

kpic それなりの使用法
Ver.2.26

浅岡信義
nob.asaoka@jcom.home.ne.jp

2019/12/31(Manual 2019/12/31)

TeXで数学のプリントを作成するとき、簡単に図を書くために`kplic.sty`を作成しました。サイエンス社発行の「`LaTeX` 自由自在」及び`eclarith.sty`を参考（というよりほとんどそのまま）に取り敢えず動くものを1994年10月に作成し、校内¹の数学の教員で使用していたのですが、同僚からの要望と前からの直したいと思っていたので、今回わかりやすくするためにほとんど書き直し、おまけにマニュアル²まで作ってしまいました。

このマクロは以下のスタイルファイルを読み込みますので、必ずパスの通ったところに用意して置いて下さい。用意しておけば、`kplic.sty`で読み込みます。

```
epic.sty
eepic.sty
eclarith.sty
```

さらに、未公開の「Ver.2.13」からは境界を白い線で消すという謀略に出たため、マクロによっては`color.sty`も必要になりました。

●「Ver.2.02」までで出来たことは、

- `picture` 環境と同時に「枠」「方眼紙」「座標」を設定することができる。
- `eclarith.sty` の`\node` の定義を拡張し、「座標の表示文字の位置と距離を指定」「極座標設定」ができる。
- 上を使用して新たな点を定義したり、作図ができる。
- 2点間の線の長さ、名称を表示できる。2直線間に直角の記号、度数法で角度の記号が表示できる。

●今回公開した「Ver.2.22」では領域を簡単に表示するためのマクロを作成し、実際の領域の表示方法について解説しました。主な追加は次のとおりです。(2012年1月21日)

- 長方形と円の内部に任意の傾きの斜線を引いて領域を表示できるようにした。
- 高等学校の数学で使用する領域を簡単に描けるようにした。
- `Pic` 環境の四隅を定義できるようにした。
- 直線を `Pic` 環境を通る時の左側、右側、そして両方の境界の点の座標名を定義できるようにした。
- 環境全体に座標軸を自動で作成する命令と斜線を引く命令を作成した。
- 直前に定義した点から座標軸へ点線を引く命令を作成した。
- `node` 座標を定義するとき、背後の図形を消して文字、数字等を表示できるようにした。

などです。

●補足「Ver.2.23」では `\KLineArc`, `\KVecArc`, `\KLineArc` の片側が点線になるバクをとり、第5章の図を変更しました(2012年1月28日)

●補足「Ver.2.25」では 円の接線の交点を求める`\IntersectionOfTangent` を作成し、座標軸表示関係で幅または高さが0の場合の軸の非表示と数直線、0 (0°) の表示を作成しました。(2012年8月14日)

●補足 2014年9月15日に30ページの文書の一部追加しました。

●補足 2017年7月23日に41ページの文書の一部追加しました。

●補足「Ver.2.26」では第5章の「点を結ぶ」で `\KPath` と `\KVec` の順番が違うとエラーするバクをとり、`\KVec` で塗られた矢印を表示、第1章では座標軸の矢印を表示しないスイッチを作成しました。(2019年12月31日)

¹2017年8月現在、神奈川県立上矢部高校のこと。タイトルを含むほとんどのマクロがKではじまるのは、KAMIYABEのKだから。こうすると他のスタイルのマクロを名前がぶつからないから。(私は2017年4月に平塚湘風高校(平塚市)へ転勤となり現在に至っています)

²今読んでいるこの文書です。これが欲しいと同僚に泣きつかれました。でもこんなのを作っているよりマクロを書いている方が楽しい。

目次

第 1 章	picture 環境の拡張	1
1.1	Pic 環境	1
1.1.1	基本環境 ---- Pic	1
1.1.2	Pic 環境で原点を中心にする環境 ---- PicC	2
1.1.3	枠付きの環境 ---- PicFrame[C]	2
1.1.4	実線格子を表示する環境 ---- Pic[Frame]Grid[C]	3
1.1.5	点線格子を表示する環境 ---- Pic[Frame]Dot[C]	4
1.2	Axes 環境 (Pic 環境に xy 座標軸を表示する)	5
1.2.1	基本環境 ---- Axes[Frame][C]	5
1.2.2	実線格子を表示する環境 ---- Axes[Frame]Grid[C]	5
1.2.3	点線格子を表示する環境 ---- Axes[Frame]Dot[C]	6
1.2.4	目盛りと数値を入れる環境 ---- Axes[Frame]Scale[C]	7
1.3	Pic 環境, Axes 環境に点線格子の表示・非表示をする %kdrawtrue	8
第 2 章	環境内で使用する	9
2.1	格子関係マクロ	9
2.1.1	指定した範囲に実線格子を表示する ---- %KGrid	9
2.1.2	指定した範囲に点線格子を表示する ---- %KDot	10
2.2	平面座標系マクロ	11
2.2.1	座標軸を表示する ---- %KAxes	11
2.2.2	環境内に座標軸を自動で表示する ---- %KAxesAuto	11
2.2.3	座標軸に目盛りをふる ---- %KScale	12
2.2.4	座標軸に数字をふる ---- %KScaleNumber	12
2.3	応用例	15
2.3.1	三角関数を書くための方眼紙	15
第 3 章	表示文字と表示位置	17
3.1	表示文字と表示位置	17
3.1.1	その座標を中心として文字を表示	17
3.1.2	文字を置く位置	17
3.1.3	座標からの距離	17
3.1.4	置く位置の微調整	18
第 4 章	点を定義	19
4.1	点の定義	19
4.1.1	共通項目	19
4.1.2	平面座標 (xy 座標) で定義 ---- %Knode	20
4.1.3	極座標で定義 ---- %Pnode	20
4.1.4	Pic 環境の四隅を一括定義 ---- %KnodeCorner	22
4.1.5	Pic 環境の四隅を個別に定義 ---- %KnodeLB, %KnodeRB, %KnodeRT, %KnodeLT	22
4.1.6	直線と Pic 環境の交点を定義 ---- %KLineLeft(or Right or Both)Edge	22
4.2	定義した点から新たな点を定義	23
4.2.1	内分点 ---- %Inode	23
4.2.2	外分点 ---- %Enode	23
4.2.3	2 点から指定した距離の交点 ---- %TwoCirclesRight(Left)	24
4.2.4	垂線の交点の座標 ---- %Perpendicularfoot	24
4.2.5	2 つの線分の交点の座標 ---- %Intersection	25
4.2.6	三角形の重心 ---- %Barycenter	25
4.2.7	三角形の外心 ---- %Circumcenter	25
4.2.8	三角形の内心 ---- %Incenter	26
4.2.9	ある点を基準として点の拡大縮小 (点対称を含む) ---- %Tnode	26
4.2.10	円の 2 接線の交点 ---- %IntersectionOfTangent	27
4.3	定義した点を利用する	27
4.3.1	定義された点を元の点とする ---- %Kput	27
4.3.2	2 点を半径とする円を描く ---- %Kcircle	27
4.3.3	定義された点から軸へ破線を引く ---- %PointDashX, %PointDashY, %PointDashXY	28

4.3.4	定義された 2 点を通る直線と軸の交点の数値を表示する	---- <code>%AxesCrossX(or Y)</code>	28
第 5 章	点を結ぶ		29
5.1	<code>eclairth.sty</code> の拡張		29
5.1.1	線の種類	---- <code>%KPen</code>	29
5.1.2	置いた点を結ぶ	---- <code>%KPath</code>	29
5.2	2 点を結ぶ		30
5.2.1	2 点を結び、名前等を入れる	---- <code>%KLine</code>	30
5.2.2	ベクトルと名前を入れる	---- <code>%KVec</code>	31
5.2.3	線に名前を入れる	---- <code>%KLineName</code>	31
5.2.4	範囲を実線で示す	---- <code>%~Arc</code>	31
5.2.5	範囲を破線で示す	---- <code>%~DashArc</code>	32
第 6 章	円弧・一般角・直角		33
6.1	共通項目		33
6.1.1	文字表示の基準点		33
6.1.2	文字		34
6.1.3	円弧の直径		34
6.2	円弧・角度の表示		34
6.2.1	<code>%arc</code> の拡張	---- <code>%KArc</code>	34
6.2.2	3 つの座標から角を表示	---- <code>%KAngle</code>	34
6.2.3	<code>%Pnode</code> を利用して角を表示	---- <code>%PAngle</code>	35
6.3	矢印を付ける		35
6.3.1	矢印の大きさ	---- <code>%KArrowLen</code>	35
6.3.2	正の方向に付ける	---- <code>%Arrow~</code>	36
6.3.3	負の方向に付ける	---- <code>%RevArrow~</code>	36
6.4	破線で円弧・一般角を描く		37
6.4.1	破線のパターン		37
6.4.2	破線で描く	---- <code>%KDashArc</code> , <code>%KDashAngle</code> , <code>%PDashAngle</code>	37
6.4.3	矢印を正の方向へ付ける	---- <code>%Arrow~</code>	38
6.4.4	矢印を負の方向に付ける	---- <code>%RevArrow~</code>	38
6.5	応用例		39
6.5.1	一般角で $2\pi(360^\circ)$ 以上を描く		39
6.6	直角の記号表示	---- <code>%KNinty</code>	39
第 7 章	領域を斜線表示する		41
7.1	はじめに		41
7.2	領域を 1 つの命令で表示する		41
7.2.1	直線を境界とする領域の表示	---- <code>%[Not]LineUpper(or Lower)Domain</code>	42
7.2.2	円を境界とする領域の表示	---- <code>%[Not]CircleInside(or Outside)Domain</code>	43
7.2.3	連立不等式の表す領域の表示 1	---- <code>%[Not]LineUpper(or Lower)Upper(or Lower)Domain</code>	44
7.2.4	連立不等式の表す領域の表示 2	---- <code>%[Not]CircleInside(or Outside)LineUpper(or Lower)Domain</code>	45
7.3	領域の斜線を表示		47
7.3.1	長方形に斜線を図示	---- <code>%Khatch</code>	47
7.3.2	環境全体に斜線を図示	---- <code>%KhatchAll</code>	47
7.3.3	円の内部の領域表示を図示	---- <code>%Khatchcircle</code>	47
7.4	領域を図示するために必要な命令		48
7.4.1	指定した閉曲線内を白で塗りつぶす	---- <code>%whiten(eepic.sty マクロ)</code>	48
7.4.2	線の幅を指定する	---- <code>%allinethickness(eclairth.sty マクロ)</code>	48
7.4.3	色付きで線を引く	---- <code>%color(color.sty マクロ)</code>	49
7.4.4	座標名で指定した多角形の内部と線を消す	---- <code>%AreaClear</code>	49
7.5	領域を図示する		49
7.5.1	領域を表示する流れ		50
7.5.2	応用例		50
7.5.3	現状できないこと		54
参考文献			55

第1章 picture 環境の拡張

原点を中央にする `picture` 環境を定義、さらに枠も付ける。そうすると以下のようになり、同じ座標を2度ずつ書かなくてはなりません。そして数値を変更する場合も、2度手間となります。

```

\begin{picture}(8,4)(-4,-2)
  \put(-4,-2){\framebox(8,4){}}
\end{picture}

```

同じことが方眼紙のようなますめを書く時や、座標軸を書く時にも考えられます。そこで、`kpic.sty` では、これらに対応した環境を設定しました。とはいっても内部で置き換えているだけです。

ここでは `\Knode` 命令が出てきますが、原点の場所を表示しているだけです。詳しいことは 4.1.2 を見て下さい。

この章の定義で「(大きさ)」の所は、「(横の大きさ, 縦の大きさ)」を入れ、「(左下座標)」は「(左下の x 座標, 左下の y 座標)」を入れて下さい。

以下にこの章に出てくる環境名を表にしておきました。

基本形	実線で格子を描く	点線で格子を描く
<code>Pic</code>	<code>PicGrid</code>	<code>PicDot</code>
<code>PicC</code>	<code>PicGridC</code>	<code>PicDotC</code>
<code>PicFrame</code>	<code>PicFrameGrid</code>	<code>PicFrameDot</code>
<code>PicFrameC</code>	<code>PicFrameGridC</code>	<code>PicFrameDotC</code>

基本形	実線で格子を描く	点線で格子を描く	目盛りをふる
<code>Axes</code>	<code>AxesGrid</code>	<code>AxesDot</code>	<code>AxesScale</code>
<code>AxesC</code>	<code>AxesGridC</code>	<code>AxesDotC</code>	<code>AxesScaleC</code>
<code>AxesFrame</code>	<code>AxesFrameGrid</code>	<code>AxesFrameDot</code>	<code>AxesFrameScale</code>
<code>AxesFrameC</code>	<code>AxesFrameGridC</code>	<code>AxesFrameDotC</code>	<code>AxesFrameScaleC</code>

上の表の「基本形」にあたる列の環境は、初めに `\kdrawtrue` と入れておくと、1目盛り単位の「点線で格子を描く」環境になります。繰り返して作図をしながら最後に格子を取るとき便利かと思い作成しました (1.3 を参照して下さい)。

1.1 Pic 環境

1.1.1 基本環境 ---- `Pic`

定義

```

\begin{Pic}(大きさ)[(左下座標)]
  -----
\end{Pic}

```

`Pic` 環境は、`picture` 環境と同じ働きをします。しかし内部で上下左右の数字を記憶していますので。特にこの `kpic.sty` の命令を使う時や作成中の `KGraph.sty` (2012 年 1 月現在未完成) を使用するときは、`picture` 環境ではなく、この `Pic` 環境を使用して下さい。

1.1.2 Pic 環境で原点を中心にする環境 ---- PicC

定義

```

\begin{PicC}(大きさ)
  - - - - -
\end{PicC}

```

PicC 環境は、picture 環境の横と縦の大きさだけで原点を中央にします。次の例を参照して下さい。

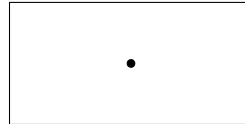
入力

```

\unitlength=4mm%
\begin{PicC}(8,4)
  \put(-4,-2){\framebox(8,4){}}
  \Knode*(0,0){0}
\end{PicC}

```

出力



どこかわかるように枠表示に、`\framebox` も入れてあります。このように枠も同時に表示するための命令が、次の `PicFrame[C]` 環境です。

この章の全環境の最後に「C」をつけることによって同様の働きをします。

1.1.3 枠付きの環境 ---- PicFrame[C]

定義

```

\begin{PicFrame[C]}(大きさ)[(左下座標)]
  - - - - -
\end{PicFrame[C]}

```

picture 環境と同時に枠も付けられる環境です。最後に「C」を付けることによって、中心を原点にします。枠の太さは `\FrameLen` で変更できます。初期値は `\FrameLen=0.8pt`、`\thicklines` と同じ太さです。

以下の例を参考にして下さい。

●基本

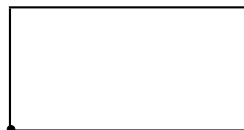
入力

```

\unitlength=4mm%
\begin{PicFrame}(8,4)
  \Knode*(0,0){0}
\end{PicFrame}

```

出力



●原点をずらす

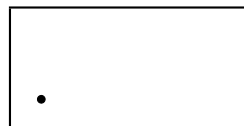
入力

```

\unitlength=4mm%
\begin{PicFrame}(8,4)(-1,-1)
  \Knode*(0,0){0}
\end{PicFrame}

```

出力



●中心が原点で枠の太さを 1mm にした例

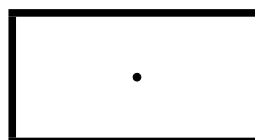
入力

```

\unitlength=4mm%
\FrameLen=1mm
\begin{PicFrameC}(8,4)
  \Knode*(0,0){0}
\end{PicFrameC}

```

出力



1.1.4 実線格子を表示する環境 ---- Pic[Frame]Grid[C]

定義

```
¥begin{Pic[Frame]Grid[C]}[間隔](大きさ)[(左下座標)]
```

```
-----
```

```
¥end{Pic[Frame]Grid[C]}
```

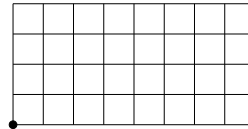
picture 環境で指定された環境内を、指定された間隔で実線の格子を引く命令です。間隔に置く数字の数によって動作が異なります。詳細は、9 ページの表を参考にして下さい。

●基本

入力

```
¥unitlength=4mm%
¥begin{PicGrid}(8,4)
  ¥Knode*(0,0){0}
¥end{PicGrid}
```

出力

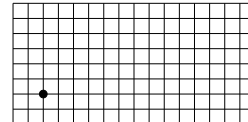


●原点をずらして間隔を半分にした

入力

```
¥unitlength=4mm%
¥begin{PicGrid}[0.5](8,4)(-1,-1)
  ¥Knode*(0,0){0}
¥end{PicGrid}
```

出力

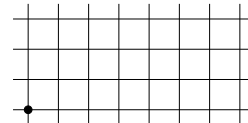


●原点を単位系以外にする

入力

```
¥unitlength=4mm%
¥begin{PicGrid}(8,4)(-0.5,-0.5)
  ¥Knode*(0,0){0}
¥end{PicGrid}
```

出力

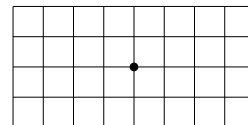


●中心を原点にした

入力

```
¥unitlength=4mm%
¥begin{PicGridC}(8,4)
  ¥Knode*(0,0){0}
¥end{PicGridC}
```

出力

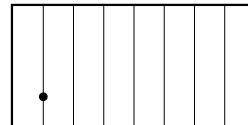


●枠付きで原点をずらして、横の線をひかない

入力

```
¥unitlength=4mm%
¥begin{PicFrameGrid}[1,0](8,4)(-1,-1)
  ¥Knode*(0,0){0}
¥end{PicFrameGrid}
```

出力

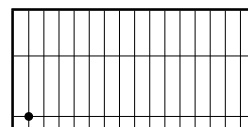


●枠付きで原点をずらし、横は 0.5 ずつ、縦は 2 ずつにした

入力

```
¥unitlength=4mm%
¥begin{PicFrameGrid}[0.5,2](8,4)(-0.5,-0.5)
  ¥Knode*(0,0){0}
¥end{PicFrameGrid}
```

出力

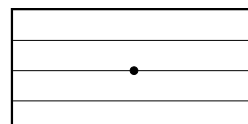


●枠付きで原点中心で、縦の線をひかない

入力

```
¥unitlength=4mm%
¥begin{PicFrameGridC}[0,1](8,4)
  ¥Knode*(0,0){0}
¥end{PicFrameGridC}
```

出力



1.1.5 点線格子を表示する環境 ---- Pic[Frame]Dot[C]

定義

```
¥begin{Pic[Frame]Dot[C]}[間隔](大きさ)[(左下座標)]
```

```
-----
```

```
¥end{Pic[Frame]Dot[C]}
```

picture 環境で指定された環境内を、指定された間隔で点線の格子を引く命令です。Ver.2.00 より点の間隔に絶対値を使用することをやめました。初期値は、1 単位に点を 6 個入れてあります。詳細は、10 ページの表を参考にしてください。

●基本

入力

```
¥unitlength=4mm%
¥begin{PicDot}(8,4)
  ¥Knode*(0,0){0}
¥end{PicDot}
```

出力

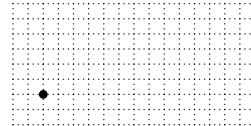


●原点をずらして間隔を半分にした

入力

```
¥unitlength=4mm%
¥begin{PicDot}[0.5](8,4)(-1,-1)
  ¥Knode*(0,0){0}
¥end{PicDot}
```

出力



●原点を単位系以外にする

入力

```
¥unitlength=4mm%
¥begin{PicDot}(8,4)(-0.5,-0.5)
  ¥Knode*(0,0){0}
¥end{PicDot}
```

出力

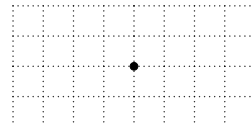


●中心を原点にした

入力

```
¥unitlength=4mm%
¥begin{PicDotC}(8,4)
  ¥Knode*(0,0){0}
¥end{PicDotC}
```

出力

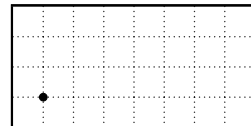


●枠付きで原点を単位系以外にする

入力

```
¥unitlength=4mm%
¥begin{PicFrameDot}(8,4)(-1,-1)
  ¥Knode*(0,0){0}
¥end{PicFrameDot}
```

出力

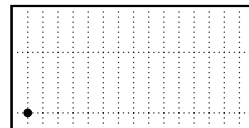


●枠付きで原点をずらし、横は 0.5 づつ、縦は 2 づつにした

入力

```
¥unitlength=4mm%
¥begin{PicFrameDot}[0.5,2](8,4)(-0.5,-0.5)
  ¥Knode*(0,0){0}
¥end{PicFrameDot}
```

出力

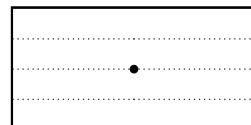


●枠付きで原点中心で縦方向は表示しない

入力

```
¥unitlength=4mm%
¥begin{PicFrameDotC}[0,1](8,4)
  ¥Knode*(0,0){0}
¥end{PicFrameDotC}
```

出力



1.2 Axes 環境 (Pic 環境に xy 座標軸を表示する)

1.2.1 基本環境 ---- Axes[Frame][C]

定義

`\begin{Axes[Frame][C]}`(大きさ)[(左下座標)]

