KETCindy Command Reference

KETCindy Project Team

December 4, 2019

- ver.3.2 -

Contents

1 Plane figure

1.1 Setting and Defining

1.1.1 Setting environment

Ketinit

Usage Ketinit();

Description Generic function to initialize KETCindy.

Examples

`Ketinit();` The work sub folder is set to `"fig"` in the folder of the cindy file.
`Ketinit("");` The work folder is set to the folder of the cindy file.

Details

This function should be written at the first line on Draw slot page. In case of space figure (KeTCindy's 3D-mode), write it in the initialization slot page `ketlib`.

⇒Command List

Initglist

Usage Initglist(), Setglist(), Addglist()

Description Add the list generated in `"ketlib"` slot to that of `"ketlib"` slot.

Examples

```
Initglist(); // in ketlib slot
Implicitplot(''1'',fun,rng);
Setglist();

Ketinit(); // in figures slot
Addglist();
```

⇒Command List

Setfiles

Usage Setfiles(filename)

Description Generic function to set the name of texfile.

Details Default file name is working Cinderella file name.

Examples

If working Cinderella file name is "triangle.cdy" then default files name are "triangle.tex".

By `Setfiles("grav");` output files name are "grav.tex".

⇒Command List

Setparent

Usage Setparent(filename)

Description Generic function to set the name of texfile by using the Parent push button.

Details There is no default file name when we use the Figpdf() function and the Parent push button, so we have to define the name of output texfile.

Examples

If working Cinderella file name is ”triangle.cdy” , by `Setparent("grav");` output files name are ”triangle.tex” and ”grav.tex”. PDF name is ”grav.pdf”.

⇒Command List

Changework

Usage Changework(name of pass)

Description Generic function to change the working directory(folder).

Default working directory is ”fig”.

⇒Command List

Addpackage

Usage Addpackage(list of style fies)

Description Generic function to add packages of TEX to the main file for previewing.

Details Basically, `ketpic,ketlayer,amsmath,amssymb,graphicx,color` are used.

Examples

```
Addpackage(["[dvipdfmx]{media9}","[dvipdfmx]{animate}","ketmedia"]);
```

⇒Command List

Usegraphics

Usage Usegraphics(”pict2e”)

Description This function changes the graphics package to `"pict2e"`.

Details The defaut package is `"tpic"`.

Examples

```
Usegraphics("pict2e");
```

⇒Command List

1.1.2 Drawing and defining

Addax

Usage Addax(1/0);

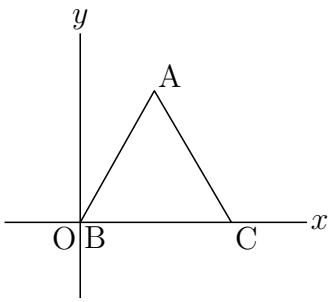
Description Generic function to decide axis are drawn or not.

Details If argument is 1, axis are output in the TeX file (default) but there are no axis on the Euclidean view.

Examples

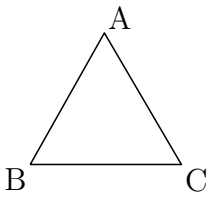
To draw a triangle.

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



Hide coordinate axes.

```
Addax(0);
Listplot([B,A,C,B]);
Letter([A,"ne","A",B,"sw","B",C,"se","C"]);
```



⇒Command List

Setax

Usage Setax(a list of parameters);

Description Generic function to set the style of axis.

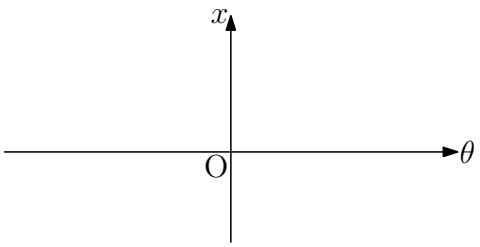
Details Parameters are:

1. Style of axis (”l” ; line(default), ”a” : arrow)
2. Name of horizontal ax (default is x)
3. Posion of horizontal name (default is ”e”)
4. Name of horizontal ax (default is y)
5. Posion of horizontal name (default is ”n”)

6. Name of origin (default is O)
7. Position of origin (default is "sw")
8. Linestyle
9. Color of axes
10. Color of labels

Examples

```
Setax(["a","","","","","","nw"]);
Setax(["","","","","","","","do","red"]);
Setax([7,"nw"]);
Setax(["a","\theta","","x","w"]);
```



⇒Command List

Drwxy

Usage Drwxy(), Drwxy(options)

Description Generic function to draw axis in the TEX figure.

Details

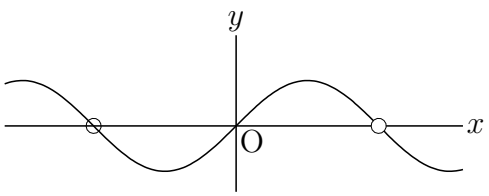
By default the axes are drawn last. Use this function when axis should be drawn in the middle of commands. There are no axis on the Euclidean view.

Options is a list of ["Origin=", "Xrng=", "Yrng="].

Examples

To draw a point in the void mode.

```
Setax([7,"se"]);
Setpt(8);
Pointdata("1",[[-pi,0]],["Inside=0"]);
Drwxy();
Plotdata("1","sin(x)","x",["dr","Num=300"]);
Pointdata("2",[[pi,0]],["Inside=0"]);
```



⇒Command List

Definecolor

Usage Definecolor(name of a color,colorcode)

Description Generic function to define the name of colorcode in the TEX figure.

Examples

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

⇒Command List

Setcolor

Usage Setcolor(color,options)

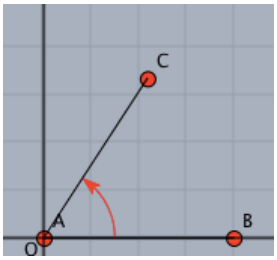
Description Generic function to set the color of figures and characters in the TEX figure.

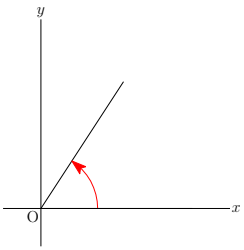
Examples

```
C.xy=B.xy|/|C.xy|*C.xy;|
Listplot([B,A,C]);
Setcolor([1,0,0]);
Anglemark("1",[B,A,C],[3]); //size=3
Arrowhead(1,"ag1",[2]); //position=1,size=2
```

Remark You can also use color option in each command of drawing.

```
Arrowhead(1,"ag1",["Color=[1,0,0]"]);
```





Refer to Color table on Appendix.

⇒Command List

Deffun

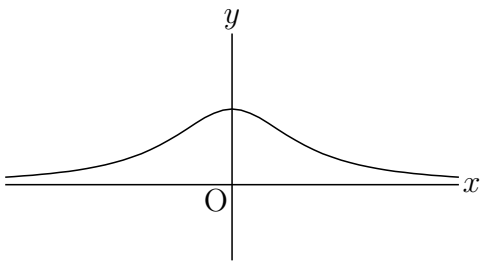
Usage Deffun(name of a function,a list of commands);

Description Generic function to define a function common to both Cindyscript and R.

Examples

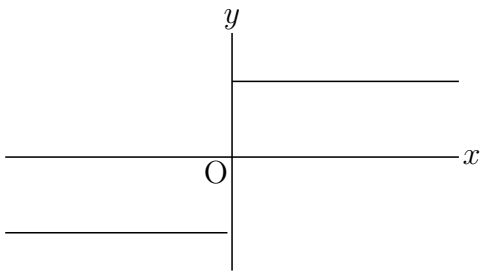
$$f(x) = \frac{1}{x^2+1}$$

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



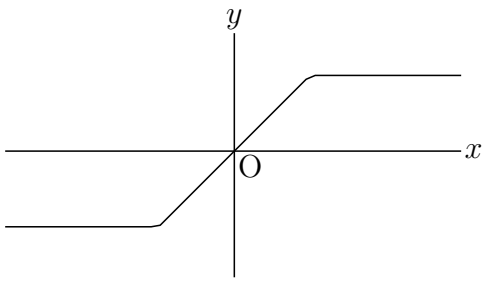
$$f(x) = \begin{cases} 1(x \geq 0) \\ -1(x < 0) \end{cases}$$

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



"If" command can be nesting.

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



⇒Command List

Defvar

Usage Defvar([name,value,...]);

Description Generic function to define variables common to both Cindyscript and R.

Examples

```
Defvar(["const",3]);    //const=3;
Defvar(["a",3,"b",1]); //a=3;b=1;
```

⇒Command List

Fontsize

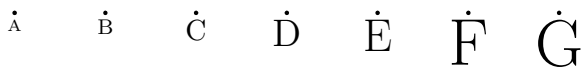
Usage Fontsize(size symbol)

Description Generic function to define the font size in the TEX figure.

Details The symbol is `"t", "ss", "f", "s", "n", "la", "La", "LA", "h", "H"`.

Examples

```
Ptsize(2);
Drawpoint([A,B,C,D,E,F,G]);
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"]);
```



⇒Command List

Ptsize

Usage Ptsize(ratio);

Description Generic funtion to set the size of points.

Details This function is same as Setpt().

⇒Command List

Setpt

Usage Setpt(ratio);

Description Generic funtion to set the size of points.

Details ”ratio” is the ratio from the standard size.
Size can be change as a option of ”Pointdata”.

Examples

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

⇒Command List

Setmarklen

Usage Setmarklen(real number)

Description Generic function to set the length of tickmarks on the axsis.

Details Set the length of tickmarks on the axsis when we use the functions Htickmark() and Vtickmark().

⇒Command List

Setorigin

Usage Setorigin(coordinate)

Description Generic function to set or transtate the coordinate of apparent origin.

Examples

```
Setorigin([3,2]);
```
if A is identification name of some point, `Setorigin(A);`

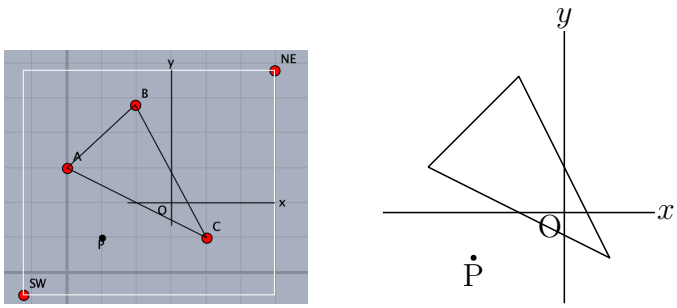
Remark Coordinate system is not changed as the following examples.

Examples

The coordinate of apparent origin is $(3, 2)$ but we use the original coordinate system in the script.

```
Setorigin([3,2]);
Listplot([A,B,C,A]);
Ptsize(3);
Drawpoint([1,1]);
Letter([[1,1],"s2","P"]);
```

Left figure is Euclidean view, right figure is the result of TEX.



⇒Command List

Setpen

Usage Setpen(real number)

Description Generic function to set the thickness of lines.

⇒Command List

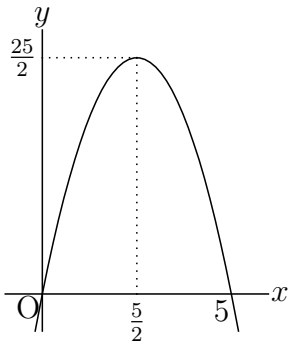
Setscaling

Usage Setscaling(scale)

Description Generic function to set the scale of vertical direction. Argument is real number or list. If it is a real number, vertical scaling. If the list [a, b], scaling a in the horizontal direction and b in the vertical direction.

Examples

```
Setscaling(0.5);
Plotdata("1","-2*x^2+10*x","x");
p1=[5/2,0]; p2=[5/2,25/2]; p3=[0,25/2];
Listplot(``1",[p1,p2,p3],["da"]);
Expr([[5,0],"s2w","5",p3,"w2","\frac{25}{2}",p1,"s4","\frac{5}{2}"]);
```



⇒Command List

Setunitlen

Usage Setunitlen(scale);

Description Generic function to set the scale of unit length. (default is 1cm)
It is recommended to put this function to the beginning of a script.

Examples

```
Setunitlen("8mm");
```

⇒Command List

Setwindow

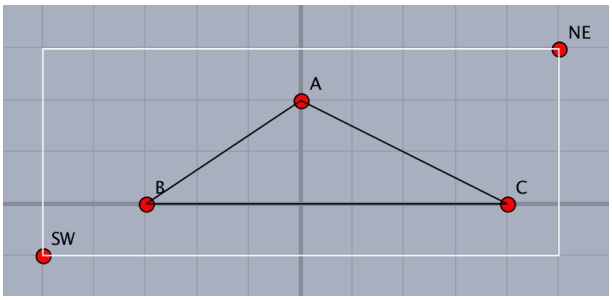
Usage Setwindow(range of x , range of y);

Description Generic function to set a output area on a Euclidean view.

Details A output area is normally specified by a rectangle with SW and NE as diagonal two vertices. (i.e range of x is [XMIN,XMAX] and range of y is [YMIN,YMAX]) By dragging these two vertices on a Euclidean view, we can change the output area. This command is used to set the window manually and fix it.

Examples

```
Setwindow([-5,5],[-1,3]);
```



⇒Command List

1.2 Commands for Drawing

1.2.1 Options of drawing command

Options of drawing command

Line type

”dr, n”	solid line n : thickness
”da(,m,n)”	broken line m : length, n : gap m,n option are not draw Euclidean view and can be omitted.
”id(,m,n)”	broken line start gap.
”do(,m,n)”	dot line m : gap, n : thickness

Color

”Color=col” : col: RGB or CMYK or color name

Num

”Num=n” : Number of divisions of plotting data

Example

```
Plotdata("1","x^2","x",["Color=red","do,2,3","Num=100"]);
```

Output

”notex”	not output to TEX.
”nodisp”	not output to TEXand Euclidean view but make PD.
”Size=n”	size of point and thin of line
”Num=n”	Number of PD

Direction

The direction is represented by e(east : right), w(west: left), n(north : upper), s(south:lower) and c(center). The distance from the specified position can also be given as a numerical value. For example, ”e2” and ”e3” are placed twice and three times of the slightly unit distance away from ”e”, respectively.

Others

In addition, there are options specific to each function.

⇒Command List

1.2.2 Point, line

Pointdata

Usage Pointdata(name, point list, options)

Description Generic function to make a point data.

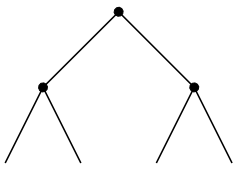
Detailse Options are ”Size=”, ”Color=”, ”Inside=”,”notex/nodisp”.

Examples

```
Pointdata("1",[[1,2],[-2,3]]);  // make 2 points (1,2),(-2,3)
Pointdata("2",[A,B]);           // A and B are draw by drawing tool.
Pointdata("3",A,["size=4"]);    // size of point A is 4.
Pointdata("4",[A,B],["Inside=0"]);        // white circles
Pointdata("5",[[3,4],[5,6]],["notex"]);  //not draw in the TEXfile.
Pointdata("6",[[3,4],[5,6]],["nodisp"]); //not draw TEXfile and Euclidean view.
```

Draw node of tree.

```
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]]);
```



Remark Comparative chart of drawing of points

⇒Command List

Putpoint

Usage Putpoint(name of point, A, B);

Description Generic function to put a point.

Details Put a point at A. If there already exists a point at A, it is put at B.

Examples

```
Putpoint("P",[1,1]);              // P is fixed point.
Putpoint("P",[1,1],[P.x,P.y]);    // for a movable point.
```

Remark Comparative chart of drawing of points

⇒Command List

Putintersect

Usage Putintersect(name of point, PD1 ,PD2, [Number])

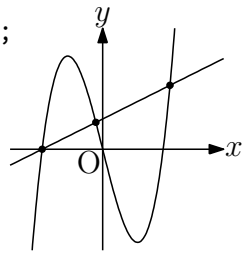
Description Generic function to make a intersection point of two curves.

Details PD1 and PD2 are plotting data names of two curves. Only one intersection point exists inside the drawing range, we have the point. If there exist many intersection points inside the drawing range then we have the list of coordinates for the points and the message:"Choose point number" on the console. The "Number" argument is this point number. We have to use the function Pointdata() when we need the figure of points in the output TEXfile.

Examples

In the following example We have three intersection points for a cubic curve and a line.

```
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"]);
```



If there exist no such points, we have the message:"No intersect point" on the console.

⇒Command List

PutonCurve

Usage PutonCurve(name of point, PD, options);

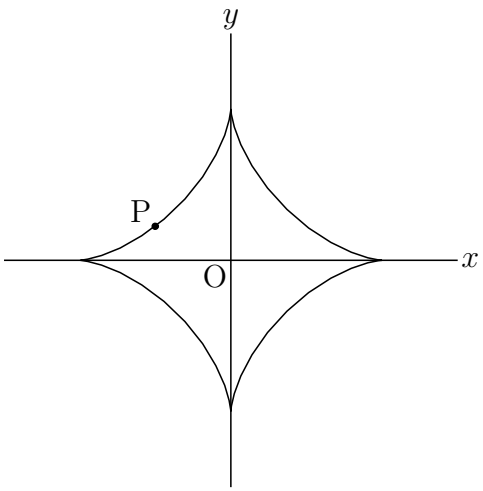
Description Generic function to put a point on the curve.

Details Put a point on the curve of PD.

Examples

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

This Point P on the asteroid can be move along the curve on the Euclidean view.



⇒Command List

PutonLine

Usage PutonLine(name of point, A, B);

Description Generic function to put a point on the line.

Details Put a point on the straight line through the two points A and B.

Examples

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

⇒Command List

PutonSeg

Usage PutonSeg(name of point, A, B);

Description Generic function to put a point on the segment.

Details Put a point on the line segment AB.

Examples

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

⇒Command List

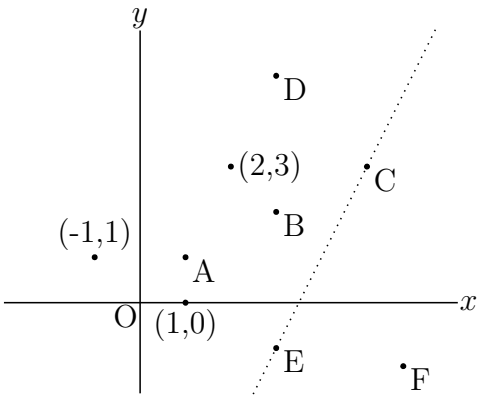
Reflectpoint

Usage Reflectpoint(a point, center or axis of symmetry);

Description Generic function do return the reflect point.

Examples

```
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"]);
```



⇒Command List

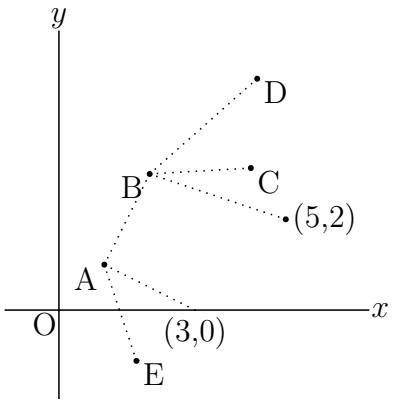
Rotatepoint

Usage Rotatepoint(point, angle(degree), center);

Description Generic function to rotate a point.

Examples

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



⇒Command List

Scalepoint

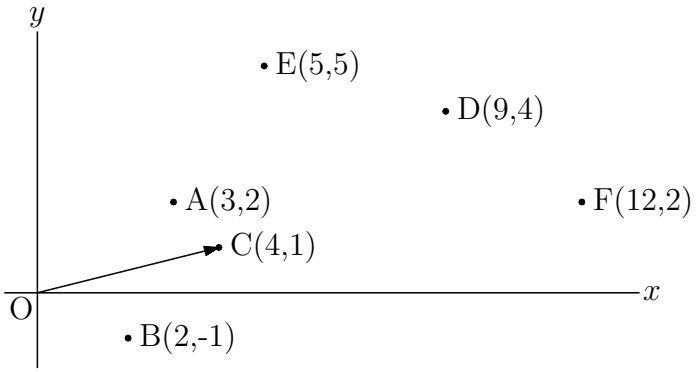
Usage Scalepoint(point, scale, center):

Description Generic function to scale a point.

Examples

```
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+")"]);
```



⇒Command List

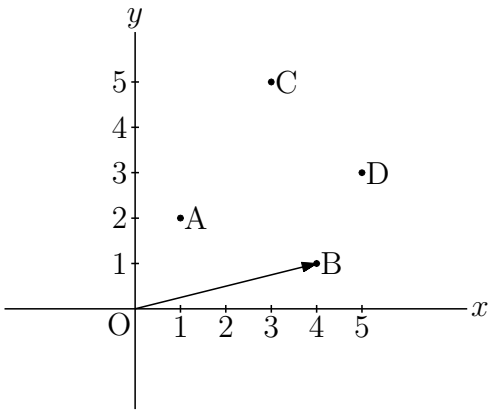
Translatepoint

Usage Translatepoint(point, vector);

Description Generic function to translate a point.

Examples

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



⇒Command List

Setarrow

Usage Setarrow(size(1), angle(18), position(1), cut(0.2)

Description set the style of arrow

⇒Command List

Arrowdata

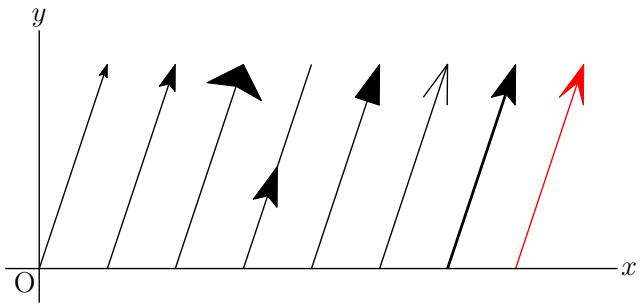
Usage Arrowdata(name,[starting point, ending point] , options)

Description draw an arrow line between two points.

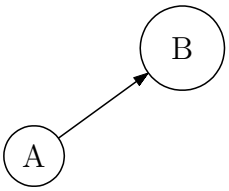
Options : Those of Setarrow and "Line=n(y)"(with line),"Cutend="(trimming),"Color="

Examples

```
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]);
Arrowdata("6",[[5,0],[6,3]],[3,"Line=y"]);
Arrowdata("7",[[6,0],[7,3]],[3,"dr,2"]);
Arrowdata("8",[[7,0],[8,3]],[3,1,1,0.5,"Color=red"]);
```




```
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"]);
```



⇒Command List

Arrowhead

Usage (1) Arrowhead(point, direction , options)
(2) Arrowhead(point/positon ratio, PD(curve), options)

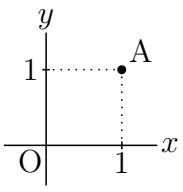
Description (1) draw an arrowhead with specified direction at a point.
(2) draw an arrowhead on a PD(curve)

Options are the same as Arrowdata

Position ratio of a point on the curve is from 0 to 1.

Examples

Let A=[1,1].

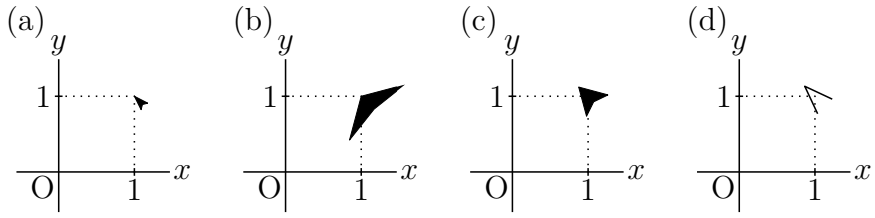


(a) `Arrowhead(A,[-1,1]);`

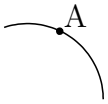
(b) `Arrowhead([1,1],[-1,1],[2,60]);`

(c) `Arrowhead(A,[-1,1],[2,30,0.5);`

(d) `Arrowhead([1,1],[-1,1],[2,20,0.5,"Line=y"]);`



Let A whose position ratio is 0.6 be on the curve cr1.

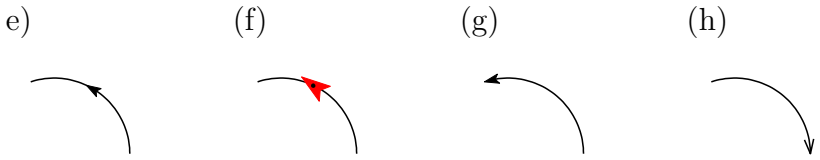


(e) `Arrowhead(A,"cr1");`

(f) `Arrowhead(0.6,"cr1",[2,1,0.5,"Color=red"]);`

(g) `Arrowhead(1,"cr1");`

(h) `Arrowhead(1,"Invert(cr1)",["Line=y"]);`



⇒Command List

Lineplot

Usage Lineplot(name, [A, B], options)

Description Draw the straight line through the two points A, B.

Details The list of two points is given by the coordinates or the geometric elements.
If the list of points is given by geometric elements, "name" can be omitted.
options : "+" means drawing a half straight line.
Both the line type and "+" can be specified as a list.

Example

Draw a straight line connecting the coordinates.

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

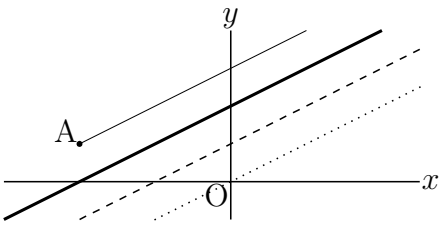
Draw the two points A, B in the Cinderella main screen and draw a straight line AB.

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

Some examples of options.

```
Lineplot([A,B],["dr,0.5","+"]); // Draw a half line with A as the end point.
Lineplot([C,D],["dr,2"]);  // Draw the straight line CD with double thickness.
Lineplot([E,F],["da"]);    // Draw the straight line EF as a broken line.
Lineplot([G,H],["do"]);    // Draw the straight line GH as a dotted line.
```

The results are shown in order from the top left of the next figure.



⇒Command List

Listplot

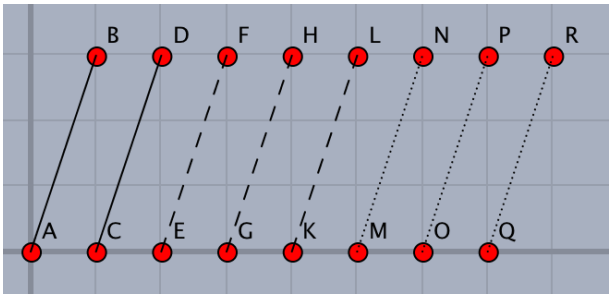
Usage Listplot(name, a list of points, options)

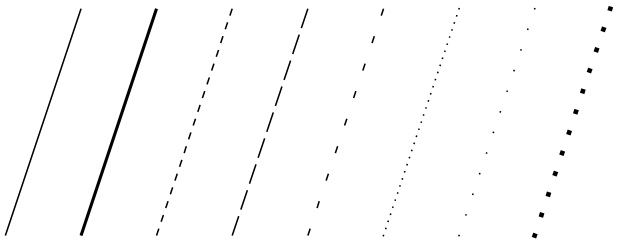
Description Connect points by line segments.

Details The list of two points is given by the names of the coordinates or the geometric elements.
If the list of points is given by geometric element names, the name of the plotting data can be omitted.

Example1 Line style

```
Listplot([A,B]);
Listplot([C,D],["dr,2"]);
Listplot([E,F],["da"]);
Listplot([G,H],["da,3,1"]);
Listplot([K,L],["da,1,3"]);
Listplot([M,N],["do"]);
Listplot([O,P],["do,3"]);
Listplot([Q,R],["do,3,3"]);
```

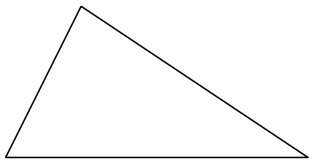




Example2 Draw a triangle.

Draw the triangle ABC or simply creating 3 points A, B, C with the Euclidean view.

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



The position of the points can be specified by coordinates. In this case "name" is necessary.

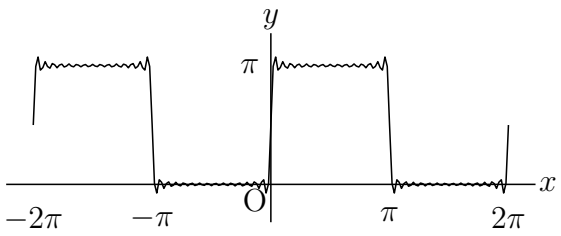
```
Listplot("1",[[0,0],[2,0],[1,2],[0,0]]);
```

Example3 Expansion of finite Fourier series

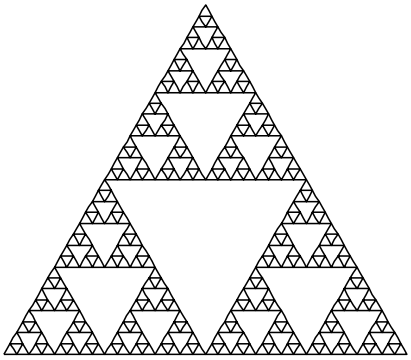
$$\frac{\pi}{2}+\sum_{n=0}^{30}\frac{1-(-1)^n}{n}\sin nx$$

The plotting data is a list of the coordinates of points. Therefore, define the function in Cindyscript as follows, create plotting data `pd` and pass it as argument.

```
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"]);
```



There is a limit on the length of the list, so it is impossible to use a long list or to use it many times. For example, in the Shellpinski gasket using Turtle Graphics, the next size is possible, but in the growth model of plants there are many branches so it can not be a big figure. We devise a script and divide it into lists of about 200.



⇒Command List

Mksegments

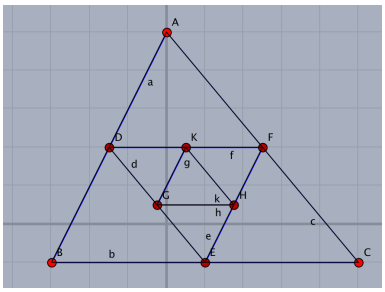
Usage Mksegments()

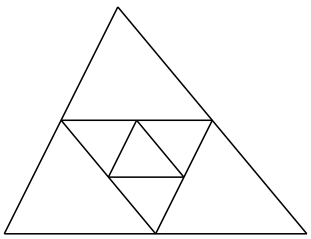
Description Create plotting data of all geometric segments.

Details All the line segments drawn by the "Add line segment" tool in the Euclidean view are used as plotting data as they are. For example, if the line segment AB is created, plotting data `sgAB` is created. After that, if you change the identification name of point B (for example to Q) in the inspector of the Euclidean view, the plotting data name is also changed. Even if the line segmen has already been drawn, it can be changed.

