の点がすでに存在する場合はなにもしない。

【例】いくつか記述例を示す。

```
Putpoint3d(["A",[2,1,3]]);
Putpoint3d(["A",[1,1,1],"C",[1,0,1]],"fix");
Putpoint3d(["A",[2,1,3]],"free");
```

なお，この関数は幾何点を作るものであり，TeX には出力されない。TeX に点を出力するには，Pointdata() を併用する。
空間における点の座標は，点名に”3d”を付加した名前の変数に代入される。たとえば，点 A の座標は A3d である。これにより，点の座標を取得できる。

関数 Putaxes3d([x,y,z])

機能 軸上に幾何点を作る。

説明 引数のリスト [x,y,z] に対し，点 X(x,0,0) ,Y(0,y,0) , Z(0,0,z) および 原点 O を主画面上にとり，副画面上に対応する点 Xz，Yz，Zz，Oz を作る。すでに同じ名称の点がある場合は，指定された位置に移動する。
引数は，実数にすることもでき，Putaxes3d(a) は，Putaxes3d([a,a,a]) と同じになる。

【例】
Putaxes3d(5); 原点と，$x(5,0,0), y(0,5,0), z(0,0,5)$ を作る。
Putaxes3d([1,2,3]); 原点と，$x(1,0,0), y(0,2,0), z(0,0,3)$ を作る。

関数 Putoncurve3d(点名，PD)
機能 空間曲線上に点をとる
説明 プロットデータ PD の曲線上に，点名の点をとる。
とった点は固定点ではなく，曲線上にインシデントとなる。したがって，ドラッグして曲線上を動かすことができる。例は Partcrv3d() を参照のこと。

関数 Putonseg3d(点名，点 1，点 2)
機能 線分上に点を作る
説明 点 1 と点 2 の中点に，指定された名前の点を取る。点 1 と点 2 が線分として結ばれていなくてもよい。とった点は線分にインシデントとなる（線分が描かれていなくても）。点 1 と点 2 はリストにすることもできる。指定した点がすでに存在する場合は動かさない。
点 1，点 2 は幾何点の名称または座標で指定する。

【例】A(1,-1,0) と B(0,2,2) の中点に点 C をとる。つぎのいずれでもよい。
```
Putonseg3d("C",A,B);
Putonseg3d("C",[A,B]);
Putonseg3d("C",[[1,-1,0],[0,2,2]]);
```

関数 Spaceline(name,list)
機能 折れ線を描く
説明 点の名称または座標のリストを与えて折れ線を描く。平面での Listplot() にあたる。
options は線種（dr,da,do）

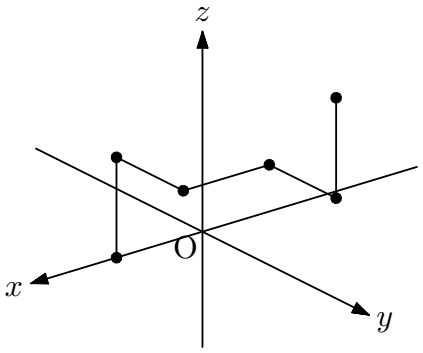
【例】いくつか示す。
・Spaceline(”1”,[[2,5,1],[4,2,3]]); 指定された 2 点を結んだ線分を描く。
・Spaceline(”2”,[A,B,C,A]); 作図されている 2 点 A,B,C を結んだ三角形を描く。
・節点を表示する場合は，Pointdata3d() で描画する。

```
pt=[[2,0,0],[2,0,2],[2,2,2],[0,2,2],[0,4,2],[0,4,4]];
Spaceline("1",pt);
Pointdata3d("1",pt,["Size=3"]);
```

点の名前が必要であれば

```
pname=apply(1..6,"P"+text(#));
```

のようにして，名前リストを作ることができる。



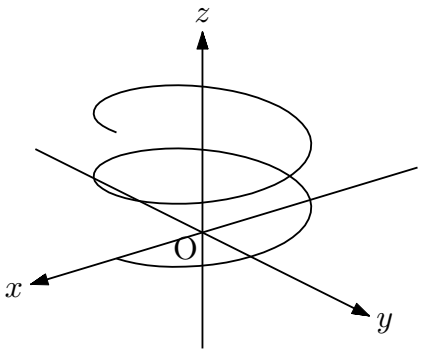
関数 Spacecurve(name, 式, 定義域,options)

機能 空間曲線を描く

説明 媒介変数で表された曲線を描く。option は解像度 Num

【例】螺旋を描く

```
Spacecurve("1","[2*cos(t),2*sin(t),0.2*t]","t=[0,4*pi]",["Num=100"]);
```



関数 Bezier3d(name, リスト 1, リスト 2)

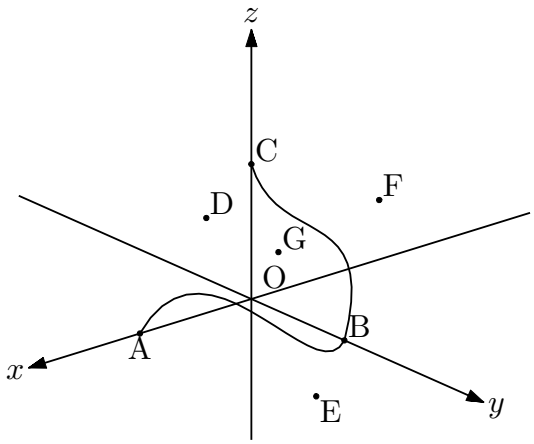
機能 空間ベジェ曲線を描く

説明 引数はリスト 1 が端点リスト，リスト 2 が制御点リスト
１組の端点につき，2 つの制御点を使う。

【例】いくつかの点をベジェ曲線で結ぶ
端点 A,B に対し，制御点を D,E とする。

```
Bezier3d("1",["A","B"],["D","E"]);
```

端点 A,B に対し，制御点を D,E とし，端点 BC に対し制御点を E,F とする。
`Bezier3d("1",["A","B","C"],["D","E","E","F"]);`
端点 A,B に対し，制御点を D,E とし，端点 BC に対し制御点を F,G とする。（図）
`Bezier3d("1",["A","B","C"],["D","E","F","G"]);`



⇒ 関数一覧

関数 Mkbezierptcrv3d(点リスト)
機能 制御点を自動的にとる空間ベジェ曲線
説明 リストで与えた点に対し，制御点を自動的に生成してベジェ曲線を描く。
制御点は，2 つの点に対して，その点を端点とする線分上に 2 つ作られる。これを適宜移動して任意の曲線にすることができる。空間ベジェ曲線 Bezier3d() を参照のこと。

【例】`Mkbezierptcrv3d(["A","B","C","D"]);`
線分 AB 上に 2 点 a1p,a2p，線分 BC 上に 2 点 a2p,a2q，線分 CD 上に 2 点 a3p,a3q ができる。

関数 Skeletonparadata(name,PD リスト,PD リスト,option)
機能 陰線処理（スケルトン処理）をおこなう
説明 描画されている線と軸について陰線処理をおこなう。
第 2 引数の線（プロットデータ）が，第 3 引数の線（プロットデータ）によって隠される部分を消去する。第 2，第 3 引数を省略した場合は，すべての線について，互いの陰線処理をおこなう。option で消去する部分の長さを指定できる。
他のオプション
`"No=`**点リスト**`"` 点リストの点が選ばれているときは実行しない
`"File=y/m/n (n)"` データファイルを作るか
`"Check=`**点リスト**`"` 点リストの点が変更されていたら，ファイルを作り直す

【例】螺旋と線分，座標軸の陰線処理

次のように螺旋と線分，座標軸を描いておく。

```
Xyzax3data("","x=[-5,5]","y=[-5,4]","z=[-5,3]");
Putpoint3d(["A",[0,-2,-2]]);
Putpoint3d(["B",[-1,1,3]]);
Spaceline([A,B]);
Spacecurve("1","[2*cos(t),2*sin(t),0.2*t]","t=[0,4*pi]",["Num=100"]);
```

座標軸のプロットデータは ax3d，線分は AB3d，螺旋は sc3d1 である。これに対し，

```
Skeletonparadata("1");
```

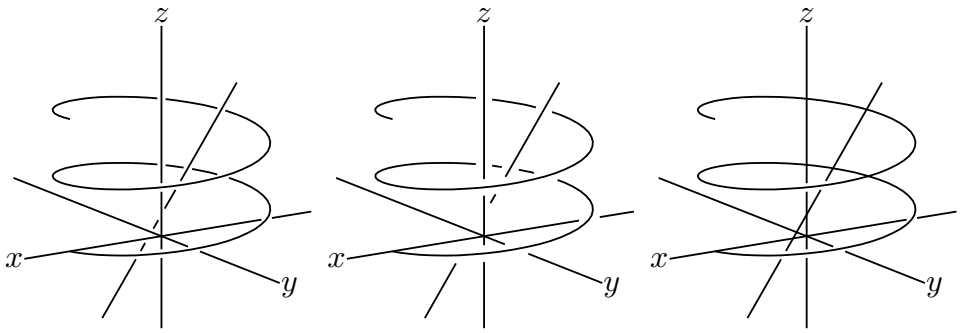
描画されている線と軸について陰線処理をおこなう。（図左）

```
Skeletonparadata("1",[2]);
```

重なった部分の空きを 2 にする。（図中央）

```
Skeletonparadata("1",["AB3d","ax3d"],["sc3d1"]);
```

螺旋によって隠れる部分だけ消去する。（図右）



このほか，次も可能。

```
Skeletonparadata("1",["AB3d","ax3d"],["sc3d1"],[2]);
Skeletonparadata("1",["AB3d"],["ax3d","sc3d1"]);
```

⇒ 関数一覧

5.3.2 多面体

多面体の描画について，四面体の場合を例にして説明する。

四面体は 4 つの面からなっている。

頂点を A,B,C,D とすると，4 つの面は

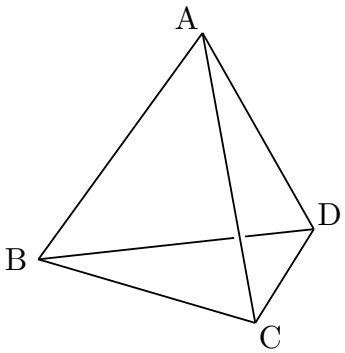
△ ABC，△ ABD，△ ACD，△ BCD

である。

頂点のリスト [A,B,C,D] に対し，A から順に番号をつけると，各面の頂点の順番は

[1,2,3],[1,2,4],[1,3,4],[2,3,4]

と表現できる。



頂点のリストと，この面リストを組にして　[[A,B,C,D],[[1,2,3],[1,2,4],[1,3,4],[2,3,4]]]　としたものを「面データ」という。この面データを使って，多面体を描画するのが VertexEdgeFace() である。

多面体の陰線処理は 2 通りある。ひとつは，多面体を線画と考えて，隠れる部分だけを処理する方法で，Skeletonparadata() を用いる。Concatobj() の例を参照のこと。

もうひとつは，面と考えて，面に隠れる部分を点線で描いたり，非表示にしたりする方法で，Phparadata() を用いる。 Phparadata() の例を参照のこと。

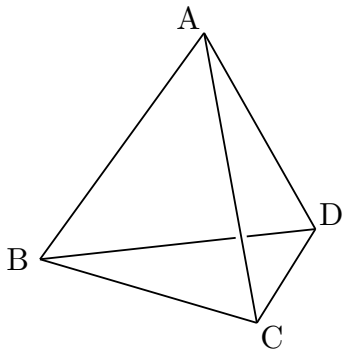
関数　Concatobj(リスト,option)

機能　いくつかの obj データを結合する

説明　多面体の各面の頂点リストから面データ（頂点リストと面リスト）を作る。
たとえば，`Concatobj([[A,B,C],[A,B,D],[A,C,D],[B,C,D]]);` とすると，面データ [[A,B,C,D],[[1,2,3],[1,2,4],[1,3,4],[2,3,4]]] が返される。

【例】4 点 A,B,C,D を幾何点として作り，これを頂点とする四面体を描く。

```
Putpoint3d("A",2*[0,0,sqrt(3)]);
Putpoint3d("B",2*[1,-1/sqrt(3),0]);
Putpoint3d("C",2*[0,sqrt(3)-1/sqrt(3),0]);
Putpoint3d("D",2*[-1,-1/sqrt(3),0]);
phd=Concatobj([[A,B,C],[A,B,D],[A,C,D],[B,C,D]]);
VertexEdgeFace("1",phd);
Skeletonparadata("1");
Letter3d([A3d,"ne","A",B3d,"sw","B",C3d,"se","C",D3d,"e","D"]);
```



＜参考 1＞

幾何点を作らないで四面体を描くのであれば,

```
a=2*[-1,-1/sqrt(3),0];
b=2*[1,-1/sqrt(3),0];
c=2*[0,sqrt(3)-1/sqrt(3),0];
d=2*[0,0,sqrt(3)];
phd=Concatobj([[a,b,c],[a,b,d],[a,c,d],[b,c,d]]);
```

としてもよい。

＜参考２＞四面体のような凸型多角形の場合は，CindyScript の convexhull3d() 関数を用いて次のようにすることができる。面リストではなく頂点リストを与えるだけなので手間を省くことができる。

```
a=2*[0,0,sqrt(3)];
b=2*[1,-1/sqrt(3),0];
c=2*[0,sqrt(3)-1/sqrt(3),0];
d=2*[-1,-1/sqrt(3),0];
phd=convexhull3d([a,b,c,d]);
```

⇒ 関数一覧

関数 VertexEdgeFace(name, 面データ,options)

機能 面データを用いて多面体を描く

説明 面データは, たとえば四面体 ABCD の場合は, [[A,B,C,D],[[1,2,3],[1,2,4],[1,3,4],[2,3,4]]] である。

4 点 A,B,C,D をとっておき，このリストを引数に与えると，四面体が描かれる。

生成されるプロットデータは,

phv3d：頂点のリスト

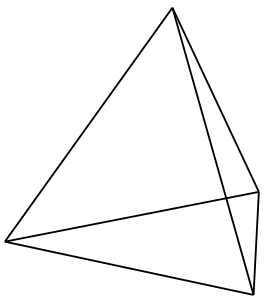
phe3d：辺のリスト

phf3d：面リスト

なお，それぞれ末尾に name が付加される。

【例】4 点 A,B,C,D を取り，正四面体 ABCD を描く

```
Putpoint3d("A",2*[-1,-1/sqrt(3),0]);
Putpoint3d("B",2*[1,-1/sqrt(3),0]);
Putpoint3d("C",2*[0,sqrt(3)-1/sqrt(3),0]);
Putpoint3d("D",2*[0,0,sqrt(3)]);
phd=[[A,B,C,D],[[1,2,3],[1,2,4],[1,3,4],[2,3,4]]];
VertexEdgeFace("1",phd);
```



面リストは，Congatobj() を使って作ることができる。陰線処理して描く場合も含め，Concatobj() の例を参照のこと。

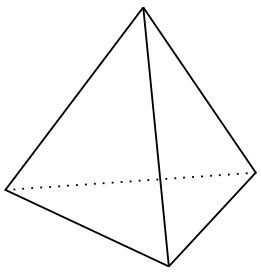
関数 Phparadata(name,name2,options)

機能 多面体を陰線処理して描く

説明 多面体のプロットデータを VertexEdgeFace() で作る。このプロットデータに対し，隠れている面（辺）を陰線処理して表示する。第 1 引数は通常の name，第 2 引数の name2 は，VertexEdgeFace() で与えた name と同じものとする。
options は，全体の線種（”dr,2”など）と，陰線の線種を”Hidden=線種” で指定できる。 初期設定では陰線は表示しない。