`\end{Axes[Frame][C]}`

Axes 環境は、Pic 環境を宣言すると同時に、 xy 座標軸を表示します。座標軸を表示する以外は、Pic 環境、PicC 環境、PicFrame[C] 環境と同じ動作をします。中学校の教科書では矢印を描かないらしいので、矢印を描かない命令を追加しました。`\knoaxesttrue` とすると xy 軸の矢印の両方を、`\knoaxesXtrue` または `\knoaxesYtrue` とするとそれぞれ指定した軸の矢印が描かれません。座標軸の文字を変更する場合は 11 ページを参照して下さい。大きさで x または y の数値を 0 にするとその軸及び原点 O は描かれません。`\kaxesnolabeltrue` とすると座標も表示されません。

●基本

入力

```
\unitlength=4mm%
\begin{Axes}(8,4)
\end{Axes}
```

出力

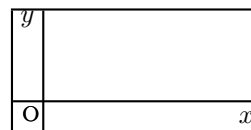


●原点をずらして枠付き

入力

```
\knoaxesttrue% 両方の矢印を表示しない
\unitlength=4mm%
\begin{AxesFrame}(8,4)(-1,-1)
\end{AxesFrame}
```

出力

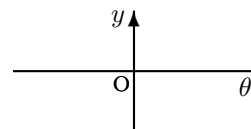


●中心を原点にして、横軸ラベルを θ にした

入力

```
\knoaxesXtrue% x 軸の矢印を表示しない
\unitlength=4mm%
\def\AxesLabelX{\theta}
\begin{AxesC}(8,4)
\end{AxesC}
```

出力



1.2.2 実線格子を表示する環境 ---- Axes[Frame]Grid[C]

定義

`\begin{Axes[Frame]Grid[C]}`[間隔](大きさ)[(左下座標)]

`\end{Axes[Frame]Grid[C]}`

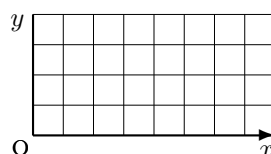
Axes 環境をベースに、PicGrid 環境が重なった環境です。間隔は、9 ページの表を参考にして下さい。

●基本

入力

```
\knoaxesYtrue% y 軸の矢印を表示しない
\unitlength=4mm%
\begin{AxesGrid}(8,4)
\end{AxesGrid}
```

出力



●横だけ間隔半分で原点をずらす

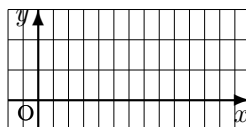
入力

```

\unitlength=4mm%
\begin{AxesGrid}[0.5,1](8,4)(-1,-1)
\end{AxesGrid}

```

出力



●中心を原点にした

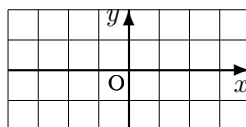
入力

```

\unitlength=4mm%
\begin{AxesGridC}(8,4)
\end{AxesGridC}

```

出力



●枠付きで中心を原点にした

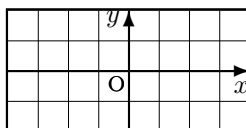
入力

```

\unitlength=4mm%
\begin{AxesFrameGridC}(8,4)
\end{AxesFrameGridC}

```

出力



1.2.3 点線格子を表示する環境 ---- Axes[Frame]Dot[C]

定義

```

\begin{Axes[Frame]Dot[C]}[間隔](大きさ)[(左下座標)]

```

```

-----

```

```

\end{Axes[Frame]Dot[C]}

```

Axes 環境をベースに、PicDot 環境が重なった環境です。間隔は、9 ページの表を参考にして下さい。

●基本

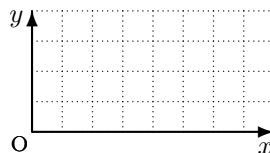
入力

```

\unitlength=4mm%
\begin{AxesDot}(8,4)
\end{AxesDot}

```

出力



●縦間隔半分で原点をずらす

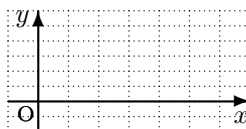
入力

```

\unitlength=4mm%
\begin{AxesDot}[1,0.5](8,4)(-1,-1)
\end{AxesDot}

```

出力



●中心を原点にした

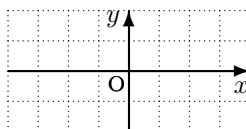
入力

```

\unitlength=4mm%
\begin{AxesDotC}(8,4)
\end{AxesDotC}

```

出力



●枠付きで中心を原点にした

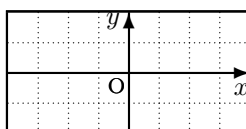
入力

```

\unitlength=4mm%
\begin{AxesFrameDotC}(8,4)
\end{AxesFrameDotC}

```

出力



1.2.4 目盛りと数値を入れる環境 ---- Axes[Frame]Scale[C]

定義

`\begin{Axes[Frame]Scale[C]}` [目盛間隔, 数値間隔] (大きさ) [(左下座標)]

`\end{Axes[Frame]Scale[C]}`

xy 座標軸に目盛りと数値をつけます。数値を表示したくない場合は、「数値間隔」を省略して下さい。数値の大きさは文字サイズで指定するか 12 ページの「`\KScaleNumberWordTypeX`」か「`\KScaleNumberWordTypeY`」を使用して下さい。目盛から数値までの距離は 17 ページの「`\WordSep`」で定義されています。「目盛間隔」も省略すると 1 単位で表示されます。この環境で使用する場合は、以下の制限があります。

- 目盛りも、数値も縦横同じ割合でしか表示されない。
- 目盛りの種類は 1 種類だけ。
- 実線格子や点線格子を同時に表示することが出来ない

以上の制約外で使いたいときは、12 ページを見て下さい。また、以下の目盛の長さを変える命令は使用することができます。

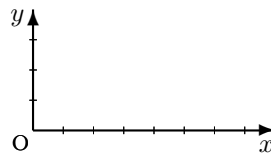
補助命令	説明	初期値
<code>\KSmallScaleLen</code>	目盛りの長さ (片側)	0.5mm

● 基本 (目盛だけ入れる)

入力

```
\unitlength=4mm%
\begin{AxesScale}(8,4)
\end{AxesScale}
```

出力

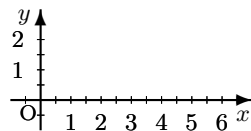


● 原点をずらして間隔半分

入力

```
\unitlength=4mm%
\small
\begin{AxesScale}[0.5,1](8,4)(-1,-1)
\end{AxesScale}
```

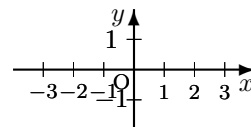
出力

● 中心を原点、 x 座標だけ `\footnotesize` サイズにして目盛を片側 1mm にした

入力

```
\unitlength=4mm%
\KSmallScaleLen=1mm%
\def\KScaleNumberWordTypeX{\footnotesize}
\begin{AxesScaleC}[1,1](8,4)
\end{AxesScaleC}
```

出力

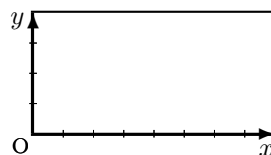


● Frame 環境での基本

入力

```
\unitlength=4mm%
\begin{AxesFrameScale}(8,4)
\end{AxesFrameScale}
```

出力

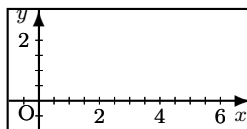


● 原点をずらして、目盛間隔を 0.5、数字を 2 ごとにした

入力

```
%unitlength=4mm%
%WordSep=2mm
%footnotesize
%begin{AxesFrameScale}[0.5,2](8,4)(-1,-1)
%end{AxesFrameScale}
```

出力

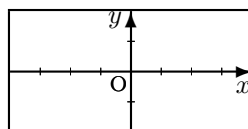


● Frame 環境で中心を原点にした

入力

```
%unitlength=4mm%
%begin{AxesFrameScaleC}(8,4)
%end{AxesFrameScaleC}
```

出力

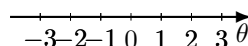


● 高さをゼロにして x 座標を θ にした

入力

```
%unitlength=4mm%
%def%AxesLabelX{${\theta}}
%begin{AxesScaleC}[1,1](8,0)
%end{AxesScaleC}
```

出力



1.3 Pic 環境, Axes 環境に点線格子の表示・非表示をする %kdrawtrue

ここまでの Pic 環境, Axes 環境のうち, Pic[C], PicFrame[C], Axes[C], AxesPicFrame[C] の環境で, 図形を作図中には (実線や点線) 格子を表示させたいが, 完成したら不要になることがあります。¹

完成版の環境の中から, Grid や Dot を取り除けばよいだけですが, 直すのが面倒くさいです。そこで, 「%kdrawtrue」を作成したい環境の前に 1 行入れるだけで, そのことをかなえてくれます。完成したらその行を削除するだけです。

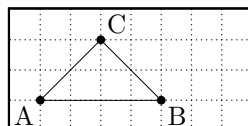
必要であれば, 作業中の図の最後に 「%kdrawfalse」を入れないと最後までこの命令が続きます。「%Knode, %KPath」の説明は, 20 ページと 29 ページを参照して下さい。

● PicFrame 環境で格子を引いて確認しながら三角形を作った

入力

```
%kdrawtrue
%unitlength=4mm%
%begin{PicFrame}(8,4)
  %Knode*(1,1){A}[%KSame][bl]
  %Knode*(5,1){B}[%KSame][br]
  %Knode*(3,3){C}[%KSame][tr]
  %KPath{ABCA}
%end{PicFrame}
```

出力

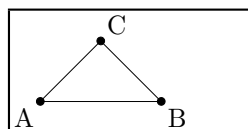


● PicFrame 環境で完成したから %kdrawtrue を削除した

入力

```
%unitlength=4mm%
%begin{PicFrame}(8,4)
  %Knode*(1,1){A}[%KSame][bl]
  %Knode*(5,1){B}[%KSame][br]
  %Knode*(3,3){C}[%KSame][tr]
  %KPath{ABCA}
%end{PicFrame}
```

出力



¹ 金沢光則氏がグラフの描画 (<http://www005.upp.so-net.ne.jp/mikana/schlmath/graph2g.pdf>) で「格子を消したい場合は, Dot を消せ」と書いてあったので作成しました

第2章 環境内で使用する

第1章で説明した環境の命令の内、いくつかは別に定義してあります。

2.1 格子関係マクロ

この節で説明するマクロの、指定された間隔で実線・点線の格子を引く命令です。間隔に置く数字の数によって動作が異なります。以下の表を参考にして下さい。

間隔に置く数	動作
省略	縦横とも単位ごとに実線・点線を引きます
1つ	縦横ともその間隔ごとに実線・点線を引きます
2つ	初めが横、次が縦の間隔ごとの実線・点線を引きます

数字に0を指定すると線を描きません

2.1.1 指定した範囲に実線格子を表示する ---- ¥KGrid

定義

¥KGrid[線の間隔](大きさ)[(左下座標)]

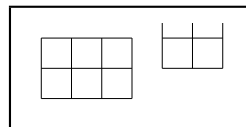
図の中の一部に格子を描きたいときに使います。次の例は「線の間隔」を指定していないので、間隔は1単位で描きます。右の図は1.5の高さなので上が欠けています。

●基本

入力

```
¥unitlength=4mm%
¥begin{PicFrame}(8,4)(-1,-1)
  ¥put(0,0){¥KGrid(3,2)}
  ¥put(4,1){¥KGrid(2,1.5)}
¥end{PicFrame}
```

出力



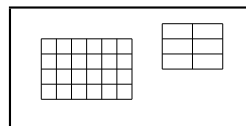
次に線の間隔を指定してみます。左の図は「線の間隔」で1つしか指定していないので、縦横同じ間隔になります。右の図は横と縦で大きさを変えています。

●線の間隔を指定

入力

```
¥unitlength=4mm%
¥begin{PicFrame}(8,4)(-1,-1)
  ¥put(0,0){¥KGrid[0.5](3,2)}
  ¥put(4,1){¥KGrid[1,0.5](2,1.5)}
¥end{PicFrame}
```

出力



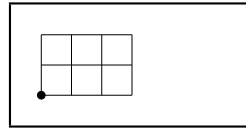
¥put を使用しない場合は、Pic 環境で指定した原点が有効になります。

● ¥put を使用しない場合

入力

```
¥unitlength=4mm%
¥begin{PicFrame}(8,4)(-1,-1)
  ¥KGrid(3,2)
  ¥Knode*(0,0){0}
¥end{PicFrame}
```

出力



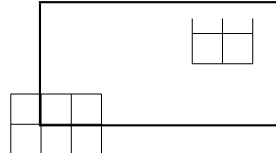
最後に「左下座標」を付け加えた場合は、¥put 命令をつけたのと同じことになります。

● 左下の座標を指定

入力

```
¥unitlength=4mm%
¥begin{PicFrame}(8,4)(-1,-1)
  ¥KGrid(3,2)(-2,-2)
  ¥put(4,1){¥KGrid(2,1.5)}
¥end{PicFrame}
```

出力



2.1.2 指定した範囲に点線格子を表示する ----- ¥KDot

¥KGrid マクロと同様に Pic 環境の中で自由に点線の格子を記述することが出来ます。

定義

¥KDot [線の間隔] (大きさ) [(左下座標)]

Ver.2.00 から、¥KDotSpace の使用を止めました。理由は、図形の拡大縮小で点線の交点が一致しないからです。そして、なによりも ¥dottedline が思うように描いてくれないからです。Ver.2.00 からは、指定した範囲の中にくつつ点を入れるかにしました。その分図形の拡大縮小で再定義をする場合がありますが、弧度法表示も楽になるなどのメリットも多いための変更です。初期値は、1 単位に 6 個の点を置きます。

新しい定義の補助命令は、

補助命令	説明	初期値
¥kdotinterval{ 範囲 }{ 点の数 }	x 軸, y 軸方向同時にいくつつ点を入れるか	範囲に 6 個
¥kdotintervalX{ 範囲 }{ 点の数 }	x 軸方向にいくつつ点を入れるか	範囲に 6 個
¥kdotintervalY{ 範囲 }{ 点の数 }	y 軸方向にいくつつ点を入れるか	範囲に 6 個

また、点も大きさや形の変更が可能になりました。初期値は、以下の表のとおり ¥circle*{0.01} です。

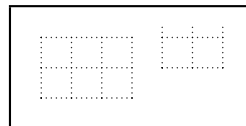
補助命令	説明	初期値
¥kdotpoint	点として置く物	¥circle*{0.01}

● 基本

入力

```
¥unitlength=4mm%
¥begin{PicFrame}(8,4)(-1,-1)
  ¥put(0,0){¥KDot(3,2)}
  ¥put(4,1){¥KDot(2,1.5)}
¥end{PicFrame}
```

出力

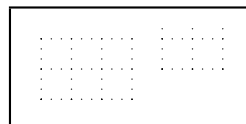


● 点線の間隔を 1 単位あたり 3 にした

入力

```
¥unitlength=4mm%
¥begin{PicFrame}(8,4)(-1,-1)
  ¥kdotinterval{1}{3}
  ¥put(0,0){¥KDot(3,2)}
  ¥put(4,1){¥KDot(2,1.5)}
¥end{PicFrame}
```

出力



2.2 平面座標系マクロ

xy 座標に関するマクロです。軸ラベルの x と y はそれぞれ、`¥AxesLabelX` と `¥AxesLabelY` に入っているので変更可能です。また、`¥kaxesnolabeltrue` と書いておくと、軸ラベルと原点が表示されません。軸ラベルと原点で

すが、
`¥KAxesLabelSep=3mm`
`¥KAxesLabel(右端の x 座標,0){b1}{1}{¥AxesLabelX}`
`¥KAxesLabel(0,上端の y 座標){1}{t1}{¥AxesLabelY}`
`¥KAxesLabelSep=2.3mm`
`¥KAxesLabel(0,0){b1}{1}{¥footnotesize¥ 0}`

で定義表示しますので、`¥kaxesnolabeltrue` で変更可能です。大きさのどちらかを 0 にするとその座標軸と軸ラベル、及び原点を表示しません。

2.2.1 座標軸を表示する ---- `¥KAxes`

定義

`¥KAxes(大きさ)[(左下座標)]`

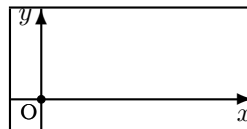
環境に関係なく、 xy 平面座標を表示します。

● 基本

入力

```
¥unitlength=4mm%
¥begin{PicFrame}(8,4)(-1,-1)
  ¥KAxes(8,4)(-1,-1)
  ¥Knode*(0,0){0}
¥end{PicFrame}
```

出力



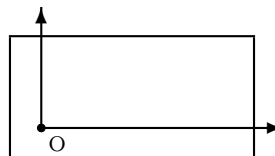
次は、あえて軸ラベルを右下の原点だけにして、数字によって枠からはみ出るようにしました。

● 枠からはみ出した例

入力

```
¥unitlength=4mm%
¥begin{PicFrame}(8,4)(-1,-1)
  ¥kaxesnolabeltrue
  ¥KAxesLabel(0,0){br}{1}
  {¥footnotesize¥rm 0}
  ¥KAxes(8,4)
  ¥Knode*(0,0){0}
¥end{PicFrame}
```

出力



2.2.2 環境内に座標軸を自動で表示する ---- `¥KAxesAuto`

定義

`¥KAxesAuto[方向]`

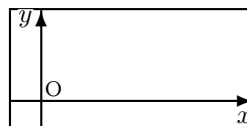
指定した環境通りに xy 平面座標を表示します。第 7 章で斜線の一部消すため、消した後でこれを使って座標軸を描くために作成しました。原点 O は方向を省略すると左下 (lb) に表示されますが、斜線に被らない方向へ表示されるため「rt,rb,lt,lb(文字の順番は問わない)」を方向にするとそこへ表示します。なお、`color` パッケージを使用している場合のみ、下の図形を消して原点 O 、 x 、 y を表示します。

● 原点 O を右上にした

入力

```
¥unitlength=4mm%
¥begin{PicFrame}(8,4)(-1,-1)
  ¥KAxesAuto[rt]
¥end{PicFrame}
```

出力



2.2.3 座標軸に目盛りをふる ---- ¥KScale

定義

¥KScale(小目盛り間隔)[(大目盛り間隔)]

7 ページの Axes[Frame]Scale[C] では目盛りの細かい設定が出来ません。このマクロを使用すると、目盛りのを2種類使用することが出来ます。大目盛り間隔を省略すると、Axes[Frame]Scale[C] と同じ目盛りになります。間隔のどちらかを0にすることで原点にも目盛りを振ることが出来ます。

補助命令	説明	初期値
¥KSmallScaleLen	小目盛りの長さ (片側)	0.5mm
¥KLargeScaleLen	大目盛りの長さ (片側)	1mm

小目盛り間隔を大目盛り間隔と同じにすると当然小目盛りは表示されません。

また、目盛りを軸の上下か、上か下の一方か選択できる命令が次の表の命令です。

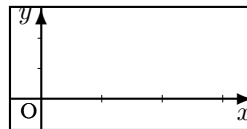
補助命令	説明	上下 (初期値)	上だけ	下だけ
¥KScaleDirectionX	x 軸の目盛りを表示する方向	0	1	-1
¥KScaleDirectionY	y 軸の目盛りを表示する方向	0	1	-1

●大目盛りを省略して x 軸は小目盛り 2、 y 軸は小目盛り 1 で軸の左だけ)

入力

```
¥unitlength=4mm%
¥begin{AxesFrame}(8,4)(-1,-1)
  ¥def¥KScaleDirectionY{-1}
  ¥KScale(2,1)
¥end{AxesFrame}
```

出力

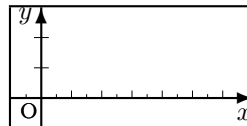


● x 軸は小目盛り 0.5、大目盛り 1 で目盛りは上だけ、 y 軸は大目盛り 1(小目盛りも 1) で軸の両側に表示

入力

```
¥unitlength=4mm%
¥begin{AxesFrame}(8,4)(-1,-1)
  ¥def¥KScaleDirectionX{1}
  ¥KScale(0.5,1)(1,1)
¥end{AxesFrame}
```

出力



2.2.4 座標軸に数字をふる ---- ¥KScaleNumber

軸に数字をふるマクロです。7 ページの Axes[Frame]Scale[C] では数字の細かい設定が出来ません。

定義

¥KScaleNumber[*][(軸に振る初期値)][(目盛りの間隔)]

数字を置く場所は次の命令で指定可能です。

補助命令	説明	初期値
¥KScaleNumberDirectionX	x 軸に置く数字の位置	b
¥KScaleNumberDirectionY	y 軸に置く数字の位置	l

数字の大きさは次の命令で変更可能です (初期値に ¥normalsize と書いていますが、実際は何も入っていません)。

補助命令	説明	初期値
¥KScaleNumberWordTypeX	x 軸に置く数字の大きさ	¥normalsize
¥KScaleNumberWordTypeY	y 軸に置く数字の大きさ	¥normalsize

●何も指定しないと 1 きざみずつ表示

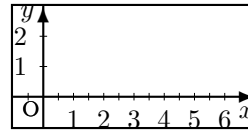
入力

```

¥unitlength=4mm%
¥begin{AxesFrame}(8,4)(-1,-1)
  ¥KScale(0.5,1)
  ¥KScaleNumber
¥end{AxesFrame}

```

出力

● x 軸は単位系を 2 にした (1 きざみ分で 2 ずつ表示)

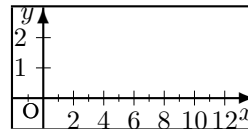
入力

```

¥unitlength=4mm%
¥begin{AxesFrame}(8,4)(-1,-1)
  ¥KScale(0.5,1)(1,1)
  ¥KScaleNumber(2,1)(1,1)
¥end{AxesFrame}

```

出力

● x 軸は単位系を 2 にして、上に表示 (2 きざみ分で 2 ずつ表示)、 y 軸の数字は右に表示

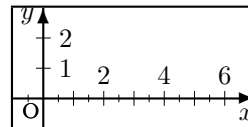
入力

```

¥unitlength=4mm%
¥begin{AxesFrame}(8,4)(-1,-1)
  ¥def¥KScaleNumberDirectionX{t}
  ¥def¥KScaleNumberDirectionY{r}
  ¥KScale(0.5,1)(1,1)
  ¥KScaleNumber(2,1)(2,1)
¥end{AxesFrame}

```

出力



¥KScaleNumber の後ろに * をつけ、さらに軸に振る初期値のどちらかを 0 にしたときだけ、0 の値 (度数法は 0°) を表示します。どちらかを 0 にすれば自動で振ることも可能でしたが、この後説明する x 軸と y 軸に別々に数値をふるとき表示すると支障をきたすので、あえて * をつけました。

●座標軸を x 軸だけにして、目盛りと数値も x 軸だけ表示

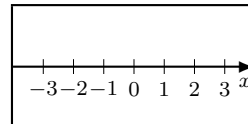
入力

```

¥unitlength=4mm%
¥begin{PicFrameC}(8,4)
  ¥footnotesize
  ¥KAxes(8,0)(-4,0)
  ¥KScale(1,0)
  ¥KScaleNumber*(1,0)
¥end{PicFrameC}

```

出力



Ver.2.00 から隠し命令の度数法表記の他に、弧度法表記を可能にしました。表示する分数も教科書的になっています (ユークリットの互除法を使用!)。置く数字を度数法表示と弧度法表示に変えることができます。

まず度数法表示ですが、¥kSNDegree>true と記述すると数字に「°」付きます。

●数字に度がついた (y 座標にも!)

