picture 環境支援マクロ集

emathP.sty ver.0.66

emathPb.sty ver.0.01

emathPg.sty ver.0.00

emathPh.sty ver.2.45

emathPk.sty ver.0.94

emathPxy.sty ver.0.31

emathT.sty ver.0.42

使用例

tDB

2005/11/02

概 要

中学・高校で数学のプリントにつける図の作成に必要な記号，コマンド，環境を集めたマクロ集です.

このマクロ集のマクロについてのご質問，バグ報告，修正・追加の提案等は

http://emath.s40.xrea.com/

の掲示板へどうぞ。

目 次

0　emath パッケージにおけるグラフィックス

emath パッケージでは，図を描画するのに二つの手法を用意しています。

(1) `tpic specials` を用いる方法

emathPh.sty, emathPk.sty, emathP.sty では，`tpic specials` を用いて，図を描画します。

(2) `Post Script` を用いる方法

emathPs.sty では，`Post Script` を用いて，図を描画します。

いずれにせよ，`¥special` コマンドを用いてグラフィックスを扱うのは，OS・dvi-ware 依存となります。従って，emathP?.sty 等のロードに先立って

graphicx.sty

を，グラフィックスドライバを指定してロードしておく必要があります。例えば

```
¥usepackage[dvips]{graphicx}
¥usepackage[dviout]{graphicx}
¥usepackage[dvipdfm]{graphicx}
¥usepackage[dvipdfmx]{graphicx}
```

カラーを用いる場合は，

color.sty

を，graphicx.sty と同一のドライバを指定してロードしておく必要があります。

```
¥usepackage[dvips]{graphicx,color}
¥usepackage[dviout]{graphicx,color}
¥usepackage[dvipdfm]{graphicx,color}
¥usepackage[dvipdfmx]{graphicx,color}
```

注 1. TEX 環境がデフォルトのグラフィックスドライバを指定している場合は，単に

```
¥usepackage{graphicx,color}
```

とすれば，デフォルトのドライバが用いられます。

注 2. dvipdfmx は，EPS において，座標が負の部分を無条件にカットする，などの振る舞いがありますので，emath ではお勧めいたしかねます。emath パッケージでは，`dvips` を標準としています。

また dvipdfmx は図形の回転処理が独特ですから，どうしても dvipdfmx を使用したい場合は

```
¥usepackage[dvipdfmx]{graphicx,color}
```

としておく必要があります。こうしても，EPS ファイルの座標が負の部分をカットすることは回避できません。

なお，`dvipdfmx` には，カラーの tpic が扱えないバージョンがあります。

注 3. dvipdfm は，EPS において，座標が負の部分をカットする点は dvipdfmx と同様です。

注 4. dviout を用いたとき，白く塗りつぶしたはずが黒くなる，という場合の対策です。

(a) dviout の画面で黒塗りとなる場合は，グラフィックスドライバが不適切です。

```
¥usepackage[dvips]{graphicx,color}
```

と，グラフィックスドライバを明示するのがよいでしょう（ emath などの読み込みに先立って）。

【補足】 dviout のメニューバーから

```
Option
  Setup Parameters
    Graphic
```

とたどって，Ghostscript の項が

On(default)

になっていることも必要です。

(b) dviout の画面では正常だが，印刷すると黒塗りになるという場合は，dviout の設定の問題です。

具体的な手順です：

dviout のメニューバーから

```
Option
  Setup Parameters
    Graphic
```

とたどり，その下方

```
color specials
```

の左にチェックが入っていることを確認し，
右のドロップダウンリストで

```
auto mode(rep)
replace(def)
replace(bak)
```

のいずれかを選び，
その右の `Save` **ボタン**
さらに下の `OK` **ボタン**
を押します。

1 zahyou 環境

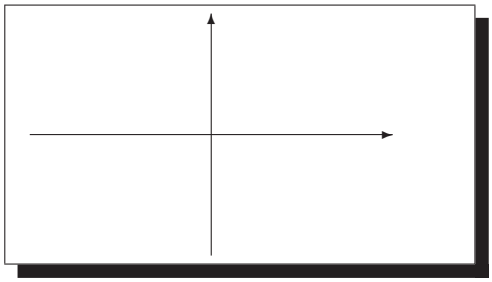
1.1 なぜ zahyou 環境か

LaTeX で座標平面を描画するには，picture 環境があります。ただし，負の数を扱うには，picture 環境の引数を計算しなければなりません。例えば，$-3 < x < 3$, $-2 < y < 2$ の範囲を描画するには

picture 環境

```
¥unitlength=8mm
¥begin{picture}(6,4)(-3,-2)%
¥put(-3,0){¥vector(1,0){6}}%
¥put(0,-2){¥vector(0,1){4}}%
¥end{picture}%
```

→



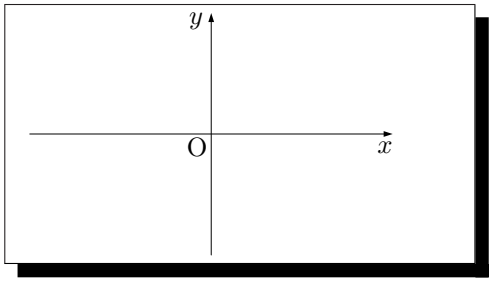
引数の与え方が面倒であることと，座標軸の描画もまとめて面倒を見てしまおう，ということで zahyou 環境です。

1.2 zahyou 環境

zahyou 環境

```
¥unitlength=8mm
¥begin{zahyou}(-3,3)(-2,2)%
¥end{zahyou}%
```

→



picture 環境とは，引数の与え方が異なり，

```
(xmin, xmax)(ymin, ymax)
```

と，x, y の区間を与えます。

なお，この環境内では，次の変数が定義されています。

¥xmin x **の区間の最小値**
¥xmax x **の区間の最大値**
¥ymin y **の区間の最小値**
¥ymax y **の区間の最大値**

¥O **原点**
¥XMAX x **軸の右端の点**
¥XMIN x **軸の左端の点**
¥YMAX y **軸の上端の点**
¥YMIN y **軸の下端の点**
¥RT **描画領域右上のコーナー**
¥RB **描画領域右下のコーナー**
¥LT **描画領域左上のコーナー**

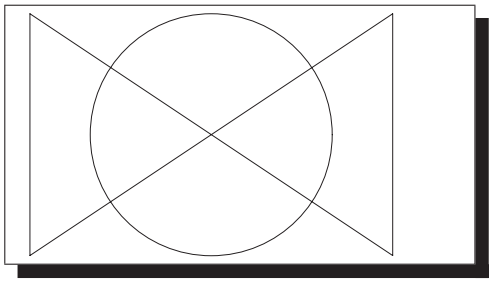
¥LB　**描画領域左下のコーナー**

座標軸を描画する必要がない場合に対しては zahyou*環境を用意してあります。

zahyou*環境

```
¥unitlength=8mm
¥begin{zahyou*}(-3,3)(-2,2)%
  ¥Drawline{¥LB¥RT¥RB¥LT¥LB}
  ¥En¥O{2}%
¥end{zahyou*}%
```

→



上記リスト中，¥Drawline は，指定点を折れ線で結びます。¥En¥O{2}は点 ¥O（原点）を中心とする半径 2 の円を描画しています。詳細は後述します。

zahyou 環境には，細かい微調整をするためのオプションがありますが，これについては後述します。

2　直線図形

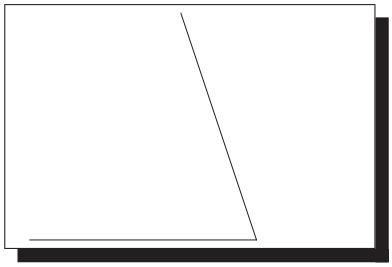
2.1　折れ線

折れ線を描画するのに，epic.sty で定義されている ¥drawline を用いることができます.

¥drawline

```
¥begin{picture}(3,3)%
¥drawline(0,0)(3,0)(2,3)%
¥end{picture}
```

⟶

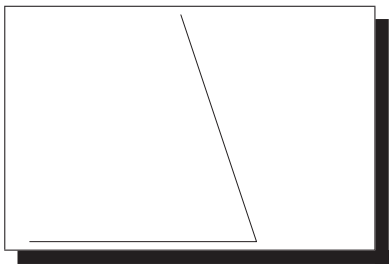


このスタイルファイルでは，点を ¥def¥A{(3,0)} などと変数で表すことにしています．その場合は，¥drawline は使えません．代わりに ¥Drawline を用意しました．このコマンドは内部で ¥drawline を呼び出していますから，epic.sty を必要とします.

¥Drawline

```
¥begin{picture}(3,3)%
¥def¥O{(0,0)}%
¥def¥A{(3,0)}%
¥def¥B{(2,3)}%
¥Drawline{¥O¥A¥B}%
¥end{picture}
```

⟶



¥Drawline の書式は

¥Drawline#1
#1 : **点列**

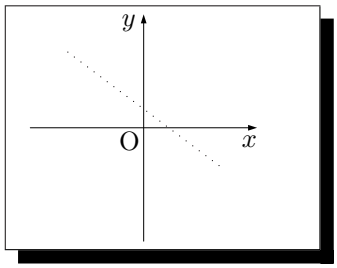
2.2 点線

点線を描画するには ¥emdottedline を用います。

¥emdottedline

```
¥begin{zahyou}[ul=5mm](-3,3)(-3,3)
  ¥emdottedline(-2,2)(2,-1)
¥end{zahyou}
```

→



2.2.1 ¥emDottedline

emath では，点列を一つにまとめて扱いますので，¥emDottedline を定義しました。

ただし，このコマンドは multido.sty で定義されている ¥multido を用います。¥emDottedline コマンドを使用するには，プリアンブルで

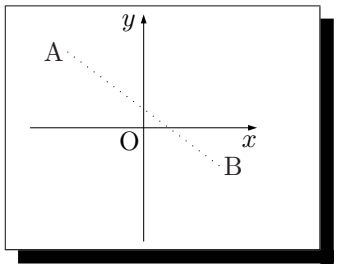
```
¥usepackage{multido}
```

を宣言しておく必要があります。

¥emDottedline

```
¥begin{zahyou}[ul=5mm](-3,3)(-3,3)
  ¥tenretu{A(-2,2)w;B(2,-1)e}
  ¥emDottedline{¥A¥B}
¥end{zahyou}
```

→



点と点の間隔は，デフォルトは 3pt としてありますが，これを変更するには<G=..>オプションを用います。

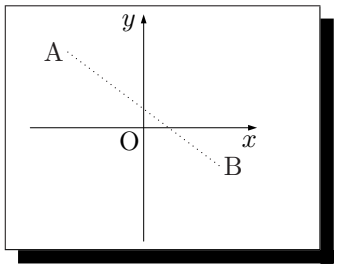
2.2.2 <G=..>オプション

さて，その<G=..>オプションを用いて，もう少し点を稠密にして見ましょう。

<G=..>オプション

```
¥begin{zahyou}[ul=5mm](-3,3)(-3,3)
  ¥tenretu{A(-2,2)w;B(2,-1)e}
  ¥emDottedline<G=2pt>{¥A¥B}
¥end{zahyou}
```

→



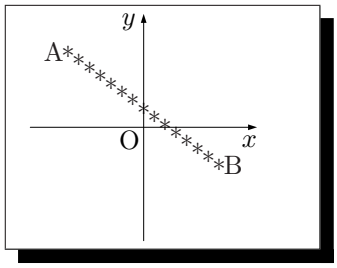
2.2.3 <C=..>オプション

配置する点の形状を変更するには，<C=..>オプションを用います。

<C=..>オプション

```
¥begin{zahyou}[ul=5mm](-3,3)(-3,3)
  ¥tenretu{A(-2,2)w;B(2,-1)e}
  ¥emDottedline<C=$*$,G=5pt>{¥A¥B}
¥end{zahyou}
```

→

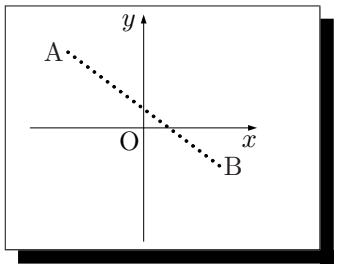


小さい円（黒塗りの）にしますか。

小さい円を配置

```
¥begin{zahyou}[ul=5mm](-3,3)(-3,3)
  ¥ukansan{1pt}¥tyokkei
  ¥tenretu{A(-2,2)w;B(2,-1)e}
  ¥emDottedline<C=¥circle*¥tyokkei>{¥A¥B}
¥end{zahyou}
```

→



ここで登場しているコマンド¥ukansan#1#2は，#1に与えられた，単位付の長さを，¥unitlengthを単位とする無名数に換算した結果を#2の制御綴に与えるものです。

2.2.4 応用例

奥村先生の掲示板 Q&A に「行列の点々」という投稿がありました。emath を用いる一案です。

$$\begin{bmatrix} a_1 & b_1+d_1 & c_1 & & & \\ & a_2 & b_2+d_2 & c_2 & & \\ & & a_3 & b_3+d_3 & c_3 & \\ & & & & & \\ & & & & & \\ & & & a_{n-1} & b_{n-1}+d_{n-1} & c_{n-1} \\ & & & & a_n & b_n+d_n & c_n \end{bmatrix}$$

2.3 破線

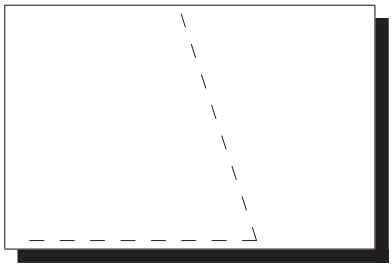
2.3.1 ¥Dashline

破線を引くコマンドは ¥Dashline です.

¥Dashline

```
¥begin{picture}(3,3)%
¥def¥O{(0,0)}%
¥def¥A{(3,0)}%
¥def¥B{(2,3)}%
¥Dashline{0.2}{¥O¥A¥B}%
¥end{picture}
```

→



このコマンドは内部で eepic.sty の ¥dashline を呼び出しています．書式もほぼ同様で

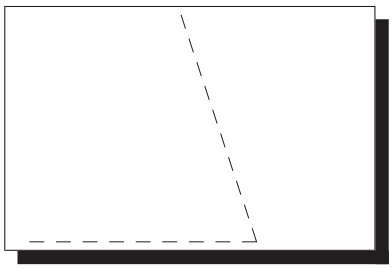
```
¥Dashline[#1]#2#3
    #1 : stretch
    #2 : dashlength
    #3 : 点列
```

です．[dashdotgap] はサポートしていません．[stretch] で代用してください．

¥Dashline[stretch]

```
¥begin{picture}(3,3)%
¥def¥O{(0,0)}%
¥def¥A{(3,0)}%
¥def¥B{(2,3)}%
¥Dashline[25]{0.2}{¥O¥A¥B}%
¥end{picture}
```

→



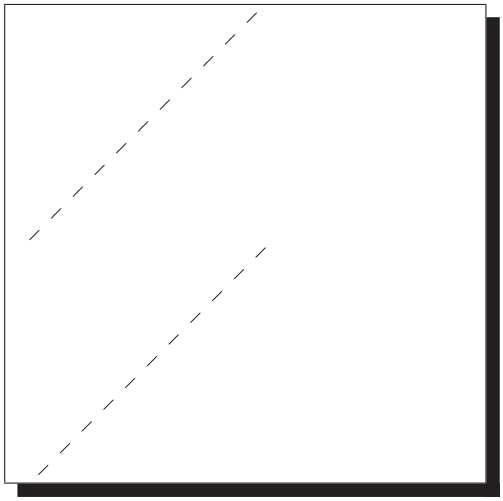
2.3.2 ¥hasen

¥dashline の第 1 引数は ¥unitlength の値によって変えなければなりません。

¥Dashline と ¥unitlength

```
¥unitlength=10mm
¥begin{picture}(3,3)
¥dashline{.2}(0,0)(3,3)
¥end{picture}¥¥
¥unitlength=1mm
¥def¥B{(30,30)}
¥begin{picture}(30,30)
¥dashline{2}(0,0)(30,30)
¥end{picture}
```

→

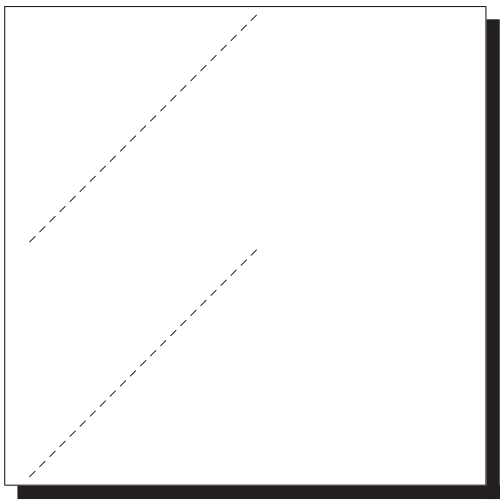


これは面倒ですから，¥hasen を作りました。

¥hasen

```
¥unitlength=10mm
¥begin{picture}(3,3)
¥hasen(0,0)(3,3)
¥end{picture}¥¥
¥unitlength=1mm
¥begin{picture}(30,30)
¥hasen(0,0)(30,30)
¥end{picture}
```

→



また，¥dashline における第 1 引数は不要とし，¥hasen の形状はオプション引数で指定する方式を取りました。¥hasen の書式です。

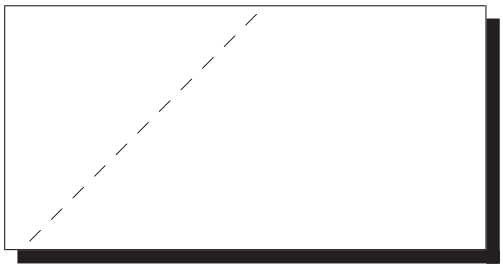
```
¥hasen[#1](x1,y1)(x2,y2).....(xN,yN)
    #1 : L=（破線の実線部分の長さ） デフォルト値=1mm
       : G=（破線のギャップの長さ） デフォルト値=0.9mm
    (x1,y1)...(xN,yN) : 折れ線の頂点列
```

オプション引数を与えて破線の形状を変更する例です。

¥hasen の形状変更

```
¥unitlength=10mm
¥begin{picture}(3,3)
¥hasen[L=2mm,G=2mm](0,0)(3,3)
¥end{picture}
```

→



ただし，ギャップの長さは必ずしも指定した長さにはなりません。というのは，破線の両端における実線部分が指定した長さとなるようにギャップを調整しているからです。

細かいことですが，¥drawline で描画した直線と ¥dashline で描画した破線の位置がずれることがあります。これを修正するのも ¥hasen を開発した目的の一つです。

2.3.3 ¥Hasen

¥drawline, ¥dashline に対して，¥Drawline, ¥Dashline があるように，¥hasen に対しても ¥Hasen を定義しています。その書式は

¥Hasen[#1]#2

#1 : L=**（破線の実線部分の長さ） デフォルト値**=1mm

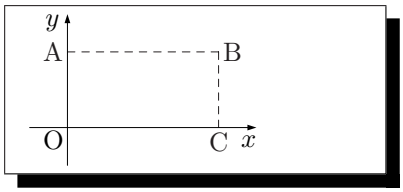
: G=**（破線のギャップの長さ） デフォルト値**=0.9mm

#2 : **破線で結ぶ点列**

¥Hasen

```
¥begin{zahyou}[ul=10mm](-.5,2.5)(-.5,1.5)
¥tenretu{A(0,1)w;B(2,1)e;C(2,0)s}
¥Hasen{¥A¥B¥C}
¥end{zahyou}
```

→



複数の点列を定義する ¥tenretu など，まだ説明していないコマンドが登場してしまいましたが，ここでは折れ線を破線で引く例としてご覧ください。

2.3.4 ¥Hasen*

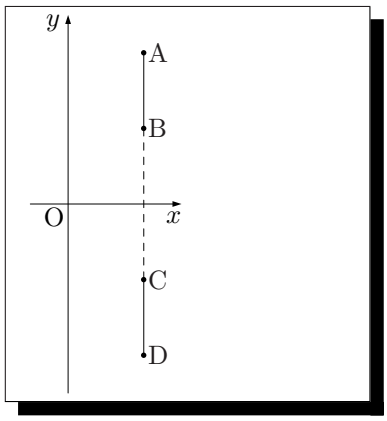
2 点を破線で結ぶコマンド ¥hasen, ¥Hasen は，両端から実線部分が始まりますが，これを両端からは空白部分が始まるようにしたものが*つきコマンドです。

下の例では，実線 AB と破線 BC の境界が B ではなくなってしまいます。

¥Hasen

```
¥begin{zahyou}[ul=10mm](-.5,1.5)(-2.5,2.5)
¥tenretu{A(1,2)e;B(1,1)e;C(1,-1)e;D(1,-2)e}
¥kuromaru{¥A;¥B;¥C;¥D}
¥Drawlines{¥A¥B;¥C¥D}
¥Hasen{¥B¥C}
¥end{zahyou}
```

→

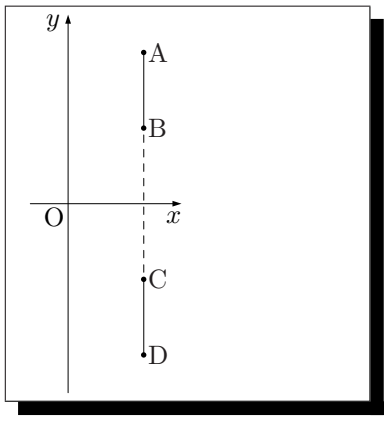


¥Hasen*と比較してみてください。

¥Hasen*

```
¥begin{zahyou}[ul=10mm](-.5,1.5)(-2.5,2.5)
¥tenretu{A(1,2)e;B(1,1)e;C(1,-1)e;D(1,-2)e}
¥kuromaru{¥A;¥B;¥C;¥D}
¥Drawlines{¥A¥B;¥C¥D}
¥Hasen*{¥B¥C}
¥end{zahyou}
```

→

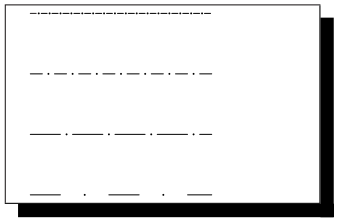


2.4 鎖線

鎖線をひくコマンドは ¥Chainline です。

¥Chainline

```
¥begin{picture}(3,3)%
¥def¥A{(0,0)}%
¥def¥B{(3,0)}%
¥put(0,3){¥Chainline{¥A¥B}}%
¥put(0,2){¥Chainline[.2][.1]{¥A¥B}}%
¥put(0,1){¥Chainline[.5][.1]{¥A¥B}}%
¥put(0,0){¥Chainline[.5][.4]{¥A¥B}}%
¥end{picture}
```

→ 

その書式は

¥Chainline[#1][#2]#3

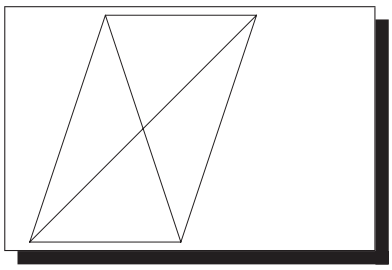
- #1 : **一つの線分の長さ**
- #2 : **線分と線分の間の長さ**
- #3 : **点列**

2.5 複数の折れ線

複数の折れ線を描画するコマンド ¥Drawlines もあります。複数の折れ線を ';' で区切ります。

¥Drawlines

```
¥begin{picture}(3,3)%
¥def¥O{(0,0)}%
¥def¥A{(2,0)}%
¥def¥B{(3,3)}%
¥def¥C{(1,3)}%
¥Drawlines{¥O¥A¥B¥C¥O¥B;¥A¥C}%
¥end{picture}
```

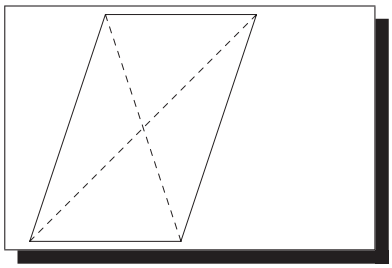
→ 

折れ線の線種を変更するには，<...> オプションを用います。

線種の変更

```
¥begin{picture}(3,3)%
¥def¥O{(0,0)}%
¥def¥A{(2,0)}%
¥def¥B{(3,3)}%
¥def¥C{(1,3)}%
¥Drawline{¥O¥A¥B¥C¥O}%
¥Drawlines<%
sensyu=¥dashline[40]{.1}>{%
¥O¥B;¥A¥C}%
¥end{picture}
```

→



¥Drawlines の書式です。

```
¥Drawlines<#1>#2
  <#1>  : <sensyu=¥dashlines[40]{.1}>などによる
          線種のローカルな変更
   #2   : 複数の点列を ';' で区切る
```

2.6 矢線

矢線を引くコマンドは 2 種類用意してあります。
¥yasen は，成分を指定します。
もうひとつの ¥ArrowLine は，始点と終点を指定します。

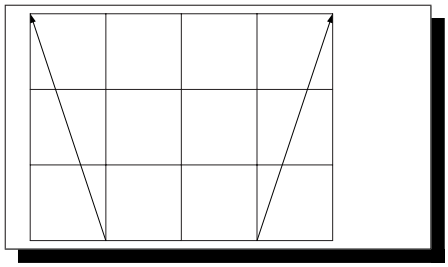
2.6.1 ¥yasen

まずは成分を与えて矢線を描画するコマンド ¥yasen を使用する一例です：

¥yasen

```
¥begin{zahyou*}[ul=10mm](0,4)(0,3)
  ¥put(0,0){¥kousi43}%
  ¥put(1,0){¥yasen(-1,3)}%
  ¥put(3,0){¥yasen(1,3)}%
¥end{zahyou*}
```

→

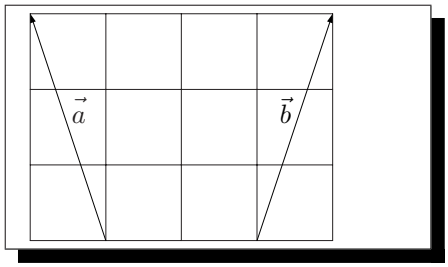


このコマンドは，矢線の傍に文字列を配置するオプションをもっています。

<.>オプション

```
¥begin{zahyou*}[ul=10mm](0,4)(0,3)
  ¥put(0,0){¥kousi43}%
  ¥put(1,0){%
    ¥yasen<[ne]{¥beku a}>(-1,3)}%
  ¥put(3,0){%
    ¥yasen<[nw]{¥beku b}>(1,3)}%
¥end{zahyou*}
```

→

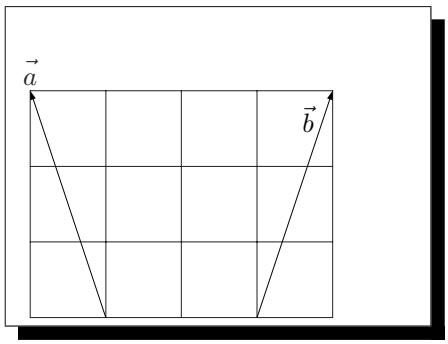


すなわち<...>オプションに，文字列を与えます。その際，¥Put の配置オプションを附加することができます。なお，文字列を配置する基準点は，矢線の中点です。これを動かすのが [..] オプションです。

[..] オプション

```
¥begin{zahyou*}[ul=10mm](0,4)(0,4)
  ¥put(0,0){¥kousi43}%
  ¥put(1,0){%
    ¥yasen[1]<[n]{¥beku a}>(-1,3)}%
  ¥put(3,0){%
    ¥yasen[.8]<[nw]{¥beku b}>(1,3)}%
¥end{zahyou*}
```

→



[..] オプションのデフォルト値は 0.5，すなわち中点です。これを [1] とすれば，終点，[0] とすれば，始点がそれぞれ文字列配置の基準点となります。

¥yasen の書式です。

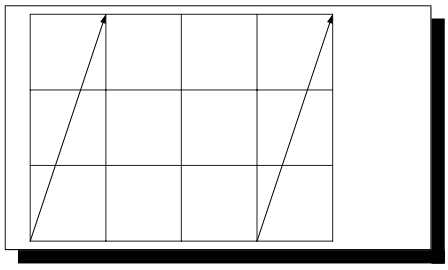
```
¥yasen[#1]<#2>(#3,#4)
¥Yasen[#1]<#2>#3
  #1 : ラベルを置く位置（デフォルト=0.5，中点）
  #2 : ラベル（位置指定を含めて）
  (#3,#4) : 成分（始点は ¥put（¥Put）で指定）
```

成分を，まとめてひとつにして扱うものが ¥Yasen です。

¥Yasen

```
¥begin{zahyou*}[ul=10mm](0,4)(0,3)
  ¥def¥avec{(1,3)}
  ¥put(0,0){¥kousi43}%
  ¥put(0,0){¥Yasen¥avec}%
  ¥put(3,0){¥Yasen¥avec}%
¥end{zahyou*}
```

→



¥Yasen に対しても，[...], <...> オプションが ¥yasen と同様に使用できます。

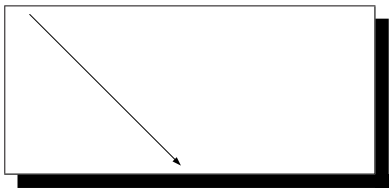
2.6.2 ¥ArrowLine

矢線を引くコマンドが ¥ArrowLine です.

¥ArrowLine

```
¥begin{picture}(2,2)
¥def¥A{(0,2)}%
¥def¥B{(2,0)}%
¥ArrowLine¥A¥B%
¥end{picture}
```

→



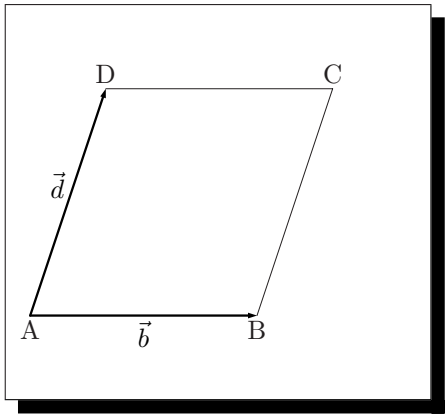
矢線の傍に文字列を配置することも可能です。

¥ArrowLine の文字列配置オプション

```
  ¥begin{zahyou*}[ul=10mm](0,4)(-1,4)
    ¥tenretu{A(0,0)s;B(3,0)s;
      C(4,3)n;D(1,3)n}
    ¥Drawline{¥B¥C¥D}
    {¥thicklines
    ¥ArrowLine<putstr=[s]{¥protect¥beku
b}>
      ¥A¥B

¥ArrowLine<putstr=[nw]{¥protect¥beku
d}>
      ¥A¥D}%
  ¥end{zahyou*}
```

→

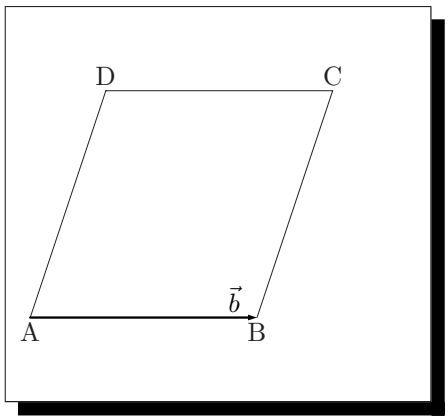


すなわち，<...>オプションとして，putstr の右辺値に，文字列を配置オプションとともに記述します。配置基準点（デフォルトは中点）を変更するには，キーワード putpos を用います。

¥ArrowLine の文字列配置基準点の変更

```
  ¥begin{zahyou*}[ul=10mm](0,4)(-1,4)
    ¥tenretu{A(0,0)s;B(3,0)s;
      C(4,3)n;D(1,3)n}
    ¥Drawline{¥B¥C¥D¥A}
    {¥thicklines
    ¥ArrowLine<putpos=0.9,
      putstr=[n]{¥beku b}>
        ¥A¥B}
  ¥end{zahyou*}
```

→



¥ArrowLine の書式です。

¥ArrowLine<#1>[#2]#3#4
  #1 : key=val
       sensyu=
       putstr=　　**（矢線の傍に置く配置オプション+文字列）**
       putpos=　　**（文字列の配置基準位置：デフォルトは** 0.5**，中点）**
  #2 : **矢印を置く位置（デフォルト** = 1 **すなわち終点）**
       ただし #2=b **のときは，両端に矢印**
  #3 : **始点**
  #4 : **終点**

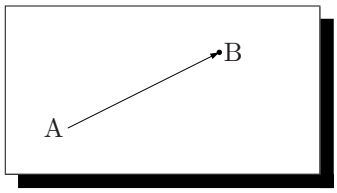
2.6.3　始点・終点位置の微調整

¥ArrowLine で引かれる矢線の始点・終点を少しずらしたいときがあります。例えば

¥ArrowLine

```
¥begin{zahyou*}[ul=10mm](-.5,2.5)(-.5,1.5)
  ¥tenretu{A(0,0)w;B(2,1)e}
  ¥Kuromaru¥B
  ¥ArrowLine¥A¥B
¥end{zahyou*}
```

→



のような場合，矢印が終点の黒丸にめり込んでいます。矢印の終点を左下に少しずらしたいときなどのために

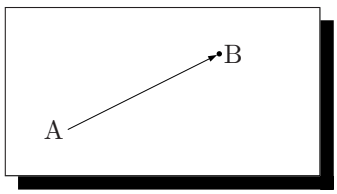
<Henvi=..>　　**始点に対する修正ベクトル**
<Henvii=...>　**終点に対する修正ベクトル**

オプションを新設しました。右辺値はベクトルで，成分は単位を伴った長さです。右辺値には‘,’が入りますから，右辺値全体を{...}でくくる必要があります。

<Henvii=..>オプション

```
¥begin{zahyou*}[ul=10mm](-.5,2.5)(-.5,1.5)
  ¥tenretu{A(0,0)w;B(2,1)e}
  ¥Kuromaru¥B
  ¥ArrowLine<Henvii={(-.8pt,-.4pt)}>¥A¥B
¥end{zahyou*}
```

→



なお

<henvi=..>　　**始点に対する修正ベクトル**
<henvii=...>　**終点に対する修正ベクトル**

も同機能ですが，右辺値の成分は ¥unitlength を単位とする無名数で与えます。

2.6.4 鏃の形状

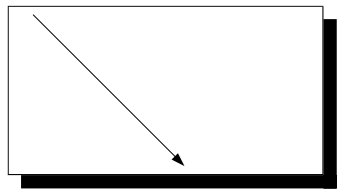
鏃のサイズを変更することもできます．例えば

```
¥changeArrowHeadSize{1.5}
```

とすれば，鏃の大きさは 1.5 倍になります。

¥changeArrowHeadSize

```
¥begin{picture}(2,2)
¥def¥A{(0,2)}
¥def¥B{(2,0)}
{¥changeArrowHeadSize{1.5}
¥ArrowLine¥A¥B}
¥end{picture}
```



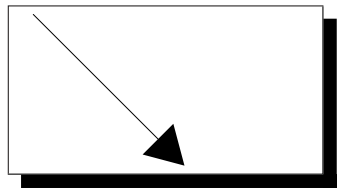
鏃の開き具合も調整できます．デフォルトは

```
¥def¥ArrowHeadAngle{18}%
```

となっています．これを 30 とすると，鏃は正三角形になります.