Example Examples of geometric progression

Draw a figure of a geometric progression that makes triangles by connecting the midpoints of each edge of a triangle one after another.
First draw the triangle ABC with the "Add line segment" tool in the Euclidean view.
Take the midpoint of each edge with the "Add midpoint" tool in the Euclidean view and connect the midpoints with the "Add line segment" tool in the Euclidean view.
Repeat this process. If you write `Mksegments();`, you can obtain the data of the figure at the completion of drawing, without writing `Listplot ([A, B, C]);`.





⇒Command List

Framedata

Usage Framedata(name,expr,options)

Description Generic function to draw a rectangle.

Details expr type1 : [center,lx,ly] : lx and ly are a half of the horizontal and vertical length.

expr type2 : [p1, p2] : if p1 and p2 are name of point, 1st argument can be omitted.

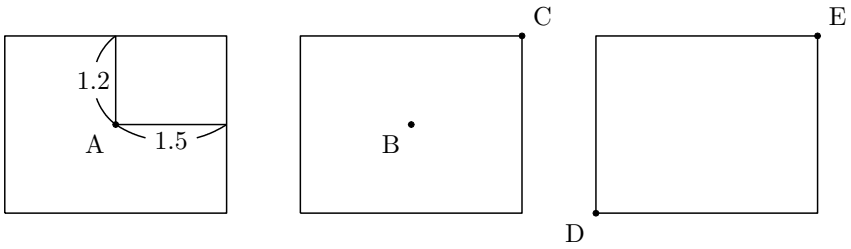
options : usual options and "center"/"corner" (type2).

If "center", p1 is center, p2 is apex of rectangle. (Default)

If "corner", p1 and p2 are diagonal point of rectangle.

Examples

```
Framedata("1");              // same as Framedata([SW,NE],["corner"]);
Framedata("2",[[0,0],2,2]);
Framedata("3",[A,1.5,1.2]);       // left figure
Framedata([B,C]);                 // center figure
Framedata([D,E],["corner"]);      // right figure
```



Reference Ovaldata.

⇒Command List

Polygonplot

Usage Polygonplot(name, point list, integer, options)

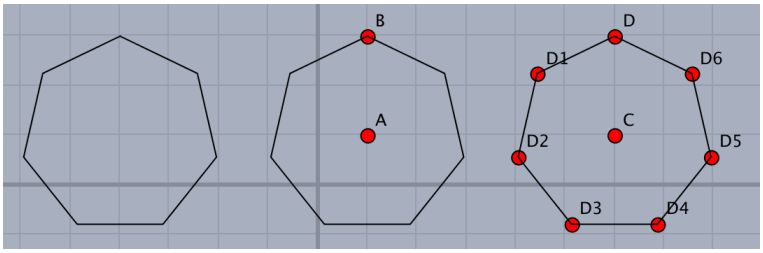
Description Generic function to draw a polygon inscribed inside the circle.

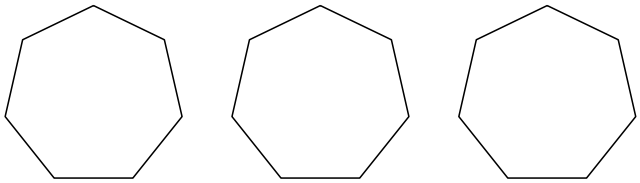
Details If the point list is [A,B] then the center is A and the radius is AB for the circle. Corresponding circle is not drawing. Two points A,B allowed to be coordinates.

option : If A and B are geometric point , make geometric apex by "Geo=y".

Examples

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





We can draw the regular polygon whose one side is the line segment AB.

```
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` is the list of complex numbers correspond to each vertex, `pt` is the list of coordinates of vertexes.

⇒Command List

1.2.3 Curved line

Bezier

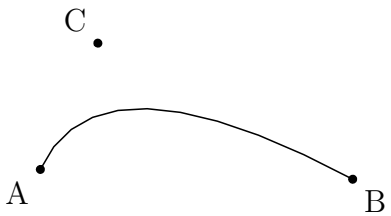
Usage Bezier(name,nodes of curve, control points, options)

Description Draw a bezier curve.
For each interval, control points are given in two lists for 3rd-order and one list for 2nd-order Bezier curve.
You can specify the number of division among nodes (default value is 10).

Examples

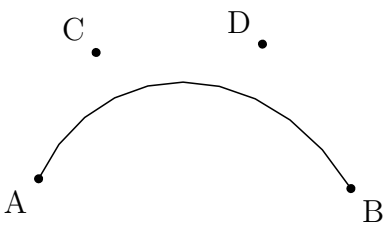
2nd-order Bezier curve
`Bezier("1",[A,B],[C]);`

bz1



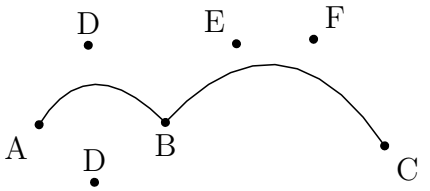
3rd-order Bezier curve
`Bezier("c",[A,B],[C,D]);`

bzc



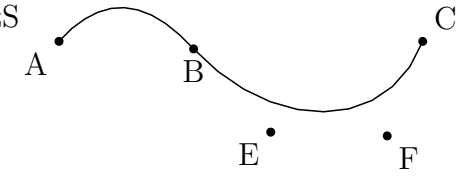
Connecting two curves,
`Bezier("3",[A,B,C],[[D],[E,F]]);`

bz3

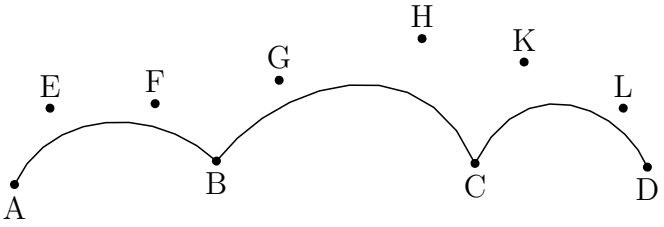


If D,B,E are on the straight line,
the curve becomes smoothly.
`Bezier("S",[A,B,C],[[D],[E,F]]);`

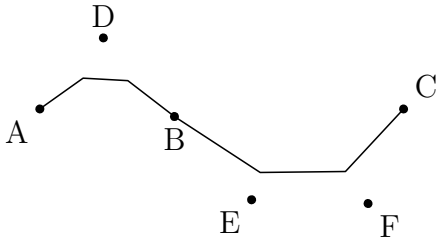
bzS



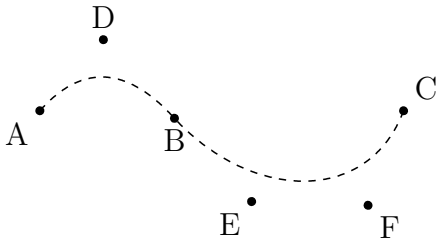
`Bezier("name",[A,B,C,D],[E,F,G,H,K,L] );`



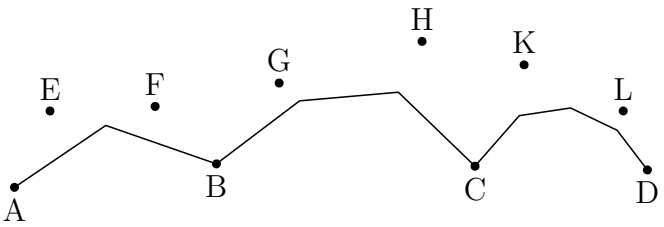
`Bezier("1a",[A,B,C],[[D],[E,F]],["Num=3"]);`



`Bezier("d5e",[A,B,C],[[D],[E,F]],["Num=200","da"]);`



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



⇒Command List

Beziersmooth

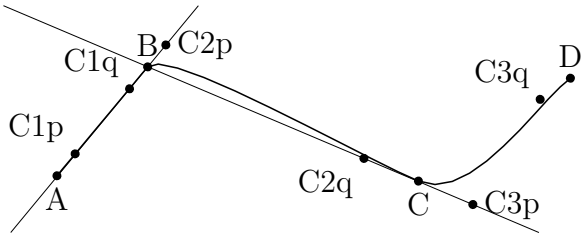
Usage Beziersmooth(name, a list of nodes, options);

Description Generic function to draw a smooth Bézier curve.

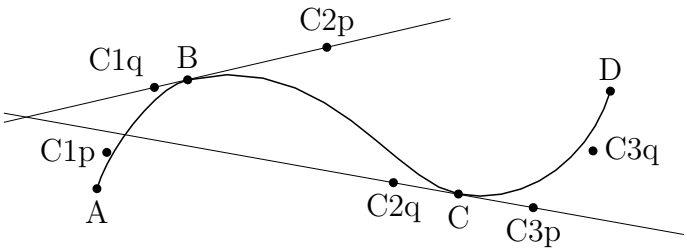
Details Control points are added to keep smoothness.

Examples

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



Remark Control points are movable.



⇒Command List

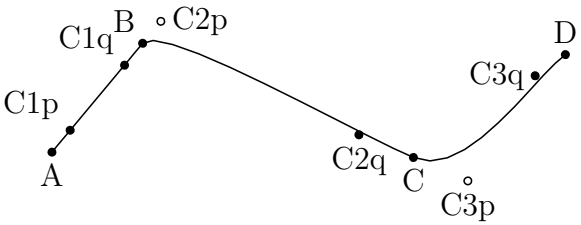
Beziersym

Usage Beziersym(name, a list of nodes, options);

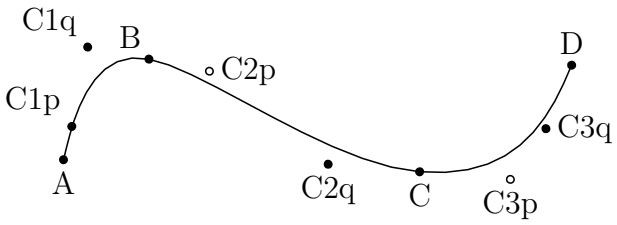
Description Generic function to draw a smooth Bézier curve.

Details Control points are added to be symmetric with respect to each node.

Examples



Remark Some control points are movable.



⇒Command List

Mkbeziercrv

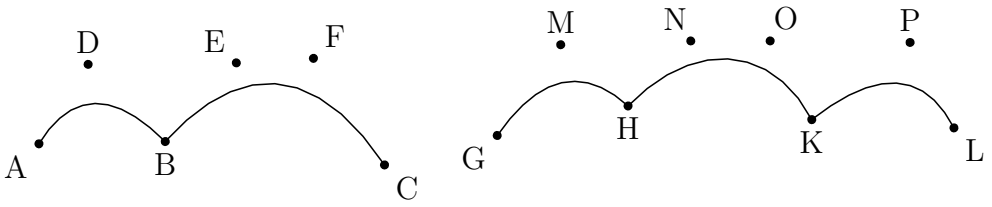
Usage Mkbeziercrv(name, [nodes, control points], options)

Description Draw some Bézier curves.

Details In the case of a single Bézier curve, `[ ]` outside the list can be omitted.
Mkbeziercrv(name, [nodes, control points], options) is same as Bezier(name, [nodes, control points], options).
`Mkbeziercrv("n",[[A,B,C],[[D],[E,F]]])` is same as `Bezier("n",[A,B,C],[[D],[E,F]])`. The name of the plotting data is "bz".

Example1

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



⇒Command List

Mkbezierptcrv

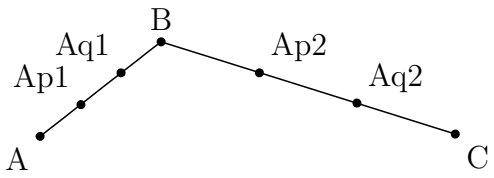
Usage Mkbezierptcrv(a list of points, options)

Description Draw a Bézier curve.

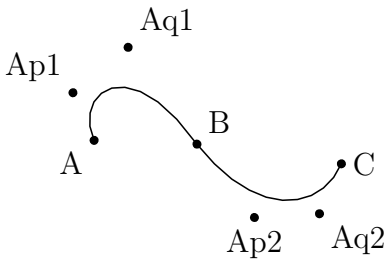
Details Arrange the control points automatically. After that, move the nodes and the control points and correct the Bézier curve to what you want to draw.
In the case of multiple curves, `[ ptlist1, ptlist2.... ]`
The name is automatically attached in order from A.
The options are as follows:
"Deg=..." You can specify the degree (Default is 3rd order).
"Num=..." You can specify the partition number (the partition point number − 1) for each section (Default is 10).

Example

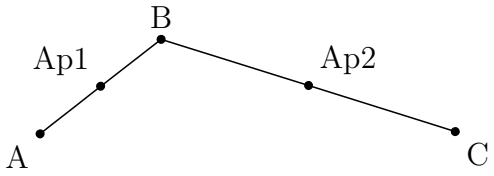
```
Mkbezierptcrv([A,B,C]);
```



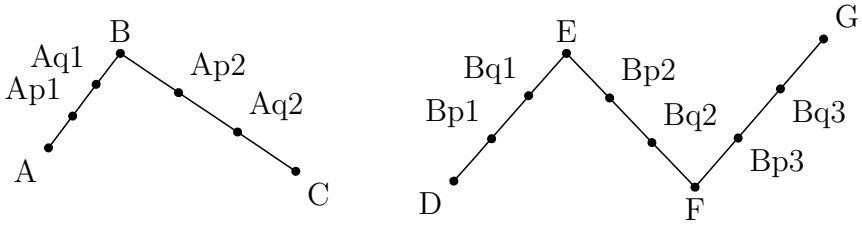
After that, move the nodes and the control points and correct the Bézier curve to what you want to draw.




```
Mkbezierptcrv([A,B,C],["Deg=2"]);
```
If Deg = 2, it is the Bézier curve of 2nd order.
One control point can be set for each section.



In the case of multiple curves, `[ ptlist1, ptlist2.... ]`
```
Mkbezierptcrv([[A,B,C],[D,E,F,G]]);
```



⇒Command List

Bspline

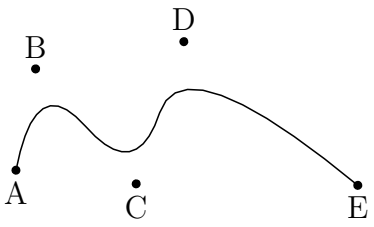
Usage Bspline(name ,list of control points, options)

Description Draw second degree B-spline curve.

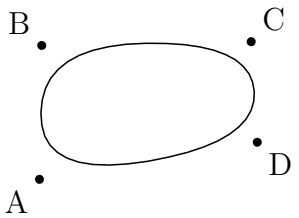
Details Though not displayed, nodal points are calculated automatically.

Examples

`Bspline("1",[A,B,C,D,E]);(=Bezier("1",[A,(B+C)/2,(C+D)/2,E],[B,C,D]);)`
The name becomes `bzb1` instead of `bz1`. Endpoints can be moved instead of control points.



`Bspline("1",[A,B,C,D,A]);`
The generated curve becomes closed when the first component of the list is the same as the last one.



⇒Command List

CRspline

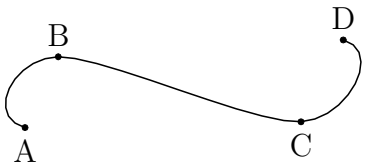
Usage CRspline(name, list of node points, options)

Description Draw single Catmull-Rom spline curve.

Details Only node points are free and control points cannot be moved.
Extra options is :
`"size->"` specifies the thickness of line on the Euclidean view.

Examples

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



⇒Command List

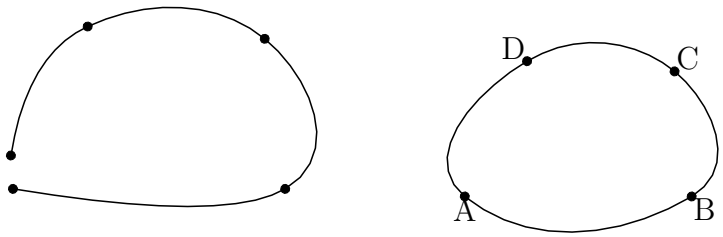
Ospline

Usage Ospline(name, a list of control points, options);

Description Generic function to draw a spline curve of Oshima.

Examples

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



Reference Bspline.

⇒Command List

Circledata

Usage Circledata(name, list, options)

Description Draw a circle or polygon.

Details The list consists of the central point and some point on the circle or the radius. It is also permitted that three points on the circle are given in the list. The name can be omitted when the central point and a point on the circle are given with the names of their geometric components.
Options :
`"Rng=[`θ_1, θ_2`]"` specifies the range of argument in which the circle is drawn.
`"Num=number of division"` specifies the number of dividing points used to draw circle. When this number is small, the corresponding polygon is drawn.

Examples

The circle with center `[0,0]` or A and radius 2 (draw A by drawing tool)

```
Circledata("1",[[0,0],[2,0]]);(or [[0,0],2])
Circledata("1",[A,A+[2,0]]);(or [A,2])
```

The circle with center A and radius AB

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

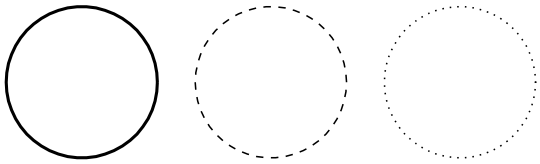
The circle which passes through three points A, B, and C

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

When we use `Circledata([A,B,C])`, the central point of the circle can be drawn by the following command.

```
Pointdata("1",[crABCcenter]);
```

When we add options `"dr,2"`, `"da"`, `"do"`, the following figures are generated respectively.



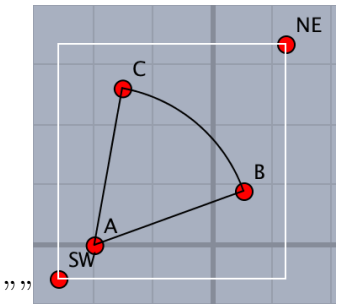
The circular arc with center A, radius AB, and the range of argument $\left[0, \frac{\pi}{3}\right]$

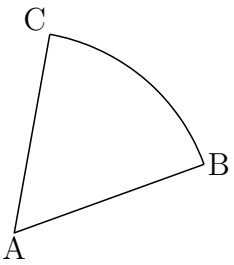
```
Circledata([A,B],["Rng=[0,pi/3]"]);
```

The sector.

Draw A, B , C by drawing tool.

```
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"]);
```



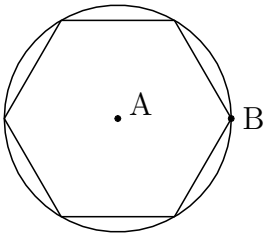


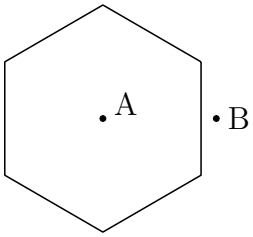
The circle with center A and radius AB, together with the inscribed equilateral hexagon (left figure)

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

The position of vertices can be changed via the option `"Rng="`. (right figure)

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





⇒Command List

Mkcircles

Usage Mkcircles()

Description Create plotting data of all geometric circles.

Details All circles drawn by the ”add circle” tool (any one of three types) in the Euclidean view are used as plotting data as they are. For example, if you create a circle with the center A and the point on the circumference as B, the plotting data `crAB` is created. After that, if you change the identification name of point B (for example to Q) in the inspector of the Euclidean view, the geometric point name is also changed. Even if the circle has already been drawn, it can be changed.

⇒Command List

Ellipseplot

Usage Ellipseplot(name, [F1,F2,A/a], range, options)

Description Generic function to draw ellipse.

Examples

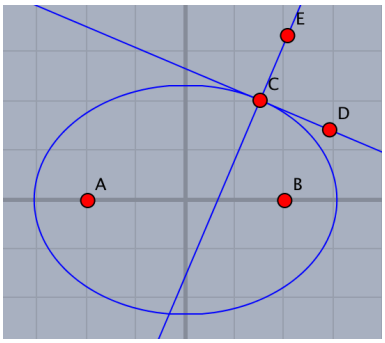
F1,F2 are focus points, A is a point on the ellipse, a is the length F1-A-F2.
Default of the range is [-5,5].

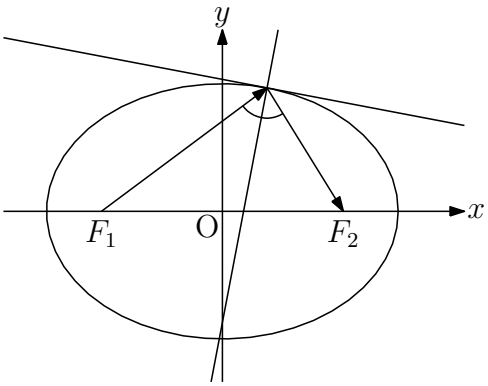
Examples

```
Ellipseplot("1",[A,B,4]);            //sum of distance from Focus is 4.
Ellipseplot("1",[A,B,C],"[0,pi]"); //half of ellipse.
```

Now draw tangent and normal. Draw figures by draw tool on Euclidean view. Put point D on tangent and E on normal.

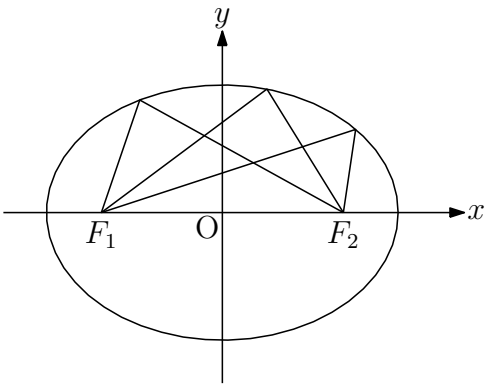
```
Ellipseplot("1",[A,B,C]);
Lineplot([C,D]);
Lineplot([C,E]);
Arrowdata([A,C]);
Arrowdata([C,B]);
Anglemark([A,C,B]);
Expr([A,"s2","F_1",B,"s2","F_2"]);
```





Now draw point D and E on ellipse.

```
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"]);
```



⇒Command List

Hyperbolaplot

Usage Hyperbolaplot(name,[F1,F2,A], range, options)

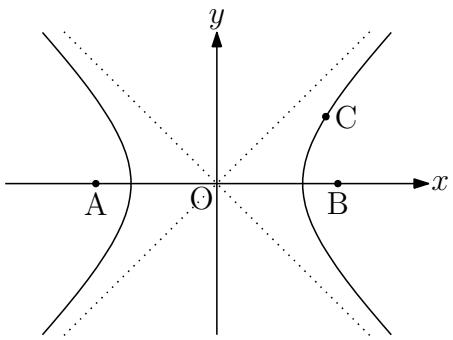
Description Generic function to draw a hyperbola.

Details Option is as usual except "Asy=line style".This option is for drawing asymptotes.

Reference Ellipseplot and Parabolaplot.

Examples

```
Hyperbolaplot("1",[A,B,C]);
Hyperbolaplot("1",[A,B,2]);
Hyperbolaplot("1",[A,B,C],["Asy=do"]);
```



⇒Command List

Parabolaplot

Usage Parabolaplot(name, [A, B, C], range, options);

Description Generic function to draws a parabola.

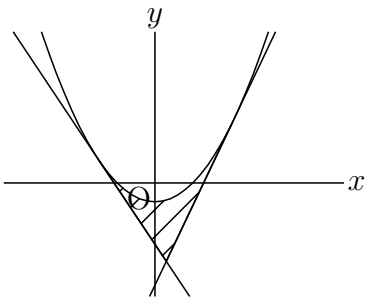
Details A is the focus point, BC is the directrix. Default of the range is $[-5, 5]$.

Examples

```
Parabolaplot("1",[A,B,C]);
Parabolaplot("1",[A,B,C],"[-4,4]");          //range is [-4,4]
Parabolaplot("1",[[0,1],[-1,-1],[1,-1]]); // coordinate
```

Area enclosed by parabola and tangent

```
Parabolaplot("1",[A,B,C]);
Putoncurve("D","gr1para");
Putoncurve("E","gr1para");
Tangentplot("1","gr1para","x="+D.x);
Tangentplot("2","gr1para","x="+E.x);
pts=Intersectcurves("lntn1","lntn2");
Listplot("1",[E,pts_1,D]);
Hatchdata("1",["ii"],[["gr1para","s"],["sg1","n"]]);
```



⇒Command List

Ovaldata

Usage Ovaldata(name, [A, B], options);

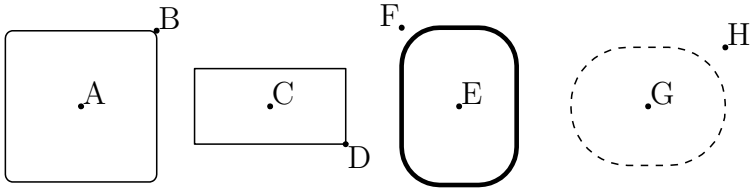
Description Generic function to draw a rectangle with rounded corners.

Details A is the center, B is a diagonal point.

option : ratio of the roundness (default is 0.2) .

Examples

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



⇒Command List

1.2.4 Graph of function

Plotdata

Usage Plotdata(name, function, variable and range, options)

Description Generic function to draw the graph of function.

Details Options : next options and usual options.

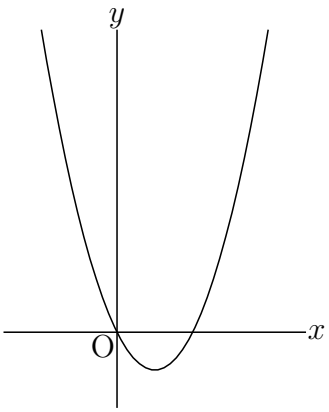
"Dis=real number": discontinuity

"Exc=list of real numbers": exclusion points

"Exc=function": exclude the zero points of the function

Examples

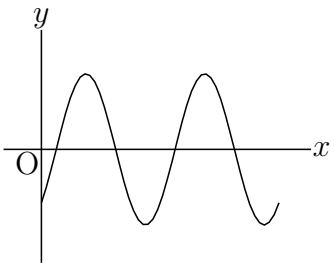
```
Plotdata("1","x^2-2*x","x");
```



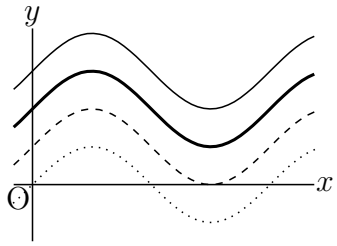
Draw in red.

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

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

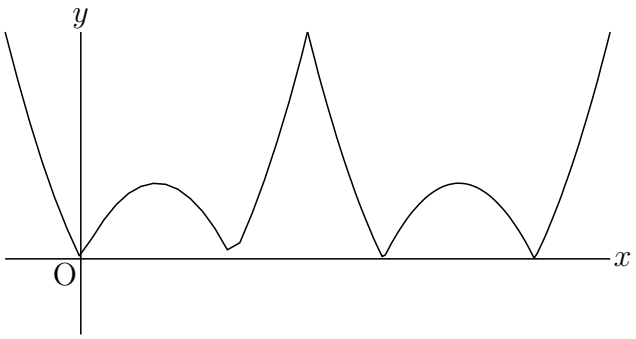



```
Plotdata("1","sin(x)","x",["do"]);
Plotdata("2","sin(x)+1","x",["da"]);
Plotdata("3","sin(x)+2","x",["dr,2"]);
Plotdata("4","sin(x)+3","x");
```



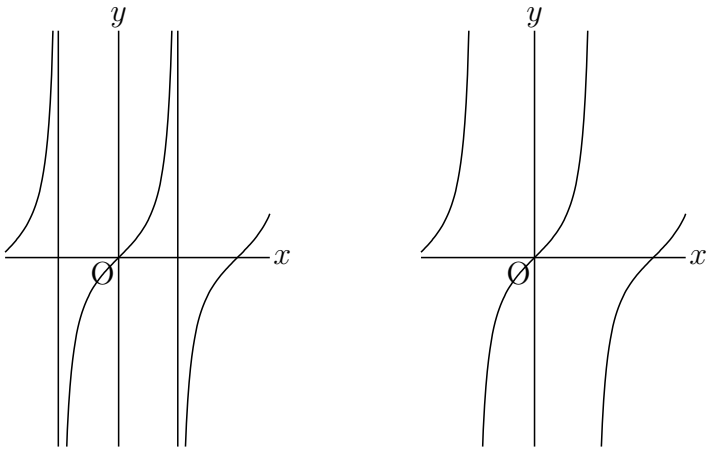
Draw smoothly by "Num=n" option.

Left figure: "Num=50"(default), Right figure: "Num=200"



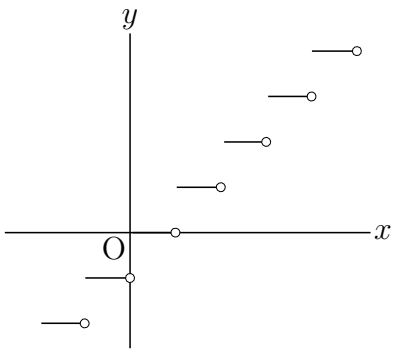
Draw discontinuity accurately by "Dis" option.

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



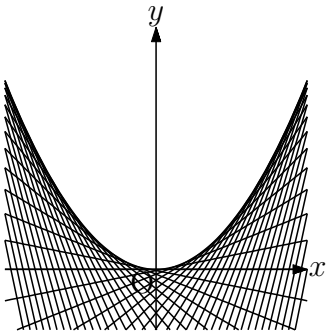
Draw floor function.