【例】四面体を描く。

```
Putpoint3d("A",2*[-1,-1/sqrt(3),0]);
Putpoint3d("B",2*[1,-1/sqrt(3),0]);
Putpoint3d("C",2*[0,sqrt(3)-1/sqrt(3),0]);
Putpoint3d("D",2*[0,0,sqrt(3)]);
phd=Concatobj([[A,B,C],[A,B,D],[A,C,D],[B,C,D]]);
VertexEdgeFace("1",phd);
Phparadata("1","1",["Hidden=do"]);
```

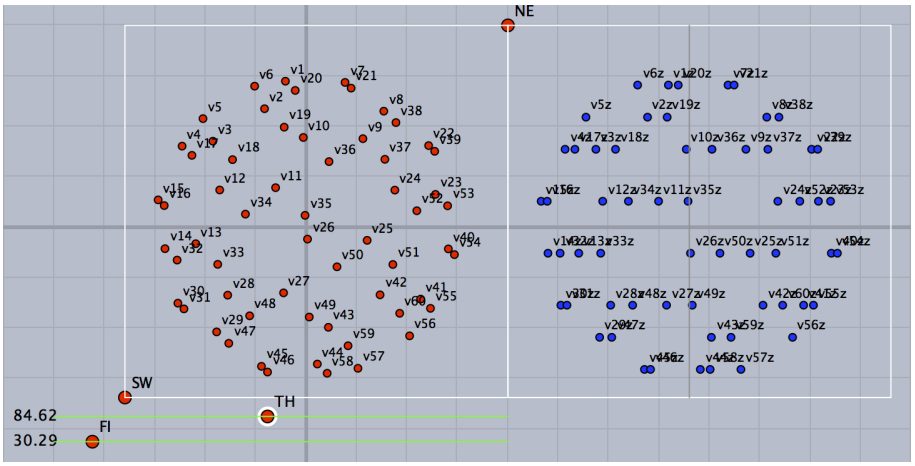


なお，VertexEdgeFace() で四面体が描かれるが，Phparadata() により非表示になる。Figure ボタンで描き出せば正しく出力されるので，Phparadata() を実行する前に画面上に表示して確認してから Phparadata() を実行するとよい。

【例】切頂二十面体を描く。
小林・鈴木・三谷による多面体データ `polyhedrons_obj` を用いて，s06 の切頂二十面体（サッカーボール型）を描く。 `polyhedrons_obj` は KeTCindy システムの data ディレクトリにあるので，Setdirectory() でカレントディレクトリを作業ディレクトリと切替ながら出力する。

```
Setdirectory( Dirhead+"/data/polyhedrons_obj");
phd=Readobj("s06.obj",["size=3"]);
Setdirectory(Dirwork);
VertexEdgeFace("s06",phd);
Phparadata("1","s06");
```

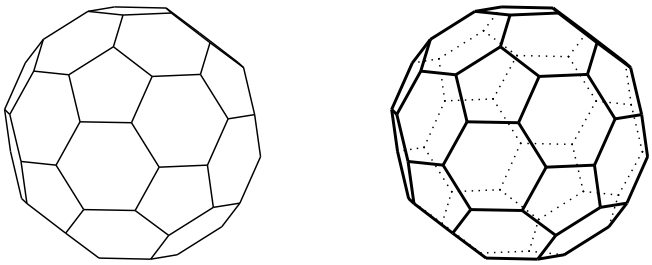
VertexEdgeFace() の name は通常の ”1” でもよい。その場合は,`Phparadata("1","1");` とするが，わかりにくいので上のようにした。
実行すると，Cinderella の描画面は次のように頂点だけが描かれる。



全体の線種と，陰線の線種を

```
Phparadata("1","s06",["dr,2","Hidden=do"]);
```

で指定したのが下図右である。



【注意】

polyhedrons obj のデータを使って，続けて異なる多面体を描きたい場合は注意が必要である。Readobj() だけを変更して別のデータを読めばよさそうであるが，前のデータが残っていてうまくいかない。VertexEdgeFace() の name を（したがって，Phparadata() の第 2 引数も）書き換えるか，作業フォルダ (fig) の中身を削除してから実行する。たとえば，上のコードで切頂二十面体を描いた後，正八面体（r02）を描こうとするならば，

```
Setdirectory( Dirhead+"/data/polyhedrons_obj");
phd=Readobj("r02.obj",["size=3"]);
Setdirectory(Dirwork);
VertexEdgeFace("2",phd);
Phparadata("1","2");
```

のようにする。

⇒ 関数一覧

関数　Nohiddenbyfaces(name,PD1,PD2,option1,option2)

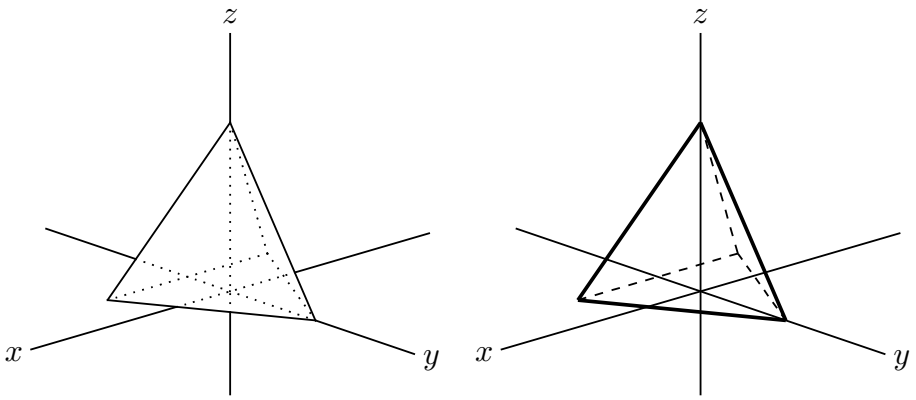
機能　面に対し曲線を陰線処理する

説明　PD2 で与えられた面に対し，曲線 PD1 の面に隠れている部分を陰線処理する。
引数 PD1 を省略するとすべての曲線が対象となる。陰線処理された線は初期設定では点線で表される。この線種は option2 で変更できる。たとえば，["da"] とすると破線になる。["nodsip"] とすると，陰線は表示されない。option1 は曲線全体の option であるので，option2 だけを指定する場合は，option1 として空リスト [ ] が必要である。option2 では，"Eps=" で，陰線処理時の許容限界を設定できる。陰線処理がうまくいかないときは，この値を `Eps=10^(-4)` のように変えてみるとよい。初期設定は `Eps=10^(-2)`。

【例】座標平面上に正四面体を描き，各軸と正四面体の辺を陰線処理する。（下図左）

```
Xyzax3data("","x=[-5,5]","y=[-5,5]","z=[-5,4]");
Putpoint3d("A",2*[-1,-1/sqrt(3),0]);
Putpoint3d("B",2*[1,-1/sqrt(3),0]);
Putpoint3d("C",2*[0,sqrt(3)-1/sqrt(3),0]);
Putpoint3d("D",2*[0,0,2*sqrt(6)/3]);
phd=Concatobj([[A,B,C],[A,B,D],[A,C,D],[B,C,D]]);
VertexEdgeFace("1",phd);
Nohiddenbyfaces("1","phf3d1");
```

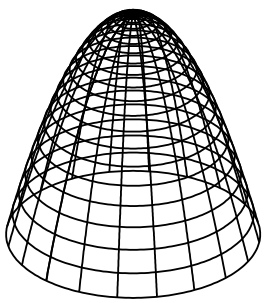
`VertexEdgeFace("1",phd);` によって，辺，頂点，面のプロットデータが作られる。`phf3d1` は，面のプロットデータである。
ここで，`Nohiddenbyfaces("1","phe3d1","phf3d1",["dr,2"],["da"]);` とすると，座標軸は陰線処理されず，正四面体の辺（`phe3d1`）だけが陰線処理されて破線で描かれる。四面体は太く描かれる。（下図右）



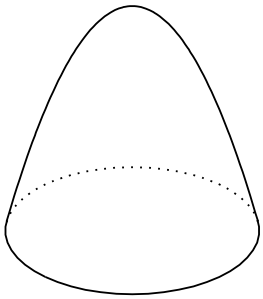
同様に，
`Nohiddenbyfaces("1","ax3d","phf3d1",[],["da"]);`
とすれば，座標軸だけが陰線処理されて破線で描かれる。

5.3.3 曲面

曲面の描画には，ワイヤーフレームモデルとサーフェスモデルがある。ワイヤーフレームモデルは，曲面を編目で表すものであり，サーフェスモデルは編目のない面として，その輪郭線を描くものである。



ワイヤーフレームモデル



サーフェスモデル

KeTCindy では，それぞれ次の関数を用いて描画する。

陰線処理をしないワイヤーフレームモデル	Sf3data(name,form,options)
サーフェスモデル	Sfbdparadata(name,form,options)
サーフェスモデルにワイヤーを描く	Wireparadata(name,PD,form,n1,n2,options)

ワイヤーフレームモデルで陰線処理をするためには，面のデータが必要なので，Sfbdparadata() で描画した後，Wireparadata() で描画する。

また，サーフェスモデルの描画では，陰線処理に時間がかかるため，C 言語の使用を前提としている。したがって，C 言語を用いて描画を行う ExeccmdC() を併用する。

引数の form は，方程式 と，変数の定義域を文字列のリストにしたものである。方程式のパターンは次の 3 通りがある。

(1) $z = f(x, y)$

【例】式：$z = x^2 - y^2$

定義域：$x = (-2, 2), y = (-2, 2)$

(2) $z = f(x, y), x = g(r, t), y = h(r, t)$

【例】式：$z = 4 - (x^2 + y^2), x = r\cos t, y = r\sin t$

定義域：$r = (0, 2), t = (0, 2\pi)$

(3) $x = f(u, v), y = g(u, v), z = h(u, v),$

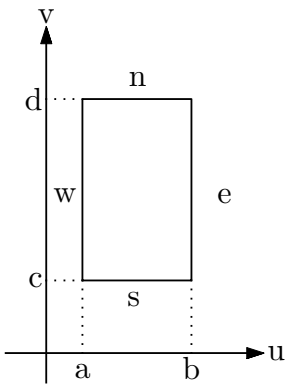
【例】式：$x = 2\sin u\cos v, y = 2\sin u\sin v, z = 2\cos u$

定義域：$u = (0, \pi), v = (0, 2\pi)$

ここで，(2) と (3) は媒介変数型で，x, y, z それぞれの式と，媒介変数 2 つの定義域からなっている。そのままでは区別がつかないので，引数として与えるときは，(3) の型には，識別文字として ”p” を先頭に付加する。

また，定義域については，開区間でとる場合と閉区間でとる場合がある。その区別を境界指定として”ewsn”で表す（ともに閉区間)。”ewsn” の意味は次のように考える。

変数が u, v のとき，u, v 平面において，$a \leq u \leq b, c \leq v \leq d$ の矩形を考え，境界値を東西南北 (ewsn) で示す。それぞれの文字が書かれたときは境界値を含む。



この境界指定を最後に付加するが，省略することもでき，省略した場合は初期値の ”ewsn” （閉区間）とする。ともに開区間とする場合は，”” とする。ただし，陰線処理をしない Sf3data() では，境界にも線を引くので，この指定は無効となる。

具体的な例を示そう。

次は，円錐の form である。（form は form data を短縮した fd がよく使われる）

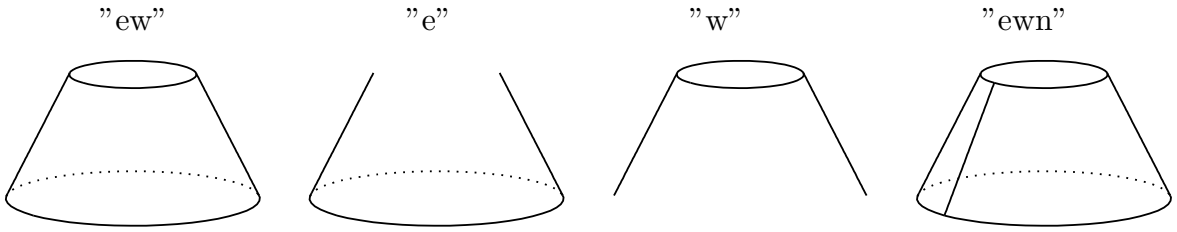
```
fd=["p","x=r*cos(t)","y=r*sin(t)","z=2*(2-r)","r=[1,2]","t=[0,2*pi]","ew"]
```

”ew” は $1 \leq r \leq 2$ であることを示す。したがって，円錐台の上面と底面が表示される。

これを ”e” とすると $1 < r \leq 2$ となり，上面が表示されない。

また，これを ”w” とすると $1 \leq r < 2$ となり，底面が表示されない。

さらに，”ewn” あるいは ”ews” としたり，指定を略して初期値の ”ewsn” とすると，$t = (0, 2\pi)$ の左右いずれかの値が含まれることになり，不要な境界線が現れる。



球面を描く場合はこの点で留意することがある。球面をサーフェスモデルで描くと単に円にしか見えないので，多くの場合はサーフェスモデルで描いた球面に Wireparadata() で陰線処理したワイヤーを入れることになるだろう。その場合，Wireparadata() では境界線には線を引かないので，境界指定を ”s” として，境界線を引く必要がある。境界指定を ”” とす

ると，経線が 1 本足りなくなるので注意されたい。具体例は，Wireparadata() に例示してある。

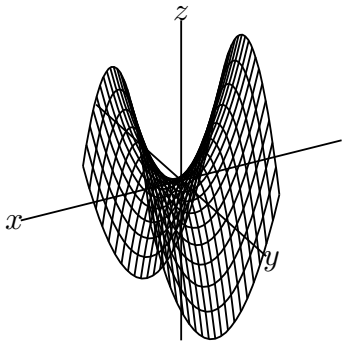
関数 Sf3data(name, リスト,options)

機能 陰線処理なしの曲面をワイヤーフレームモデルで描く

説明 options は，メッシュの密度と解像度（各変数に対応する分割数)。
メッシュ密度は，縦横で ”Wire=[a,b]” で指定。初期値は a,b とも 20。
解像度は，”Num=[a,b]” で指定。初期値は a,b とも 25。

【例】 $z = f(x, y)$ 型
$z = x^2 - y^2$ を定義域 $x = [-2, 2], y = [-2, 2]$ で描画する。

```
fd=["z=x^2-y^2","x=[-2,2]","y=[-2,2]"];
Sf3data("1",fd);
```



メッシュの数を縦横とも 10，解像度を x,y とも 10 にするとメッシュ密度，解像度とも下げるので粗い描画となる。

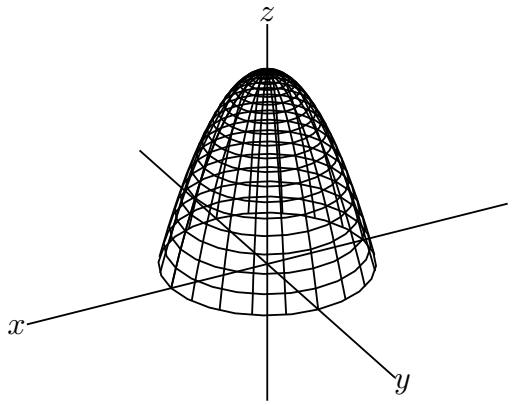
```
Sf3data("1",fd,["Num=[10,10]","Wire=[10,10]"]);
```