¥ArrowHeadAngle

```
{¥changeArrowHeadSize{3}%
¥begin{picture}(2,2)
¥def¥A{(0,2)}%
¥def¥B{(2,0)}%
¥changeArrowHeadSize[30]{1.5}%
¥ArrowLine¥A¥B%
¥end{picture}}
```



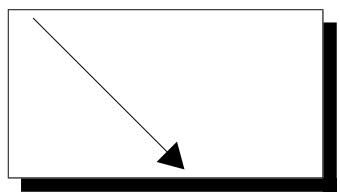
サイズ・開き具合の両方を同時に変更するときは

```
¥changeArrowHeadSize[30]{3}
```

と ¥changeArrowHeadSize のオプションで開き角を指定することもできます。

¥ArrowHeadAngle

```
{¥changeArrowHeadSize[30]{3}%
¥begin{picture}(2,2)
¥def¥A{(0,2)}%
¥def¥B{(2,0)}%
¥ArrowLine¥A¥B%
¥end{picture}}
```



¥ArrowLine で引かれる矢線において，鏃は二等辺三角形を塗りつぶしていますが，その底辺に窪みをつけることも可能です。窪みの深さの「二等辺三角形の高さを 1 としての比率」を ¥ArrowHeadPit で指定します。

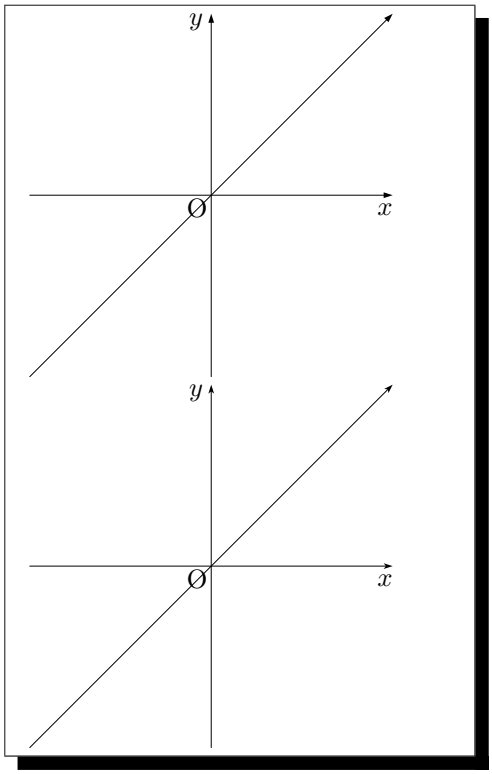
次の図では，座標軸が 2 個描画されますが，上が従来の形状，下が窪みをつけた形状です。

¥ArrowHeadPit

```
¥begin{zahyou}[ul=8mm](-3,3)(-3,3)
  ¥ArrowLine¥LB¥RT
¥end{zahyou}

¥def¥ArrowHeadPit{0.25}
¥begin{zahyou}[ul=8mm](-3,3)(-3,3)
  ¥ArrowLine¥LB¥RT
¥end{zahyou}
```

→



¥changeArrowHeadSize に<...>オプションを用いて，窪みを指定することもできます。

```
¥def¥ArrowHeadPit{.25}
```

と

```
¥changeArrowHeadSize<.25>{1}
```

は同値です。前者の方が簡潔ですが，矢印の長さ・開き角を変更する際には後者の方が便利でしょうか。

そんな馬鹿な，という極端な例です：

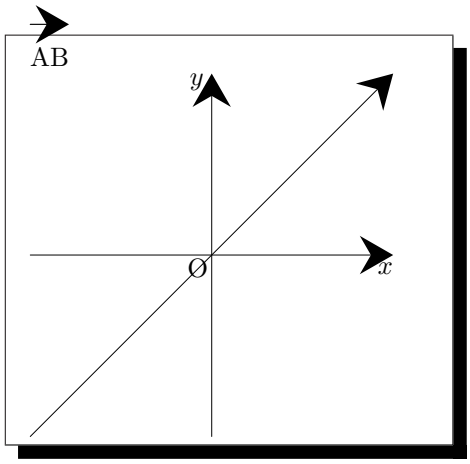
¥changeArrowSize による指定

```
¥changeArrowHeadSize[30]<0.3333>{4}
¥bekutorukata{fill}

¥bekutoru{AB}

¥begin{zahyou}[ul=8mm](-3,3)(-3,3)
  ¥ArrowLine¥LB¥RT
¥end{zahyou}
```

→



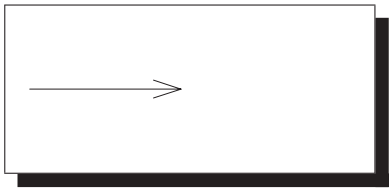
なお，これら矢線では鏃が塗りつぶされますが，これを枠線のみ描画させるには

```
¥renewcommand¥ArrowHeadType{1}
```

とします。デフォルトは ¥newcommand¥ArrowHeadType{f} としてあります.

塗りつぶさない鏃

```
{¥def¥ArrowHeadType{1}%
¥changeArrowHeadSize{3}%
¥begin{picture}(2,2)
¥def¥A{(0,1)}%
¥def¥B{(2,1)}%
¥ArrowLine¥A¥B%
¥end{picture}}
```

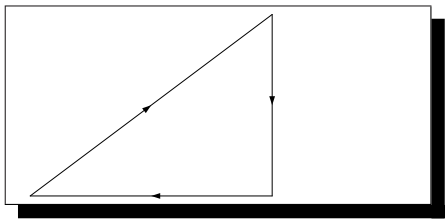
→ 

2.6.5 矢印の位置

矢印を途中に付けるためには，[...] オプションで，矢印を置く位置を指定します.

¥ArrowLine[0.5]

```
{¥unitlength8mm¥small
¥begin{picture}(4,3)%
¥def¥O{(0,0)}%
¥def¥A{(4,3)}%
¥def¥B{(4,0)}%
¥ArrowLine[.5]¥O¥A%
¥ArrowLine[.5]¥A¥B%
¥ArrowLine[.5]¥B¥O%
¥end{picture}}%
```

→ 

また，矢印を両向きにつけたいときは [b] オプションをつけます。

両向き矢印

```
¥unitlength10mm¥relax
¥begin{picture}(2,1)
¥def¥A{(0,.5)}
¥def¥B{(2,.5)}
¥ArrowLine[b]¥A¥B
¥end{picture}
```

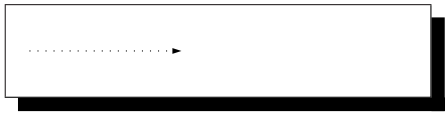
→ 

2.6.6 矢線を点線・破線

矢線の線種を変更するには ¥ArrowLine に<sensyu=...> オプションを付与します。

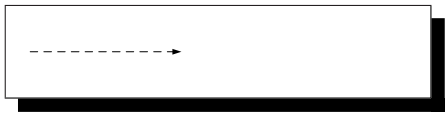
矢線を点線で

```
¥unitlength10mm¥relax
¥begin{picture}(2,1)
¥def¥A{(0,.5)}
¥def¥B{(2,.5)}
¥ArrowLine%
  <sensyu=¥protect¥emdottedline>%
  ¥A¥B
¥end{picture}
```

→ 

矢線を破線で

```
¥unitlength10mm¥relax
¥begin{picture}(2,1)
¥def¥A{(0,.5)}
¥def¥B{(2,.5)}
¥ArrowLine%
  <sensyu=¥protect¥hasen>%
  ¥A¥B
¥end{picture}
```

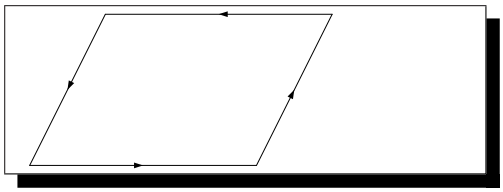
→ 

2.6.7 複数の矢線

複数の矢線を引くコマンドが ¥ArrowLines です。始点・終点を ';' で区切って並べます。

¥ArrowLines

```
¥unitlength=10mm
¥begin{picture}(4,2)
¥def¥A{(0,0)}
¥def¥B{(3,0)}
¥def¥C{(4,2)}
¥def¥D{(1,2)}
¥ArrowLines[.5]{%
  ¥A¥B;¥B¥C;¥C¥D;¥D¥A}
¥end{picture}
```

→ 

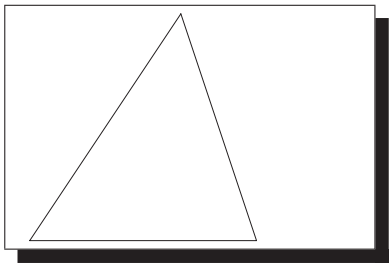
2.7 多角形

多角形を描画するコマンドが ¥Takakkei です。

¥ 多角形

```
¥begin{picture}(3,3)%
¥def¥O{(0,0)}%
¥def¥A{(3,0)}%
¥def¥B{(2,3)}%
¥Takakkei{¥O¥A¥B}%
¥end{picture}
```

→



¥Takakkei の始点 O と終点 B を結ぶ線分が強制的に付加されます。

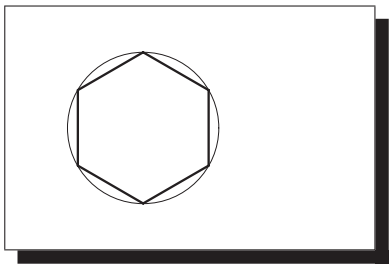
2.8 正多角形，極座標→直交座標

円周上に点をとるときは，極座標形式の方が便利です．これを直交座標に変換するコマンド ¥kyokuTyoku です．使用例として，正六角形を描画してみました．

¥kyokuTyoku

```
¥begin{picture}(3,3)(-1.5,-1.5)%
¥kyokuTyoku(1,90)¥A%
¥kyokuTyoku(1,150)¥B%
¥kyokuTyoku(1,210)¥C%
¥kyokuTyoku(1,270)¥D%
¥kyokuTyoku(1,330)¥E%
¥kyokuTyoku(1,30)¥F%
¥def¥O{(0,0)}%
¥En¥O{1}%
¥thicklines
¥Drawline{¥A¥B¥C¥D¥E¥F¥A}%
¥thinlines
¥end{picture}
```

→



```
¥kyokuTyoku(1,90)¥A
```

で，極座標 (1,90) を直交座標に変換した (0,1) が ¥A にセットされます．¥kyokuTyoku の書式です．

```
¥kyokuTyoku(#1,#2)#3
    (#1,#2) : 極座標
    #3      : 変換した直交座標を代入するコントロールシーケンス
```

(1) このスタイルファイルでは，原則として，角の単位は六十分法で表します．

(2) ¥kyokuTyoku は点を一つずつ定義しますが，複数の点列を定義するコマンド

¥rtenretu, ¥rtenretu*

もあります。

(3) 正多角形を描画するための emPoly.sty もあります。

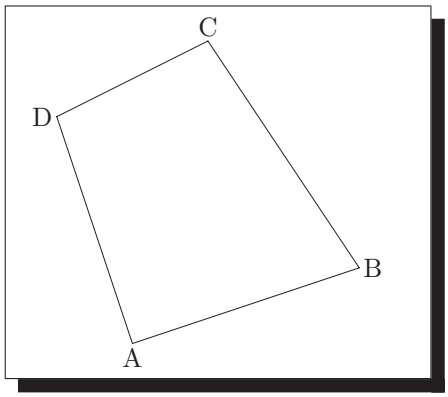
2.9 角の丸い多角形

多角形を描画するコマンド ¥Takakkei のバリエーション ¥ovalTakakkei コマンドは多角形のコーナーを丸くするコマンドです。まずは，¥Takakkei の使用例からご覧ください。

¥Takakkei

```
¥begin{zahyou*}%
    [ul=10mm,Ueyohaku=1em,
      Hidariyohaku=1em,%
      Sitayohaku=1em]%
    (0,4)(0,4)
  ¥tenretu{A(1,0)s;B(4,1)e;%
    C(2,4)n;D(0,3)w}
  ¥Takakkei{¥A¥B¥C¥D}
¥end{zahyou*}
```

→

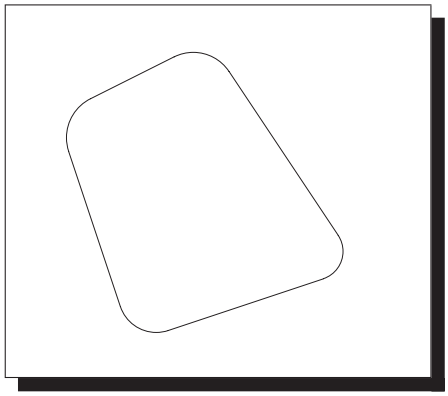


コーナーを切り取って，円弧で結びます。切り取る線分の長さを ¥ovalTakakkei の第 1 引数に与えます。単位を伴った数値を指定します。

¥ovalTakakkei

```
¥begin{zahyou*}%
    [ul=10mm,Ueyohaku=1em,
      Hidariyohaku=1em,%
      Sitayohaku=1em]%
    (0,4)(0,4)
  ¥tenretu*{A(1,0)s;B(4,1)e;%
    C(2,4)n;D(0,3)w}
  ¥ovalTakakkei{5mm}{¥A¥B¥C¥D}
¥end{zahyou*}
```

→



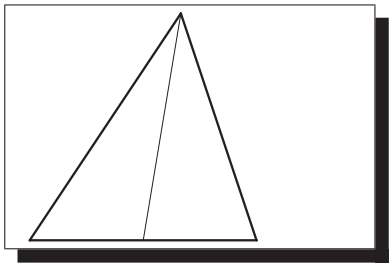
2.10 分点

線分を内分（外分）する点を求めるコマンド ¥Bunten です．△ABC の辺 BC の中点 M と A を結ぶ線分を描画する例です．

¥Bunten

```
¥begin{picture}(3,3)%
¥def¥A{(2,3)}%
¥def¥B{(0,0)}%
¥def¥C{(3,0)}%
¥Bunten¥B¥C{1}{1}¥M%
¥Drawline{¥A¥M}
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥thinlines
¥end{picture}
```

$\longrightarrow$



¥Bunten の書式は

¥Bunten#1#2#3#4#5

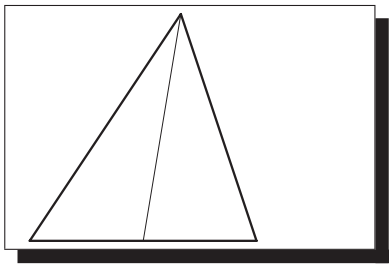
- #1 : **端点** 1
- #2 : **端点** 2
- #3#4 : **内（外）分比**
- #5 : **分点**

すなわち #1 と #2 を結ぶ線分を #3:#4 に分ける点の座標を #5 にセットします.
特に 1 : 1 の内分点，すなわち中点を求めるコマンド ¥Tyuuten も便利です。

¥Bunten

```
¥begin{picture}(3,3)%
¥def¥A{(2,3)}%
¥def¥B{(0,0)}%
¥def¥C{(3,0)}%
¥Tyuuten¥B¥C¥M%
¥Drawline{¥A¥M}
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥thinlines
¥end{picture}
```

$\longrightarrow$



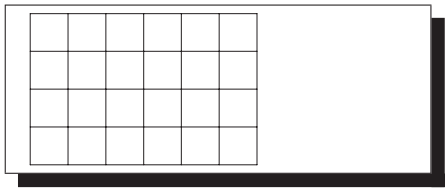
2.11 格子

碁盤の目状の街路図などは良く登場します．格子を描画するコマンド ¥kousi です.

¥kousi

```
¥unitlength5mm
¥begin{picture}(6,4)%
¥kousi{6}{4}%
¥end{picture}%
```

→

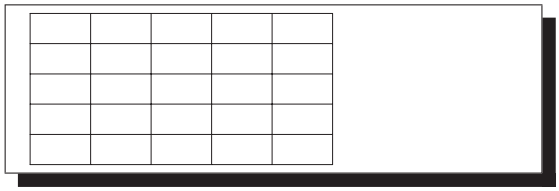


縦横のサイズを変更したいときは，（**横サイズ**, **縦サイズ**）オプションを指定します．サイズの単位は ¥unitlength です．

¥kousi(..,..) **オプション**

```
{¥unitlength8mm¥small
¥begin{picture}(5,2.5)%
¥kousi(1,0.5){5}{5}%
¥end{picture}}%
```

→



¥kousi の書式です．

¥kousi(#1,#2)#3#4

- #1 ： **横方向 1 区画の長さ（デフォルト=1）**
- #2 ： **縦方向 1 区画の長さ（デフォルト=1）**
- #3 ： **横方向のブロック数**
- #4 ： **縦方向のブロック数**

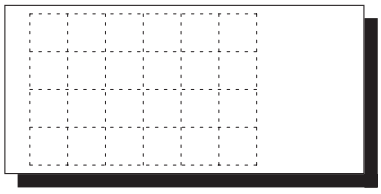
置く位置は ¥put で指定する．

格子を点線・破線で引いてみます。

格子線を破線で引く

```
¥unitlength5mm
¥begin{picture}(6,4)%
{¥def¥sensyu{¥dashline[40]{.1}}%
¥kousi{6}{4}}%
¥end{picture}%
```

→

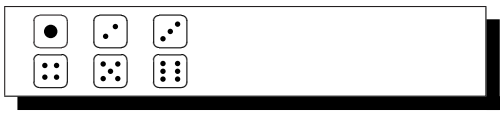


2.12 さいころ

さいころの目を表すコマンド ¥saikoro を用意しました．

さいころ

```
¥saikoro{1}~
¥saikoro{2}~
¥saikoro{3}¥¥
¥saikoro{4}~
¥saikoro{5}~
¥saikoro{6}
```

→ 

ただし，このコマンドは ascmac.sty で定義されている ¥keytop コマンドを使用していますから，このスタイルファイルを使用することが前提です．さらに，このスタイルファイルは emathP から読み込まれる eepic.sty と相性が悪いことがあります．その対策を itembbox.sty で施していますので，ascmac.sty に引き続き itembbox.sty も読み込んでおく必要があります．

3　文字列

3.1　文字列

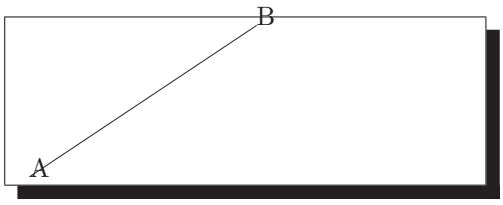
3.1.1　¥Put

LaTeX の picture 環境において，文字列を配置するコマンドは ¥put です。
emath では，点の位置を ¥A などの変数名で指定します。そこで ¥Put の登場となります。

¥Put

```
¥begin{picture}(3,2)
¥def¥A{(0,0)}
¥def¥B{(3,2)}
¥Drawline{¥A¥B}
¥Put¥A{A}
¥Put¥B{B}
¥end{picture}
```

→



うまくないですね。単に ¥Put¥A{A} では，文字 'A' の左下が指定した点 A(0,0) にくるように配置されます。これを修正するのに

```
(dx,dy)[pos]
```

オプションを用意しました。次節以降この解説をしますが，とりあえず ¥Put の書式をご覧ください。

```
¥Put#1(#2,#3)[#4]#5
    #1 : 文字列を置く位置（座標）
    (#2,#3) : 位置の修正ベクトル（長さの単位が必要です。）
    #4 : 配置 (l,r,t,b)
    #5 : 文字列
```

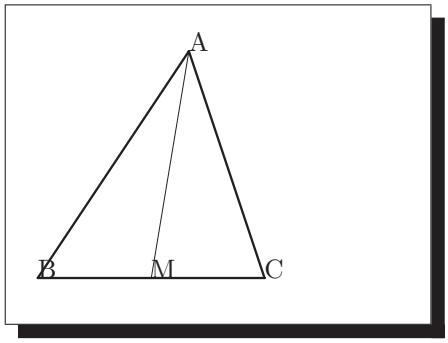
3.1.2　文字位置の調整

文字列位置の微調整を行う例です。

位置の調整前

```
¥begin{picture}%
(3.2,4)(-.1,-.5)%
¥def¥A{(2,3)}%
¥def¥B{(0,0)}%
¥def¥C{(3,0)}%
¥Bunten¥B¥C{1}{1}¥M%
¥Put¥A{A}%
¥Put¥B{B}%
¥Put¥C{C}%
¥Put¥M{M}%
¥Drawline{¥A¥M}
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥thinlines
¥end{picture}
```

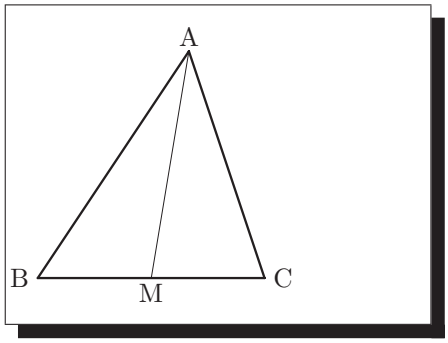
→



位置の調整後

```
¥begin{picture}%
(3.2,4)(-.1,-.5)%
¥def¥A{(2,3)}%
¥def¥B{(0,0)}%
¥def¥C{(3,0)}%
¥Bunten¥B¥C{1}{1}¥M%
¥Put¥A(0,2pt)[b]{A}%
¥Put¥B(0,0)[r]{B }%
¥Put¥C(0,0)[l]{ C}%
¥Put¥M(0,-2pt)[t]{M}%
¥Drawline{¥A¥M}
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥thinlines
¥end{picture}
```

→



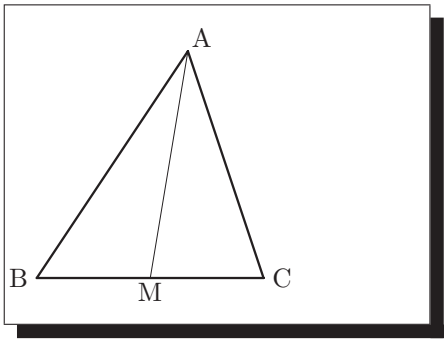
A の調整は (0,2pt) の指定により，頂点 A の上 2pt のところが基準になります。[b] 指定により，基準点が文字の bottom となるように配置されます。(x,y) の x, y は単位をつけた数値です。0 だけは単位をつけなくてもよい，としてあります。

文字位置の調整量 (#2,#3) は直交座標成分ですが，[r](#2,#3) とすれば，極座標指定とみなされます。A の位置を変えてみます。

調整量を極座標指定

```
¥begin{picture}%
(3.2,4)(-.1,-.5)%
¥def¥A{(2,3)}%
¥def¥B{(0,0)}%
¥def¥C{(3,0)}%
¥Bunten¥B¥C{1}{1}¥M%
¥Put¥A[r](2pt,45)[lb]{A}%
¥Put¥B(0,0)[r]{B }%
¥Put¥C(0,0)[l]{ C}%
¥Put¥M(0,-2pt)[t]{M}%
¥Drawline{¥A¥M}
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥thinlines
¥end{picture}
```

→



3.1.3 文字列位置の簡易指定

位置の調整を記述するのは面倒なので，簡易指定オプションを用意しました。その書式です。

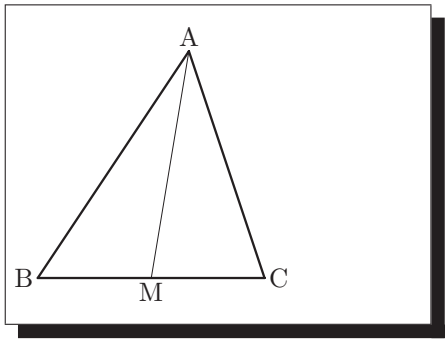
```
¥Put#1[#2]#3
    #1 : 文字列を置く位置（座標）
    #2 : 位置指定オプション
      n = 北　（上）
      nw= 北西（左上）
      w = 西　（左）
      sw= 南西（左下）
      s = 南　（下）
      se= 南東（右下）
      e = 東　（右）
      ne= 北東（右上）
    #3 : 文字列
```

これを用いると，先の例は次のように記述できます。

位置の簡易指定オプション

```
¥begin{picture}%
(3.2,4)(-.1,-.5)%
¥def¥A{(2,3)}%
¥def¥B{(0,0)}%
¥def¥C{(3,0)}%
¥Bunten¥B¥C{1}{1}¥M%
¥Put¥A[n]{A}% 位置の
¥Put¥B[w]{B}% 簡易指定
¥Put¥C[e]{C}% オプション
¥Put¥M[s]{M}%
¥Drawline{¥A¥M}
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥thinlines
¥end{picture}
```

→



すなわち，点からみて文字列をおくべき方位を指定します。

（注）¥Put の正しいコマンド名は ¥emathPut です。コマンド名 ¥Put が他のスタイルファイルと競合した場合は，¥emathPut をお使いください。

3.1.4 文字列の回転

¥Put[#1] には前節のほか次のオプションがあります。

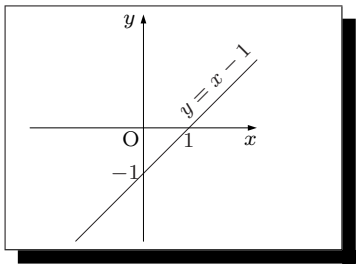
kaiten	=ddd	**文字を回転する**
houkou	=vvv	**回転角を間接的に指定**
from	=PPP	**次の** to=QQQ **と併せて回転角を間接的に指定**
to	=QQQ	

次は kaiten=ddd で文字を回転させます。回転角 ddd は符号付きの六十分法でです。

kaiten=

```
¥unitlength6mm¥footnotesize
¥begin{zahyou}(-2.5,2.5)(-2.5,2.5)
¥tenretu*{A(0,-1);B(1,0)}
¥Put¥A[w]{$-1$}
¥Put¥B[s]{1}
¥Tyokusen¥A¥B{}{}
¥Put[kaiten=45]¥B(0,1mm)[lb]{%
$y=x-1$}
¥end{zahyou}
```

→

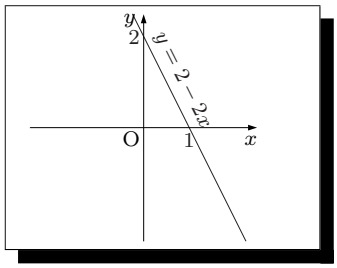


回転角を方向ベクトルによって与えるオプション houkou=vvv です。

houkou=

```
¥unitlength6mm¥footnotesize
¥begin{zahyou}(-2.5,2.5)(-2.5,2.5)
¥tenretu*{A(0,2);B(1,0)}
¥Put¥A[w]{2}
¥Put¥B[s]{1}
¥Tyokusen¥A¥B{}{}
¥Subvec¥B¥A¥AB
¥Put[houkou=¥AB]¥A(1mm,0)[lb]{%
$y=2-2x$}
¥end{zahyou}
```

→

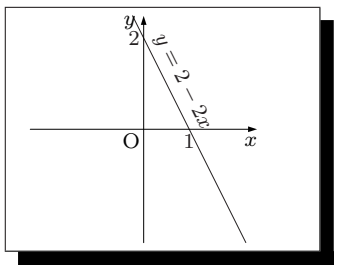


2 点を与えて回転角を指示する例です。

from= , to=

```
¥unitlength6mm¥footnotesize
¥begin{zahyou}(-2.5,2.5)(-2.5,2.5)
¥tenretu*{A(0,2);B(1,0)}
¥Put¥A[w]{2}
¥Put¥B[s]{1}
¥Tyokusen¥A¥B{}{}
¥Put[from=¥A,to=¥B]¥A(1mm,0)[lb]{%
$y=2-2x$}
¥end{zahyou}
```

→

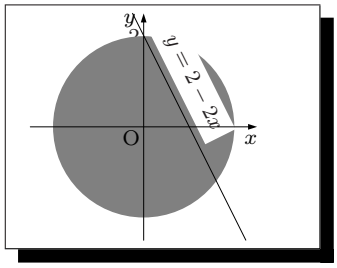


文字列を白抜きにします。

白抜き

```
¥unitlength6mm¥footnotesize
¥begin{zahyou}(-2.5,2.5)(-2.5,2.5)
¥tenretu*{O(0,0);A(0,2);B(1,0)}
¥Put¥A[w]{2}
¥Put¥B[s]{1}
¥En*¥O{2}
¥Tyokusen¥A¥B{}{}
¥Put[from=¥A,to=¥B]¥A(%
1mm,0)[lb]{¥colorbox{white}{$y=2-2x$}}
¥end{zahyou}
```

→

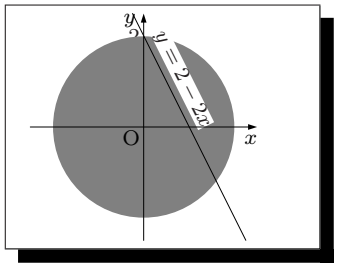


文字の外側の余白は ¥fboxsep により調整できます。

¥fboxsep による余白調整

```
¥unitlength6mm¥footnotesize
¥begin{zahyou}(-2.5,2.5)(-2.5,2.5)
¥tenretu*{O(0,0);A(0,2);B(1,0)}
¥Put¥A[w]{2}
¥Put¥B[s]{1}
¥En*¥O{2}
¥Tyokusen¥A¥B{}{}
{¥fboxsep=0pt¥Put[from=¥A,to=¥B]¥A(%
1mm,0)[lb]{¥colorbox{white}{$y=2-2x$}}}
¥end{zahyou}
```