```
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"]);
);
```



Assign a value to the letter "b".

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



⇒Command List

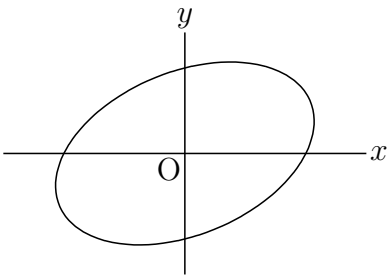
Implicitplot

Usage Implicitplot(name,functionstring,range of x, range of y, options);

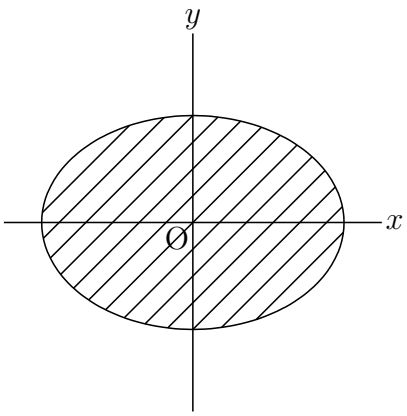
Usage Generic function to draw the graph of a implicit function.

Examples

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




```
Implicitplot("1","x^2+2*y^2=4","x=[-2,2]","y=[-2,2]");
Hatchdata("1",["i"],[["imp1"]]);
```



⇒Command List

Deqplot

Usage Deqplot(name, expression, names of variations, options)

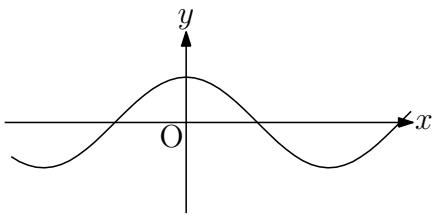
Description Draw the solution curve of a differential equation.

Details The differential equation and its initial conditions should be specified as arguments.

Examples

The solution curve of the equation $y" = -y$ with initial conditions $y(0) = 1, y'(0) = 0$

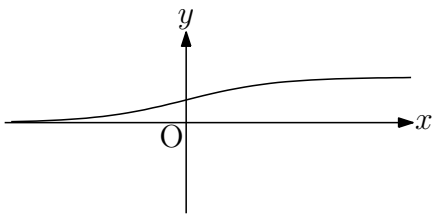
```
Deqplot("1","y``=-y","x",0, [1,0]);
```



Remark Derivative symbol `y`` is a backquate, not a single quote.

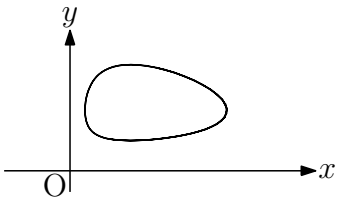
The solution curve of the equation $y' = y * (1 - y)$ with initial condition $y(0) = 0.5$

```
Deqplot("2","y`=y*(1-y)","x",0, 0.5,["Num=100"]);
```



The solution curve of the equation $[x, y]' = [x(1-y), 0.3y(x-1)]$ of variable t with initial conditions $x(0) = 1, y(0) = 0.5$

```
Deqplot("3","[x,y]`=[x*(1-y),0.3*y*(x-1)]","t=[0,20]",[1,0.5],["Num=200"]);
```



⇒Command List

Paramplot

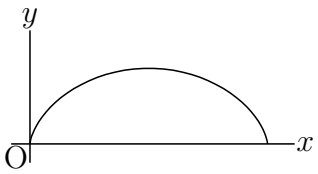
Usage Paramaplot(name, expression, variable and domain, options);

Description Generic function to draw a curve of parametric representation.

Examples

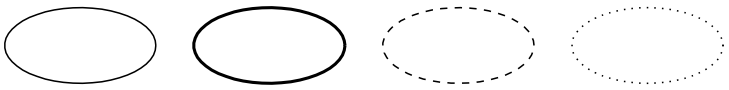
Draw a cycloid curve.

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



Draw ellipses with options.

```
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"]);
```



⇒Command List

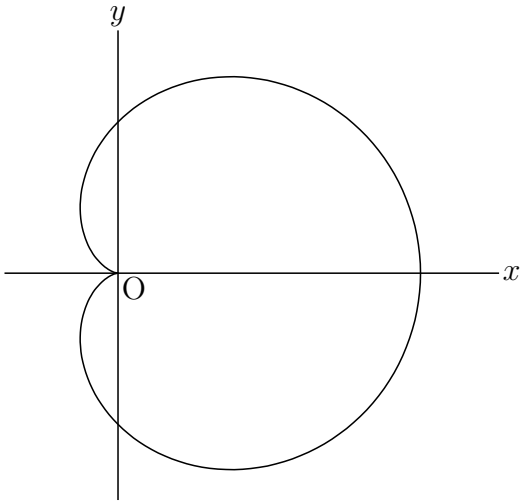
Polarplot

Usage Polarplot(name, expression, variable and domain, options);

Description This function draws a curve of polar equation.

Examples To draw a cardioid.

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



⇒Command List

Periodfun

Usage Periodfun(defL,repeat,options)

Description Function to draw the graph of a periodic function.

Details `defL` is a list of fun(str),interval,division number.

The options are "Con=n/do, Color=name" for discontinuous parts.

ex. "Con=do,Color=red" , " Con=n". Default is broken line and draw.

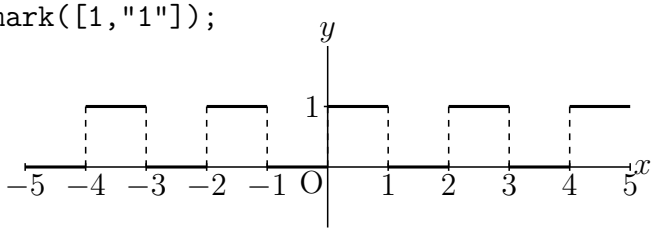
Repeat count is a count number or a list of count numbers of left side and right side.

The return value are a list of function in Maxima format and the period.

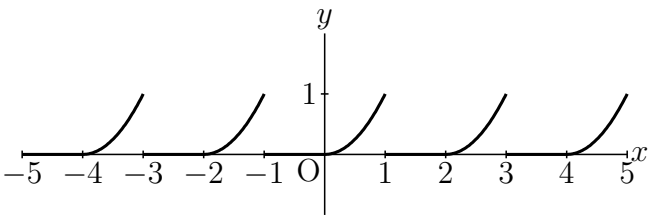
Remark The functions should be defined on the symmetrical interval [-a,a].

Examples

```
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"]);
```




```
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"]);
```



⇒Command List

Fourierseries

Usage Fourierseries(name,coeff,period,terms)

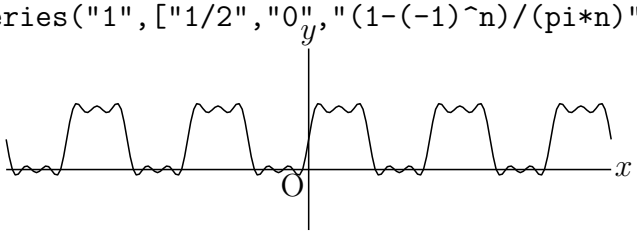
Description Function to draw the graph of a fourier series.

Details $a_0 + \sum_{n=1}^{\infty}(a_n \cos nx + b_n \sin nx)$

`coeff` is a list of $[a_0, a_n, b_n]$. Each element are string.
`term` is a number of terms.

Examples

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



⇒Command List

Tangentplot

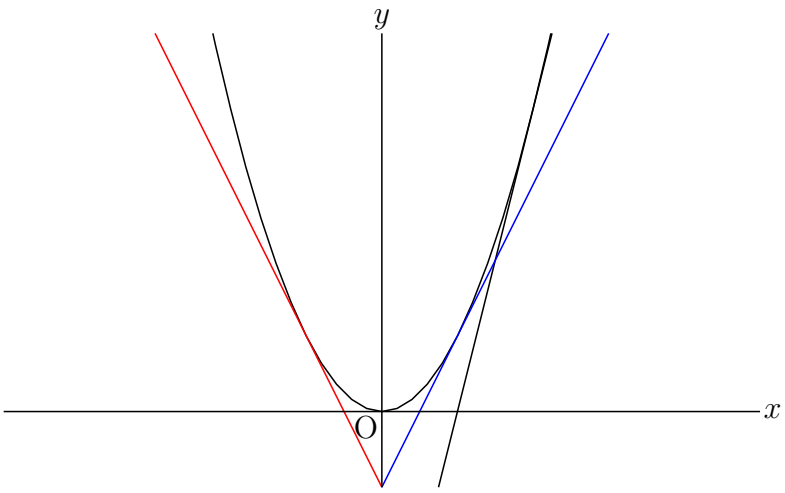
Usage Tangentplot(name,PD, pointinfo, options);

Description Generic funtion to draw a tangent line of a plotting data.

Details The pointinfo is one of ”x=xvalue”,”y=yvalue”, [point, parameter].
The option ”nth” is used to set the number when plotting data has multi intersects.

Examples

```
Plotdata(`"1","x^2","x")
Tangentplot("1","gr1","x=2");
Tangentplot("2","gr1","y=1",["Color=red"]);
Tangentplot("3","gr1","y=1",[2,"Color=blue"])
```



Reference Derivative.

⇒Command List

1.2.5 Letter

Letter

Usage Letter([position, direction, string],options)

Description Display the string.

Details Write the string at the position specified by position (or coordinates) and direction.
The position (or coordinates) can also be specified by the geometric point name.
The direction is ”e”, ”w”, ”n”, ”s”, ”c”. The distance from the specified position can also be given as a numerical value. For example, ”e2” and ”e3” are placed twice and three times of the slightly unit distance away from ”e”, respectively.
Multiple strings can be passed in the form of a list.

Remark The derivative symbol ′ uses $ ′ $ (single quart) in mathematical mode (interleaved with two $ s). Option is size of font. For example, [”size=32”]

Example

```
Letter([[2,1] ,"se","P"]);  // Display P in the southeast of the coordinates (2, 1).
Letter([C ,"c", "C"]);      // Display C with the point C as the center.
Letter([A,"sw","A",E,"s","$ f(x)=\frac{1}{4} x^2 $"]);
```

//Display A in the southwest of point A and $f(x) = \frac{1}{4}x^2$ in the south of the point E.

⇒Command List

Letterrot

Usage Letterrot([pos, dir, move, string])

Description Rotate a string and display it.

Details At the position of the coordinates, rotate to the direction specified by the direction vector and write the string.
The third argument is a minute movement amount and can be abbreviated.

Example

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

It is also possible to write as follows, abbreviated for the amount of movement.

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

Reference Exprrot.

⇒Command List

Expr

Usage Expr([pos, dir, string]);

Description Generic function to write an expression in TEXstyle.

Details pos : position

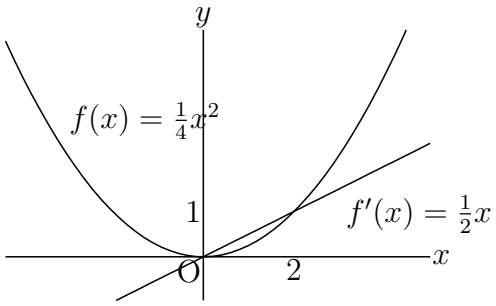
dir : direction(e,w,s,n,ne,nw,se,sw,c)

string : expression

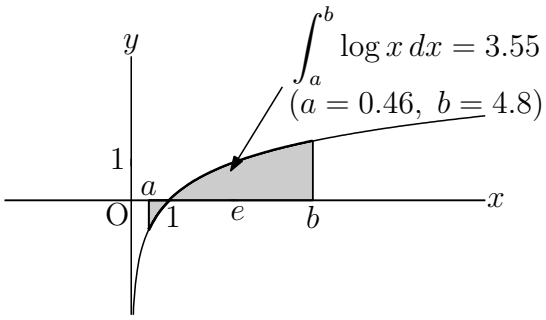
Also see Letter

Examples

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




```
Arrowdata(Q,P);
Expr([Q,"ne2","\displaystyle\int_a^b \log x\,dx="
      +text(L.x*(log(L.x)-1)-G.x*(log(G.x)-1))]);
```



⇒Command List

Exprrot

Usage Exprrot([pos, dir,[move(optional)], string);

Description Generic function to write a rotated expression in TEXstyle.

Details pos : position : coordinate or name

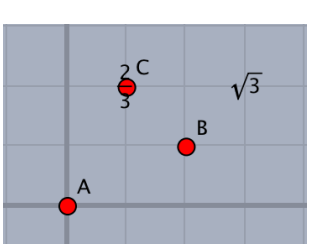
dir : direction vector : coordinate or name

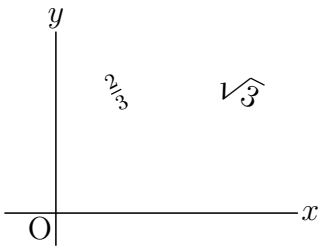
move: ”t”:tangent , ”n”:normal

string : expression

Examples

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





⇒Command List

1.2.6 Marking

Anglemark

Usage Anglemark(a list of points, options);

Description draw an angle mark with an arc at the angle determined by [A,B,C]

Options :

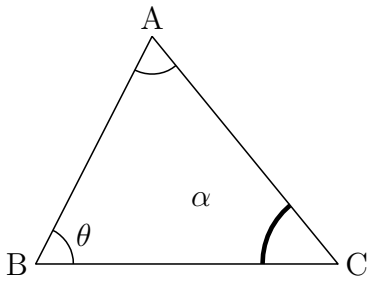
numerical value size of mark (default is 1)

draw text `"Expr=n,str"` or `"Let=n,str"`

Examples

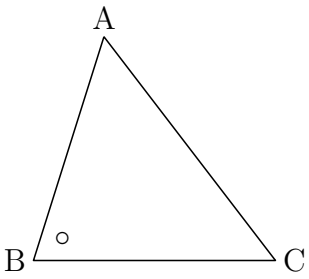
draw an angle mark at interior angles of a triangle, write characters.

```
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"]);
```



draw ∘ at interior angles of a triangle.

```
Listplot([A,B,C,A]);
Letter([A,"n1","A",B,"w1","B",C,"e1","C"]);
Anglemark([C,B,A],["Expr=\circ","nodisp"]);
```



Remark You can draw an angle mark with a parallelogram. Refer to Paramark .

⇒Command List

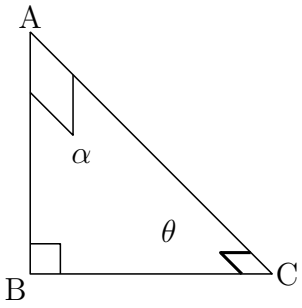
Paramark

Usage Paramark([A, B, C], options);

Description Generic function to draw an angle mark with a parallelogram at the angle determined by [A,B,C].
Options : numerical value size of mark (default is 1) and usual options.

Examples Draw an angle mark at interior angles of a triangle, write characters.

```
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"]);
```



Reference Anglemark.

⇒Command List

Bowdata

Usage Bowdata(a list of points, options);

Description draw the shape of bow connecting two points in the list counterclockwise

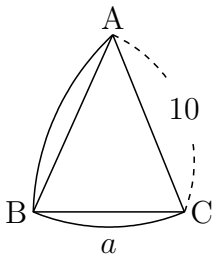
Details Options :

curvature (default is 1)
size of the blank space in the middle of bow
expression located at the blank space `"Expr=expressions"`
The location of expressions can be modified via `"Expr=tn, expressions"` where `t` specifies the movement in the direction of segment and `n` specifies that of normal direction. Both positive and negative numbers are permitted.
line type `"dr,n"` , `"da,m,n"` , `"do,m,n"`

Examples

draw the shapes of bow along with the edges of triangle ABC and add marks.

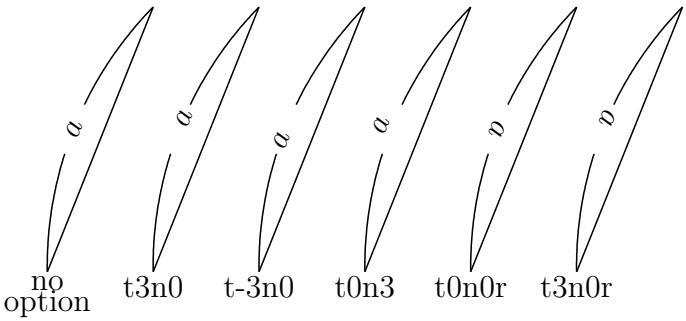
```
Listplot([A,B,C,A]);
Letter([A,"n1","A",B,"w1","B",C,"e1","C"]);
Bowdata([A,B]);
Bowdata([B,C],[1,"Expr=t0n3,a"]);
Bowdata([C,A],[2,1.2,"Expr=10","da"]);
```



Expressions can be displayed in rotated manner via `"Exprrot=tn,expressions"` though the Euclidean view does not correspond to this modification. Adding `r` to `tn` results in the turning round.

Examples

```
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"]);
```



⇒Command List

Drawsegmark

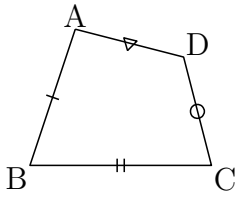
Usage Drawsegmark(name, list, options) or Segmark(name, list, options)

Description Add a mark to a segment.

Details Add a mark to the segment determined by the end points specified in the list. Four kinds of marks can be used.
Extra options :
`"Type=n"` (n=1,2,3,4) specifies the kind of mark.
`"Width="` specifies the distance between two segments of the mark (in case when $n = 2$).

Examples

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



⇒Command List

Htickmark

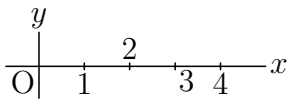
Usage Htickmark([x-coord,[direction(optional)],expression,...]);

Description Generic function to tick on the horizontal axis.

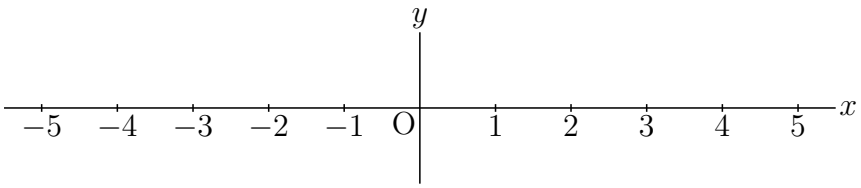
Details Default of direction is `"s1"`. Minor adjustments are not displayed on the Euclidean view, you have to check the results on the PDF file. The length of tickmarks can be set by the function Setmarklen().

Examples

```
Htickmark([1,"1",2,"n1","2",3,"se","3",4,"4"]);
```




```
ticks=apply(-5..5,x,[x,text(x)]); // ticks is [ [-5,"5"],...,[5,"5"] ]
ticks=remove(ticks,[[0,"0"]]);    // [0,"0"] is removed
ticks=flatten(ticks);             // ticks becomes [-5,"5",...,5,"5"]
Htickmark(ticks);
```



Reference Vtickmark.

⇒Command List

Vtickmark

Usage Vtickmark([y-coord,[direction(optional)],expression,...]);

Description Generic function to tick on the vertical axis.

Details Default of direction is `"w1"`.

Examples

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

Reference Htickmark.

⇒Command List

Rulerscale

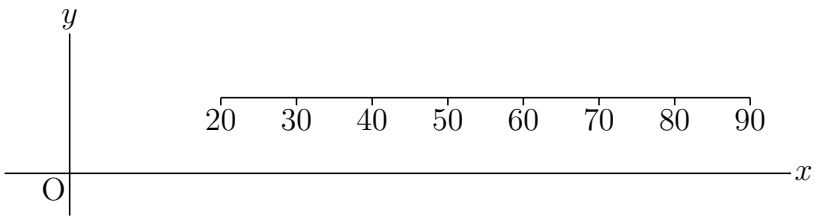
Usage Rulerscale(starting point, horizontal marks, vertical marks);

Description Generic functio to put ruler marks.

Details The marks are give as a list.
`["r",a,b,c,d]` to put marks from `a` to `b` with intervals `c`, scales `d`.
`["f",n1,"str",n2,"str", ]` to put marks as the same format as Htickmark.

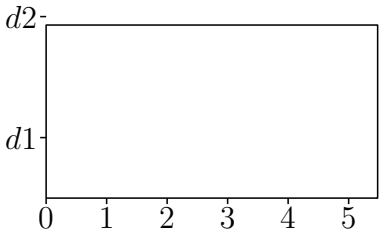
Examples1

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



Examples2

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



⇒Command List

1.3 Using plotting data

Changestyle

Usage Changestyle(list of PD, options)

Description Change the option for drawing.

Details Change the option for drawing several shapes altogether.

Examples

Draw segment AB and Circle AB with broken line on the Euclidean view and keep them from being drawn on TEX final output.

```
Listplot([A,B]);
Circledata([A,B]);
Changestyle(["sgAB","crAB"],["da","notex"]);
```

⇒Command List

Drawfigures

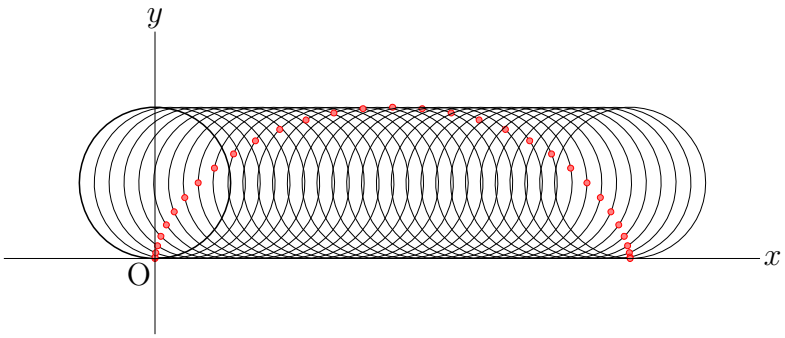
Usage Drawfigures(or Drwfigs)(name,List of PDs,List of Options)

Description Manipulate a plural number of PDs together.

Remark List of Options should corrensponds to that of PDs.

Examples After manipulating PDs of a circle and a point on the circle by AddGraph, you can translate or rotate them together.

```
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]);
);
```



⇒Command List

Invert

Usage Invert(PD)

Description Rearrange plotting data in the reverse order.

Examples

See the examples in Shade

⇒Command List

Joincrvs

Usage Joincrvs(name, list of PDs, options)

Description Create a plotting data of one curve by connecting a list of plotting data of adjacent curves.

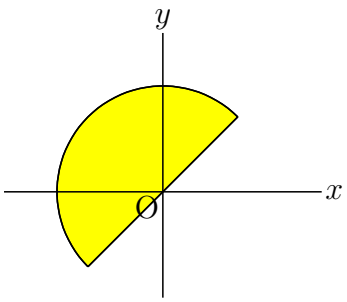
Details The list of curves is specified in the adjacent order.
Options is line type.

Examples

Draw the closed curve obtained from the line segment $y = x$ $(-\sqrt{2} \leq x \leq \sqrt{2})$ and the half circle, and paint the interior of the closed curve using the yellow color.
Put the point A at the origin and the point B in the appropriate place.

```
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"]);
```



⇒Command List

Partcrv

Usage Partcrv(name, A, B, PD, options)

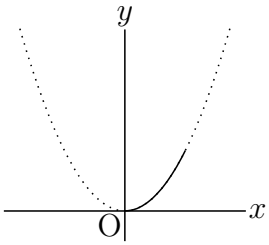
Description Generic function to make a piece of curve from the PD between the points A and B.

Details The order of two points A, B must be same as the direction of the curve. Options are ”dr, n”, ”da,m,n” or ”do,m,n”

Examples

In the following example We draw a parabola with dotted line and draw a piece of curve with real line.

```
Plotdata("1","x^2","x",["do"]);
Partcrv("1", [0,0], [1,1], "gr1");
```



In the next example we draw a piece of circle with real line. The direction of a circle is counterclockwise direction.

```
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");
```

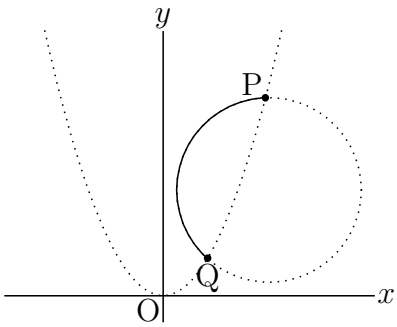


figure of part1

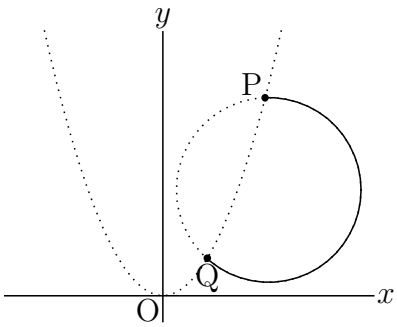
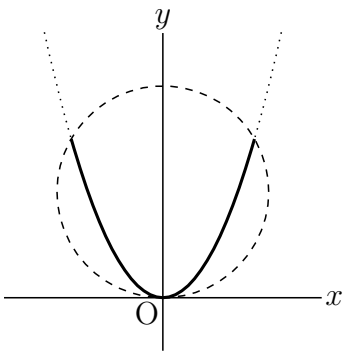


figure of part2

In the last example we draw the pice of parabola:$y = x^2$ which is cut off by the circle.

```
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"]);
```



⇒Command List

Enclosing

Usage Enclosing(name, a list of plotdata, options);

Description This function makes a closed curve form the list of plotdata.

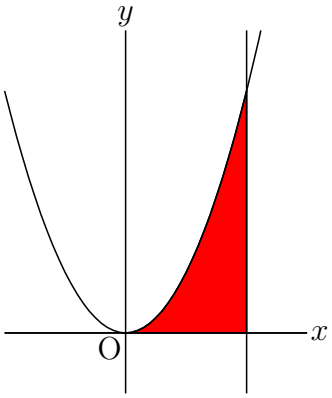
Details Options are:

near point from start position : Set in case where the first curve and the last curve have multi intersects.

`"dr","da", "do","`notex","nodisp","Color= "` : as usual.

Examples

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



RemarkThe followings have the opposite direction.

```
Enclosing("1",["ln1","ln2","Invert(gr1)"]);
Enclosing("1",["gr1","Invert(ln2)","Invert(ln1)"]);
```

⇒Command List

Hatchdata

Usage Hatchdata(name, a list of ”i” or ”o” , a list of a list of PD , options)

Description Generic function to draw hatch lines in the close curve.

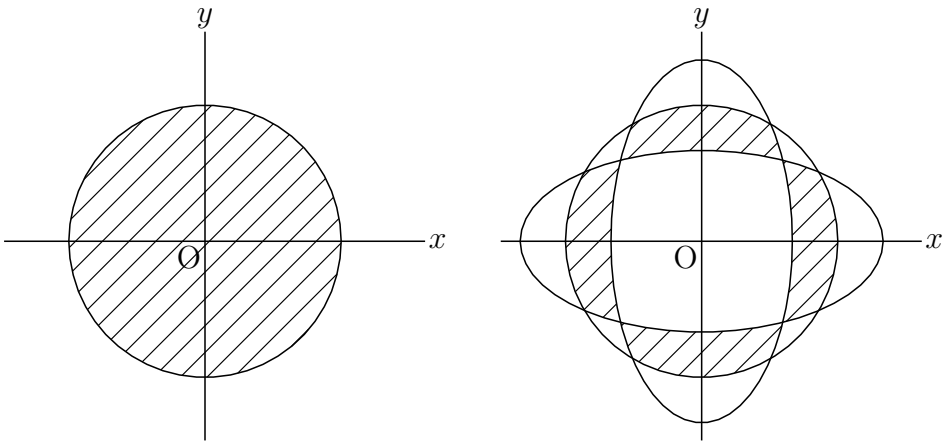
Details Options are:

angle(degree,45), interval(ratio,1) of hatches,
`"Max=(default:20)"` maximum of the number of hatches.
`"No=pointlist"` not executed when any point is selected
`"File=y/m/n(default:n)"` whether to make data file or not
`"Check=pointlist"` data file updated if any point is changed

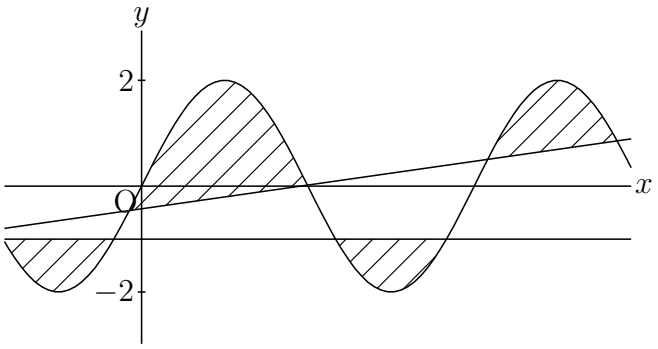
Examples

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

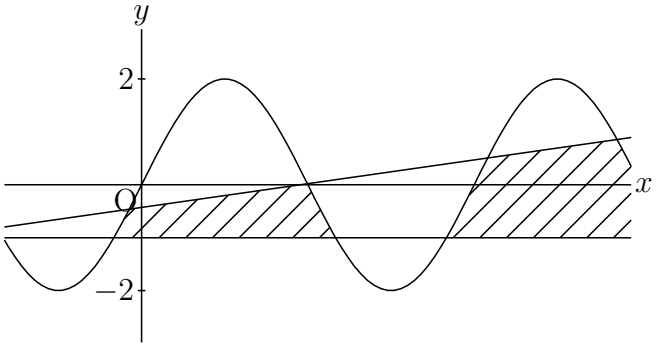
```
Circledata([A,B],["dr"]);
Paramplot("1","[4*cos(t),2*sin(t)]","t=[0,2*pi]");
Paramplot("2","[2*cos(t),4*sin(t)]","t=[0,2*pi]");
Hatchdata("1",["ioi"],[["crAB"],["gp1"],["gp2"]],["dr,0.7"]);
Hatchdata("2",["iio"],[["crAB"],["gp1"],["gp2"]],["dr,0.7"]);
```



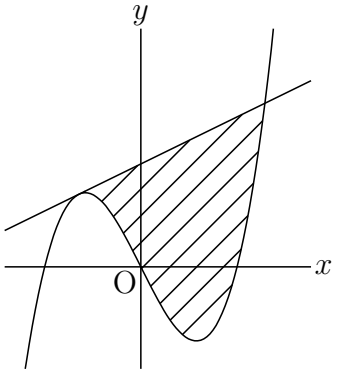
```
Plotdata("1","2*sin(x)","x=[-pi,3*pi]",["Num=100"]);
Listplot([A,B]);
Listplot([A,C]);
Hatchdata("1",["ii"],[["sgAB","n"],["gr1","s"]],["dr,0.7"]);
Hatchdata("2",["ii"],[["sgAC","s"],["gr1","n"]],["dr,0.7"]);
```



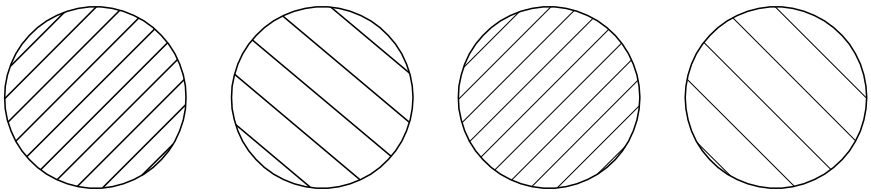

```
Plotdata("1","2*sin(x)","x=[-pi,3*pi]",["Num=100"]);
Listplot([A,B]);
Listplot([A,C]);
Hatchdata("1",["iio"],[["sgAB","s"],["sgAC","n"],["gr1","n"]]);
```



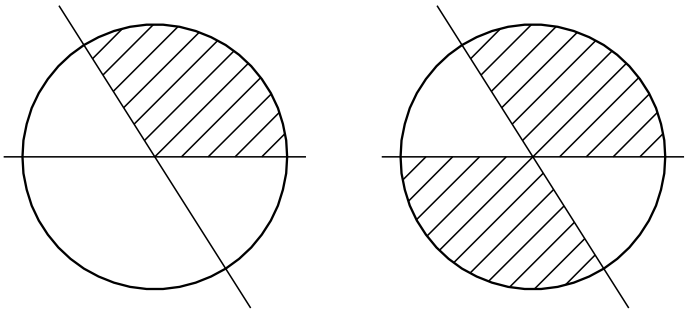

```
Deffun("f(x)",["regional(y)","y=x^3-2*x","y"]);
Plotdata("1","f(x)","x",["Num=100"]);
Putoncurve("A","gr1");
coef=Derivative("f(x)","x",A.x);
Defvar(["coef",coef]);
Deffun("g(x)",["regional(y)","y=coef*(x-A.x)+A.y","y"]);
Plotdata("2","g(x)","x",["Num=1"]);
if(!Ptselected(), // if any point is not selected
  Enclosing("1",["gr2","Invert(gr1)"],[A,"nodisp"]);
  Hatchdata("1",["i"],[["en1"]]);
);
```



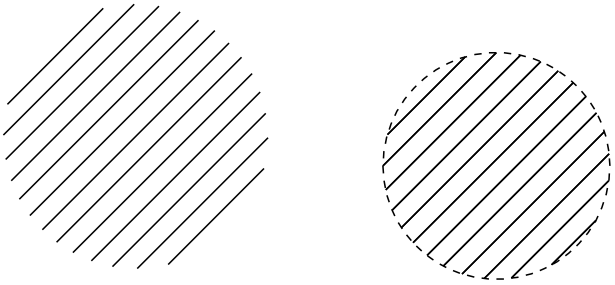
```
Circledata([A,B]);
Hatchdata("1",["i"],[["crAB"]]);
Hatchdata("2",["i"],[["crAB"]],[-40,2]);// angle = −40°, interval = ×2|
Hatchdata("3",["i"],[["crAB"]],["dr,0.5"]);
Hatchdata("4",["i"],[["crAB"]],[-45,2,"dr,0.5"]);
```



```
Lineplot("1",[A,B]);// name of this data is ln1
Lineplot("2",[A,C]);
Hatchdata("5",["iii"],[["crAB"],["ln1","n"],["ln2","n"]]);
Hatchdata("6",["ioo"],[["crAB"],["ln1","n"],["ln2","n"]]);
```



```
Circledata([A,B],["nodisp"]);
Hatchdata("7",["i"],[["crAB"]]);
Circledata([A,B],["da"]);
Hatchdata("8",["i"],[["crAB"]]);
```



⇒Command List

Dotfilldata

Usage Dotfilldata(name, list of the dotted sides ”i” or ”o”, list of PD, option)

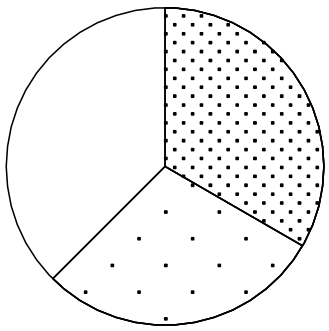
Description Fill a domain with dots.

Details R is called to generate the data. Arguments are the same as Hatchdata. Option is the density of dots from 0.1 to 0.8 (default is 0.3).

Examples

A pie chart
After making closed curve via `Partcrv()` and `Enclosing()`, the surrounded region is filled with dots.