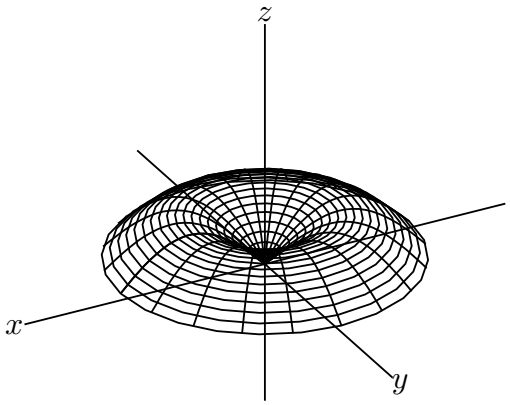
【例】$z = f(x, y), x = g(r, t), y = h(r, t)$ 型
次図左

```
fd=["z=4-(x^2+y^2)","x=r*cos(t)","y=r*sin(t)","r=[0,2]","t=[0,2*pi]"];
Sf3data("1",fd);
```

次図右

```
fd=["z=sin(sqrt(abs(x^2+y^2)))","x=r*cos(t)","y=r*sin(t)",
 "r=[0,3]","t=[0,2*pi]"];
Sf3data("1",fd);
```

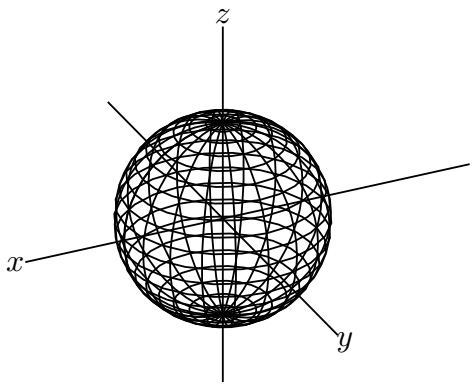




(3) $x = f(u, v), y = g(u, v), z = h(u, v)$, 型
この場合，(2) と区別するために，”p” を先頭につけておく。

【例】球面

```
fd=["p","x=2*sin(u)*cos(v)","y=2*sin(u)*sin(v)","z=2*cos(u)",
"u=[0,pi]","v=[0,2*pi]",""]; Sf3data("1",fd);
```



⇒ 関数一覧

関数 Sfbdparadata(name, 式,options)

機能 サーフェスモデルの輪郭線を描く

説明 この関数はデータを作るだけなので，表示するには ExeccmdC() を併用する。

options は，”Wait=n”,”r”,”m”, および線種。Wait の初期値は 30。

”r”,”m”に関しては,

オプションなしまたは,”　のとき

i) データファイルがなければ，新しく作る

ii) データファイルが既にあればそれを読み込む

”m”のとき，強制的にデータファイルを作り直す。

”r” のとき，すでにあるデータファイルを読み込む。

この処理は時間がかかるため，この関数を実行した状態で画面上のスライダやその他の点を動かそうとすると反応が悪くなる。そこで，Isangle() または Isptselected() を用いて，スライダの点を選択しているときはワイヤフレームモデルを描画するようにするとよい。

なお，C 言語の環境がない場合は，ExeccmdC() が使えないので，SfbdparadataR(name, 式,optons1,optons2) を使う。options2 には陰線の表示方法について ”nodisp” または線種 を指定する。 初期設定は”nodisp” 。options2 のみ指定するときは，options1 に [””] （空文字）を書いておく。

作図例は 次の ExeccmdC を参照のこと。

関数 ExeccmdC(name,options1,options2)

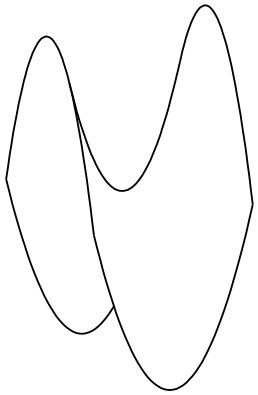
機能 曲面を表示する。戻り値は，対象にしたプロットデータのリスト。

説明 データが作成された曲面を表示する。

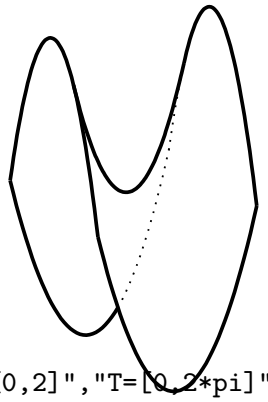
options1 には”r”,”m”, ”Wait=n” と輪郭線の線種が指定できる。
options2 には 軸の陰線について ”nodisp” または線種が指定できる。 初期設定は ”do”。
options2 だけを指定したい場合は，options1 を空リスト [ ] にする。

【例】サドル面
陰線を消去して表示

```
fd=["z=x^2-y^2","x=[-2,2]","y=[-2,2]"];
if(Isangle(),
Sf3data("1",fd);
,
Startsurf();
Sfbdparadata("1",fd);
ExeccmdC("1",[],["nodisp"]);
 );
```



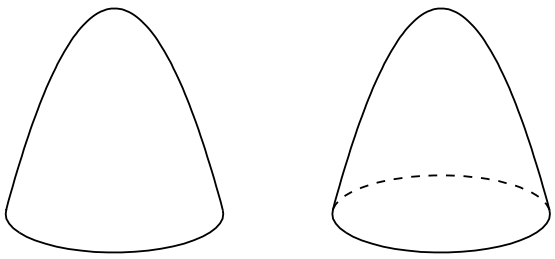
全体を実線で太めにして，陰線は初期設定の
点線で表示。
ExeccmdC() を変更する。
ExeccmdC("1",["dr,2"]);



【例】放物面： 式を変更する。

```
fd=["z=4-(x^2+y^2)","x=R*cos(T)","y=R*sin(T)","R=[0,2]","T=[0,2*pi]","e"];
```

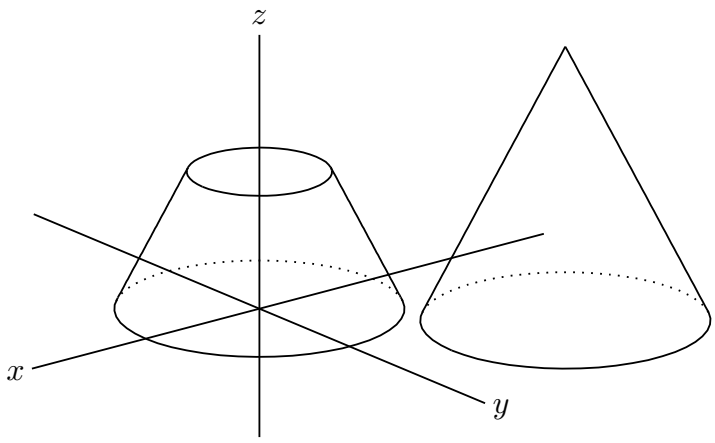
陰線を消去（下図左）：ExeccmdC("1",[],["nodisp"]);
陰線を破線で表示（下図右）：ExeccmdC("1",[],["da"]);



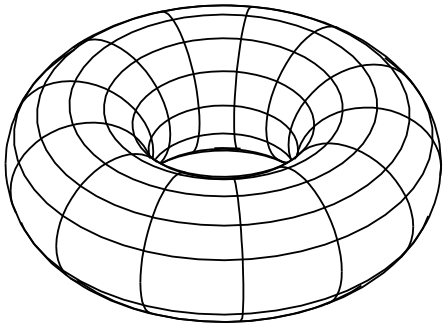
【例】曲面を 2 つ表示するときは，Sfbdparadata() の name は ”1” と ”2” にするが，まとめて ExeccmdC("1") で表示できる。

```
fd=[
 "p",
```

```
 "x=r*cos(t)","y=r*sin(t)","z=2*(2-r)",
 "r=[1,2]","t=[0,2*pi]","ew"
];
fd2=[
 "p",
 "x=r*cos(t)-3","y=r*sin(t)+3","z=2*(2-r)",
 "r=[0,2]","t=[0,2*pi]","ew"
];
if(!ptselected(),
 Startsurf();
 Sfbdparadata("1",fd);
 Sfbdparadata("2",fd2);
 ExeccmdC("1");
);
```



【例】トーラスを描く：軸は非表示にしておく。



```
 fd=["p","x=(2+cos(u))*cos(v)",
 "y=(2+cos(u))*sin(v)","z=sin(u)",
 "u=[0,2*pi]","v=[0,2*pi]","s"];
 if(Ptselected(),
 Sf3data("1",fd);
,
 Startsurf();
 Sfbdparadata("1",fd);
 Wireparadata("1","sfbd3d1",fd,12,12,[],["nodisp"]);
 ExeccmdC("1",[],["nodisp"]);
 );
```

⇒ 関数一覧

関数　Wireparadata(name,PD, 式, 整数, 整数,optons)

機能　Sfbdparadata で作成した曲面について，陰線処理したワイヤーを描く

説明　PD は，第 3 引数の式を用いて Sfbdparadata() で描いたサーフェスモデルのプロットデータ名。第 4，第 5 引数は分割線の数。

options には ”r”,”m”,”Wait=n” が指定できる。

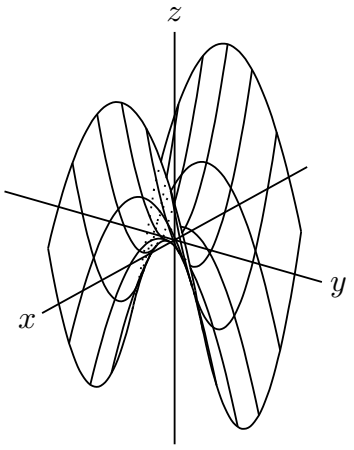
Sfbdparadata() とペアで使い，そのあとに ExeccmdC で描画する。

C 言語の環境がない場合は, ExeccmdC() が使えないので, WireparadataR(name,PD, 式, 整数, 整数,optons1,optons2) を使う。options2 には陰線の表示方法について ”nodisp” または線種 を指定する。 初期設定は”nodisp” 。options2 のみ指定するときは，options1 に [””] （空文字）を書いておく。

【例】

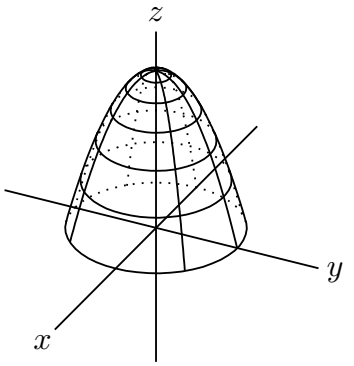
サドル面

```
fd=["z=x^2-y^2","x=[-2,2]","y=[-2,2]"];
if(Isangle(),
Sf3data("1",fd);
,
Startsurf();
Sfbdparadata("1",fd);
Wireparadata("1","sfbd3d1",fd,4,5);
ExeccmdC("1");
);
```



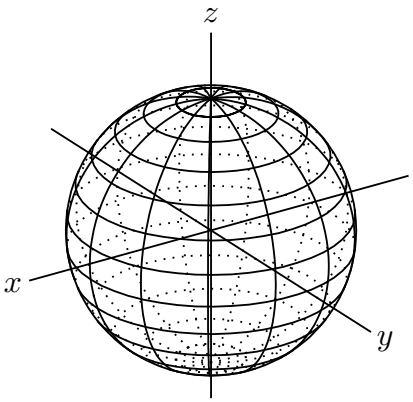
回転放物面：次を変更

```
fd=["z=4-(x^2+y^2)","x=r*cos(t)",
"y=r*sin(t)","r=[0,2]","t=[0,2*pi]","e"];
Wireparadata("1","sfbd3d1",fd,5,7);
```



球面

```
fd=["p","x=sin(u)*cos(v)","y=sin(u)*sin(v)","z=cos(u)",
"u=[0,pi]","v=[0,2*pi]","s"];
if(Ptselected(),
Sf3data("1",fd);
,
Startsurf();
Sfbdparadata("1",fd);
Wireparadata("1","sfbd3d1",fd,12,12);
ExeccmdC("1");
);
```



関数 Crvsfparadata(name,PD1,PD2, 式)

機能 曲面による曲線の陰線処理を行う。

説明 曲線 PD1 と曲面 PD2 について，PD1 は PD2 による陰線処理を行う。

Crvsfparadata() のあとに ExeccmdC() でまとめて描画する。

C 言語が使えない場合は，CrvsfparadataR(name,PD1,PD2, 式,options1,options2)

を使う。options1 は分割数と誤差限界， options2 は陰線の線種。

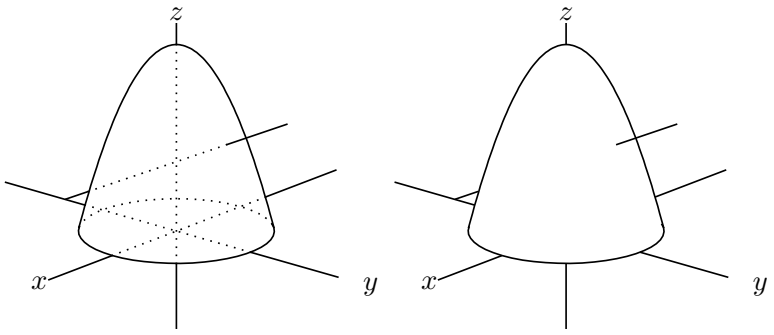
【例】回転放物面と座標軸，線分を陰線処理したデータを作って表示する。線分の端点 A,B はあらかじめ作図しておく。

ExeccmdC() の 初期設定では陰線は点線で表示される。（下図左）

```
Xyzax3data("","x=[-5,5]","y=[-5,5]","z=[-5,5]");
Putpoint3d(["A",[0,-3,0],"B",[0,3,3]]);
Spaceline([A,B]);
fd=["z=4-(x^2+y^2)","x=R*cos(T)","y=R*sin(T)","R=[0,2]","T=[0,2*pi]","e"];
Startsurf();
Sfbdparadata("1",fd);
Crvsfparadata("1","AB3d","sfbd3d1",fd);
Crvsfparadata("2","ax3d","sfbd3d1",fd);
ExeccmdC("1");
```

ExeccmdC() の options2 を [”nodisp”] にすると，陰線は非表示になる。（下図右）

```
ExeccmdC("1",[],["nodisp"]);
```



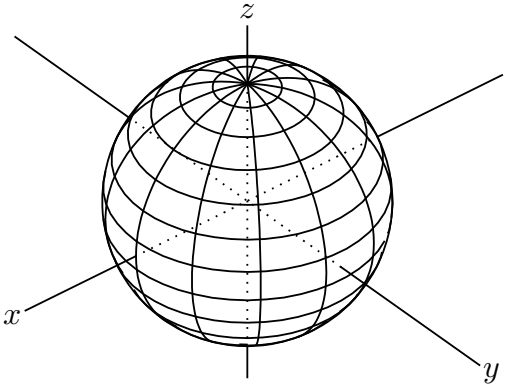
戻り値を使うと，Changestyle3d() を使って陰線のスタイル（線種，色）を変えることができる。戻り値の内容は，コンソールに「readoutdata from template3D1.txt : 」として表示されるので，これをテキストエディタで開き，操作対象を決めればよい。たとえば，上の左図で，線分 AB の陰線はリストの 4 番目の crvsfh3d1 なので，

```
ret=ExeccmdC("1");
Changestyle3d(ret_4,["da","Color=red"]);
```

とすると，赤の破線にすることができる。

【例】球面で座標軸を陰線処理し，球面の陰線は非表示で表す。

```
fd=["p","x=2*sin(u)*cos(v)",
 "y=2*sin(u)*sin(v)",
 "z=2*cos(u)",
 "u=[0,pi]","v=[0,2*pi]",""];
if(Isangle(),
Sf3data("1",fd);
,
Startsurf();
```



```
Sfbdparadata("1",fd);
Wireparadata("1","sfbd3d1",fd,12,12,[],["nodisp]);
Crvsfparadata("1","ax3d","sfbd3d1",fd);
ExeccmdC("1");
);
```

⇒ 関数一覧

5.4 プロットデータの操作

関数 Datalist2d()