→



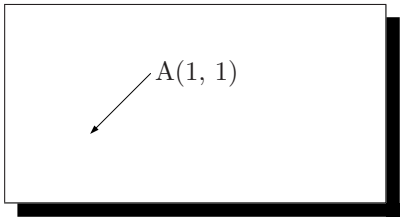
3.1.5 文字列から基準点への矢線

文字列を置きたい付近で図が込み入っている場合，文字列を少し離れた位置において，そこから該当個所に矢線を引くためのコマンドが ¥PutStr です。

¥PutStr

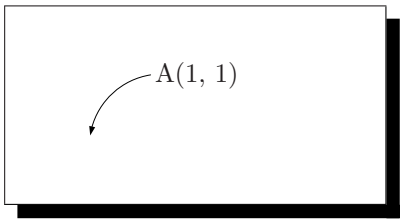
```
{¥unitlength8mm¥relax
¥begin{picture}(3,3)
¥def¥A{(1,1)}
¥def¥AA{(2,2)}
¥PutStr¥AA[e]{A(1, 1)}to¥A
¥end{picture}}
```

⟶



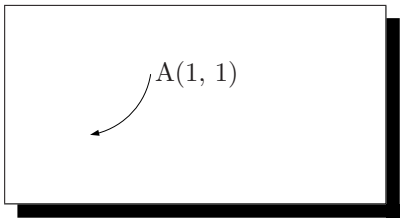
矢線を円弧にしたいときは to の後ろに [**半径**] オプションをつけます。半径は無名数で単位は ¥unitlength です。

```
¥PutStr
{¥unitlength8mm¥relax
¥begin{picture}(3,3)
¥def¥A{(1,1)}
¥def¥AA{(2,2)}
¥PutStr¥AA[e]{A(1, 1)}to[1.2]¥A
¥end{picture}}
```



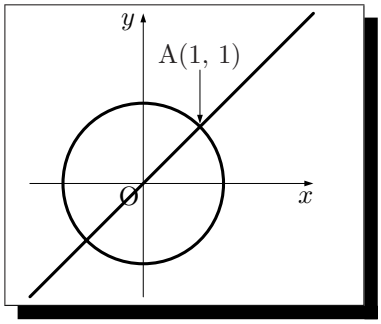
半径に負の値を指定すると，矢印が負の回転を表す方向につきます。

```
¥PutStr
{¥unitlength8mm¥relax
¥begin{picture}(3,3)
¥def¥A{(1,1)}
¥def¥AA{(2,2)}
¥PutStr¥AA[e]{A(1, 1)}to[-1.2]¥A
¥end{picture}}
```



addvec=**オプション**　矢印の先端が曲線に埋没しているのが気になる場合の対策です。矢印の先端の位置を微調整するためにのオプションが [addvec=..] です。上の例で，矢印の先端を上方に 1.1pt 持ち上げてみます。

```
¥PutStr
¥begin{zahyou}[ul=7.5mm](-2,3)(-2,3)
  ¥PutStr{(1,2)}(0,0)[b]{A(1, 1)}%
    to[addvec={(0,1.1pt)}]{(1,1)}
  {¥Thicklines
    ¥En¥O{1.414}
    ¥kTyokusen¥O{45}{}{}
  }%
¥end{zahyou}
```



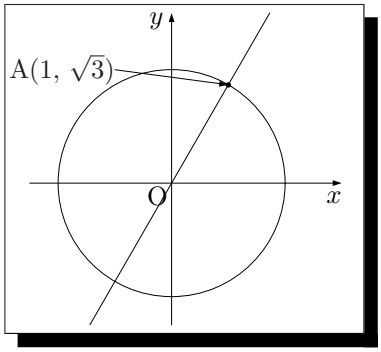
すなわち，addvec の右辺値に矢印の位置を修正するベクトルを与えますが，その成分は，単位を伴った数値です。x 成分，y 成分の間に ‘,’ が入りますから，addvec={(0,1.1pt)}と，右辺全体を{ }でくくっておく必要があります。

極座標形式の指定法　また，線幅が細くても対象点に黒丸をつけたりした場合も矢印が埋没します。

¥PutStr

```
¥begin{zahyou}[ul=7.5mm](-2.5,3)(-2.5,3)
  ¥def¥A{(1,1.732)}
  ¥PutStr{(-1,2)}(0,0)[r]%
    {A(1, $¥sqrt3$)}%
    to¥A
  ¥En¥O{2}
  ¥kTyokusen¥O{60}{}{}
  ¥Kuromaru¥A
¥end{zahyou}
```

→

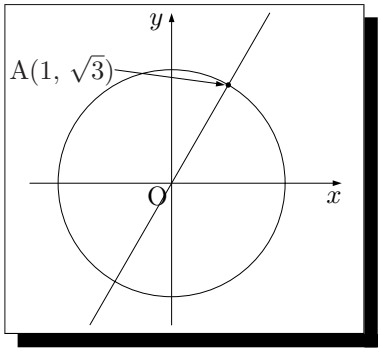


この場合は，微調整ベクトルを極座標形式で与える方法も用意しました。

¥PutStr

```
¥begin{zahyou}[ul=7.5mm](-2.5,3)(-2.5,3)
  ¥def¥A{(1,1.732)}
  ¥PutStr{(-1,2)}(0,0)[r]%
    {A(1, $¥sqrt3$)}%
    to[addvec={r(1.2pt,180)}]¥A
  ¥En¥O{2}
  ¥kTyokusen¥O{60}{}{}
  ¥Kuromaru¥A
¥end{zahyou}
```

→



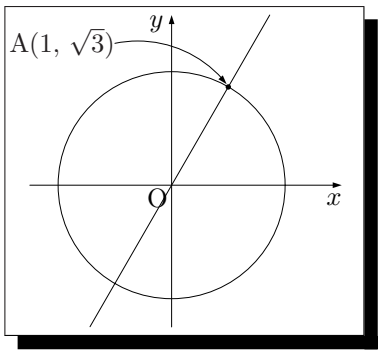
すなわち，r(.,.) と，先頭に ‘r’ を附加します。

円弧にする場合の半径指定　矢線を円弧にするには，[...] オプションに，[hankei=..] を附加します。

¥PutStr

```
¥begin{zahyou}[ul=7.5mm](-2.5,3)(-2.5,3)
  ¥def¥A{(1,1.732)}
  ¥PutStr{(-1,2.5)}(0,0)[r]%
    {A(1, $¥sqrt3$)}%
    to[hankei=-2,addvec={r(1.2pt,120)}]¥A
  ¥En¥O{2}
  ¥kTyokusen¥O{60}{}{}
  ¥Kuromaru¥A
¥end{zahyou}
```

→



右辺値は，¥unitlength を単位とする無名数であるのは，今までの仕様を引きずっています。

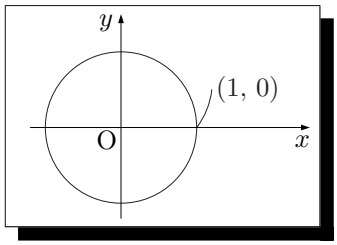
¥PutStr の arrowheadsize=オプション　¥PutStr は，デフォルトでは矢印がつきます。

矢印のサイズを変更するためにのオプションが arrowheadsize=.. です。特に右辺値を 0 とすれば，矢印がつかないこととなります。

¥PutStr

```
¥begin{zahyou}[ul=10mm](-1.2,2.5)(-1.2,1.5)
  ¥En¥O{1}
  ¥PutStr{(1.2,.5)}[e]{(1, 0)}to
    [hankei=-1,arrowheadsize=0]%
    {(1,0)}
¥end{zahyou}
```

→



¥PutStr の書式　¥PutStr の書式です。

```
¥PutStr#1[#2]#3to[#4]#5
   #1 : 文字列を置く位置
   #2 : #1 から見ての方位（デフォルト = e ）
   #3 : 文字列
   #4 : 文字列から出る矢印を円弧にしたいときは
        その半径を指定する。
        key=val の形式も可能
          hankei=..
          addvec=..
          arrowheadsize=..
   #5 : 文字列から出る矢印の終点
¥PutStr#1(#2,#3)[#4]#5to[#6]#7
   #1〜#5 : ¥Put 文と同じ
   #6 : 文字列から出る矢印を円弧にしたいときは
        その半径を指定する。
   #7 : 文字列から出る矢印の終点
```

3.2　複数の点の定義とラベル付け

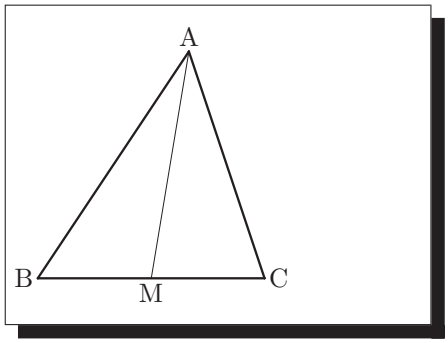
3.2.1　¥tenretu

複数の点の定義とラベル付けを同時に片付けようというのが ¥tenretu コマンドです。

¥tenretu

```
¥begin{picture}%
(3.2,4)(-.1,-.5)%
¥tenretu{%
  A(2,3)n;B(0,0)w;C(3,0)e}%
¥Bunten¥B¥C{1}{1}¥M%
¥Put¥M[s]{M}%
¥Drawline{¥A¥M}%
{¥thicklines
¥Drawline{¥A¥B¥C¥A}}%
¥end{picture}
```

→



複数の点の定義とラベルつけ

¥tenretu#1

¥tenretu*#1**（定義のみ　ラベル付けはオプション）**

#1 **は点列を‘;’で区切った列**

#1 **は**

[##1]##2(##3,##4)##5

の形式で点列を‘;’で区切る。

##1 : **オプションで，点の位置に置く文字列**
（省略時は，##2 **と同じ）**

##2 : ¥###2 **という変数の頂点名**

[r] : **をつけると極座標形式とみなす**

[s] : **をつけると相対移動とみなす**

(##3,##4) : **頂点の座標**

##5 : **頂点の近傍に置く文字列の配置オプション**
［方角］オプションに限り
区切子‘[’,‘]’を省略可能

¥edef¥###2{(##3,##4)}**として，点** ¥###2 **が定義され，**

¥Put **に**

¥Put¥###2##5{##1}

として引き渡される。）

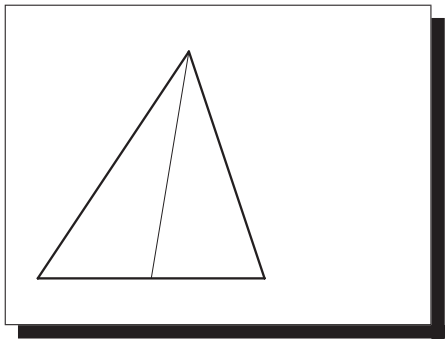
3.2.2 ¥tenretu*

を附加した ¥tenretu は，点を定義するだけで，ラベル付けはしません。

¥tenretu

```
¥begin{picture}%
(3.2,4)(-.1,-.5)%
¥tenretu*{%
  A(2,3);B(0,0);C(3,0)}%
¥Bunten¥B¥C{1}{1}¥M%
¥Drawline{¥A¥M}%
{¥thicklines
¥Drawline{¥A¥B¥C¥A}}%
¥end{picture}
```

→



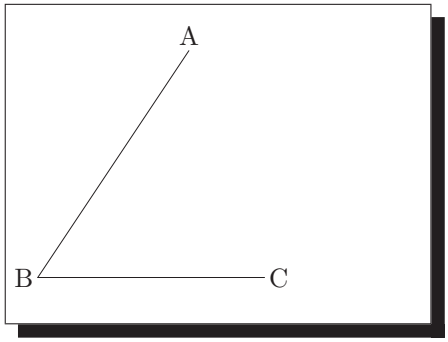
3.2.3 ¥oresen

更にそれらを折れ線で結んでしまおう，というのが ¥oresen です。

¥oresen

```
¥begin{picture}%
(3.2,4)(-.1,-.5)%
¥oresen{%
  A(2,3)n;B(0,0)w;C(3,0)e}%
¥end{picture}
```

→

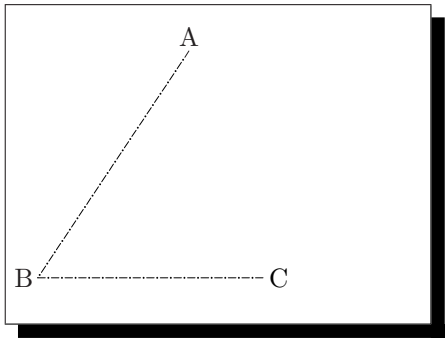


<sensyu=...>オプションを利用して，折れ線を点線・破線・鎖線にすることも可能です。

線種の変更（鎖線）

```
¥begin{picture}%
(3.2,4)(-.1,-.5)%
¥oresen<sensyu=¥chainline>{%
  A(2,3)n;B(0,0)w;C(3,0)e}%
¥end{picture}
```

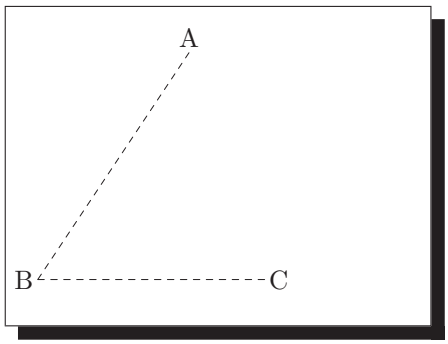
→



線種の変更（破線）

```
¥begin{picture}%
(3.2,4)(-.1,-.5)%
¥oresen<%
sensyu=¥dashline[40]{.1}>{%
  A(2,3)n;B(0,0)w;C(3,0)e}%
¥end{picture}
```

→



¥oresen の書式です。

¥oresen<#1>#2
  <#1> : **線種を変更するときのオプション**
    sensyu=¥dashline[40]{.1}
    sensyu=¥chainline[.4][.2]
    など
  #2 **は折れ線の頂点列を‘;’で区切った列**
    #2 **は**
      [##1]##2(##3,##4)##5
    の形式で点列を‘;’で区切る。
      ##1 : **オプションで，点の位置に置く文字列**
            （省略時は，##2 **と同じ）**
      ##2 : ¥###2 **という変数の頂点名**
      [r] : **をつけると極座標形式とみなす**
      [s] : **をつけると相対移動とみなす**
      (##3,##4) : **頂点の座標**
      ##5 : **頂点の近傍に置く文字列の配置オプション**
            ［方角］オプションに限り
            区切子‘[’,‘]’を省略可能
¥edef¥###2{(##3,##4)}**として，点** ¥###2 **が定義され，**
¥Put **に**
    ¥Put¥###2##5{##1}
として引き渡される。）

3.2.4 ¥rtenretu

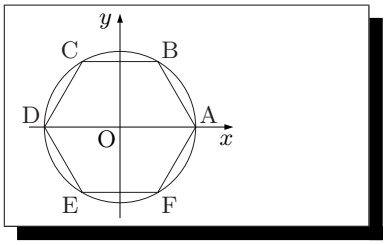
¥tenretu(*) による点の定義は直交座標を前提としています。[r] オプションで極座標形式にすることができますが，多くの点を極座標形式で定義するには，煩雑です。

そこで，すべての点が極座標形式であるときに使用するために ¥rtenretu(*) を作りました。

¥rtenretu

```
¥begin{zahyou}[ul=10mm](-1.2,1.5)(-1.2,1.5)
¥small
¥rtenretu{A(1,0)ne;B(1,60)ne;C(1,120)nw;
D(1,180)nw;E(1,240)sw;F(1,300)se}
¥Drawline{¥A¥B¥C¥D¥E¥F¥A}
¥En¥O{1}
¥end{zahyou}
```

→

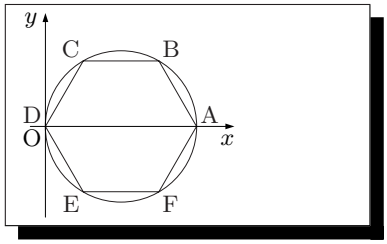


中心が原点以外のときは [...] オプションで指定します。

¥rtenretu[極]

```
¥begin{zahyou}[ul=10mm](-0.2,2.5)(-1.2,1.5)
¥small
¥def¥Tyuusin{(1,0)}
¥rtenretu[¥Tyuusin]{A(1,0)ne;B(1,60)ne;
C(1,120)nw;D(1,180)nw;E(1,240)sw;F(1,300)se}
¥Drawline{¥A¥B¥C¥D¥E¥F¥A}
¥En¥Tyuusin{1}
¥end{zahyou}
```

→



*付の ¥rtenretu*と ¥rtenretu の関係は，¥tenretu*と ¥tenretu の関係と同じです。

3.2.5 座標に計算式を記述

¥tenretu に<perl>オプションを付加すると，座標に計算式を記述することができます。ただし，perl との連携機能を前提としますから，samplePp.tex をご覧ください。

3.3 線分に文字列

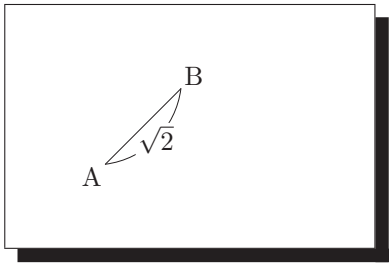
3.3.1 線分の長さ表記

線分の両端を円弧で結び，その中央部に長さなどを記入するためのコマンド ¥HenKo です。

¥HenKo

```
¥begin{zahyou*}(0,3)(0,3)
¥tenretu{A(1,1)sw;B(2,2)ne}
¥HenKo¥A¥B{$¥sqrt{2}$}%
¥Drawline{¥A¥B}
¥end{zahyou*}
```

→



¥HenKo は基本的には，

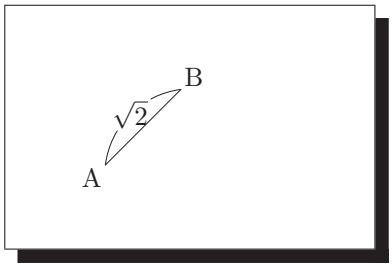
線分の両端と中央部に置く文字列

の 3 個の引数を与えます。なお，与える両端の点の順序を逆転させると，円弧と文字列が線分の反対側に表示されます。

端点の順序

```
¥begin{zahyou*}(0,3)(0,3)
¥tenretu{A(1,1)sw;B(2,2)ne}
¥HenKo¥B¥A{$¥sqrt{2}$}%
¥Drawline{¥A¥B}
¥end{zahyou*}
```

→



書式は

¥HenKo[#1]<#2>#3#4#5

#1: **弧を点線にする場合，点の個数（*を指定した場合は，一任）**
#2: key=val
henkoH=.. **辺と弧の距離（単位付数値）デフォルト値**=1.6ex
henkohi=.. **右辺値は無名数で，辺と弧の距離をデフォルトの何倍にするかを指定**
putoption=..
henkosep=..（**白抜きボックスの** ¥fboxsep）
yazirusi=a/r/b
henkomozikaiten=1/-1
linewidth=..
henkotype=0/1（0:**円弧（デフォルト）**，1:**楕円**，2:**折れ線**）
#3,#4 : **両端の点**
#5 : **配置する文字列**

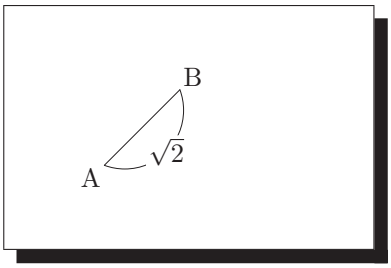
引数が多いですが，必須のものは，上の例のように #3，#4，#5 だけです。

線分と弧の間隔を調整するには，henkoH=.. オプションを用います。デフォルト値の 1.6ex から大きくしてみましょう。

文字と線分との間隔

```
¥begin{zahyou*}(0,3)(0,3)
¥tenretu{A(1,1)sw;B(2,2)ne}
¥HenKo<henkoH=3ex>¥A¥B{$¥sqrt{2}$}%
¥Drawline{¥A¥B}
¥end{zahyou*}
```

→

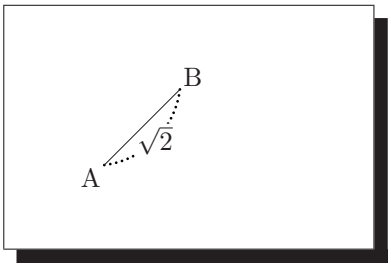


弧の部分を点線にするには [#1] で，弧上に置く点の個数を指定します。

弧を点線で

```
¥begin{zahyou*}(0,3)(0,3)
¥tenretu{A(1,1)sw;B(2,2)ne}
¥HenKo[20]¥A¥B{$¥sqrt{2}$}%
¥Drawline{¥A¥B}
¥end{zahyou*}
```

→

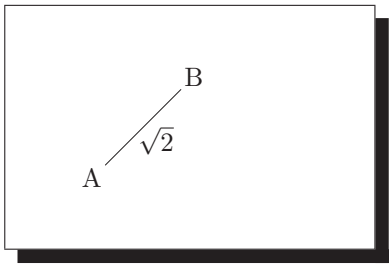


極端な話し，これを [0] と指定すれば，円弧は描かれません。

文字のみ

```
¥begin{zahyou*}(0,3)(0,3)
¥tenretu{A(1,1)sw;B(2,2)ne}
¥HenKo[0]¥A¥B{$¥sqrt{2}$}%
¥Drawline{¥A¥B}
¥end{zahyou*}
```

⟶

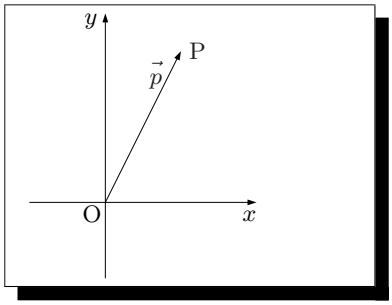


なお，文字列は線分の中央に置かれますが，これをどちらかに寄せるコマンドが ¥sPut です。

¥sPut

```
{¥unitlength10mm¥small
¥begin{zahyou}(-1,2)(-1,2.5)
¥def¥O{(0,0)}%
¥def¥P{(1,2)}%
¥Put¥P(0,0)[l]{ P}%
¥sPut[.75]¥O¥P(0,0)[rb]{%
    $¥beku p$}%
¥ArrowLine¥O¥P%
¥end{zahyou}}%
```

⟶



書式は

```
¥sPut[#1]#2#3(#4,#5)[#6]#7
    #1 : 比率 (0~1)
    #2 : 始点
    #3 : 終点
  (#4,#5) : 位置の修正ベクトル 単位必須
    #6 : ¥makebox の [..] オプション
    #7 : 文字列
#2 から #3 へ向かう線分の #1 倍の位置 (¥X) に，
    ¥Put¥X(#4,#5)[#6]#7 として文字列 (#7) を置く。
```

（注）dviout.exe による印刷で，辺の傍に記した長さの文字などが黒いボックスになってしまう場合は，dviout の graphic — color specials 設定を replace(def) または replace(bak) にしてみてください。

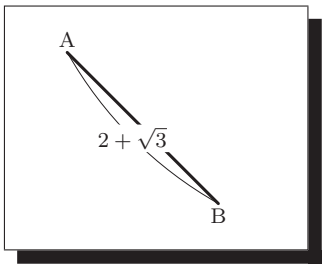
3.3.2 ¥HenKo：文字列の配置調整

¥HenKo で線分の傍に置く文字列が長くなったときなど文字位置を調整したいときがあります。その一例です。

修正前

```
¥unitlength10mm¥footnotesize
¥drawaxisfalse
¥begin{zahyou}(-.5,2.5)(-.5,2.5)
¥Thicklines
¥oresen{A(0,2)n;B(2,0)s}
¥thinlines
¥HenKo¥A¥B{$2+¥sqrt3$}
¥end{zahyou}
```

→



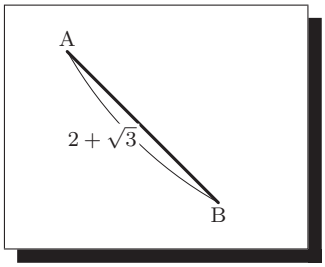
文字列 $2+\sqrt{3}$ が線分にかかっていますので，左に動かします。調整に入る前に，デフォルトの文字列配置について述べます。

基準点は円弧の中点で，文字列ボックスの中心がこの基準点にくるように配置されます。では，基準点を左に 4mm 動かしてみましょう。

平行移動

```
¥unitlength10mm¥footnotesize
¥drawaxisfalse
¥begin{zahyou}(-.5,2.5)(-.5,2.5)
¥Thicklines
¥oresen{A(0,2)n;B(2,0)s}
¥thinlines
¥HenKo<putoption={(-4mm,0)}>¥A¥B%}>%
{$2+¥sqrt3$}
¥end{zahyou}
```

→



<putoption={(-4mm,0)}>の部分が平行移動をするためのオプションで，¥Put にオプションとして引き渡されます。すなわち，円弧の中点を ¥Q として

```
¥Put¥Q(-4mm,0){$2+¥sqrt3$}
```

が実行されます。

文字列が長いときは，文字列を回転して線分と平行にする方がよいでしょうか。

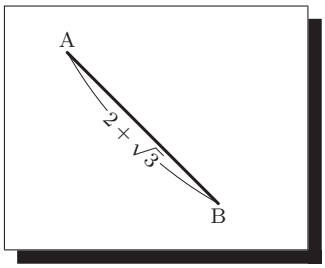
<henkomozikaiten=1> **オプション**

です。

回転

```
¥unitlength10mm¥footnotesize
¥drawaxisfalse
¥begin{zahyou}(-.5,2.5)(-.5,2.5)
¥tenretu{A(0,2)n;B(2,0)s}
¥HenKo<henkomozikaiten=1>¥A¥B%
{$2+¥sqrt3$}
{¥Thicklines¥Drawline{¥A¥B}}
¥end{zahyou}
```

→

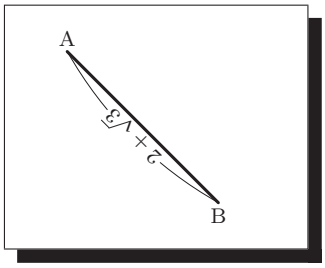


オプションの右辺値を -1 とすれば，文字は 180 度回転します。

逆回転

```
¥unitlength10mm¥footnotesize
¥drawaxisfalse
¥begin{zahyou}(-.5,2.5)(-.5,2.5)
¥tenretu{A(0,2)n;B(2,0)s}
¥HenKo<henkomozikaiten=-1>¥A¥B%
{$2+¥sqrt3$}
{¥Thicklines¥Drawline{¥A¥B}}
¥end{zahyou}
```

→



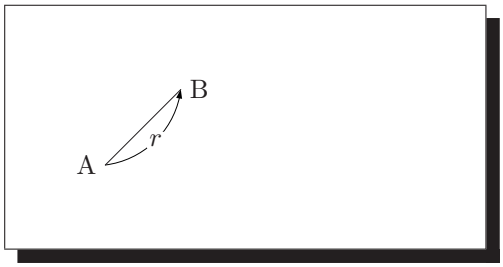
3.3.3 弧に矢印

弧の端末に矢印をつけるオプションが<yazirusi=a>オプションです。

<yazirusi=a>オプション

```
¥begin{zahyou*}(0,3)(0,3)
¥def¥A{(1,1)}%
¥def¥B{(2,2)}%
¥Put¥A{¥makebox(0,0)[r]{A }}%
¥Put¥B{¥makebox(0,0)[l]{ B}}%
¥HenKo<yazirusi=a>¥A¥B{$r$}%
¥Drawline{¥A¥B}
¥end{zahyou*}
```

→

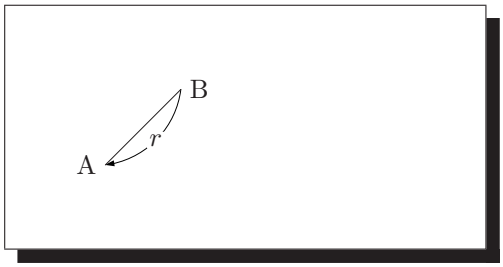


逆向きにつけるには

<yazirusi=r>オプション

```
¥begin{zahyou*}(0,3)(0,3)
¥def¥A{(1,1)}%
¥def¥B{(2,2)}%
¥Put¥A{¥makebox(0,0)[r]{A }}%
¥Put¥B{¥makebox(0,0)[l]{ B}}%
¥HenKo<yazirusi=r>¥A¥B{$r$}%
¥Drawline{¥A¥B}
¥end{zahyou*}
```

→

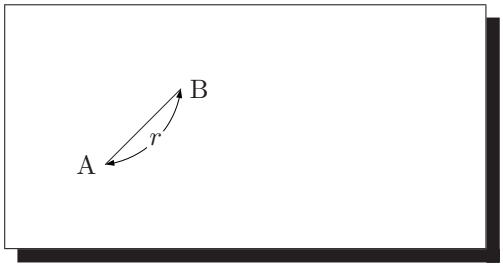


両向きにつけるには

<yazirusi=b>オプション

```
¥begin{zahyou*}(0,3)(0,3)
¥def¥A{(1,1)}%
¥def¥B{(2,2)}%
¥Put¥A{¥makebox(0,0)[r]{A }}%
¥Put¥B{¥makebox(0,0)[l]{ B}}%
¥HenKo<yazirusi=b>¥A¥B{$r$}%
¥Drawline{¥A¥B}
¥end{zahyou*}
```

→



3.3.4 ¥HenKo の中点，中心

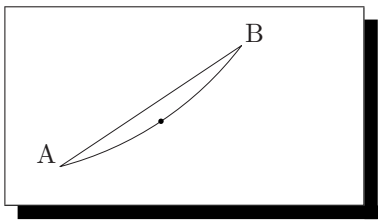
¥HenKo コマンド発行後，円弧の中点，円弧の中心位置を知りたいことがあります。

¥HenKoTyuuten　¥HenKo で描画される円弧の中点は ¥HenKoTyuuten に保存されています。

¥HenKoTyuuten

```
¥begin{zahyou*}[ul=8mm](-.5,3)(-.5,2.5)
  ¥tenretu{A(0,0)nw;B(3,2)ne}
  ¥Drawline{¥A¥B}
  ¥HenKo¥A¥B{}
  ¥Kuromaru¥HenKoTyuuten
¥end{zahyou*}
```

→

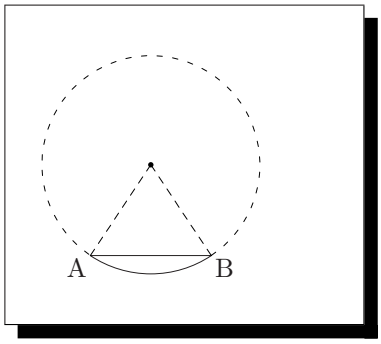


¥HenKoTyuusin　¥HenKo で描画される円弧の中心を ¥HenKoTyuusin に保存します。

¥HenKoTyuuten

```
¥begin{zahyou*}[ul=8mm](-1,4)(-1,4)
  ¥tenretu{A(0,0)sw;B(2,0)se}
  ¥Drawline{¥A¥B}
  ¥HenKo¥A¥B{}
  ¥Kuromaru¥HenKoTyuusin
  ¥Hasen{¥A¥HenKoTyuusin¥B}
  ¥Enko<hasen={[.5][.8]}>¥HenKoTyuusin%
    {tuukaten=¥A}%
    {hazimeten=¥B}{owariten=¥A}
¥end{zahyou*}
```

→

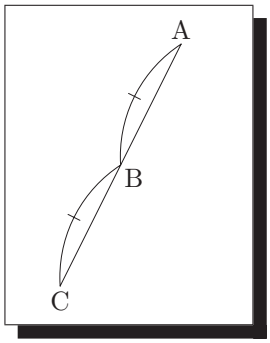


応用例　¥HenKoTyuuten, ¥HenKoTyuusin を利用した例です。

応用例

```
¥begin{zahyou*}[ul=8mm](-.5,2.5)(-.5,4.5)
  ¥tenretu{A(2,4)n;B(1,2)se;C(0,0)s}
  ¥Drawline{¥A¥C}
  ¥HenKo¥A¥B{}
  ¥Bunten¥HenKoTyuusin¥HenKoTyuuten{.95}{.05}¥P
  ¥Bunten¥HenKoTyuusin¥HenKoTyuuten{1.05}{-.05}¥Q
    ¥Drawline{¥P¥Q}
  ¥HenKo¥B¥C{}
  ¥Bunten¥HenKoTyuusin¥HenKoTyuuten{.95}{.05}¥P
  ¥Bunten¥HenKoTyuusin¥HenKoTyuuten{1.05}{-.05}¥Q
    ¥Drawline{¥P¥Q}
¥end{zahyou*}
```

→



3.3.5　¥HenKo の形状いろいろ

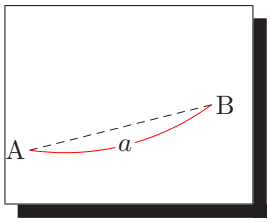
¥HenKo の<henkotype=..>オプションにより，円弧部分のバリエーションについて説明します．この節では，¥HenKo の方が焦点ですから，辺のほうは破線描画としておきます。