```
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]);
```



⇒Command List

Shade

Usage Shade((”name”),list of PD, options);

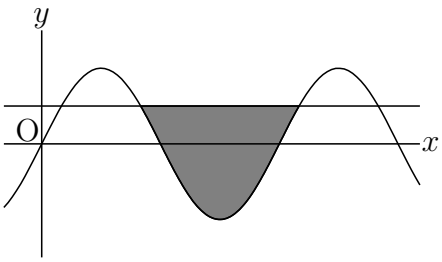
Description This function fills a domain surrounded by a closed curve.

Details Options are as follows. () means the default.

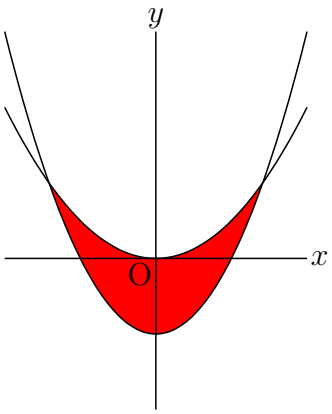
- Starting point. The defaults are the intersect of the first curve and the last one.
- ”Color=”
- ”Enc=n(y)” Whether `Enclosing` is used.
- ”Trim=n(y)” Whether it is trimmed in the window.
- ”First=y(n)”
  Whether `Shade` is placed at the top in TEX. The default is just before Gdata.

Examples

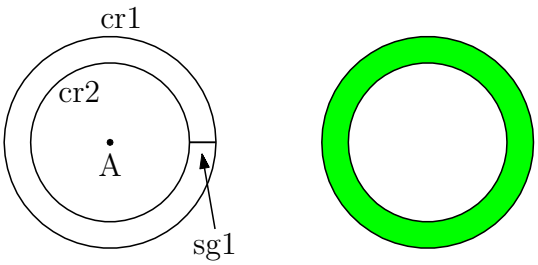
```
Setax([7,"nw"]);
Plotdata("1","2*sin(x)","x",["Num=100"]);
Lineplot("1",[[0,1],[1,1]]);
Shade(["ln1","Invert(gr1)"],[[2.5,1],"Color=0.2*[0,0,0,1]"]);
```




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




```
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"]);
```



Reference Joincrvs.

⇒Command List

Reflectdata

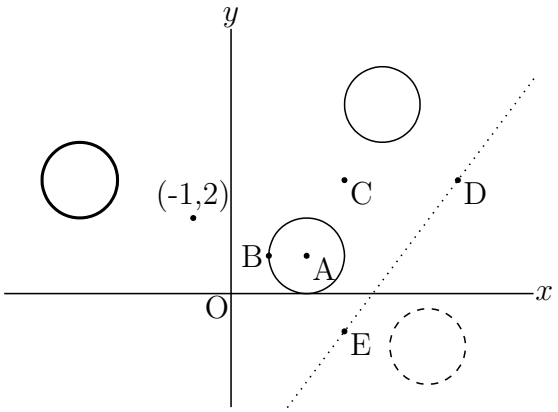
Usage Reflectdata(name, PD, center or axis of symmetry, options);

Description Generic function to draw a reflective curve.

Details axis of symmzetry is defined as a list of 2 points.

Examples

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



⇒Command List

Rotatedata

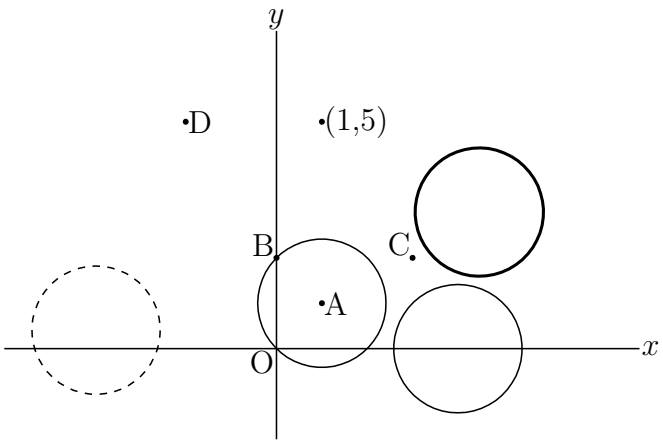
Usage Rotatedata(name, (a list of)PD, angle(degree), options);

Description Generic function to rotate plotting data.

Details Options are center, and as usual. The default of center is [0,0].

Examples

```
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"]);
```



⇒Command List

Scaledata

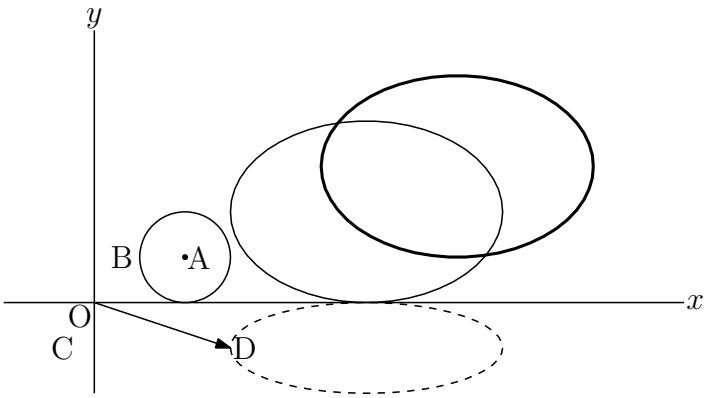
Usage Scaledata(name, list of PD, horizontal ratio, vertical ration, [options]);

Description Generic function to scale plotting data.

Details Options are Center, and as usual. The default of center is [0,0].

Examples

```
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"]);
```



⇒Command List

Translatedata

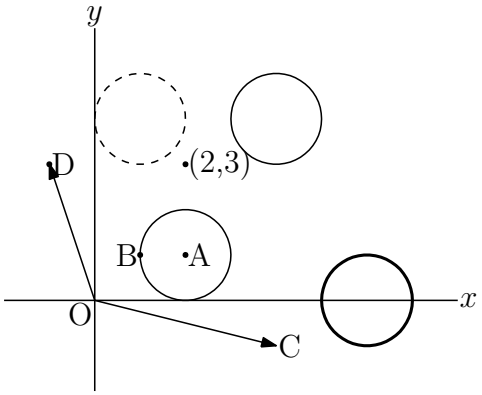
Usage Translatedata(name, list of PD, vector, options);

Description Generic function to translate plotting data.

Details Options are Center, and as usual. The default of center is [0,0].

Examples

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



⇒Command List

1.4 Calculus and I/O

Derivative

Usage

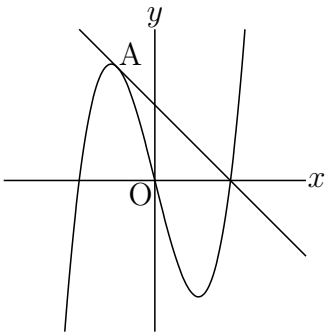
Derivative(function(string), variable(string), value);
Derivative(PD(string), pointinfo, ([nth of intersects]));

Description Generic funtion to find the derivative of a function or a plotting data.

Details The pointinfo is one of "x=xvalue","y=yvalue", [point, parameter].
The option of nth is used to set the number when plotting data has multi intersects.

Examples

```
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"]);
```



Reference Tangentplot.

⇒Command List

Integrate

Usage

Integrate(function or name of PD, "varname=range", [options]);

Description Generic funtion to find the value of numerical integration.

Details Oshima's Bezier formula is used.

Examples

```
f(x):=x^3-2*x^2+2;
val=Integrate("f(x)","x=[0,3]");
println(val); // 8.25 will be displyed.
plotting data("1","x^3-2*x^2+2","x");
println(Integrate("gr1",[0,3]));
```

⇒Command List

Inversefun

Usage Inversefun(function(string), range, value);

Description Generic function to find the value of the inversefunction.

Details The value is found in the range.

Examples

```
x=Inversefun("sin(x)","x=[0,pi/2]",0.5);
```
The value of x is 0.5236.

⇒Command List

Crossprod

Usage Crossprod(vec1, vec2);

Description Generic function to return the cross product of 2 vectors.

Details The vectors are a list with length 3 or 2.

Examples

```
v=Crossprod([1,0,0],[1,1,1]); // The result is v=[0,-1,1].
```

⇒Command List

Dotprod

Usage Dotprod(vec1, vec2);

Description Generic function to return the dot product of 2 vectors.

Examples

```
v=Dotprod([1,2,3],[1,-1,1]); // The result is v=2.
```

⇒Command List

Findarea

Usage Findarea(plotting data(or string of pd));

Description Generic function to return the area enclosed with a close curve.

Details Oshima's Bézier formula is used.

Examples

```
Paramplot("1","[3*cos(t),2*sin(t)]","t=[0,2*pi]");
area=Findarea("gp1");
println(Sprintf(area,6)); // The result is 18.849536.
```

⇒Command List

Findlength

Usage Findlength(plotting data(or string of pd));

Description Generic function to return the length of a curve.

Details Oshima's Bézier formula is used.

Examples

```
Circledata("1",[[0,0],[2,0]]);
len=Findlength("cr1");
println(Sprintf(len,6)); // The result is 12.558097.
```

⇒Command List

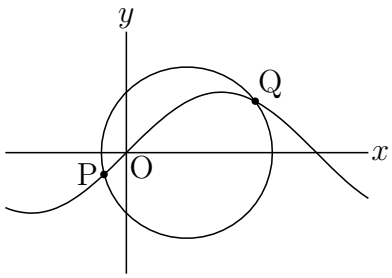
Intersectcurves

Usage Intersectcrvs(plotting data1(string), plotting data2(string));

Description Generic funtion to return a list of intersects of 2 plotting data.

Examples

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



⇒Command List

IntersectcurvesPp

Usage IntersectcrvsPp(plotting data1(string), plotting data2(string));

Description Generic funtion to return a list of intersects with parameters of 2 plotting data.

Details Parameters are positons of the intersect.

⇒Command List

Nearestpt

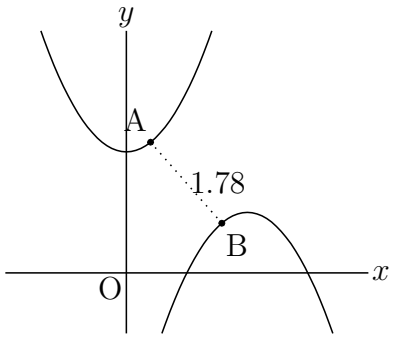
Usage Nearestpt(plotting data1, plotting data2);

Description Generic funtion to return the nearest point with the parameter and the distance.

Examples

```
Plotdata("1", "x^2+2", "x=[-2,2]");
Plotdata("2", "-(x-2)^2+1","x=[0,4]");
plist=Nearestpt("gr1","gr2");
Listplot("1",plist_1,plist_3);
pB=plist_3;
```

Remark The returned list is [[0.4,2.16],31,[1.58,0.82],20.73,1.78].



⇒Command List

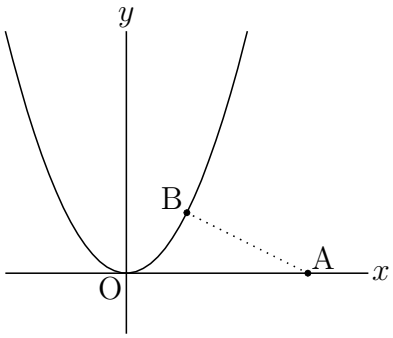
Nearestptcrv

Usage Nearestptcrv(point1, PD);

Description Generic funtion to return the nearest point on the PD from the point1.

Examples

```
Plotdata("1", "x^2", "x");
tmp=Nearestptcrv(A,"gr1"); //Tjhe coordnates will be returned.
Putpoint("B",tmp);
Listplot([A,B],["do"]);
```



Remark The return value is [[0.4,2.16],31,[1.58,0.82],20.73,1.78].

⇒Command List

Numptcrv

Usage Numptcrv (PD)

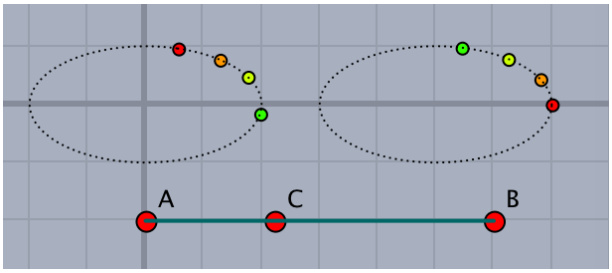
Description Generic function to return the number of PD.

Details This is the same as `length(PD)`.

Examples

Compare the order of PD, Implicit() and Paramplot(). (on Euclidean view)

```
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)]); //display number of PD on console
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));
);
```



⇒Command List

Paramoncurve

Usage Paramoncurve(point, PD)

Description Generic function to return the parameter value of the point on the curve.

Details The integer part is the number of the segment on which the point lies, the fractional part is the position on the segment.

Examples

```
Listplot([A,B,C,A]);
Putonseg("D",[B,C]);
tmp=Paramoncurve(D,"sgABCA");
println(tmp); // for example display 2.35 on console.
```

⇒Command List

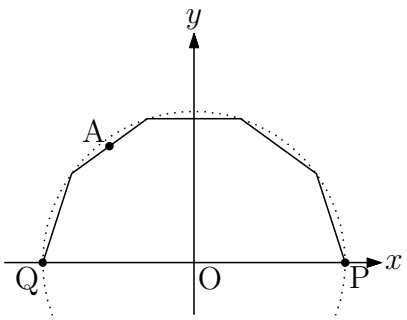
Pointoncurve

Usage Pointoncrv(parameter value, plotting data);

Description Generic function to return the point which has the parameter value

Examples

```
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");
```



⇒Command List

Ptcrv

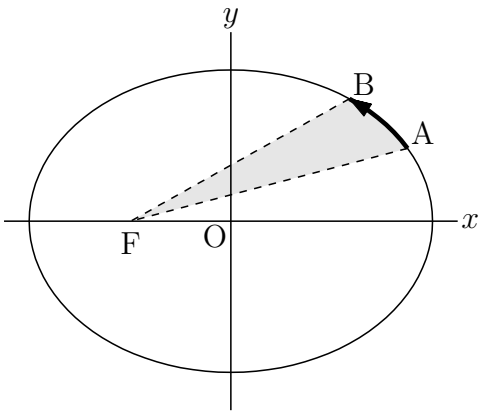
Usage Ptcrv(n,PD);

Description Returns n-th point from PD.

Details Same as `PD_n` of Cindyscript.

Examples

```
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"]);
```



⇒Command List

Ptstart, Ptend

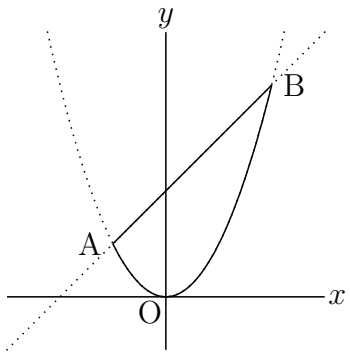
Usage Ptstart(PD) , Ptend(PD)

Description Returns start point and end point of PD. respectively.

Details It returns coodinates of point.

Examples Gets the points at both ends of the graph with limited domain and draw the line segments.

```
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"]);
```



⇒Command List

Readcsv

Usage Readcsv(path,filename,option)

Description read an external data file in csv format. The return value is a list of the data.

Details The first argument sets a path to the current working folder where the data file is (the default is fig). If you put the data file in fig folder, the pathname can be omitted. Otherwise a full pathname is required.
option: By the argument "Flat=y", you can flatten a list of the data (the default is "Flat=n").

Examples Examples can be found in the command Boxplot().

⇒Command List

Readlines

Usage Readlines(path,filename,option)

Description read a text file line by line. The return value is a list of strings.

Details The first argument sets a path to the current working folder where the data file is (the default is fig). If you put the data file in fig folder, the pathname can be omitted. Otherwise a full pathname is required.
option: By the argument "Flat=y", you can flatten a list of the data (the default is "Flat=n").

⇒Command List

ReadOutData

Usage ReadOutdata(filename);

Description Generic function to read external data of KETCindy format.

Details If the data is outside the working directory, add the path name as the first argument. For example,

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

KETCindy format data is next style.

```
variable name//
start //              : start of list
[ , , ], .... //        : coordinates ( 2 or 3 dimension )
....
end//                 : end of list
start//               : start of next list
....
end//
variable name//
start//
…
end////
```

Reference WriteOutData.

⇒Command List

WriteOutData

Usage WriteOutdata(filename, a list of varname and value);

Description Function to write out data in KETCindy format.

Details The file is available commonly from KETCindy, R and C.

Examples

Write out the plotting data of the parabola and the circle.

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

The written data is as follows.

```
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]]//
```

and so on

```
[[5.82965,33.98479]]//
end//
```

```
cr1//
start// [[1,0],[0.99211,0.12533],[0.96858,0.24869], ,[0.80902,0.58779]]//
```

and so on

```
[[0.87631,-0.48175],[0.92978,-0.36812], ,[1,0]]//
end////
```

Reference See ReadOutData.

⇒Command List

Extractdata

Usage Extractdata(dataname,options);

Description Function to add properties to a data.

Details The default properties are [”dr”].

Examples

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

Reference See WriteOutData and ReadOutData.

⇒Command List

1.5 Making Tables

Tabledata

Usage Tabledata(a list of widths, a list of height,a list of removals,[options]);

Descritption Table function to draw rules of a table.

Details The lower left is the origin.

The options are

”Setwindow=y/n” if ”n”, command setwindow is not executed.

”Move=point” The lower left changes to the point.

”Geo=n/y” if ”y”, geometric points are created.

integer decides to put names par each interval.

The unit of length is 1/10 of the grid of Euclidean view. The default is 1mm.

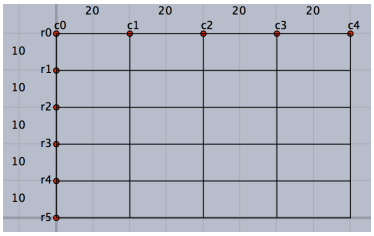
Control points are put on the row and column. The names are r0,r1,... and c0,c1,....

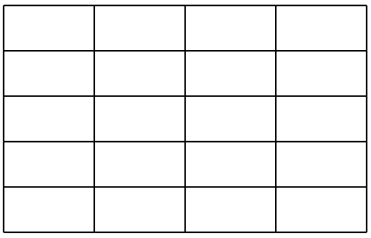
The points are movable.

Remark See `Tabledatalight`

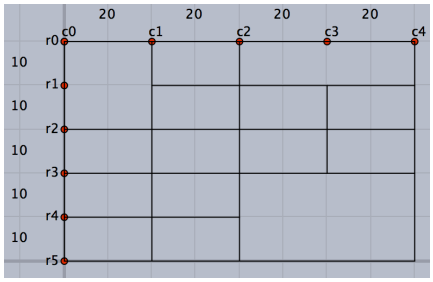
Examples

```
xL=[20,20,20,20];
yL=[10,10,10,10,10];
Tabledata(xL,yL,["Geo=y"]);
```





```
Rmv=["r1c0c1","c3r0r1","c3r3r5","r4c2c4"];
Tabledata(xL,yL,Rmv);
```



⇒Command List

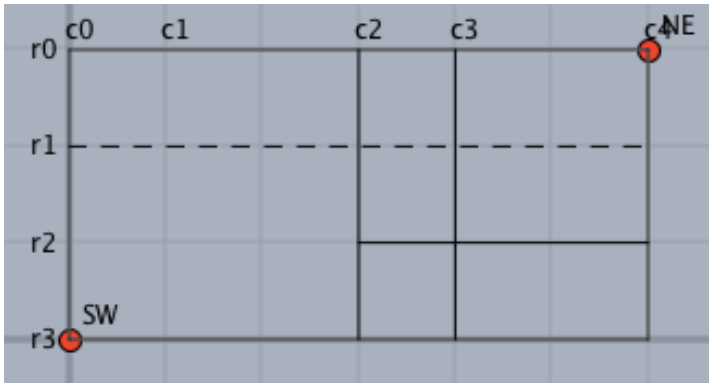
Changetablestyle

Usage ChangeTablestyle(a list of Rules, [changed style]);

Description Table function to change line styles of rules.

Examples

```
Tabledata([10,20,10,20],[10,10,10],[]);
ChangeTablestyle(["r1c0c4"],["da"]);
ChangeTablestyle(["r2c0c2","c1r0r3"],["nodisp"]);
```



⇒Command List

Findcell

Usage Findcell(grid name of upper left, grid name of lower right);

Description Table function to return the information of a cell.

Details The grid name is, for example, "c0r1".
The result is a list of center, half of width, half of height.

Examples

```
Tabledata([10,20,10,20],[10,10,10],[]);
tmp=Findcell("c2r0","c3r1"); The return is [[3.5,2.5],0.5,0.5].
tmp=Findcell("c0r1","c2r3"); The return is [[1.5,1],1.5,1] .
```

⇒Command List

Putcell, Putcellexpr

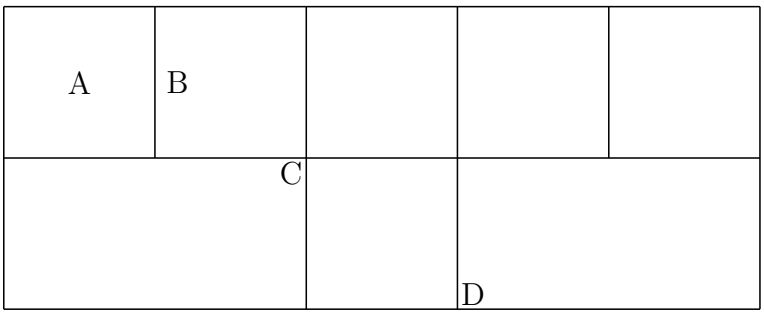
Usage Putcell(grid name of upper left, grid name of lower right, postion, a string);

Description Table function to put a string at the cell.

Details The position is one of `c, r, l, t, b` (center, right, left, top, bottom).
Minute movements can be added.

Examples

```
xL=apply(1..5,20);
yL=apply(1..2,20);
rL=["c2r2r3","c5r2r3"];
Tabledata(xL,yL,rL);
Putcell("c0r0","c1r1","c","A");
Putcell("c1r0","c3r1","l2","B");
Putcell("c0r1","c2r2","rt","C");
Putcell("c3r1","c5r2","lb","D");
```



⇒Command List

Putcol

Usage Putcol (column number, position, a list of strings);

Description Table function to put strings to a column.

Details The position is as `Putcell`.
It's unnecessary to enclose with double quotes in case of numbers.
Null string is available.

Reference Putrow.

⇒Command List

Putcolexpr

Usage Putcolexpr (column number, position, a list of mathematical expressions);

Description Table function to put strings to a column.

Reference Putrowexpr.

⇒Command List

Putrow

Usage Putrow (row number, position, a list of strings);

Description Table function to put strings to a row.

Reference Putcol.

⇒Command List

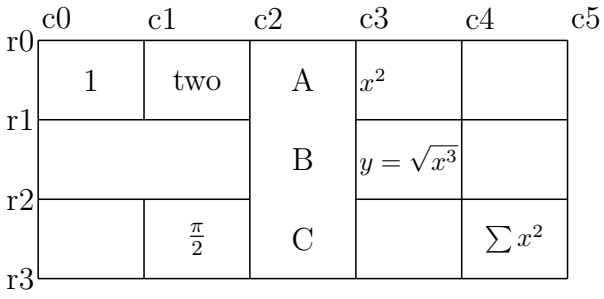
Putrowexpr

Usage Putrowexpr(row number, position, a list of strings);

Description Table function to put strings to a row.

Examples In Putcolexpr (), Putrowexpr (), formulas and general TEXsentences can be entered.

```
xL=apply(1..5,20);
xL=apply(1..3,15);
Tabledata(xL,yL,["c1r1r2","r1c2c3","r2c2c3"]);
Putcol(3,"c",["A","B","C"]);
Putcolexpr(4,"l",["x^2","y=\sqrt{x^3}"]);
Putrow(1,"c",[1,"two"]);
Putrowexpr(3,"c",["","\frac{\pi}{2}","","","\sum{x^2}"]);
```

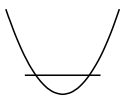
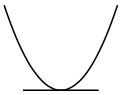
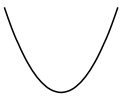


Remark r0, c0, ... are numbers displayed on the screen.

Examples The graphs can be placed in the cells of the table. These are drawn at the position of the cell.

```
Tabledata("",3,3,120,90,["dr,2"]);
ChangeTablestyle(["r1c0c3"],["dr"]);
ChangeTablestyle(["r2c0c3"],["da"]);
Plotdata("1","(x-2)^2+1.5","x=[0.5,3.5]");
Plotdata("2","(x-6)^2+2","x=[4.5,7.5]");
Plotdata("3","(x-10)^2+2.5","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","1","0"]);
Letter(G,"c","The discriminant and the number of intersections");
```

The discriminant and the number of intersections

$D > 0$	$D = 0$	$D < 0$
2	1	0
		

Examples

```
Column=[6,6,10,6,10,6,40];
Row=[30,6,6,6];
Rmv=["c1r0r1","c2r0r1","c3r0r1","c4r0r1","c5r0r1", "r1c6c7","r2c6c7","r3c6c7"];
Tabledata("",Column,Row,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$\vspace{1mm}\\\hspace*{5mm}$x=
\dfrac{1}{4}$\vspace{1mm}The following table is obtained.\\\end{minipage}}" );
Plotdata("1","x-sqrt(x)","x=[0,3]",["do","notex"]);
Listplot("2",[[0,0],[3,0]],["do","notex"]);
Listplot("3",[[0,-0.5],[0,3]],["do","notex"]);
Translatedata("1","gr1",[4.9,1],["dr"]);
Translatedata("2","sg2",[4.9,1],["dr"]);
Translatedata("3","sg3",[4.9,1],["dr"]);
Letter(Ptend(tr2),"e1","\small{$x$}");
Letter(Ptend(tr3),"n1","\small{$y$}");
Letter(Ptstart(tr2),"w1","\small O");
Expr(Ptend(tr1),"nw-2","y=x-\sqrt{x}");
```

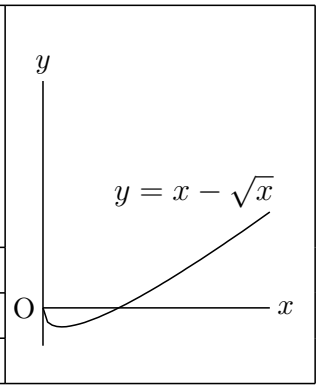
$y = x - \sqrt{x}$

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

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

The table is as follows.

x	0	$\cdots$	$\frac{1}{4}$	$\cdots$	4
y'		$-$	0	$+$	
y	0	↘	$-\frac{1}{4}$	↗	2



Examples

```
Tabledata("",8,4,80,40,[]);
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	↗	$\frac{1}{\sqrt{e}}$	↷	1	↘	$\frac{1}{\sqrt{e}}$	↘

Remark The arrows here are defined in ketpic.sty.
nelarrow, nerarrow, selarrow, serarrow, NELarrow, NERarrow, SELarrow, SERrarrow
The first ne and se represent northeast and southeast (upper right and lower right), respectively. The next r and l represent the direction of rotation (r: right: counterclockwise, l: left: clockwise).
The straight arrows are NEarrow, SEarrrow. Since these arrows do not exist in CindyTEX, they are not displayed on the drawing surface of Cinderella.