機能 画面上のプロットデータのリストを取得する

説明 画面に描かれているすべてのプロットデータのリストを返す。
空間図形は，Cinderella の画面上に射影し表示する。そのため，KeTCindy3D は，空間におけるプロットデータと，画面上に表示するプロットデータの 2 つを作っている。Datalist2d() では，画面上に表示するプロットデータのリストを返す。

【例】

```
Xyzax3data("","x=[-5,5]","y=[-5,5]","z=[-5,5]");
Putpoint3d(["A",[0,-3,0],"B",[0,3,3]]);
Spaceline("1",[A,B]);
println("PD="+Datalist2d());
```

とすると，コンソールに PD=[ax2d,AB2d] と表示される。ax2d は座標軸のプロットデータ ax3d に，AB2d は線分 AB のプロットデータ AB3d に対応している。

関数 Datalist3d()

機能 空間のプロットデータのリストを取得する

説明 空間に描かれているすべてのプロットデータのリストを返す

【例】

```
Xyzax3data("","x=[-5,5]","y=[-5,5]","z=[-5,5]");
Putpoint3d(["A",[0,-3,0],"B",[0,3,3]]);
Spaceline("1",[A,B]);
println("PD="+Datalist3d());
```

とすると，コンソールに PD=[ax3d,AB3d] と表示される。

関数 Changestyle3d(リスト, リスト)

機能　3D プロットデータの属性を変更

説明　第 1 引数のプロットデータの属性を，第 2 引数に変更する。
たとえば，補助線など，画面には描いても TeX に書き出さない線を描画するときは，option に [”notex”] をつけるが，これをあとから付加したい場合に利用する。プロットデータはリストにできるので，複数のプロットデータの属性をまとめて変更することができて便利である。

【例】4 つの点で四面体の辺を描き，まとめて notex にする。点 A,B,C,D はとってあるものとする。

```
Spaceline("1",[A,B]);
Spaceline("2",[A,C]);
Spaceline("3",[B,C]);
Spaceline("4",[A,D]);
Spaceline("5",[B,D]);
Spaceline("6",[C,D]);
edges=apply(1..6,"sl3d"+text(#));
Changestyle3d(edges,["notex"]);
```

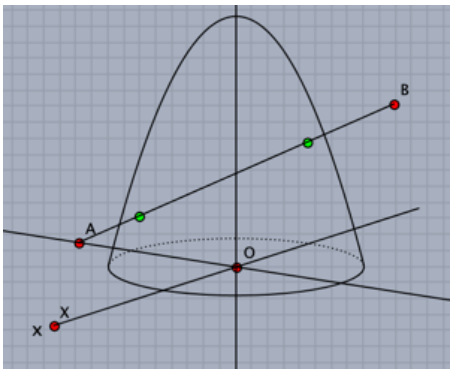
関数　Intersectcrvsf(name,PD，式)

機能　曲線と曲面の交点の座標を求める

説明　PD は曲線のプロットデータ。式は曲面の式。
曲面は，Sfbdparadata() でデータを作成し，ExeccmdC() で表示しておく。交点の座標は，”intercrvsf”+name に代入される。コマンドの実行順序は次の例のようにする。

【例】回転放物面と線分の交点の座標を表示する。

```
Putpoint3d(["A",[0,-3,0],"B",[0,3,2]]);
Spaceline("1",[A,B]);
fd=[
 "z=4-(x^2+y^2)","x=R*cos(T)","y=R*sin(T)",
 "R=[0,2]","T=[0,2*pi]","e"
];
Startsurf();
Sfbdparadata("1",fd);
Intersectcrvsf("1","sl3d1",fd);
ExeccmdC("1",[""]);
println("Intersect="+intercrvsf1);
Pointdata3d("1",intercrvsf1);
```



実行すると，コンソールに

```
Intersect=[[0,1.57,1.52],[0,-1.91,0.36]]
```

のように表示され，画面には交点が表示される。

⇒ 関数一覧

関数 IntersectsgpL(点名，線分，面，描画方法)

機能 空間の線分（直線）と平面の交点を求める。

説明 引数の線分は線分の端点を ”A-B” の形もしくは空間座標のリストで与える。

引数の面は，面内の 3 点を ”C-D-E” の形もしくは空間座標のリストで与える。

戻り値は，[pt,flag1,flag2,val1,val2]

pt：直線と平面の交点の座標。直線と平面が平行で交点が存在しない場合は空リスト []

flag1 ： 交点が線分内にあれば true ， なければ false

flag2 ： 交点が面内にあれば true，なければ false

val1,val2 ： 線分についてのパラメータ値，平面についてのパラメータ値

描画方法は，”put” または ”i” , ”e” 。

put ： 幾何点を作る

i ： 線分内にあれば点を描く

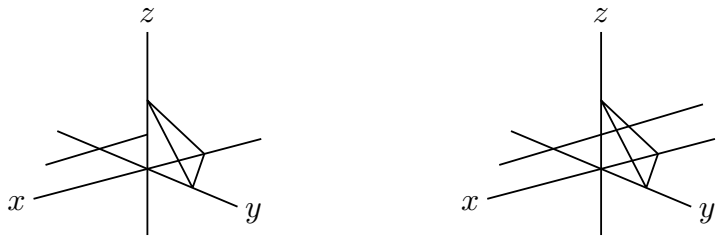
e :平面で交われば点を描く

【例】交点の有無と戻り値

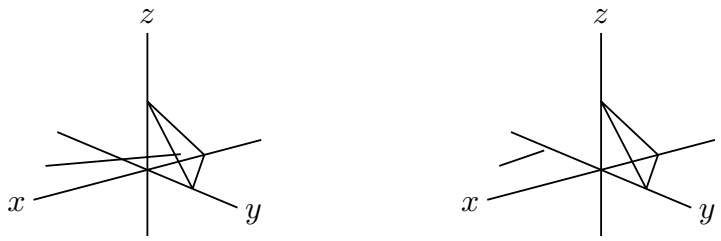
次のスクリプトで p2 を変えたときの戻り値の flag1, flag2

```
p1=[1,-1,0];
p2=[0,0,1/2];
p3=[0,1,0];
p4=[-1,0,0];
p5=[0,0,1];
Spaceline("1",[p1,p2]);
Spaceline("2",[p3,p4,p5,p3]);
ret=IntersectsgpL("P",[p1,p2],[p3,p4,p5],"put");
println("flag1="+ret_2+": flag2="+ret_3);
```

```
p2=[0,0,1/2]; p2=[-1,1,1];
flag1=false : flag2=trueflag1=true : flag2=true
```



```
p2=[1,2,1]; p2=[1,0,1/2];
flag1=true : flag2=false    flag1=false : flag2=false
```



【例】立方体を平面で切った図を描く。

いろいろな手順が考えられるが，ここでは次の手順で描く。

(1) 立方体の頂点をとる。１辺の長さを Hn とし，軸上の点を Putaxes3d() でとる。

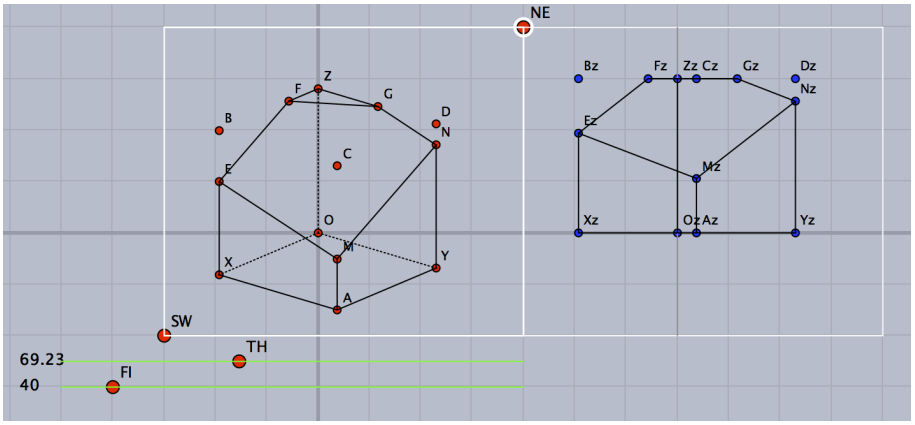
(2) 切断面を決める点 E,F,G を辺上の自由点として Putonseg3d() でとる。

(3) E,F,G を通る平面と，辺 AC,DY との交点をとり，M,N とする。

(4) 全体を多面体として面データを作って描画する。

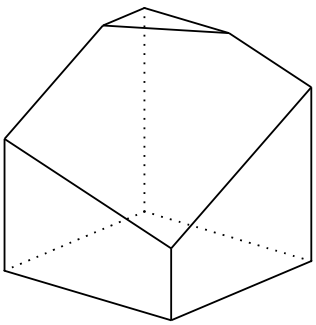
```
Hn=3;
Putaxes3d(Hn);
Putpoint3d("A",[Hn,Hn,0]);
Putpoint3d("B",[Hn,0,Hn]);
Putpoint3d("C",[Hn,Hn,Hn]);
Putpoint3d("D",[0,Hn,Hn]);
Putonseg3d("E",X,B);
Putonseg3d("F",Z,B);
Putonseg3d("G",Z,D);
IntersectsgpL("M","A-C","E-F-G","put");
IntersectsgpL("N","D-Y","E-F-G","put");
phd=Concatobj([[O,X,A,Y],[X,A,M,E],[A,Y,N,M],[Y,N,G,Z,O],
[O,Z,F,E,X],[Z,F,G],[E,M,N,G,F]]);
VertexEdgeFace("1",phd);
Nohiddenbyfaces("1","phf3d1");
```

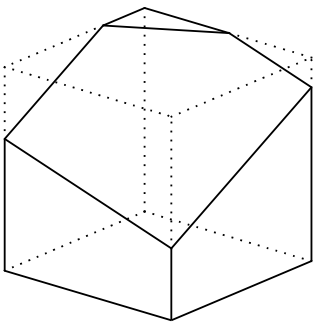
スクリプトを実行後，点 E,F,G をドラッグして適当な位置の断面にする。



できた図は下図左。これに，次のスクリプトを追加すれば，断面上方の立方体の各辺も点線で描かれる。（下図右）

```
Spaceline("1",[E,B,F],["do"]);
Spaceline("2",[B,C,M],["do"]);
Spaceline("3",[C,D,N],["do"]);
Spaceline("4",[D,G],["do"]);
```





⇒ 関数一覧

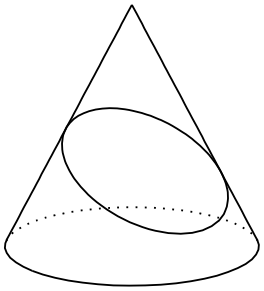
関数　Sfcutparadatacdy(name, 面, 曲面,options)

機能　平面と曲面の交線を求める。

説明　面の方程式は x,y,z の方程式，曲面は媒介変数表示で与える。

【例】円錐を平面 $y + 2z = 3$ で切った断面を表示する。

```
fd=[
 "p",
 "x=r*cos(t)","y=r*sin(t)","z=2*(2-r)",
 "r=[0,2]","t=[0,2*pi]","e"
];
Startsurf();
Sfbdparadata("1",fd);
Sfcutparadatacdy("1","y+2*z=3",fd);
ExeccmdC("1");
```



関数　Partcrv3d(name，始点，終点，PD)

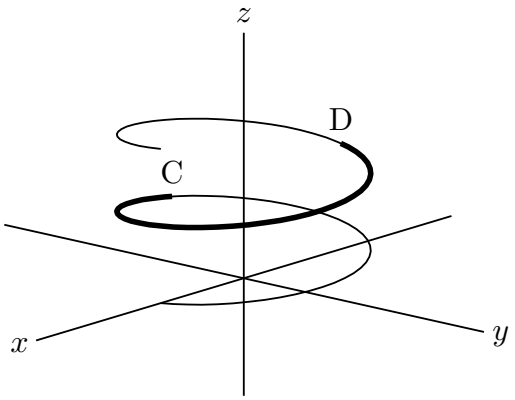
機能　部分曲線のプロットデータを作成する

説明　曲線 PD において，始点から終点までのプロットデータを作成する。
始点と終点は，プロットデータの番号もしくは曲線上にとった点の識別名で示す。

【例】螺旋を描き一部分を太くする。Putoncurve3d() で螺旋上に点 C,D ができるので，ドラッグして適当な位置に移動する。

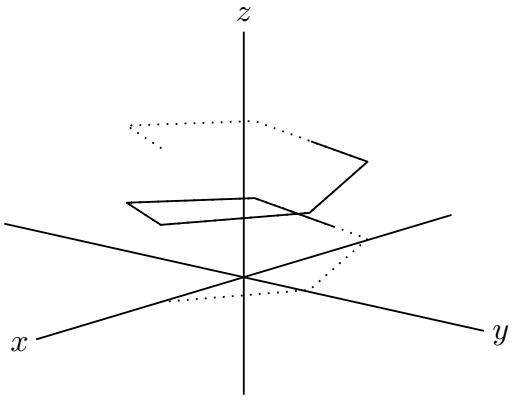
```
Xyzax3data("","x=[-5,5]","y=[-5,5]","z=[-5,4]");
Spacecurve("1","[2*cos(t),2*sin(t),0.2*t]","t=[0,4*pi]",["Num=100"]);
Putoncurve3d("C","sc3d1");
Putoncurve3d("D","sc3d1");
Partcrv3d("1",C,D,"sc3d1",["dr,3"]);
Letter([C,"n2","C",D,"n2","D"]);
```

ここで，"sc3d1" は，螺旋，"part3d1" は，部分曲線のプロットデータである。



【例】稲妻状の螺旋を点線で描き，その一部を実線にする。位置はプロットデータの番号で示す。小数にすると曲線を分割している線分の途中の位置になる。

```
Spacecurve("1","[2*cos(t),2*sin(t),0.2*t]","t=[0,4*pi]",["Num=10","do"]);
Partcrv3d("1",3.3,8.5,"sc3d1");
```



⇒ 関数一覧

関数 Reflectdata3d(name , PDlist , list , options)

機能 PD の鏡映を作る

説明 第 3 引数のタイプにより，点に関する鏡映，直線に関する鏡映，面に関する鏡映を作る。戻り値は鏡映したプロットデータのリスト。

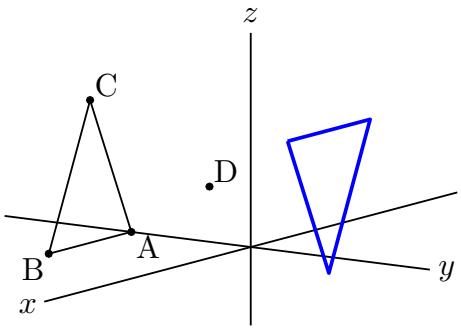
【例】点 A,B,C,D,E を空間にとり，三角形 ABC の鏡映を作る。

```
Putpoint3d(["A",[0,-2,0],"B",[2,-2,0],"C",[1,-2,2],"D",[1,0,1],"E",[1,0,0]]);
Spaceline("1",[A,B,C,A]);
```

で点をとり，三角形を描いておく。

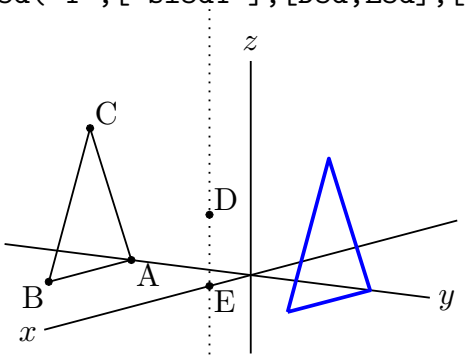
点 D に関する鏡映

```
Reflectdata3d("1",["sl3d1"],[D3d],["Color=blue","dr,2"]);
```