色つけ　円弧の部分に色をつけるには <henkocolor=...> オプションを用います。

<henkocolor=..>オプション

```
¥begin{zahyou*}[ul=6mm](0,5)(0,4)
  ¥tenretu{A(0,1)w;B(4,2)e}
  ¥Hasen{¥A¥B}
  ¥HenKo<henkocolor=red>¥A¥B{$a$}
¥end{zahyou*}
```

→



<henkotype=..>**オプション**　デフォルトでは，辺の両端を円弧で結びますが，この形状を変えるオプションが <henkotype=..>オプションです。

<henkotype=ellipse>　円弧ではなく，楕円弧にするオプションが

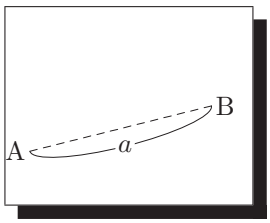
<henkotype=1> **または** <henkotype=ellipse>

です。

<henkotype=ellipse>オプション

```
¥begin{zahyou*}[ul=6mm](0,5)(0,4)
  ¥tenretu{A(0,1)w;B(4,2)e}
  ¥Hasen{¥A¥B}
  ¥HenKo<henkotype=ellipse>¥A¥B{$a$}
¥end{zahyou*}
```

→



<henkotype=triangle>　円弧ではなく，辺の両端点・文字列を配置する点を結ぶ三角形を描画するオプションが

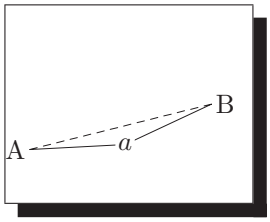
<henkotype=2> **または** <henkotype=triangle>

です。

<henkotype=triangle>オプション

```
¥begin{zahyou*}[ul=6mm](0,5)(0,4)
  ¥tenretu{A(0,1)w;B(4,2)e}
  ¥Hasen{¥A¥B}
  ¥HenKo<henkotype=triangle>¥A¥B{$a$}
¥end{zahyou*}
```

→



<henkotype=parallel>　円弧ではなく，辺と平行な線分を描画するオプションが

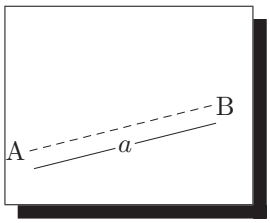
<henkotype=3> **または** <henkotype=parallel>

です。

<henkotype=parallel>オプション

```
¥begin{zahyou*}[ul=6mm](0,5)(0,4)
  ¥tenretu{A(0,1)w;B(4,2)e}
  ¥Hasen{¥A¥B}
  ¥HenKo<henkotype=parallel>¥A¥B{$a$}
¥end{zahyou*}
```

→

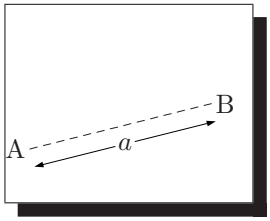


yazirusi=a/r/b **オプション**　この形状は，線分 AB の長さを表すときなどに利用できそうです。そして，¥HenKo で描画される方の線分には矢印をつけることが多いようです。

<yazirusi=b>オプション

```
¥begin{zahyou*}[ul=6mm](0,5)(0,4)
  ¥tenretu{A(0,1)w;B(4,2)e}
  ¥Hasen{¥A¥B}
  ¥HenKo<henkotype=parallel,yazirusi=b>¥A¥B{$a$}
¥end{zahyou*}
```

→



¥HenKo<yazirusi=..>¥A¥B{...}における yazirusi=.. の右辺値は

- a: **点 ¥A から点 ¥B に向かう向き**
- r: **点 ¥B から点 ¥A に向かう向き**
- b: **両向き**

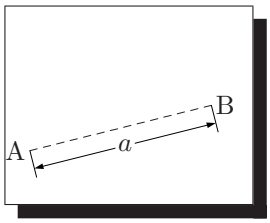
のいずれかです。

<henkosideb=..,henkosidet=..> **オプション**　この場合，さらに補助線 — 線分の両端と ¥HenKo によって引かれる並行線分の両端を結ぶ線分（若干延長して) が欲しくなるかもしれません。

<henkosideb=..,henkosidet=..>オプション

```
¥begin{zahyou*}[ul=6mm](0,5)(0,4)
  ¥tenretu{A(0,1)w;B(4,2)e}
  ¥Hasen{¥A¥B}
  ¥HenKo<henkotype=parallel,yazirusi=b,%
    henkosideb=0,henkosidet=1.5>¥A¥B{$a$}
¥end{zahyou*}
```

→



この場合，補助線は，辺の端点と ¥HenKo によって引かれる線分（矢線）の端点を結ぶ線分を

¥henkosideb : 1-¥henkosideb

と

¥henkosidet : 1-¥henkosidet

に分ける 2 点を結ぶ線分

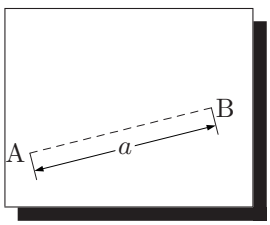
となります。

なお，<henkocolor=..>オプションによる色づけは ¥HenKo による線分（両向き矢線）に対してのみ働きます。

<henkocolor=..>オプション

```
¥begin{zahyou*}[ul=6mm](0,5)(0,4)
  ¥tenretu{A(0,1)w;B(4,2)e}
  ¥Hasen{¥A¥B}
  ¥HenKo<henkotype=parallel,yazirusi=b,%
    henkocolor=red,%
    henkosideb=0,henkosidet=1.5>¥A¥B{$a$}
¥end{zahyou*}
```

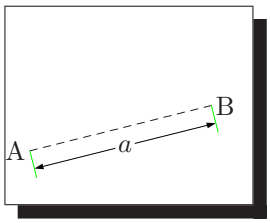
→



補助線に対する色づけは<henkosidecolor=..>オプションで行います。

<henkosidecolor=..>オプション

```
¥begin{zahyou*}[ul=6mm](0,5)(0,4)
  ¥tenretu{A(0,1)w;B(4,2)e}
  ¥Hasen{¥A¥B}
  ¥HenKo<henkotype=parallel,yazirusi=b,%
    henkocolor=red,henkosidecolor=green,%
    henkosideb=0,henkosidet=1.5>¥A¥B{$a$}
¥end{zahyou*}
```

→ 

まさかねぇ〜(^^ゞ

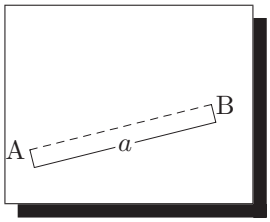
<henkotype=bracket>　円弧ではなく，大括弧にするオプションが

<henkotype=1> **または** <henkotype=bracket>

です。

<henkotype=bracket>オプション

```
¥begin{zahyou*}[ul=6mm](0,5)(0,4)
  ¥tenretu{A(0,1)w;B(4,2)e}
  ¥Hasen{¥A¥B}
  ¥HenKo<henkotype=bracket>¥A¥B{$a$}
¥end{zahyou*}
```

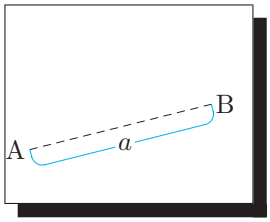
→ 

角を丸く　コーナーを丸くするオプションが<Oval=..>オプションです。右辺値は，単位を伴った長さでコーナーの四分円の半径を指定します。

<Oval=...>オプション

```
¥begin{zahyou*}[ul=6mm](0,5)(0,4)
  ¥tenretu{A(0,1)w;B(4,2)e}
  ¥Hasen{¥A¥B}

¥HenKo<henkotype=bracket,Oval=4pt,henkocolor=cyan>
    ¥A¥B{$a$}
¥end{zahyou*}
```

→ 

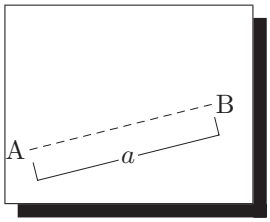
注：<henkotype=brace>はありません。後述の ¥rotUbrace コマンドをご覧ください.

<agezoko=..>**オプション**　¥HenKo の端点を辺から浮かせたいときがあります。そのためのオプションです。右辺値は無名数で単位は ¥unitlength です。単位をつけた数値で指定したいときは<Agezoko=..>オプションを用います。

<agezoko=>オプション

```
¥begin{zahyou*}[ul=6mm](0,5)(0,4)
  ¥tenretu{A(0,1)w;B(4,2)e}
  ¥Hasen{¥A¥B}
  ¥HenKo<henkotype=bracket,Agezoko=5pt>¥A¥B{$a$}
¥end{zahyou*}
```

→



<agezokovi=..,agezokovii=..>**オプション**　¥HenKo の端点をオプションの右辺値だけずらすオプションです。右辺値はベクトルで，成分は無名数—単位は ¥unitlength です。単位をつけた数値で指定したいときは<Agezokovi=..,¥Agezokovii=..>オプションを用います。末尾‘i’が始点に，‘ii’が終点に対する補正ベクトルです。

<agezokovi=>オプション

```
¥begin{zahyou*}[ul=6mm](0,5)(0,2)
  ¥tenretu{A(0,1)nw;B(4,1)ne}
  ¥Hasen{¥A¥B}
  ¥HenKo<henkotype=bracket>¥A¥B{}
  ¥HenKo<henkotype=bracket,henkocolor=red,%
    Agezokovi={(-2pt,0)},Agezokovii={(2pt,0)},%
    henkoH=3pt>¥A¥B{}
¥end{zahyou*}
```

→

3.3.6　辺に ¥brace

¥underbrace, ¥overbrace を傾いた線分に対して使用するコマンドが

¥rotUbrace, ¥rotObrace

です。

まずは，¥rotUbrace の書式から

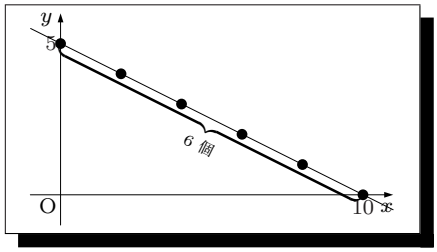
¥rotUbrace[#1]#2#3#4
点#2 **から点**#3 **へ向かう有向線分の，**
進行方向右側に ¥underbrace **をつけ，**
中央下部に文字列#4 **を配置する。**
（#4 **は数式モード内と解釈される。**--- ¥scriptstyle **）**
¥underbrace **記号と有向線分の間隔を空けたいときは**#1 **に**
depth=3pt
などと，間隔を単位付き数値で指定する。

では，簡単な使用例です。

¥rotUbrace

```
¥footnotesize
¥begin{zahyou}[ul=4mm](-1,11)(-1,6)%
¥tenretu*{A(0,5);B(10,0)}%
¥Put¥A[w]{5}¥Put¥B[s]{10}%
¥kuromaru[2pt]{¥A;(2,4);(4,3);
  (6,2);(8,1);¥B}%
¥Tyokusen¥A¥B{}{}%
¥rotUbrace¥A¥B{¥text{6 個}}%
¥end{zahyou}
```

→



すなわち点 A(0, 5) から点 B(10, 0) へ向かう線分の下に ¥underbrace をつけ，その中央部に格子点の個数を示しています。この文字列はデフォルトでは ¥scriptstyle で小さすぎますので，¥textstyle を付加しています。

¥underbrace を線分から少し離したい，という場面もあるでしょう。そのためのオプションが

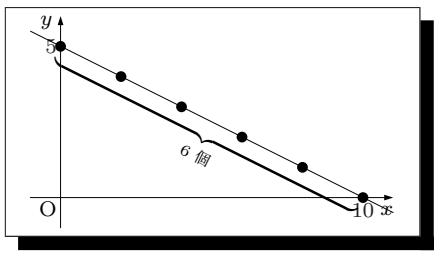
```
[depth=...]
```

です。右辺値は単位を伴った数値です。

[depth=... オプション

```
¥footnotesize
¥begin{zahyou}[ul=4mm](-1,11)(-1,6)%
¥tenretu*{A(0,5);B(10,0)}%
¥Put¥A[w]{5}¥Put¥B[s]{10}%
¥kuromaru[2pt]{¥A;(2,4);(4,3);
(6,2);(8,1);¥B}%
¥Tyokusen¥A¥B{}{}%
¥rotUbrace[depth=3pt]¥A¥B%
  {¥text{6 個}}%
¥end{zahyou}
```

→

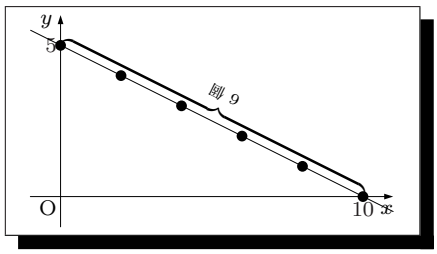


基準となる線分は有向線分として扱われます。上の例で，向きを入れ変えると

有向線分の向きを逆

```
¥footnotesize
¥begin{zahyou}[ul=4mm](-1,11)(-1,6)%
¥tenretu*{A(0,5);B(10,0)}%
¥Put¥A[w]{5}¥Put¥B[s]{10}%
¥kuromaru[2pt]{¥A;(2,4);(4,3);
(6,2);(8,1);¥B}%
¥Tyokusen¥A¥B{}{}%
¥rotUbrace¥B¥A{¥text{6 個}}%
¥end{zahyou}
```

→

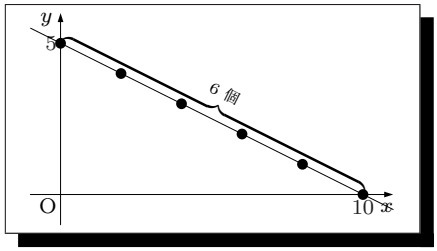


¥brace 記号が線分 AB に関して対称な位置に付きます。文字の向きがおかしいですか。これは ¥underbrace を回転させているからで，文字の向きを逆にするには，¥overbrace を用いればよいでしょう。ということで，つぎは ¥rotObrace の話しに移ります。

¥rotObrace

```
¥footnotesize
¥begin{zahyou}[ul=4mm](-1,11)(-1,6)%
¥tenretu*{A(0,5);B(10,0)}%
¥Put¥A[w]{5}¥Put¥B[s]{10}%
¥kuromaru[2pt]{¥A;(2,4);(4,3);
(6,2);(8,1);¥B}%
¥Tyokusen¥A¥B{}{}%
¥rotObrace¥A¥B{¥text{6 個}}%
¥end{zahyou}
```

→

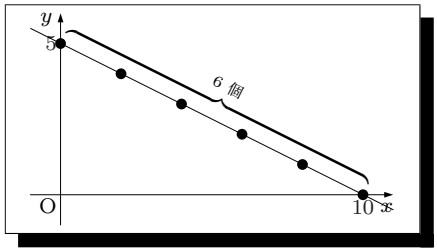


線分と ¥overbrace 記号を離すためのオプションは [height=...] です。

[height=...] **オプション**

```
¥footnotesize
¥begin{zahyou}[ul=4mm](-1,11)(-1,6)%
¥tenretu*{A(0,5);B(10,0)}%
¥Put¥A[w]{5}¥Put¥B[s]{10}%
¥kuromaru[2pt]{¥A;(2,4);(4,3);
(6,2);(8,1);¥B}%
¥Tyokusen¥A¥B{}{}%
¥rotObrace[height=3pt]¥A¥B{¥text{6 個}}%
¥end{zahyou}
```

→



¥rotObrace の書式です。

¥rotObrace[#1]#2#3#4

点#2 **から点**#3 **へ向かう有向線分の，**
進行方向左側に ¥overbrace **をつけ，**
中央上部に文字列#4 **を配置する。**
（#4 **は数式モード内と解釈される。**--- ¥scriptstyle **）**
¥overbrace **記号と有向線分の間隔を空けたいときは**#1 **に**
height=3pt
などと，間隔を単位付き数値で指定する。

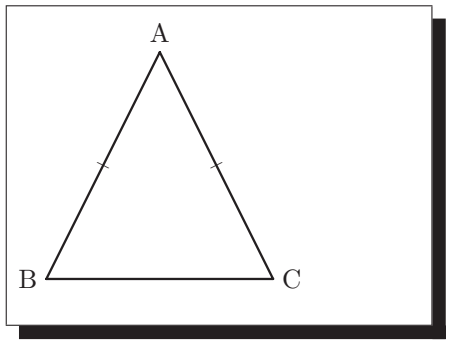
3.3.7　等辺記号

2 つの線分の長さが等しいときに，縦棒を引いたりして表現するためのコマンド ¥Touhenkigou です。

¥Touhenkigou

```
¥begin{picture}
(3.4,4)(-.2,-.5)%
¥def¥B{(0,0)}%
¥def¥C{(3,0)}%
¥def¥A{(1.5,3)}%
¥Put¥A{¥makebox(0,0.5){A}}%
¥Put¥B{¥makebox(0,0)[r]{B }}%
¥Put¥C{¥makebox(0,0)[l]{ C}}%
¥Touhenkigou¥A¥B
¥Touhenkigou¥A¥C
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥thinlines
¥end{picture}
```

→



書式です。

```
¥Touhenkigou[#1]<#2><#3>(#4)#5#6
```

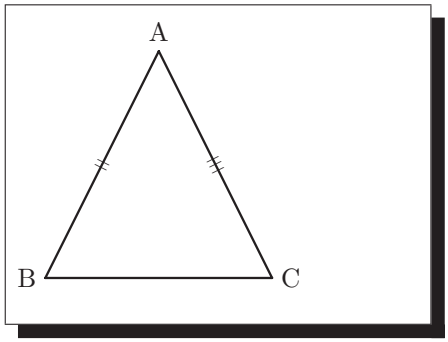
- #1 : **記号（デフォルトは | ）**
- #2 : **個数**
- #3 : #2 **で複数を指定した場合の記号間間隔（デフォルト** 0.5pt**）**
- #4 : **位置（デフォルトは** 0.5**，すなわち中点）**
- #5，#6 : **線分の両端**

辺上に置く縦棒を 2 本（3 本）にしたいときは <#2> オプションを用います。

¥Touhenkigou

```
¥begin{picture}
(3.4,4)(-.2,-.5)%
¥def¥B{(0,0)}%
¥def¥C{(3,0)}%
¥def¥A{(1.5,3)}%
¥Put¥A{¥makebox(0,0.5){A}}%
¥Put¥B{¥makebox(0,0)[r]{B }}%
¥Put¥C{¥makebox(0,0)[l]{ C}}%
¥Touhenkigou<2>¥A¥B
¥Touhenkigou<3>¥A¥C
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥thinlines
¥end{picture}
```

→

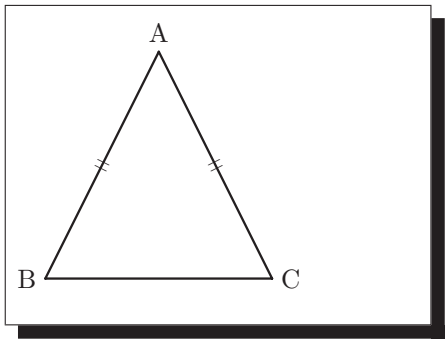


複数の線分に等辺記号をつけるコマンドが ¥touhenkigou です。線分を ';' で区切って並べます。

¥touhenkigou

```
¥begin{picture}
(3.4,4)(-.2,-.5)%
¥def¥B{(0,0)}%
¥def¥C{(3,0)}%
¥def¥A{(1.5,3)}%
¥Put¥A{¥makebox(0,0.5){A}}%
¥Put¥B{¥makebox(0,0)[r]{B }}%
¥Put¥C{¥makebox(0,0)[l]{ C}}%
¥touhenkigou<2>{¥A¥B;¥A¥C}%
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥thinlines
¥end{picture}
```

→



書式です。

¥touhenkigou[#1]<#2>#3

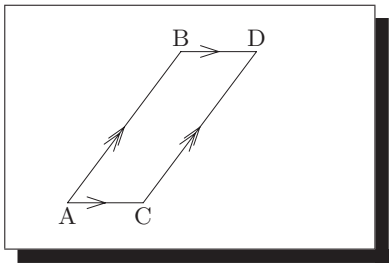
- #1 : **辺上に置く記号（デフォルトは | ）**
- #2 : **個数**
- #3 : **線分列（区切子は ';'）**

3.3.8 平行記号

¥heikoukigou

```
¥unitlength5mm¥small
¥begin{picture}(6,6)
¥tenretu{A(1,1)s;B(4,5)n;
  C(3,1)s;D(6,5)n}
¥heikoukigou{¥A¥C;¥B¥D}
¥heikoukigou[2]{¥A¥B;¥C¥D}
¥end{picture}
```

→



(1) すなわち一番簡単な使用法は

```
¥heikoukigou{¥P¥Q;¥R¥S}
```

などと，新コマンド ¥heikoukigou の引数に平行記号をつけたい線分を ';' で区切って並べます.

(2) (1) の場合，記号 '>' は 1 個だけつきますが，これを 2 個にしたければ

```
¥heikoukigou[2]{¥P¥Q;¥R¥S}
```

と，¥heikoukigou に [2] オプションをつけます.

(3) さらに記号のサイズ，間隔，位置などを調整したいときは [...] オプションに

```
key=val,key=val,....
```

の形のオプションを列記します．どの様なオプションがあるかは，次の書式をご覧ください.

平行記号
線分の中央に，記号 '>' を配置する。
¥heikoukigou[#1]#2
#1 : **記号の個数**
または key=val
heikoukigoukosuu=**記号の個数（デフォルト値は** 1**）**
heikoukigouiti =**記号の位置（デフォルト値は** 0.5**，すなわち線分の中点）**
heikoukigoukankaku=**記号を複数配置するときの間隔（デフォルト値は** 1mm**）**
heikoukigousize =**記号の大きさ（デフォルト値は** 2**）**
#2 : **線分の列を ';' 区切りで列記する.**

3.4 角の内部に記号

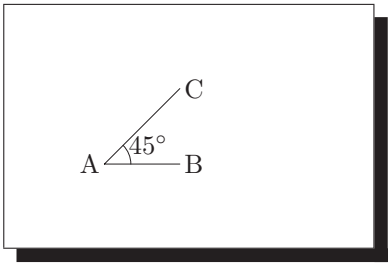
3.4.1 ¥Kakukigou

角の内部に円弧などを描き，角の大きさなどを表示させるコマンド
¥Kakukigou です。

¥Kakukigou

```
¥begin{picture}(3,3)%
¥tenretu{A(1,1)w;B(2,1)e;C(2,2)e}%
¥Kakukigou¥B¥A¥C{45¥Deg}%
¥Drawline{¥C¥A¥B}%
¥end{picture}
```

→



¥Kakukigou の基本的な使用法は

角を構成する 3 点（真ん中が角の頂点）と角内に置く記号（文字列）

の 4 つの引数を与えます。書式は

角の内部に円弧を描き，その傍に記号などを置く。

¥Kakukigou[#1]<#2>#3#4#5<#6><#7>

- #1 : **円弧の上に置く記号**
  - #1=a **のときは角記号に矢印をつける。**
  - #1=r **のときは角記号に逆向きの矢印をつける。**
  - #1=b **のときは角記号に両向きの矢印をつける。**
  - #1=| **のときは，円弧中央に円弧と垂直な短い線分**
    注：| は，アルファベット小文字のエル（l）ではなく，縦棒（|）です。
- #2 : **円弧の個数**
- #3#4#5 : **角** BAC
  - #4 **が角の頂点**
  - **半直線** #4#3 **を 回転して** #4#5 **に重ねる回転を表示する。**
- #6 : **円弧と頂点の距離係数（デフォルト値** 1 **）**
  - hankei=**（無名数） 半径の指定**
  - Hankei=**（単位付）** 〃
  - siteiten= **円弧が指定点を通過**
  - hasen=[**破線の長さ**][**破線の間隔**] **円弧を破線で**
- #7 : **円弧の中心と角の頂点位置（デフォルト値** 0 **）**
- **以下，**¥emathPut **に続く。**

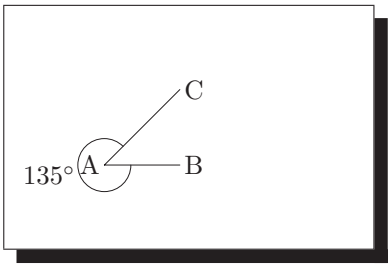
¥Kakukigou* : **円弧の内部を塗りつぶします。**

#3 と #5 を入れ替えると，

角の向き

```
¥begin{picture}(3,3)%
¥tenretu{A(1,1)w;B(2,1)e;C(2,2)e}%
¥Kakukigou¥C¥A¥B[w]{135¥Deg}%
¥Drawline{¥C¥A¥B}%
¥end{picture}
```

→



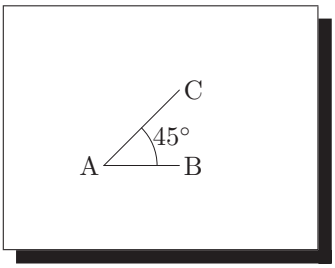
すなわち，半直線 #4#3 を正の向きに回転して #4#5 に重ねる回転を表示します。

円弧の位置を調整するには <#6> オプションを用います。デフォルトを 1 として，1 より大きくすれば頂点から離れ，1 より小さくすると頂点に近づきます。

円弧の位置

```
¥begin{picture}(3,3)%
¥tenretu{A(1,1)w;B(2,1)e;C(2,2)e}%
¥Kakukigou¥B¥A¥C<2>{45¥Deg}%
¥Drawline{¥C¥A¥B}%
¥end{picture}
```

→

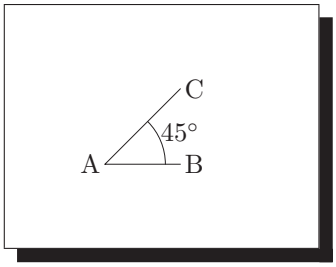


円弧の半径を直接指定することも可能です。

円弧の半径指定

```
¥begin{picture}(3,3)%
¥tenretu{A(1,1)w;B(2,1)e;C(2,2)e}%
¥Kakukigou¥B¥A¥C<hankei=.8>{45¥Deg}%
¥Drawline{¥C¥A¥B}%
¥end{picture}
```

→

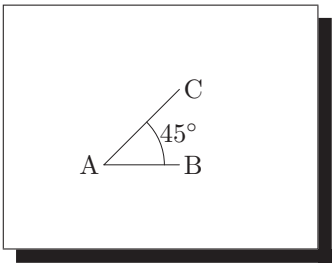


hankei=の右辺値の単位は ¥unitlength です。これを ¥unitlength に依存しない値で指定するには，Hankei=とします。

円弧の半径指定（単位付）

```
¥begin{picture}(3,3)%
¥tenretu{A(1,1)w;B(2,1)e;C(2,2)e}%
¥Kakukigou¥B¥A¥C<Hankei=8mm>{45¥Deg}%
¥Drawline{¥C¥A¥B}%
¥end{picture}
```

→

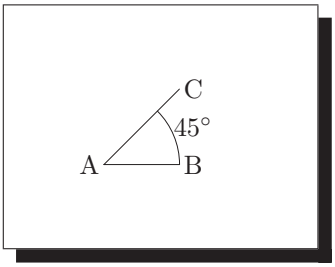


円弧が通過すべき一点を指定する方法もあります。

円弧の通過点指定

```
¥begin{picture}(3,3)%
¥tenretu{A(1,1)w;B(2,1)e;C(2,2)e}%
¥Kakukigou¥B¥A¥C<siteiten=¥B>{45¥Deg}%
¥Drawline{¥C¥A¥B}%
¥end{picture}
```

→

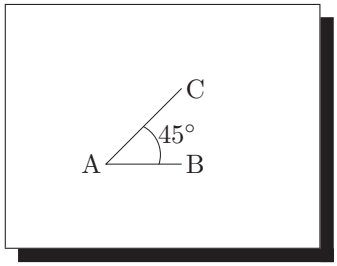


円弧の曲がり具合は，<#7> オプションで調整します。デフォルトは 0 ですが，1 に近づけると半径が小さくなっていきます。

¥Kakukigou

```
¥begin{picture}(3,3)%
¥tenretu{A(1,1)w;B(2,1)e;C(2,2)e}%
¥Kakukigou¥B¥A¥C<2><.5>{45¥Deg}%
¥Drawline{¥C¥A¥B}%
¥end{picture}
```

→

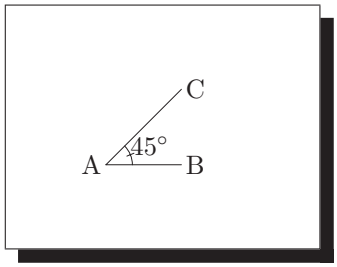


円弧の中央部に円弧に垂直な縦線を 1 本入れる表示法もあります。[|] オプションです。

円弧の中心に縦棒

```
¥begin{picture}(3,3)%
¥tenretu{A(1,1)w;B(2,1)e;C(2,2)e}%
¥Kakukigou[|]¥B¥A¥C{45¥Deg}%
¥Drawline{¥C¥A¥B}%
¥end{picture}
```

→

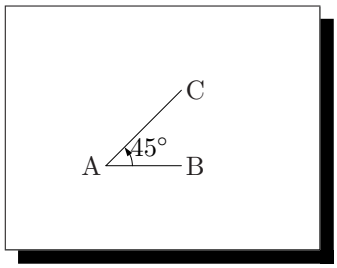


円弧に矢印をつけるには ¥Kakukigou に [a] オプションをつけます。

円弧に矢印

```
¥begin{picture}(3,3)%
¥tenretu{A(1,1)w;B(2,1)e;C(2,2)e}%
¥Kakukigou[a]¥B¥A¥C{45¥Deg}%
¥Drawline{¥C¥A¥B}%
¥end{picture}
```

→

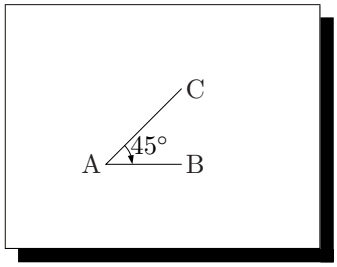


[a] オプションでは，矢印は正の回転方向につきます。これを逆向きにつけたいときは，[a] オプションに代えて [r] オプションをつけます。

逆向きに矢印

```
¥begin{picture}(3,3)%
¥tenretu{A(1,1)w;B(2,1)e;C(2,2)e}%
¥Kakukigou[r]¥B¥A¥C{45¥Deg}%
¥Drawline{¥C¥A¥B}%
¥end{picture}
```

→

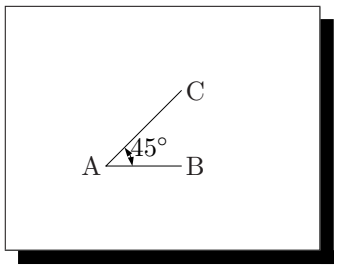


[b] オプションをつけると，両向きに矢印がつきます。

両向きに矢印

```
¥begin{picture}(3,3)%
¥tenretu{A(1,1)w;B(2,1)e;C(2,2)e}%
¥Kakukigou[b]¥B¥A¥C{45¥Deg}%
¥Drawline{¥C¥A¥B}%
¥end{picture}
```

→

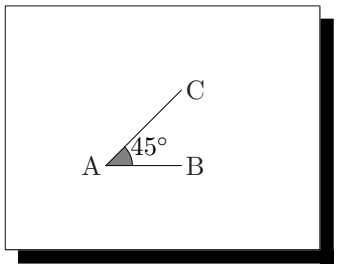


円弧の内部を塗りつぶすには，¥Kakukigou*コマンドを用います。

円弧の内部を塗りつぶす

```
¥begin{picture}(3,3)%
¥tenretu{A(1,1)w;B(2,1)e;C(2,2)e}%
¥Kakukigou*¥B¥A¥C{45¥Deg}%
¥Drawline{¥C¥A¥B}%
¥end{picture}
```

→

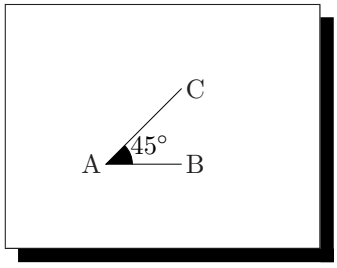


塗りつぶしの濃度を指定するには [...] オプションを用います。0（白）と 1（真黒）の間の数を指定します。

円弧の内部を黒で塗りつぶす

```
¥begin{picture}(3,3)%
¥tenretu{A(1,1)w;B(2,1)e;C(2,2)e}%
¥Kakukigou*[1]¥B¥A¥C{45¥Deg}%
¥Drawline{¥C¥A¥B}%
¥end{picture}
```