⇒Command List

Tgrid

Usage Tgrid(grid name);

Description Table function to return the coordinates of the grid name.

⇒Command List

Tlistplot

Usage Tlistplot(grid name, grid name);

Description Table function to connect two lattice points by line segments.

Examples

```
Tlistplot(["c0r1","c1r2"]);
```

⇒Command List

1.6 Data Processing

This section describes data processing by KETCindy. Cooperation with spreadsheet software enables efficient data processing.

Tab2list

Usage Tab2list(string data, option);

Description Sheet function to convert contents of string data to list.

Ditails The options are as follows.

`"Blank=a"` : translate cells that is NULL to “a”

`"Sep=b"` : separators of the string are “b”. The default separators are Tab code.

Examples

In the Cindyscrpt editor, prepare a local variable, for example ”data”.

```
1 Ketinit();
2 Setfiles("DNA");
3
4 data="";
5
6 Windispg();
```

Copy the data on the spreadsheet to the clipboard.

	A	B	C	D	E
1		A	T	G	C
2	colon bacillus	24.7	23.6	26	25.7
3	wheat	27.4	27.1	22.7	22.8
4	salmon	29.7	29.1	20.8	20.4
5	human	30.9	29.4	19.9	19.8
6					

Paste it during double quotes.

```
4 data="  A T G C
5 colon bacillus  24.7  23.6  26  25.7
6 wheat 27.4  27.1  22.7  22.8
7 salmon  29.7  29.1  20.8  20.4
8 human 30.9  29.4  19.9  19.8
9 ";
```

By executing `"Tab2list(data)"` get a list of matrix form.

```
10 dlist=Tab2list(data);
11 println(dlist);
```

```
/kc.sh executable
[[,A,T,G,C],[colon bacillus,24.7,23.6,26,25.7],[wheat,
27.4,27.1,22.7,22.8],[salmon,29.7,29.1,20.8,20.4],[human,
30.9,29.4,19.9,19.8]]
```

If it contains a null character cell (NULL), it defaults to null character.Therefore, if you want to set NULL to 0 for questionnaire processing etc., use option `Blank`.

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

When CSV format data is copied from the file, the option is set to `sep`.

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

⇒Command List

Dispmat

Usage Dispmat(list);

Description Display the list to matrix form in the console.

Examples In the example of Tab2list, put the obtained data in a matrix format.

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

```
/kc.sh executable
          A       T       G       C
colon bacillus    24.7    23.6    26      25.7
wheat     27.4    27.1    22.7    22.8
salmon    29.7    29.1    20.8    20.4
human     30.9    29.4    19.9    19.8
```

You can copy this directly to spreadsheet.

⇒Command List

Writecsv

Usage Writecsv(namelist, data, filename, option);

Description Make a CSV file consisting of the contents of data.

Ditails namelist is item name added to the first line of the CSV file. If the namelist omitted, the item names ”c1, c2, ...” are appended.

The filename is the name of CSV file.

option : `"Col=nn"` : Specify the number of columns in the CSV file as a natural number `nn`.

When specifying the number of columns is omitted, if data is a matrix, use that number of columns, and if data is a vector, use the number of items in namelist.

Examples

Let `data=[13,25,17,22,14,26] , name2=["aa","ab"] , name3=["ba","bb","bc"]`

```
Writecsv(name2,data,"aaa");
```

makes the file ”aaa.csv” consists of
aa,ab
13,25
17,22

14,26

`Writecsv(name3,data,"aaa");`
makes the file ”aaa.csv” consists of
ba,bb,bc
13,25,17
22,14,26

`Writecsv(,data,"aaa",["Col=3"]);`
makes the file ”aaa.csv” consists of
c1,c2,c3
13,25,17
22,14,26

⇒Command List

1.7 Others

Assign

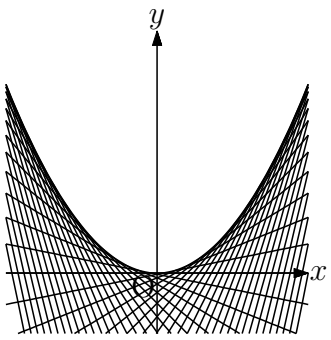
Usage Assign(string0, string1, number)

Description Generic function to replace the string1 in the string0 with the number. Number is real number or string of real number.

Examples

```
Assign("x^2+a*x","a","1.3");    // a*x → 1.3*x
Assign("x^2+a*x","a",1.3);      //same as above


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



Perform multiple replacements by list.

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

⇒Command List

BBdata

Usage BBdata(file name, options);

Description Generic function to return the size of an image file.

Details In the TEXdocument, find the BB size when pasting the image with the input-graphics command. Create BB data from an image file using extractbb of TEXprocessing system and write it as a text file to the working directory. Read this and write the in-gludegarphics command to the console.
Options : specifies width and height. "w=" : width, "h=" : height

The value of bb is not an integer value, and it is indicated by rounding off the high definition value to two decimal places.
The image files are PDF, PNG, JPG, and so on.

Examples

```
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}
```

⇒Command List

Asin

Usage Asin(real) , Acos(real)

Description Return arcsine and arccosine.

⇒Command List

Sqr

Usage Sqr(real)

Description Return square root.

⇒Command List

Colorcode

Usage Colorcode(colortype1,colortype2,colorcode)

Description Generic function to change colorcode from colortype1 to colortype2.

Details Return value is changed color code.

Color type is one of ”rgb”,”cmyk”,”hsv”.

Example

RGB to CMYK

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

CMYK to RGB

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

RGB to HSV

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

⇒Command List

Dqq

Usage Dqq(string);

Description This function returns a string surrounded by double quotes.

Exaample

```
parse("a"); // The value of variable a is returned.
parse(Dqq("a")); // String "a" is returned.
```

⇒Command List

Factorial

Usage Factorial(n);

Description This function returns the factorial of `n`.

[Details]`n` should be a positive integer.

Exaample
```
x=Factorial(5); // x is 120.
```

⇒Command List

Figpdf

Usage Figpdf(option)

Description Generic function to make a pdf with the same size of figure.

Details Option is a list of margin and the amount of translation.

(1) Set the output file name with the command `Setparent("filename")`.

(2) Push the "Parent" button.

"figure.tex" and "filename.tex" is created in fig folder. (use "figure.cdy") filename.tex creates filename.pdf using figure.tex.

Examples

```
Figpdf();                    : default
Figpdf([5,5,10,10]);         : left and right margins are 5mm
                             : top and bottom margins are 10mm.
Figpdf([[5,10]]);            : translation to right 5mm and to down 10mm.
Figpdf([5,8,10,10,[5,-5]]);  : margin and translation
```

We have to take the right margin at least 3mm to draw the axis name.

⇒Command List

Cindyname

Usage Cindyname();

Description Generic function to return the name of a current file without ".cdy".

Examples

```
name=Cindyname(); // If cuurent file is "sample.cdy", name="sample".
```

⇒Command List

Indexall

Usage Indexall(string1,string2);

Description Generic function to return all positions of string2 in string1.

Examples

```
str="abcadeaf"
pos=Indexall(str,"a");// Result is [1,4,7].
```

Remarks This command is an extension of "indexof" which is a command of CindyScript.

⇒Command List

Help

Usage Help(string)

Description Generic function to display usages of the function.

Examples

`Help("L");` then we have the following result in console.

```
Letter([C,"c","Graph of $f(x)$"]);
Letter([C,"c","xy"],["size->30"]);
Letterrot(C,B-A,"AB");
Letterrot(C,B-A,"t0n5","AB");
Letterrot(C,B-A,0,5,"AB");
.........
```

⇒Command List

Norm

Usage Norm(vector);

Description This function returns the norm of the vector.

Details The vector is 2D or 3D.
If two vectors `v1`, `v2` are given, the value of `Norm(v2-v1)` is returned.

Examples `Norm([1,1,2]);` // $\sqrt{6}$ is returned.

⇒Command List

Op

Usage Op(number, list or string);

Description Generic function to return the n-th element of a list or a string.

Examples

```
str="abcde"
list=[3,1,2,5];
s=Op(2,str);      // Result is "b".
x=Op(3,list);      // Result is 2.
```

⇒Command List

Ptselected

Usage Ptselected(name of points)

Description Generic function to returns "true" if a point is selected.

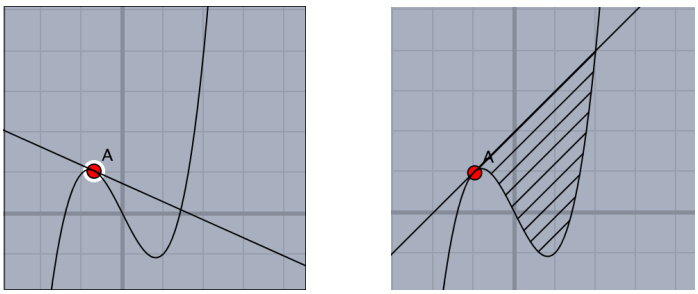
Details Commands such as `Hachdata` take time to execute, so interactive operations slow down the reaction. Therefore, while interactively operating, you can use this command to stop drawing.

Examples

Draw the point A near the origin.

```
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"]]);
);
```

Dragging point A (select) Unselected



⇒Command List

Reparse

Usage Reparse(string or list of string)

Description function to return the real part after `parse`

Remark `parse` of CindyJS has a bug to return an imaginary number in some cases.

Examples

```
str="(0-1)^2";
format(parse(str),0); // returns 1+i*0 in CindyJS
format(Reparse(str),0); // returns 1
```

⇒Command List

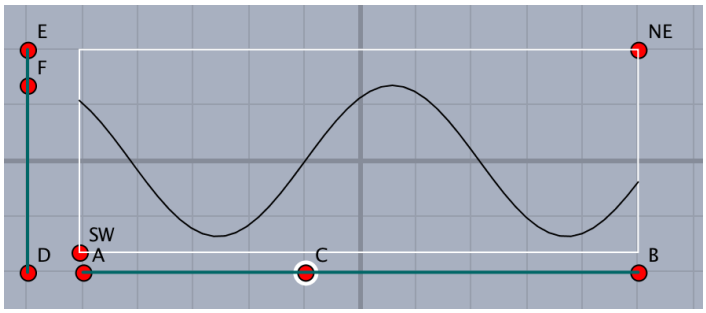
Slider

Usage Slider(”endpoint1-pt-endpoint2”,endpoint1,endpoint2);
Slider(”pt”,endpoint1,endpoint2);

Description Generic function to make a slider on a Euclidean view.

Examples

```
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");
```



⇒Command List

Sprintf

Usage Sprintf(value,number);

Description Converts a real number to a string.

Details Convert a real value to a string to the specified number of digits after the decimal point.

Examples

```
Sprintf(pi,2); // returns "3.14".
Sprintf(pi,7); // returns "3.1415927".
```

`Remark` : `pi` is a reserved variable in Cindyscript, representing the number π.

Reference See Textformat.

⇒Command List

Strsplit

Usage Strsplit(string,char);

Description Generic function to return the list of strings separated by `char`.

Examples

```
str="abcadeaf"
strL=Strsplit(str,"a");      // Result is ["","bc","de","f"].
```

⇒Command List

Texcom

Usage Texcom(command);

Description Generic function to add the command in the TEXfile.

Details Command is a TEXcommand in string.

Examples

```
Texcommand("{");
Texcommand("}");
```

⇒Command List

Textformat

Usage Textformat(value,number);

Description Converts a real number to a string.

Details Convert a real value to a string up to the specified number of digits after the decimal point. ”value” is can be list.

Cindyscript has a function `format(value, number)`, like as Textformat.

Examples

```
Textformat(1/6,4); // return value is string ”0.1667”
format(1/6,4);     // return value is string ”0.1667”

dt=[1/6,0.5];
Textformat(dt,4);  // return valu is string ”[ 0.1667 , 0.5 ]”
format(dt,4);      // return value is list [”0.1667” , ”0.5” ]
Sprintf(dt,4);     // return value is list [”0.1667”,”0.5000”]
```

Reference See Sprintf.

⇒Command List

Toupper

Usage Toupper(string);

Description This function returns the upper case letters of the string.

Examples

```
Toupper("aBc123"); // ”ABC123” is returned.
```

⇒Command List

Windispg

Usage Windispg();

Description Generic function to display all graphs on Euclidean view.

Remark This command must be put on the final line.

⇒Command List

Windispg

Usage Windispg();

Description Generic function to display all graphs on Euclidean view.

Remark This command must be put on the final line.

⇒Command List

Fracform

Usage Fracform(number,list of denominators/max number[, allowable error(5)])

Description returns Tex-like form of the fraction

Examples `Fracform(0.33,[2,3]); => [fr(1,3),''err=0.00333'',1,3]`

⇒Command List

Totexform

Usage Totexform(Tex-like form)

Description returns TeX form

Examples `Totexform(fr(1,3)); => frac{1}{3}`

⇒Command List

Tocindyform

Usage Tocindyform(Tex-like form)

Description returns Cindy form

Examples `Tocindyform(fr(1,3)); => (1)/(3)`

⇒Command List

2 Calling Other Softwares

2.1 R

Rfun

Usage Rfun(name, ommand,list of arguments,options)

Description This function executes a R command and returns the.

Examples `Rfun("1","rnorm",[10]);` // The result will be assigned to "R1".

Remark Option "Cat=n" supresses display of the result in the console.

⇒Command List

CalcbyR

Usage CalcbyR(var,command,options)

Description executes R commands and returns the execution result to Cinderella.

Details exchange data with R through a batch file (kc.bat) or a shell file (kc.shell).

Examples

Generate 10 random samples from the standard normal distribution by R and return the result (data) to Cinderella.

```
cmdL=[
  "dt=rnorm",[10,50,5],
  "m=mean(dt)",[],
  "u=var(dt)",[],
  "dt::m::u",[]
 ]
CalcbyR("ans",cmdL);

println("Data : "+ans_1);
println("Mean : "+format(ans_2,4));
println("UD : "+format(ans_3,4));
```

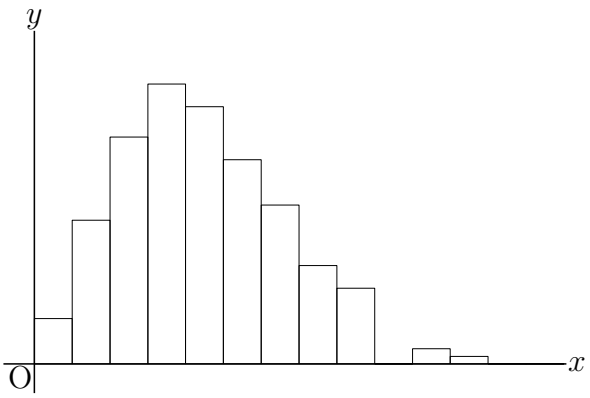
generate 200 random samples from the Poisson distribution with mean 5 and calculate the sample mean and the unbiased variance of the data.

```
cmdL=[
  "tmp1=rpois",[200,5],
  "tmp2=mean",["tmp1"],
  "tmp3=var",["tmp1"],
  "tmp2::tmp3::tmp1",[]
];
CalcbyR("rd",cmdL);
dt=rd_(3..length(rd));
nn=length(dt);
mx=rd_1;
vx=rd_2*(nn-1)/nn;
sx=sqrt(vx);
```

```
println(dt);
println(["m="+format(mx,4),"v="+format(vx,4)]);
Setscaling(1/5);
```

create a histogram for the data, `Breaks=seq(0,14,1)` specifies the bin size.

```
Histplot("1",dt,["Breaks=seq(0,14,1)","dr,0.5"]);
```



generate 2000 random samples from the Poisson distribution and calculate 200 sample means in 10 samples.

```
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);
```

⇒Command List

Boxplot

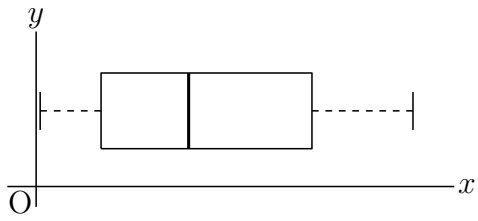
Usage Boxplot(name, data, vertical position, height of box,options);

Description draw boxplots

Examples

draw a boxplot of 100 uniform random numbers less than 5.

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

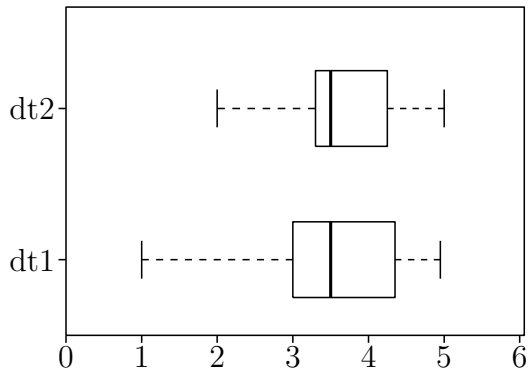


read an external data file in csv format and draw a boxplot of the data.

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

You can read a csv file with more than one column using `Readcsv`. The csv flie should be stored in current working folder (defalut is fig folder). Using `Framedata` and `Rulerscale` together, you can mark with a scale. Before you use `Framedata`, you need to take two diagonal points of the drawing area on the Euclidean view.

```
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}"]);
```



⇒Command List

Histplot

Usage Histplot(name,data,options)

Description create histograms.

Details data is given in a list or read an external data file in csv format.
Return value is list of breaks and frequency.

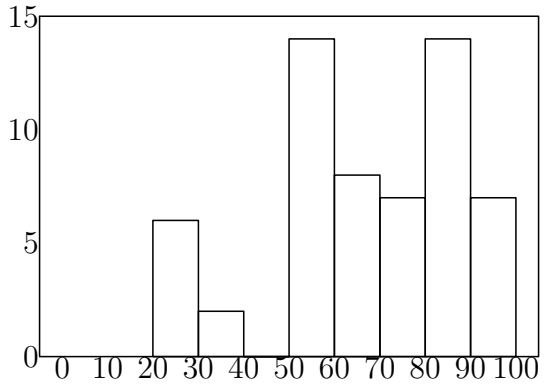
You can specify the breaks as a vector of points to get exactly what is wanted, for example `"Breaks=[0,10,20,30,40,50,60]"` .
The Sturges algorithm is the default.
Other options:
`"Rel=yes/no"` : draw a histogram of proportions or frequencies (default is no)

Examples

Read the data file in csv format (datafile.csv) and create a histogram of the data in a frame with a scale.

```
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]);
```



⇒Command List

PlotdataR

Usage PlotdataR(name,formula,var)

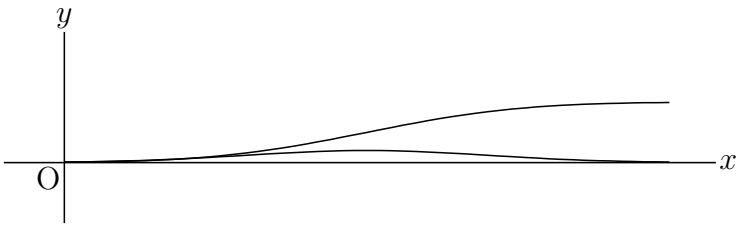
Description Draw graph of R's statistical probability function.

Details Draw graphs of functions not built-in Cindyscript.

Examples

Example1
draw graphs of the probability density function (p.d.f.) and the cummulative distribution function of $N(5, 2^2)$.

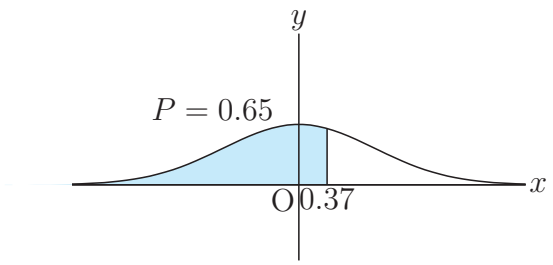
```
PlotdataR("1", "dnorm(x,5,2)", "x=[0,10]");
PlotdataR("2", "pnorm(x,5,2)", "x=[0,10]");
```



Example2
1. draw a graph of the p.d.f. of standard normal distribution.
2. shade the region under the graph and above x-axis to the left of `A.x`.
3. find the area of the shaded region.

```
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)]);
```



⇒Command List

PlotdiscR

Usage PlotdiscR(name,fromaula,var)

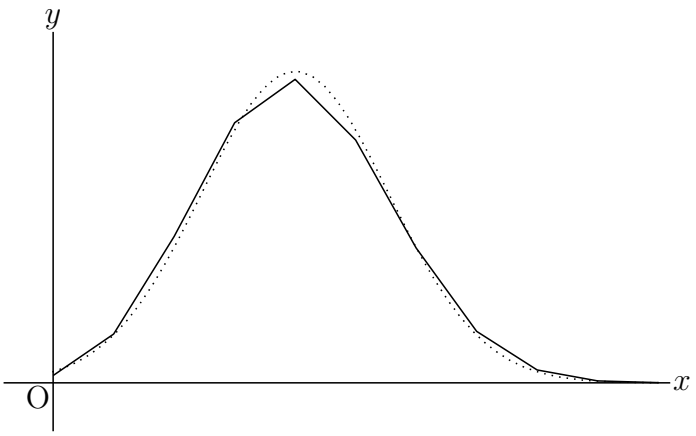
Description draw graphs of discrete distributions by calling R's built-in functions.

Details The "d" functions in R to draw graphs of discrete distributions:
dbinom (binomial distribution), dpois (poisson distribution), dgeom (geometric distribution), etc.

Examples

The normal distribution with the same mean and standard deviation as the binomial distribution

```
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"]);
```



Example2

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

⇒Command List

Scatterplot

Usage Scatterplot(name,filename/datalist,options1,options2)

Description This command draw scatter plot reading a csv file.

Details Datafile is next style csv format.

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

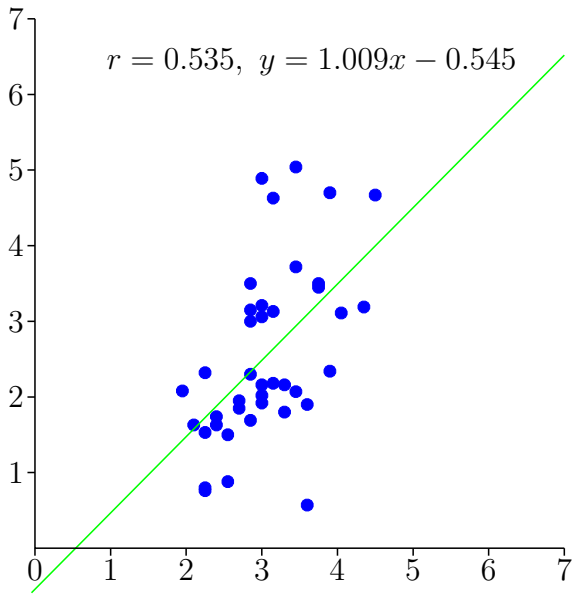
If 2nd argment is datalist, next format.

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

Options1 are switch of draw the regression line or no , style of point.
`"Reg=yes(no:default)"` to decide whether to draw the regression line.
Options2 are position of drawing the regression line and style of line.
Position is coordinate or name of point.

Examples

```
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]);
```



⇒Command List

2.2 Maxima

CalcbyM

Usage CalcbyM(name, command,options)

Description Maxima's script execution

Details The second argument is a command to be executed by Maxima.

Create a list (eg. cmdL) consisting of a repetition of commands and argument lists, and execute at once.

There is no return value. For the result (of undefined value), the value of the variable of the command list last described (argument is the empty list) is assigned to the variable specified by "name". When you want to return more than one result, if you describe it by separating it with "::", it will be substituted into the list.

Examples

Example1: derivative

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

Example2: solution of quadratic equation

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

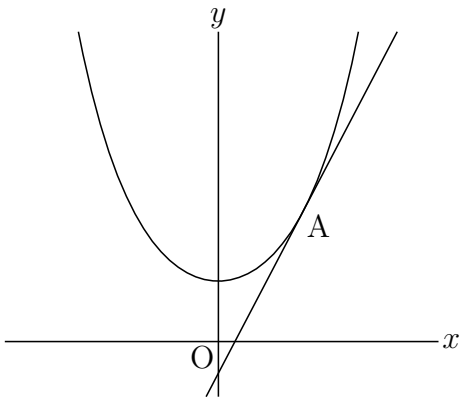
Example3:

```
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);
tn1=Assign(tn1,["%e^a","exp(a)","%e^-a","exp(-a)"]);
Plotdata("1",fx,"x");
PutonCurve("A","gr1");
tmp=Assign(tn1,["a",A.x]);
plotting data("2",tmp,"x",["Num=2"]);
Letter([A,"se","A"]);
```

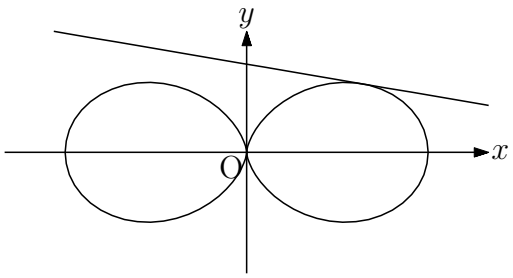
Example4: Parametric

```
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]");
```



Example3



Example4

⇒Command List

Mxbatch

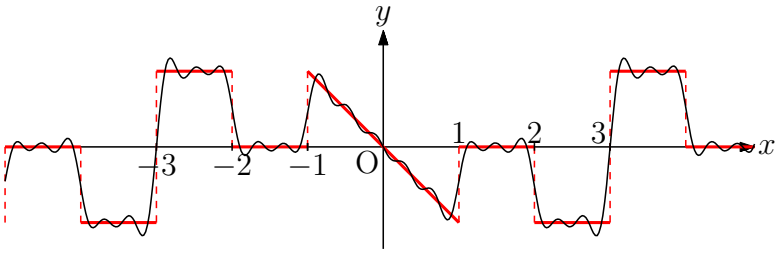
Usage Mxbatch(filename)

Description Creation command to execute Maxima file

Details Create a command for CalcbyM to execute the file in ketcindy/ketlib/maximaL. ketcindy/ketlib/maximaL contains three files: fourier_sec.max, matoperation.max and poincare.mac. For example, when dealing with Fourier series, use fourier_sec.max.

Examples

```
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)]);
```



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

⇒Command List

Mxfun

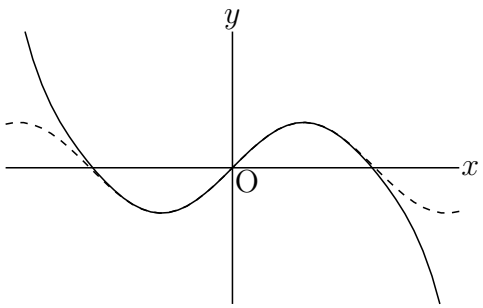
Usage Mxfun(name,formula,list,options)

Description Execution of Maxima's function

Details The second argument "formula" is Maxima's function name. The third argument "list" is a list of arguments to pass to the function.
The return value is a string if there is at least one character in the expression of the second argument. In the case of all numbers (including +, -, .), it becomes a number if it is 16 or less digits, and it becomes a string if it is more than 16 digits. Also, the return value is also assigned to the variable "mxname".
If "Disp = no" is added to the option, the result is not displayed on the console.

Examples

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



⇒Command List

Mxtex

Usage Mxtex(name, formula)

Description Conversion of expression to TeX format

Details The second argument "formula" is the expression directly written or the return value of Mxfun. Convert it to TeX format.
The return value is also assigned to the variable "txname".

Examples

Example1

```
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"]);
Expr([0,3],"e",form);
Expr([0,1],"e",dform);
```

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

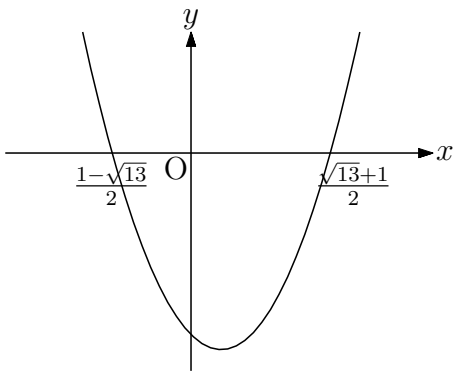
$$x^3(x+1)(x+2)=8x+2-1x+1+x-3$$

Decomposition into partial fractions

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

Example2

```
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);
```



⇒Command List

2.3 Risa/Asir

CalcbyA

Usage CalcbyA(name, command,options)

Description Risa/Asir's script execution

Details The second argument is a command to be executed by Risa/Asir.
Create a list (eg. cmdL) consisting of a repetition of commands and argument lists, and

execute at once.
There is no return value. The result (of undefined value) is assigned to the variable specified by "name", the value of the variable of the command list last described (argument is the empty list). If you want to return more than one result, if you describe it by separating it with "::", it will be substituted into the list.

⇒Command List

Asirfun

Usage Asirfun(name, formula, list,options)

Description Execution of Risa/Asir's function

Details The second argument "formula" is the function name of Risa/Asir. The third argument "list" is a list of arguments to pass to the function.
The return value is a string if there is at least one character in the expression of the first argument. In case of all numbers (including +, -, .), it becomes a number if it is 16 digits or less, and it becomes a string if it is more than 16 digits. Also, the return value is also assigned to the variable "asname".
If "Disp = no" is added to the option, the result is not displayed on the console.

⇒Command List

2.4 MeshLab

Write next script in Initialization slot for use KETCindy 3D.