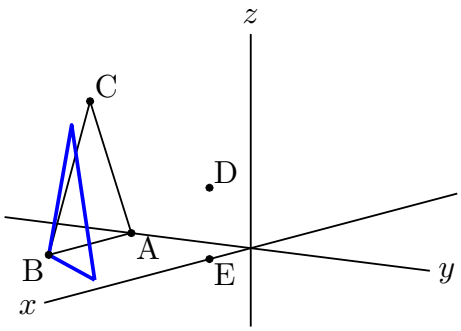
直線 DE に関する鏡映

```
Reflectdata3d("1",["sl3d1"],[D3d,E3d],["Color=blue","dr,2"]);
```



平面 BDE に関する鏡映

```
Reflectdata3d("1",["sl3d1"],[D3d,E3d,B3d],["Color=blue","dr,2"]);
```



関数　Reflectpoint3d(座標, リスト)

機能　点の鏡映点を求める

説明　第 2 引数のタイプにより，点に関する鏡映，直線に関する鏡映，面に関する鏡映のそれぞれの点の座標を返す。

【例】点 A,B,C,D を空間にとり，点 A の鏡映点の座標を求める。
点 B に関する鏡映点 `Reflectpoint3d(A3d,[B3d]);`
直線 BC に関する鏡映点 `Reflectpoint3d(A3d,[B3d,C3d]);`
平面 BCD に関する鏡映点 `Reflectpoint3d(A3d,[B3d,C3d,D3d]);`

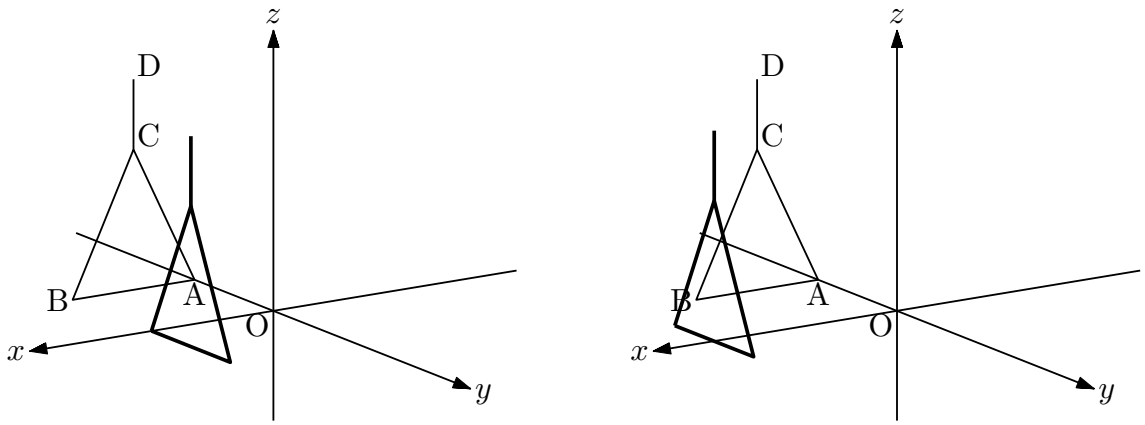
⇒ 関数一覧

関数 Rotatedata3d(name,PD リスト,vec, 角度,options)
機能 プロットデータを回転
説明 プロットデータを，原点を始点とするベクトル vec 周りに回転する。複数のプロットデータをまとめて回転することができる。戻り値は回転したプロットデータのリスト。options として，中心点（vec の始点），線種を指定することができる。

【例】コード例と結果を示す。

```
Xyzax3data("","x=[-5,4]","y=[-5,5]","z=[-5,4]",["a","O"]);
Putpoint3d(["A",[0,-2,0],"B",[2,-2,0],"C",[1,-2,2],"D",[1,-2,3]]);
Spaceline("1",[A,B,C,A]);
Spaceline([C,D]);
Rotatedata3d("1",["sl3d1","CD3d"],[0,0,1],pi/2,["dr,2"]);
Letter([A,"s","A",B,"w","B",C,"ne","C",D,"ne","D"]);
```

これを
`Rotatedata3d("1",["sl3d1","CD3d"],[0,0,1],pi/2,[[1,0,0],"dr,2"]);`
とした場合が右図である。



関数 Rotatepoint3d(座標, vec , 角度, [点])
機能 点の位置を回転する
説明 点を vec の周りに回転する。角度は弧度法で与える。
第 4 引数に点を与えた場合，vec の始点が第 4 引数の位置になる。デフォルトは原点
点 A を，$(0,-1,0)$ に置いたときの記述例と戻り値

```
Putpoint3d("A",[0,-1,0]);
Rotatepoint3d(A3d,[0,0,1],pi/2);// 戻り値は [1,0,0]
Rotatepoint3d(A3d,[0,0,1],pi/2,[1,1,0]); // 戻り値は [3,0,0]
```

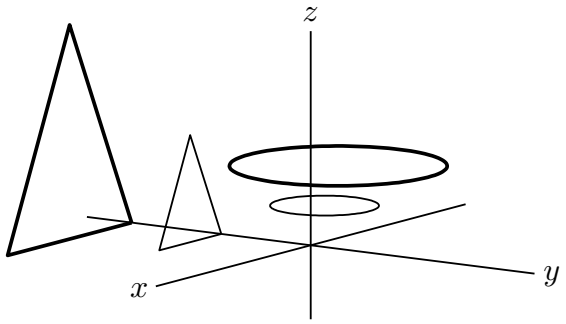
関数 Scaledata3d(name , PD リスト , vec , [中心,options])

機能 PD を拡大/縮小する。

説明 点は空間座標，vec は 3 次元ベクトルで倍率を表す。
中心と options はリストで与える。

【例】三角形と円を拡大/縮小する。

```
Putpoint3d(["A",[0,-2,0],"B",[2,-2,0],"C",[1,-2,2]]);
Spaceline("1",[A,B,C,A]);
Spacecurve("1","[cos(t)+1,sin(t)+1,1]","t=[0,2*pi]",["Num=100"]);
Scaledata3d("1",["sl3d1","sc3d1"],[2,2,2],[[0,0,0],"dr,2"]);
```



関数 Scalepoint3d(点,vec, 中心)

機能 点の位置を拡大/縮小する。

説明 点は空間座標，vec は 3 次元ベクトルで与える。

【例】コード例と結果を示す。

```
Putpoint3d(["A",[2,-1,2]]);
pt=Scalepoint3d(A3d,[3,2,4],[1,1,1]);
Putpoint3d(["B",pt]);
\vspace{\baselineskip}
```

点 B の位置は (4,-3,5) になる。

関数 Translatedata3d(name,PD, 平行移動量)

機能 空間プロットデータを平行移動

説明 PD で表される図形を，平行移動する。戻り値は平行移動したプロットデータのリ

スト。

【例】曲線 sc3d1 を y 軸方向に 2 だけ平行移動する。

```
Translatedata3d("1",["sc3d1"],[0,2,0]);
```

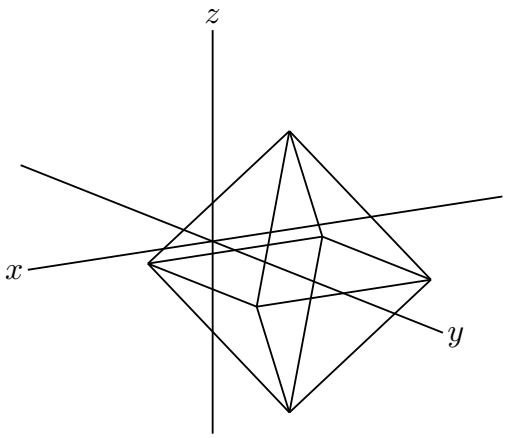
結果として，もとの曲線と平行移動した曲線の 2 つが描かれる。

【例】多面体の平行移動

VertexEdgeFace() で描いた多角形はこの関数では平行移動できないので，面データを直接操作して平行移動を行う。

たとえば，小林・鈴木・三谷による多面体データ polyhedrons obj を用いて正八面体を描く場合，次のようにする。y 軸方向に 2 だけ平行移動する場合である。

```
Setdirectory( Dirhead+"/data/polyhedrons_obj");
phd=Readobj("r02.obj",["size=2"]);
Setdirectory(Dirwork);
dn=length(phd_1);
repeat(dn,s,phd_1_s=phd_1_s+[0,2,0]);
VertexEdgeFace("1",phd);
```



関数　Translatepoint3d(座標，平行移動量)

機能　空間点を平行移動

説明　点を平行移動する。

【例】点 A(1,0,0) を (-1,1,1) だけ平行移動した点を B とする。点 A の空間座標は A3d で表される。

```
Putpoint3d(["A",[1,0,0]]);
pt=Translatepoint3d(A3d,[-1,1,1]);
Putpoint3d(["B",pt]);
```

⇒ 関数一覧

5.5 その他

関数 Perpplane(点名, 点, ベクトル，option)

機能 点を通り線分に垂直な平面上に基準点を 2 つとる

説明 引数の点名は，作成する 2 点で ”A-B” の形

第 2 引数は通る点の名称または座標

第 3 引数は法線ベクトル

option は ”put”で，2 つの幾何点を作図する。option がない場合は幾何点は作らず，無名の点のみを表示する。put 以外の文字列を書いたときは無効な命令とし，何も作成されない。

記述例を示すと

```
Perpplane("A-B","P",[1,1,1],"put");
```

点 P を通り，法線ベクトル (1,1,1) に垂直な平面上に点 A,B をとる。

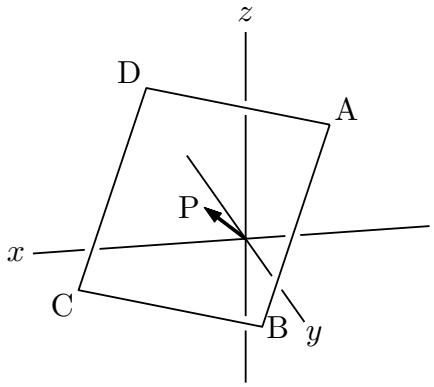
```
Perpplane("A-B","P",P3d-O3d);
```

点 P を通り，線分 OP に垂直な平面上に点 A,B をとる。これらにおいて，PA と PB は垂直で，PA=PB=1 となる。

【例】ベクトル $\vec{p} = (1, 1, 1)$ に垂直で点 $(1, 1, 1)$ を通る平面 ABCD を描く。

点 A,B,C,D は作図ツールで適当に取っておく。正確な位置はスクリプトで決める。

```
Xyzax3data("","x=[-5,5]","y=[-5,5]","z=[-5,4]");
Putpoint3d(["O",[0,0,0]]);
Putpoint3d(["P",[1,1,1]]);
Perpplane("E-F","P",P3d-O3d,"put");
vec1=2*(E3d-P3d);
vec2=2*(F3d-P3d);
Putpoint3d(["A",P3d+vec1+vec2]);
Putpoint3d(["B",P3d+vec1-vec2]);
Putpoint3d(["C",P3d-vec1-vec2]);
Putpoint3d(["D",P3d-vec1+vec2]);
Spaceline("1",[A,B,C,D,A]);
Arrowdata([O,P],["dr,2"]);
Letter([P,"w","P",A,"ne","A",B,"e","B",C,"ws","C",D,"nw","D",]);
Skeletonparadata("1");
```



関数 Perppt(点名，点，点リスト,option)

機能 平面に下ろした垂線の足を求める

説明 第 2 引数の点から，第 3 引数の点リストで決まる平面に下した垂線の足を，第 1 引数の名前の点とする。

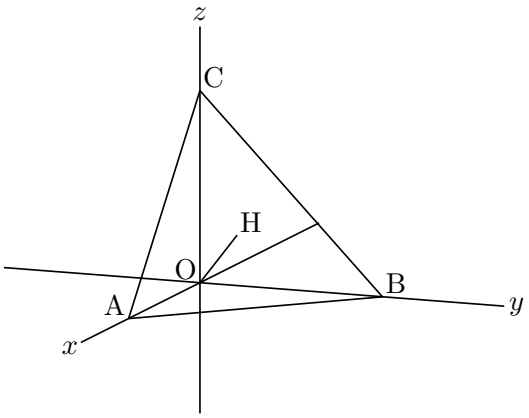
オプションは次の通り。 初期設定は ”draw”

draw：点を打つ。幾何点は作らない
put ：幾何点を作る
none：計算だけ行い，点は作図しない。

【例】原点から点 ABC を通る平面に下した垂線の足 H の座標を求める。
`Perppt("H","O","A-B-C","none");` 表示はされない。
`Perppt("H","O","A-B-C");` Hの位置に緑色の点が表示される。
`Perppt("H","O","A-B-C","put");` 幾何点 H が作図される。
いずれの場合も，H の座標は変数 H3d に代入される

作図例

```
Xyzax3data("","x=[-5,5]","y=[-5,5]","z=[-5,4]");
Putpoint3d("O",[0,0,0]);
Putpoint3d("A",[3,0,0]);
Putpoint3d("B",[0,3,0]);
Putpoint3d("C",[0,0,3]);
Perppt("H","O","A-B-C","put");
Spaceline("1",[A,B,C,A]);
Spaceline("2",[O,H]);
Letter([A,"nw","A",B,"ne","B",C,"ne","C",O,"nw","O",H,"ne","H"]);
```



⇒ 関数一覧

関数 Projcoordpara(座標)
機能 投影座標を求める
説明 空間座標を平面に投影した座標を求める。
戻り値の第 1，第 2 要素は Cinderella の描画面の x,y 座標。第 3 要素は xy 平面に垂直な z の座標で．投影面からの（符号付）距離を表す。

【例】`Projcoordpara([3,1,2]);`
戻り値は [-0.65,1.7,3.27] のようになる。（視点によって値は異なる）

関数　Readobj(ファイル名)

機能　obj ファイルを読み込む。

説明　小林・鈴木・三谷による整面凸多面体のデータは

http://mitani.cs.tsukuba.ac.jp/polyhedron/

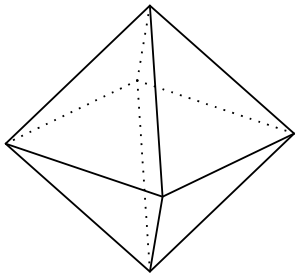
からダウンロードできる。polyhedrons_obj を，例えば，ユーザホームの ketcindy 作業フォルダに入れておく。

```
Setdirectory(gethome()+"/ketcindy/polyhedrons_obj");
polydt=Readobj("r02.obj");
Setdirectory(Dirwork);
```

これで，r02.obj データが，変数 polydt に代入される。
オプションは ”size=n”で，n 倍したデータにする。負の数にすると上下が反転される。
データは KeTCindy の data フォルダの中にある。したがって，次のようなスクリプトを書く。読み込むのは一度だけなので， Draw スロットではなく Initialization スロットに置けばよいが，コードの可読性を高めるには Draw スロットでもよい。
この多面体データは，頂点リストと面リストからなっているが，頂点リストは座標のリストなので，読み込んで表示するときには，点の名称を v1,v2,・・・ とする。
読み込んだあとの使い方を含めて例を示す。

【例】polydt を用いて正八面体を描く

```
VertexEdgeFace("1",polydt);
Nohiddenbyfaces("1","phf3d1");
```



主なデータは次の通り。

番号	名称	番号	名称	番号	名称
r01	正四面体	s02	二十・十二面体	s08	斜立方八面体
r02	正八面体	s03	切頭四面体	s09	斜十二・二十面体
r03	正六面体	s04	切頭八面体	s10	切頭立方八面体
r04	正十二面体	s05	切頭立方体	s11	頭切二十・十二面体
r05	正二十面体	s06	切頭二十面体	s12L/R	変形立方体
s01	立方八面体	s07	切頭十二面体	s13L/R	変形十二面体