→

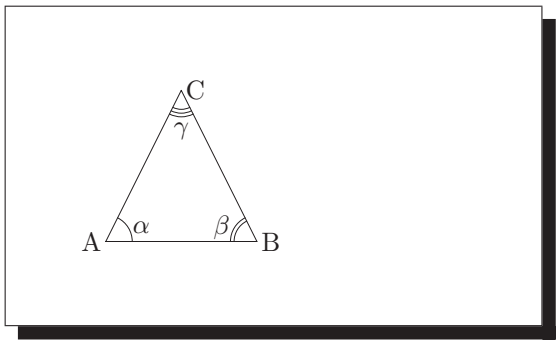


複数の角を区別するため，角内の円弧を 2 重，3 重にすることができます。<..>オプションです。

円弧を二重に

```
¥begin{picture}(4,4)%
¥tenretu{A(1,1)w;B(3,1)e;C(2,3)e}%
¥Kakukigou¥B¥A¥C[e]{$¥alpha$}%
¥Kakukigou<2>¥C¥B¥A[w]{$¥beta$}%
¥Kakukigou<3>¥A¥C¥B[s]{$¥gamma$}%
¥Drawline{¥C¥A¥B¥C}%
¥end{picture}
```

→

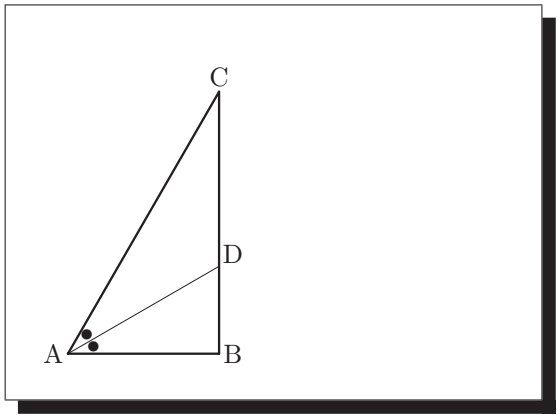


逆に <..> オプションに <0> と指定すれば，角内に円弧は描かず，記号類だけを置くこともできます。

円弧なし

```
¥begin{picture}(4,5)(-0.5,-0.5)%
¥tenretu{A(0,0)w;B(2,0)e;C(2,3.464)n}%
¥Bunten¥B¥C{1}{2}¥D%
¥Put¥D[ne]{D}%
¥Kakukigou<0>¥B¥A¥D(0,0)[c]{$¥bullet$}%
¥Kakukigou<0>¥D¥A¥C(0,0)[c]{$¥bullet$}%
¥Drawline{¥A¥D}%
{¥thicklines
¥Drawline{¥C¥A¥B¥C}}%
¥end{picture}
```

$\longrightarrow$

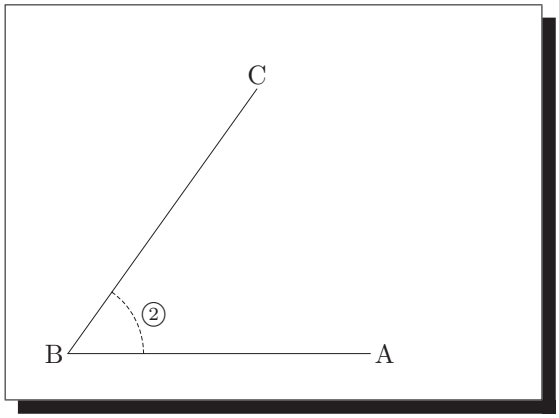


円弧を破線で描画することも可能です。

破線の円弧

```
¥begin{picture}(5,5)%
  ¥tenretu{A(4.5,0.5)e;B(0.5,0.5)w;C(3,4)n}
  ¥Drawline{¥A¥B¥C}
  ¥Kakukigou¥A¥B¥C<hasen=[.5][.5],Hankei=1cm>(%
    2pt,2pt)[l]{¥maru2}
¥end{picture}
```

$\longrightarrow$

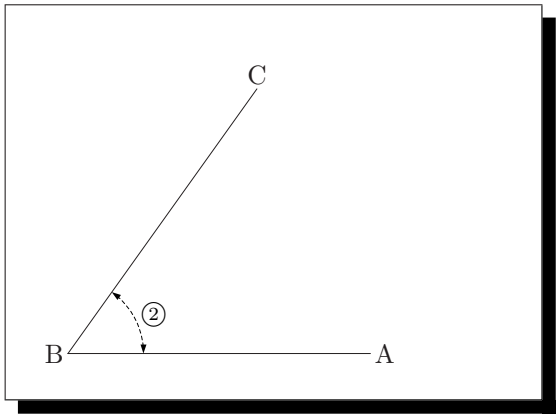


さらに矢印を付けることも可能です。

破線の円弧に矢印

```
¥begin{picture}(5,5)%
  ¥tenretu{A(4.5,0.5)e;B(0.5,0.5)w;C(3,4)n}
  ¥Drawline{¥A¥B¥C}
  ¥Kakukigou[b]¥A¥B¥C<hasen=[.5][.5],Hankei=1cm>(%
    2pt,2pt)[l]{¥maru2}
¥end{picture}
```

→

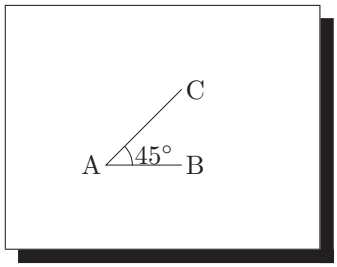


最後に，角内に置く文字列の位置調整です。これは，¥Put の文字列位置調整オプションと同じです。

¥Kakukigou

```
¥begin{picture}(3,3)%
¥tenretu{A(1,1)w;B(2,1)e;C(2,2)e}%
¥Kakukigou¥B¥A¥C[e]{45¥Deg}%
¥Drawline{¥C¥A¥B}%
¥end{picture}
```

→

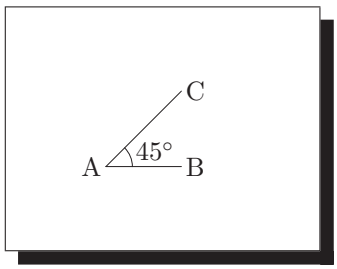


と簡易指定オプション [e] も使えますが，角内は狭いですから

¥Kakukigou

```
¥begin{picture}(3,3)%
¥tenretu{A(1,1)w;B(2,1)e;C(2,2)e}%
¥Kakukigou¥B¥A¥C(2pt,2pt)[l]{45¥Deg}%
¥Drawline{¥C¥A¥B}%
¥end{picture}
```

→



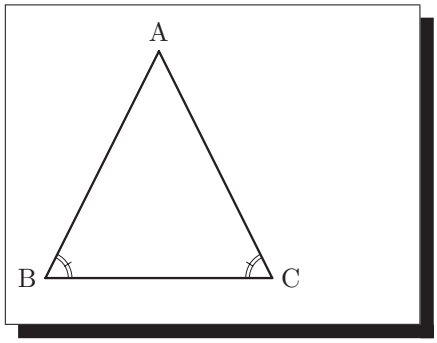
などと細かく指定した方が良いでしょう。蛇足ながら文字列を配置する基準点は，円弧の中点です。

等しい角に同じ記号をつけるためのコマンドが ¥toukakukigou です。角を ';' で区切って並べるというのは，¥touhenkigou と同様です。

¥toukakukigou

```
¥begin{picture}
(3.4,4)(-.2,-.5)%
¥def¥B{(0,0)}%
¥def¥C{(3,0)}%
¥def¥A{(1.5,3)}%
¥Put¥A{¥makebox(0,0.5){A}}%
¥Put¥B{¥makebox(0,0)[r]{B }}%
¥Put¥C{¥makebox(0,0)[l]{ C}}%
¥toukakukigou<2>{¥C¥B¥A;¥A¥C¥B}%
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥thinlines
¥end{picture}
```

→

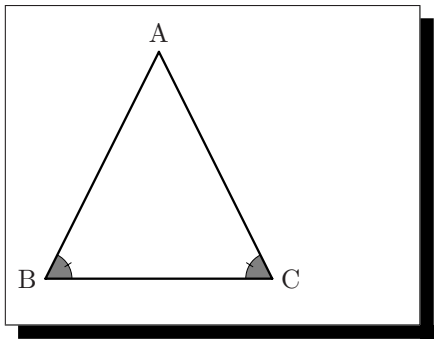


¥toukakukigou*と‘∗’をつけると角内を塗りつぶします。

¥toukakukigou*

```
¥begin{picture}
(3.4,4)(-.2,-.5)%
¥def¥B{(0,0)}%
¥def¥C{(3,0)}%
¥def¥A{(1.5,3)}%
¥Put¥A{¥makebox(0,0.5){A}}%
¥Put¥B{¥makebox(0,0)[r]{B }}%
¥Put¥C{¥makebox(0,0)[l]{ C}}%
¥toukakukigou*{¥C¥B¥A;¥A¥C¥B}%
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥thinlines
¥end{picture}
```

→



書式です。

複数の角に角記号をつける。

¥toukakukigou[#1]<#2>#3

¥toukakukigou*[#1]<#2>#3

#1 : **円弧の上に置く記号**

[|] **のときは，円弧中央に円弧と垂直な短い線分（デフォルト）**

注：| **は，アルファベット小文字のエル**（l）**ではなく，縦棒**（|）**です。**

[] **のときは，なにもつけない（円弧のみ）**

*付**のときは，塗りつぶしの濃度**（0〜1）

#2 : **円弧の個数**

#3 : **角の列 区切り子は**‘;’

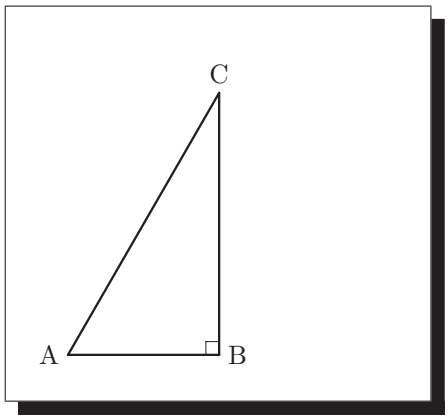
3.4.2 直角記号

特に直角を表す記号です。

<..>オプション

```
¥begin{picture}%
(3,5)(-0.5,-0.5)%
¥def¥A{(0,0)}%
¥def¥B{(2,0)}%
¥def¥C{(2,3.464)}%
¥Bunten¥B¥C{1}{2}¥D%
¥Put¥A{¥makebox(0,0)[r]{A }}%
¥Put¥B{¥makebox(0,0)[l]{ B}}%
¥Put¥C{¥makebox(0,0.5){C}}%
¥Tyokkakukigou¥A¥B¥C
¥thicklines
¥Drawline{¥C¥A¥B¥C}%
¥thinlines
¥end{picture}
```

→



書式は

¥Tyokkakukigou[#1](#2)#3#4#5

#1 : **直角記号内を塗りつぶすときの濃さ**

#2 : **直角記号のサイズ**

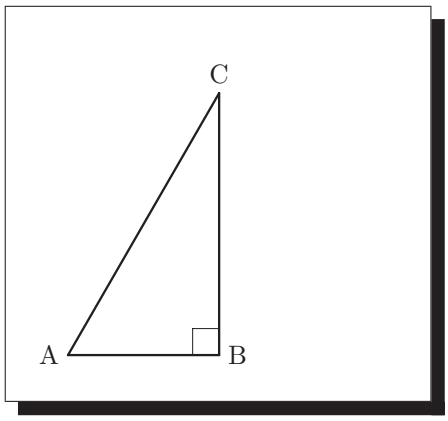
#3#4#5 : **直角**（#4 **が頂点**）

直角記号のサイズは (#2) で指定できます。デフォルトは 5(pt) です。10pt にしたいときは, (10) と指定します。

<..>オプション

```
¥begin{picture}%
(3,5)(-0.5,-0.5)%
¥def¥A{(0,0)}%
¥def¥B{(2,0)}%
¥def¥C{(2,3.464)}%
¥Bunten¥B¥C{1}{2}¥D%
¥Put¥A{¥makebox(0,0)[r]{A }}%
¥Put¥B{¥makebox(0,0)[l]{ B}}%
¥Put¥C{¥makebox(0,0.5){C}}%
¥Tyokkakukigou(10)¥A¥B¥C
¥thicklines
¥Drawline{¥C¥A¥B¥C}%
¥thinlines
¥end{picture}
```

⟶

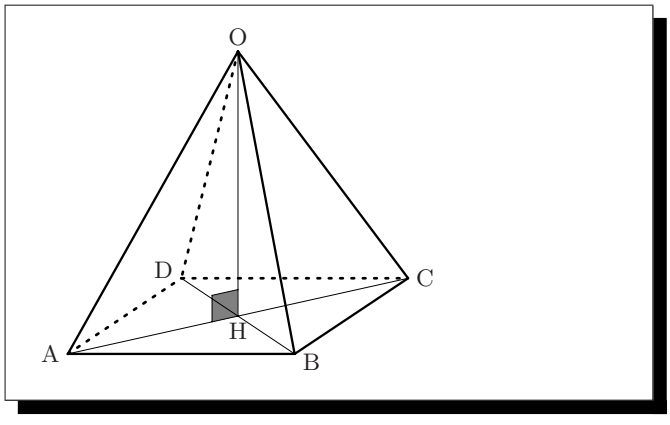


空間図形など複雑な図形で，直角記号を塗りつぶしたいときは，[#1] オプションに塗りつぶしの濃度 (0〜1) を指定します。

<..>オプション

```
{¥unitlength5mm¥small
¥begin{picture}(11,10)(-1,-1)
¥def¥takasa{7}%
¥def¥A{(0,0)}¥def¥B{(6,0)}¥def¥D{(3,2)}%
¥Addvec¥B¥D¥C
¥Bunten¥A¥C{1}{1}¥H
¥Addvec¥H{(0,¥takasa)}¥O
¥Put¥O{¥makebox(0,0.8){O}}%
¥Put¥B{¥makebox(0,0)[lt]{ B}}%
¥Put¥A{¥makebox(0,0)[r]{A }}%
¥Put¥D{¥makebox(0,0)[rb]{D }}%
¥Put¥C{¥makebox(0,0)[l]{ C}}%
¥Put¥H{¥makebox(0,-0.8){H}}%
%¥Tyokkakukigou[.5]¥O¥H¥A%     直角記号塗りつぶし [.5] オプション
¥Tyokkakukigou[.5](10)¥O¥H¥A% (10) はサイズオプション
¥Drawline{¥A¥C}%
¥Drawline{¥B¥D}%
¥Drawline{¥O¥H}%
¥thicklines
¥Drawline{¥O¥A¥B¥C¥O¥B}%
¥Dottedline{0.3}{¥A¥D¥C}%
¥Dottedline{0.3}{¥O¥D}%
¥thinlines
¥end{picture}%
}
```

→



複数の角に直角記号をつけるコマンド ¥tyokkakukigou もあります。

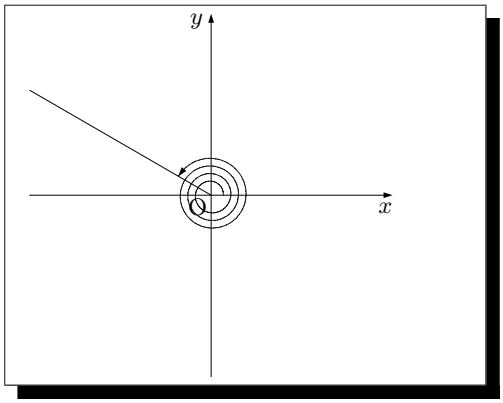
3.4.3 一般角

360° を越える角に角記号をつけるコマンド ¥ippankaku の基本的な使用法は，引数に角度を与えるだけです。

¥ippankaku

```
¥unitlength8mm¥small
¥begin{zahyou}(-3,3)(-3,3)%
¥ippankaku{1230}%
¥kHantyokusen{(0,0)}{1230}%
¥end{zahyou}
```

→

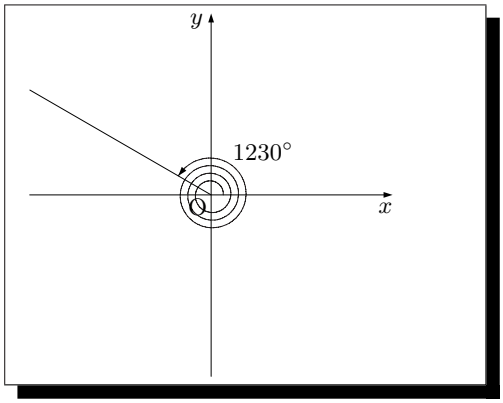


¥rasenP　この記号の適当なところに文字列を配置するために，角度を指定して螺旋上の点を求めるコマンド ¥rasenP を用意しました。¥emathPut と併用して文字列を配置します。

¥rasenP

```
¥unitlength8mm¥small
¥begin{zahyou}(-3,3)(-3,3)%
¥ippankaku{1230}%
¥rasenP{1140}¥P
¥kHantyokusen{(0,0)}{1230}%
¥Put¥P[ne]{1230¥Deg}%
¥end{zahyou}
```

→



螺旋の形状変更　螺旋 $r = a\theta + b$ の係数 a, b のデフォルト値は $a = 0.02$, $b = 0.2$ としてあります。これを変更するためのオプションが

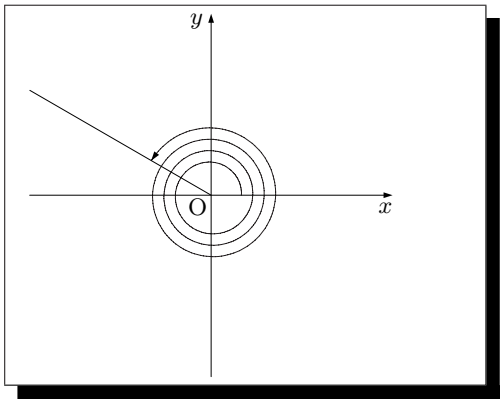
```
¥ippankaku<#1><#2>
#1 : a を #1 倍する。
#2 : b を #2 倍する。
```

その効果は

<...>オプション

```
¥unitlength8mm¥small
¥begin{zahyou}(-3,3)(-3,3)%
¥ippankaku<1.5><2.5>{1230}%
¥kHantyokusen{(0,0)}{1230}%
¥end{zahyou}
```

→

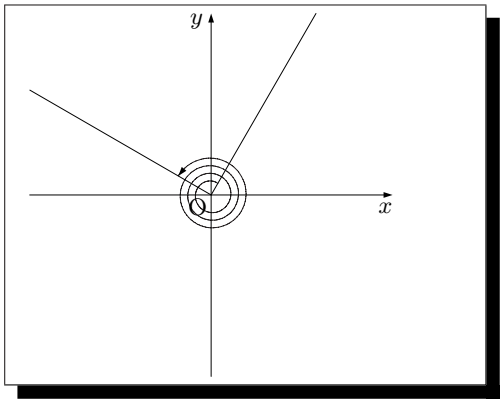


始線変更オプション　始線の位置を変更するには [...] オプションを用います。

[...] オプション

```
¥unitlength8mm¥small
¥begin{zahyou}(-3,3)(-3,3)%
¥ippankaku[60]{1230}%
¥def¥O{(0,0)}%
¥kHantyokusen{(0,0)}{60}%
¥kHantyokusen{(0,0)}{1230}%
¥end{zahyou}
```

→

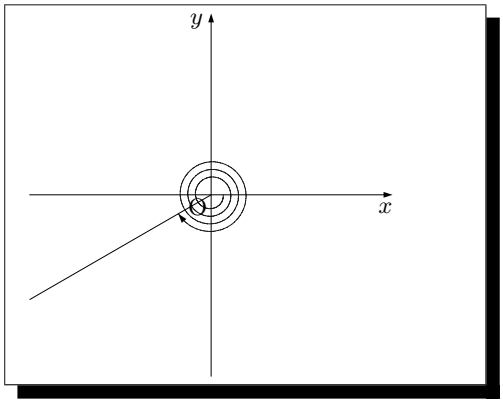


角が負の場合の処理　角が負の場合は，内部処理で $r = a\theta + b$ の a の符号を逆転させています。

負の角

```
¥unitlength8mm¥small
¥begin{zahyou}(-3,3)(-3,3)%
¥ippankaku{-1230}%
¥kHantyokusen{(0,0)}{-1230}%
¥end{zahyou}
```

→

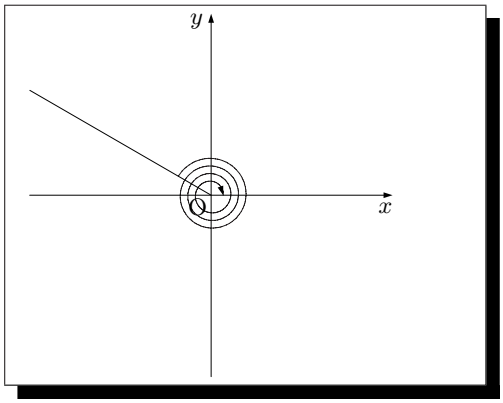


矢印の制御　矢印の制御は（#4）オプションで行います。デフォルトは終端に矢印をつけます。これを逆に始端につけたいときは（s）オプションをつけます。

(s) オプション

```
¥unitlength8mm¥small
¥begin{zahyou}(-3,3)(-3,3)%
¥ippankaku(s){1230}%
¥kHantyokusen{(0,0)}{1230}%
¥end{zahyou}
```

→

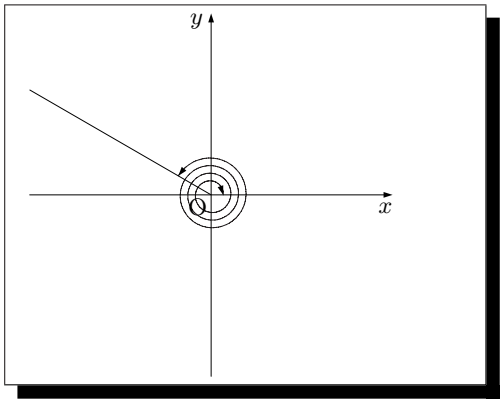


両端につけるには (b) オプションを用います。

(b) オプション

```
¥unitlength8mm¥small
¥begin{zahyou}(-3,3)(-3,3)%
¥ippankaku(b){1230}%
¥kHantyokusen{(0,0)}{1230}%
¥end{zahyou}
```

→

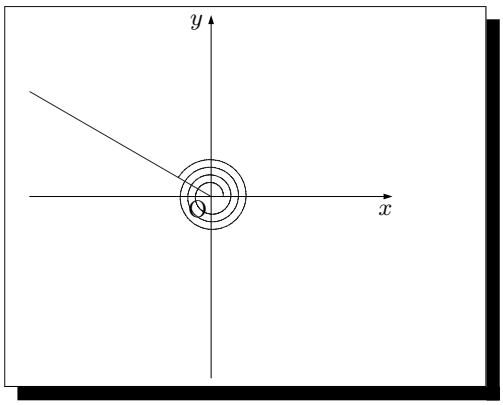


矢印をつけたくないときは (n) オプションです。

(n) オプション

```
¥unitlength8mm¥small
¥begin{zahyou}(-3,3)(-3,3)%
¥ippankaku(n){1230}%
¥kHantyokusen{(0,0)}{1230}%
¥end{zahyou}
```

→

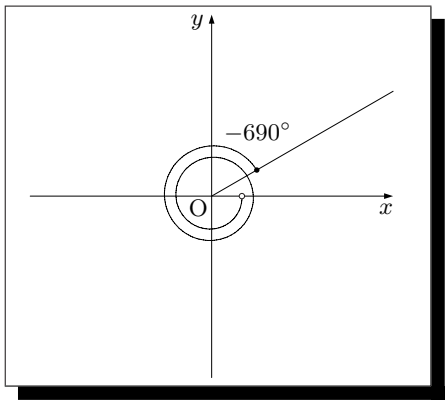


両端に ¥Kuromaru, ¥Siromaru　両端に黒丸, 白丸を配置したいときは，らせん上の 1 点を求める ¥rasenP コマンドを用います。

$-690^\circ \leqq \theta < 0^\circ$

```
¥small
¥begin{zahyou*}[ul=8mm](-3,3)(-3,3)%
¥drawXYaxis
¥ippankaku<1.5><2.5>(n){-690}%
¥kHantyokusen{(0,0)}{-690}%
¥rasenP{0}¥Start¥Siromaru¥Start
¥rasenP{-690}¥End¥Kuromaru¥End
¥rasenP{-640}¥P
¥Put¥P[ne]{$-690¥Deg$}%
¥end{zahyou*}
```

→



¥ippankaku の書式　¥ippankaku の書式です。

```
¥ippankaku<#1><#2>[#3](#4)#5
        #1 : a の倍率
        #2 : b の倍率
        #3 : 始め角（六十分法）
        #4 : 矢印の配置
          e = 終端（デフォルト）
          s = 始端
          b = 両端
          n = なし
        #5 : 終り角（六十分法）
```

3.5　数式に picture 環境を併置

3.5.1　sikipicture 環境

数式に picture 環境を併設し, 数式を飾り立てようというのが sikipicture 環境です。これは，実質 zahyou*環境です。したがって，その環境内には，zahyou*環境内に記述できるものはすべて記述可能です。

sikipicture 環境

```
a¥begin{sikipicture}{%
  x^2,
  +,
  2,
  x,
  +,
  ¥protect¥bunsuu12}
¥Takakkei{¥LT¥LB¥RB¥RT}
¥put(0,0){¥color{red}¥Kuromaru¥0}%
¥end{sikipicture}z
```

→ a$x^2 + 2x + \frac{1}{2}$z

座標領域を確認するため，四隅を結ぶ四角形と座標原点に赤丸印を描画しておきました。

注 1. ¥begin{sikipicture}{...}の引数内に ¥bunsuu などのコマンドを入れるときは ¥protect をかぶせておく必要があります。

注 2. TEX が認知するサイズは上図の四角形です。¥put を用いてこの四角形の外部に文字列を配置することは可能ですが，TEX はその存在を認知しませんから，前後の文章とかぶります。¥vspace{...}などで調整する必要があります。

注 3. sikipicture 環境の ¥unitlength は，1pt（固定）です。

注 4. sikixypos も同義語として使えます。

3.5.2 ¥sikiBi, ¥sikiTi

¥begin{sikipicture}{...}の引数内で，分割記述された各項の中央下部の点の座標が

¥sikiBi, ¥sikiBii, ¥sikiBiii, ...

に定義されています。そこに赤丸を打ってみましょう。

¥sikiBi,...

```
a¥begin{sikipicture}{%
  x^2,
  +,
  2,
  x,
  +,
  ¥protect¥bunsuu12}
¥put(0,0){¥color{red}¥kuromaru{%
  ¥sikiBi;¥sikiBii;¥sikiBiii;%
  ¥sikiBiv;¥sikiBv;¥sikiBvi}}%
¥end{sikipicture}z
```

→ a$x^2 + 2x + \frac{1}{2}$z

＋の下の赤丸がずれていますね。これは，二項演算子，関係演算子には両側にアキが入ることによるずれです。正しい位置が欲しければ，¥begin{sikipicture}{...}の引数内の演算子両側に{}を補ってやります。

二項演算子に対する補正

```
a¥begin{sikipicture}{%
  x^2,
  {}+{},
  2,
  x,
  {}+{},
  ¥protect¥bunsuu12}
¥put(0,0){¥color{red}¥kuromaru{%
  ¥sikiBi;¥sikiBii;¥sikiBiii;%
  ¥sikiBiv;¥sikiBv;¥sikiBvi}}%
¥end{sikipicture}z
```

→ $a x^2 + 2x + \frac{1}{2}z$

x^2 の下もずれてるって ？これはずれていません。x^2 の中央下です。底の x の中央下が欲しいなら，x とその冪を分離して与えます。

ついでに，各項の中央上にも緑丸をつけてみます。(¥sikiTi，¥sikiTii，.....)

二項演算子に対する補正

```
a¥begin{sikipicture}{%
  x,
  {}^2,
  {}+{},
  2,
  x,
  {}+{},
  ¥protect¥bunsuu12}
¥put(0,0){¥color{red}¥kuromaru{%
  ¥sikiBi;¥sikiBii;¥sikiBiii;%
  ¥sikiBiv;¥sikiBv;¥sikiBvi;%
  ¥sikiBvii}}%
¥put(0,0){¥color{green}¥kuromaru{%
  ¥sikiTi;¥sikiTii;¥sikiTiii;%
  ¥sikiTiv;¥sikiTv;¥sikiTvi;%
  ¥sikiTvii}}%
¥end{sikipicture}z
```

→ $a x^2 + 2x + \frac{1}{2}z$

3.5.3 ¥sikixposi, ¥sikiyhposi, ¥sikiydposi

なお，丸印をつけた点の x 座標は

¥sikixposi, ¥sikixposii, ...

です。また y 座標は，赤丸のほうが

¥sikiydposi, ¥sikiydposii, ...

緑丸のほうが

¥sikiyhposi, ¥sikiyhposii, ...

となっています。

¥sikixpos..

```
a¥begin{sikipicture}{%
  x,
  {}^2,
  {}+{},
  2,
  x,
  {}+{},
  ¥protect¥bunsuu12}
¥Takakkei{¥LT¥LB¥RB¥RT}
¥put(0,0){¥color{red}¥kuromaru{%
  (¥sikixposi,¥ymin);%
  (¥sikixposii,¥ymin);%
  (¥sikixposiii,¥ymin)}}%
¥put(0,0){¥color{green}¥kuromaru{%
  (¥sikixposi,¥ymax);%
  (¥sikixposii,¥ymax);%
  (¥sikixposiii,¥ymax)}}%
¥end{sikipicture}z
```

→ $\mathrm{a}x^2 + 2x + \frac{1}{2}\mathrm{z}$

3.5.4 ¥sikixlposi, ¥sikixrposi

¥sikixposi などは各項の中央 x 座標を表しますが，各項・左右端の x 座標はそれぞれ

¥sikixlposi, ¥sikilposii, ...
¥sikixrposi, ¥sikirposii, ...

などです。

¥sikixlpos.., ¥sikixrpos..

```
a¥begin{sikipicture}{%
  x,
  {}^2,
  {}+{},
  2,
  x,
  {}+{},
  ¥protect¥bunsuu12}
¥Takakkei{¥LT¥LB¥RB¥RT}
¥put(0,0){¥color{red}¥kuromaru{%
  (¥sikixlposiv,¥sikiyhposiv);%
  (¥sikixrposiv,¥sikiyhposiv)}}%
¥put(0,0){¥color{green}¥kuromaru{%
  (¥sikixposiv,¥sikiydposiv)}}%
¥end{sikipicture}z
```

→ $a\boxed{x^2 + \frac{1}{2}x + \frac{1}{2}}z$

3.5.5 使用例

式展開の説明図　式を展開するときの流れを説明する図です.

式展開の説明図

```
¥vskip 24pt

¥begin{sikipicture}{(,a,+,b,),¥kern1em(,x,+,y,)=,ax,+,ay,+,bx,+,by}
  ¥HenKo{(¥sikixposvii,¥ymax)}{(¥sikixposii,¥ymax)}{¥maru1}
  ¥HenKo<henkoH=15pt>{(¥sikixposix,¥ymax)}{(¥sikixposii,¥ymax)}{¥maru2}
  ¥HenKo{(¥sikixposiv,¥ymin)}{(¥sikixposvii,¥ymin)}{¥maru3}
  ¥HenKo<henkoH=15pt>{(¥sikixposiv,¥ymin)}{(¥sikixposix,¥ymin)}{¥maru4}
  ¥Put{(¥sikixposxi,¥ymax)}(0,0)[b]{¥maru1}
  ¥Put{(¥sikixposxiii,¥ymax)}(0,0)[b]{¥maru2}
  ¥Put{(¥sikixposxv,¥ymin)}(0,0)[t]{¥maru3}
  ¥Put{(¥sikixposxvii,¥ymin)}(0,0)[t]{¥maru4}
¥end{sikipicture}
¥vskip 10pt

¥mbox{}
```

→ $(a+b)\,(x+y) = \overset{①}{a}x + \overset{②}{a}y + \underset{③}{b}x + \underset{④}{b}y$

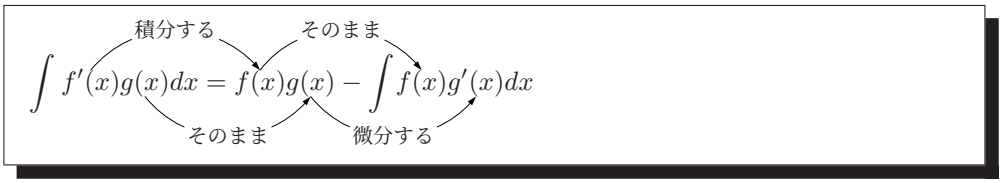
部分積分の流れ　部分積分の流れを示すのにも使えます.