```
Ketinit();
Ketinit3d();
```

Mkobjcmd

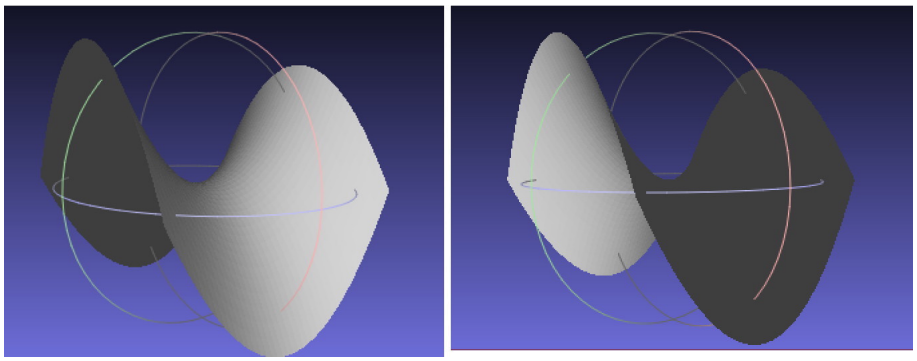
Usage Mkobjcmd(name,formula,option)

Description generate commands for obj formatted files of surfaces without thickness.

Examples

```
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"]);
);
```

Option "+" is for the left figure, and "−" for the right.



⇒Command List

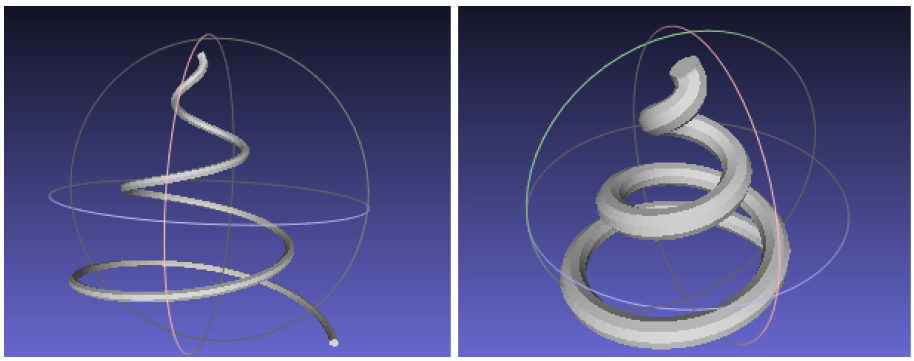
Mkobjcrvcmd

Usage Mkobjcrvcmd(name,PD,option)

Description generate commands for obj formatted files of spatial curves.

Examples

```
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",[0.1,8,"yz"]);
Meshlab():=(
Mkviewobj("spiral",oc1,["m","v"]);
);
```



⇒Command List

Mkobjnrm

Usage Mkobjnrm(name,formula)

Description calculate normal vector of surface.

Details Normal vector is calculated using the formula of surface.

Examples

```
Mkobjnrm("1","[x,y,x*y/sqrt(x^2+y^2)],x,y");
```

⇒Command List

Mkobjplatecmd

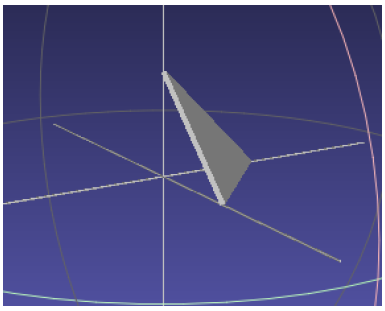
Usage Mkobjplatecmd(name,facedata,options)

Description generate commands for obj formatted files of plates.

Examples

```
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"]);
```



⇒Command List

Mkobjpolycmd

Usage Mkobjpolycmd(name,PD,options)

Description generate commands for obj formatted files of polyhedra.

Examples

```
Setdirectory(Dirhead+"/data/polyhedrons_obj");
polydt=Readobj("r01.obj",["size=-3.5"]);
Setdirectory(Dirwork);
pd=VertexEdgeFace("1",polydt,["Pt=fix","Edg=nogeo"]);
Mkobjpolycmd("1",pd,[[0,0,0]]);
Mkviewobj("plane",oc1,["m","v"]);
```

The polyhedron obj data is downroaded from
http://mitani.cs.tsukuba.ac.jp/polyhedron/

⇒Command List

Mkobjsymbcmd

Usage Mkobjsymbcmd(PD,real,real,vector,vector)

Description generate commands for obj formatted files of some characters.

Details Ploting data are available for characters x, y, z, t, n, P, Q, and R. The arguments are their siizes, angles of rotations, directions of the viewpoints, positions.

Examples

```
Mkobjsymbcmd("P",0.5,pi/3,[0,-1,0],[0,0,6]);
Mkobjsymbcmd("x",0.5,0,[0,-1,0],[6,0,0]);
Circledata("1",[[0,0],[1,0]],["nodisp"]);
Mkobjsymbcmd("cr1",0.5,0,[0,-1,0],[0,5,0]);
```

⇒Command List

Mkobjthickcmd

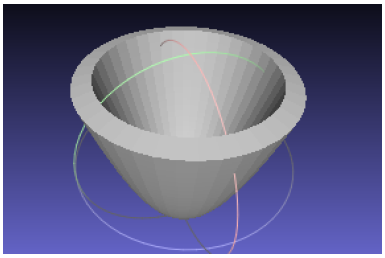
Usage Mkobjthickcmd(name,formula)

Description generate commands for obj formatted files of surfaces with thickness.

Examples

This function uses Maxima.

```
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"]);
```



⇒Command List

Mkviewobj

Usage Mkviewobj(name,PD,options)

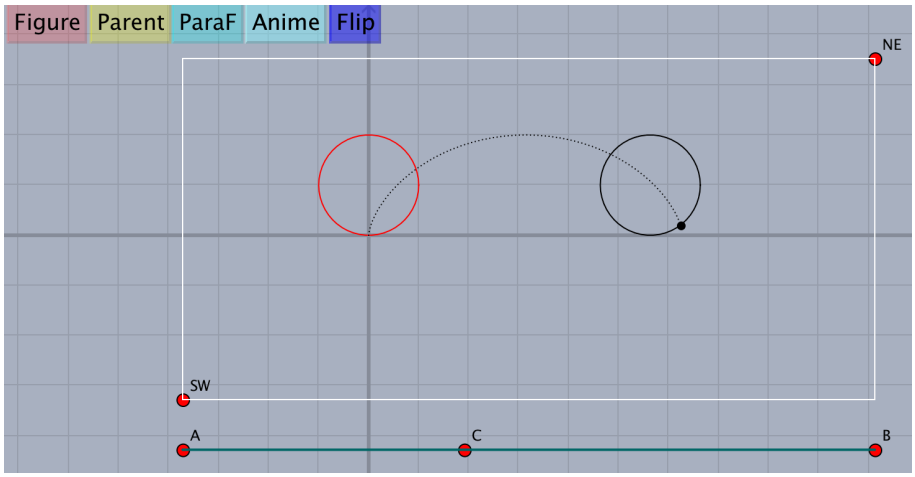
Description generate obj formatted files.

Details options

”m” or ”make”	for generating data
”v” or ”view”	for starting meshlab and viewing
”W=n” or ”Wait=n”	for setting culculate time
”Unit=mm”	for setting unit of length

⇒Command List

3 Animation



Operation of Buttons.

Figure	Viewtex();	Making figure.tex
Parent	same code	Making figure.pdf by Figpdf()
ParaF	Parafolder();	Making data folder of animation data
Anime	Mkanimation();	Making flip animation
Flip	Mkflipanime();	Making animation

Setpara

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

Description Set up the animation control system.

Details ”fname” is the name of output file. ”funcstr” is the name of animation function. ”range” is the range of parameter.

options1

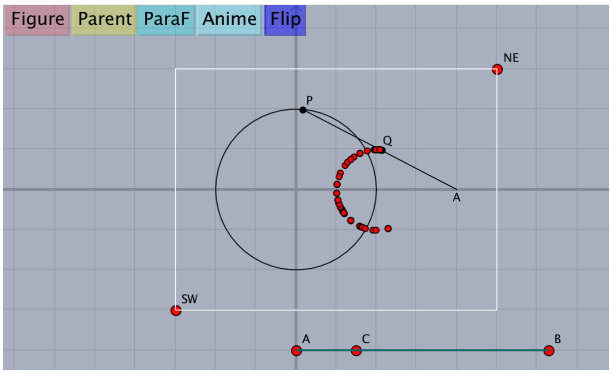
m/r	Remake the new data file / Reread the existing data file (default=r)
Div=n	Total number of frames (default n=25).

options2

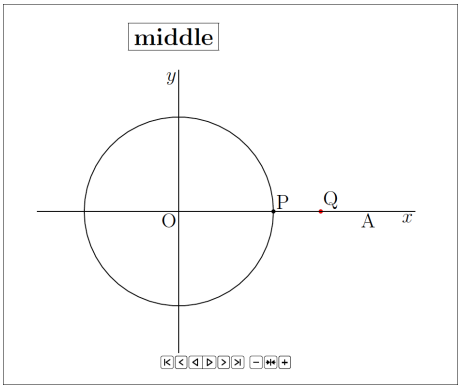
Frate=n	Number of frames per second (default n=20)
Title=str	Title
Scale=n	Magnification factor of the figures
opA	option for animate.sty loop: loop, controls: Show control button, buttonsize step: Mode of frame feed/frame retrun Default is `"OpA=[loop,controls,buttonsize=3mm]"` Use ”+” then you can add a mode, for example `"OpA=+step"` then we have `"OpA=[loop,controls,buttonsize=3mm,step]"`

Examples

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



When we make the animation, comment out `//mf(C.x);` and click the Anime button. The following figure is the first page of the animatemiddle.pdf file.



The animation is continued for 5 seconds with the following options.
`Setpara("middle","mf(t)","t=[0,4*pi]",["Div=30"],["Frate=6"]);`
A smooth-looking animation is achieved with the options: `["Div=150"],["Frate=30"]`.

⇒Command List

4 KETCindy Slide

Setslidebody

Usage Setslidebody(bodycolor,bodystyle,density)

Description Set up the color and density of the letters in slide body.

Details Meanings and defaults of options are

bodycolor	color of letters	`"blue"`
bodystyle	style of letters	`"\Large\bf\boldmath"`
density	density of thin letters	`0.1` (The range is from 0 to 1).

Remark : density can be changed by `\setthin{density}` in the text file.

⇒Command List

Setslidehyper

Usage Setslidehyper(”dvipdfmx”,options)

Description Use `hyperref.sty`.

Details if the 1st argument is null, it will be replaced with ”dvipdfmx”.

options : `["cl=true,lc=blue,fc=blue","Pos=[125,73]","Size=1"]`

Meanings and defaults of options are

”cl=...”	colorlinks	`cl=true`
”lc=...”	linkcolor	`lc=blue`
”fc=...”	filecolor	`fc=blue`
”Pos=...”	start position of buttons	`"Pos=[125,73]"`
”Size=...”	size of buttons	`"Size=1"`.

⇒Command List

Setslidemain

Usage Setslidemain([letterc,boxc,framec,xpos,size]);

Description Set up the main slide (a section delimiter).

Details Meanings and defaults of options are

letterc	color of letters	`[0.98,0.13,0,0.43]`
boxc	color of box	`[0,0.32,0.52,0]`
framec	color of frame	`[0,0.32,0.52,0]`
xpos	horizontal position of title	`62`
size	magnification of title	`2.`

Remark If some arguments are null, the defaul is used.

```
Setslidemain([,,,,3]);
```

⇒Command List

Setslidepage

Usage Setslidepage([letterc,boxc,framec,shadowc,xpos,size]);

Description Set up each page of slides.

Details Meanings and defaults of options are

letterc	color of letters	`[0.98,0.13,0,0.43]`
boxc	color of box	`[0,0.32,0.52,0]`
framec	color of frame	`[0,0.32,0.52,0]`
shadowc	color of shadow	`[0,0,0,0.5]`
xpos	horizontal position of title	`6`
size	magnification of title	`1.3`.

Remark If some arguments are null, the defaul is used.

```
Setslidepage([,"red"]);
```

⇒Command List

Setslidemargin

Usage Setslidepage([leftmarginchange,topmarginchange]);

Description This function changes the margin of slides from the default.

Example Setslidemargin([+5,-10]);

⇒Command List

Settitle

Usage Settitle(list of title components,options)

Description Make a title slide.

Details Meanings and defaults of options are

”Title=...”	name of the title file	`"Title=slide0"`
”Layery=...”	starting vertical position	`"Layery=0"`
”Color=...”	color of letters	`"Color=blue"`.

Examples

```
Settitle([
  "s{60}{20}{Main Title}",
  "s{60}{50}{Name}",
  "s{60}{60}{Affiliation}",
  "s{60}{70}{Info}"
   ],
   ["Title=SlideA","Color=[1,1,0,0]"]
);
```

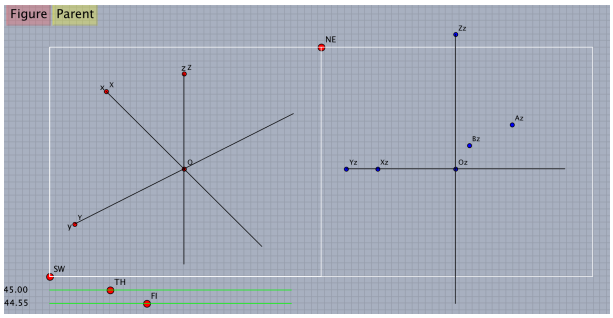
⇒Command List

5 KETCindy3D

5.1 Screen

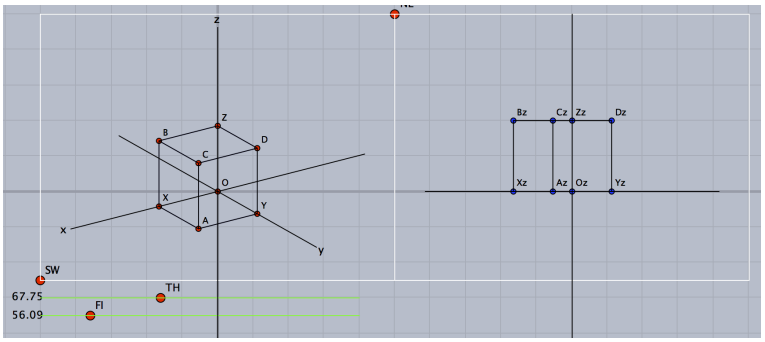
The screen of KETCindy3D is structured as follows.

There are two areas surrounded by a white rectangle on the drawing surface of Cinderella. The area on the left side where the NE and the SW are diagonal is referred to as the main screen, and the area on the right side is referred to as the sub screen.



As in the case of a flat surface, the main screen shows the range output to TeX and it can be changed by dragging two points of NE and SW. The viewpoint can be moved with the slider below the main screen, and the axis rotates on the main screen. You can think of the sub screen as a viewpoint placed on the xy plane.

When you draw points and line segments with Cinderella's drawing tool on the main screen, points corresponding to the secondary screen are drawn. You can change the x, y coordinates by dragging the point on the main screen, and drag the point on the sub screen to change the z coordinate.



KeTCindy3D performs hidden line processing on lines and surfaces. Hidden line processing speeds up processing in cooperation with C language.

It is necessary to develop an environment that uses C language, but now it is standardized. If you can not use C language, you will use a function to compute with R, but in that case it will take quite a while.

5.2 Setting and Defining

Ketinit3d

Usage Ketinit3d()

Description Declare the use of KeTCindy3D

Details Euclidean view of Cinderella becomes 3D mode. Two sliders are created to indicate the viewing angle TH(θ), FI(ϕ). The initial values are TH = 0 and FI = 0.
Caution This funcition and Ketinit() have to write on Initializaiton slot.

Remark If `Ketinit3d(0)` is used, the subscreen is not displayed. Ketinit () is also placed in the Initialization Slot, unlike 2D.

⇒Command List

Setangle

Usage Setanlge(TH,FI)

Description Specify rotation angle

Details Specify the TH and FI values of the slider that determines the rotation angle (position of the viewpoint).
For example, if you set `Setangle (70,40)` , then TH and FI are in that position. Since the slider is fixed, if you want to activate the slider again, comment it and re-execute it.
If you want to decide only the initial state

```
if (!Ptselected (), Setangle (70, 40));
```

or

```
if (!Isangle (), Setangle (70, 40));
```

The slider becomes effective.

If any point on the slider is selected, the figure button is also valid. Click anywhere on the screen and return to the original if you deselect the point selection state.

⇒Command List

Getangle

Usage Getanlge()

Description Acquire rotation angle

Details Acquires the rotation angle (viewpoint position) TH and FI that can be set with the slider. This is the value displayed on the left side of the slider. The return value is the list [TH, FI], and the angle is expressed by the Degree measure. In addition, the internal variables are THETA and PHI, expressed in circular measure.

⇒Command List

Start3d

Usage Start3d(option)

Description 3d function to initialize limited variables.

Details This funcition should be written at the beginning of Draw slot.
The option is a list of geometric points which are not regarded as 3D points.

Example
If option is given a list of exclusion points, that point is not a point of space. (The position does not change even if moving the viewpoint with the slider)

```
Start3d([A,B,C]);
Slider("A-C-B"); // A,C,B should not be 3D points.
```

⇒Command List

Startsurf

Usage Startsurf(options)

Description Defines values related to surface rendering.

Details Values are number to divide, size of C, limit of error. Omitted options selects [50,50],[1500,500,200],[0.01,0.1].
Drawing of a curved surface with hidden line processing is performed in the following procedure.
(1) Startsurf();
(2) Making data wtih draw function.
(3) Draws shapes in batch in C language using function ExeccmdC();.

⇒Command List

Xyzax3data

Usage Xyzax3data(name, range of x, range of y, range of z, options)

Description Generic function to draw the coordinate axis.

Details Name can be null string.
Options are the followings.
"an": arrowhead, n is size.
"Onesw": origin and its position.

Examples

```
Xyzax3data("","x=[-5,5]","y=[-5,5]","z=[-5,5]");
Xyzax3data("","x=[-5,5]","y=[-5,5]","z=[-5,5]","a"); //arrowhead
Xyzax3data("","x=[-5,5]","y=[-5,5]","z=[-5,5]",["a2"]); //big arrowhead
Xyzax3data("","x=[-5,5]","y=[-5,5]","z=[-5,5]",["O"]);
Xyzax3data("","x=[-5,5]","y=[-5,5]","z=[-5,5]",["a","Oe2n2"]); //set origin upper right
```

⇒Command List

5.3 Command for Drawing

5.3.1 Point and line

Drawpoint3d

Usage Drawpoint3d(list of coordinates)

Description Generic function to draw 3D-points.

Details These points are not geometric point. To convert the geometric point, use Putpoint3d(). To output in the TEXfile, use Pointdata() or Drawpoint().

Examples

```
Drawpoint3d([1,1,1]);
Drawpoint3d([[1,1,1],[0,1,0]]);
```

Remark Comparative chart of drawing of points

⇒Command List

Pointdata3d

Usage Pointdata3d(name, point list, options)

Description Generic function to generate data of the point list.

Details Options are ”Size=”,”Color=”.

Examples

```
Pointdata3d("1",[[0,1,0],[1,1,2]],["Size=2","Color=red"]);
```

⇒Command List

Putpoint3d

Usage Putpoint3d(list of 3D-points, option)

Description Generic function to draw the geometric point in the space.

Details Option is ”free” or ”fix”(default).

Examples

```
Putpoint3d(["A",[2,1,3]]);
Putpoint3d(["A",[2,1,3]],"free");
Putpoint3d(["A",[1,1,1],"C",[1,0,1]]);
```

These points don't output in the TEXfile. To output in the TEXfile use the following Pointdata() or Drawpoint()

In the 3D-drawings the coordinate of the point name A is A3d.

Remark Comparative chart of drawing of points

⇒Command List

Putaxes3d

Usage Putaxes3d([x,y,z])

Description Generic function to make the geometric points on the coordinate axis.

Details For the argument [x,y,x] we get the four geometric points X(x,0,0), Y(0,y,0), Z(0,0,z) and O(0,0,0).

Examples

```
Putaxes3d([1,2,3]);
Putaxes3d(a); //this equals to Putaxes3d([a,a,a]);
```

⇒Command List

PutonCurve3d

Usage PutonCurve3d(name, PD)

Description Generic function to make the geometric point on the 3D-curve.

Details This point moves along the curve by mouse dragging.

Examples

Make reference to Partcrv3d()

⇒Command List

Putonseg3d

Usage Putonseg3d(name, point1, point2)

Description Generic function to make the geometric point on the 3D-segment.

Details We get the middle point between the two points. This point moves along the segment by mouse dragging.

Examples

```
Putonseg3d("C",A,B); //Put C on the center of A and B.
Putonseg3d("C",[A,B]); //same as above
```

⇒Command List

Spaceline

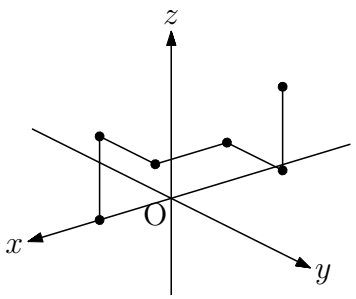
Usage Spaceline(name, list)

Description Generic function to draw the space polygonal lines.

Details Options are line type: ”dr” or ”da” or ”do”.

Examples

```
Spaceline("1",[[2,5,1],[4,2,3]]); //draw the line between two points
Spaceline("2",[A,B,C,A]); //draw the triangle ABC
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"]);
```



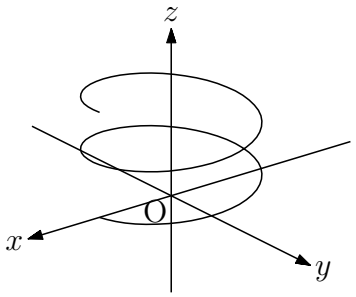
⇒Command List

Spacecurve

Usage Spacecurve(name, formula, domain, options)

Description Generic function to draw the space curve.

Examples `Spacecurve("1","[2*cos(t),2*sin(t),0.2*t]","t=[0,4*pi]",["Num=100"]);`
option=[”Num=100”]: division number of the interval ”t=[0,4*pi]”



⇒Command List

Bezier3d

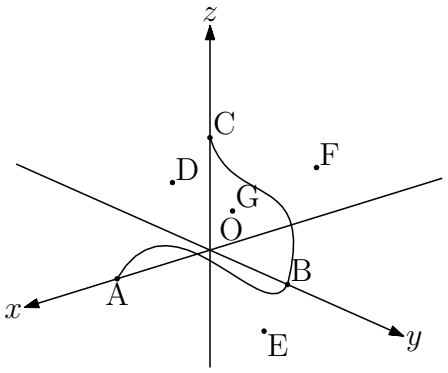
Usage Bezier3d(name,list1,list2)

Description Draw a Cubic Bézier curve.

Details list1 is list of anchor points, and list2 is list of handle points

Examples

```
Bezier3d("1",["A","B","C"],["D","E","F","G"]);
```



⇒Command List

Mkbezierptcrv3d

Usage Mkbezierptcrv3d(list)

Description Draw a cubic Bézier curve from nodes.

Details Arrange the control points automatically. After that, move the nodes and the control points and correct the cubic Bézier curve to what you want to draw. See the function `Bezier3d`.

Examples

```
Mkbezierptcrv3d(["A","B","C","D"]);
```

⇒Command List

Skeletonparadata

Usage Skeletonparadata(name, PDlist, PDlist, option)

Description Generic function to draw the lines by performing hidden line processing.

Details This function draw the second argument(the list of the lines) by performing hidden line processing which are hidden by the third argument(the list of the lines). If both arguments are omitted the function draw all lines by performing hidden line processing.

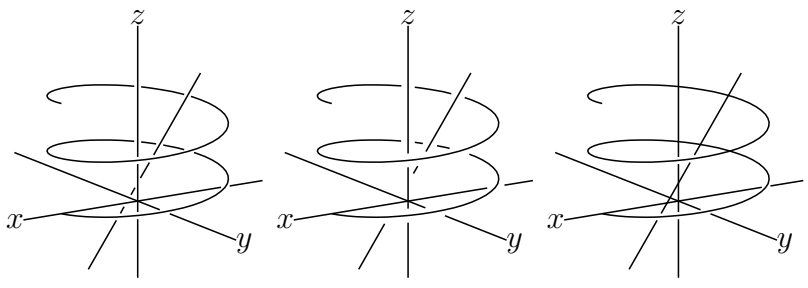
Options:

`real number` gap of line
`"No=pointlist"` not executed when any point is selected
`"File=y/m/n(default:n)"` whether to make data file or not
`"Check=pointlist"` data file updated if any point is changed

Examples

```
Xyzax3data("","x=[-5,5]","y=[-5,4]","z=[-5,3]"); //Data name is "ax3d".
Putpoint3d(["A",[0,-2,-2]]);
Putpoint3d(["B",[-1,1,3]]);
Spaceline([A,B]); //Data name is "AB3d".
Spacecurve("1","[2*cos(t),2*sin(t),0.2*t]","t=[0,4*pi]",["Num=100"]); //Data name is "sc3d1".
```

```
Skeletonparadata("1"); //(left figure)
Skeletonparadata("1",[2]); //option=[2]: gap of lines=2 (center figure)
Skeletonparadata("1",["AB3d","ax3d"],["sc3d1"]); //(right figure)
```



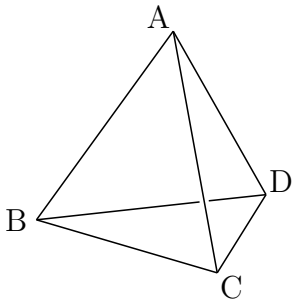
⇒Command List

5.3.2 Polyhedron

The description of polyhedron drawing will be explained by taking the case of tetrahedron as an example.

The tetrahedron is composed of four sides. Letting the vertices be A, B, C, D, the four faces are

△ABC, △ABD, △ACD, △BCD



If numbers are given to the vertex list [A, B, C, D] in order from A, the vertex order of each face is [1, 2, 3], [1, 2, 4], [1, 3 , 4], [2, 3, 4].

[A, B, C, D], [[1, 2, 3], [1, 2, 4], [1, 3, 4], [2 , 3, 4]]] is called "surface data". VertexEdgeFace () draws a polyhedron using this surface data.

There are two kinds of hidden line processing of polyhedron. The first method is to treat polyhedron as a line drawing, and to process only the hidden part, using Skeletonparadata ().

The other is to use Phparadata () as a way to draw a part hidden in the surface with a dotted line or hide it, considering it as a surface.

Concatobj

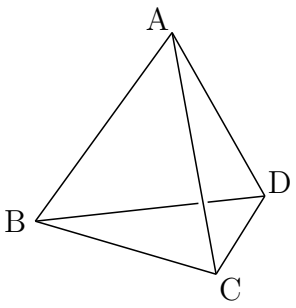
Usage Concatobj(list,options)

Description Concatenates several objects.

Examples

A tetrahedron by four vertecies A,B,C,D.
The tetrahedron consists of four planes △ABC, △ABD, △ACD, △BCD.

```
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"]);
```



If you are drawing tetrahedrons without creating geometric points, you can do as follows.

```
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]]);
```

In the case of a convex polygon such as a tetrahedron, we can use CindyScript 's convexhull 3 d () function as follows. You can save time and effort by simply providing a vertex list instead of a surface list.

```
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]);
```

⇒Command List

Vertexedgeface

Usage VertexEdgeFace(name, list, options)

Description Generic function to draw the polyhedron.

Details We use the faces data of the polyhedron.
The second argument is the list of vertexes list and the faces list.
For example, the faces data of the tetrahedron is [[A,B,C,D],[[1,2,3],[1,2,4],[1,3,4],[2,3,4]]].

The generated data is as follows.

phv3d: list of vertices

phe3d: list of edges

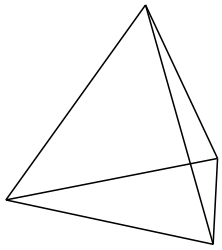
phf3d: Surface list

Each name is appended to the end.

Examples

```
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);
```
//Three data lists are made, phv3d1:vertex, phe3d1:edge and phf3d1:face.



⇒Command List

Phparadata

Usage Phparadata(name, name2, list of options)

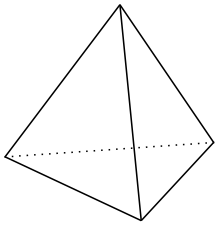
Description Generic function to draw the polyhedron by performing hidden line processing.

Details Make polyhedral plot data with VertexEdgeFace (). For this plot data, hidden surfaces (sides) are hidden-line processed and displayed. The second argument name2 is the same as the name given by VertexEdgeFace (). The hidden line type is specified by the option "Hidden = line type". Hidden lines are not displayed by default setting.

Examples

To draw a tetrahedron,

```
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"]);
```



A tetrahedron is drawn by VertexEdgeFace (), but it is hidden by Phparadata (). Since it is correctly output if it is drawn with the figure button, it is good to execute Phparadata () after confirming it by displaying it on the screen before executing Phparadata ().

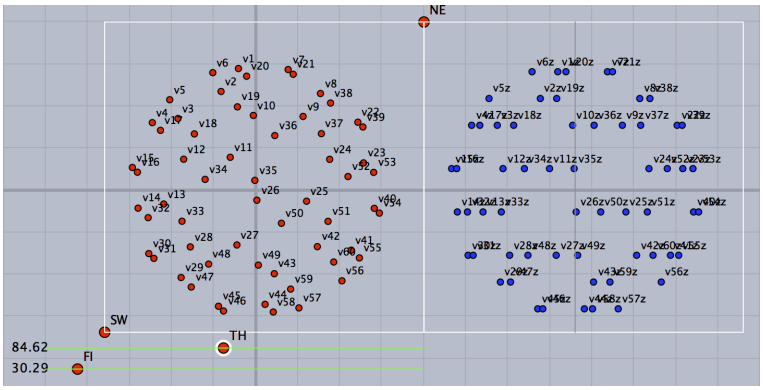
Draw a truncated icosahedron of s06 (soccer ball type) using polyhedron data `polyhedrons_obj` by Kobayashi, Suzuki, Mitani.

`Setdirectory( Dirhead+"/data/polyhedrons_obj");` //Many polyhedron data exist in this directory.
`phd=Readobj("s06.obj",["size=3"]);` //"s06" is the name of truncated icosahedron data.
`Setdirectory(Dirwork);` //Chage work space.
`VertexEdgeFace("s06",phd);`

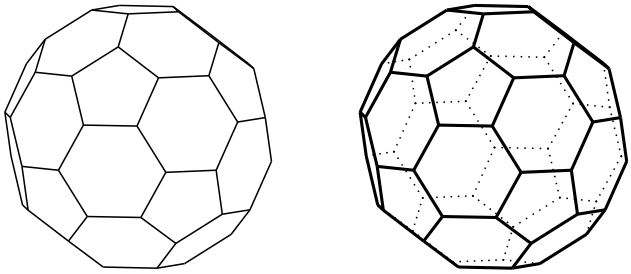
```
Phparadata("1","s06"); //default usage, left figure
```

The last two lines we can write the following.