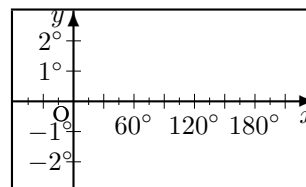
入力

```

¥unitlength=4mm%
¥begin{AxesFrame}(10,6)(-2,-3)
  ¥kSNDegree>true
  ¥KScale(0.5,1)(1,1)
  ¥KScaleNumber(60,1)(2,1)
  ¥kSNDegree>false
¥end{AxesFrame}

```

出力



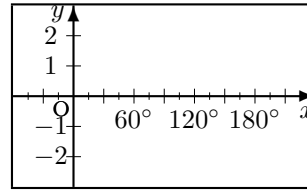
この命令の欠点としては、両方の座標に「°」がついてしまいます。そこで、(ほとんどが x 軸だと思うが) 片方の軸だけつけるには次のようにします。

● x 軸数字と y 軸数字を2回に分けて表示

入力

```
%unitlength=4mm%
%begin{AxesFrame}(10,6)(-2,-3)
  %kSNDegree>true
  %KScale(0.5,1)(1,1)
  %KScaleNumber(60,0)(2,0)
  %kSNDegree>false
  %KScaleNumber(0,1)(0,1)
%end{AxesFrame}
```

出力



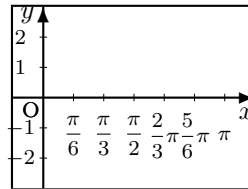
次に弧度法表示ですが、`%kSNRadiantrue` と記述すると数字が弧度法で表示されます。このとき、目盛りは度数で入力してください。分子が1のときは π を分子に寄せ、分母が1の時は分数表記をやめ、分数は既約分数になるようにしてあります。分数になるため `%WordSep` を増やしてあげるか x 軸の数値の大きさを小さく (`%small` 等) しないと軸に重なります。

● x 軸は5mm 離して弧度法表示、 y 軸は3mm の初期値に戻して普通の数字

入力

```
%unitlength=4mm%
%begin{AxesFrame}(8,6)(-1,-3)
  %footnotesize
  %kSNRadiantrue
  %KScale(1,1)
  %WordSep=5mm
  %KScaleNumber(30,0)(1,0)
  %kSNRadianfalse
  %WordSep=3mm
  %KScaleNumber(0,1)(0,1)
%end{AxesFrame}
```

出力



さて、度数法表示や弧度法表示の時の横軸と縦軸の大きさを同じにする、例えば横軸が弧度法の1のとき縦軸が1となるようにするためには、横の間隔を弧度法にします。`eclairth.sty` では次の値が定義されています。

定義	値	弧度法表示	補足
<code>%Piq</code>	0.78539816	$\pi/4$	q は quarter
<code>%Pih</code>	1.57079633	$\pi/2$	h は half
<code>%Pie</code>	3.14159265	π	e は equal?
<code>%Pii</code>	6.28318531	2π	i が2つで2つ分?

もう少し小さい値も欲しいので、`kplic.sty` では次の値も定義しました。

定義	値	弧度法表示	補足
<code>%Pit</code>	1.04719755	$\pi/3$	t は third
<code>%Pif</code>	0.62831853	$\pi/5$	f は fifth
<code>%Pis</code>	0.52359877	$\pi/6$	s は sixth

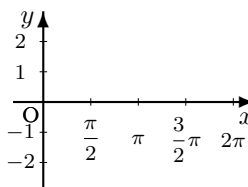
さて、環境から弧度法を意識して考えていきます。`Axes` 環境で 2π まで表示し負の方向に y 軸数字を表示するため横幅を8とし、縦幅は ± 2 まで表示するため6としたのが次です。目盛りを `%Pih` とし、数字も 90° (実際は弧度法を指定あるので $\pi/2$) 単位で表示、数字の間隔も `%Pih` ごとになりました。

● x 軸は $\pi/2$ ごと、 y 軸は普通の数字

入力

```
%unitlength=4mm%
%begin{Axes}(8,6)(-1,-3)
  %kSNRadiantrue
  %KScale(%Pih,1)
  %footnotesize
```

出力



```

%WordSep=5mm%
%KScaleNumber(90,0)(%Pih,0)
%kSNRadianfalse
%WordSep=3mm%
%KScaleNumber(0,1)(0,1)
%end{Axes}

```

2.3 応用例

この章の応用として、以下の例を参考にして活用して下さい。

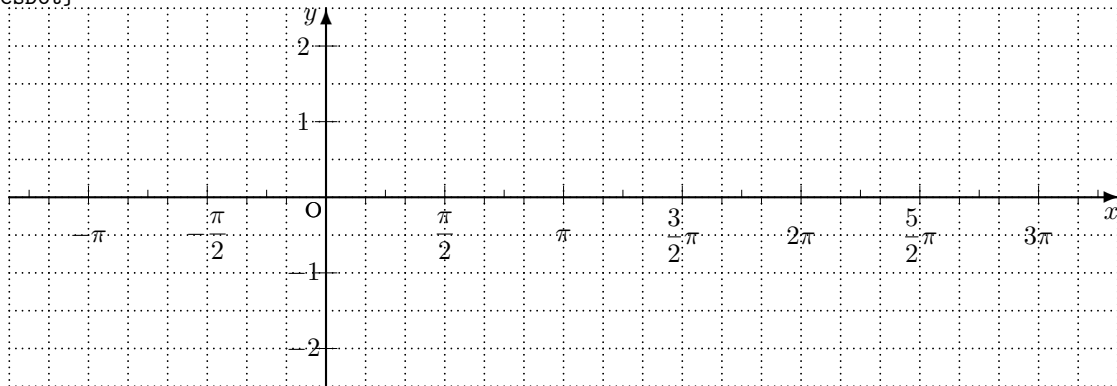
2.3.1 三角関数を書くための方眼紙

●格子をたくさんふる例 (格子が縦横同じに見えますが、横は 0.52359877 きざみです)

```

%unitlength=10mm%
%WordSep=5mm%
%kdotintervalX{%Pis}{6}% %pi/6 に 6 個の点
%kdotintervalY{1}{12}% 1 単位に 12 個
%begin{AxesDot}[%Pis,0.5](14.7,5)(-4.2,-2.5)
%KSmallScaleLen=1mm%
%KLargeScaleLen=1.5mm%
%def%KScaleDirectionX{1}
%KScale(%Piq,1)(%Pih,1)
%def%KScaleDirectionX{-1}
%KScale(%Pis,0)(%Pih,0)
%kSNRadiantrue
%WordSep=5mm%
%KScaleNumber(90,0)(%Pih,0)
%kSNRadianfalse
%WordSep=3mm%
%KScaleNumber(0,1)(0,1)
%end{AxesDot}

```

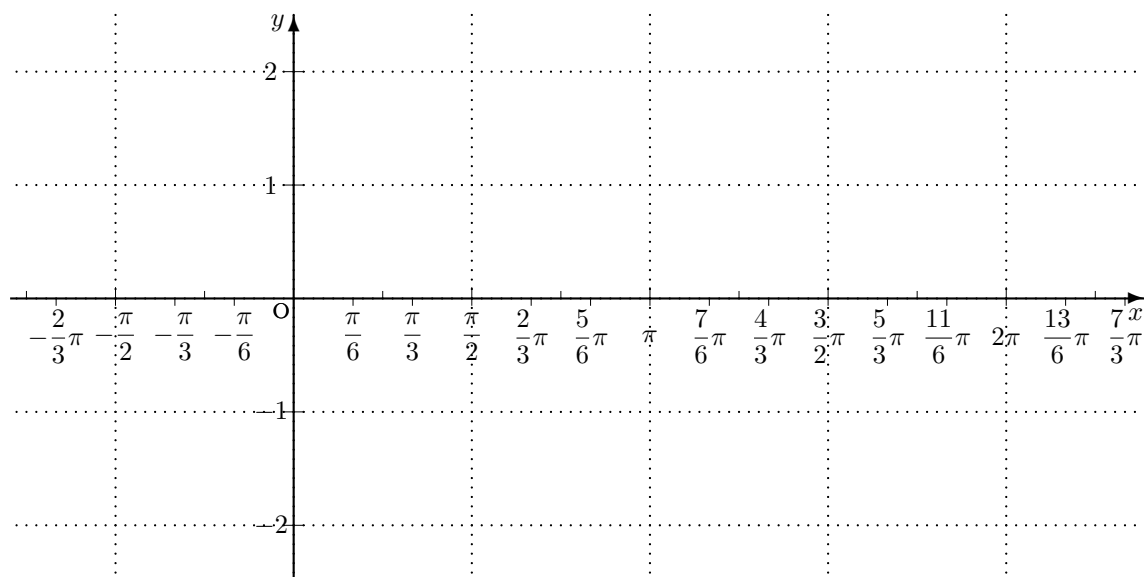


●格子は少なく、目盛りに数字をたくさんふる例

```

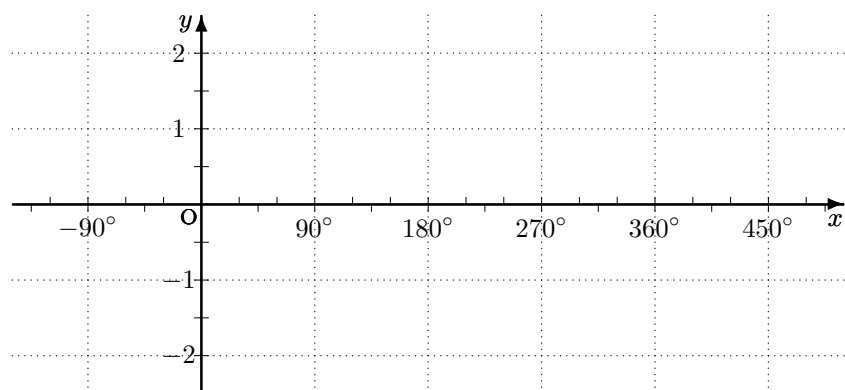
%unitlength=15mm
%WordSep=5mm
%kdotintervalX{%Pis}{6}% %pi/6 に 6 個の点
%kdotintervalY{1}{12}% 1 単位に 12 個
%begin{AxesDot}[%Pih,1](10,5)(-2.5,-2.5)
%KSmallScaleLen=1mm%
%KLargeScaleLen=1.5mm%
%def%KScaleDirectionX{1}
%KScale(%Piq,1)(%Pih,1)
%def%KScaleDirectionX{-1}
%KScale(%Pis,0)(%Pih,0)
%kSNRadiantrue
%WordSep=5mm%
%KScaleNumber(30,0)(%Pis,0)
%kSNRadianfalse
%WordSep=3mm%
%KScaleNumber(0,1)(0,1)
% %FDraw{%FESin}
%end{AxesDot}

```



●度数法の場合の例 (縦横の比は適当です)

```
%begin{center}
%unitlength=5mm%
%begin{AxesDot}[3,2](22,10)(-5,-5)
%KAxes(22,10)(-5,-5)
% 軸の目盛り
%def%KScaleDirectionX{1}
%KScale(1,1)(1,1)
%def%KScaleDirectionX{-1}
%KScale(1.5,0)(1.5,0)
% x 軸の数値
%kSNDegree>true
%KScaleNumber(90,0)(3,0)
% y 軸の数値
%kSNDegree>false
%KScaleNumber(0,1)(0,2)
%end{AxesDot}
%end{center}
```



第3章 表示文字と表示位置

この章では、この章以降に頻繁に出てくる命令をまとめました。

3.1 表示文字と表示位置

この章以降に、「表示文字名」「表示位置」に関する命令が多く出てきます。使用法は同じなので、ここでまとめておきます。

表示する文字、文字を置く位置、置く位置の微調整は省略可能ですが、使用する場合は「[]」（大カッコ）は必要です。

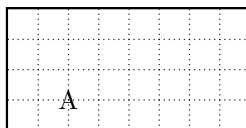
3.1.1 その座標を中心として文字を表示

次の例は、横 8、縦 4（1 単位は 4mm）、左下を原点とする方眼の座標（2, 1）の位置に A を表示します。でもこれくらいなら`\put(2,1){\makebox(0,0){A}}`で充分ではないでしょうか。

入力

```
\unitlength=4mm%
\begin{PicFrameDot}(8,4)
  \Knode(2,1){A}{\KSame}
\end{PicFrameDot}
```

出力



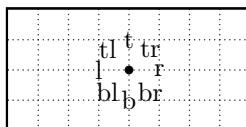
3.1.2 文字を置く位置

文字を置く位置ですが、座標に対してどの位置に置くかを指定します。`picture` 環境の`\framebox` や `\makebox` の位置と同じようにしています。8 方向の例と表示位置を参考にして下さい。

入力

```
\begin{PicFrameDot}(8,4)
  \WordSep=4mm%
  \Knode*(4,2){A}[r][r]
  \Knode(4,2){A}[tr][tr]
  \Knode(4,2){A}[t][t]
  \Knode(4,2){A}[tl][tl]
  \Knode(4,2){A}[l][l]
  \Knode(4,2){A}[bl][bl]
  \Knode(4,2){A}[b][b]
  \Knode(4,2){A}[br][br]
\end{PicFrameDot}
```

出力



3.1.3 座標からの距離

座標からの距離は`\WordSep` で定義されています。デフォルトは 3mm です。この数値を変えるだけで座標からの距離が変わります。何故絶対指定にしたのかというと、図形の大きさを変えても文字との距離が変わらない方が図を修正する方が楽だからです。

●文字の位置を左は 4mm, 右は 8mm にした例

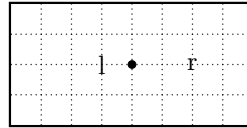
入力

```

%unitlength=4mm
%begin{PicFrameDot}(8,4)
  %WordSep=4mm%
  %Knode*(4,2){A}[l][l]
  %WordSep=8mm%
  %Knode(4,2){A}[r][r]
%end{PicFrameDot}

```

出力



●%Knode を使用しないで図を縮小した例

入力

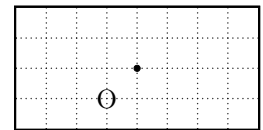
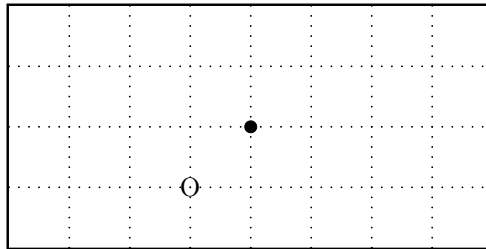
```

%def%例図{%
  %begin{PicFrameDot}(8,4)
    %put(4,2){%circle*{0.2}}
    %put(3,1){%makebox(0,0){0}}
  %end{PicFrameDot}
}% End of %例図

%unitlength=8mm%
%例図
%unitlength=4mm%
%例図

```

出力



●%Knode を使用して図を縮小した例

入力

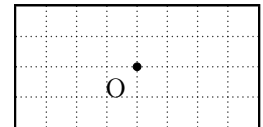
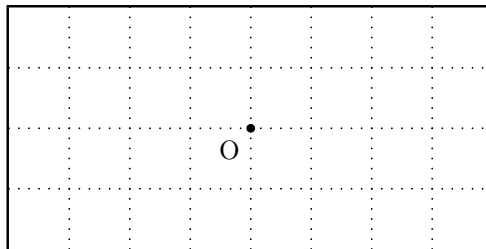
```

%def%例図{%
  %begin{PicFrameDot}(8,4)
    %WordSep=4mm%
    %Knode*(4,2){0}{%KSame}[b1]
  %end{PicFrameDot}
}% End of %例図

%unitlength=8mm
%例図
%unitlength=4mm
%例図

```

出力



これを指定したのは、文字と線が重ならないための工夫です。

3.1.4 置く位置の微調整

最後の定義は前にふれた置く位置を微調整します。次の例を見て下さい。実際に置く文字が多いと、こういう結果になってしまいます。

●置く文字多い例

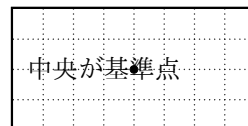
入力

```

%unitlength=4mm%
%begin{PicFrameDot}(8,4)
  %WordSep=4mm%
  %Knode*(4,2){A}[中央が基準点][l]
%end{PicFrameDot}

```

出力



●文字を置く場所の基準点を右にして表示

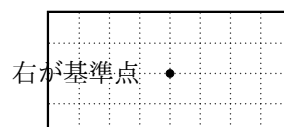
入力

```

%unitlength=4mm%
%begin{PicFrameDot}(8,4)
  %WordSep=4mm%
  %Knode*(4,2){A}[右が基準点][l,r]
%end{PicFrameDot}

```

出力



第4章 点を定義

`picture` 環境では、線を多く表示するとき、座標を何回も書かなければなりません。それが、`picture` 環境を使いにくくしている原因のひとつであると思います。しかし、「 \LaTeX 自由自在」を読んだとき、点を定義しておいて何回も使用できることがわかり、その本を見よう見まねでスタイルファイルを書いたのがこの `kpic.sty` の始まりです。

この `kpic.sty` の作成の開始は、「 \LaTeX 自由自在」の `\node` の改良から始まりました。点を定義する方法として、平面は「平面座標」「極座標」の2つを基本としました。空間座標も作っていましたが、現在は平面に絞り作業はほぼあきらめ、このマニュアルからは削除してあります。また、定義した点がわかりやすいように第5章以降の命令も使用しています。

4.1 点の定義

4.1.1 共通項目

ここでは、この章で扱われる命令の共通な項目を記載しました。

- 座標名はアルファベット 1 文字で定義 (`A~Z,a~z` 等で定義)
- 半角「*」を付けると一部を除いてその座標に塗りつぶされた点 (`\circle*{直径}`) を置きます。直径は `\KBullet` で定義され、初期値は `\KBullet=1mm` です。
- 「表示文字」「文字位置」については第3章を参照して下さい。
- 「表示文字」については以下の命令をこの章のほぼ全部で使うことができます。

<code>\KSame</code>	定義した座標名を表示する
<code>\Cnode</code>	定義した座標を <code>()</code> をつけて表示する
<code>\CnodeX</code>	定義した x 座標を表示する
<code>\CnodeY</code>	定義した y 座標を表示する
<code>\CnodeName</code>	定義した座標名と座標 <code>()</code> をつけて表示する

- 上記の、座標に関して数字は初期値「`\def\KFigure{1}`」で小数第1位までに丸めていますので、必要であれば数字を変えて下さい。
- 上記の、`\Cnode` の座標の間の空白ですが、「`\def\KSpace{ }`」で定義しています。
- この章の `node` 座標を定義する命令の前に「`\KMaskingtrue`」と書いておくと下にある図形を消して表示します。「`\KMaskingfalse`」が出てくるまで有効です。また、文字の余白も変更可能です。「`\KMaskingPadding`」で設定変更です。初期値は「`\KMaskingPadding=0pt`」です。(要 color)

4.1.2 平面座標 (xy 座標) で定義 ---- ¥Knode

定義

1. ¥Knode[*] (x 座標, y 座標) {座標名} [表示文字] [文字位置]
2. ¥Knode[*] [{node で定義した座標}] (x 座標, y 座標) {座標名} [表示文字] [文字位置]

(x 座標, y 座標) を指定した「座標名」で定義する命令です。「表示文字」と「文字位置」は省略可能です。「文字位置」だけ記載することは無意味です。

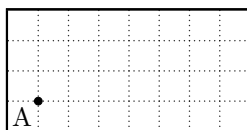
2 は、定義済みの座標を基準 (仮原点) として座標を定義します。「¥Knode」単独では手計算のできるものであまり意味がありませんが、次の「¥Pnode」と組み合わせると有効でしょう。