不定積分の流れ

```
¥vskip15pt

¥begin{sikipicture}{%
  ¥dint{}{},f'(x),g(x),dx=,f(x),g(x),-,¥dint{}{},f(x),g'(x),dx
}
  ¥HenKo<henkoH=15pt,yazirusi=r>¥sikiTv¥sikiTii{¥footnotesize 積分する}
  ¥HenKo<henkoH=15pt,yazirusi=r>¥sikiTix¥sikiTv{¥footnotesize そのまま}
  ¥HenKo<henkoH=15pt,yazirusi=a>¥sikiBiii¥sikiBvi{¥footnotesize そのまま}
  ¥HenKo<henkoH=15pt,yazirusi=a>¥sikiBvi¥sikiBx{¥footnotesize 微分する}
¥end{sikipicture}
¥vskip10pt
¥mbox{}
```

→



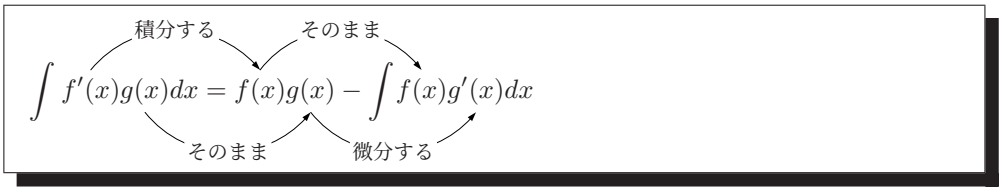
¥HenKo による円弧を式から少し離したいときは，¥HenKo に対し<agezoko=..>オプションをつけます。

不定積分の流れ

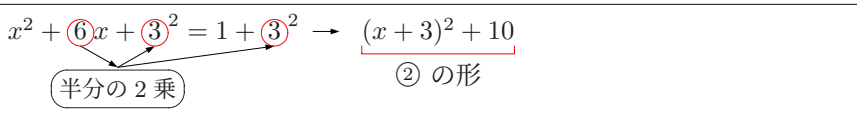
```
¥vskip18pt

¥begin{sikipicture}{%
  ¥dint{}{},f'(x),g(x),dx=,f(x),g(x),-,¥dint{}{},f(x),g'(x),dx
}
  ¥HenKo<henkoH=15pt,yazirusi=r,agezoko=3>¥sikiTv¥sikiTii{¥footnotesize
積分する}
  ¥HenKo<henkoH=15pt,yazirusi=r,agezoko=3>¥sikiTix¥sikiTv{¥footnotesize
そのまま}
  ¥HenKo<henkoH=15pt,yazirusi=a,agezoko=3>¥sikiBiii¥sikiBvi{¥footnotesize
そのまま}
  ¥HenKo<henkoH=15pt,yazirusi=a,agezoko=3>¥sikiBvi¥sikiBx{¥footnotesize
微分する}
¥end{sikipicture}
¥vskip10pt
¥mbox{}
```

→



談話室 No.107 ふたたび　newPh232.tex でも取り上げた使用例を再度取り上げてみます。ソースリストは re107a.tex です。



右側の部分，前回は ¥phkasen でアンダーラインを引きましたが，今回は ¥HenKo で

```
<henkotype=bracket>
```

オプションを用いてみました。

3.5.6 bunpicture 環境

sikipicture 環境は，数式に対するもので

```
¥begin{sikipicture}{.....}
```

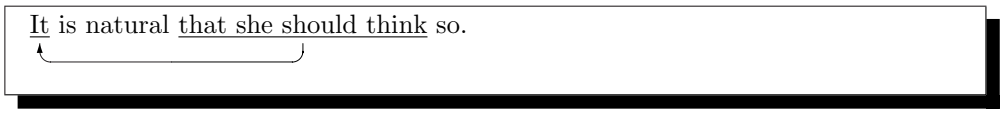
の引数{.....}は数式モードに入ると仮定されています。

これに対して，テキストモードに入るものを bunpicture 環境と称することとします。

bunpicture 環境

```
¥begin{bunpicture}{%
    ¥underline{It}, is natural  ,¥underline{that she should think}, so.}
  ¥HenKo<yazirusi=r,henkotype=bracket,Oval=4pt,Agezoko=2pt>¥bunBi¥bunBiii{}
¥end{bunpicture}
¥vskip5pt

¥mbox{}
```

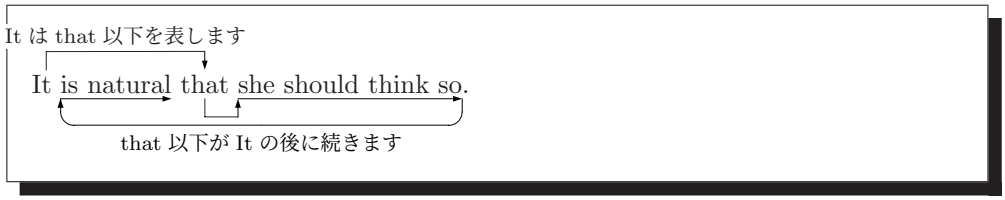
→ 

さらに手が込んで

応用例

```
¥mbox{}¥vskip1¥baselineskip
¥begin{bunpicture}{It ,is natural, that ,she should think so,.}
  ¥HenKo<henkotype=bracket,yazirusi=r,Agezoko=2pt,putoption={(0,1pt)[b]}>%
    ¥bunTiii¥bunTi{¥footnotesize It は that 以下を表します}
  ¥HenKo<henkotype=bracket,yazirusi=a,Agezoko=2pt>%
    {(¥bunxposiii,¥ymin)}{(¥bunxlposiv,¥ymin)}{}
  ¥Add¥ymin{-2}¥yy
¥put(0,0){¥color{red}%
  ¥ArrowLine{(¥bunxlposiv,¥yy)}{(¥bunxrposiv,¥yy)}%
  ¥HenKo<henkotype=bracket,yazirusi=r,henkoH=10pt,Oval=5pt,%
        putoption={(0,-2pt)[t]},Agezoko=2pt>%
    {(¥bunxlposii,¥ymin)}{(¥bunxrposiv,¥ymin)}%
    {¥color{black}¥footnotesize that 以下が It の後に続きます}
  ¥ArrowLine{(¥bunxlposii,¥yy)}{(¥bunxrposii,¥yy)}%
}%
¥end{bunpicture}
¥vskip1¥baselineskip
¥mbox{}
```

→


It は that 以下を表します
It is natural that she should think so.
that 以下が It の後に続きます

4　円・楕円

4.1　円

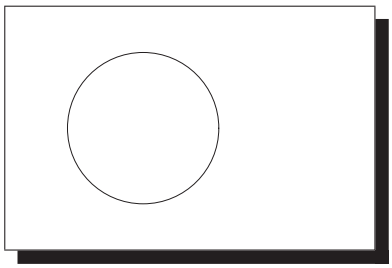
4.1.1　円

中心と半径を指定　円を描画するには，LaTeX で ¥circle がありますが，直径を指定することになっています．半径の方が使いやすいので，¥En を用意しました．次のように使います．

¥En

```
¥begin{picture}(3,3)%
¥def¥C{(1.5,1.5)}%
¥En¥C{1}%
¥end{picture}
```

⟶



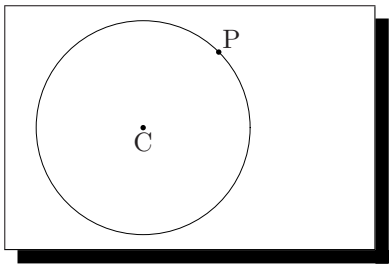
すなわち，¥En は，中心と半径の 2 つの引数をとります．

半径の間接指定　半径を指定する代わりに，円周上の 1 点を指定して描画する方法もあります。

円周上の 1 点を指定

```
¥begin{picture}(3,3)%
¥tenretu{C(1.5,1.5)s;P(2.5,2.5)ne}
¥kuromaru{¥C;¥P}
¥En¥C{tuukaten=¥P}%
¥end{picture}
```

⟶

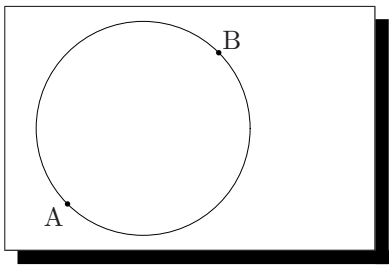


直径を指定　2 点を指定して，それを結ぶ線分を直径とする円を描画するコマンド ¥EnT もあります。

直径を指定

```
¥begin{picture}(3,3)%
¥tenretu{A(.5,.5)sw;B(2.5,2.5)ne}
¥kuromaru{¥A;¥B}
¥EnT¥A¥B
¥end{picture}
```

⟶



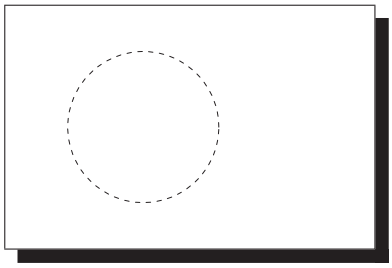
4.1.2　円の破線描画

円を破線で描画するには，

円周を破線で

```
¥begin{picture}(3,3)%
¥def¥C{(1.5,1.5)}%
¥En<hasen=[.6][.8]>¥C{1}%
¥end{picture}
```

→



と<hasen=[**破線の長さ**][**破線の間隔**]>オプションを与えます。

¥En<#1>#2#3
<#1>: hasen=xx **で円弧を破線で描画するときの**
[破線の長さ][破線の間隔]
#2 : **中心の座標**
#3 : **半径**

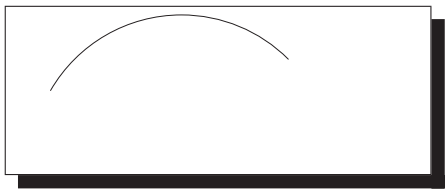
4.1.3 円弧

円弧を描画するために，¥Enko を用意しました.

¥Enko

```
¥begin{picture}(3,2)%
¥def¥C{(2,0)}%
¥Enko¥C{2}{45}{150}
¥end{picture}
```

→



すなわち，¥Enko は,

¥Enko{**中心**}{**半径**}{**開始角**}{**終了角**}

中心，半径，開始角，終了角と 4 つの引数を取ります.

円弧を破線で描画するには，<#1> に

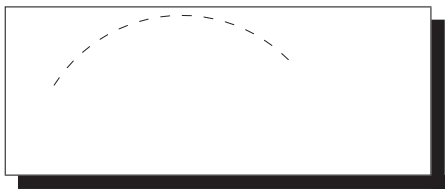
<hasen=[**破線の長さ**][**破線の間隔**]>

オプションを与えます。

円弧を破線で

```
¥begin{picture}(3,2)%
¥def¥C{(2,0)}%
¥Enko<hasen=[.6][.8]>%
¥C{2}{45}{150}
¥end{picture}
```

→



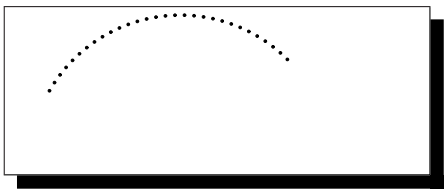
点線で描画するには，<#1>に

<ten=周上に置く点の個数>

オプションを与えます。

円弧を点線で

```
¥begin{picture}(3,2)%
¥def¥C{(2,0)}%
¥Enko<ten=30>%
¥C{2}{45}{150}
¥end{picture}
```



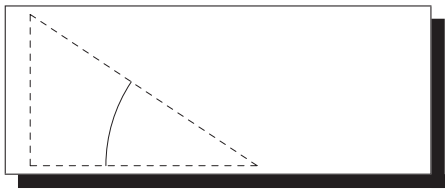
¥Enko の書式です。

```
% 円弧
¥Enko<#1>#2#3#4#5
  #1 : オプション
       hasen=xx で円弧を破線で描画するときの
                [破線の長さ][破線の間隔]
             オプションを与える。
       ten=xx 円周を点線で描画するときの周上の点の個数
       yazirusi=a : 正方向に矢印をつける
                =r : 負方向に矢印をつける
                =b : 両方向に矢印をつける
                =n : 矢印をつけない
  #2 : 中心
  #3 : 半径を直接与えるか
       tuukaten=xx として，円弧の周上の一点を与える
  #4 : 始め角を直接与えるか
       hazimeten=xx として，中心を始点，xx を終点とするベクトルの
       方向角を 始め角とするように指定する。
  #5 : 終り角を直接与えるか
       owariten=xx として，中心を始点，xx を終点とするベクトルの
       方向角を 終り角とするように指定する。
```

始め角，終り角を六十分法の角度で与える代わりに，点を指定して，中心から指定した点に向かうベクトルの方向角で与えるオプションの利用例です。

両端の間接的指定

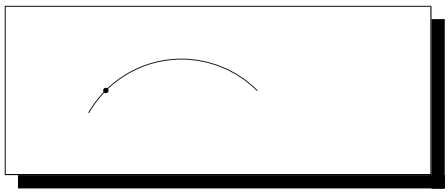
```
¥begin{picture}(3,2)%
¥def¥A{(0,0)}%
¥def¥B{(0,2)}%
¥def¥C{(3,0)}%
¥Enko¥C{2}%
  {hazimeten=¥B}{owariten=¥A}%
¥Dashline[40]{.1}{¥C¥A¥B¥C}%
¥end{picture}
```

→ 

半径も円周上の一点を指定することにより，間接的に与えることもできます。

半径の間接的指定

```
¥begin{picture}(3,2)%
¥def¥A{(1,1)}%
¥def¥C{(2,0)}%
¥Enko¥C{tuukaten=¥A}%
{45}{150}
¥Kuromaru¥A
¥end{picture}
```

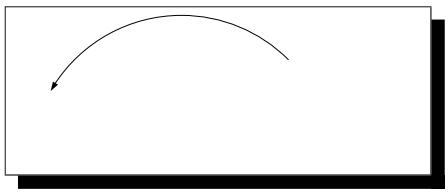
→ 

4.1.4 矢印付きの円弧 (1) 偏角指定

円弧に矢印をつけるには，¥Enko に<yazirusi=a>オプションをつけます。

<yazirusi=a>

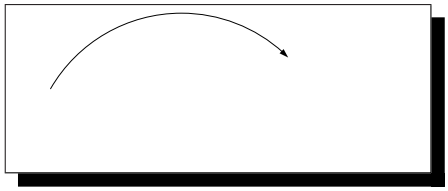
```
¥begin{picture}(3,2)%
¥Enko<yazirusi=a>{(2,0)}{2}{45}{150}
¥end{picture}
```

→ 

矢印を逆向きにしたいときは，オプションの右辺値を ‘r’ とします。

<yazirusi=r>

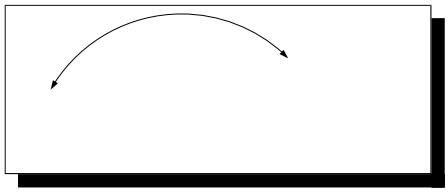
```
¥begin{picture}(3,2)%
¥Enko<yazirusi=r>{(2,0)}{2}{45}{150}
¥end{picture}
```

→ 

また，オプションの右辺値を ‘b’ とすれば，両向きの矢印がつきます。

<yazirusi=b>

```
¥begin{picture}(3,2)%
¥Enko<yazirusi=b>{(2,0)}{2}{45}{150}
¥end{picture}
```

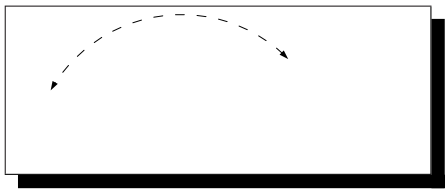
→ 

さらには，<hasen=..>オプションと併用も可能です。

<yazirusi=b>

```
¥begin{picture}(3,2)%
¥Enko<hasen=[.6][.8],yazirusi=b>%
{(2,0)}{2}{45}{150}
¥end{picture}
```

→



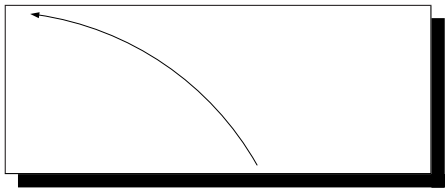
4.1.5 矢印付きの円弧 (2) 端点指定

矢印付きの円弧を，半径と端点を指定して描画するのが ¥ArrowArc コマンドです.

¥ArrowArc

```
¥begin{picture}(3,2)%
¥def¥A{(3,0)}%
¥def¥B{(0,2)}%
¥ArrowArc{4}¥A¥B
¥end{picture}
```

→

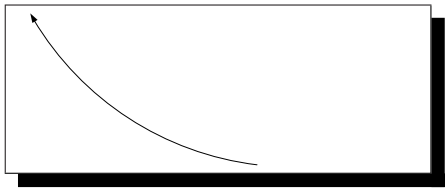


半径に負の値を与えると，回転が負の方向となるように描画されます.

回転の向き

```
¥begin{picture}(3,2)%
¥def¥A{(3,0)}%
¥def¥B{(0,2)}%
¥ArrowArc{-4}¥A¥B
¥end{picture}
```

→



¥ArrowArc[#1]#2#3#4

#1 :

#2 : **半径（負のときは回転が負となる）**

#3 : **始点**

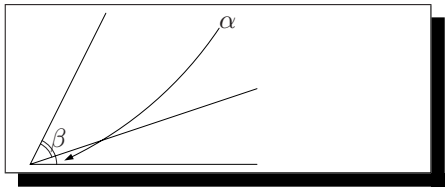
#4 : **終点**

このコマンドを造った理由の一つは，図が込み入って来たりして，文字列を該当個所に置けないとき，ゆったりしたところに文字列を置き，そこから矢線を引こう，というねらいがあります．少し曲げた方が見やすいのではないか，と思います.

¥ArrowArc の一応用

```
¥begin{picture}(3,2)%
¥def¥O{(0,0)}%
¥def¥A{(3,0)}%
¥def¥B{(1,2)}%
¥def¥C{(3,1)}%
¥Kakukigou¥A¥O¥C{}%
¥Kakukigou<2>¥C¥O¥B{$¥beta$}%
¥put(2.5,1.8){$¥alpha$}%
¥ArrowArc[r]{5}%
{(2.5,1.8)}{(0.45,0.05)}%
¥Drawline{¥A¥O¥C}%
¥Drawline{¥O¥B}%
¥end{picture}
```

→



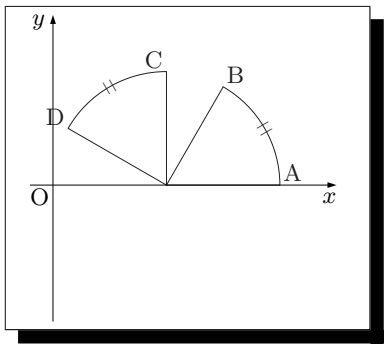
4.1.6 等弧記号

二つの円弧の長さが等しいことを表すのに，円弧の中央部分に短い縦線を入れるコマンドが ¥Toukokigou です。

¥Toukokigou

```
¥begin{zahyou}[ul=15mm](-0.2,2.5)(-1.2,1.5)
¥small
¥def¥O{(1,0)}
¥rtenretu[¥O]{A(1,0)ne;B(1,60)ne;
C(1,90)nw;D(1,150)nw}
¥Enko¥O{1}{0}{60}
¥Enko¥O{1}{90}{150}
¥Toukokigou<2>¥O¥A¥B
¥Toukokigou<2>¥O¥C¥D
¥Drawlines{¥A¥O¥B;¥C¥O¥D}
¥end{zahyou}
```

→



書式は

¥Toukokigou<#1>#2#3#4

#1 : **中央に配置する短い縦棒の個数（デフォルト値**=1**）**

#2 : **円弧の中心**

#3 : **弧の端点** 1

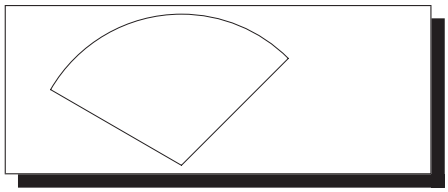
#4 : **弧の端点** 2

4.1.7 扇形

扇形を描画するために，¥ougigata を用意しました．

¥ougigata

```
¥begin{picture}(3,2)%
¥put(2,0){%
¥ougigata{2}{45}{150}}%
¥end{picture}
```



すなわち，¥ougigata は，半径，開始角，終了角と 3 つの引数を取ります．

¥ougigata#1#2#3

#1 : **半径を直接与えるか**
tuukaten=xx **として，円弧の周上の一点を与える**

#2 : **始め角を直接与えるか**
hazimeten=xx **として，中心を始点，**xx **を終点とするベクトルの方向角を 始め角とするように指定する。**

#3 : **終り角を直接与えるか**
owariten=xx **として，中心を始点，**xx **を終点とするベクトルの方向角を 終り角とするように指定する。**

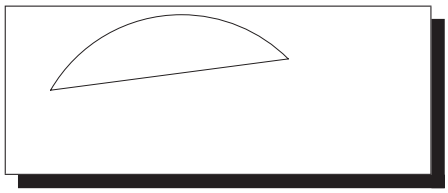
（中心は ¥put（¥emathPut）**で指定する．）**

4.1.8 弓形

弓形を描画するために，¥yumigata を用意しました．

¥yumigata

```
¥begin{picture}(3,2)%
¥put(2,0){%
¥yumigata{2}{45}{150}}
¥end{picture}
```

→ 

すなわち，¥yumigata は，半径，開始角，終了角と 3 つの引数を取ります.

¥yumigata#1#2#3

#1 : **半径を直接与えるか**
tuukaten=xx **として，円弧の周上の一点を与える**

#2 : **始め角を直接与えるか**
hazimeten=xx **として，中心を始点，**xx **を終点とするベクトルの方向角を 始め角とするように指定する。**

#3 : **終り角を直接与えるか**
owariten=xx **として，中心を始点，**xx **を終点とするベクトルの方向角を 終り角とするように指定する。**

(中心は ¥put（¥emathPut）**で指定する.)**

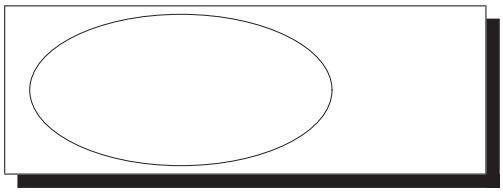
4.2 楕円

4.2.1 楕円

軸が水平，垂直な楕円を描画するコマンドは ¥Daen です.

¥Daen

```
¥begin{picture}(3,2)%
¥Daen{(2,1)}{2}{1}%
¥end{picture}
```

→ 

中心，横方向半径，縦方向半径 と三つの引数をとります.

¥Daen#2#3#4

#1 : **中心の座標**

#2 : **横軸方向の半径**

#3 : **縦軸方向の半径**

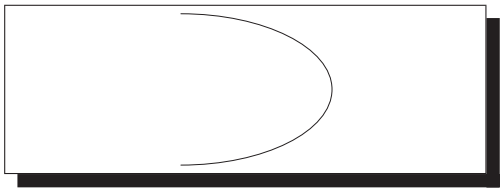
4.2.2 楕円弧

楕円の一部を描画する ¥Daenko です.

¥Daenko

```
¥begin{picture}(4,2)%
¥Put{(2,1)}{%
¥Daenko{2}{1}{-90}{90}}%
¥end{picture}
```

→



¥Daenko<#1>#2#3#4#5

- #1 : key=val
  - hasen=[**破線の長さ**][**破線の間隔**]
  - yazirusi=a : **正方向に矢印をつける**
  - =r : **負方向に矢印をつける**
  - =b : **両方向に矢印をつける**
  - =n : **矢印をつけない**
- #2 : **横軸方向の半径**
- #3 : **縦軸方向の半径**
- #4 : **始め角**
- #5 : **終り角**

（中心は ¥put（¥emathPut）**で指定する.）**

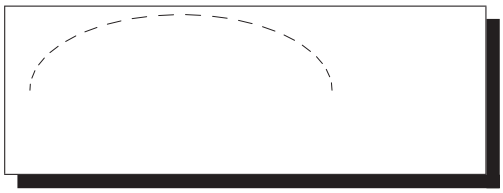
4.2.3 破線

（楕）円（弧）を破線で描画するには ¥Daenko に<hasen=..>オプションを与えます.

楕円弧の破線

```
¥begin{picture}(4,2)%
¥Put{(2,1)}{%
¥Daenko<hasen=[1]>{2}{1}{0}{180}}%
¥end{picture}
```

→

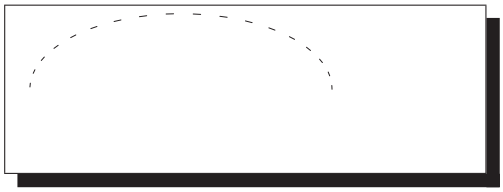


破線の長さを変更するには，オプションの値を変更します．上の図を標準として比率を指定します.

破線の長さ

```
¥begin{picture}(4,2)%
¥Put{(2,1)}{%
¥Daenko<hasen=[0.5]>{2}{1}%
{0}{180}}%
¥end{picture}
```

→

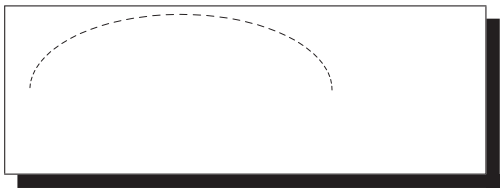


破線の間隔を調整するには，第 2 の [..] オプションです．やはり，標準に対する比率です．

破線の間隔

```
¥begin{picture}(4,2)%
¥Put{(2,1)}{%
¥Daenko<hasen=[0.5][0.5]>%
  {2}{1}{0}{180}}%
¥end{picture}
```

→



なお，横軸方向の半径と縦軸方向の半径を同じ値にすれば，円（弧）を破線で描画することもできます．

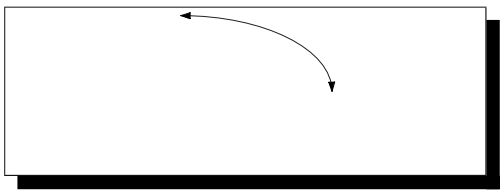
4.2.4 矢印

（楕）円（弧）に矢印をつけるには ¥Daenko に<yazirusi=..>オプションを与えます．

楕円弧に矢印

```
¥begin{picture}(4,2)%
¥Put{(2,1)}{%
¥Daenko<yazirusi=b>%
  {2}{1}{0}{90}}%
¥end{picture}
```

→



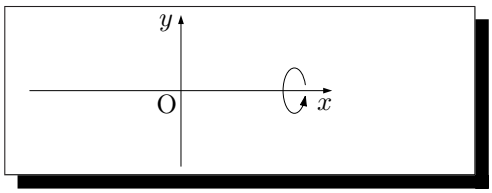
4.2.5 回転記号

楕円弧に矢印を付ける機能を利用して，回転軸の周りに回転を表す記号を付けることができます。

¥kaitenkigou

```
¥begin{zahyou}[ul=10mm](-2,2)(-1,1)
¥Put{(1.5,0)}{¥kaitenkigou}
¥end{zahyou}
```

→

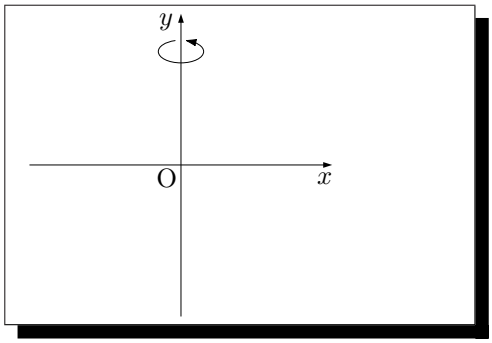


y 軸の周りの回転を表すには，[90] オプションを付けます。

y 軸の周りの回転

```
¥begin{zahyou}[ul=10mm](-2,2)(-2,2)
¥Put{(0,1.5)}{¥kaitenkigou[90]}
¥end{zahyou}
```

→



ここで，矢印は正の回転を表すようにつきますが，これを負の回転を表すようにつけるためには

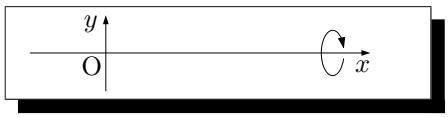
```
<muki=r>
```

オプションを用います。

矢印を逆向き

```
¥begin{zahyou}[ul=10mm](-1,3.5)(-.5,.5)
  ¥Put{(3,0)}{¥kaitenkigou<muki=r>}
¥end{zahyou}
```

→

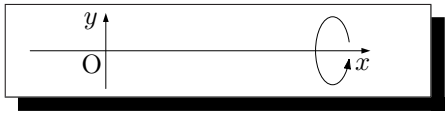


<..>オプションは，本来は倍率指定オプションです。

倍率指定

```
¥begin{zahyou}[ul=10mm](-1,3.5)(-.5,.5)
  ¥Put{(3,0)}{¥kaitenkigou<1.5>}
¥end{zahyou}
```

→

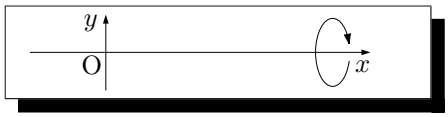


倍率指定と矢印向きなど他のオプションと併用したいときは<bairitu=..>オプションを用います。

倍率指定

```
¥begin{zahyou}[ul=10mm](-1,3.5)(-.5,.5)
  ¥Put{(3,0)}{%
    ¥kaitenkigou<bairitu=1.5,muki=r>}
¥end{zahyou}
```

→



楕円のサイズは，デフォルトでは

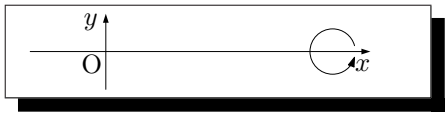
```
tyouhankei=3mm
tanhankei=1.5mm
```

としてあります。これを変更するオプションです：

サイズ変更

```
¥begin{zahyou}[ul=10mm](-1,3.5)(-.5,.5)
  ¥Put{(3,0)}{%
    ¥kaitenkigou<tanhankei=3mm>}
¥end{zahyou}
```

→

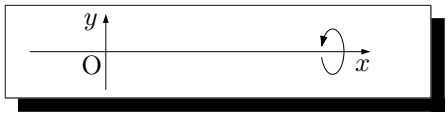


楕円の右端を一部切って矢印をつけていますが，切る場所を変えるオプションを紹介します。

矢印の位置変更

```
¥begin{zahyou}[ul=10mm](-1,3.5)(-.5,.5)
  ¥Put{(3,0)}{¥kaitenkigou%
    <hazimekaku=-165,owarikaku=165>}
¥end{zahyou}
```

→



デフォルト値は

```
hazimekaku=15, owarikaku=345
```

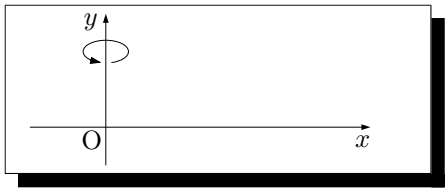
となっています。

回転オプションと併用するときは，上記の指定角は回転する前の状況での値です。上の図を 90° 回転してみましょう。

回転との併用

```
¥begin{zahyou}[ul=10mm]%
    (-1,3.5)(-.5,1.5)
  ¥Put{(0,1)}{¥kaitenkigou%
    <hazimekaku=-165,%
    owarikaku=165>[90]}
¥end{zahyou}
```

→



最後に，¥kaitenkigou の書式です。

直線のまわりに回転させることを表す記号

¥kaitenkigou<#1>[#2]

#1 : **倍率**

または key=val

bairitu= **（倍率）デフォルト値** : 1

muki = r/n n

r **で，負の向き**

n **で，正の向き**

hazimekaku= 15

owarikaku= 345

tyouhankei= 3mm

tanhankei= 1.5mm

#2 : **回転角**

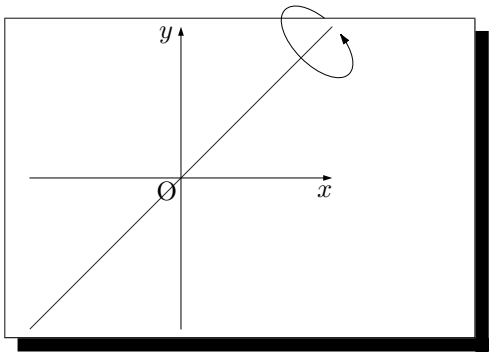
位置は ¥emathPut **で指定**

直線 $y = x$ の周りに回転させる記号を，2 倍のサイズで描いてみます。

y 軸の周りの回転

```
¥begin{zahyou}[ul=10mm](-2,2)(-2,2)
¥Put{(1.8,1.8)}{¥kaitenkigou<2>[45]}
¥kTyokusen¥O{45}{}{}
¥end{zahyou}
```