```
VertexEdgeFace("1",phd);
Phparadata("1","1");
```




```
Phparadata("1","s06",["dr,2","Hidden=do"]); //right figure
```



⇒Command List

Nohiddenbyfaces

Usage Nohiddenbyfaces(name,PD1,PD2,option1,option2)

Description Generic function to draw hidden lines by the surfaces.

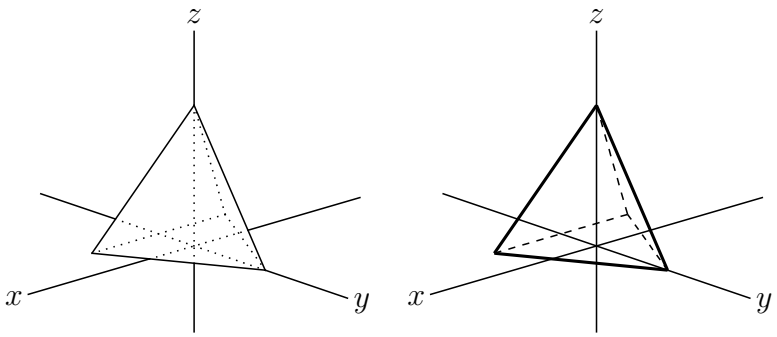
Details PD1 are hidden lines, PD2 are surfaces.
If we omit PD1 then all lines are processing objects.
By default, hidden lines are drawn with dotted lines.
Option1=line type of PD2 and option2=line type of hidden lines.
If we specify only option2 then option1 must be null list:[].

Examples

```
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");
```

(left figure)

```
Nohiddenbyfaces("1","phe3d1","phf3d1",["dr,2"],["da"]);
```
(right figure)



We draw hidden axes with broken line in the following example.

```
Nohiddenbyfaces("1","ax3d","phf3d1",[],["da"]);
```

⇒Command List

5.3.3 Surface

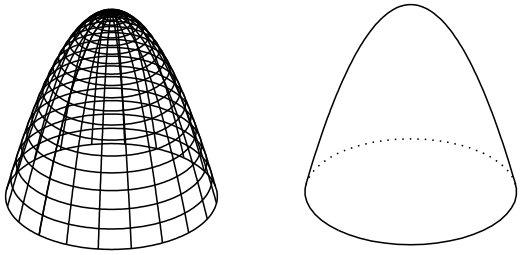
There are wire frame models and surface models for drawing curved surfaces. The wire frame model represents a curved surface with stitches, and the surface model draws its contour as a stitch-free surface.

In KeTCindy, each drawing is done using the following function.

Wire frame model without hidden wire	Sf3data(name,form,options)
Surface model	Sfbdparadata(name,form,options)
Hidden-line wireframe model	Wireparadata(name,PD,form,n1,n2,options)

However, in order to do hidden line processing, surface data is necessary, so after drawing with Sfbdparadata (), draw hidden lines with ireparadata ().

Also, in the drawing of the surface model, it takes time to process the hidden line, so it is assumed to use the C language. Therefore, ExeccmdC () which draws using C language is used together.



Wireframe Model　　Surface Model

The form of the argument is an equation and a list of character strings for the domain of the variable. There are three patterns of equations as follows.

(1) $z = f(x, y)$
　　Example formula : $z = x^2 - y^2$
　　range : $x = (-2, 2), y = (-2, 2)$
(2) $z = f(x, y), x = g(r, t), y = h(r, t)$

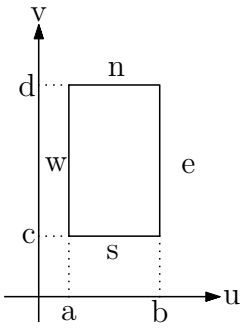
Example formula : $z = 4 - (x^2 + y^2), x = r\cos t, y = r\sin t$
range : $r = (0, 2), t = (0, 2\pi)$

(3) $x = f(u, v), y = g(u, v), z = h(u, v)$,
Example formula : $x = 2\sin u\cos v, y = 2\sin u\sin v, z = 2\cos u$
range : $u = (0, \pi), v = (0, 2\pi)$

Here, (2) and (3) are parametric types, each consisting of expressions of x, y, z and two domain of parametric variables. Since it is indistinguishable as it is, when giving it as an argument, "p" is added to the type of (3) as the identification character at the beginning.

Regarding the domain of definition, there are cases where it is taken in the open section and in the closed section. The distinction is indicated by "ewsn" as boundary designation (both are closed segments). I think the meaning of "ewsn" as follows.

For variable u, v , $a \leq u \leq b, c \leq v \leq d$



This boundary designation is added at the end, but it can be omitted, and if omitted, it is the initial value "ewsn" (closed interval).

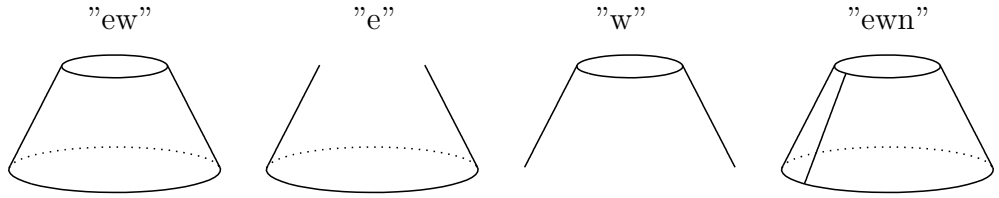
To make both open segments, add "". However, do not perform hidden line processing Sf3data () draws a line also on the boundary, so you can omit this specification.

```
fd=["p","x=r*cos(t)","y=r*sin(t)","z=2*(2-r)","r=[1,2]","t=[0,2*pi]","ew"]
```

If this is "e", $1 < r \leq 2$ is obtained, and the top face is not displayed.

Also, if this is set to "w", $1 \leq r < 2$ and the bottom is not displayed.

Furthermore, if you specify "ewn" or "ews" or abbreviate the initial value "ewsn", it will contain either the left or right value of $t = (0, 2\ pi)$, A border appears.



Sf3data

Usage Sf3data(name, list, list of options)

Description Generic function to draw the wire frame model of the surface.

Details Second argument is the list of equations and ranges.

Options are the followings.

"Num=[a,b]": x- and y-division number, default(or initial values) are a=b=25.

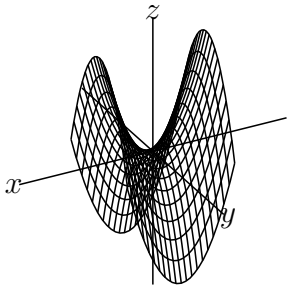
"Wire=[a,b]": x- and y-wire number, default(or initial values) are a=b=20.

"ewsn": From east to south, this indicates the boundary.

Examples

```
Sf3data("1",["z=x^2-y^2","x=[-2,2]","y=[-2,2]"]);
```

//This is the first expression of the equation for the surface. Second argument is the list of equation, x-range and y-range.

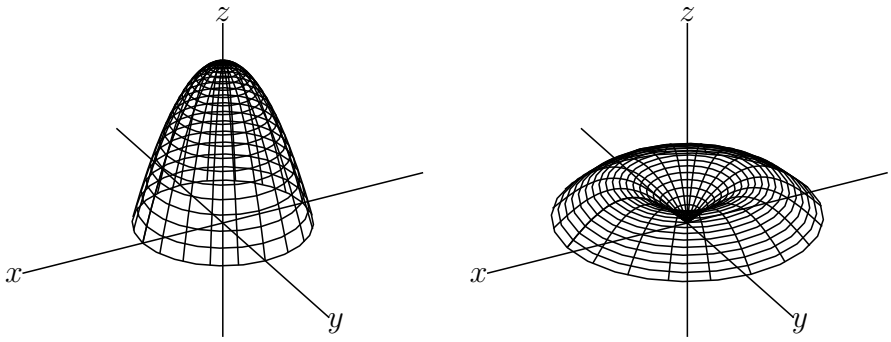



```
fd=["z=4-(x^2+y^2)","x=R*cos(T)","y=R*sin(T)","R=[0,2]","T=[0,2*pi]"];
Sf3data("1",fd); //fd is the second argument.
```

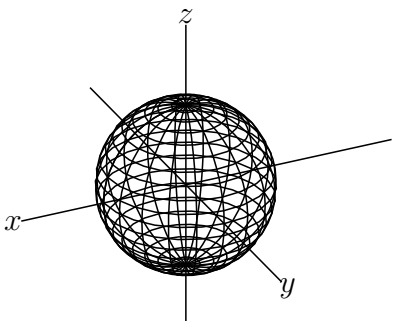
(left figure)

```
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);
```

(right figure)




```
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]"]; //"p" indicates the 3D-parameter expression.
Sf3data("1",fd);
```



⇒Command List

Sfbdparadata

Usage Sfbdparadata(name, list, list of options)

Description Generic function to make the surface by performing hidden line processing.

Details Second argument is the list of equations and ranges same as the function "Sf3data".
options1=no option or " "(space) or "r" or "m" and "Wait=integer". Default value of Wait is 20.
No option or " "(space) means
(1) If there exist no deta then it make a new data file.
(2) If there exist deta then it read the data file.
"m" means that it remake the new data file.
"r" means that it reread the existing data file.

option2="nodisp" or line type of hidden line. Default is "nodisp".

If we specify only option2 then we denote that option1 is empty list:[].

⇒Command List

ExeccmdC

Usage ExeccmdC(name,options1,options2)

Description Generic function to draw 3D-surface. The return value is the list of processed plot data.

Details options1=no option or " "(space) or "r" or "m" and "Wait=integer" , line type. Default value of Wait is 20.
No option or " "(space) means
(1) If there exist no deta then it make a new data file.
(2) If there exist deta then it read the data file.
"m" means that it remake the new data file.
"r" means that it reread the existing data file.

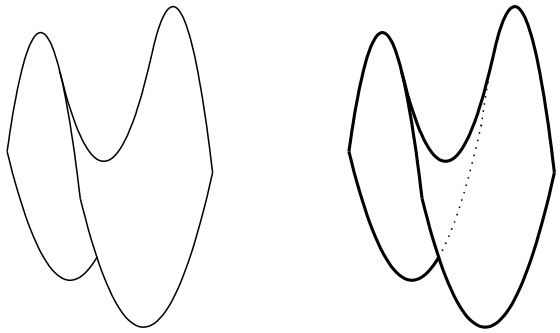
option2="nodisp" or line type of hidden line. Default is "do".

If we specify only option2 then we denote that option1 is empty list:[].

Examples

Hidden lines are not shown or shown.

```
fd=["z=x^2-y^2","x=[-2,2]","y=[-2,2]"];
if(Isangle(),
  Sf3data("1",fd);
  ,
  Startsurf();
  Sfbdparadata("1",fd,[],["nodisp"); // Change "nodisp" to "do"
  ExeccmdC("1"]);
);
```



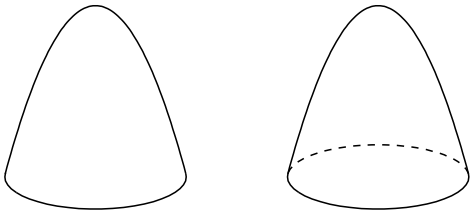
Make the whole thick with a solid line and
display the hidden line with a dotted line (Default).

```
ExeccmdC("1",["dr,2"]);
```

Paraboloid

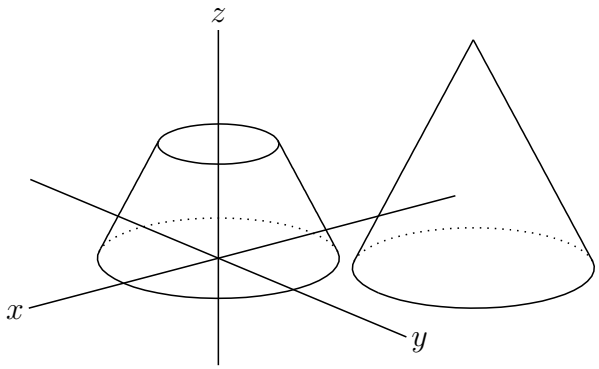
```
fd=["z=4-(x^2+y^2)","x=R*cos(T)","y=R*sin(T)","R=[0,2]","T=[0,2*pi]","e"];
```

Delete hidden line (left figure) `ExeccmdC("1",[],["nodisp"]);`
Hidden lines are indicated by broken lines (right figure) `ExeccmdC("1",[],["da"]);`

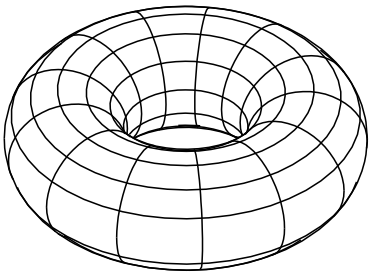


When displaying two curved surfaces, name of Sfbdparadata () is set to ”1” and ”2”, but it can be displayed together as |ExeccmdC (”1”) |.

```
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");
);
```




```
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"]);
   );
```



⇒Command List

Wireparadata

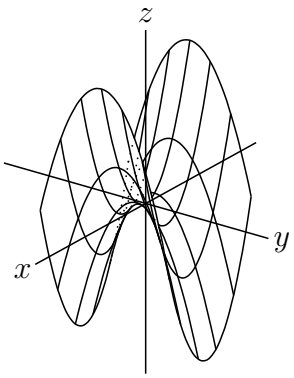
Usage Wireparadata(name, PD, formula, integer, integer, options)

Description Generic function to draw the surface by wire frame data with performing hidden line processing.

Details The second argument PD is the surface data made by Sfbdparadata function. options=no option or " "(space) or "r" or "m" and "Wait=integer". Default value of Wait is 30.
No option or " "(space) means
(1) If there exist no deta then it make a new data file.
(2) If there exist deta then it read the data file.
"m" means that it remake the new data file.
"r" means that it reread the existing data file.

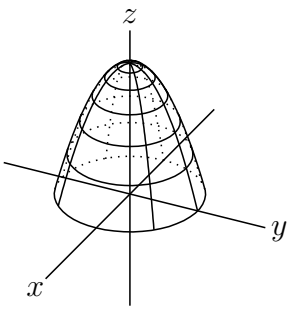
Examples

```
fd=["z=x^2-y^2","x=[-2,2]","y=[-2,2]"];
if(Isangle(),
  Sf3data("1",fd);
  ,
  Startsurf();
  Sfbdparadata("1",fd);  //We get the data named as "sfbd3d1".
  Wireparadata("1","sfbd3d1",fd,4,5,[""]);  //number of wires are 4 and 5.
  ExeccmdC("1");  //draw the wires
);
```

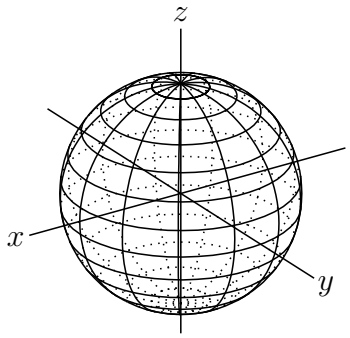


Change the following code.

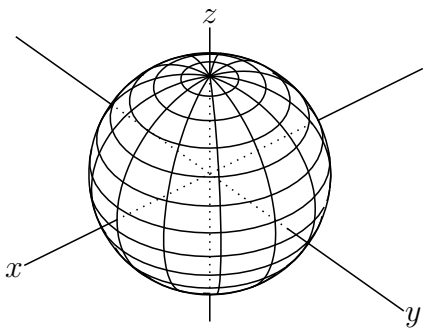
```
fd=["z=4-(x^2+y^2)","x=r*cos(t)","y=r*sin(t)","r=[0,2]","t=[0,2*pi]","e"];
Wireparadata("1","sfbd3d1",fd,5,7,[""]);
```




```
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]","s"];
if(Ptselected(),
  Sf3data("1",fd);
  ,
  Startsurf();
  Sfbdparadata("1",fd);
  Wireparadata("1","sfbd3d1",fd,12,12);
  ExeccmdC("1");
);
```




```
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]","s"];
if(Isangle(),
   Sf3data("1",fd);
  ,
   Startsurf();
   Sfbdparadata("1",fd);
   Wireparadata("1","sfbd3d1",fd,12,12,[""]);
   Crvsfparadata("1","ax3d","sfbd3d1",fd);
    ret=ExeccmdC("1");
    forall(1..length(ret),
      if(indexof(ret_#,"wireh")>0,
        Changestyle3d([ret_#],["nodisp"]);
      );
    );
);
```



⇒Command List

Crvsfparadata

Usage Crvsfparadata(name,PD1,PD2,formula)

Description Remove curves hidden by curved face.

Examples

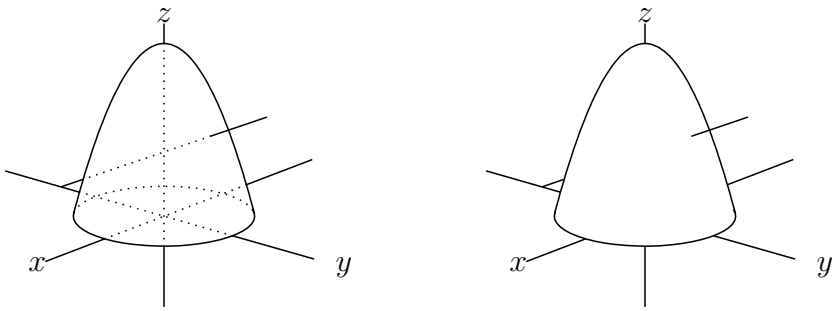
left figure

```
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");
```

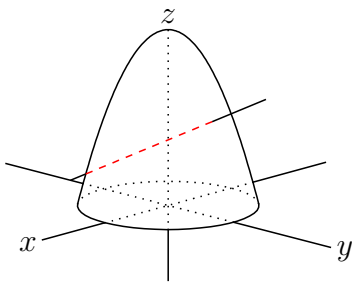
right figure

```
ExeccmdC("1",[],["nodisp"]);
```



By using the return value, you can change the hidden line style (line style, color). The same as the return value is displayed as "readoutdata from template3D1.txt:" on the console, so you can decide the operation target by looking at it. For example, in the left diagram above, the hidden line of line AB is the fourth crvsfh3d1 in the list, so you can make it a red dashed line as follows.

```
ret=ExeccmdC("1");
Changestyle3d(ret_4,["da","Color=red"]);
```



⇒Command List

5.4 Using Plot data

Datalist2d

Usage Datalist2d()

Description Generic function to get a list of 2D-plotting data on the screen.

Examples

We execute the following program then the computer will display "PD=[ax2d,AB2d]" on the console.

```
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());
```

⇒Command List

Datalist3d

Usage Datalist3d()

Details Generic function to get a list of 3D-plotting data.

Examples

We execute the following program then the computer will display ”PD=[ax3d,AB3d]” on the console.

```
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());
```

⇒Command List

Changestyle3d

Usage Changestyle3d(PD,option)

Description Change the attribute of PD.

Details Change the attribute of PD to one with option specification. PD is a plotting data or a list of plotting data.

Examples

Make a tetrahedron by four points of space.

```
Spaceline("1",[A,B]);
Spaceline("2",[A,C]);
Spaceline("3",[B,C]);
Spaceline("4",[A,D]);
Spaceline("5",[B,D]);
Spaceline("6",[C,D]);
```

then

```
Changestyle3d("sl3d1",["dr,3"]); // one edge become thick.
```

or

```
edges=apply(1..6,"sl3d"+text(#));
Changestyle3d(edges,["notex"]); // all edges become “notex”.
```

⇒Command List

Intersectcrvsf

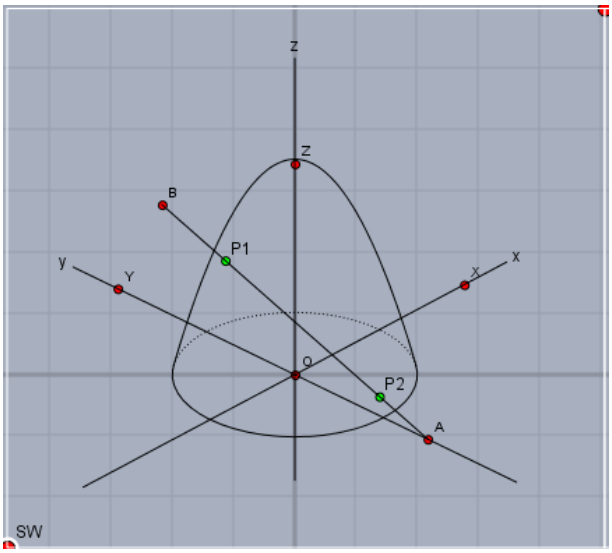
Usage Intersectcrvsf(name,PD,formula)

Description Returns a list of intersects of a curve and curved face.

Details PD is plotting data of curve. Curved face is given by formula.

Examples

```
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);// The result [[0,1.57,1.52],[0,-1.91,0.36]] will be shown in the console.
ExeccmdC("1",[""]);
println("Intersect="+intercrvsf1);
Drawpoint3d(intercrvsf1);
Letter(Parapt(intercrvsf1_1),"ne","P1");
Letter(Parapt(intercrvsf1_2),"ne","P2");
```



⇒Command List

IntersectsgpL

Usage IntersectsgpL(name,segment,plane,option)

Description Returns a intersection of a line segment and plane.

Details Specify a line segment with two endpoints. Specify the plane as three points that it contains. Options are ”put” or ”I” or ”e” .

put : Create geometric points

i : Draw a point if it is within a line segment

e : Draw a point if you meet on the plane

Following two programs return the same result.

```
IntersectsgpL("P","A-B","C-D-E");
IntersectsgpL("P",[A3d,B3d],[C3d,D3d,E3d]);
```

Return value is [pt,flag1,flag2,val1,val2]

pt : The coordinates of the intersection of the straight line and the plane. If the straight line and the plane are parallel and the intersection does not exist, the empty list [].

flag1 : True if the intersection is within the line segment, false otherwise

flag2 : True if intersection is in plane, false otherwise

val1,val2 :Parameter values for line segments, parameter values for planes

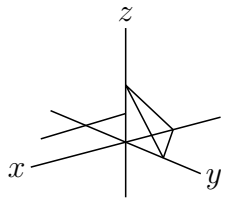
Examples

Presence or absence of intersection and return value.

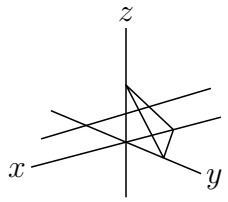
The return value of |flag 1, flag 2 | when changing |p2 | with the following script

```
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);
```

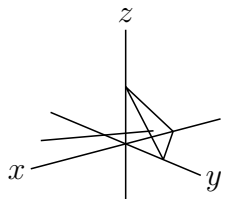
```
p2=[0,0,1/2];
flag1=false : flag2=true
```



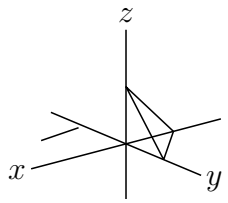

```
p2=[-1,1,1];
 flag1=true : flag2=true
```




```
p2=[1,2,1];
flag1=true : flag2=false
```



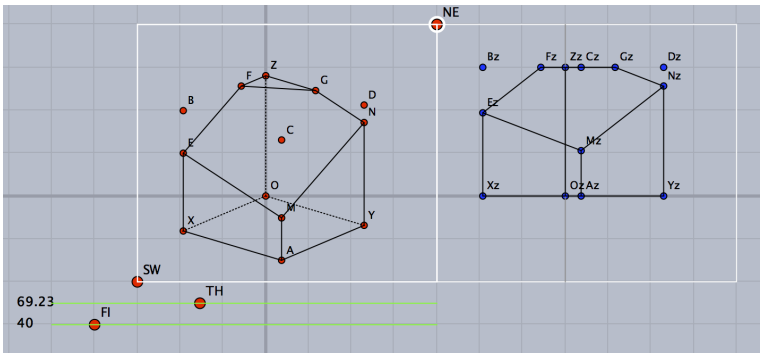

```
p2=[1,0,1/2];
flag1=false : flag2=false
```



Cutcube

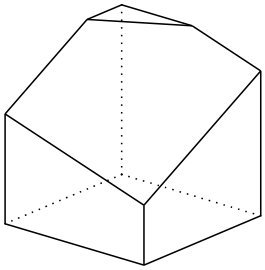
```
Hn=3;
Putaxes3d(Hn);
Putpoint3d("A",[Hn,Hn,0]);
Putpoint3d("B",[Hn,0,Hn]);
Putpoint3d("C",[Hn,Hn,Hn]);
Putpoint3d("D",[0,Hn,Hn]);
Putonseg3d("E",X,B);
```

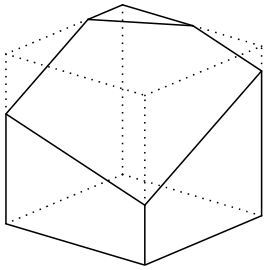
```
Putonseg3d("F",Z,B);
Putonseg3d("G",Z,D);
IntersectsgpL("M","A-C","E-F-G","put");
IntersectsgpL("N","D-Y","E-F-G","put");
phd=Concatobj([[O,X,A,Y],[X,A,M,E],[A,Y,N,M],[Y,N,G,Z,O],
    [O,Z,F,E,X],[Z,F,G],[E,M,N,G,F]]);
VertexEdgeFace("1",phd);
Nohiddenbyfaces("1","phf3d1");
```



add next script (right figure)

```
Spaceline("1",[E,B,F],["do"]);
Spaceline("2",[B,C,M],["do"]);
Spaceline("3",[C,D,N],["do"]);
Spaceline("4",[D,G],["do"]);
```





⇒Command List

Sfcutparadatacdy

Usage Sfcutparadatacdy(name,string,list,options)

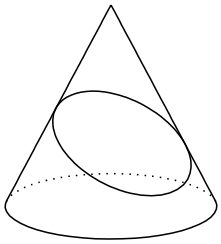
Description Obtain a line of intersection between a plane and a curved surface.

Details string is equation of plane, list is equation of a surface.

Examples Cross section of cone.

```
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");
```



⇒Command List

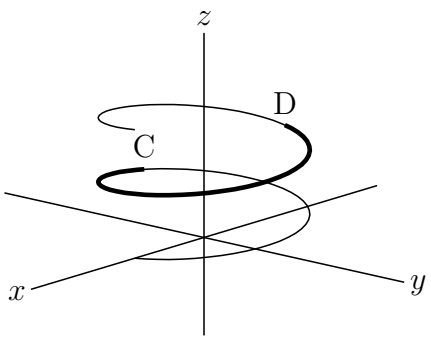
Partcrv3d

Usage Partcrv3d(name, start point, end point, PD)

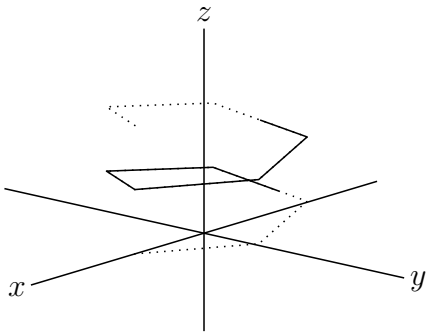
Description Generic function to draw the part curve of the curve PD.

Examples

```
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"]);
```




```
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"); // 3.3 and 8.5 are plotting data number of the points.
```



⇒Command List

Reflectdata3d

Usage Reflectdata3d(name, list of PD, list, options)

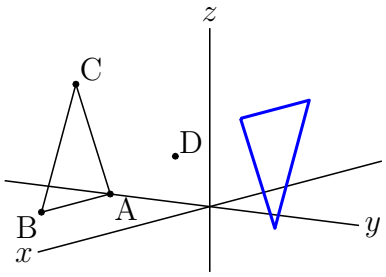
Description Generic function to draw the reflection of plotting data.

Examples

```
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]);
```

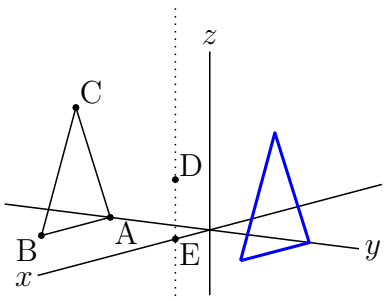
Reflection on the point D

```
Reflectdata3d("1",["sl3d1"],[D3d],["Color=blue","dr,2"]);
```



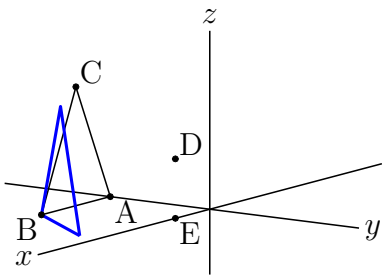
Reflection on the straight line DE

```
Reflectdata3d("1",["sl3d1"],[D3d,E3d],["Color=blue","dr,2"]);
```



Reflection on the plane BDE

```
Reflectdata3d("1",["sl3d1"],[D3d,E3d,B3d],["Color=blue","dr,2"]);
```



⇒Command List

Reflectpoint3d

Usage Reflectpoint3d(coordinate,list)

Description Return the coordinate of the reflect point.

Details Argument ”list” is the list of 3D-coordinate of the points. The following examples are the details.

Examples

```
Reflectpoint3d(A3d,[B3d]);          // reflection of the point A on the point B
Reflectpoint3d(A3d,[B3d,C3d]);      // reflection of the point A on the line BC
Reflectpoint3d(A3d,[B3d,C3d,D3d]); // reflection of the point A on the plane BCD
```

⇒Command List

Rotatedata3d

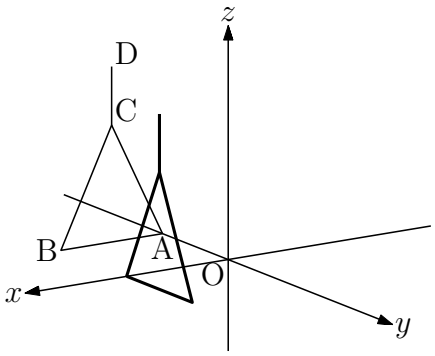
Usage Rotatedata3d(name, list of PD, vec, angle, options)

Description Generic function to rotate plotting data around the vector vec starting from the origin.

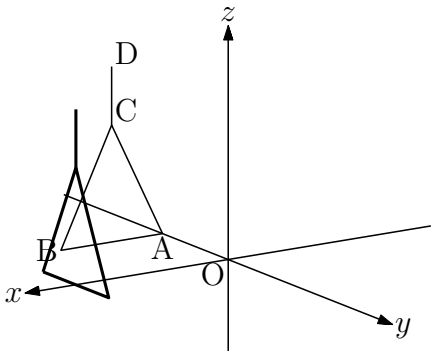
Details The options are the center point (the starting point of vec) and so on.

Examples

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



```
Rotatedata3d("1",["sl3d1","CD3d"],[0,0,1],pi/2,[[1,0,0],"dr,2"]);
```



⇒Command List

Rotatepoint3d

Usage Rotatepoint3d(coordinate,vec,angle,center)

Description Return the coordinate of the rotate point.

Details ”vec”(3D-vector) represents the axis of rotation and ”center” means the start point of 3D-vector. Default value of center is the origin (of the coordinate axes).

Examples

```
Putpoint3d("A",[0,-1,0]);
Rotatepoint3d(A3d,[0,0,1],pi/2);          // return value is [1,0,0].
Rotatepoint3d(A3d,[0,0,1],pi/2,[1,1,1]); // return value is [3,0,0].
```

⇒Command List

Scaledata3d

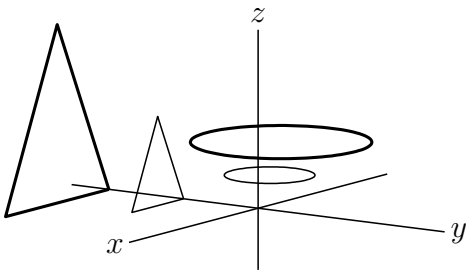
Usage Scaledata3d(name, list of PD, vec, [options])

Description Generic function to scale plotting data.

Details Vec is a three-dimensional vector to express ratio. The center and options are given in a list.

Examples

```
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"]);
```



⇒Command List

Scalepoint3d

Usage Scalepoint3d(point,vector,center)

Description Execute scale transformation for the coordinate of the point.

Details Scalepoint3d($[a_i], [v_i], [c_i]$) = $[(a_i - c_i)v_i + c_i]$

Examples

```
Putpoint3d(["A",[2,-1,2]]);
pt=Scalepoint3d(A3d,[3,2,4],[1,1,1]); //pt=[4,-3,5]
Putpoint3d(["B",pt]);
```

⇒Command List

Translatedata3d

Usage Translatedata3d(name, PD, vector)

Description Generic function to translate plotting data.

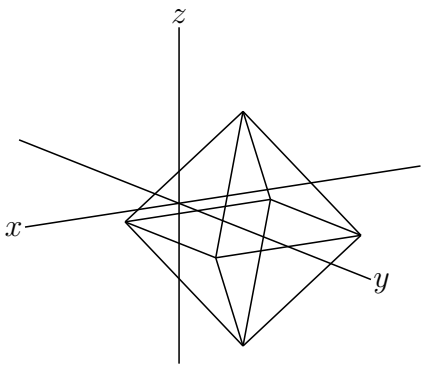
Examples

The curve sc3d1 is translated by 2 in the y axis direction. As a result, two curves parallel to the original curves are drawn.