●点 A(1,1) を定義

入力

```
¥unitlength=4mm
¥begin{PicFrameDot}(8,4)
  ¥Knode*(1,1){A}[¥KSame][bl]
¥end{PicFrameDot}
```

出力

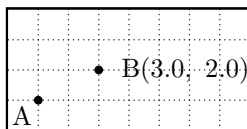


●点 A を基準として点 B(3,2) を定義

入力

```
¥unitlength=4mm
¥begin{PicFrameDot}(8,4)
  ¥Knode*(1,1){A}[¥KSame][bl]
  ¥Knode*{A}(2,1){B}[¥CnodeName][r,l]
¥end{PicFrameDot}
```

出力



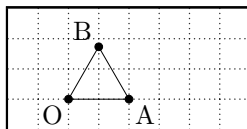
¥Knode の欠点というより、直交座標の欠点ですが、角度が出てくる図形、例えば正三角形の3点を定義する場合のように小数の座標が出てきてしまいます。次のマクロのように2点を整数にしても、残りの1点は小数となってしまいます。

●¥Knode で正三角形を定義

入力

```
¥unitlength=4mm%
¥begin{PicFrameDot}(8,4)(-2,-1)
  ¥Knode*(0,0){O}[¥KSame][lb]
  ¥Knode*(2,0){A}[¥KSame][rb]
  ¥Knode*(1,1.73205){B}[¥KSame][lt]
  ¥KPath{OABO}
¥end{PicFrameDot}
```

出力



このように複雑にしないで、簡単に点を定義する命令として、次の¥Pnode を作成しました。

4.1.3 極座標で定義 ---- ¥Pnode

定義

- ¥Pnode[*] [(元になる点の x 座標, 元になる点の y 座標)] (大きさ, 角度) {座標名} [表示文字] [文字位置]
- ¥Pnode[*] [{node で定義した座標}] (大きさ, 角度) {座標名} [表示文字] [文字位置]

前述したように¥Knode だと、正三角形を表示するのに事前に計算をしなければなりません。また普段数学で使用する角度以外の特殊な角度は三角関数表を見るか、関数電卓で調べなければなりません。そこで極座標の考えを利用して、基準になる点からの距離と角度で座標を定義します。角度は「度 (degree)」使用して下さい。ラジアンでは小数表記なのでわかりにくいです。極座標を内部で直交座標に変換して定義しています。このために必要な「 $\sin \theta$ と $\cos \theta$ 」は eclairth.sty で用意されているのできました。

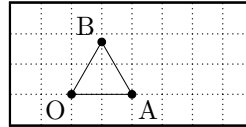
ここでも表示文字の代わりに`\Pnode`の5つの命令が利用できます。ただし、座標は計算された数値が表示されるので、小数以下の桁数(無しも含めて)を考えなくてはなりません。小数点以下の桁数は`\KFigure`で設定できます。詳しいことは「初期設定(2012年1月現在未作成)」を参照して下さい。

● `\Pnode` で正三角形を定義

入力

```
\unitlength=4mm%
\begin{PicFrameDot}(8,4)(-2,-1)
  \Pnode*(0,0){O}[\KSAME][lb]
  \Pnode*(2,0){A}[\KSAME][rb]
  \Pnode*(2,60){B}[\KSAME][lt]
  \KPath{OABO}
\end{PicFrameDot}
```

出力

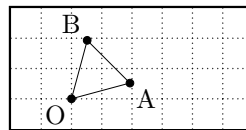


● `\Pnode` で 15° 回転した正三角形を定義

入力

```
\unitlength=4mm%
\begin{PicFrameDot}(8,4)(-2,-1)
  \Pnode*(0,0){O}[\KSAME][lb]
  \Pnode*(2,15){A}[\KSAME][rb]
  \Pnode*(2,75){B}[\KSAME][lt]
  \KPath{OABO}
\end{PicFrameDot}
```

出力

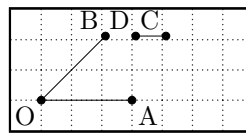


● こんなこともできます

入力

```
\unitlength=4mm%
\begin{PicFrameDot}(8,4)(-1,-1)
  \Pnode*(0,0){O}[\KSAME][lb]
  \Pnode*(3,0){A}[\KSAME][rb]
  \Pnode*(3,45){B}[\KSAME][lt]
  \Pnode*(2,0)(3,45){C}[\KSAME][lt]
  \Pnode*(C)(1,180){D}[\KSAME][lt]
  \KPath{AOB,CD}
\end{PicFrameDot}
```

出力

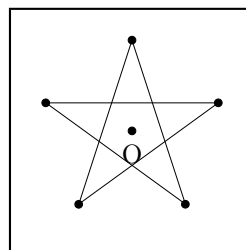


● 星形に結ぶ、72°おきに座標を定義

入力

```
\unitlength=4mm%
\begin{PicFrameC}(8,8)
  \Pnode*(0,0){O}[\KSAME][b]
  \Pnode*(3,18){A}
  \Pnode*(3,90){B}
  \Pnode*(3,162){C}
  \Pnode*(3,234){D}
  \Pnode*(3,306){E}
  \KPen{\drawline}
  \KPath{ACEBDA}
\end{PicFrameC}
```

出力

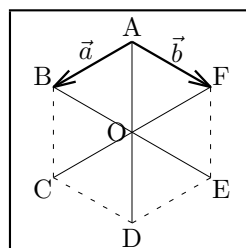


● 六角形問題

入力

```
\unitlength=4mm%
\begin{PicFrameC}(8,8)
  \WordSep=2mm%
  \Pnode(0,0){O}[\KSAME][l]
  \Pnode(3,90){A}[\KSAME][t]
  \Pnode(3,150){B}[\KSAME][lt]
  \Pnode(3,210){C}[\KSAME][lb]
  \Pnode(3,270){D}[\KSAME][b]
  \Pnode(3,330){E}[\KSAME][rb]
  \Pnode(3,30){F}[\KSAME][rt]
  \KPath[\dashline{0.2}]{BCDEF}
  \KPath[\drawline]{AD,BE,CF}
  \thicklines
  \KVec{AB}[\$vec{a}\$][t,r]
  \KVec{AF}[\$vec{b}\$][t,l]
  \thinlines
\end{PicFrameC}
```

出力



4.1.4 Pic 環境の四隅を一括定義 ---- `\KnodeCorner`

定義

`\KnodeCorner`{左下にする座標名, 右下にする座標名, 右上にする座標名, 左上にする座標名}

`Pic` 環境の四隅の座標を一括で定義する命令です。左下から反時計回りで順番に 4 つ指定して下さい。これも事前に座標名で予約しても良かったのですが、普段使用することはないと思います。第 7 章で斜線を消すときに使用するくらいでしょう。

4.1.5 Pic 環境の四隅を個別に定義 ---- `\KnodeLB`, `\KnodeRB`, `\KnodeRT`, `\KnodeLT`

定義

1. `\KnodeLB[*]`{座標名}[表示文字][文字位置] ... 左下の座標を座標名で定義
2. `\KnodeRB[*]`{座標名}[表示文字][文字位置] ... 右下の座標を座標名で定義
3. `\KnodeRT[*]`{座標名}[表示文字][文字位置] ... 右上の座標を座標名で定義
4. `\KnodeLT[*]`{座標名}[表示文字][文字位置] ... 左上の座標を座標名で定義

`Pic` 環境の四隅の座標を個別に「座標名」で定義する命令です。`\KnodeCorner` が 4 つの座標を同時に必要となるので少なければこちらがいいと思い作成しました。こちらも事前に座標名で予約しても良かったのですが、普段使用することはないと思います。第 7 章で斜線を消すときに使用するくらいでしょう。

4.1.6 直線と Pic 環境の交点を定義 ---- `\KLineLeft(or Right or Both)Edge`

定義

1. `\KLineLeftEdge[*]`{直線の傾き, y 切片}{座標名}[表示文字][文字位置]
... 直線と `Pic` 環境の交点の左側を座標名で定義
2. `\KLineRightEdge[*]`{直線の傾き, y 切片}{座標名}[表示文字][文字位置]
... 直線と `Pic` 環境の交点の右側を座標名で定義
3. `\KLineBothEdge`{直線の傾き, y 切片}{左座標名, 右座標名}
... 直線と `Pic` 環境の交点の両方を座標名で定義

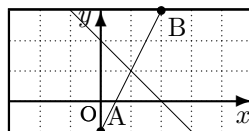
直線を傾きと y 切片で指定したとき $x = a$ でない限り環境から出るとき、必ず左側と右側に分かります。ここでは、その時の座標を「座標名」定義します。`\KLineBothEdge` は定義するだけで、座標名の表示は出来ません。これも第 7 章で斜線を消すときと直線を描くために作成しました。

●直線の両端を傾きと y 切片で定義

入力

```
\unitlength=4mm%
\begin{AxesFrameDot}(8,4)(-3,-1)
  \KLineLeftEdge*{2,-1}{A}{\KSame}[lt]
  \KLineRightEdge*{2,-1}{B}{\KSame}[rb]
  \KLineBothEdge*{-1,2}{C,D}
  \KPath{AB,CD}
\end{AxesFrameDot}
```

出力



4.2 定義した点から新たな点を定義

`\Knode` 等で定義した点を元に、新たな点を作成できます。ここで、作成した点も定義されますので、その座標を `\Knode` と同様に使用することができます。作成可能な点は、この節の以下で記載しているものです。この節のマクロ名は、長い命令でしかも他のマクロとぶつからないと思い、初めの「K」を省略しています。

4.2.1 内分点 ---- `\Inode`

定義

`\Inode[*]{両端の座標}(内分する比){座標名}[文字][方向, 文字位置]`

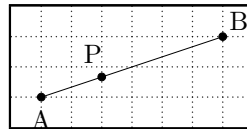
定義した 2 点を元に内分点を計算します。

- 線分 AB を 1:2 の比に内分

入力

```
\unitlength=4mm%
\begin{PicFrameDot}(8,4)
  \Knode*(1,1){A}[\KSAME][b]
  \Knode*(7,3){B}[\KSAME][rt]
  \Inode*{AB}(1:2){P}[\KSAME][t,r]
  \KPen{\drawline}
  \KPath{AB}
\end{PicFrameDot}
```

出力



4.2.2 外分点 ---- `\Enode`

定義

`\Enode[*]{両端の座標}(外分する比){座標名}[文字][方向, 文字位置]`

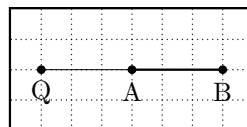
定義した 2 点を元に外分点を計算します。

- 線分 AB を 1:2 比に外分

入力

```
\unitlength=4mm%
\begin{PicFrameDot}(8,4)
  \Knode*(4,2){A}[\KSAME][b]
  \Knode*(7,2){B}[\KSAME][b]
  \thicklines
  \KLine{AB}
  \thinlines
  \Enode*{AB}(1:2){Q}[\KSAME][b]
  \KLine{QA}
\end{PicFrameDot}
```

出力

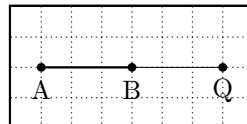


- 線分 AB を 2:1 比に外分

入力

```
\unitlength=4mm%
\begin{PicFrameDot}(8,4)
  \Knode*(1,2){A}[\KSAME][b]
  \Knode*(4,2){B}[\KSAME][b]
  \thicklines
  \KLine{AB}
  \thinlines
  \Enode*{AB}(2:1){Q}[\KSAME][b]
  \KLine{QB}
\end{PicFrameDot}
```

出力



4.2.3 2点から指定した距離の交点 ---- `\TwoCirclesRight(Left)`

定義

`\TwoCirclesRight[*]{両端の座標}(両端からの距離){座標名}[文字][方向, 文字位置]`
`\TwoCirclesLeft[*]{両端の座標}(両端からの距離){座標名}[文字][方向, 文字位置]`

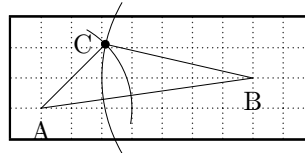
定義してある2点から指定した距離の点を計算します。以下の例を参考にして下さい。

- A から 3, B から 5 の距離の点の A から左側

入力

```
\unitlength=4mm%
\begin{PicFrameDot}(10,4)(-1,-1)
  \Knode(0,0){A}[\KSame][b]
  \Knode(7,1){B}[\KSame][b]
  \TwoCirclesLeft*{AB}(3,5){C}[\KSame][l]
  \KPath{CABC}
  \KArc[6]{A}(350,60)
  \KBrc[10]{B}(150,210)
\end{PicFrameDot}
```

出力

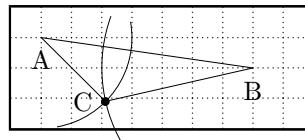


- A から 3, B から 5 の距離の点の A から右側

入力

```
\unitlength=4mm%
\begin{PicFrameDot}(10,4)(-1,-1)
  \Knode(0,2){A}[\KSame][b]
  \Knode(7,1){B}[\KSame][b]
  \TwoCirclesRight*{AB}(3,5){C}[\KSame][l]
  \KPath{CABC}
  \KArc[6]{A}(280,10)
  \KBrc[10]{B}(160,210)
\end{PicFrameDot}
```

出力



4.2.4 垂線の交点の座標 ---- `\Perpendicularfoot`

定義

`\Perpendicularfoot[*]{線分の2点}{垂線上の他の点}{垂点座標名}[文字][方向, 文字位置]`

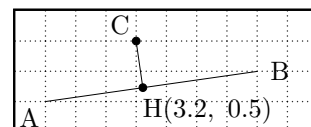
2点の線分上に他の点からの垂線の交点の座標を求めます。以下の例を参考にして下さい。

- 線分 AB 上の, 点 C からの垂線の交点の座標 H

入力

```
\unitlength=4mm%
\begin{PicFrameDot}(10,4)(-1,-1)
  \Knode(0,0){A}[\KSame][b1]
  \Knode(7,1){B}[\KSame][r]
  \Knode*(3,2){C}[\KSame][t1]
  \Perpendicularfoot*{AB}{C}{H}[\CnodeName][b,1]
  \KPath{AB,CH}
\end{PicFrameDot}
```

出力



4.2.5 2つの線分の交点の座標 ---- `\Intersection`

定義

`\Intersection[*]{直線上の2点}{他の直線上の2点}{交点座標名}[文字][方向, 文字位置]`

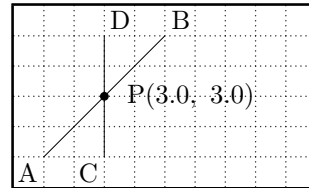
定義された2点を通る直線と、別に定義された2点を通る直線の交点の座標を求めます。以下の例を参考にして下さい。

●線分 AB と線分 CD の交点

入力

```
\unitlength=4mm%
\begin{PicFrameDot}(10,6)
  \Knode(1,1){A}[\KSame][bl]
  \Knode(5,5){B}[\KSame][rt]
  \Knode(3,1){C}[\KSame][bl]
  \Knode(3,5){D}[\KSame][tr]
  \Intersection*{AB}{CD}{P}[\CnodeName][r,l]
  \KPath{AB,CD}
\end{PicFrameDot}
```

出力

4.2.6 三角形の重心 ---- `\Barycenter`

定義

`\Barycenter[*]{3点の座標}{座標名}[文字][方向, 文字位置]`

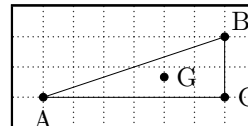
定義された3点を結ぶ三角形の重心の座標を求めます。

●三角形 ABC の重心

入力

```
\unitlength=4mm%
\begin{PicFrameDot}(8,4)
  \Knode*(1,1){A}[\KSame][b]
  \Knode*(7,3){B}[\KSame][rt]
  \Knode*(7,1){C}[\KSame][r]
  \Barycenter*{ABC}{G}[\KSame][r]
  \KPen{\drawline}
  \KPath{ABCA}
\end{PicFrameDot}
```

出力

4.2.7 三角形の外心 ---- `\Circumcenter`

定義

`\Circumcenter[*]{3点の座標}{座標名}[文字][方向, 文字位置]`

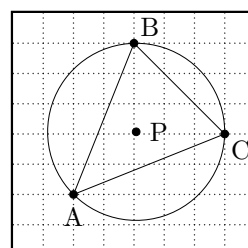
定義された3点を結ぶ三角形の外心の座標を求めます。その直後なら、半径は「`\KRadius`」、直径は「`\KDiameter`」に記憶されますので使用可能です。以下の例を参考にして下さい。

●三角形 ABC の外心

入力

```
\unitlength=4mm%
\begin{PicFrameDot}(8,8)
  \Knode*(2,2){A}[\KSame][b]
  \Knode*(4,7){B}[\KSame][rt]
  \Knode*(7,4){C}[\KSame][rb]
  \Circumcenter*{ABC}{P}[\KSame][r]
  \Kput{P}{\circle{\KDiameter}}
  \KPath{ABCA}
\end{PicFrameDot}
```

出力



4.2.8 三角形の内心 ---- ¥Incenter

定義

¥Incenter[*]{3 点の座標}{座標名}[文字][方向, 文字位置]

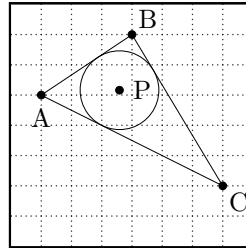
定義された3点を結ぶ三角形の内心の座標を求めます。その直後なら、半径は「¥KRadius」、直径は「¥KDiameter」に記憶されますので使用可能です。以下の例を参考にしてください。

●三角形 ABC の内心

入力

```
¥unitlength=4mm%
¥begin{PicFrameDot}(8,8)
  ¥Knode*(1,5){A}[¥KSame][b]
  ¥Knode*(4,7){B}[¥KSame][rt]
  ¥Knode*(7,2){C}[¥KSame][rb]
  ¥Incenter*{ABC}{P}[¥KSame][r]
  ¥Kput{P}{¥circle{¥KDiameter}}
  ¥KPath{ABCA}
¥end{PicFrameDot}
```

出力



4.2.9 ある点を基準として点の拡大縮小 (点対称を含む) ---- ¥Tnode

定義

¥Tnode[*]{元の座標名}(対称となる平面座標)[拡大縮小率]{座標名}[文字][方向, 文字位置]

¥Tnode[*]{元の座標名}{対称となる座標名}[拡大縮小率]{座標名}[文字][方向, 文字位置]

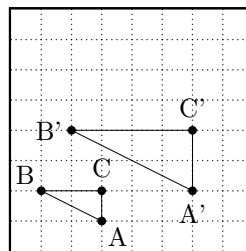
定義された点(元の座標名)を対称となる平面座標または定義された点(座標名)を基準として指定された拡大率、縮小率で新たに点を定義します。拡大率・縮小率は負の値を入れると対称となる点の反対側に定義されます(-1を指定すると点対称)。以下の例を参考にしてください。

●原点を基準にして2倍した

入力

```
¥unitlength=4mm%
¥begin{PicFrameDot}(8,8)
  ¥Knode*(0,0){O}
  ¥Knode*(3,1){A}[¥KSame][br]
  ¥Knode*(1,2){B}[¥KSame][lt]
  ¥Knode*(3,2){C}[¥KSame][t]
  ¥Tnode*{A}{O}[2]{D}[A'][b]
  ¥Tnode*{B}{O}[2]{E}[B'][l]
  ¥Tnode*{C}{O}[2]{F}[C'][t]
  ¥KPath{ABCA,DEFD}
¥end{PicFrameDot}
```

出力

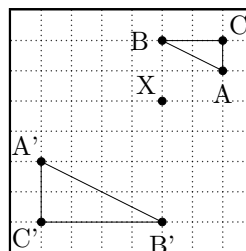


●点 X を基準にして -2 倍した

入力

```
¥unitlength=4mm%
¥begin{PicFrameDot}(8,8)
  ¥Knode*(7,6){A}[¥KSame][b]
  ¥Knode*(5,7){B}[¥KSame][l]
  ¥Knode*(7,7){C}[¥KSame][tr]
  ¥Knode*(5,5){X}[¥KSame][tl]
  ¥Tnode*{A}{X}[-2]{D}[A'][lt]
  ¥Tnode*{B}{X}[-2]{E}[B'][b]
  ¥Tnode*{C}{X}[-2]{F}[C'][bl]
  ¥KPath{ABCA,DEFD}
¥end{PicFrameDot}
```

出力



4.2.10 円の 2 接線の交点 ---- `\IntersectionOfTangent`

定義

`\IntersectionOfTangent[*]{中心, 接点, 接点 (の座標名)}{定義文字名}[文字][方向, 文字位置]`

円の 2 接線の交点の座標を定義します。次の例は、まず 3 点 A,B,C を定義してその外接円の中心を点 O として定義します。次に今回のマクロで点 A と点 B の接線の交点を点 D で定義します。線分 AD と線分 BD を表示すれば完成なのですが、線分外の点を外分点で定義して結んでいます。