この他に n01〜n92 まで整面凸多面体がある。

関数 Xyzcoord(P.x,P.y,Pz.y)

機能 主副画面で決まる点の座標

説明 Cinderella の描画面上の点が表す空間座標を求める
点Pについて，主画面の点Pに対応するのが副画面の Pz である。点 P の 2 次元座標は P.x,P.y で，Pz の y 座標は Pz.y で表される。これを引数として与えると，点Pの空間座標が返される。

【例】点 A をドラッグして動かしたとき，A の座標を求める。
`println(Xyzcoord(A.x,A.y,Az.y));`
により，コンソールに座標が表示される。

関数 Isangle()

機能 角度スライダ（視点スライダ）の選択判断

説明 角度スライダを選択しているときは true ，そうでないときは false を返す。
曲面の描画・陰線処理には時間がかかるため，角度スライダを動かすと反応が悪くなる。そこで，角度スライダを選択しているときは曲面の描画をしないようにすることで反応がよくなる。

【例】放物面の描画
次のようにすると，スライダの点を選んでいる間はワイヤフレームモデルが描かれ，画面上の他の部分をクリックして選択状態が解除されると陰線処理された放物面が描かれる。

```
fd=[
 "z=4-(x^2+y^2)",
 "x=R*cos(T)","y=R*sin(T)",
 "R=[0,2]","T=[0,2*pi]","e"
];
if(Isangle(),
 Sf3data("1",fd);
 ,
 Startsurf();
 Sfbdparadata("1",fd);
 Crvsfparadata("1","ax3d","sfbd3d1",fd);
 ExeccmdC("1");
 );
```

関数 Dist3d(a1,a2)

機能 空間の 2 点間の距離を返す

説明 引数 a1,a2 は作図点の名称，空間点の名称のいずれでもよい。
次の 3 通りの記法は同じ結果を返す。混在も可

```
Dist3d("A","B");
Dist3d(A,B);
Dist3d(A3d,B3d);
```

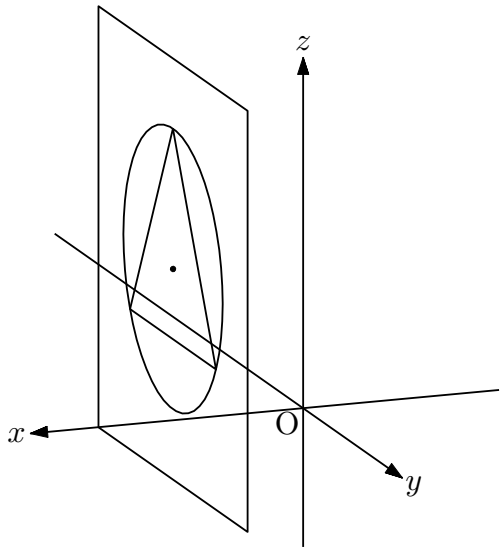
関数　Embed(name,PD リスト, 式, 変数リスト)

機能　2 D図形の空間内平面へ埋め込む

説明　第 2 引数は 2 Dの図形のプロットデータのリスト，式と変数は平面を記述する式と変数。平面は原点 vo と 2 つの基本ベクトル $\vec{vx}, \vec{vy}$ を用いて，$vo + x \cdot \vec{vx} + y \cdot \vec{vy}$ の形で表すことができる。変数（基本ベクトルの係数）は x, y でなく，s, t でもよい。式，変数リストともに文字列にする。また，基本ベクトルは直交していなくてもよいし，長さが異なってもよいが，縦横同じスケールの直交座標系にするのがわかりやすいだろう。

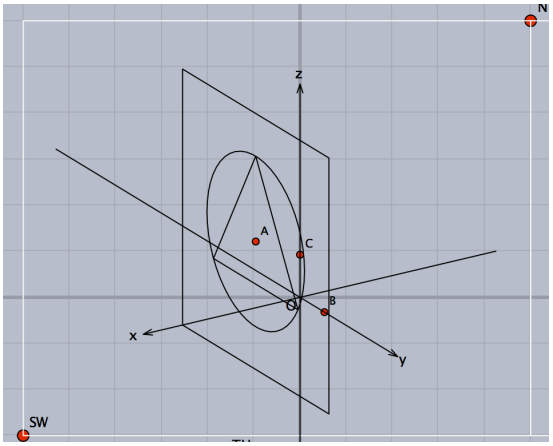
【例】正三角形と外接円を空間内の平面に埋め込む

```
Xyzax3data("","x=[-5,4]","y=[-10,4]","z=[-5,5]",["a","O"]);
Spaceline("1",[[3,0,0],[3,6,0],[3,6,6],[3,0,6],[3,0,0]]);
Defvar("vo=[3,3,3]");
Defvar("vx=[0,1,0]");
Defvar("vy=[0,0,1]");
Putpoint3d(["A",[3,3,3]],["fix"]);
Circledata("1",[[0,0],[2,0]],["nodisp"]);
Listplot("1",[[0,2],[-sqrt(3),-1],[sqrt(3),-1],[0,2]],["nodisp"]);
Embed("1",["cr1","sg1"],"vo+x*vx+y*vy","[x,y]");
Pointdata("1",[A],["Size=3"]);
```



ここで，Embed() で引き渡す vo,vx,vy については，R での変数定義が必要なので（KETCindy では行わない）Defvar() によって定義をしている。
原点，基本ベクトルを，点を作図して次のようにすることもできる。この場合は Defvar() は不要。

```
Putpoint3d(["A",[3,3,3],"B",[0,1,0],"C",[0,0,1]]);
Embed("1",["cr1","sg1"],"A3d+x*B3d+y*C3d","[x,y]");
```

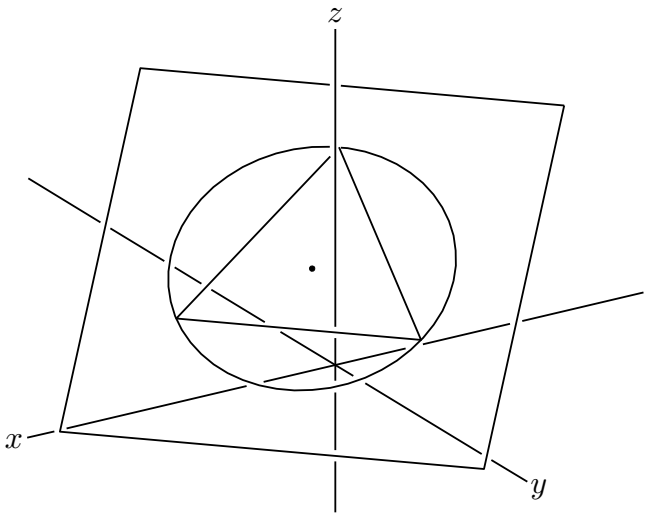


この場合，点 B,C の座標がそのまま基本ベクトルとなっているが，原点 A に対して描画平面上には B,C がないので図がわかりにくい。図をわかりやすくするならば次のようにする。

```
Putpoint3d(["A",[3,3,3],"B",[3,4,3],"C",[3,3,4]]);
Embed("1",["cr1","sg1"],"A3d+x*(B3d-A3d)+y*(C3d-A3d)","[x,y]");
```

また，平面を記述するのに，平面の原点と法線ベクトルを用いて Perpplane() を用いると，基本ベクトルが生成されるので、これを利用することができる。次のスクリプトでは，Skeletonparadata() を用いて陰線処理もしている。

```
Xyzax3data("","x=[-5,5]","y=[-8,5]","z=[-5,5]");
Putpoint3d(["O",[0,0,0],"P",[1,1,2]]);
Perpplane("E-F","P",P3d-O3d,"put");
vec1=3*(E3d-P3d);
vec2=3*(F3d-P3d);
Putpoint3d(["A",P3d+vec1+vec2]);
Putpoint3d(["B",P3d+vec1-vec2]);
Putpoint3d(["C",P3d-vec1-vec2]);
Putpoint3d(["D",P3d-vec1+vec2]);
Spaceline("1",[A,B,C,D,A]);
Circledata("1",[[0,0],[2,0]],["nodisp"]);
Listplot("1",[[0,2],[-sqrt(3),-1],[sqrt(3),-1],[0,2]],["nodisp"]);
Embed("1",["cr1","sg1"],"P3d+x*(E3d-P3d)+y*(F3d-P3d)","[x,y]");
Pointdata("1",[P],["Size=3"]);
Skeletonparadata("1");
```



⇒ 関数一覧

関数　Parapt(座標)

機能　点の投影面での座標

説明　引数の空間座標に対応する Cinderella の描画面の座標を返す。

【例】`Parapt([2,1,5]);` により，点 (2,1,5) が表示されている描画面の座標，たとえば [-0.52,3.27] が返される。

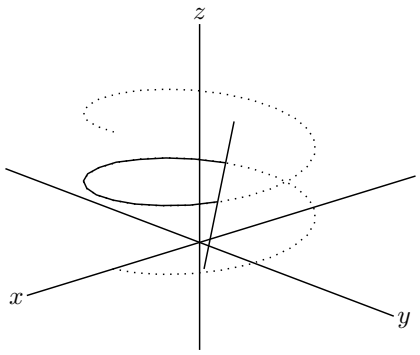
関数　Invparapt(座標，PD)

機能　描画面上の座標に対応する曲線上の点の座標を返す

説明　Cinderella の描画面上の座標を与えて，それに対応する曲線上の 3 次元座標を返す。空間内の曲線を作図すると，曲線の空間内のプロットデータとともに，描画面上に描くためのプロットデータも作られる。これを利用すると，描画面上の位置から曲線上の座標を求めることができる。

【例】螺旋と線分を描いたとき，描画面上での交点（空間内の交点ではない）に対応する螺旋上の点の座標を求め部分曲線を描く。

```
Spaceline("1",[[-1,-1,-1],[1,2,3]]);
Spacecurve("1","[2*cos(t),2*sin(t),0.2*t]","t=[0,4*pi]",["do"]);
tmp=Intersectcrvs("sl2d1","sc2d1");
p1=Invparapt(tmp_1,"sc3d1");
p2=Invparapt(tmp_2,"sc3d1");
Partcrv3d("1",p1,p2,"sc3d1");
```



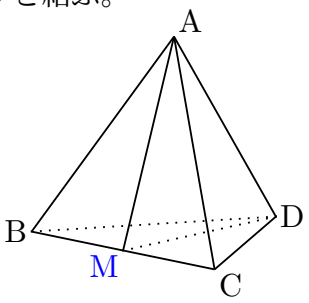
ここで，sl2d1,sc2d1 は線分と螺旋の描画面上での（平面の）プロットデータである。Intersectcrvs() で平面上の交点の座標（複数あるのでリストが返る）を求め，Invparapt() で対応する螺旋上の点の座標を求めて部分曲線を描いている。実際に交わる点での部分曲線ではないことに注意。

関数　Expr3d([位置, 方向, 文字列],option)
機能　文字列を表示する
説明　Expr() と同じ書式。「位置（座標）」だけ，空間座標にする。

関数　Letter3d([位置, 方向, 文字列],option)
機能　文字列を表示する
説明　Letter() と同じ書式。「位置（座標）」だけ，空間座標にする。

【例】三角錐 ABCD を描いて，BC の中点に点 M を取って A,D と結ぶ。

```
Putpoint3d("A",2*[0,0,2*sqrt(6)/3]);
Putpoint3d("B",2*[1,-1/sqrt(3),0]);
Putpoint3d("C",2*[0,sqrt(3)-1/sqrt(3),0]);
Putpoint3d("D",2*[-1,-1/sqrt(3),0]);
Putpoint3d("M",(B3d+C3d)/2);
phd=Concatobj([[A,B,C],[A,B,D],[A,C,D],[B,C,D]]);
VertexEdgeFace("1",phd);
Spaceline("1",[A,M,D]);
Nohiddenbyfaces("1","phf3d1");
Letter3d([A3d,"ne","A",B3d,"w","B",C3d,"se","C",D3d,"e","D"]);
Letter3d(M3d,"sw","M",["Color=blue"]);
```



⇒ 関数一覧

関数　Getangle()
機能　回転角の取得
説明　スライダで設定できる回転角（視点の位置）TH と FI を取得する。これは，スライダ

の左側に表示されている値である。戻り値は，リスト [TH,FI] で，角は度数法で表される。なお，内部変数は，THETA と PHI で，弧度法で表されている。
回転角の設定については，回転角の設定を参照のこと。

⇒ 関数一覧

6 KeTJS

6.1 CindyJS と KeTJS

CindyJS は，Cinderella の作品を HTML にして Web 上で動かせるようにしたものである。Cinderella のファイルメニューには,「CindyJS に書き出す」があり, これにより, HTML ファイルができる。できた HTML ファイルをそのままダブルクリックすると，JavaScript で記述されたプログラムが走る。

現在のところは Cinderella と完全互換ではなく，Web 上では Mouse スロットなどが使えなかったり，線種が指定できなかったりする。それでも，自作のボタンは有効なので，これで Web 上のアプリケーションを作ることができる。

KeTJS では，CindyScript にはない KeTCindy の関数（コマンド）を使って，作図アプリケーションを作ることができる。CindyJS の拡張版ともいえる。

ただし，CindyJS が Cinderella と完全互換でないように，KeTJS も KeTCindy と完全互換ではない。たとえば，いまのところ空間図形は描けない。それでも，KeTCindy のシステムが入っていない環境でも Web ブラウザで教材が使えることのメリットは大きいだろう。なお，HTML なので，TeX のファイルへの書き出し機能はない。

KeTJS で HTML を作成するには, いったん Cinderella のファイルメニューから「CindyJS に書き出す」を選んで HTML を作成した後,「KeTJS」ボタンをクリックする。「CindyJS に書き出す」で書き出された HTML は，実行のための環境設定しか書かれていない。これに，作図プログラムを追加するのが「KeTJS」「KeTJSoff」ボタンである。この 2 つのボタンの違いについては，次節「動作環境」を参照のこと。

また，ひな形として，template2slide.cdy または samples フォルダの s16 に入っているものを使うとよい。「KeTJS」「KeTJSoff」ボタンが設定してある。

6.2 KeTJS の動作環境

Cinderella のファイルメニューから「CindyJS に書き出す」を選ぶと，CSS とランタイムへのリンク，および JavaScript のコードが書かれた HTML ファイルが書き出される。CSS とランタイムへのリンクは

```
<link rel="stylesheet" href="https://cindyjs.org/dist/v0.8/CindyJS.css">
<script type="text/javascript" src="https://cindyjs.org/dist/v0.8/Cindy.js">
</script>
```

となっている。つまり，Web 上からランタイム Cindy.js をオンラインでダウンロードして JavaScript を動かすことになる。そのため，ファイルをブラウザで開いたときに少し時間がかかる。また，インターネットに接続できないと実行できない。

この CSS とランタイムは，自分のコンピュータに置くことができ，KeTCindy では，KeTCindy のライブラリとともに，ketcindyjs というフォルダ内に入っている。

また，JavaScript のコードの方は，これだけでは KeTCindy のコマンドでの作図はできない。

そこで，KeTJS ボタンをクリックすると，リンク先はそのままで，JavaScript のコードを追加して，ファイル名に json を追加した HTML ファイルを作る。

また，KeTJSoff ボタン（オフラインで使う）をクリックすると，ランタイムをダウンロードして，Cinderella のファイルと同じフォルダに ketcindyjs フォルダを作る。リンク先は次のように ketcindyjs に変え，JavaScript のコードを追加して，ファイル名に jsoffL を追加した HTML ファイルを作る。

```
<link rel="stylesheet" href="ketcindyjs/CindyJS.css">
<script type="text/javascript" src="ketcindyjs/Cindy.js"></script>
```