```
Translatedata3d("1",["sc3d1"],[0,2,0]);
```

Since polygons drawn with VertexEdgeFace() can not be translated by this function, parallel movement is performed by directly manipulating the surface data. For example, to draw a regular octahedron using the polyhedron data obj of Kobayashi, Suzuki, and Mitani, do the following. This is the case of parallel movement by 2 in the y axis direction.

```
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);
```



⇒Command List

Translatepoint3d

Usage Translatepoint3d(coordinate,vector)

Description Return the translated coordinate for the point.

Details Translatepoint3d($[a_i], [v_i]$) = $[a_i + v_i]$

Examples

```
Putpoint3d(["A",[1,0,0]]);
pt=Translatepoint3d(A3d,[-1,1,1]);
Putpoint3d(["B",pt]);
```

⇒Command List

5.5 Others

Perpplane

Usage Perpplane(name, point, vector, option)

Description Generic function to return the two points on the plane which is passing through the point and orthogonal to the vector.

Details The name is the two points name such as the form "A-B".
Point is the name or the coordinate of the point through which the plane is passing.
The vector is the normal of the plane.
If option is "put" then the function draw two geometric points.

Examples

Return the points A,B on the plane which is passing through the point P and orthogonal to the vector [1,1,1]

```
. Perpplane("A-B","P",[1,1,1],"put");
```

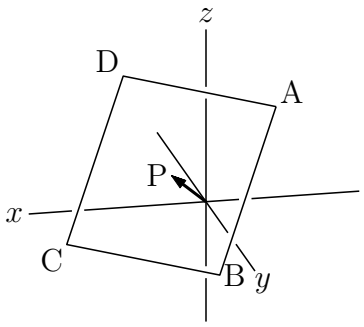
Return the points A,B on the plane which is passing through the point P and orthogonal to the line segment OP. In this situation PA and PB is orthogonal and length of PA and PB are 1.

```
. Perpplane("A-B","P",P3d-O3d);
```

Draw point A,B,C,D by draw tool of Cinderella.

```
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");
```



⇒Command List

Perppt

Usage Perppt(name, point, list of points, option)

Description Generic function to get the foot of a perpendicular for the plane from the point.

Details We specify the plane by the list of points.
Option is the following.
"draw": draw the point, don't make the geometric point(default).
"put" : make the geometric point.
"none": only make the data and don't draw.

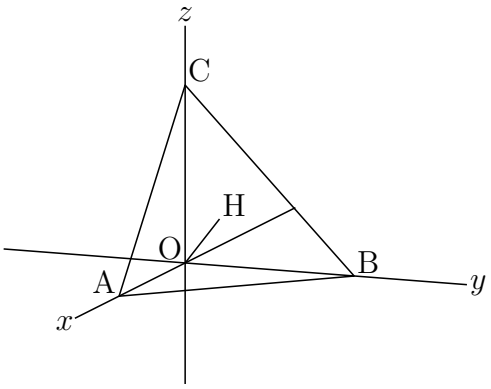
Examples

We get the coordinate of the point H in the variable H3d for the following examples.

```
Perppt("H","O","A-B-C","none");
Perppt("H","O","A-B-C");
Perppt("H","O","A-B-C","put");
```

Example

```
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"]);
```



⇒Command List

Projcoordpara

Usage Projcoordpara(3D-coordinate)

Description Generic function to get the projection coordinate on the Euclidean view coordinate system.

Examples

`println(Projcoordpara([3,1,2]));` //printed value is such as [-0.65, 1.7, 3.27] where the third element means the (signed) distance from the projection plane.

⇒Command List

Readobj

Usage Readobj(filename, option)

Description Read in the polyhedron data in the folder name `polyhedrons_obj`

Details

Data of all `Johnson solid` can be downloaded from
http://mitani.cs.tsukuba.ac.jp/polyhedron/
Store the folder into the work folder of KETCindy for example, and execute

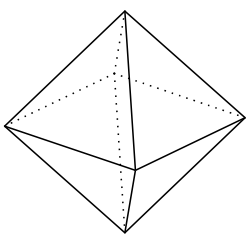
```
Setdirectory(gethome+"/ketcindy/polyhedrons_obj");
polydt=Readobj("r02.obj",["size=2"]);
Setdirectory(Dirwork);
```

Then the data of `r02.obj` are assigned to the variable `polydt`.

Option is ["size=n"] then we get the magnification of n times. If n is negative value then we have the image of vertical inversion.

Examples

```
VertexEdgeFace("1",polydt); //output data name is phf3d1
Nohiddenbyfaces("1","phf3d1");
```



The main polyhedral data is as follows.

No	name	No	name	No	name
r01	Tetrahedron	s02	Icosidodecahedron	s08	Rhombicuboctahedron
r02	Octahedron	s03	Truncatedtetrahedron	s09	Rhombicosidodecahedron
r03	Cube	s04	Truncatedoctahedron	s10	Truncatedcuboctahedron
r04	Dodecahedron	s05	Truncatedcube	s11	Truncatedicosidodecahedron
r05	Icosahedron	s06	Truncatedicosahedron	s12L/R	snubcube
s01	Cuboctahedron	s07	Truncateddodecahedron	s13L/R	Snubdodecahedoron

⇒Command List

Xyzcoord

Usage Xyzcoord(P.x, P.y, Pz.y)

Description Generic function to return the 3D-coordinate of the point P.

Details (P.x, P.y) is the coordinate of P in the mainarea and Pz.y is the y-coordinate of P in the subarea.

Examples

```
println(Xyzcoord(A.x,A.y,Az.y)); //print the 3D-coordinate of point A on the console.
```

⇒Command List

Isangle

Usage Isangle()

Description Decide the selection of the angle slider.

Details Returns “true” if select slider, and “false” if not.
In drawing including hidden line processing, reaction is bad when recalculating while moving the viewpoint. With this function, you can write code that does not recalculate while moving the viewpoint.

Examples

```
fd=[
"z=4-(x^2+y^2)",
"x=R*cos(T)","y=R*sin(T)",
"R=[0,2]","T=[0,2*pi]","e" \verb ];|
if(Isangle(),
 Sf3data("1",fd);
 ,
 Startsurf();
 Sfbdparadata("1",fd);
 Crvsfparadata("1","ax3d","sfbd3d1",fd);
 ExeccmdC("1");
);
```

⇒Command List

Dist3d

Usage Dist3d(a1,a2)

Description Generic function to get the 3D-distance of two points.

Examples

Following three programs return the same result.

```
Dist3d("A","B");
Dist3d(A,B);
Dist3d(A3d,B3d);
```

⇒Command List

Embed

Usage Embed(name,PDlist,formula,varlist)

Description Embed plotting data of 2D in plane of 3D.

Details PDlist is list of plotting data of 2D. Plane of 3D is given by formula and varlist.

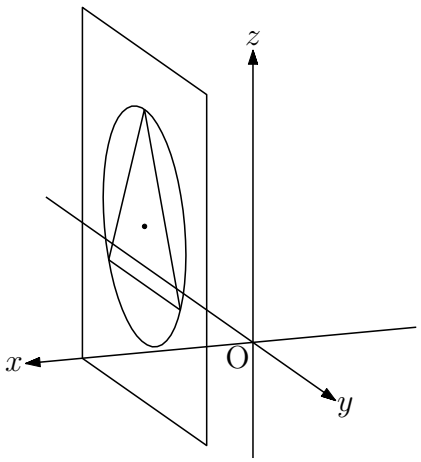
Examples

Embed an equilateral triangle and its circumscribed circle in a plane in 3D space.

(1) vo, vx, vy are defined with function Defvar that uses R.

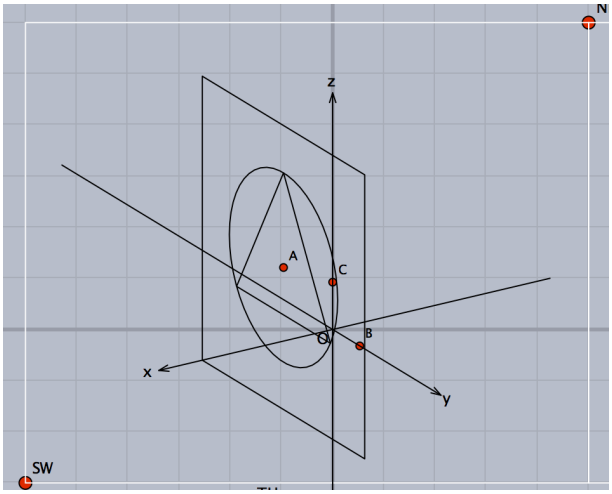
```
Xyzax3data("","x=[-5,4]","y=[-10,4]","z=[-5,5]",["a","O"]);
Spaceline("1",[[3,0,0],[3,6,0],[3,6,6],[3,0,6],[3,0,0]]);
Defvar("vo=[3,3,3]"); // Defined in R
Defvar("vx=[0,1,0]"); // Defined in R
Defvar("vy=[0,0,1]"); // Defined in R
Putpoint3d(["A",[3,3,3]]);
Circledata("1",[[0,0],[2,0]],["nodisp"]);
Listplot("1",[[0,2],[-sqrt(3),-1],[sqrt(3),-1],[0,2]],["nodisp"]);
Embed("1",["cr1","sg1"],"vo+x*vx+y*vy","[x,y]");
Ptsize(3);
Drawpoint(A);
```

Following view is as TH=75,FI=70.



(2) A, B, and C are defined instead of vo, vx, vy defined by Defvar. But, in this case, points B and C are not drawn in the plane. So, the figure may be difficult to understand.

```
Putpoint3d(["A",[3,3,3],"B",[0,1,0],"C",[0,0,1]]);
Embed("1",["cr1","sg1"],"A3d+x*B3d+y*C3d","[x,y]");
```

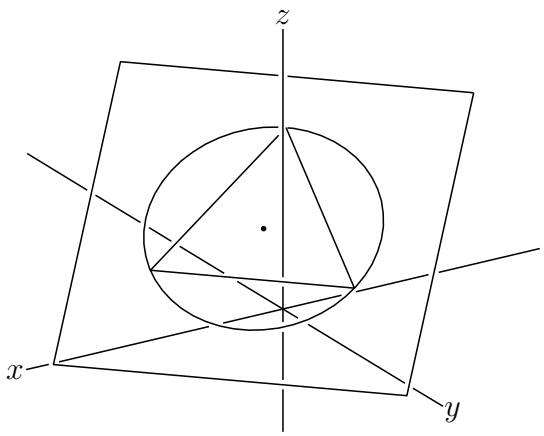


To draw the B and C on the embedded figure, code changes as follows.

```
Putpoint3d(["A",[3,3,3],"B",[3,4,3],"C",[3,3,4]]);
Embed("1",["cr1","sg1"],"A3d+x*B3d+y*C3d","[x,y]");
```

(3) The function Perpplane is used in next.

```
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]");
Ptsize(3);
Drawpoint(P);
Skeletonparadata("1");
```



⇒Command List

Parapt

Usage Parapt(3D-coordinate)

Description Generic function to return the 2D-coordinate on the plane of projection for the 3D-point.

Examples

```
println(Parapt([2,1,5]));
```

⇒Command List

Invparapt

Usage Invparapt(coordinate,PD)

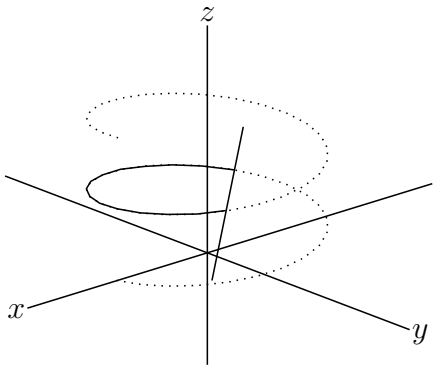
Description Returns the point on the curve that is corresponding to the coordinates on the Euclidean view.

Details Returns the 3D-coordinates of the point on the curve(`PD`) from the `coordinate` on the Euclidean view.

Examples

Find on the screen (not in the space) intersection points (`tmp_1`, `tmp_2`, ...) of the spiral curve and the space line. Draw a part of the spiral whose end points (`p1` and `p2`) are selected from the intersection points.

```
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");
```



⇒Command List

Expr3D

Usage Expr([position, direction, string],options)

Description Display the string.

Details The position is the space coordinate. Other than that it is the same as Expr().

⇒Command List

Letter3D

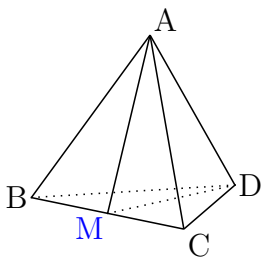
Usage Letter([position, direction, string],options)

Description Display the string.

Details The position is the space coordinate. Other than that it is the same as Letter().

Examples

```
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"]);
```



⇒Command List

6 KeTCindyJS

6.1 How to create HTML

KETCindy can create the HTML file from a KETCindy file using CindyJS.

(1) Use a KETCindy file with buttons of KeTJS in CindyScreen, for example, templateketcindyjs.cdy.

(2) Select from the top menu, `File > Export to CindyJS`.

(3) Press button "KeTJS" for on-line mode, or "KeTJSoff" for off-line mode.

(4) Then the HTML file will be created in the same folder of the cdy file.

6.2 Control of code writing

- `no ketjs (on/off)` for not writing to HTML
- `only ketjs (on/off)` for only writing to HTML
- `on, off` are used for multi lines of scripts.

Example

```
str="x^2"; //no ketjs
//str=Textedit(0); //only ketjs
```

6.3 Commands of KeTCindyJS

Ptpos

Usage Ptpos(geometric point)

Description Function to return the latest position

Examples `Ptpos(A)`

⇒Command List

Setketcindyjs

Usage Setketcindyjs(list of options)

Description Function to set options of KeTCindyJS.

Options

"Scale="	Ratio of scaling (default : 1)
"Nolabel="	list of points without label. `all` can be used
"Color="	Name or code of background color (default : lightgray)
"Grid="	Grid size (mm)
"Figure=y"	To set window size to that of KETCindy
"Axes=false"	Not to display axes of Cinderella

Example Setketcindyjs(["Nolabel=all","Grid=2","Color=white"]);

⇒Command List

Ketcindyjsdata

Usage Ketcindyjsdata(list of name, value of variables)

Description Function to write the variables into ”csinit”.

Examples

```
Mxfun("1","integrate",["x*sin(x)","x"]);// no ketjs
Ketcindyjsdata(["mx1",mx1]);
Plotdata("1",mx1,"x");
```

⇒Command List

Ketcindyjsbody

Usage Ketcindyjsbody(prependlist, appendlist)

Description Function to add scripts to the first/last of body tag.

Examples

```
Ketcindyjsbody(["<p,f10>_;_;Title"],[]);
   =>     <p><font size="10">  Title</font></p>
```

⇒Command List

Animationparam

Usage Animationparam(init, speed(/sec), range)

Description to get parameter value of buttons for animation.

Examples `ss=Animationparam(0,1,[-60,60]);`

Buttons

- ”Play” Parameter value set to the initial value, and starts changing
- ”Stop” Parameter value set to the initial value, and ends changing

⇒Command List

Textedit

Usage Texedit(number of identifier)

Description Function to get a string from an input box in HTML.

Remark To create the input box,

(1) Use button ”Define function”.

(2) Input a string into ”text” and press ”Evaluation”.
Rem)To create a blank box, input only ”=”, and add ”Equal=” to Setketcindyjs.

(3) With the inspector, confirm the number of identifier and change styles.

Examples Let the number of identifier be 50.

```
str="x^2";
//str=Textedit(50); //only ketjs
Plotdata("1",str,"x");
```

⇒Command List

Movetojs

Usage Movetojs(identifier or name,position,font size)

Description Function to set the position and font size of text box in HTML.

Example `Movetojs(50,[0,-5],15);// no ketjs`

⇒Command List

Setplaybuttons

Usage Setplaybuttons(coord, font size [, space])

Description Function to set the position of play buttons in HTML.

Example `Setplaybuttons([0,-5],15,[1]);// no ketjs`

⇒Command List

7 Appendix

7.1 To use other text editor

For example, let the cdyfie be `template.cdy`.

(1) Create `template.txt` and put it in the same folder as the cdy file.

(2) Describe in Figures slot as follows, and execute.

```
Ketinit();
setdirectory(Dircdy);
import(Cdyname()+''.txt'');
setdirectory(Dirwork);
Windispg();
```

(3) Write scripts in `template.txt` and save it.
Ex) `Putpoint("A",[0,0],A.xy);`
`Plotdata("1'',''x^2'',''x'');`

(4) Click any position on the screen.

7.2 Color table

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]	

Rem) lightgray [0,0,0,0.15], offwhite [0,0,0,0.3], cindycolor [0.66,0,69,0.71] have been added.

7.3 Comparative chart of drawing of points

return : use return value
draw : draw Euclidean view
geo : make geometric point on Euclidean view
Tex : output Tex file

command	return	draw	geo	TeX
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	◯	-	-	-

△ : use PD

8 Command List

To index

Setting and Defining

Addax	decide axis are drawn or not.
Addpackage	add packages of TEXto the main file for previewing.
Assign	replace the string1 in the string0 with the string2.
Changework	change the working directory.
Deffun	define a function common to both Cindy and R.
Definecolor	define the name of colorcode in the TEX figure.
Defvar	define variables common to both Cindy and R.
Drwxy	draw axis in the TEX figure.
Fontsize	define the font size in the TEX figure.
Ketinit	initialize KETCindy.
Initglist	add the list in ketlib slot to that of figures slot.
Ptsize	set the size of points.
Setarrow	set the style of arrow.
Setax	set the style of axis.
Setcolor	set the color of figures and characters in the TEX figure.
Setfiles	set the name of texfile.
Setparent	set the name of texfile by using the Parent push button.
Setmarklen	set the length of tickmarks on the axsis.
Setorigin	set or transtate the coordinate of apparent origin.
Setpen	set the thickness of lines.
Setpt	set the size of points.
Setscaling	set the scale of vertical direction.
Setunitlen	set the scale of unit length. (default is 1cm)
Setwindow	set a drawing area on a Euclidean view.
Strsplit	return the list of strings separated by a string.
Usegraphics	change to pict2e.

Drawing

Drawfigures	manipulate a plural number of PDs together.
Anglemark	draw an angle mark.
Setarrow	set styles of arrows.
Arrowdata	draw an arrow line between two points.
Arrowhead	draw an arrowhead with specified direction at a designated point.
Bezier	draw a Bezier curve.
Beziersmooth	draw a smooth Bezier curve.
Beziersym	draw a smooth Bezier curve.
Bowdata	draw the shape of bow connecting two points.
Bspline	draw second degree B-spline curve.
Changestyle	change the option for drawing.
Circledata	draw a circle or polygon.
CRspline	draw single Catmull-Rom spline curve.
Deqplot	draw the solution curve of a differential equation.
Dotfilldata	fill a domain with dots.
Drawppoint	draw a point.
Drawsegmark	Add a mark to a segment.
Ellipseplot	draw ellipse.
Enclosing	make a closed curve form the list of plotting data.

Expr	write an expression in TEXstyle.
Exprrot	write a rotated expression in TEXstyle.
Fourierseries	draw the graph of a fourier series.
Framedata	draw a rectangle.
Hatchdata	draw hatch lines in the close curve.
Htickmark	tick on the horizontal ax.
Hyperbolaplot	draw a hyperbola.
Implicitplot	draw the graph of a implicit function.
Invert	rearrange plotting data in the reverse order.
Joincrvs	create a plotting data of connecting in list of plotting data.
Letter	display the string.
Letterrot	rotate a string and display it.
Lineplot	draw the straight line through the two points.
Listplot	connect points by line segments.
Mkbeziercrv	draw some bezier curves.
Mkbezierptcrv	draw a bezier curve.
Mkcircles	create plotting data of all geometric circles.
Mksegments	create plotting data of all geometric segments.
Ospline	draw a spline curve of Oshima.
Ovaldata	draw a rectangle with rounded corners.
Parabolaplot	draws a parabola.
Paramark	draw an angle mark with a parallelogram.
Paramplot	draw a curve of parametric representation.
Polarplot	draw a curve of polar equation.
Partcrv	make a piece of curve from the PD.
Periodfun	draw the graph of a periodic function.
Plotdata	draw the graph of function.
Pointdata	make a point data.
Polygonplot	draw a polygon inscribed inside the circle.
Putintersect	make a intersection point of two curves.
PutonCurve	put a point on the curve.
PutonLine	put a point on the line.
Putonseg	put a point on the segment.
Putpoint	put a point.
Reflectdata	draw a reflective curve.
Reflectpoint	return the reflect point.
Rotatedata	rotate plotting data.
Rotatepoint	rotate a point.
Rulerscale	put ruler marks.
Scaledata	scale plotting data.
Scalepoint	scale a point.
Segmark	add a mark to a segment.
Shade	fill a domain surrounded by a closed curve.
Tangentplot	draw a tangent line of a plotting data.
Translatedata	translate plotting data.
Transelatepoint	translate a point.
Vtickmark	tick on the vertical ax.

Calculus and I/O

Asin	return arcsine and arccosine.
Crossprod	return the cross product of 2 vectors.

Derivative	find the derivative of a function or a plotting data.
Dotprod	return the dot product of 2 vectors.
Extractdata	add properties to a data.
Findarea	return the area enclosed with a close curve.
Findlength	return the length of a curve.
Integrate	find the value of numerical integration.
Intersectcurves	return a list of intersects of 2 plotting data.
IntersectcurvesPp	return a list of intersects with parameters of 2 plotting data.
Inversefun	find the value of the inversefunction.
Nearestpt	return the nearest point with the parameter and the distance.
Nearestptcrv	return the nearest point on the plotting data from the point1.
Numptcrv	return the number of plotting data.
Paramoncurve	return the parameter value of the point on the curve.
Pointoncurve	point which has the parameter value
Ptstart, Ptend	returns start point and end point of PD.
Ptcrv	Returns n-th point from PD.
Readcsv	read a file in csv format.
Readlines	read a text file line by line.
ReadOutData	read external data.
Sqr	return square root.
WriteOutData	write out data in KETCindy format.
Making Table	
Changetablestyle	change line styles of rules.
Findcell	return the information of a cell.
Putcell	put a string at the cell.
Putcellexpr	put a math expression at the cell.
Putcol	put strings to a column.
Putcolexpr	put math expressions to a column.
Putrow	put strings to a row.
Putrowexpr	put math expressions to a row.
Tabledata	draw rules of a table.
Tgrid	return the coordinates of the grid name.
Tlistplot	connect two lattice points by line segments.
Data Processing	
Dispmat	display the list in the console matrix.
Tab2list	convert contents of string data to list.
Writecsv	make a CSV file consisting of the contents of data.
Others	
Assign	replace the string1 in the string0 with the string2.
BBdata	return the size of an image file.
Cindyname	return the name of a current file.
Colorcode	change colorcode from colortype1 to colortype2.
Dqq	return the string surrounded by double quotes.
Factorial	return the factorial.
Figpdf	make a pdf file with the same size of figure.
Help	display usages of the function.
Indexall	return all positions of string2 in string1.
Norm	return the norm of a vector.
Op	return the n-th element of a list or a string.
Ptselected	tests whether the point is selected.

Reparse	return the real part afger parsing.
Slider	make a slider on a Euclidean view.
Sprintf	converts a real number to a string.
Texcom	add the command in the TEXfile.
Textformat	converts a real number to a string.
Toupper	return the upper case letters of a string.
Windispg	display all graphs on Euclidean view.
Fracform	return TeX-like form of the fraction.
Totexform	return TeX form.
Tocindyform	return Cindy form.
R	
Boxplot	draw boxplots.
CalcbyR	executes R commands and returns the execution result to Cinderella.
Histplot	create histograms.
PlotdataR	draw graph of R's statistical probability function.
Rfun	execute a R command.
Maxima	
CalcbyM	execute Maxima's script.
Mxbatch	make a command to execute the Maxima file.
Mxfun	execute Maxima's function.
Mxtex	convert expression to TeX format.
Risa/Asir	
CalcbyA	execute Risa/Asir's script.
Asirfun	execute Risa/Asir's function.
MeshLab	
Mkobjcmd	obj formatted files of surfaces without thickness.
Mkobjcrvcmd	obj formatted files of spatial curves.
Mkobjnrm	calculate normal vector of surface.
Mkobjplatecmd	obj formatted files of plates.
Mkobjpolycmd	obj formatted files of polyhedra.
Mkobjsymbcmd	generate commands for obj formatted files of some characters.
Mkobjthickcmd	generate commands for obj formatted files of surfaces with thickness.
Mkviewobj	generate obj formatted files.
Animation	
Setpara	set up the animation control system.
KETCindy Slide	
Setslidebody	set up the color and density of the letters in slide body.
Setslidehyper	use `hyperref.sty`.
Setslidemain	set up the main slide.
Setslidepage	set up each page of slides.
Setslidemargin	change the margin of slides.
Settitle	make a title slide.
KETCindy3D	
Bezier3d	draw a Cubic Bézier curve.
Changestyle3d	change the attribute of PD.
Concatobj	concatenates several objects.
Crvsfparadata	remove curves hidden by curved face.
Datalist2d	get a list of 2D-plotting data on the screen.
Datalist3d	get a list of 3D-plotting data.
Dist3d	get the 3D-distance of two points.

Drawpoint3d	draw 3D-points.
Embed	embed plotting data of 2D in plane of 3D.
ExeccmdC	draw 3D-surface.
Expr3d	display the string.
Intersectcrvsf	return a list of intersects of a curve and curved face.
IntersectsgpL	return a intersection of a line segment and plane.
Invparapt	return the point on the curve.
Ketinit3d	declare the use of KeTCindy3D
Letter3d	display the string.
Mkbezierptcrv3d	draw a cubic Bezier curve from nodes.
Nohiddenbyfaces	draw hidden lines by the surfaces.
Parapt	return the 2D-coodinate on the plane.
Partcrv3d	draw the part curve of the curve PD.
Perpplane	create a basic vector on a vertical plane
Perppt	get the foot of a perpendicular for the plane from the point.
Phparadata	draw the polyhedron by performing hidden line processing.
Pointdata3d	generate data of point list.
Projcoordpara	get the projection coordinate.
Putaxes3d	make the geometric points on the coordinate axis.
PutonCurve3d	make the geometric point on the 3D-curve.
Putonseg3d	make the geometric point on the 3D-segment.
Putpoint3d	draw the geometric point in the space.
Readobj	read in the polyhedron data in the folder name `polyhedrons_obj`
Reflectdata3d	draw the reflection of plotting data.
Reflectpoint3d	return the coordinate of the reflect point.
Rotatedata3d	rotate plotting data around the vector
Rotatepoint3d	return the coordinate of the rotate point.
Scaledata3d	scale plotting data
Scalepoint3d	execute scale transformation for the coordinate of the point.
Sf3data	draw the wire frame model of the surface.
Sfbdparadata	draw the surface by performing hidden line processing.
Sfcutparadatacdy	Display intersection line of surface and surface.
Skeletonparadata	draw the lines by performing hidden line processing.
Spacecurve	draw the space curve.
Spaceline	draw the space polygonal lines.
Start3d	creates subarea, and recognize 3D points.
Startsurf	defines values related to surface rendering.
Translatedata3d	translate plotting data
Translatepoint3d	return the translated coordinate for the point.
Vertexedgeface	draw the polyhedron.
Wireparadata	draw the surface by wire frame data with performing hidden line processing.
Xyzax3data	draw the coordinate axis.
Xyzcoord	return the 3D-coordinate of the point P.
KeTCindyJS	
Ptpos	Return the latest position of geometric point
Setketcindyjs	Set options of KeTCindyJS
Ketcindyjsdata	Write into csinit
Ketcindyjsbody	Add scripts to the first/last of body tag
Animationparam	Get parmeter value of animation
Textedit	Get string from input box of KeTCindyJS

Movetojs	Set position and fontsize of text box in HTML
Setplaybuttons	Set position of play buttons in HTML