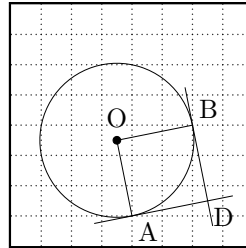
●円の接線の交点

入力

```
\unitlength=4mm%
\begin{PicFrameDot}(8,8)
  \Knode(4,1){A}[\KSame][br]
  \Knode(6,4){B}[\KSame][rt]
  \Knode(1,4){C}
  \Circumcenter*{ABC}{O}[\KSame][t]
  \Kput{O}{\Kcircle{\KDiameter}}
  \KPen{\drawline}
  \IntersectionOfTangent{O,A,B}{D}[\KSame][br]
  \KPath{OA,OB,AD,BD}

  \KPath{OA,OB}
  \ENode{AD}{1:3}{P}
  \ENode{AD}{4:1}{Q}
  \KPath{PQ}
  \ENode{BD}{1:3}{P}
  \ENode{BD}{4:1}{Q}
  \KPath{PQ}
\end{PicFrameDot}
```

出力



4.3 定義した点を利用する

ここまで新たに定義した点を利用して便利な命令を作成してみました。

4.3.1 定義された点を元の点とする ---- `\Kput`

定義

`\Kput{定義名}`

定義された点を「`\Kput`」のようにして使うことができます。使い方は、ここまで何度か出てきているのでそちらをご覧ください。

4.3.2 2点を半径とする円を描く ---- `\Kcircle`

定義された 2 点を半径とする円を描きます。どちらかを中心とすることで 2 点の距離 (半径) を計算せずに表示できます。

定義

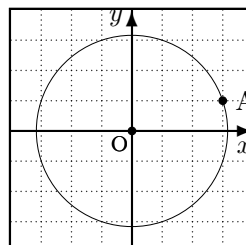
`\Kcircle{2 点の定義名}`

●入力した 2 点の node 座標を半径とする円を描く

入力

```
\unitlength=4mm%
\begin{AxesFrameDotC}(8,8)
  \Knode*(0,0){O}
  \Knode*(3,1){A}[\KSame][r]
  \Kput{O}{\Kcircle{OA}}
\end{AxesFrameDotC}
```

出力



4.3.3 定義された点から軸へ破線を引く ---- `\PointDashX`, `\PointDashY`, `\PointDashXY`

定義

`\PointDashX[*] [stretch]{破線の長さ}{座標名}[表示文字][文字位置]` ... 定義点から x 軸へ破線を引く
`\PointDashY[*] [stretch]{破線の長さ}{座標名}[表示文字][文字位置]` ... 定義点から y 軸へ破線を引く
`\PointDashXY[*] [stretch]{破線の長さ}{座標名}` ... 定義点から x 軸, y 軸へ破線を引く

`\PointDashX`, `\PointDashY` は定義された座標名から x 軸または y 軸へ破線を垂線を引く命令です。点の座標は `\CnodeX`, `\CnodeY` に入れ直しますので表示文字で利用することが出来ます (`\Cnode` でもよい)。また、`\KSame` を指定すると「文字名と座標軸名」を、`\CnodeName` を指定すると「文字名と座標軸名と数字」が表示されます。文字位置は座標名の定義と同じものが使えます。

`\PointDashXY` は同時に x 座標、 y 座標に破線の垂線を引き、座標の数値を自動で表示する命令です。文字位置は垂線が引かれる反対側に表示します。文字位置の細かい指定は出来ません。

命令の後に「*」をつけると、座標に子目盛りを表示します。また、表示用の計算された数値は小数第1位までの表示（定義された値はそのまま）が初期値ですが、変更したい場合は `\def\KFigure{1}` の値を変更して下さい。

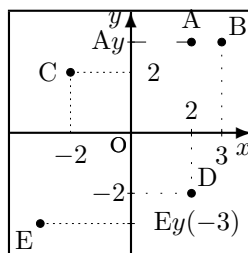
さて、破線は `eepic.sty` の `\dashline` を使用しています。この命令の `stretch` の値は省略すると `stretch=1` を初期値としました。破線の間隔は1単位当たりの実線と空白の間隔のように思えるのですが、値を変えてもよくわからない、また `stretch` もよくわからないので、次の例を参考して下さい。

●座標から軸へ破線を引く

入力

```
\unitlength=4mm%
\begin{AxesFrameC}(8,8)
  \def\KFigure{0}% 小数第0位まで表示
  \Knode*(2,3){A}[\KSame][t]
  \Knode*(3,3){B}[\KSame][rt]
  \Knode*(-2,2){C}[\KSame][l]
  \Knode*(2,-2){D}[\KSame][rt]
  \Knode*(-3,-3){E}[\KSame][lb]
  \PointDashY*{0.5}{A}[\KSame][l]
  \PointDashX*{0.1}{B}[\Cnode][b]
  \PointDashXY[100]{0.1}{C}
  \PointDashXY[10]{0.1}{D}
  \PointDashY[100]{0.1}{E}[\CnodeName][r,l]
\end{AxesFrameC}
```

出力



4.3.4 定義された2点を通る直線と軸の交点の数値を表示する ---- `\AxesCrossX`(or `\AxesCrossY`)

定義

`\AxesCrossX{2点の座標名}[文字位置]` ... x 軸との交点の数値を表示
`\AxesCrossY{2点の座標名}[文字位置]` ... y 軸との交点の数値を表示

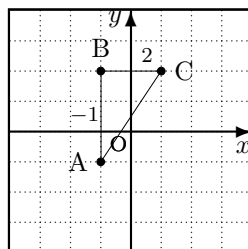
定義された2点を通る直線と x 軸, y 軸との交点の数値を表示する命令です。座標軸と交わらない場合は表示しないようにしています。線分ではなく直線で計算しているので、線分の延長線上の座標軸との交点を計算しています。

●直線と軸の交点を表示

入力

```
\unitlength=4mm%
\begin{AxesFrameDotC}(8,8)
  \def\KFigure{0}% 小数第0位まで表示
  \Knode*(-1,-1){A}[\KSame][l]
  \Knode*(-1,2){B}[\KSame][t]
  \Knode*(1,2){C}[\KSame][r]
  \KPath{AB,AC,BC}
  \footnotesize
  \AxesCrossX{AB}[lt]
  \AxesCrossY{CB}[rt]
\end{AxesFrameDotC}
```

出力



第5章 点を結ぶ

5.1 eclairth.sty の拡張

5.1.1 線の種類 ---- ¥KPen

¥KPen 命令は、¥KPath 命令で結ぶ線の種類を指定します。eclairth.sty の¥throughbrush と同じ働きというより、¥throughbrush のマクロをコピーして名前を変えただけです。理由は、スペルが長いからです。変更して元に戻したいときは、¥path と定義して下さい。

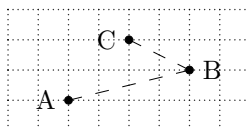
定義

```
¥KPen{線種}
```

入力

```
¥unitlength=4mm%
¥begin{PicDot}(8,4)
  ¥Knode*(2,1){A}[¥KSame][l]
  ¥Knode*(6,2){B}[¥KSame][r]
  ¥Knode*(4,3){C}[¥KSame][l]
  ¥KPen{¥dashline{0.5}}
  ¥KPath{ABC}
¥end{PicDot}
```

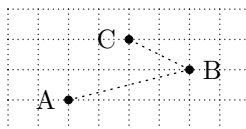
出力



入力

```
¥unitlength=4mm%
¥begin{PicDot}(8,4)
  ¥Knode*(2,1){A}[¥KSame][l]
  ¥Knode*(6,2){B}[¥KSame][r]
  ¥Knode*(4,3){C}[¥KSame][l]
  ¥KPen{¥dottedline{0.2}}
  ¥KPath{ABC}
¥end{PicDot}
```

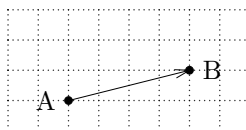
出力



入力

```
¥def¥ArrowHeadSize{0.5}%
¥unitlength=4mm%
¥begin{PicDot}(8,4)
  ¥Knode*(2,1){A}[¥KSame][l]
  ¥Knode*(6,2){B}[¥KSame][r]
  ¥KPen{¥arrow¥drawline}
  ¥KPath{AB}
¥end{PicDot}
```

出力



5.1.2 置いた点を結ぶ ---- ¥KPath

¥Knode, ¥Pnode 等の第4章で定義した命令は点の座標を記憶しています。これを結んで表示するのが¥KPath 命令です。基本的に eclairth.sty の¥through と同じ働きをしますが、カンマで区切れば複数指定可能にしました。また、¥KPen の線種を最初に指定しておくと、¥KPen の役割もします。これらの命令は、eclairth.sty の¥through のマクロを改良しています。

定義

```
¥KPath[線種]{node 座標(, node 座標, ...)}
```

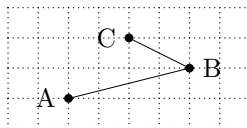
入力

```

¥unitlength=4mm%
¥begin{PicDot}(8,4)
  ¥Knode*(2,1){A}[¥KSame][l]
  ¥Knode*(6,2){B}[¥KSame][r]
  ¥Knode*(4,3){C}[¥KSame][l]
  ¥KPen{¥drawline}
  ¥KPath{ABC}
¥end{PicDot}

```

出力



5.2 2点を結ぶ

この節では、前節の続きで定義した点の内、2点を結ぶ直線、ベクトルについて解説します。この章で扱う命令は以下の表のものです。なお「¥~Arc」と「¥~DashArc」は3点を通る外接円を利用しているため、大きい座標を指定するとエラーします。「¥unitlength=0.5mm¥Knode(80,40){A}」とせずに、「¥unitlength=5mm¥Knode(8,4){A}」のようにして下さい。

基本命令	範囲を実線で示す	範囲を破線で示す
¥KLine	¥KLineArc	¥KLineDashArc
¥KVec	¥KVecArc	¥KVecDashArc
¥KLineName	¥KLineNameArc	¥KLineNameDashArc

●この節に限り「方向, 文字位置」の所に以下の命令が追加使用可能です。

left	中点で始点から左 90° の位置に表示
right	中点で始点から右 90° の位置に表示

距離は¥WordSep を使用します。

●また、同様にこの節に限り「文字」の所に以下の命令が使用可能です。

¥KSame	2点の座標名を表示する
¥KTrue	2点間の実距離を表示する

¥KTrue の小数の有効桁は¥KFigure で指定します。

5.2.1 2点を結び、名前等を入れる ---- ¥KLine

2つの座標を結び、名前を入れます。数字でもかまいません。2点の中点を基準として、「方向・文字位置」に基づいて文字等を表示します。

定義

¥KLine{始点終点の座標名}[文字][方向, 文字位置]

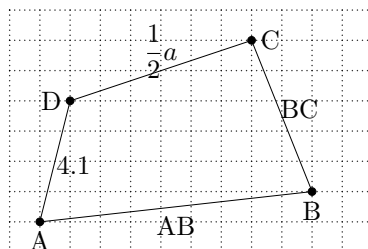
入力

```

¥unitlength=4mm%
¥begin{PicDot}(12,8)
  ¥WordSep=2.5mm%
  ¥Knode*(1,1){A}[¥KSame][b]
  ¥Knode*(10,2){B}[¥KSame][b]
  ¥Knode*(8,7){C}[¥KSame][r]
  ¥Knode*(2,5){D}[¥KSame][l]
  ¥KLine{AB}[¥KSame][b]
  ¥KLine{BC}[¥KSame][right]
  ¥KLine{CD}[$¥displaystyle¥frac{1}{2}a$][t,t1]
  ¥KLine{DA}[¥KTrue][left]
¥end{PicDot}

```

出力



5.2.2 ベクトルと名前を入れる ---- $\text{\textbackslash KVec}$

2 つの座標を結んだ線に名前を入れます。矢の長さは、 $\text{\textbackslash KVecHeadSize}$ で絶対的に定義されています。初期値は $\text{\textbackslash KVecHeadSize}=3\text{mm}$ です。 $\text{\textbackslash KArrowHeadFilltrue}$ とすると矢印が塗られます。

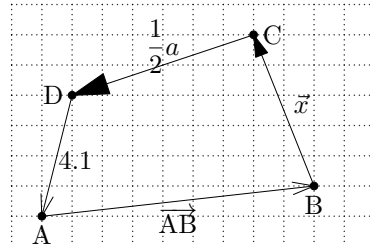
定義

$\text{\textbackslash KVec}$ {始点終点の座標名}[文字][方向, 文字位置]

入力

```
\unitlength=4mm%
\begin{PicDot}(12,8)
  \WordSep=2.5mm%
  \Knode*(1,1){A}[\KSame][b]
  \Knode*(10,2){B}[\KSame][b]
  \Knode*(8,7){C}[\KSame][r]
  \Knode*(2,5){D}[\KSame][l]
  \KVec{AB}[\KSame][b]
  \KArrowHeadFilltrue% 矢印を塗る
  \KVec{BC}[\$vec{x}\$][right]
  \KVecHeadSize=5mm% 矢印の大きさを変える
  \KVec{CD}[\$displaystyle\frac{1}{2}a\$][t,tl]
  \KArrowHeadFillfalse% 矢印を塗るのをやめる
  \KVecHeadSize=3mm% 矢印の大きさを初期値へ戻す
  \KVec{DA}[\KTrue][left]
\end{PicDot}
```

出力



5.2.3 線に名前を入れる ---- $\text{\textbackslash KLineName}$

2 つの座標の間に名前を入れます。数字でもかまいません。この命令は、線を描かない以外は、 $\text{\textbackslash KLine}$ と同じです。

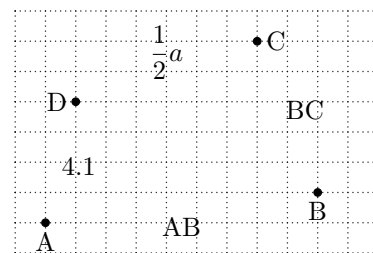
定義

$\text{\textbackslash KLineName}$ {始点終点の座標名}[文字][方向, 文字位置]

入力

```
\unitlength=4mm%
\begin{PicDot}(12,8)
  \WordSep=2.5mm%
  \Knode*(1,1){A}[\KSame][b]
  \Knode*(10,2){B}[\KSame][b]
  \Knode*(8,7){C}[\KSame][r]
  \Knode*(2,5){D}[\KSame][l]
  \KLineName{AB}[\KSame][b]
  \KLineName{BC}[\KSame][right]
  \KLineName{CD}[\$displaystyle\frac{1}{2}a\$][t,tl]
  \KLineName{DA}[\KTrue][left]
\end{PicDot}
```

出力



5.2.4 範囲を実線で示す ---- $\text{\textbackslash ~Arc}$

ここまで出てきた 3 つの命令の始点から終点にかけて、(表示) 文字を中心に範囲を実線で示します。範囲は、2 点と表示する文字の基準点の 3 点を通る円弧を描きます。文字にかからないように、文字の基準点から指定した長さは描きません。ただし、 $\text{\textbackslash KAngle}$ の中心マクロに掃き出しますので、時間がかかります。

定義

$\text{\textbackslash KLineArc}$ {始点終点の座標名}[文字][方向, 文字位置]

$\text{\textbackslash KVecArc}$ {始点終点の座標名}[文字][方向, 文字位置]

$\text{\textbackslash KLineNameArc}$ {始点終点の座標名}[文字][方向, 文字位置]

- 文字にかからない長さは以下の命令で変更可能

説明	命令	初期値
開始方向から文字の基準点まで	<code>¥KLineArcLenS</code>	4mm
文字の基準点から終了方向まで	<code>¥KLineArcLenE</code>	4mm

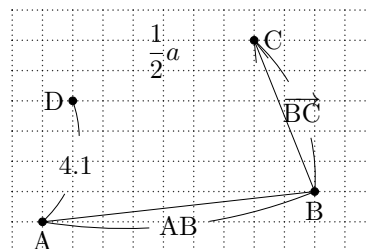
入力

```

¥unitlength=4mm%
¥begin{PicDot}(12,8)
  ¥WordSep=2.5mm%
  ¥Knode*(1,1){A}[¥KSame][b]
  ¥Knode*(10,2){B}[¥KSame][b]
  ¥Knode*(8,7){C}[¥KSame][r]
  ¥Knode*(2,5){D}[¥KSame][l]
  ¥KLineArc{AB}[¥KSame][b]
  ¥KVecArc{BC}[¥KSame][right]
  ¥KLineName{CD}[$¥displaystyle¥frac{1}{2}a$][t,t1]
  ¥KLineNameArc{DA}[¥KTrue][left]
¥end{PicDot}

```

出力



5.2.5 範囲を破線で示す ---- ¥~DashArc

こちらは、(表示) 文字を中心に範囲を破線で示します。ただし、きちんと表示しないバグがあります。こちら、文字間の空白は実線と同様、またこちらも同様に`¥KAngle`の中心マクロに引き出しますので、時間がかかります。

定義

`¥KLineDashArc{始点終点の座標名}[文字][方向, 文字位置]`
`¥KVecDashArc{始点終点の座標名}[文字][方向, 文字位置]`
`¥KLineNameDashArc{始点終点の座標名}[文字][方向, 文字位置]`

- 破線部分と空白部分の設定は以下の命令で変更可能

説明	命令	初期値
破線部分	<code>¥KDashArcLine</code>	1mm
空白部分	<code>¥KDashArcSpace</code>	0.5mm

- 文字にかからない長さは以下の命令で変更可能

説明	命令	初期値
開始方向から文字の基準点まで	<code>¥KLineArcLenS</code>	4mm
文字の基準点から終了方向まで	<code>¥KLineArcLenE</code>	4mm

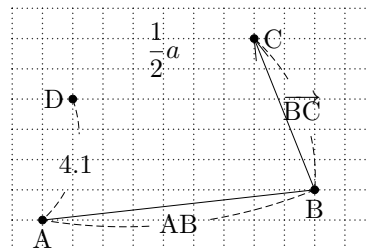
入力

```

¥unitlength=4mm%
¥begin{PicDot}(12,8)
  ¥WordSep=2.5mm%
  ¥Knode*(1,1){A}[¥KSame][b]
  ¥Knode*(10,2){B}[¥KSame][b]
  ¥Knode*(8,7){C}[¥KSame][r]
  ¥Knode*(2,5){D}[¥KSame][l]
  ¥KLineDashArc{AB}[¥KSame][b]
  ¥KVecDashArc{BC}[¥KSame][right]
  ¥KLineName{CD}[$¥displaystyle¥frac{1}{2}a$][t,t1]
  ¥KLineNameDashArc{DA}[¥KTrue][left]
¥end{PicDot}

```

出力



第6章 円弧・一般角・直角

eepic.sty の`\arc` を使用すれば円弧・角度を表示することができます。しかし、弧度法でしかも `tpic` に吐き出すので、時計方向の角を指定しなければなりません。そこで、度数法・一般角に対応した、時計の逆回りの円弧・角度を表示するための命令を定義しました。いずれのマクロも処理の最後は、`eepic.sty` の`\arc` に掃き出しています。

以下の表は、この章に出てくる基本命令です。

●実線で円弧を描く

基本形	正の方向に矢印を付ける	負の方向に矢印を付ける
<code>\KArc</code>	<code>\ArrowKArc</code>	<code>\RevArrowKArc</code>
<code>\KAngle</code>	<code>\ArrowKAngle</code>	<code>\RevArrowKAngle</code>
<code>\PAngle</code>	<code>\ArrowPAngle</code>	<code>\RevArrowPAngle</code>

●破線で円弧を描く

基本形	正の方向に矢印を付ける	負の方向に矢印を付ける
<code>\KDashArc</code>	<code>\ArrowKDashArc</code>	<code>\RevArrowKDashArc</code>
<code>\KDashAngle</code>	<code>\ArrowKDashAngle</code>	<code>\RevArrowKDashAngle</code>
<code>\PDashAngle</code>	<code>\ArrowPDashAngle</code>	<code>\RevArrowPDashAngle</code>

6.1 共通項目

まずは、この章で共通な項目について説明します。

6.1.1 文字表示の基準点

円弧・一般角と同時に表示する文字の基準点について説明します。基準点は、表示する円弧・一般角上の中央部分になるようにしてあります。次の図の黒い点はその基準点です。この基準点を元に、「方向」「文字位置」を指定して下さい。

この条件外に表示したいときは、改めて`\Knode`等で設定して下さい。

●基準点を表示する

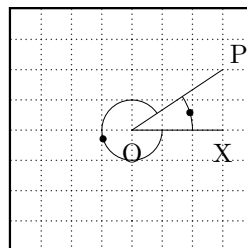
入力

```

\unitlength=4mm%
\begin{PicFrameDotC}(8,8)
  \Knode(0,0){O}[\KSAME][b]
  \Knode(3,0){X}[\KSAME][b]
  \Knode(3,2){P}[\KSAME][rt]
  \KPen{\drawline}
  \KPath{XOP}
  \KAngle[2]{POX}[\circle*{0.2}]
  \KAngle[4]{XOP}[\circle*{0.2}]
\end{PicFrameDotC}

```

出力



6.1.2 文字

「文字」の所に以下の命令が使用可能です。

<code>¥KSame</code>	座標名を表示する
<code>¥KTrue</code>	角度を度数で表示する

`¥KTrue` は座標を指定したときも、自動で角度を計算してくれます。また小数の有効桁は`¥KFigure` で指定します。

6.1.3 円弧の直径

角の直径は、「`¥AngleLen`」で指定します。初期値は `¥AngleLen=10mm` です。`¥circle` や`¥arc` と互換性を持たせる為に、直径にしたので注意して下さい。