→



5　円・直線の交点

直線と直線，円と直線，円と円の交点を求めるコマンド類です．コマンド名は ¥*and* の形で * のところは

C : 円
L : 2 点を与えた直線
l : 1 点と方向ベクトルを与えた直線
k : 1 点と方向角を与えた直線

のいずれかで，

¥CandC
¥CandL
¥Candl
¥Candk
¥LandL
¥Landl
¥landl
¥Landk
¥kandk

の 9 種類があります．

5.1　2 直線の交点

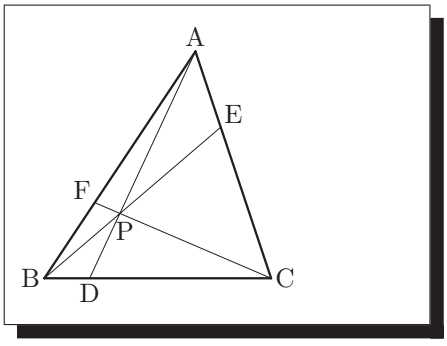
5.1.1　2 直線の交点 (1) ¥LandL

¥LandL の使用例です．

¥LandL

```
¥begin{picture}(3.4,4)(-.2,-.5)%
¥tenretu{B(0,0)w;C(3,0)e;A(2,3)n}
¥Bunten¥A¥B21¥F
¥Bunten¥C¥A21¥E
¥LandL¥B¥E¥C¥F¥P% BE と CF の交点 P
¥LandL¥A¥P¥B¥C¥D% AP と BC の交点 D
¥Put¥F[nw]{F}%
¥Put¥E[ne]{E}%
¥Put¥D[s]{D}%
¥Put¥P(-1.5pt,-3pt)[lt]{P}%
¥Drawlines{¥A¥D;¥B¥E;¥C¥F}%
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥end{picture}
```

→



¥LandL#1#2#3#4#5

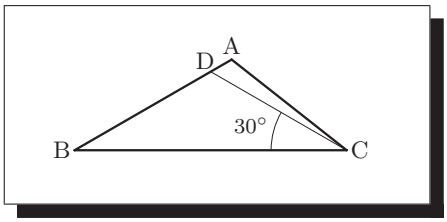
2 点 #1，#2 を通る直線と
2 点 #3，#4 を通る直線の交点を #5 に与える．

5.1.2 2 直線の交点 (2) ¥Landl

△ABC の辺 AB 上に ∠BCD = 30° となる点 D を求める例です．

¥Landl

```
¥begin{zahyou*}[ul=12mm]%
(-.5,3.5)(-.5,1.5)¥small
¥tenretu{A(1.732,1)n;B(0,0)w;C(3,0)e}
¥Landl¥A¥B¥C{(-1.7321,1)}¥D
¥Put¥D(-2pt,1pt)[b]{D}%
¥Kakukigou¥D¥C¥B<Hankei=10mm>%
  (-2pt,2pt)[r]{¥footnotesize 30¥Deg}%
¥Drawline{¥C¥D}%
¥thicklines¥Drawline{¥A¥B¥C¥A}%
¥end{zahyou*}
```



¥Landl#1#2#3#4#5

2 点 #1，#2 を通る直線と
点 #3 を通り，方向ベクトルが #4 の直線の交点を #5 に与える．

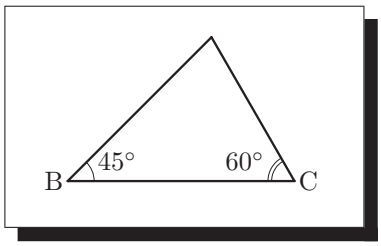
5.1.3 2 直線の交点 (3) ¥landl

直線を 1 点と方向ベクトルで与える場合の，交点を求めるコマンドが ¥landl です．

¥landl

```
¥begin{zahyou*}[ul=10mm]%
  (-.5,3.5)(-.5,2.2)
¥tenretu{B(0,0)w;C(3,0)e}
¥landl¥B{(1,1)}¥C{(-1,1.7321)}¥A
¥Kakukigou¥C¥B¥A{}
¥Put¥B(4mm,4pt)[lb]{45¥Deg}
¥Kakukigou<2>¥A¥C¥B{}
¥Put¥C(-4mm,4pt)[rb]{60¥Deg}
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥end{zahyou*}
```

→



¥landl#1#2#3#4#5

点 #1 を通り，方向ベクトルが #2 の直線と
点 #3 を通り，方向ベクトルが #4 の直線の交点を #5 に与える．

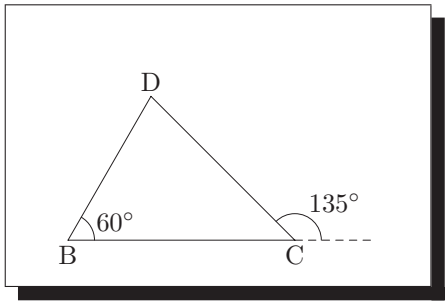
5.1.4 2 直線の交点 (4) ¥Landk

直線を 1 点と方向角 (x 軸の正の向きとなす角—六十分法）で与えた場合です。まずは ¥kandk の使用例です。

¥kandk

```
¥begin{zahyou*}(-.5,4)(-.5,3)
¥tenretu{B(0,0)s;C(3,0)s}
¥kandk¥B{60}¥C{135}¥D
¥Put¥D[n]{D}
¥Kakukigou¥C¥B¥D(2pt,2pt)[l]{60¥Deg}
¥Kakukigou¥XMAX¥C¥D[ne]{135¥Deg}
¥Drawline{¥B¥C¥D¥B}%
¥Hasen{¥C¥XMAX}
¥end{zahyou*}
```

→



¥kandk#1#2#3#4#5

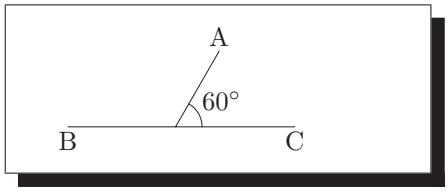
点 #1 を通り，方向角が #2 の直線と
点 #3 を通り，方向角が #4 の直線の交点を #5 に与える．

ついで，¥Landk の使用例です。

¥Landk

```
¥begin{zahyou*}[ul=10mm]%
  (-.5,3.5)(-.5,1.5)
¥tenretu{B(0,0)s;A(2,1)n;C(3,0)s}
¥Landk¥B¥C¥A{60}¥D
¥Kakukigou¥C¥D¥A[ne]{60¥Deg}
¥Drawlines{¥B¥C;¥A¥D}
¥end{zahyou*}
```

→



¥Landk#1#2#3#4#5

2 点 #1，#2 を通る直線と
点 #3 を通り，方向角が #4 の直線の交点を #5 に与える．

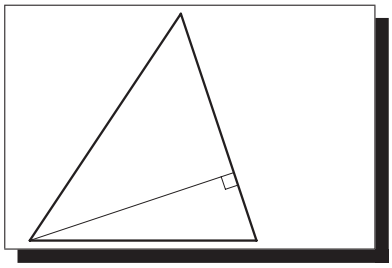
5.1.5 垂線の足

三角形の頂点から対辺（またはその延長）に下した垂線の足を求めるコマンド ¥Suisen です.

¥Suisen

```
¥begin{picture}(3,3)%
¥def¥O{(0,0)}%
¥def¥A{(3,0)}%
¥def¥B{(2,3)}%
¥Suisen¥O¥A¥B¥H
¥Tyokkakukigou¥O¥H¥A
¥Drawline{¥O¥H}%
¥thicklines
¥Drawline{¥O¥A¥B¥O}%
¥thinlines
¥end{picture}
```

→



¥Suisen#1#2#3#4

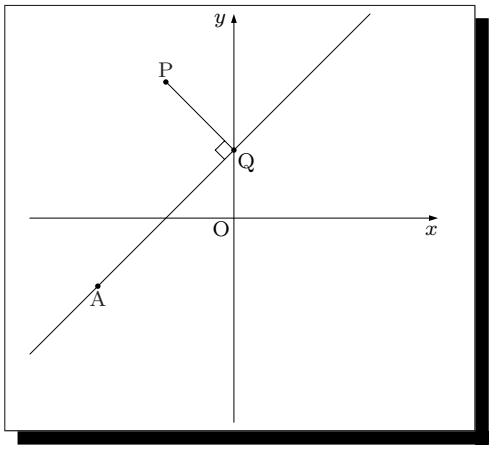
点 #1 から直線 #2#3 へ下ろした垂線の足を #4 にセット

関連して，直線が 1 点と方向ベクトルまたは方向角で与えられている場合に用いるコマンドがそれぞれ ¥mSuisen, ¥kSuisen です。使用例を一つずつあげます。

¥mSuisen

```
¥unitlength9mm¥footnotesize
¥begin{zahyou}(-3,3)(-3,3)
¥tenretu{A(-2,-1)s;P(-1,2)n}
¥def¥houkouV{(1,1)}
¥mSuisen¥P¥A¥houkouV¥Q
¥kuromaru{¥A;¥P;¥Q}
¥Put¥Q[se]{Q}
¥Drawline{¥P¥Q}
¥Tyokkakukigou¥P¥Q¥A
¥mTyokusen¥A¥houkouV{}{}
¥end{zahyou}
```

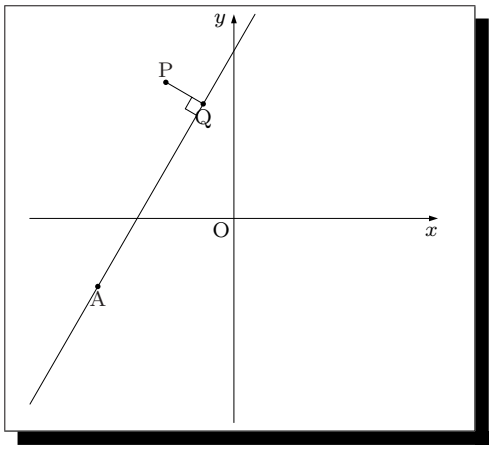
→



¥kSuisen

```
¥unitlength9mm¥footnotesize
¥begin{zahyou}(-3,3)(-3,3)
¥tenretu{A(-2,-1)s;P(-1,2)n}
¥def¥kaku{60}
¥kSuisen¥P¥A¥kaku¥Q
¥kuromaru{¥A;¥P;¥Q}
¥Put¥Q[s]{Q}
¥Drawline{¥P¥Q}¥Tyokkakukigou¥P¥Q¥A
¥kTyokusen¥A¥kaku{}{}
¥end{zahyou}
```

→



5.1.6 直線に関する対称点

前節の垂線を発展させて，点の直線に関する対称点を求めるコマンド

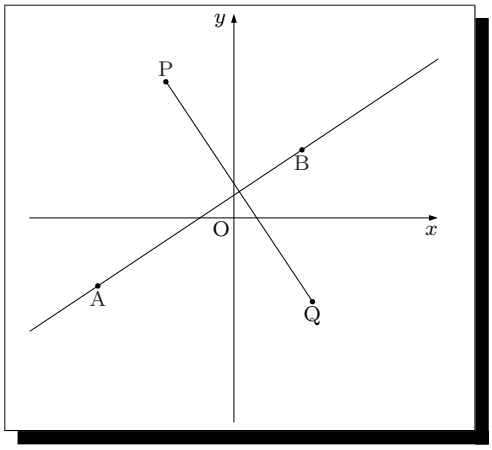
¥Taisyouten 2 **点を通る直線**
¥mTaisyouten 1 **点と方向ベクトルを指定した直線**
¥kTaisyouten 1 **点と方向角を指定した直線**

を新設しました。以下，その使用例を並べます。

¥Taisyouten

```
¥unitlength9mm¥footnotesize
¥begin{zahyou}(-3,3)(-3,3)
¥tenretu{A(-2,-1)s;B(1,1)s;P(-1,2)n}
¥Taisyouten¥P¥A¥B¥Q
¥kuromaru{¥A;¥B;¥P;¥Q}
¥Put¥Q[s]{Q}
¥Drawline{¥P¥Q}
¥Tyokusen¥A¥B{}{}
¥end{zahyou}
```

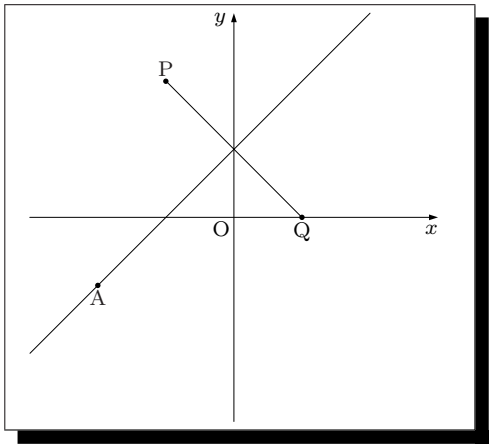
→



¥mTaisyouten

```
¥unitlength9mm¥footnotesize
¥begin{zahyou}(-3,3)(-3,3)
¥tenretu{A(-2,-1)s;P(-1,2)n}
¥def¥houkouV{(1,1)}
¥mTaisyouten¥P¥A¥houkouV¥Q
¥kuromaru{¥A;¥P;¥Q}
¥Put¥Q[s]{Q}
¥Drawline{¥P¥Q}
¥mTyokusen¥A¥houkouV{}{}
¥end{zahyou}
```

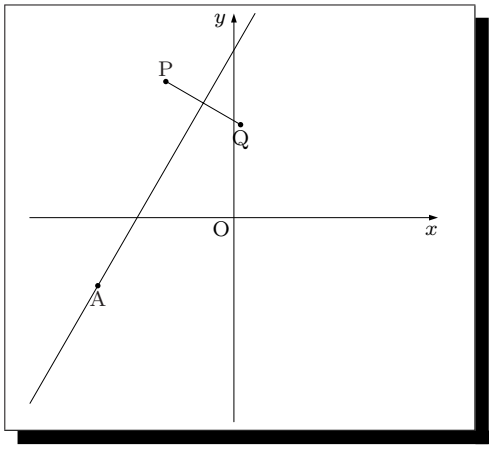
→



¥kTaisyouten

```
¥unitlength9mm¥footnotesize
¥begin{zahyou}(-3,3)(-3,3)
¥tenretu{A(-2,-1)s;P(-1,2)n}
¥def¥kaku{60}
¥kTaisyouten¥P¥A¥kaku¥Q
¥kuromaru{¥A;¥P;¥Q}
¥Put¥Q[s]{Q}
¥Drawline{¥P¥Q}
¥kTyokusen¥A¥kaku{}{}
¥end{zahyou}
```

→

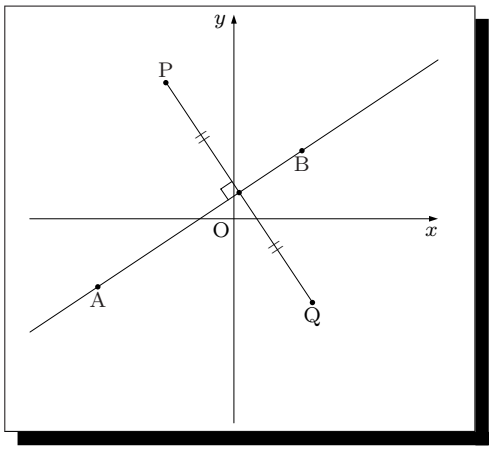


対称の中心（垂線の足）が必要なときは，オプション引数 [#4] にその点を受取る制御綴を与えておきます。

対称の中心

```
¥unitlength9mm¥footnotesize
¥begin{zahyou}(-3,3)(-3,3)
¥tenretu{A(-2,-1)s;B(1,1)s;P(-1,2)n}
¥Taisyouten¥P¥A¥B[¥H]¥Q
¥kuromaru{¥A;¥B;¥P;¥Q;¥H}
¥Put¥Q[s]{Q}
¥Tyokkakukigou¥P¥H¥A
¥touhenkigou<2>{¥P¥H;¥H¥Q}
¥Drawline{¥P¥Q}
¥Tyokusen¥A¥B{}{}
¥end{zahyou}
```

→

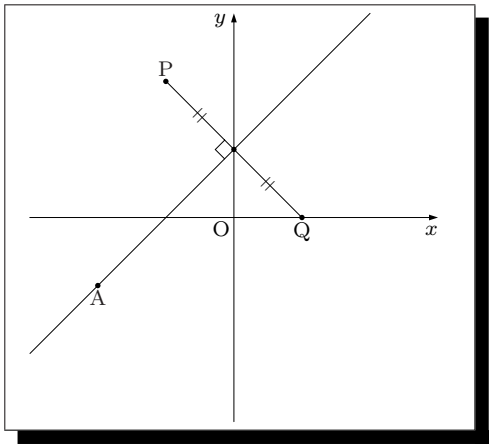


この機能は，¥mTaisyouten, ¥kTaisyouten にも使用できます。

¥mTaisyouten

```
¥unitlength9mm¥footnotesize
¥begin{zahyou}(-3,3)(-3,3)
¥tenretu{A(-2,-1)s;P(-1,2)n}
¥def¥houkouV{(1,1)}
¥mTaisyouten¥P¥A¥houkouV[¥H]¥Q
¥kuromaru{¥A;¥P;¥Q;¥H}
¥Put¥Q[s]{Q}
¥Drawline{¥P¥Q}
¥Tyokkakukigou¥P¥H¥A
¥touhenkigou<2>{¥P¥H;¥H¥Q}
¥mTyokusen¥A¥houkouV{}{}
¥end{zahyou}
```

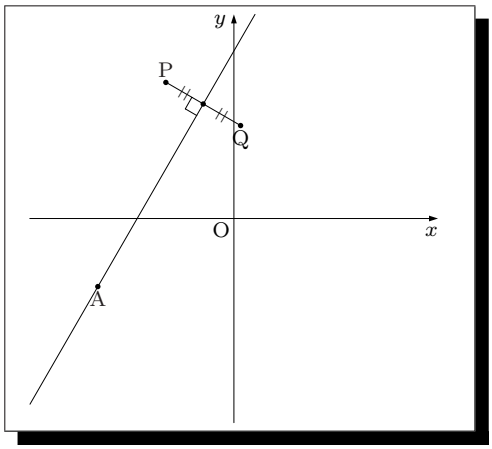
→



¥kTaisyouten

```
¥unitlength9mm¥footnotesize
¥begin{zahyou}(-3,3)(-3,3)
¥tenretu{A(-2,-1)s;P(-1,2)n}
¥def¥kaku{60}
¥kTaisyouten¥P¥A¥kaku[¥H]¥Q
¥kuromaru{¥A;¥P;¥Q;¥H}
¥Put¥Q[s]{Q}
¥Drawline{¥P¥Q}
¥Tyokkakukigou¥P¥H¥A
¥touhenkigou<2>{¥P¥H;¥H¥Q}
¥kTyokusen¥A¥kaku{}{}
¥end{zahyou}
```

→



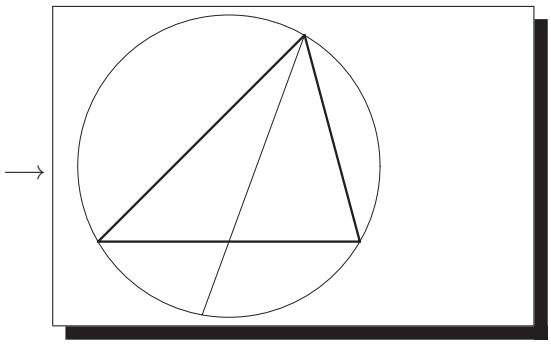
5.2 円と直線の交点

5.2.1 円と直線の交点 (1) ¥CandL

△ABC の中線 AM の延長が △ABC の外接円と交わる点 D を求めます.

¥CandL

```
¥begin{picture}%
(4,4)(-2,-2)%
¥def¥O{(0,0)}%
¥kyokuTyoku(2,60)¥A
¥kyokuTyoku(2,-150)¥B
¥kyokuTyoku(2,-30)¥C
¥Bunten¥B¥C{1}{1}¥M
¥CandL¥O{2}¥A¥M¥D¥E
¥Drawline{¥A¥D}%
¥En¥O{2}%
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥thinlines
¥end{picture}
```



¥CandL#1#2#3#4#5#6
¥CandL*#1#2#3#4#5#6
　　点 #1 を中心とし，半径 #2 の円と
　　2 点 #3，#4 を通る直線との交点を #5 と #6 にセットする．
円と直線の 2 つの交点のどちらを #5 とするかについては
　¥CandL の場合
　　2 つの交点のうち，x 座標の小さい方が #5
　　2 つの交点の x 座標が一致するときは，y 座標の小さい方が #5
　¥CandL* の場合
　　円の中心 (#1) と 2 つの交点 (#5，#6) で作られる三角形の周を
　　#1 → #5 → #6 → #1
　　とたどる回り方が正の回転となるように定める。
　　（三角形がつぶれる場合は，¥CandL の定め方に従う）

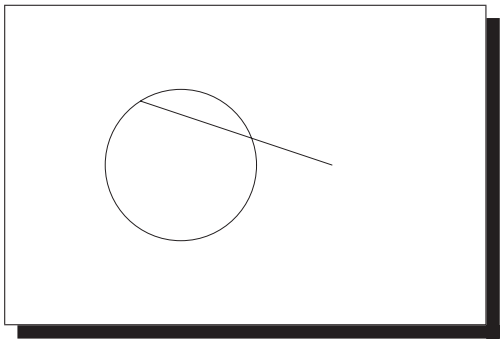
5.2.2 円と直線の交点 (2) ¥Candl

原点中心，半径 1 の円と，点 (2,0) を通り傾き $-\frac{1}{3}$ の直線との交点を求めます.

¥Candl

```
¥begin{picture}%
(4,4)(-2,-2)%
¥def¥O{(0,0)}%
¥def¥A{(2,0)}%
¥Candl¥O{1}¥A{(3,-1)}¥P¥Q
¥Drawline{¥A¥P}%
¥En¥O{1}%
¥end{picture}
```

→



¥Candl#1#2#3#4#5#6

点 #1 を中心とし，半径 #2 の円と
点 #3 を通り，方向ベクトルが #4 の直線
との交点を #5 と #6 にセットする．

2 つの交点のうち，どちらを #5 とするかは ¥CandL と同じ。

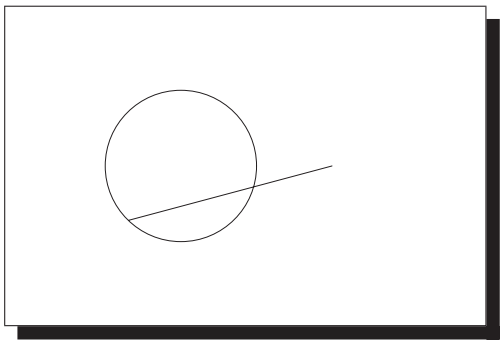
5.2.3 円と直線の交点 (3) ¥Candk

原点中心，半径 1 の円と，点 (2,0) を通り傾き方向角 15° の直線との交点を求めます.

¥Candl

```
¥begin{picture}%
(4,4)(-2,-2)%
¥def¥O{(0,0)}%
¥def¥A{(2,0)}%
¥Candk¥O{1}¥A{15}¥P¥Q
¥Drawline{¥A¥P}%
¥En¥O{1}%
¥end{picture}
```

→



¥Candk#1#2#3#4#5#6

点 #1 を中心とし，半径 #2 の円と
点 #3 を通り，方向角が #4 の直線
との交点を #5 と #6 にセットする．

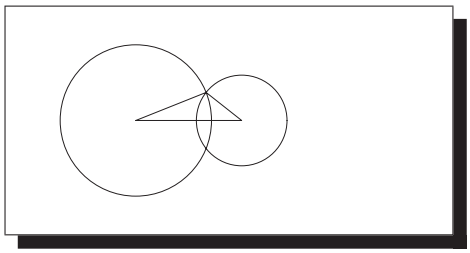
2 つの交点のうち，どちらを #5 とするかは ¥CandL と同じ。

5.3 円と円の交点 ¥CandC

線分 BC の長さが 7 のとき，B を中心とする半径 5 の円と，C を中心とする半径 3 の円との交点を A とすれば，3 辺の長さが 7, 5, 3 の三角形がえられます.

¥CandC

```
{¥unitlength2mm%
¥begin{picture}%
(12,14)(-7,-7)%
¥def¥B{(0,0)}%
¥def¥C{(7,0)}%
¥CandC¥B{5}¥C{3}¥D¥A
¥Drawline{¥A¥B¥C¥A}%
¥En¥B{5}%
¥En¥C{3}%
¥end{picture}}
```

→ 

¥CandC#1#2#3#4#5#6

点 #1 を中心，半径 #2 の円と
点 #3 を中心，半径 #4 の円との交点を#5 と#6 にセット
円と円の 2 つの交点のどちらを #5 とするかについては
¥CandC の場合
2 つの交点のうち，x 座標の小さい方が #5
2 つの交点の x 座標が一致するときは，y 座標の小さい方が #5
¥CandC* の場合
円の中心（#1）と 2 つの交点（#5，#6）で作られる三角形の周を
#1 → #5 → #6 → #1
とたどる回り方が正の回転となるように定める。
（三角形がつぶれる場合は，¥CandC の定め方に従う）

6　楕円と直線の交点

6.1　¥EandL

円と直線の交点を求めるコマンドを発展させ，楕円と直線の交点を求めるコマンドが ¥EandL です。その書式は

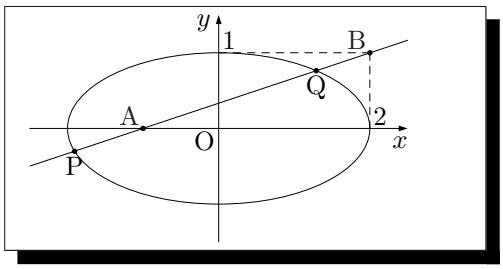
```
¥def¥EandL#1#2#3#4#5#6#7{%
```
#1 ： 楕円の中心
#2 ： x 軸方向の半径
#3 ： y 軸方向の半径
#4 ： 直線上の点 1
#5 ： 直線上の点 2
#6 ： 交点 1 を受け取る制御綴
#7 ： 交点 2 を受け取る制御綴

使用例です。2 点 A$(-1,\ 0)$, B$(2,\ 1)$ を通る直線と楕円 $\dfrac{x^2}{4}+y^2=1$ との交点 P, Q を求めます。

¥EandL

```
¥begin{zahyou}[ul=10mm]%
    (-2.5,2.5)(-1.5,1.5)
  ¥tenretu{A(-1,0)nw;B(2,1)nw}
  ¥kuromaru{¥A;¥B}
  ¥Put¥B[syaei=xy,xpos={[ne]},
    ypos={[ne]}]{}
  ¥Daen¥O{2}{1}¥Tyokusen¥A¥B{}{}
  ¥EandL¥O{2}{1}¥A¥B¥P¥Q
  ¥Put¥P[s]{P}¥Put¥Q[s]{Q}
  ¥kuromaru{¥P;¥Q}
¥end{zahyou}
```

→



6.2　¥Eandl

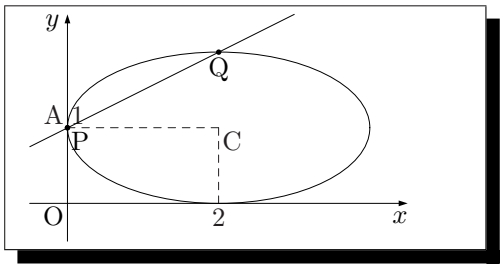
直線を 1 点と方向ベクトルで与える場合です。

楕円 $\dfrac{(x-2)^2}{4}+(y-1)^2=1$ と，点 A$(0,\ 1)$ を通り方向ベクトルが $(2,\ 1)$ の直線の交点を求めます。この場合，点 A は楕円上にありますから，交点の一方 P と A は一致します。

¥Eandl

```
¥begin{zahyou}[ul=10mm]%
    (-.5,4.5)(-.5,2.5)
  ¥tenretu{A(0,1)nw;C(2,1)se}
  ¥Put¥C[syaei=xy,ypos={[ne]}]{}
  ¥def¥m{(2,1)}%
  ¥Kuromaru{¥A}
  ¥Daen¥C{2}{1}
  ¥mTyokusen¥A¥m{}{}
  ¥Eandl¥C{2}{1}¥A¥m¥P¥Q
  ¥Put¥P[se]{P}¥Put¥Q[s]{Q}
  ¥kuromaru{¥P;¥Q}
¥end{zahyou}
```

→



6.3 ¥Eandk

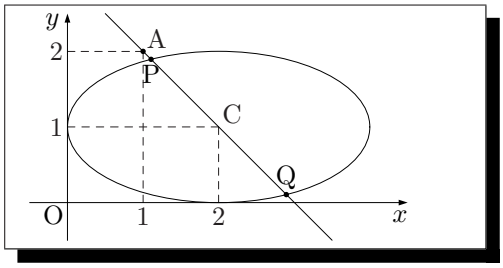
直線を 1 点と方向角（六十分法）で与える場合です。

楕円 $\frac{(x-2)^2}{4}+(y-1)^2=1$ と，点 A(1, 2) を通り方向角が $-45°$ である直線との交点を求めます。

¥Eandk

```
¥begin{zahyou}[ul=10mm]%
    (-.5,4.5)(-.5,2.5)
  ¥tenretu{A(1,2)ne;C(2,1)ne}
  ¥Put¥A[syaei=xy]{}
  ¥Put¥C[syaei=xy]{}
  ¥def¥m{(2,1)}%
  ¥Kuromaru{¥A}
  ¥Daen¥C{2}{1}
  ¥kTyokusen¥A{-45}{}{}
  ¥Eandk¥C{2}{1}¥A{-45}¥P¥Q
  ¥Put¥P[s]{P}¥Put¥Q[n]{Q}
  ¥kuromaru{¥P;¥Q}
¥end{zahyou}
```

→



6.4 楕円の接線

6.4.1 ¥DaennoSessen

楕円の周上の点における接線の方向ベクトルを求めるコマンドです。

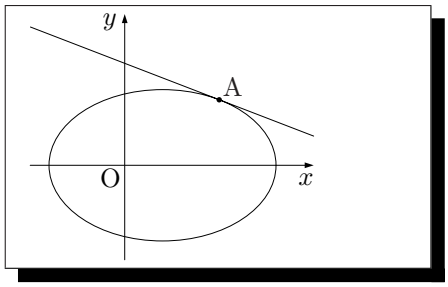
¥DaennoSessen#1#2#3#4#5{%
#1 : **楕円の中心**
#2 : x **軸方向の半径**
#3 : y **軸方向の半径**
#4 : **接点**
#5 : **接線の方向ベクトルを受け取る制御綴**

楕円 $\dfrac{(x-1)^2}{9}+\dfrac{y^2}{4}=1$ 上の点 A $\left(\dfrac{5}{2},\ \sqrt{3}\right)$ における接線を引きます。

¥EandL

```
¥begin{zahyou}[ul=5mm]%
      (-2.5,5)(-2.5,4)
  ¥tenretu{A(2.5,1.732)ne;[]C(1,0)}
  ¥Kuromaru¥A
  ¥Daen¥C{3}{2}
  ¥DaennoSessen¥C{3}{2}¥A¥uvec
  ¥mTyokusen¥A¥uvec{}{}
¥end{zahyou}
```

→



6.4.2 ¥DaenniSessen

つぎは，楕円の外部の点から楕円に引いた接線の接点を求めます。

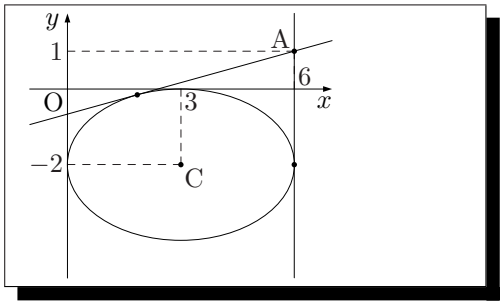
¥DaenniSessen#1#2#3#4#5#6{%
#1 : **楕円の中心**
#2 : x **軸方向の半径**
#3 : y **軸方向の半径**
#4 : **楕円の外部の点**
#5 : **接点** 1 **を受け取る制御綴**
#6 : **接点** 2 **を受け取る制御綴**

楕円 $\dfrac{(x-3)^2}{9}+\dfrac{(y+2)^2}{4}=1$ に点 A(6, 1) から接線を引きます。

¥DaennnoSessen

```
¥begin{zahyou}[ul=5mm]%
    (-1,7)(-5,2)
  ¥tenretu{A(6,1)nw;C(3,-2)se}
  ¥Put¥C[syaei=xy,xpos={[se]}]{}
  ¥Put¥A[syaei=xy,xpos={[ne]}]{}
  ¥Kuromaru{¥A}
  ¥Daen¥C{3}{2}
  ¥DaenniSessen¥C{3}{2}¥A¥P¥Q
  ¥kuromaru{¥P;¥Q;¥C}
  ¥Tyokusen¥A¥P{}{}
  ¥Tyokusen¥A¥Q{}{}
¥end{zahyou}
```

→



6.4.3 ¥Earg

楕円の媒介変数表示

$$x = x_0 + a\cos\theta$$
$$y = y_0 + b\sin\theta$$

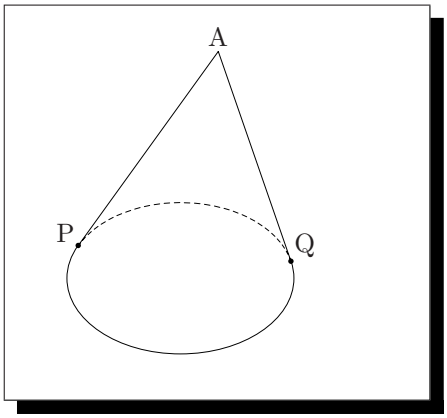
において，周上の点 $(x,\ y)$ を指定して θ を求めるマクロです。

```
¥Earg#1#2#3#4#5{%
 #1 : 楕円の中心
 #2 : x 軸方向の半径
 #3 : y 軸方向の半径
 #4 : 周上の点
 #5 : 媒介変数の値（六十分法）
```

¥Earg

```
¥begin{zahyou*}[ul=5mm](-3,5)(-4,6)
  ¥tenretu{[]C(1,-1);A(2,5)n}
  ¥DaenniSessen¥C{3}{2}¥A¥P¥Q
  ¥Put¥P[nw]{P}¥Put¥Q[ne]{Q}
  ¥kuromaru{¥P;¥Q}
  ¥Earg¥C{3}{2}¥P¥argP
  ¥Earg¥C{3}{2}¥Q¥argQ
  ¥Put¥C{¥Daenko<hasen=[.5][.5]>%
    {3}{2}{¥argQ}{¥argP}}
  ¥Add¥argQ{360}¥argQQ
  ¥Put¥C{¥Daenko{3}{2}{¥argP}{¥argQQ}}
  ¥Drawline{¥P¥A¥Q}
¥end{zahyou*}
```

→



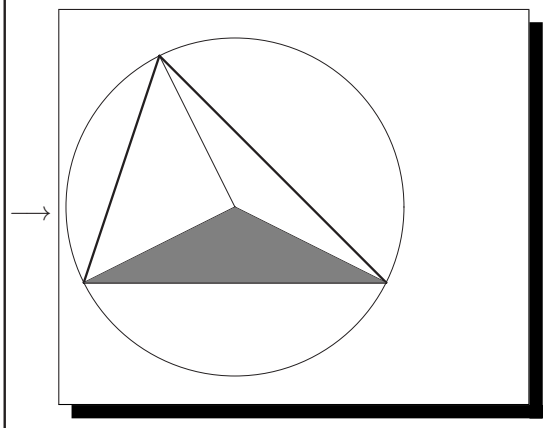
7　塗りつぶし (1)

7.1　多角形内部の塗りつぶし

多角形の内部を塗りつぶすコマンド ¥Nuritubusi です.