このとき，関数 `Setketcindyjs()` で，オプションを `"Local=n"` とすると，ランタイムとして，kettex の中にある ketcindyjs を使う。このときはファイル名に jsoff を追加する。

このときのリンク先は，Windows の場合は

```
<link rel="stylesheet" href="file:///C:/kettex/texlive/texmf-dist/
scripts/ketcindy/ketcindyjs/CindyJS.css">
<script type="text/javascript" src="file:///C:/kettex/texlive/
texmf-dist/scripts/ketcindy/ketcindyjs/Cindy.js"></script>
```

Mac の場合は

```
<link rel="stylesheet" href="file:////Applications/kettex/texlive/
texmf-dist/scripts/ketcindy/ketcindyjs/CindyJS.css">
<script type="text/javascript" src="file:////Applications/kettex/
texlive/texmf-dist/scripts/ketcindy/ketcindyjs/Cindy.js"></script>
```

である。

注）MacOS 10.14 Mojave の場合，Applications へのパスが通らないことがある。（2019 年 2 月 16 日現在）その場合は，`Setketcindyjs()` のオプションを `"Local=n"` としない。

ファイル名について

Cinderella のファイルメニューから「CindyJS に書き出す」ときのファイル名は，初期設定では作図中のファイル名と同じ。このファイル名は書き出すときに指定できる。

「KeTJS」ボタンで書き出すときのファイル名は，`Setfiles()` でファイル名が指定されていなければ，作図中のファイル名と同じ。これが「CindyJS に書き出す」で書き出したときのファイル名で同じでないと，KeTJS の HTML は作成されない。したがって，次のいずれかでファイル名を決める。

(1) ファイル名は作図中の Cinderella のファイル名と同じにする。

このときは，`Setfiles()` を使わない。

(2) 作図中の Cinderella のファイル名と別の名前にする。

このときは，`Setfiles("filename")` を使い，CindyJS に書き出すときに `"filename"` で書き出す。

6.3 KeTJS の設定

関数 Setketcindyjs(options)

機能 KeTJS の設定

説明 オプションを設定しない場合（ Setketcindyjs() ）は，初期値が使われる。
オプションは次の通り。

Local	y/n	動作環境の設定。初期値は y
Scale	実数	拡大・縮小　　初期値は 1
Grid	実数	グリッドサイズ（mm）
Nolabel	点のリスト	ラベルを表示しない点を指定する。”No”だけでも可。
		”Nolabel=all”とすると，すべての点のラベルが表示されない。
Color	色名またはコード	背景色の指定。初期値は lightgray([0,0,0,0.17])
Figure	y	書き出す範囲を NE,SW の範囲にする。
Axes	false	CindyJS が表示している座標軸を非表示にする。

【例】すべて設定するとき，次のように記述する。

```
Setketcindyjs(["Local=n","Scale=1.5","No=[A,B]","Color=lightgray"])
```

注１）Cinderella の画面に背景の方眼が描かれているとき，「CindyJS に書き出す」で，この方眼も表示される。方眼を消したい場合は，画面下の「グリッドを描く」ツールで非表示にしておく。

注２）HTML なので，画面に説明文などを表示したい場合は，HTML ファイルを開き，下の方の `<body>` と `</body>` の間，`<div id="CSCanvas"></div>` の前後に書けば表示される。改行などは，HTML のタグを用いる。

コメント化 no ketjs

機能 KeTJS への書き出しの有無を指定

説明 コマンドの末尾に　`//` に続いて書くと，その行は KeTJS に出力しない。ブロック単位で非出力にするには，ブロックを `//no ketjs on`，`//no ketjs off` ではさむ。

【例】次のスクリプトは，いずれも画面上では四角形 ABCD と対角線 AD,BC を描いている。

```
Listplot("1",[A,B,C,D,A]);
Listplot("2",[A,C]);
Listplot("3",[B,D]);// no ketjs
```

この場合，HTML では対角線 BD は描かれない。

```
Listplot("1",[A,B,C,D,A]);
// no ketjs on
Listplot("2",[A,C]);
Listplot("3",[B,D]);
// no ketjs off
```

この場合，対角線は 2 本とも描かれない。

KeTJS **だけで有効**　only ketjs

機能　KeTJS だけに書き込む

説明　コマンドの先頭に　`//` につけて，行の最後に続けて書くと，その行は KeTJS だけで有効となる。ブロック単位で有効化するには，ブロックを `only ketjs on`, `only ketjs off` ではさむ。

【例】KeTJS だけで入力窓を作る。

```
str="x^2";
//str=Textedit(50); only ketjs
Plotdata("1",str,"x");
```

注）`Textedit` は KeTJS で入力窓からの入力を得るコマンド

⇒ 関数一覧

6.4　KeTJS のコマンド

関数　Ptpos(幾何点)

機能　幾何点の現在（直前）座標を返す。

説明　幾何点を制御可能範囲外に移動した場合に保持されるもとの座標

【例】`Ptpos(A)`

関数　Ketcindyjsdata(変数名と値のリスト)

機能　KeTJS ファイルの script ”csinit”の最後にデータを書き込む

説明　Maxima の返り値など KeTJS では得られないデータを使えるようにする。

【例】
```
Mxfun("1","integrate",["x*sin(x)","x"]);// no ketjs
Ketcindyjsdata(["mx1",mx1]);
Plotdata("1",mx1,"x");
```

関数　Ketcindyjsbody(prependlist,appendlist)

機能　KeTJS ファイルの body の最初と最後にスクリプトを追加する。

```
【例】Ketcindyjsbody(["<p,f10>__Title''],[]);
        =>      <p><font size="10">  Title</font></p>
```

関数　Animationparam(初期値，速度，範囲)

機能　アニメーションボタンのパラメータ値を取得する。

例　ss=Animationparam(0,1,[-60,60]);

説明

- ”Play”　パラメータ値が初期値にセットされ，変化が始まる。
- ”Stop”　パラメータ値が初期値にセットされ，変化が終わる。
- 速度　パラメータ値の変化速度（秒速）
- 範囲　パラメータ値が端点に達したら停止する。

関数　Textedit(識別番号)

機能　KeTJS で入力窓に入れた文字列を取得する。

説明　KeTJS での入力窓の作り方

(1)　CindyScreen の”f(x)”を選び，適当な初期値を入れて Evaluate を押す。

注）”=”だけを入れて，Setketcindyjs のオプションに

”Equal=”

を追加すると，KeTJS の入力窓は空欄になる。

(2)　”要素を動かす”に戻り，(1) を選んでインスペクタを開く。

(3)　識別番号を確認（修正）して，フォントサイズを変える。

【例】識別番号を 0 とする。

```
str="x^2"; //no ketjs
//str=Textedit(0); //only ketjs
Plotdata("1",str,"x");
```

関数　Movetojs(識別番号または要素名, 座標, フォントサイズ);

機能　KeTJS でテキストボタンの位置とフォントサイズを指定

例　Movetojs(0,[2,-4],15);

関数　Setplaybuttons(座標, フォントサイズ [, スペース増加量]);

機能　KeTJS で Play などのボタンの位置とフォントサイズを指定

例　Setplaybuttons([-3,-4.5],15,[1]);

注）Play,Pause,Rev,Stop の識別番号が 71,72,73,74 であることを確認しておく。

注）スペース増加量の単位は mm，リストで個々に指定することもできる。

⇒ 関数一覧

7　付録

7.1　用語解説

Cinderella で使っている用語に次のものがある。

インシデント	点が曲線（直線）上に乗っている状態を表す。 曲線上に点をとるとインシデントになり，ドラッグしたとき曲線上だけを動く。 インシデントの状態を変えるには，「点の取り付け/取り外し」ツールを使う。
幾何要素	Cinderella の作図ツールで作図した点や直線などの要素
インスペクタ	幾何要素の大きさや色などの属性を管理するウィンドウ。
幾何点	幾何要素としての点。マウスドラッグで動かすことができる。 Cindyscript や KeTCindy のスクリプトで取った点は幾何要素にならないことがある。
自由点	マウスドラッグで任意に動かすことのできる点。
固定点	マウスドラッグで移動することのできない点 2 曲線の交点などではない単独の点の場合，インスペクタで点を固定できる。
スナップ	マウスポイントが格子点の近くに来ると格子点上にぴったり移動する。 Cinderella の画面の下方ツールのうち，磁石アイコンによりこのモードになる。

7.2　Cinderella の作図ツール

動かすモードにする	：幾何要素を選択して動かす。これが標準状態
点を加える	：クリックして点を作る
直線を加える	：2 点間をドラッグする
線分を加える	：2 点間をドラッグする
中点を加える	：2 点間をドラッグする
交点を加える	：2 曲線を順にクリック
平行線を加える	：直線上から通る点へドラッグ
垂線を加える	：直線上から通る点へドラッグ
角の二等分線を加える	：2 直線を順にクリック
円を加える	：中心から半径分ドラッグ
半径つき円を加える	：中心から半径分ドラッグ
焦点と通る点で決まる楕円	：焦点と通る点を順にクリック
焦点と通る点で決まる双曲線	：焦点と通る点を順にクリック

焦点と準線で決まる放物線　：焦点と準線を順にクリック

多角形を加える　：多角形の頂点を順にクリック

角に印をつける　：2 直線を順にクリック

角度を測る　：2 直線を順にクリック

選択した要素を消去する　：選択しておいてツールをクリック

点をまとめて選択する　：点がすべて選択される

線分をまとめて選択する　：線分がすべて選択される

画面ツール（下のツールバー）

原点を移動する　：画面上の任意の位置でドラッグする

矩形領域を画面サイズに拡大　：ドラッグしてできる矩形で領域を選択する

画面を矩形領域サイズに縮小　：ドラッグしてで切る矩形で領域を選択する

格子点にスナップする　：軸と方眼を表示しスナップモードにする

グリッドを粗く / 細かくする

7.3 色名とカラーコード一覧

name	CMYK	Color	name	CMYK	Color
greenyellow	[0.15,0,0.69,0]		royalpurple	[0.75,0.9,0,0]	
yellow	[0,0,1,0]		blueviolet	[0.86,0.91,0,0.04]	
goldenrod	[0,0.1,0.84,0]		periwinkle	[0.57,0.55,0,0]	
dandelion	[0,0.29,0.84,0]		cadetblue	[0.62,0.57,0.23,0]	
apricot	[0,0.32,0.52,0]		cornflowerblue	[0.65,0.13,0,0]	
peach	[0,0.5,0.7,0]		midnightblue	[0.98,0.13,0,0.43]	
melon	[0,0.46,0.5,0]		navyblue	[0.94,0.54,0,0]	
yelloworange	[0,0.42,1,0]		royalblue	[1,0.5,0,0]	
orange	[0,0.61,0.87,0]		blue	[1,1,0,0]	
burntorange	[0,0.51,1,0]		cerulean	[0.94,0.11,0,0]	
bittersweet	[0,0.75,1,0.24]		cyan	[1,0,0,0]	
redorange	[0,0.77,0.87,0]		processblue	[0.96,0,0,0]	
mahogany	[0,0.85,0.87,0.35]		skyblue	[0.62,0,0.12,0]	
maroon	[0,0.87,0.68,0.32]		turquoise	[0.85,0,0.2,0]	
brickred	[0,0.89,0.94,0.28]		tealblue	[0.86,0,0.34,0.02]	
red	[0,1,1,0]		aquamarine	[0.82,0,0.3,0]	
orangered	[0,1,0.5,0]		bluegreen	[0.85,0,0.33,0]	
rubinered	[0,1,0.13,0]		emerald	[1,0,0.5,0]	
wildstrawberry	[0,0.96,0.39,0]		janglegreen	[0.99,0,0.52,0]	
salmon	[0,0.53,0.38,0]		seagreen	[0.69,0,0.5,0]	
carnationpink	[0,0.63,0,0]		green	[1,0,1,0]	
magenta	[0,1,0,0]		forestgreen	[0.91,0,0.88,0.12]	
violetred	[0,0.81,0,0]		pinegreen	[0.92,0,0.59,0.25]	
rhodamine	[0,0.82,0,0]		limegreen	[0.5,0,1,0]	
mulberry	[0.34,0.9,0,0.02]		yellowgreen	[0.44,0,0.74,0]	
redviolet	[0.07,0.9,0,0.34]		springgreen	[0.26,0,0.76,0]	
fuchsia	[0.47,0.91,0,0.08]		olivegreen	[0.64,0,0.95,0.4]	
lavender	[0,0.48,0,0]		rawsienna	[0,0.72,1,0.45]	
thistle	[0.12,0.59,0,0]		sepia	[0,0.83,1,0.7]	
orchid	[0.32,0.64,0,0]		brown	[0,0.81,1,0.6]	
darkorchid	[0.4,0.8,0.2,0]		tan	[0.14,0.42,0.56,0]	
purple	[0.45,0.86,0,0]		gray	[0,0,0,0.5]	
plum	[0.5,1,0,0]		black	[0,0,0,1]	
violet	[0.79,0.88,0,0]		white	[0,0,0,0]	

注）lightgray [0,0,0,0.15], offwhite [0,0,0,0.3], cindycolor [0.66,0,69,0.71] を追加

7.4 点の作図についての比較表

関数	戻り値を使う	描画する	幾何点を作る	TeX に出力
Pointdata	-	○	-	○
Putpoint	-	-	○	-
Putintersect	-	-	○	-
PutonCurve	-	-	○	-
PutonLine	-	-	○	-
PutonSeg	-	-	○	-
Reflectpoint	○	-	-	-
Rotatepoint	○	-	-	-
Scalepoint	○	-	-	-
Translatepoint	○	-	-	-
Pointdata3d	-	○	-	○
Putpoint3d	-	-	○	-
Intersectcrvsf	△	-	○	-
IntersectsgpL	-	○	○	-
Invparapt	○	-	-	-
Parapt	○	-	-	-
Perpplane	-	○	○	-
Perppt	-	○	○	-
Pointdata3d	-	○	-	○
PutonCurve3d	-	-	○	-
PutonSeg3d	-	-	○	-
Reflectpoint3d	○	-	-	-
Rotatepoint3d	○	-	-	-
Scalepoint3d	○	-	-	-
Translatepoint3d	○	-	-	-

注) Intersectcrvsf は戻り値ではなく，プロットデータを使う。

8 関数一覧

【目次】に戻る

【設定・定義】

Addax(0/1)	座標軸を描くかどうかを定める
Addpackage(package)	プレビュー用のパッケージを追加
Changework(パス)	作業ディレクトリを変更する
Deffun(関数名 , 定義 list)	関数を定義する
Definecolor(色名 , 定義 list)	ユーザー定義色の設定
Defvar(文字列)	変数を定義する
Drwxy()	座標軸を先に描く
Fontsize(記号)	フォントサイズを設定する
Ketinit(options)	KETCindy を初期化する
Initglist	ketlib スロットの描画データを追加する
Ptsize(数)	表示する点の大きさを設定する
Setarrow(size,angle,pos,cut,style	矢線の形状を設定する
Setax(list)	座標軸の書式を設定する
Setcolor(color,options)	Windispg での描画色を設定する
Setfiles(filename)	出力するファイル名を設定する
Setparent(filename)	Parent で出力するファイル名を設定する
Setmarklen(数)	軸の目盛の長さを設定する
Setorigin(座標)	表示する座標軸の原点の位置を設定する
Setpen(数)	線の太さを設定する
Setpt(数)	表示する点の大きさを設定する
Setscaling(数)	縦方向の倍率を設定する
Setunitlen(数)	単位長を設定する
Setwindow()	描画領域を設定する
Usegraphics()	グラフィクスパッケージを"pict2e" に変更する