このほか、この章の全ての定義の「直径」の所に数値を入れておくと、その角に関しての直径として処理します。ただし、単位は座標系なので相対的です。

6.2 円弧・角度の表示

6.2.1 `¥arc` の拡張 ---- `¥KArc`

定義

`¥KArc[直径](x座標,y座標)(開始角,終了角)[文字][方向,文字位置]`

`¥KArc[直径]{node 座標}(開始角,終了角)[文字][方向,文字位置]`

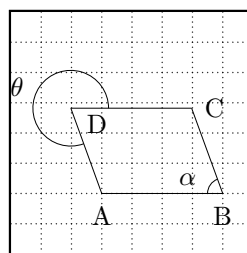
`¥KArc` は、指定した座標を中心をして一般角の開始角、終了角を指定することで、円弧を描きます。ただし、角度は degree(度) で指定します。開始角と終了角の指定範囲は、 -360° から 360° です。また、時計の反対周りに開始角と終了角がくるように指定して下さい。

● `¥KArc` の使用例

入力

```
¥unitlength=4mm%
¥begin{PicFrameDot}(8,8)(-3,-2)
  ¥Pnode(0,0){A}[¥KSame][b]
  ¥Knode(4,0){B}[¥KSame][b]
  ¥Pnode{B}(3,110){C}[¥KSame][r]
  ¥Knode{C}(-4,0){D}[¥KSame][rb,1]
  ¥KPath[¥drawline]{ABCD}
  ¥KArc[1](4,0)(110,180)[¥alpha$][l,b]
  ¥KArc{D}(0,290)[¥theta$][l]
¥end{PicFrameDot}
```

出力



6.2.2 3つの座標から角を表示 ---- `¥KAngle`

定義

`¥KAngle[直径]{3つの座標名}[文字][方向,文字位置]`

定義した3つの座標のうち、真ん中の座標を中心として角の記号を表示する命令です。初めの座標と最後の座標の間に角を表示します。時計の逆周りの順に指定しておくように。順番を変えると反対側に描かれます。

真ん中の座標に対し、他の2点の座標の一般角を計算して表示します。しかし、計算は単に2分法を使用しているだけで、「とにかく動けばいいや」で作成しましたので、少々計算時間がかかります。「将来的に早くするつもりです」と古いマニュアルでは書きましたが、パソコンが早くなったのでやめました。

● `\KAngle` の使用例

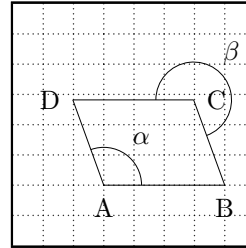
入力

```

\unitlength=4mm%
\begin{PicFrameDot}(8,8)(-3,-2)
  \Pnode(0,0){A}[\KSAME][b]
  \Knode(4,0){B}[\KSAME][b]
  \Pnode{B}(3,110){C}[\KSAME][r]
  \Knode{C}(-4,0){D}[\KSAME][l]
  \KPath{\drawline}{ABCD}
  \KAngle{BAD}[\alpha][rt]
  \KAngle{BCD}[\beta][rt]
\end{PicFrameDot}

```

出力

6.2.3 `\Pnode` を利用して角を表示 ---- `\PAngle`

定義

`\PAngle[直径]{3つの座標名}[文字][方向, 文字位置]`

`\Pnode` で定義した時、原点に対しての角度を入力している場合、ここではそれを利用して、角度を表示します。当然、`\KAngle` より高速です。ただし、原点に対して座標を入力したときのみ使用できるので、あまり使い道がないかもしれません。

定義した3つの座標のうち、真ん中の座標を中心として角の記号を表示する命令です。初めの座標と最後の座標の間に角を表示します。時計の逆周りの順に指定しておくように。順番を変えると反対側に描かれます。

● `\PAngle` の使用例

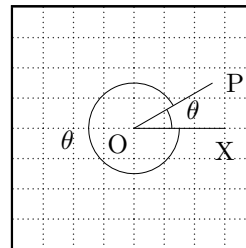
入力

```

\unitlength=4mm%
\begin{PicFrameDotC}(8,8)
  \Pnode(0,0){O}[\KSAME][bl]
  \Pnode(3,0){X}[\KSAME][b]
  \Pnode(3,30){P}[\KSAME][r]
  \KPath{\drawline}{XOP}
  \PAngle{XOP}[\theta][r,b]
  \PAngle[3]{POX}[\theta][l]
\end{PicFrameDotC}

```

出力



6.3 矢印を付ける

6.3.1 矢印の大きさ ---- `\KArrowLen`

この節に出てくる矢印の大きさは、`\KArrowLen` で設定しています。初期値は`\KArrowLen=2mm` です。小さい角度の時は、この大きさを変えて下さい。

また、初期設定で矢印は塗りつぶされます。塗られない矢印との切り替えは、「`\KArrowHeadNoFilltrue`」とします。戻すためには、「`\KArrowHeadNoFillfalse`」とします。次の例でもわかるように、余りきれいではありません。

● 塗られない矢を表示する

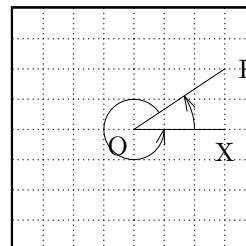
入力

```

\unitlength=4mm%
\begin{PicFrameDotC}(8,8)
  \Knode(0,0){O}[\KSAME][bl]
  \Knode(3,0){X}[\KSAME][b]
  \Knode(3,2){P}[\KSAME][r]
  \KPen{\drawline}
  \KPath{XOP}
  \KArrowHeadNoFilltrue% ここ
  \ArrowKAngle[2]{POX}
  \ArrowKAngle[4]{XOP}
  \KArrowHeadNoFillfalse% ここ
\end{PicFrameDotC}

```

出力



6.3.2 正の方向に付ける ---- ¥Arrow～

これまで出てきた3種類の角度の表示命令の前に「Arrow」を付けると一般角の正の方向に矢印が付きます。矢印をつける以外の使い方は同じなので省略します。

定義

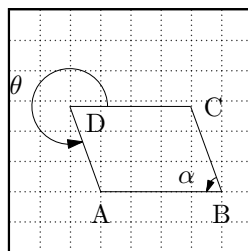
¥ArrowKArc[直径](中心)(開始角, 終了角)[文字][方向, 文字位置]
 ¥ArrowKArc[直径]{座標名}(開始角, 終了角)[文字][方向, 文字位置]
 ¥ArrowKAngle[直径]{3つの座標名}[文字][方向, 文字位置]
 ¥ArrowPAngle[直径]{3つの座標名}[文字][方向, 文字位置]

● ¥ArrowKArc の使用例

入力

```
¥unitlength=4mm%
¥begin{PicFrameDot}(8,8)(-3,-2)
  ¥Pnode(0,0){A}[¥KSame][b]
  ¥Knode(4,0){B}[¥KSame][b]
  ¥Pnode{B}(3,110){C}[¥KSame][r]
  ¥Knode{C}(-4,0){D}[¥KSame][rb,1]
  ¥KPath[¥drawline]{ABCD}
  {¥KArrowLen=1.5mm
  ¥ArrowKArc[1](4,0)(110,180)
    [¥alpha$][1,b]}
  ¥ArrowKArc{D}(0,290)[¥theta$][1]
¥end{PicFrameDot}
```

出力

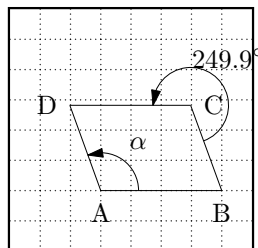


● ¥ArrowKAngle の使用例

入力

```
¥unitlength=4mm%
¥begin{PicFrameDot}(8,8)(-3,-2)
  ¥Pnode(0,0){A}[¥KSame][b]
  ¥Knode(4,0){B}[¥KSame][b]
  ¥Pnode{B}(3,110){C}[¥KSame][r]
  ¥Knode{C}(-4,0){D}[¥KSame][l]
  ¥KPath[¥drawline]{ABCD}
  ¥ArrowKAngle{BAD}[¥alpha$][rt]
  ¥ArrowKAngle{BCD}[¥KTrue][rt]
¥end{PicFrameDot}
```

出力

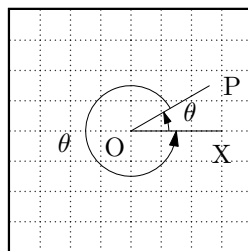


● ¥ArrowPAngle の使用例

入力

```
¥unitlength=4mm%
¥begin{PicFrameDotC}(8,8)
  ¥Pnode(0,0){O}[¥KSame][bl]
  ¥Pnode(3,0){X}[¥KSame][b]
  ¥Pnode(3,30){P}[¥KSame][r]
  ¥KPath[¥drawline]{XOP}
  {¥KArrowLen=1.5mm
  ¥ArrowPAngle{XOP}[¥theta$][r,b]}
  ¥ArrowPAngle{3}{POX}[¥theta$][l]
¥end{PicFrameDotC}
```

出力



6.3.3 負の方向に付ける ---- ¥RevArrow～

これまで出てきた3種類の角度の表示命令の前に「RevArrow」を付けると一般角の負の方向に矢印が付きます。矢印をつける以外の使い方は同じなので省略します。

定義

¥RevArrowKArc[直径](中心)(開始角, 終了角)[文字][方向, 文字位置]
 ¥RevArrowKArc[直径]{座標名}(開始角, 終了角)[文字][方向, 文字位置]
 ¥RevArrowKAngle[直径]{3つの座標名}[文字][方向, 文字位置]
 ¥RevArrowPAngle[直径]{3つの座標名}[文字][方向, 文字位置]

● `\RevArrowKArc` の使用例

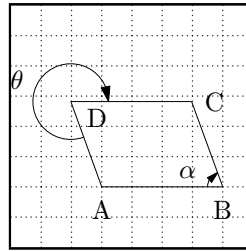
入力

```

\unitlength=4mm%
\begin{PicFrameDot}(8,8)(-3,-2)
  \Pnode(0,0){A}{\KSame}[b]
  \Knode(4,0){B}{\KSame}[b]
  \Pnode{B}(3,110){C}{\KSame}[r]
  \Knode{C}(-4,0){D}{\KSame}[rb,1]
  \KPath[\drawline]{ABCD}
  {\KArrowLen=1.5mm
  \RevArrowKArc[1](4,0)(110,180)
    [\$alpha$][l,b]}
  \RevArrowKArc{D}(0,290)[\$theta$][1]
\end{PicFrameDot}

```

出力

● `\RevArrowKAngle` の使用例

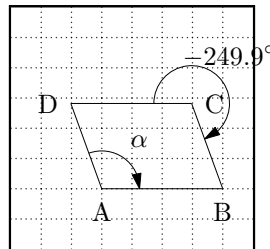
入力

```

\unitlength=4mm%
\begin{PicFrameDot}(8,8)(-3,-2)
  \Pnode(0,0){A}{\KSame}[b]
  \Knode(4,0){B}{\KSame}[b]
  \Pnode{B}(3,110){C}{\KSame}[r]
  \Knode{C}(-4,0){D}{\KSame}[l]
  \KPath[\drawline]{ABCD}
  \RevArrowKAngle{BAD}[$alpha$][rt]
  \RevArrowKAngle{BCD}{\KTrue}[rt]
\end{PicFrameDot}

```

出力

● `\RevArrowPAngle` の使用例

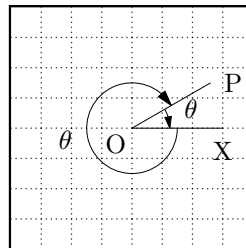
入力

```

\unitlength=4mm%
\begin{PicFrameDotC}(8,8)
  \Pnode(0,0){O}{\KSame}[bl]
  \Pnode(3,0){X}{\KSame}[b]
  \Pnode(3,30){P}{\KSame}[r]
  \KPath[\drawline]{XOP}
  {\KArrowLen=1.5mm
  \RevArrowPAngle{XOP}[$theta$][r,b]}
  \RevArrowPAngle[3]{POX}[$theta$][1]
\end{PicFrameDotC}

```

出力



6.4 破線で円弧・一般角を描く

6.4.1 破線のパターン

ここで説明する破線は以下の補助命令の数値で設定されています。

補助命令	説明	初期値
<code>\KDashArcLine</code>	破線の表示部分の長さ	1mm
<code>\KDashArcSpace</code>	破線の空白部分の長さ	0.5mm

6.4.2 破線で描く ---- `\KDashArc`, `\KDashAngle`, `\PDashAngle`

円弧・一般角を破線で描きます。それ以外は前と同じことなので、定義と簡単な例だけ載せます。

定義

```

\KDashArc[直径](中心)(開始角, 終了角)[文字][方向, 文字位置]
\KDashArc[直径]{座標名}(開始角, 終了角)[文字][方向, 文字位置]
\KDashAngle[直径]{3つの座標名}[文字][方向, 文字位置]
\PDashAngle[直径]{3つの座標名}[文字][方向, 文字位置]

```

● `\KDashArc` の使用例

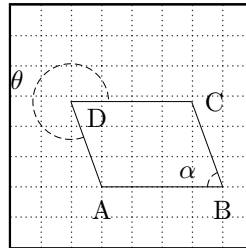
入力

```

\unitlength=4mm%
\begin{PicFrameDot}(8,8)(-3,-2)
  \Pnode(0,0){A}{\KSame}[b]
  \Knode(4,0){B}{\KSame}[b]
  \Pnode{B}(3,110){C}{\KSame}[r]
  \Knode{C}(-4,0){D}{\KSame}[rb,1]
  \KPath{\drawline}{ABCD}
  {\KArrowLen=1.5mm
  \KDashArc[1](4,0)(110,180)
    [1,1]{\alpha}}
  \KDashArc{D}(0,290)[1]{\theta}[1]
\end{PicFrameDot}

```

出力

6.4.3 矢印を正の方向へ付ける ---- `\Arrow`～

これも破線になる以外これまで出てきたものと同じなので、定義と簡単な例だけ載せます。

定義

```

\ArrowKDashArc[直径](中心)(開始角, 終了角)[文字][方向, 文字位置]
\ArrowKDashArc[直径]{座標名}(開始角, 終了角)[文字][方向, 文字位置]
\ArrowKDashAngle[直径]{3つの座標名}[文字][方向, 文字位置]
\ArrowPDashAngle[直径]{3つの座標名}[文字][方向, 文字位置]

```

● `\ArrowKDashAngle` の使用例

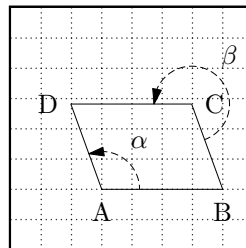
入力

```

\unitlength=4mm%
\begin{PicFrameDot}(8,8)(-3,-2)
  \Pnode(0,0){A}{\KSame}[b]
  \Knode(4,0){B}{\KSame}[b]
  \Pnode{B}(3,110){C}{\KSame}[r]
  \Knode{C}(-4,0){D}{\KSame}[l]
  \KPath{\drawline}{ABCD}
  \ArrowKDashAngle{BAD}[1]{\alpha}[rt]
  \ArrowKDashAngle{BCD}[1]{\beta}[rt]
\end{PicFrameDot}

```

出力

6.4.4 矢印を負の方向に付ける ---- `\RevArrow`～

これも破線になる以外これまで出てきたものと同じなので、定義と簡単な例だけ載せます。

定義

```

\RevArrowKDashArc[直径](中心)(開始角, 終了角)[文字][方向, 文字位置]
\RevArrowKDashArc[直径]{座標名}(開始角, 終了角)[文字][方向, 文字位置]
\RevArrowKDashAngle[直径]{3つの座標名}[文字][方向, 文字位置]
\RevArrowPDashAngle[直径]{3つの座標名}[文字][方向, 文字位置]

```

● `\RevArrowPDashAngle` の使用例

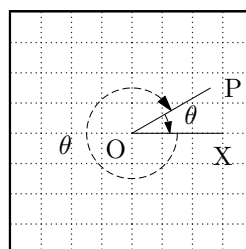
入力

```

\unitlength=4mm%
\begin{PicFrameDotC}(8,8)
  \Pnode(0,0){O}{\KSame}[bl]
  \Pnode(3,0){X}{\KSame}[b]
  \Pnode(3,30){P}{\KSame}[r]
  \KPath{\drawline}{XOP}
  {\KArrowLen=1.5mm
  \RevArrowPDashAngle{XOP}[1]{\theta}[r,b]}
  \RevArrowPDashAngle{3}{POX}[1]{\theta}[1]
\end{PicFrameDotC}

```

出力



6.5 応用例

6.5.1 一般角で $2\pi(360^\circ)$ 以上を描く

360° を超える一般角を表示します。考え方は、半円を描いたら、中心をプラスの方向へ移動し、半径をその分増やす方法です。この方法だと $0^\circ \leq \theta \leq 180^\circ$ の間に動径がある場合は、正確に出来ませんが、 $180^\circ < \theta < 360^\circ$ の場合は、中心が違うため角度を試行錯誤しなければなりません。

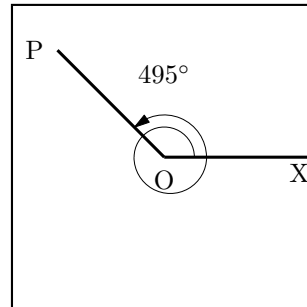
次の例は、始め $0^\circ \leq \theta \leq 180^\circ$ は中心 (0, 0), 半径 1 の半円を描き、次に $180^\circ \leq \theta \leq 360^\circ$ は中心 (0.2, 0), 半径 1.2 の半円を、最後に $360^\circ \leq \theta \leq 495^\circ$ は中心 (0, 0), 半径 1.4 の半円を描いています。

● 360° を越える角度・上側

入力

```
%unitlength=4mm
%begin{PicFrameC}(10,10)
  %Pnode(0,0){O}[O][b]
  %Pnode(5,0){X}[X][b1]
  %Pnode(5,135){P}[P][l]
  %Thicklines
  %KPath{OX}
  %KPath{OP}
  %thinlines
  %def%KFigure{0}
  %KArc[2](0,0)(0,180)
  %KArc[2.4](0.2,0)(180,360)
  %ArrowKArc[2.8](0,0)(0,135)
  %Pnode(2,90){Q}[$495^\circ\circlearrowleft$][t]
%end{PicFrameC}
```

出力



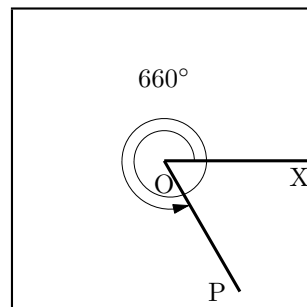
次の例は、始め $0^\circ \leq \theta \leq 180^\circ$ は中心 (0, 0), 半径 1 の半円を描き、次に $180^\circ \leq \theta \leq 360^\circ$ は中心 (0.2, 0), 半径 1.2 の半円を、 $360^\circ \leq \theta \leq 540^\circ$ は中心 (0, 0), 半径 1.4 の半円を、 $180^\circ \leq \theta \leq 292^\circ$ は中心 (0.2, 0), 半径 1.6 の半円を描いています。300° まで行くと行き過ぎるので、調整しています。

● 360° を越える角度・下側

入力

```
%unitlength=4mm
%begin{PicFrameC}(10,10)
  %Pnode(0,0){O}[O][b]
  %Pnode(5,0){X}[X][b1]
  %Pnode(5,300){P}[P][l]
  %Thicklines
  %KPath{OX}
  %KPath{OP}
  %thinlines
  %def%KFigure{0}
  %KArc[2](0,0)(0,180)
  %KArc[2.4](0.2,0)(180,360)
  %KArc[2.8](0,0)(0,180)
  %ArrowKArc[3.2](0.2,0)(180,292)
  %Pnode(2,90){Q}[$660^\circ\circlearrowright$][t]
%end{PicFrameC}
```

出力



負の角度も同様のことが言えます。

6.6 直角の記号表示 ---- %KNinty

2本の線の交点に直角の記号を表示します。空間座標や斜交座標に直角の記号を表示する事もあるので、直角でなくても記号としてです。ようは、一辺が%KNintyLenの平行四辺形として考えているだけです。

定義

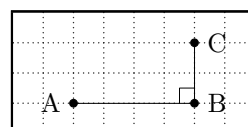
KNinty{3つの座標名}

直角の大きさは、「%KNintyLen」で指定します。初期値は%KNintyLen=2mmです。

KNinty{ABC}もKNinty{CBA}同じ場所に表示します。

入力

出力



```

%unitlength=4mm%
%begin{PicFrameDot}(8,4)
  %Knode*(2,1){A}[%KSame][l]
  %Knode*(6,1){B}[%KSame][r]
  %Knode*(6,3){C}[%KSame][r]
  %KPen{%drawline}
  %KPath{ABC}
  %KNinty{ABC}
%end{PicFrameDot}

```

第7章 領域を斜線表示する

7.1 はじめに

領域の斜線を表示します。初めは少ない命令で実行できることについて説明し、後半はこの命令ができるためのマクロについて説明しています。

やっていることは、長方形または円の内部を斜線で表示したのち、`eclairth.sty` の`\whiten` を使用して指定した場所の斜線を消します。画面上は線が消えるのですが、印刷するとうっすらと線が出るため、`color.sty` を使用して白い線を引いて消しています。画面上では周りの線が残る時がありますが、印刷では消えているはずです。

- (要 `color`) の表示がある命令はところは `color.sty` を使用するため、その命令を使用するときは必ず `\usepackage[dvout]{color}` をプリアンプル部分に入れて下さい。
- 斜線の間隔は`\KDomainInterval` で絶対値で定義しています。初期値は`\KDomainInterval=0.7071mm` ($= \frac{1}{\sqrt{2}}\text{mm}$) です。傾きを変えてもこの値の幅で描きます。
- [斜線の傾き] は省略可能です。0 や負の値も使用可能です。省略すると「1」が入ります。
- [原点位置] は[原点 0 の表示位置] のことです。省略すると「lb」が入ります。
- [斜線の傾き][原点位置] は同時に省略可能ですが、[斜線の傾き] のみは省略できません。
- 「2 点の座標名」は環境の範囲の 2 点の `node` 座標を指定して下さい。
- 2 点の座標名は、22 ページの`\KLineBothEdge` で定義すると簡単だと思います。
- 線上の点から座標軸までの破線と数値は 28 ページの`\PointDashX(or Y or XY)` を使用すると簡単です。
- 座標軸との交点は 2 点の座標名を使って 28 ページの`\AxesCorssX(or Y)` を使用すると簡単です。
- `\KMaskingtrue` を記述すると `\KMaskingfalse` が出てくるまで `node` 座標を定義するときの文字の後ろを白くします。文字の大きさを調べてその範囲だけ白く塗りますが、もう少し余白を取りたい場合`\KMaskingPadding` で指定して下さい。初期値は`\KMaskingPadding=0pt` です。「dvout」の画面上では文字の回りが欠ける場合がありますが、印刷では平気のはずです。
- 例として使用する `node` 座標は後で使用する 22 ページの`\KnodeCorner` や 22 ページの`\KnodeLB(or RB or RT or LT)` で A~D を使用したいと思い、P から始めています。
- `\WordSep=2mm` `\def\KFigure{0}` `\footnotesize` を多様しています。見栄えを良くするためですが、`\def\KFigure{0}` は小数第 1 位を四捨五入しますので、実際の値が小数以下もある場合は注意が必要です。

7.2 領域を 1 つの命令で表示する

高校の数学で教える (学ぶ?) 「不等式の表す領域」の表示を 1 つのマクロで表示できる命令です。この節の定義で、[*] は□ 無しの * を指定することで、作成する直線や円を太く (`\thicklines`) 描きます。また、マクロの途中で座標軸を描きますので、`Pic` 環境や `PicC` 環境でかまいません。

各命令の前に「Not」を入れると境界を含まない領域が表示できます。初期値は`\KDomainLineBetween=1.75mm` (約 5pt) です。これは、線に指定した幅の白い線を描いているだけなので斜線の傾きによっては領域の端では線が消えません (個人的には「境界線を含む」「境界線は含まない」を下に書けば良いと思うのだが)。

7.2.1 直線を境界とする領域の表示 ---- ¥[Not]LineUpper(or Lower)Domain

環境内での上端の点と下端の点を 2 点の座標名とする直線の¥LineUpperDomain 命令は上側に斜線の領域を、¥LineLowerDomain 命令は下側に斜線の領域を図示します。

定義 (要 color)

¥[Not]LineUpperDomain[*] [斜線の傾き] [原点位置] {2 点の座標名}

¥[Not]LineLowerDomain[*] [斜線の傾き] [原点位置] {2 点の座標名}

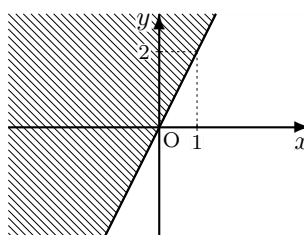
次の場合、原点を通るので原点以外の 直線が通る点を座標軸に示さなければなりません。そこで点 R(1,2) を指定して xy 軸に数字を表示します。 y 軸の座標が領域の中に入って見にくくなるので、¥KMaskingtrue(or false) を使用しています。

● $y \geq 2x$ の表す領域

入力

```
¥unitlength=5mm%
¥begin{PicC}(8,6)
  ¥KLineBothEdge{2,0}{P,Q}
  ¥LineUpperDomain*[-1]{PQ}
  ¥WordSep=2mm%
  ¥def¥KFigure{0}
  ¥footnotesize
  ¥Knode(1,2){R}
  ¥KMaskingtrue
  ¥PointDashXY*[100]{0.1}{R}
  ¥KMaskingfalse
¥end{PicC}
```

出力



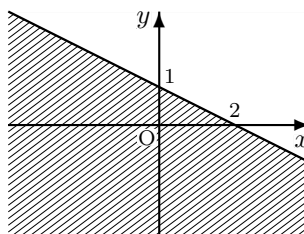
次の場合、定義した点 P と点 Q を¥AxesCrossX と¥AxesCrossY で再利用してます。¥KLineBothEdge の値を変えるだけで違う領域を示してくれます。また、斜線の傾きを 0.75 にしています。

● $y \leq -\frac{1}{2}x + 1$ の表す領域

入力

```
¥unitlength=5mm%
¥begin{PicC}(8,6)
  ¥KLineBothEdge{-0.5,1}{P,Q}
  ¥LineLowerDomain*[0.75]{PQ}
  ¥WordSep=2mm
  ¥def¥KFigure{0}
  ¥footnotesize
  ¥AxesCrossY{PQ}[rt]
  ¥AxesCrossX{PQ}[t]
¥end{PicC}
```

出力



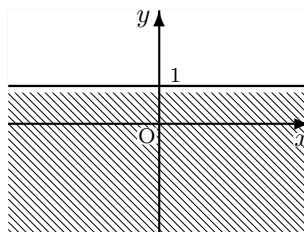
x 軸と平行な直線の表す領域も簡単に描けます。次は、斜線の傾きを -1 にして境界付近を消しています。

● $y < 1$ の表す領域

入力

```
¥unitlength=5mm%
¥begin{PicC}(8,6)
  ¥KLineBothEdge{0,1}{P,Q}
  ¥NotLineLowerDomain*[-1]{PQ}
  ¥WordSep=2mm
  ¥def¥KFigure{0}
  ¥footnotesize
  ¥AxesCrossY{PQ}[rt,1]
¥end{PicC}
```

出力



例えば、 $x \geq 1$ のような y 軸と平行領域を図示する場合、傾きがないためエラーするので作成方法は 50 ページで記載してあります。

7.2.2 円を境界とする領域の表示 ---- $\text{\texttt{\$[Not]CircleInside(or Outside)Domain}}$

環境内で中心と直径を指定することで $\text{\texttt{\$CircleInsideDomain}}$ 命令は円の内側に斜線の領域を、 $\text{\texttt{\$CircleOutsideDomain}}$ 命令は円の外側に斜線の領域を図示します。

定義 (要 color)

$\text{\texttt{\$[Not]CircleInsideDomain[*]}}$ [斜線の傾き] [原点位置] (中心の座標){直径}
 $\text{\texttt{\$[Not]CircleInsideDomain[*]}}$ [斜線の傾き] [原点位置]{中心の node 座標}{直径}
 $\text{\texttt{\$[Not]CircleOutsideDomain[*]}}$ [斜線の傾き] [原点位置] (中心の座標){直径}
 $\text{\texttt{\$[Not]CircleOutsideDomain[*]}}$ [斜線の傾き] [原点位置]{中心の node 座標}{直径}

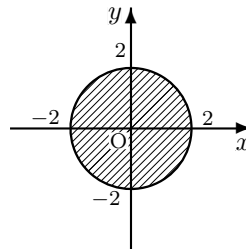
中心には xy 座標の他に node 座標も指定できます。ただし、node 座標で「*」で中心を表示しようと思っても $\text{\texttt{\$CircleOutsideDomain}}$ では円の内側を白く塗るため消されてしまいますので、必要な場合はもう一度定義する必要があります。また、円と座標軸の交点を求める命令は作成していないので手動でおこなう必要があります。

● $x^2 + y^2 \leq 4$ の表す領域

入力

```
\unitlength=4mm%
\begin{Pic}(8,8)
  \CircleInsideDomain*(0,0){4}
  \WordSep=2mm
  \def\KFigure{0}
  \footnotesize
  \Knode(2,0){P}{\CnodeX}[rt,l]
  \Knode(-2,0){P}{\CnodeX}[lt,r]
  \Knode(0,2){P}{\CnodeY}[lt,b]
  \Knode(0,-2){P}{\CnodeY}[lb,r]
\end{Pic}
```

出力

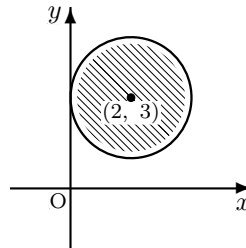


● $(x-2)^2 + (y-3)^2 < 4$ の表す領域

入力

```
\unitlength=4mm%
\begin{Pic}(8,8)(-2,-2)
  \Knode*(2,3){0}
  \NotCircleInsideDomain*[-1]{0}{4}
  \WordSep=2mm
  \def\KFigure{0}
  \footnotesize
  \KMaskingtrue
  \Knode(2,3){0}{\Cnode}[b]% 再定義
  \KMaskingfalse
  \Knode*(2,3){0}% 再再定義
\end{Pic}
```

出力

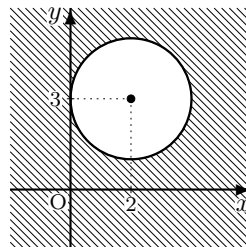


● $(x-2)^2 + (y-3)^2 \geq 4$ の表す領域

入力

```
\unitlength=4mm%
\begin{Pic}(8,8)(-2,-2)
  \Knode*(2,3){0}
  \CircleOutsideDomain*[-1]{0}{4}
  \WordSep=2mm
  \def\KFigure{0}
  \footnotesize
  \Knode*(2,3){0}% 再定義
  \KMaskingtrue
  \PointDashXY*[100]{0.1}{0}
  \KMaskingfalse
\end{Pic}
```

出力



7.2.3 連立不等式の表す領域の表示 1 ----- ¥[Not]LineUpper(or Lower)Upper(or Lower)Domain

環境内で連立1次不等式の表す範囲を表示します。1つめのUpper or Lowerは1つめの{2点の座標名}、2つめのUpper or Lowerは2つめの{2点の座標名}に対応しています。

定義 (要 color)

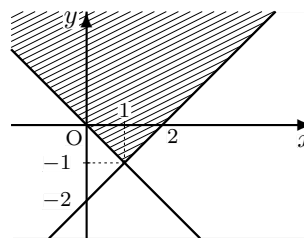
¥[Not]LineUpperUpperDomain[*] [斜線の傾き] [原点位置] {2点の座標名}{2点の座標名}
 ¥[Not]LineUpperLowerDomain[*] [斜線の傾き] [原点位置] {2点の座標名}{2点の座標名}
 ¥[Not]LineLowerUpperDomain[*] [斜線の傾き] [原点位置] {2点の座標名}{2点の座標名}
 ¥[Not]LineLowerLowerDomain[*] [斜線の傾き] [原点位置] {2点の座標名}{2点の座標名}

● $\begin{cases} y \geq x - 2 \\ y \geq -x \end{cases}$ の表す領域

入力

```
¥unitlength=5mm%
¥begin{Pic}(8,6)(-2,-3)
  ¥KLineBothEdge{1,-2}{P,Q}
  ¥KLineBothEdge{-1,0}{R,S}
  ¥LineUpperUpperDomain*[0.5]{PQ}{RS}
  ¥WordSep=2mm
  ¥def¥KFigure{0}
  ¥footnotesize
  ¥KMaskingtrue
  ¥Intersection{PQ}{RS}{T}
  ¥PointDashX*[100]{0.1}{T}[¥CnodeX][t]
  ¥PointDashY*[100]{0.1}{T}[¥CnodeY][l,r]
  ¥AxesCrossX{PQ}[rb]
  ¥AxesCrossY{PQ}[l,r]
  ¥KMaskingfalse
¥end{Pic}
```

出力

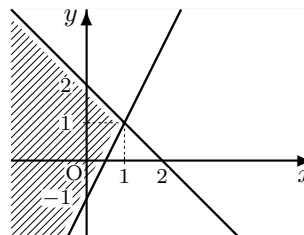


● $\begin{cases} y > 2x - 1 \\ y < -x + 2 \end{cases}$ の表す領域

入力

```
¥unitlength=5mm%
¥begin{Pic}(8,6)(-2,-2)
  ¥KLineBothEdge{2,-1}{P,Q}
  ¥KLineBothEdge{-1,2}{R,S}
  ¥NotLineUpperLowerDomain*[PQ]{RS}
  ¥WordSep=2mm
  ¥def¥KFigure{0}
  ¥footnotesize
  ¥KMaskingtrue
  ¥Intersection{PQ}{RS}{T}
  ¥PointDashX*[100]{0.1}{T}[¥CnodeX][b]
  ¥PointDashY*[100]{0.1}{T}[¥CnodeY][l,r]
  ¥AxesCrossY{PQ}[l,r]
  ¥AxesCrossX{RS}[b]
  ¥AxesCrossY{RS}[l,r]
  ¥KMaskingfalse
¥end{Pic}
```

出力



$$\bullet \begin{cases} y \leq 2x - 1 \\ y \geq -x + 2 \end{cases} \text{ の表す領域}$$

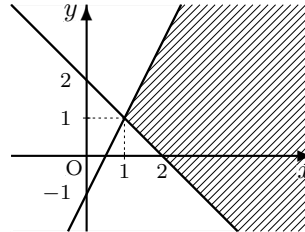
入力

出力

```

\unitlength=5mm%
\begin{Pic}(8,6)(-2,-2)
  \KLineBothEdge{2,-1}{P,Q}
  \KLineBothEdge{-1,2}{R,S}
  \LineLowerUpperDomain*{PQ}{RS}
  \WordSep=2mm
  \def\KFigure{0}
  \footnotesize
  \Intersection{PQ}{RS}{T}
  \PointDashX*[100]{0.1}{T}{\CnodeX}[b]
  \PointDashY*[100]{0.1}{T}{\CnodeY}[l,r]
  \AxesCrossY{PQ}[l,r]
  \AxesCrossX{RS}[b]
  \AxesCrossY{RS}[l,r]
\end{Pic}

```



$$\bullet \begin{cases} y < 2x + 1 \\ y < 1 \end{cases} \text{ の表す領域}$$

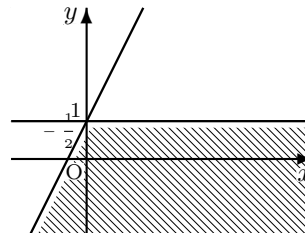
入力

出力

```

\unitlength=5mm%
\begin{Pic}(8,6)(-2,-2)
  \KLineBothEdge{2,1}{P,Q}
  \KLineBothEdge{0,1}{R,S}
  \NotLineLowerLowerDomain*[-1]{PQ}{RS}
  \WordSep=2mm
  \def\KFigure{0}
  \footnotesize
  \AxesCrossY{RS}[lt]
  \Knode(-0.5,0){T}{\tiny}
  \displaystyle{-\frac{1}{2}}[t1,b]
\end{Pic}

```



7.2.4 連立不等式の表す領域の表示2 ---- \[Not]CircleInside(or Outside)LineUpper(or Lower)Domain

環境内で円を境界とする不等式と1次不等式の表す境界との共通部分の領域を斜線で図示します。

定義 (要 color)

```

\[Not]CircleInsideLineUpperDomain[*][斜線の傾き][原点位置](中心の座標){直径}{2点の座標名}
\[Not]CircleInsideLineUpperDomain[*][斜線の傾き][原点位置]{中心の node 座標}{直径}{2点の座標名}
\[Not]CircleInsideLineLowerDomain[*][斜線の傾き][原点位置](中心の座標){直径}{2点の座標名}
\[Not]CircleInsideLineLowerDomain[*][斜線の傾き][原点位置]{中心の node 座標}{直径}{2点の座標名}
\[Not]CircleOutsideLineUpperDomain[*][斜線の傾き][原点位置](中心の座標){直径}{2点の座標名}
\[Not]CircleOutsideLineUpperDomain[*][斜線の傾き][原点位置]{中心の node 座標}{直径}{2点の座標名}
\[Not]CircleOutsideLineLowerDomain[*][斜線の傾き][原点位置](中心の座標){直径}{2点の座標名}
\[Not]CircleOutsideLineLowerDomain[*][斜線の傾き][原点位置]{中心の node 座標}{直径}{2点の座標名}

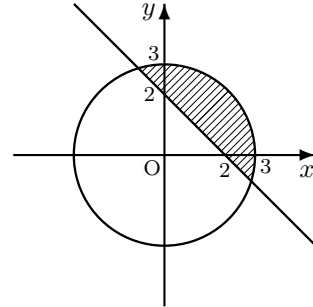
```

● $\begin{cases} x^2 + y^2 \leq 9 \\ y \geq -x + 2 \end{cases}$ の表す領域

入力

```
%unitlength=4mm%
%begin{PicC}(10,10)
  %KLineBothEdge{-1,2}{P,Q}
  %CircleInsideLineUpperDomain*(0,0){6}{PQ}
  %WordSep=2mm
  %def%KFigure{0}
  %footnotesize
  %AxesCrossX{PQ}[b]
  %AxesCrossY{PQ}[l]
  %Knode(3,0){R}{%CnodeX}[br]
  %Knode(0,3){R}{%CnodeY}[lt]
%end{PicC}
```

出力

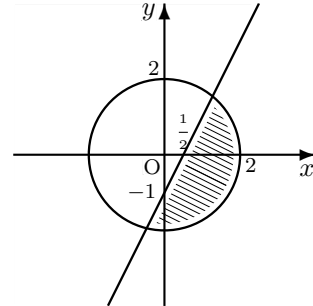


● $\begin{cases} x^2 + y^2 < 4 \\ y < 2x - 1 \end{cases}$ の表す領域

入力

```
%unitlength=5mm%
%begin{PicC}(8,8)
  %KLineBothEdge{2,-1}{P,Q}
  %Knode(0,0){O}
  %NotCircleInsideLineLowerDomain*[-0.5]{0}{4}{PQ}
  %def%KFigure{0}
  %footnotesize
  %AxesCrossY{PQ}[l]
  %Knode(0.5,0){T}{%tiny
  %displaystyle}{%frac{1}{2}}[t]
  %WordSep=2mm
  %Knode(2,0){R}{%CnodeX}[br]
  %Knode(0,2){R}{%CnodeY}[lt]
%end{PicC}
```

出力

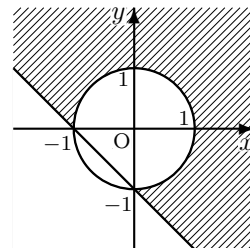


● $\begin{cases} x^2 + y^2 \geq 1 \\ y \geq -x - 1 \end{cases}$ の表す領域

入力

```
%unitlength=8mm%
%begin{PicC}(4,4)
  %KLineBothEdge{-1,-1}{P,Q}
  %Knode(0,0){O}
  %CircleOutsideLineUpperDomain*{0}{2}{PQ}
  %def%KFigure{0}
  %footnotesize
  %AxesCrossX{PQ}[lb]
  %AxesCrossY{PQ}[lb]
  %WordSep=2mm
  %Knode(1,0){R}{%CnodeX}[t1]
  %Knode(0,1){R}{%CnodeY}[b1]
%end{PicC}
```

出力

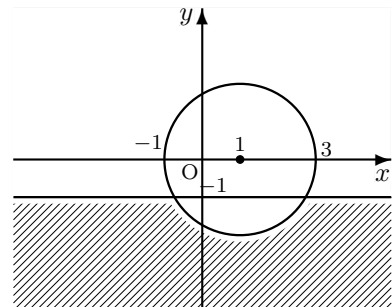


● $\begin{cases} (x-1)^2 + y^2 > 4 \\ y < -1 \end{cases}$ の表す領域

入力

```
%unitlength=5mm%
%begin{PicC}(10,8)
  %KLineBothEdge{0,-1}{P,Q}
  %Knode(1,0){O}
  %NotCircleOutsideLineLowerDomain*{0}{4}{PQ}
  %def%KFigure{0}
  %footnotesize
  %Knode(-1,0){R}{%CnodeX}[t1]
  %WordSep=2mm
  %Knode*(1,0){O}{%CnodeX}[t]% 再定義
  %Knode(3,0){R}{%CnodeX}[tr]
  %AxesCrossY{PQ}[rt]
%end{PicC}
```

出力



7.3 領域の斜線を表示

ここでは、長方形と円の領域を表示するために必要な命令について記述します。この節の定義で、[*] は [] なしの「*」を指定することで、図形の回りに線を描きます。線の太さは斜線と同じです。

7.3.1 長方形に斜線を図示 ---- ¥Khatch

定義

¥Khatch[*] [斜線の傾き] (左下座標) (右上座標)

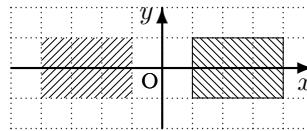
¥Khatch[*] [斜線の傾き] {2 点の node 座標}

¥Khatch は 2 点を対角とする長方形に斜線を描きます。 xy 座標を指定する場合は、(左下座標)(右上座標)の順に、node 座標を指定する場合は順番は関係なく描きます。

入力

```
¥unitlength=4mm%
¥begin{AxesDotC}(10,4)
  ¥Khatch(-4,-1)(-1,1)
  ¥Knode(1,-1){A}
  ¥Knode(4,1){B}
  ¥Khatch*[-1]{BA}
¥end{AxesDotC}
```

出力



7.3.2 環境全体に斜線を図示 ---- ¥KhatchAll

定義

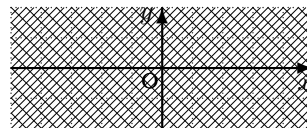
¥KhatchAll[*] [斜線の傾き]