【描画】

Drawfigures(name ,figlist,optionlist)	複数のデータのスタイルをリストで与えて描画する
Anglemark(点 list, options)	角の印を入れる
Setarrow(options)	矢線をスタイルを設定する
Arrowdata(name.[始点, 終点],options)	2 点間を結ぶ矢線を描く
Arrowhead(点, 方向,options)	点に矢じりだけを描く
Bezier(name, list,list,options)	単独のベジェ曲線を描く

Beziersmooth(name , list,options)	なめらかなベジェ曲線を描く。その 1
Beziersym(name , list,options)	なめらかなベジェ曲線を描く。その 2
Bowdata(点 list,options)	弓形を描く
Bspline(name，list，options)	2 次 B スプライン曲線を描く
Changestyle(PD list, options)	描画オプションを変更する
Circledata(name, 点 list,options)	円または正多角形を描く
CRspline(name，list，options)	単独の Catmull-Rom スプライン曲線を描く
Deqplot(name, 式, 変数名, 初期値,options])	微分方程式の解曲線を描く
Dotfilldata(name , 方向, PD , options)	領域に点を敷き詰める
Drawsegmark(name,list,options)	線分に印をつける
Ellipseplot(name,list,str,options)	楕円を描く
Enclosing(name , [位置，方向，数式])	複数の曲線から閉曲線を描く
Expr([座標, 位置, 文字列],options)	TEX 数式を書く
Exprrot(位置，向き，文字列)	傾いた TEX 数式を書く
Fourierseries(name, 係数, 周期, 項数)	フーリエ級数を描く
Framedata(name , list)	矩形を描く
Hatchdata(name , 方向, PD , options)	領域に斜線を引く
Htickmark([横座標 , 方向 , 文字])	横軸に目盛りを描く
Hyperbolaplot(name,list,str,options)	双曲線を描く
Implicitplot(name,str,,str,str,options)	陰関数のグラフを描く
Invert(PD)	プロットデータの点を逆順にする（reverse と同じ）
Joincrvs（name, PDlist, options)	2 つのプロットデータをつなげたデータを作る
Letter([座標, 位置, 文字列],options)	文字列を表示する
Letterrot(座標, 方向, 移動量, 文字列)	文字列を回転して表示する
Lineplot(name,2 点の list,options)	2 点を結ぶ直線を描く
Listplot(name, 点の list,options)	点を線分で結ぶ
Mkbeziercrv(name,list,options)	作図した点を使ってベジェ曲線を描く
Mkbezierptcrv(list，options)	制御点を自動配置してベジェ曲線を描く
Mkcircles()	幾何円のすべての PD を作成する
Mksegments()	幾何線分のすべての PD を作成する
Ospline(list，list，options)	大島のスプライン曲線を描く
Ovaldata(name, 点 list,options)	角を丸くした矩形を描く
Parabolaplot(name,list,str,options)	放物線を描く
Paramark(点 list,options)	角の印を入れる
Paramplot(name, 式, 変数と定義域,options)	媒介変数で表された曲線を描く
Polarplot(name, 式, 変数と定義域,options)	極座標表示の曲線を描く
Partcrv(name, 点 1, 点 2,PD)	部分曲線を描く

Periodfun(定義式, 周期,options)	周期関数のグラフを描く
Plotdata(name, 式, 変数と定義域,options)	関数のグラフを描く
Pointdata(name, 点 list,options)	点データを作る
Polygonplot(name, 点 list, 整数,options)	正多角形を描く
Putintersect(点名,PD1,PD2)	2 曲線の交点を作る
Putoncurve(name,PD, 初期値)	曲線上に点を作る
Putonline(点名, 座標 1, 座標 2)	直線上に点を作る
Putonseg(点名, 座標 1, 座標 2)	線分上に点を作る
Putpoint(点名, 座標 1, 座標 2)	点を作る
Reflectdata(name,PD, 点 list,options)	プロットデータの鏡映を作成
Reflectpoint(点, 対称点/対称軸)	点の鏡映を作成
Rotatedata(name,PD, 角度, 中心,options)	プロットデータを回転する
Rotatepoint(点, 角度, 中心)	点の位置を回転する
Rulerscale(点,list,list)	目盛を打つ
Scaledata(name,PD,x,y, 中心,options)	点を拡大・縮小する
Scalepoint(点, 比率ベクトル, 中心)	点の位置を拡大・縮小する
Segmark(name,list,options)	線分に印をつける
Shade（PDlist , 数)	閉曲線の内部にシェードをかける
Tangentplot(name,PD, 位置)	曲線の接線を引く
Translatedata(name,PD, ベクトル,options)	プロットデータを平行移動する
Translatepoint(点, ベクトル)	点を平行移動する
Vtickmark([横座標 , 方向 , 文字])	縦軸に目盛りを描く

【作表】

Changetablestyle(罫線 list, options)	Table の罫線の描画オプションを変更する。
Findcell(列番号, 行番号)	セルの情報 list を返す
Putcell (列番号, 行番号, 位置, 文字)	セルに文字列を入れる
Putcellexpr (列番号, 行番号, 位置, 文字)	セルに数式を入れる
Putcol (列番号, 位置，文字列 list)	１列に順に文字を書き入れる
Putcolexpr (列番号, 位置，文字列 list)	１列に順に TEX 書式の文字を書き入れる
Putrow (行番号, 位置，文字列 list)	１行に順に文字を書き入れる
Putrowexpr (行番号, 位置，文字列 list)	１行に順に TEX 書式の文字を書き入れる
Tabledata(”” , 縦横 , 除外 , options)	表の枠を作成する
Tabledatalight(”” , 縦横 , 除外 , options)	幾何点を持たない表の枠を作成する
Tgrid(セルラベル)	セル（格子点）の座標を返す
Tlistplot(セルラベル 1, セルラベル 2)	セルに斜線を引く

【値の取得と入出力】

Asin(real),Acos(real)	逆三角関数の値を返す

Crossprod(list,list)	ベクトルの外積を計算する
Derivative(関数式, 変数, 値)	関数の微分係数を求める
Dotprod(list,list)	ベクトルの内積を計算する
Extractdata(データ名, 属性)	ReadOutData で読み込んだデータに属性をつける。
Findarea(PD)	プロットデータで囲まれる部分の面積を求める
Findlength(PD)	プロットデータで描く曲線の長さを求める
Integrate(関数式, 変数, 範囲,options)	関数の定積分値を求める
Intersectcrvs(PD1,PD2)	プロットデータの交点の座標 list を返す
IntersectcrvsPp(PD1,PD2)	プロットデータの交点のパラメータ list を返す
Inversefun(関数式, 範囲, 値)	逆関数値を求める
Nearestpt(PD,PD)	2 曲線間の最も近い点を取得する
Nearestptcrv(点,PD)	点に一番近い曲線上の点を取得する
Numptcrv(PD)	曲線 PD の節点データの個数を取得する
ParamonCurve(PD,n,PtL)	PD 上にある点 P のデータを取得する
Pointoncrv(数,PD)	パラメータ値をもつプロットデータ上の点
Ptcrv(n,PD)	曲線 PD の n 番目の節点を取得する
Ptstart(PD)	プロットデータの始点・終点を取得する
ReadOutData(ファイル名)	外部データを PD として読み込む
Readcsv(name,filename,option)	csv ファイルを読む
Readlines(name,filename,option)	テキストファイルを 1 行ずつ読む
Sqr(real)	平方根を返す
Viewtex()	TEX のソースファイルを書き出す。引数なし
Workprocess()	作図の経過を取得する
WriteOutData()	PD データを書き出す

【その他】

Assign(文字列)	文字列中のある文字を値で置き換える
BBdata(ファイル名)	画像のサイズを求める
Cindyname()	作図しているファイル名を取得する。
Colorcode(文字 1，文字 2,color)	カラーコードの変換
Dqq(文字列)	文字列の前後に"をつける。
Factorial(n)	n の階乗を計算する。
Norm(ベクトル)	ベクトルの大きさを計算する。
Figpdf(option)	出力枠サイズの PDF を作る
Help(str)	コマンドヘルプを表示する
Helpkey(str)	キーワードで関数を検索する
Indexall(str1,str2)	文字列 str1 から str2 を検索しその位置をすべて返す
Op(n,list)	list または文字列から要素を抜き出す

Isptselected(点名)	点が選択されていれば true を返す
Ptselected(点名)	点が選択されていれば true を返す
Reparse(文字列（リスト）	評価して実数化する
Slider()	スライダを作る
Sprintf(実数, 長さ)	小数点以下の長さを固定した文字列に変換
Strsplit(文字列, 文字)	文字列を分解する。
Texcom(コード)	TEX のコードを書き出す
Textformat(数, 桁数)	小数点以下の桁数を指定して数値を文字列化する
Windispg()	定義されたプロットデータを描画面に描く
Fracform(数, 分母リスト)	簡易 TeX-like 書式の文字列を返す
Totexform(TeX-like 書式)	TeX 書式の文字列を返す
Tocindyform(TeX-like 書式	Cindy 書式の文字列を返す
【Rとの連携】	
Boxplot(名前, データ, 位置, 高さ,option)	箱ひげ図を描く
Rfun(変数名, コマンド, 引数, option)	R の 1 つコマンドを実行して結果を返す
CalcbyR(変数名, コマンド列, option)	R のコマンド列を実行して結果を返す
Histplot(name,data)	ヒストグラムを描く
PlotdataR(name, 式, 変数)	R の関数のグラフを描く
PlotdiscR(name, 式, 変数)	離散型のグラフを描く
Scatterplot(name,filename,option)	2 次元データを読み込み，散布図を描く
【Maxima との連携】	
CalcbyM(name,list,option)	Maxima のスクリプトを実行する
Mxbatch(list)	Maxima の外部スクリプト用コマンドを作る
Mxfun(name, 式,list,option)	Maxima の関数を実行する
Mxtex(num, 式)	式を TeX 書式にする
【Risa/Asir との連携】	
Asirfun(name, 式,list,option)	Risa/Asir の関数を実行する
CalcbyA(name,list,option)	Risa/Asir のスクリプトを実行する
【FriCAS との連携】	
CalcbyF(name,list,option)	FriCAS のスクリプトを実行する
Frfun(name, 式,list,option)	FriCAS の関数を実行する
【MeshLab との連携】	
Mkobjcmd(name, 式,option)	厚みを持たない曲面のコマンドを作成
Mkobjcrvcmd(name,PD,option)	空間曲線のコマンドを作成
Mkobjnrm(name, 式)	法線ベクトルのデータを作成
Mkobjplatecmd(name, 面データ,options)	面を描く
Mkobjpolycmd(name,PD,options)	多面体を描く

Mkobjsymbcmd(PD, 実数, 実数,vec,vec)	文字等のコマンドを作成
Mkobjthickcmd(name, 式)	厚みを持つ曲面のコマンドを作成
Mkviewobj(name,PD, options)	obj ファイルを作成
【表計算ソフトとの連携】	
Dispmat(list)	list の内容を行列型にコンソールに表示する
Tab2list(str, option)	str の内容を list に変換する
Writecsv(namelist,data,option)	data の内容を csv ファイルに書き出す
【アニメーション】	
Setpata(str)	パラパラ動画のタイトル指定
【スライド】	
Settitle(タイトル list,options)	スライドのタイトル設定
【KeTCindy3D 設定・定義】	
Ketinit3d()	KeTCindy3D の使用宣言
Isangle	角度スライダが選択されているか
Start3d()	3D の開始
Setangle	回転角の設定
Getangle	回転角の取得
Startsurf	曲面描画の初期化
【KeTCindy3D 描画】	
Bezier3d(name, list, list)	空間ベジェ曲線を描く
Changestyle3d(list, list)	3d プロットデータの属性を変更
Concatobj(list,option)	いくつかの obj データを結合
Crvsfparadata(name,PD,PD2, 式,opt,opt)	曲線の曲面による陰線処理
Datalist2d()	画面に描かれているすべてのプロットデータ
Datalist3d()	画面に描かれているすべてのプロットデータ
Dist3d(点名, 点名)	空間の 2 点の距離
Embed(name,PD, 式)	埋め込みデータ作成
ExeccmdC(name,options1,options2)	C 言語で命令実行
Expr3D([座標, 位置, 文字列],options)	文字列を表示する
Intersectcrvsf(name,PD, 式)	曲線と曲面の交点を求める
IntersectsgpL(点名, 線分, 面, 描画方法)	空間の直線と平面の交点
Invparapt(座標,PD)	描画面座標に対応する曲線上の座標
Letter3D([座標, 位置, 文字列],options)	文字列を表示する
Mkbezierptcrv3d(点 list)	制御点を自動的にとる空間ベジェ曲線
Nohiddenbyfaces(name,PD,PD,opt1,opt2)	多面体と空間曲線を陰線処理
Parapt(座標)	点の投影面での座標
Partcrv3d(name, 始点, 終点,PD)	曲線 PD の部分曲線を作る

Perpplane(点名, 点, ベクトル,option)	点を通り垂直な平面上の基準点
Perppt(点名, 点, 点 list,option)	平面に下ろした垂線の足
Phparadata(name,name2,options)	多面体を陰線処理して描く
Pointdata3d(名前, 点 list,options)	空間点のデータを作成する
Putaxes3d([x,y,z])	軸上に幾何点をとる
Putoncurve3d(点名,PD)	空間曲線上に点をとる
Putonseg3d(点名, 点 1, 点 2)	線分上に点をとる
Putpoint3d(list,option)	空間点をとる
Readobj(ファイル名)	obj ファイルを読み込む
Reflectdata3d(点,PDlist,list,options)	PD を鏡映
Reflectpoint3d(点,list)	点を鏡映
Rotatedata3d(name,PD,vec, 角度, 点)	プロットデータを回転
Rotatepoint(点 , vec , 角度 , 点)	点を回転
Scaledata3d(点, vec, 中心)	プロットデータを拡大/縮小
Scalepoint3d(点, vec, 中心)	点の位置を拡大/縮小
Sf3data(name, list,options)	陰線処理なしの空間曲面を描く
Sfbdparadata(name, 式,options)	曲面を陰線処理して描く
Sfcutparadatacdy(name, 面, 曲面,options)	平面と曲面の交線を求める。
Skeletonparadata(name,PD,PD,options)	スケルトン処理のデータ作成
Spacecurve(name, 式, 定義域,options)	空間曲線のデータ作成
Spaceline(name,list)	空間の折線データ作成
Translatedata3d(name,PD, 平行移動量)	空間プロットデータを平行移動
Translatepoint3d(座標, 平行移動量)	空間点を平行移動
VertexEdgeFace(面データ,option)	頂点と面から辺を求め，辺を描く
Wireparadata(name,PD, 式,int,int,opt,opt)	曲面のワイヤフレームを陰線処理
Xyzax3data(name, 文字, 文字, 文字,options)	座標軸の表示
Xyzcoord(P.x,P.y,Pz.y)	主副画面で決まる点の座標

【KeTJS】

Ptpos(幾何点)	幾何点の現在（直前）座標
Setketcindyjs(options)	KeTJS の設定
Ketcindyjsdata(変数名と値のリスト)	script ”csinit”にデータを書き込む
Ketcindyjsbody(prependlist,appendlist)	body の最初と最後にスクリプトを追加
Animationparam(初期値，速度，範囲)	アニメーションボタンのパラメータ値を取得
Textedit(識別番号)	KeTJS で入力窓に入れた文字列を取得
Movetojs(要素名, 座標, フォントサイズ)	Text ボタンの位置とフォントサイズを設定
Setplaybuttons(座標, サイズ [, スペース])	Play などのボタンの位置とフォントサイズを設定