¥KhatchAll は環境全体に斜線を描きます。

入力

```
¥unitlength=4mm%
¥begin{AxesDotC}(10,4)
  ¥KhatchAll
  ¥KDomainInterval=1.4142mm% 間隔の変更
  ¥KhatchAll[-1]
¥end{AxesDotC}
```

出力



7.3.3 円の内部の領域表示を図示 ---- ¥Khatchcircle

定義

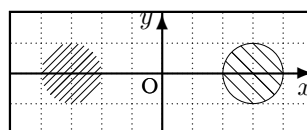
¥Khatchcircle[*] [斜線の傾き] {直径}

指定した円内に斜線を描きます。

入力

```
¥unitlength=4mm
¥begin{AxesFrameDotC}(10,4)
  ¥put(-3,0){¥Khatchcircle[1]{2}}
  ¥KDomainInterval=1.4142mm% 間隔の変更
  ¥Knode(3,0){A}
  ¥Kput{A}{¥Khatchcircle*[-1]{2}}
¥end{AxesFrameDotC}
```

出力



7.4 領域を図示するために必要な命令

ここでは、領域を図示するために必要な `eepic.sty`、`color.sty` で定義された命令について説明します。最後に `kpics.sty` で複数の命令を1つにまとめた命令を作成しています。

7.4.1 指定した閉曲線内を白で塗りつぶす ---- `%whiten(eepic.sty マクロ)`

定義

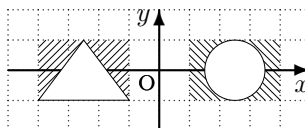
```
%whiten%path( or %circle or %ellipse or %arc)
```

`%whiten` は `eepic.sty` で定義された命令で、指定した閉曲線内を白で塗りつぶします。定義で記載したとおり `%whiten` の後に使用できるのは、`%path` (直線を結んで作られる閉曲線)、`%circle` (円、最後に「*」はつけられない)、`%ellipse` (楕円、これも最後に「*」はつけられない)、`%arc` (円弧?、使い方がわからない) とマニュアルに書いてありました。白く塗るのいいのですが、周りの線も描かれることが難点です。

入力

```
%unitlength=4mm%
%begin{AxesDotC}(10,4)
%Khatch(-4,-1)(-1,1)
%Khatch[-1](1,-1)(4,1)
%put(0,0){%whiten%path(-2.5,1)
(-4,-1)(-1,-1)(-2.5,1)}
%put(2.5,0){%whiten%circle{2}}
%end{AxesDotC}
```

出力



7.4.2 線の幅を指定する ---- `%allinethickness(eclairth.sty マクロ)`

定義

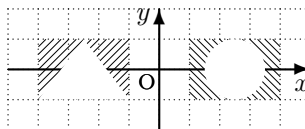
```
%allinethickness{太さ}
```

`%allinethickness` も `eepic.sty` で定義された命令で、線の太さを指定できます。そこで、`%allinethickness{0mm}` として幅のない線を描くことができます。`%thinlines` で線幅を戻すことができます。

入力

```
%unitlength=4mm%
%begin{AxesDotC}(10,4)
%Khatch(-4,-1)(-1,1)
%Khatch[-1](1,-1)(4,1)
%allinethickness{0mm}
%put(0,0){%whiten%path(-2.5,1)
(-4,-1)(-1,-1)(-2.5,1)}
%put(2.5,0){%whiten%circle{2}}
%thinlines
%end{AxesDotC}
```

出力



うまい方法だと思いました、画面上では線は消えています。でも `dviout` で印刷してみると、うっすらと線が残ってしまいます¹。これでは印刷に使えません。

¹実は昔の `dviout` では線が完璧に消えていたのですが、あるバージョンから線が印刷させるようになってしまいました。

7.4.3 色付きで線を引く ---- `\color{color.sty}` マクロ

そこで、白い線を上から書けば良いのではないかと思います、`color.sty` を導入しました。このマニュアルにも入っています。`\usepackage[dviout]{color}` と初めに記述しました。そのため \LaTeX 209 での使用は出来なくなりました。

定義 (要 color)

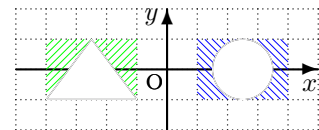
`\color{色}`

`\color` で使用できる色は、デフォルトでは「black, white, red, green, blue, cyan, magenta, yellow」だけらしいです。ここでは、内部を白くしたあと、同じ境界を白い線で描いています。斜線の色も例として変えてみました²。

入力

```
\unitlength=4mm%
\begin{AxesDotC}(10,4)
  \put(0,0){\color{green}{\Khatch(-4,-1)(-1,1)}}
  \put(0,0){\color{blue}{\Khatch[-1](1,-1)(4,1)}}
  \put(0,0){\color{black}{\whiten\path(-2.5,1)
    (-4,-1)(-1,-1)(-2.5,1)}}
  \put(2.5,0){\color{black}{\whiten\circle{2}}}
  \put(0,0){\color{white}{\path(-2.5,1)
    (-4,-1)(-1,-1)(-2.5,1)}}
  \put(2.5,0){\color{white}{\circle{2}}}
\end{AxesDotC}
```

出力



7.4.4 座標名で指定した多角形の内部と線を消す ---- `\AreaClear`

定義 (要 color)

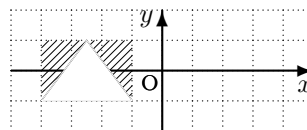
`\AreaClear{座標名... }`

ここまでまとめとして、`kpic.sty` では座標名で指定した多角形の内部と線を消す命令を作成しました。でも、4行ほどの命令を1つにただけです。ここではあまり意味を持たないかもしれませんが、7.5 で有効に働きます。

入力

```
\unitlength=4mm%
\begin{AxesDotC}(10,4)
  \Khatch(-4,-1)(-1,1)
  \Knode(-2.5,1){A}
  \Knode(-4,-1){B}
  \Knode(-1,-1){C}
  \AreaClear{ABCA}
\end{AxesDotC}
```

出力



7.5 領域を図示する

ここまでの説明を元に、7.1 で作成できない領域の図示をしてみたいと思います。ただし、条件によっては出来ないものもあることをあらかじめ言っておきます。

²カラーでないプリンターの印刷ではわかりません

7.5.1 領域を表示する流れ

1. 全体に斜線表示 or 円の内部に斜線表示

`%KhatchrAll` で環境全体に斜線を表示します。円の内部の場合だけ `%Khatchcircle` で斜線を表示します。

2. いらない部分を消す

- 斜線表示したくない部分はほとんど環境の境界の部分を含みます。そこで 4 章の `%KnodeCorner` で四隅を一括で、または `%KnodeLB`, `%KnodeLB`, `%KnodeRB`, `%KnodeRT` を使用して必要な角を定義し、`%KLineLeftEdge`, `%KLineRightEdge`, `%KLineBothEdge` で直線の境界での点を定義し、さらに 2 つの直線の交点であれば、`%Intersection` で定義して最後は `%AreaClear` で消します。 `kpic.sty` を使用しているので点の定義が自動で出来ます。

3. 境界を含まない場合

- 境界を含まない場合を表示したい場合は、`%color{white}%allinethickness{太さ}` を指定して白い太い線を描きます。`%KDomainLineBetween` を利用すると第 7.1 章の幅で描けます。

4. 座標軸を描く

- 11 ページの `%KAxesAuto` を使用して座標軸を描きます。斜線にかかるなど必要であれば原点 O を表示する位置を移動してください。

5. 必要な線を描く

- `%KPath` や `%circle` を使用して、必要な線を描きます。必要であれば、`%thicklines` で線を太くします。その後 `%thinlines` で元に戻して下さい。

6. 軸に必要な数字の表示

- 最後に座標軸に必要な数字を入れます。必要であれば `%Knode` を使用して、`%PointDashX`, `%PointDashY`, `%PointDashXY` を使用してください。この後の例では見栄えをよくするため、文字表示の距離で `%WordSep` を、文字の大きさに `%footnotesize` を使用しています。また、`node` で計算された整数の座標は小数点以下を含む場合があるので、`%def%Figure{0}` を使用して、小数点以下を消すようにして下さい。

状況によっては、順番を入れ替えます。

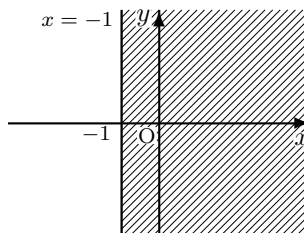
7.5.2 応用例

● $x \geq -1$

入力

```
%unitlength=5mm%
%begin{PicC}(8,6)
  %Knode(-1,-3){P}
  %Knode(-1,3){Q}
  %KnodeCorner{A,B,C,D}
  %Khatch{PC}
  %KAxesAuto
  %thicklines
  %KPath{PQ}
  %thinlines
  %WordSep=2mm
  %def%KFigure{0}
  %footnotesize
  %AxesCrossX{PQ}[lb,r]
  %Knode(-1.5,3){R}[$x = -1$][rb,r]
%end{PicC}
```

出力



$$\bullet \begin{cases} x > 0 \\ y > 0 \end{cases} \quad (\text{第1象限})$$

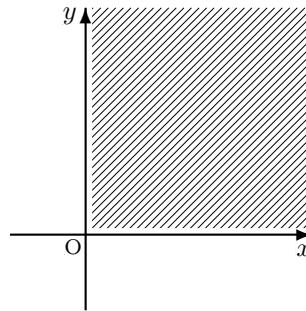
入力

```

%unitlength=5mm%
%begin{Pic}(8,8)(-2,-2)
  %Knode(0,0){O}
  %Knode(6,0){P}
  %Knode(0,6){Q}
  %KnodeRT{C}
  %put(0,0){%Khatch{OC}}
  %color{white}
  %allinethickness{%KDomainLineBetween}
  %KPath{OP,OQ}
  %thinline
  %color{black}
  %KAxesAuto
%end{Pic}

```

出力



$$\bullet \begin{cases} y \leq -x + 3 \\ y \geq -x - 1 \end{cases}$$

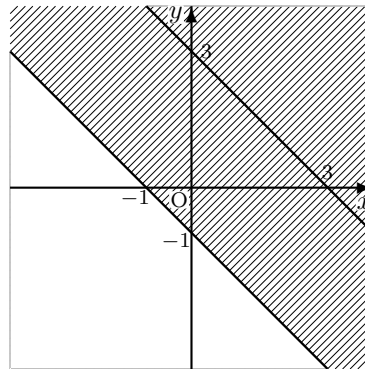
入力

```

%unitlength=6mm%
%begin{PicC}(8,8)
  %KhatchAll
  %KnodeCorner{A,B,C,D}
  %KLineBothEdge{-1,3}{P,Q}
  %KLineBothEdge{-1,-1}{R,S}
  %AreaClear{SRAS,PQCP}
  %thicklines
  %KPath{PQ,RS}
  %thinline
  %KAxesAuto
  %WordSep=2mm
  %def%KFigure{0}
  %footnotesize
  %AxesCrossX{PQ}[t]
  %AxesCrossY{PQ}[r]
  %AxesCrossX{RS}[b1]
  %AxesCrossY{RS}[1,t]
%end{PicC}

```

出力



$$\bullet \begin{cases} y \leq -3x + 6 \\ y \geq -x + 4 \end{cases} \quad (x \geq 0, y \geq 0) \quad (\text{線形計画法})$$

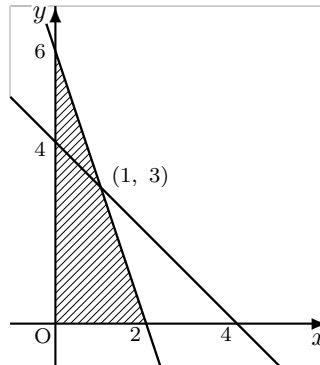
入力

```

%unitlength=6mm%
%begin{Pic}(7,8)(-1,-1)
  %Knode(0,0){P}
  %KnodeCorner{A,B,C,D}
  %Khatch{PC}
  %KLineBothEdge{-3,6}{P,Q}
  %KLineBothEdge{-1,4}{R,S}
  %AreaClear{PQBCP,RBCCR}
  %thicklines
  %KPath{PQ,RS}
  %thinline
  %KAxesAuto
  %WordSep=2mm
  %def%KFigure{0}
  %footnotesize
  %AxesCrossX{PQ}[b1]
  %AxesCrossY{PQ}[1]
  %AxesCrossX{RS}[b1]
  %AxesCrossY{RS}[1,t]
  %Intersection{PQ}{RS}{T}{%Cnode}[rt,1]
%end{Pic}

```

出力



$$\bullet \begin{cases} x^2 + y^2 \leq 16 \\ y \geq x - 2 \\ y \leq -x + 2 \end{cases}$$

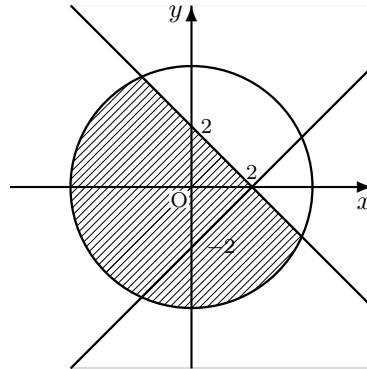
入力

```

%unitlength=4mm%
\begin{PicC}(12,12)
  \put(0,0){\Khatchcircle{8}}
  \KLineBothEdge{-1,2}{P,Q}
  \KLineBothEdge{1,-2}{R,S}
  \KnodeCorner{A,B,C,D}
  \AreaClear{PQCP,RSBR}
  \thicklines
  \KPath{PQ,RS}
  \put(0,0){\circle{8}}
  \thinlines
  \KAxesAuto
  \WordSep=2mm
  \def\KFigure{0}
  \footnotesize
  \AxesCrossX{PQ}[t]
  \AxesCrossY{PQ}[r]
  \AxesCrossY{RS}[r,l]
\end{PicC}

```

出力



$$\bullet \begin{cases} x^2 + y^2 < 36 \\ x^2 + y^2 > 4 \end{cases}$$

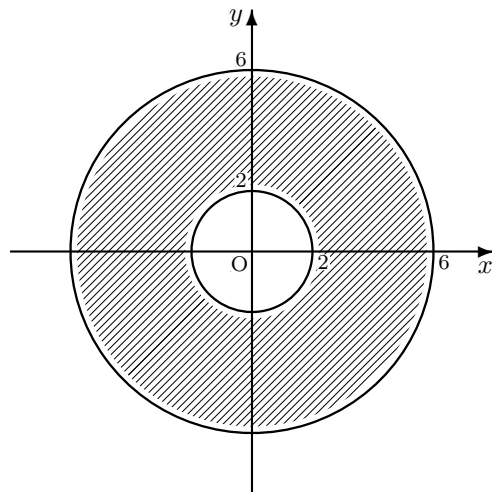
入力

```

%unitlength=4mm%
\begin{PicC}(16,16)
  \put(0,0){\Khatchcircle*{12}}
  \put(0,0){\whiten\circle{4}}
  \color{white}
  \allinethickness{\KDomainLineBetween}
  \put(0,0){\circle{12}}
  \put(0,0){\circle{4}}
  \color{black}
  \thicklines
  \put(0,0){\circle{12}}
  \put(0,0){\circle{4}}
  \thinlines
  \KAxesAuto
  \WordSep=2mm
  \def\KFigure{0}
  \footnotesize
  \KMaskingtrue
  \Knode(6,0){C}[\CnodeX][br]
  \Knode(2,0){C}[\CnodeX][br]
  \Knode(0,6){C}[\CnodeY][lt]
  \Knode(0,2){C}[\CnodeY][lt]
  \KMaskingfalse
\end{PicC}

```

出力



$$\bullet (3x - y + 5)(x - 2y + 5) \leq 0$$

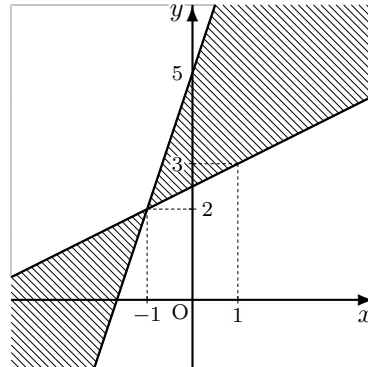
入力

```

%unitlength=6mm%
%begin{Pic}(8,8)(-4,-1.5)
%KhatchAll[-1]
%KnodeCorner{A,B,C,D}
%KLineBothEdge{3,5}{P,Q}
%KLineBothEdge{0.5,2.5}{R,S}
%AreaClear{DRSBPQD}
%KAxesAuto
%Intersection{PQ}{RS}{T}
%thicklines
%KPath{PQ,RS}
%thinlines
%WordSep=2mm
%def%KFigure{0}
%footnotesize
%PointDashXY*[100]{0.1}{T}
%AxesCrossY{PQ}[1]
%Knode(1,3){T}
%KMaskingtrue
%PointDashXY*[100]{0.1}{T}
%KMaskingfalse
%end{Pic}

```

出力



$$\bullet A \cup B$$

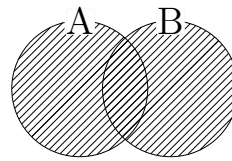
入力

```

%unitlength=3mm%
%begin{PicC}(12,8)
%put(-2,0){%Khatchcircle*{6}}
%put(2,0){%Khatchcircle*{6}}
%KMaskingtrue
%Knode(-2,3){A}{%Large A}
%Knode(2,3){B}{%Large B}
%KMaskingfalse
%end{PicC}

```

出力



$$\bullet A \cap \overline{B}$$

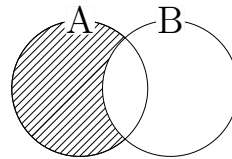
入力

```

%unitlength=3mm%
%begin{PicC}(12,8)
%put(-2,0){%Khatchcircle*{6}}
%put(2,0){%whiten%circle{6}}
%put(-2,0){%circle{6}}% 再表示
%KMaskingtrue
%Knode(-2,3){A}{%Large A}
%Knode(2,3){B}{%Large B}
%KMaskingfalse
%end{PicC}

```

出力



$$\bullet \overline{A \cup B}$$

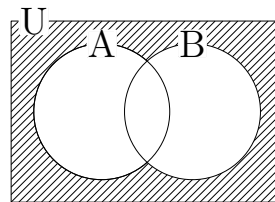
入力

```

%unitlength=3mm%
%begin{PicC}(12,8)
%KhatchAll*
%put(-2,0){%whiten%circle{6}}
%put(2,0){%whiten%circle{6}}
%put(-2,0){%circle{6}}% 再表示
%KMaskingtrue
%Knode(-5,4){A}{%Large U}
%Knode(-2,3){A}{%Large A}
%Knode(2,3){B}{%Large B}
%KMaskingfalse
%end{PicC}

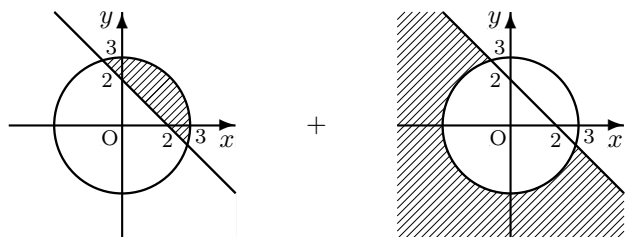
```

出力

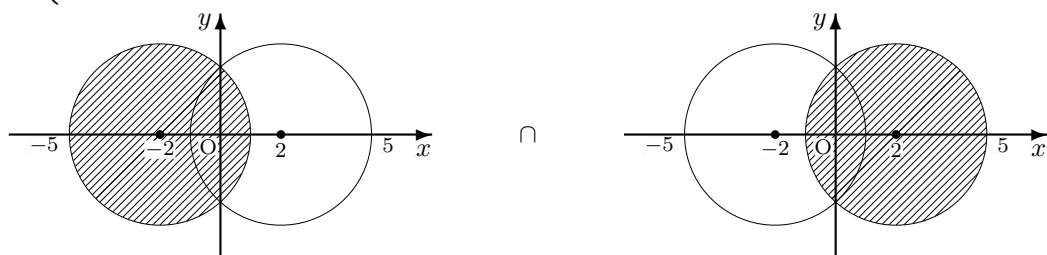


7.5.3 現状できないこと

- $(x^2 + y^2 - 9)(x + y - 2) \leq 0$ (直線と円の方方程式の積の連立が表す領域)



- $\begin{cases} (x-2)^2 + y^2 < 9 \\ (x+2)^2 + y^2 < 9 \end{cases}$ (2つの円の共通部分)



参考文献

- [1] 磯崎秀樹「 \LaTeX 自由自在」(サイエンス社、1992)
- [2] 奥村晴彦「 $\text{\LaTeX}2\text{e}$ 美文書作成入門」(技術評論社、1997)
- [3] 乙部厳己+江口庄英「 $\text{p}\text{\LaTeX}2_{\varepsilon}$ for Windows Another Manual Vol.1 Basic Kit」(ソフトバンク、1996)
- [4] 乙部厳己+江口庄英「 $\text{p}\text{\LaTeX}2_{\varepsilon}$ for Windows Another Manual Vol.2 Extended Kit」(ソフトバンク、1997)
- [5] 藤田眞作「 \LaTeX マクロの八衢」(アジソン・ウェスレイ・パブリッシャーズ・ジャパン、1995)
- [6] Michel Goossens, Frank Mittelbach, Alexander Samarin 共著 アスキー書籍編集部訳 監修「The \LaTeX コンパニオン」(アスキー、1998)