多角形の塗りつぶし

```
¥begin{picture}(4,5)(0,-1.5)
¥def¥A{(0,0)}%
¥def¥B{(4,0)}%
¥def¥C{(1,3)}%
¥Gaisetuen¥A¥B¥C%
¥Drawline{¥vGaisin¥A}%
¥Drawline{¥vGaisin¥B}%
¥Drawline{¥vGaisin¥C}%
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥thinlines
%        塗りつぶし
¥Nuritubusi{¥A¥vGaisin¥B¥A}%
¥end{picture}
```



¥Nuritubusi の書式は

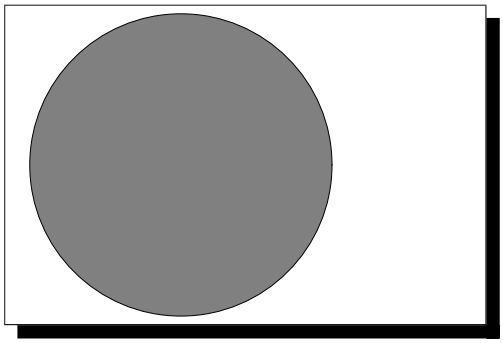
```
¥Nuritubusi[#1]#2
  #1 : 塗る濃さ
       ( 0 と 1 の間の数．0 は真っ白，1 は真っ黒．
         デフォルトは 0.5)
  #2 : 多角形の点列（閉じていなければなりません．）
```

7.2　円内部の塗りつぶし

円の内部を塗りつぶすのが ¥En* コマンドです.

円の塗りつぶし

```
¥begin{picture}(4,4)
¥def¥A{(2,2)}%
¥En*¥A{2}%
¥En¥A{2}%
¥end{picture}
```



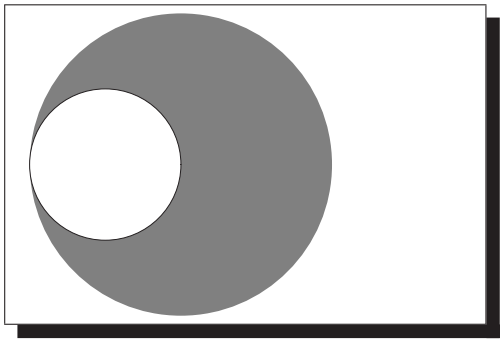
¥En*[#1]#2#3
　#1 : **塗る濃さ**
　（ 0 **と** 1 **の間の数．** 0 **は真っ白，**1 **は真っ黒．**
　デフォルトは 0.5）
　#2 : **中心の座標**
　#3 : **半径**

オプション引数 [..] で濃さを指定することができます．デフォルトは 0.5 です．特に 0 を指定すると白抜きとなります．

白抜き

```
¥begin{picture}(4,4)
¥def¥A{(2,2)}%
¥def¥B{(1,2)}%
¥En*¥A{2}%
¥En*[0]¥B{1}%
¥end{picture}
```

→

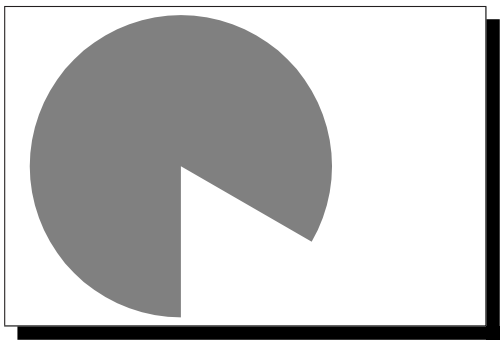


7.3 扇形の塗りつぶし

扇形の塗りつぶし

```
¥begin{picture}(4,4)
¥def¥A{(2,2)}%
¥Put¥A{%
  ¥ougigata*{2}{-30}{270}}%
¥end{picture}
```

→

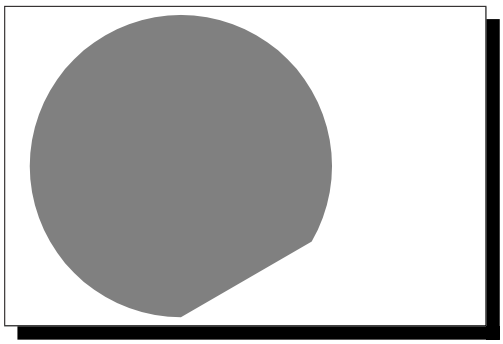


¥ougigata*[#1]<#2>#3#4#5
　#1 : **塗りつぶしの濃さ**
　#2 : **境界線描画オプション**
　　<border=1> **で境界線を描画する。**
　#3 **〜** #5 : ¥ougigata **の** #1〜#3 **と同じ**
　（中心は ¥put（¥Put）**で指定する．）**

7.4 弓形の塗りつぶし

弓形の塗りつぶし

```
¥begin{picture}(4,4)
¥def¥A{(2,2)}%
¥Put¥A{%
  ¥yumigata*{2}{-30}{270}}%
¥end{picture}
```

→ 

¥yumigata*[#1]<#2>#3#4#5

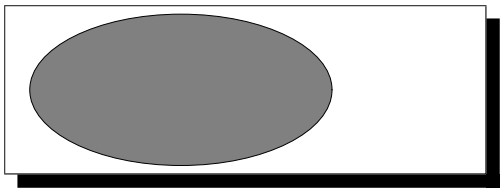
- #1 : **塗りつぶしの濃さ**
- #2 : **境界線描画オプション**
  - <border=1> **で境界線を描画する。**
- #3 **〜** #5 : ¥yumigata **の** #1〜#3 **と同じ**

（中心は ¥put（¥Put）**で指定する.）**

7.5 楕円の塗りつぶし

楕円の塗りつぶし

```
¥begin{picture}(4,2)
¥def¥A{(2,1)}%
¥Daen*¥A{2}{1}%
¥end{picture}
```

→ 

¥Daen*[#1]#2#3#4

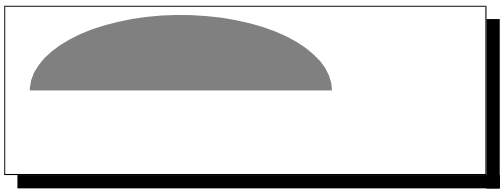
- #1 : **塗る濃さ**
  （ 0 **と** 1 **の間の数.** 0 **は真っ白,** 1 **は真っ黒.**
  デフォルトは 0.5）
  - #2 : **中心の座標**
  - #3 : **横軸方向の半径**
  - #4 : **縦軸方向の半径**

7.6 弓形（楕円弧）の塗りつぶし

弓形（楕円弧）の塗りつぶし

```
¥begin{picture}(4,2)
¥def¥A{(2,1)}%
¥Put¥A{%
¥Daenko*{2}{1}{0}{180}}%
¥end{picture}
```

→



¥Daenko*[#1]#2#3#4#5

- #1 : **塗る濃さ**
  （ 0 **と** 1 **の間の数．** 0 **は真っ白，**1 **は真っ黒．**
  デフォルトは 0.5）
- #2 : **横軸方向の半径**
- #3 : **縦軸方向の半径**
- #4 : **始め角**
- #5 : **終り角**
  （中心は ¥put（¥Put）**で指定する．）**

7.7 カラー指定

¥En*, ¥Nuritubusi において，色をつけるときは少々厄介なことがあります。

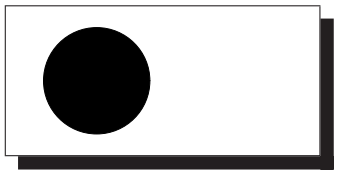
7.7.1 ¥color による塗り色指定

次の例では，

¥color による塗り色指定

```
¥begin{picture}(50,50)
  ¥color{red}¥put(25,25){¥circle*{40}}
¥end{picture}
```

→



さて，あなたの環境では円は何色で塗りつぶされていますか。実は，それは dvi-ware に依存するのです。emath では

```
 .tex ----> .dvi by platex
 .dvi ----> .ps  by dvips
(.ps  ----> .pdf by Distiller)
```

を標準としていますが，この方式で作成される .ps(, .pdf) は真っ黒に塗りつぶされています。赤はかけらも見えません。

注 このあたり，さらに分析をしておきます。

¥color{red}¥circle*{..}は分析すると

```
¥color{red}¥special{bk}¥circle{..}
```

となります。この順序を一部入れ替えて

```
¥special{bk}¥color{red}¥circle{..}
```

とすると，dvips で作られる .ps ファイルも円の内部が赤くなります。

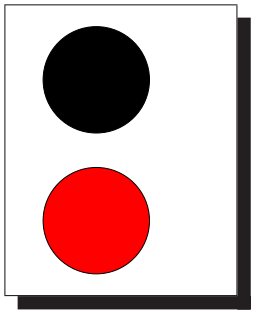
¥color コマンド発行のタイミング

```
¥begin{picture}(50,50)

¥put(25,25){¥color{red}¥special{bk}¥circle{40}}
¥end{picture}
¥begin{picture}(50,50)

¥put(25,25){¥special{bk}¥color{red}¥circle{40}}
¥end{picture}
```

→



2 つの円が描画されますが，dvips で変換した .ps ファイルでは，円の内部は

上は黒
下は赤

で塗りつぶされます。

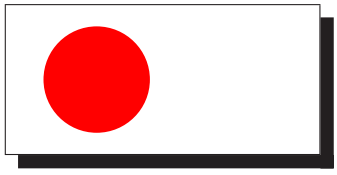
7.7.2 ¥En*の [nuriiro=..] オプション

そこで，emath では，En*などに [nuriiro=...] オプションをつけることで内部に色をつけることにしています。

¥En*の [nuriiro=..] オプション

```
¥begin{picture}(50,50)
  ¥En*[nuriiro=red]{(25,25)}{20}
¥end{picture}
```

→

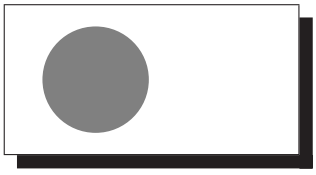


注 現時点では，[nuriiro=..] オプションではなく，¥color コマンドを用いても，同様の結果を得ることができます。

暫定仕様

```
¥begin{picture}(50,50)
  ¥color{red}¥En*{(25,25)}{20}
¥end{picture}
```

→



しかし，将来的にはこの方式は破産する予感がありますので，[nuriiro=..] オプションの使用を推奨します。

7.7.3 円周の色は？

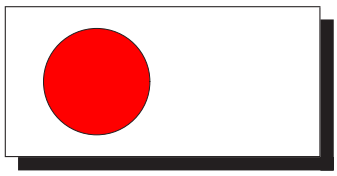
円の内部は色付けができましたが，円周は黒のままですねぇ〜。これは tpic specials の仕様かも知れません。いや，そもそも tpic specials はカラー対応しているんでしょうか。浅学にしてわかりません。(^^ゞ

そこで何とかの上塗りをすることとします。

¥En*の [border=..] オプション

```
¥begin{picture}(50,50)
  ¥En*[nuriiro=red,border=red]{(25,25)}{20}
¥end{picture}
```

→



[border=red] オプションで円周を赤で上塗りしました。円周も赤になったかに見えますが，よく見ると元の黒が一部はみ出していますね。赤の上塗りをもう少し線を太くすれば良いかもしれませんが.....

しかし，労多くして何とやらという気もします。カラーはやはり PostScript で扱うべきものでしょうか。

注 tpic specials でも

円・楕円
と
多角形

では，結果が異なり，前者は境界が黒で描画されますが，後者は境界は描画されません。どうも，tpic specials でカラーを扱うのは気が進みませんね。

¥Nuritubusi の場合

```
¥begin{picture}(50,50)
  ¥Nuritubusi[nuriiro=red]{%
    (0,0)(50,0)(50,50)(10,40)}
¥end{picture}
```

→

7.7.4 PostScript では

以上の考察から，カラー塗りつぶしは PostScript --- pszahyou **環境**を使用するのがよさそうです。

8　斜線塗り (1)

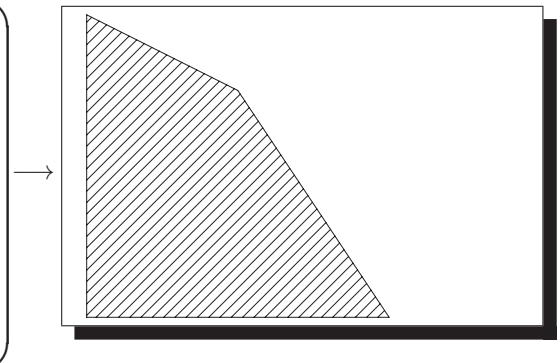
前節のコマンドの末尾に更に * を付加すると，斜線塗りとなります.

8.1　多角形

まずは，多角形の内部に斜線を引いてみます.

多角形の内部に斜線

```
¥begin{picture}(4,4)
¥def¥A{(0,0)}%
¥def¥B{(0,4)}%
¥def¥C{(2,3)}%
¥def¥D{(4,0)}%
¥Nuritubusi*{¥A¥B¥C¥D¥A}%
¥Drawline{¥A¥B¥C¥D¥A}%
¥end{picture}
```



¥Nuritubusi* の書式は

¥Nuritubusi*[#1]<#2>#3

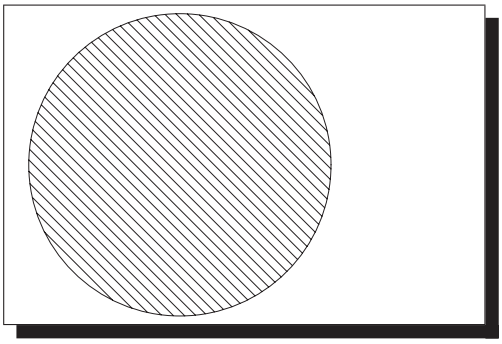
- #1 : **斜線の方向角**
  （ -90 **と** 90 **の間の数（単位は度）．デフォルトは** 45）
- #2 : **斜線の間隔　（デフォルトは** 0.125 **）**
- #3 : **多角形の点列（閉じていなければなりません．）**

8.2　円

円の内部を斜線塗りする例です．[-45] オプションをつけて，斜線の方向角を変更してみました.

円の内部に斜線

```
¥begin{picture}(4,4)
¥def¥A{(2,2)}%
¥En**[-45]¥A{2}%
¥En¥A{2}%
¥end{picture}
```



¥En** の書式です.

¥En**[#1]<#2>#3#4
　#1 : **斜線の方向角**
　　(-90 **と** 90 **の間の数（単位は度）. デフォルトは** 45)
　#2 : **斜線の間隔 （デフォルトは** 0.125 **）**
　#3 : **中心の座標**
　#4 : **半径**

8.3 扇形など

¥Daen**, ¥yumigata**, ¥ougigata**, ¥Daenko**, ¥Nuritubusi* も同様です.
代表的に ¥yumigata** の書式を記します。

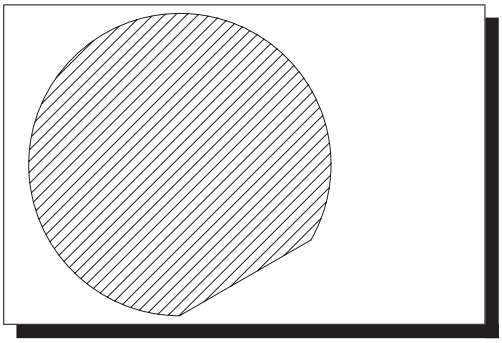
¥yumigata**[#1]<#2>#3#4#5
　#1 : **斜線塗りの傾斜角（デフォルトは** 45**）**
　#2 : **オプション**
　　<border=1> **で境界線を描画する。**
　　<syanurisiteiten=xx> **で斜線群が指定点** xx **を通るようにする。**
　　<syanurikankaku=xx> **斜線の間隔を指定する無名数（デフォルトは** 0.125**）**
　　#2 **に単に無名数を記述すれば，斜線の間隔を変更する指示と解釈される。**
　#3 **〜** #5 : ¥yumigata **の** #1〜#3 **と同じ**

（注）始め角と終り角の与え方の順序について
2 つの角を指定すると，円は 2 つの弓形に分割されます。
そのどちらが描画されるのかという話です。
始め角で指定された動径を正の向きに（時計の針と反対向き）
回転して終り角で指定した動径と重なるまでを描画します。

弓形の斜線塗り

```
¥begin{picture}(4,4)
¥def¥A{(2,2)}%
¥Put¥A{%
  ¥yumigata**{2}{-30}{270}%
  ¥yumigata{2}{-30}{270}}%
¥end{picture}
```

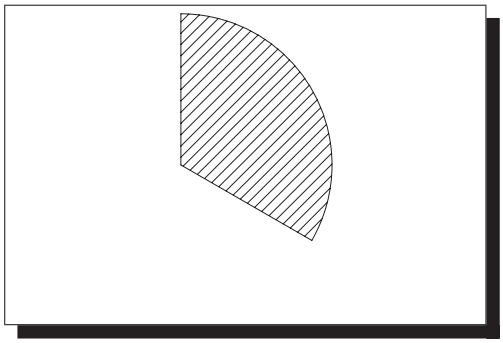
→



扇形の斜線塗り

```
¥begin{picture}(4,4)
¥def¥A{(2,2)}%
¥Put¥A{%
  ¥ougigata**{2}{-30}{90}%
  ¥ougigata{2}{-30}{90}}%
¥end{picture}
```

→

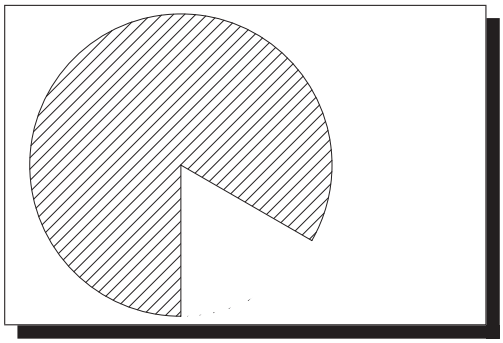


ただし，扇形の中心角は 180 度以下でなければなりません．優角の扇形の斜線塗りは，円を斜線塗りした後，余分の扇形を白塗りします．

扇形の斜線塗り（優角の場合）

```
¥begin{picture}(4,4)
¥def¥A{(2,2)}%
¥En**¥A{2}%
¥Put¥A{%
¥ougigata*[0]{2}{-90}{-30}%
¥ougigata{2}{-30}{270}}%
¥end{picture}
```

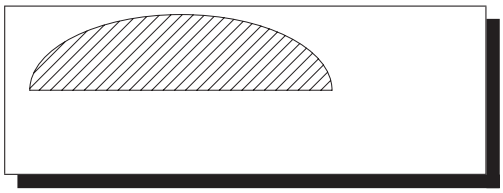
→



弓形（楕円弧）の斜線塗り

```
¥begin{picture}(4,2)
¥def¥A{(2,1)}%
¥Put¥A{%
¥Daenko**{2}{1}{0}{180}%
¥drawline(-2,0)(2,0)%
¥Daenko{2}{1}{0}{180}}%
¥end{picture}
```

→



8.4 格子セルのぬりつぶし

格子を描画する ¥kousi に

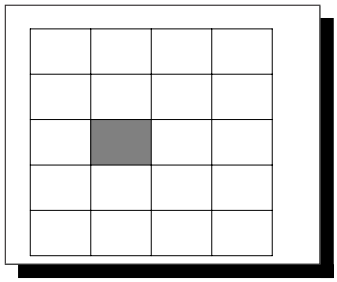
セルを塗る
セルの座標を指定しての ¥put,
黒丸配置

などのを実現する<nuri=..>オプションの説明です。<nuri=..>の右辺に，塗りつぶしを行うセルの左下コーナーの格子座標を与えます。

塗りつぶし

```
¥begin{zahyou*}[ul=4mm](0,8)(0,8)
¥kousi<nuri={(1,2)}>(2,1.5){4}{5}
¥end{zahyou*}
```

→

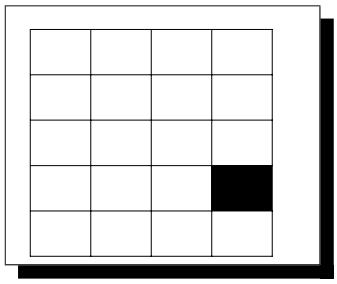


<nuri=..>の右辺は，コンマを含みますから，{...}でくくっておく必要があります。濃度指定は

濃度指定

```
¥begin{zahyou*}[ul=4mm](0,8)(0,8)
¥kousi<nuri={<thickness=1>(3,1)}>(2,1.5){4}{5}
¥end{zahyou*}
```

→

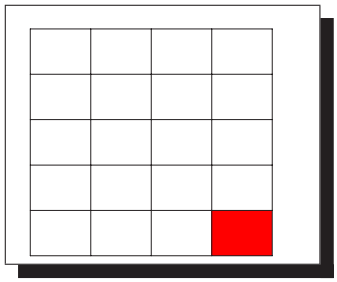


<nuri=..(x,y)>における.. の部分は ¥emPaint のオプションとして引き渡されます。赤で塗りつぶしてみましょう。

カラー塗り

```
¥begin{zahyou*}[ul=4mm](0,8)(0,8)
¥kousi<nuri={<nuriiro=red>(3,0)}>(2,1.5){4}{5}
¥end{zahyou*}
```

→

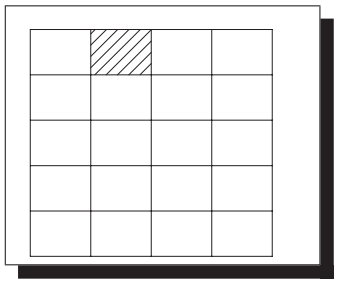


ということなれば，斜線塗りは

斜線塗り

```
¥begin{zahyou*}[ul=4mm](0,8)(0,8)
¥kousi<nuri={*(1,4)}>(2,1.5){4}{5}
¥end{zahyou*}
```

→

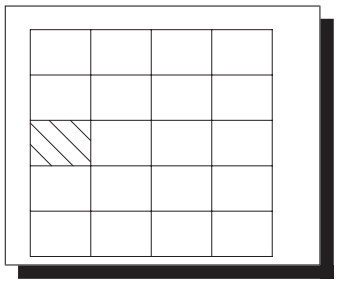


斜線の角度，間隔指定も

斜線角度，間隔

```
¥begin{zahyou*}[ul=4mm](0,8)(0,8)
¥kousi<nuri=%
  {*<slashangle=-45,%
  slashspace=.5>(0,2)}>(2,1.5){4}{5}
¥end{zahyou*}
```

→



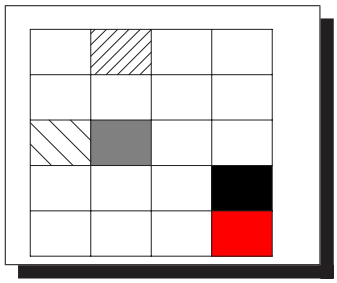
¥Nuritubusi のオプションと同様であることが納得いただけたでしょうか。

複数のセルに対して塗りを指定するには，<nuri=..>の右辺に，上記の指定を ';' で区切って並べれば実現できます。

複数指定

```
¥begin{zahyou*}[ul=4mm](0,8)(0,8)
¥kousi<nuri={%
  (1,2);%
  <thickness=1>(3,1);%
  <nuriiro=red>(3,0);%
  *(1,4);%
  *<slashangle=-45,slashspace=.5>(0,2)%
}>(2,1.5){4}{5}
¥end{zahyou*}
```

→



8.5 境界線と斜線の間を空ける

領域の境界を含まないことを示すのに，斜線と境界をくっつけず，少し空ける流儀があります。これを実現するにはいくつかの方法がありますが，田中 徹 さんが BBS #354 に投稿された方法

(1) ひとまず塗りつぶしを行う
(2) 境界線を太めの白い線で上書き
(3) 改めて境界線を引く

は有力な手法です。

そのアイデアを頂いてマクロ化してみました。emathPh.sty v 1.07 で

`syasentanmatu=...`**（単位付の長さ）**

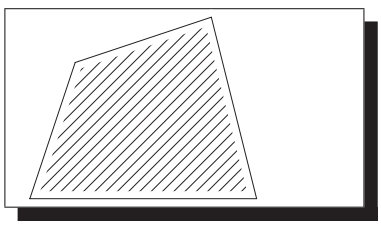
オプションにより白塗りの幅を指定することにしてみました。境界線を中心として，その左右に指定した幅を有する白い曲線を描画します。

（斜線の端点と境界線の距離が指定した長さとなり，白塗りの曲線の太さは指定した幅の 2 倍となります。）

syasentanmatu=.. オプション

```
¥begin{zahyou*}[ul=6mm](0,5)(0,4)
  ¥tenretu*{A(0,0);B(5,0);C(4,4);D(1,3)}
  ¥Nuritubusi*<syasentanmatu=1mm>%
    {¥A¥B¥C¥D¥A}
  ¥Drawline{¥A¥B¥C¥D¥A}
¥end{zahyou*}
```

→



8.6 2 円の共通部分に斜線

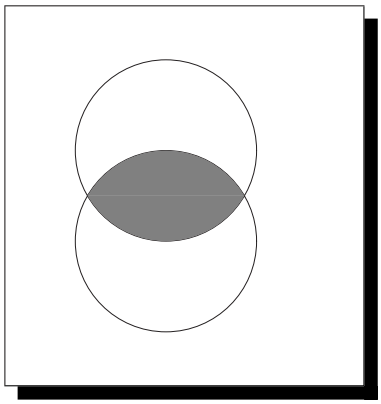
2 円の共通部分を斜線塗りするのは，結構面倒です。

まずは斜線塗りではなく，ベタ塗りの場合を検討します。これは，共通弦で 2 分割したそれぞれの弓形を塗りつぶします。

2 円の共通部分のベタ塗り

```
¥unitlength12mm¥drawaxisfalse
¥begin{zahyou}(-1.5,1.5)(-1.5,2.5)
  ¥tenretu*{A(0,0);B(0,1)}
  ¥edef¥hankeiA{1}
  ¥edef¥hankeiB{1}
  ¥En¥A¥hankeiA
  ¥En¥B¥hankeiB
  ¥CandC¥A¥hankeiA¥B¥hankeiB¥P¥Q
  ¥Put¥A{¥yumigata*¥hankeiA{%
    hazimeten=¥Q}{owariten=¥P}}
  ¥Put¥B{¥yumigata*¥hankeiB{%
    hazimeten=¥P}{owariten=¥Q}}
¥end{zahyou}
```

→

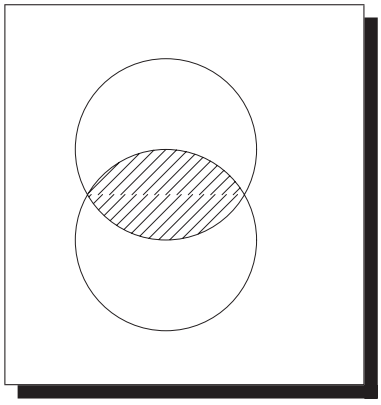


これが斜線塗りとなると

2 円の共通部分の斜線塗り (1)

```
¥unitlength12mm¥drawaxisfalse
¥begin{zahyou}(-1.5,1.5)(-1.5,2.5)
  ¥tenretu*{A(0,0);B(0,1)}
  ¥edef¥hankeiA{1}
  ¥edef¥hankeiB{1}
  ¥En¥A¥hankeiA
  ¥En¥B¥hankeiB
  ¥CandC¥A¥hankeiA¥B¥hankeiB¥P¥Q
  ¥Put¥A{¥yumigata**¥hankeiA{%
    hazimeten=¥Q}{owariten=¥P}}
  ¥Put¥B{¥yumigata**¥hankeiB{%
    hazimeten=¥P}{owariten=¥Q}}
¥end{zahyou}
```

→



2 つの弓形の斜線がつながりません。

そこで，斜線塗りに

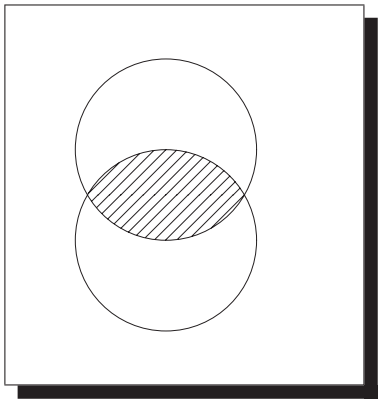
```
syanurisiteiten=
```

という書式で，斜線群が通過すべき 1 点を指定するためのオプションを付加しました。その効果は

2 円の共通部分の斜線塗り (1)

```
¥unitlength12mm¥drawaxisfalse
¥begin{zahyou}(-1.5,1.5)(-1.5,2.5)
  ¥tenretu*{A(0,0);B(0,1)}
  ¥edef¥hankeiA{1}
  ¥edef¥hankeiB{1}
  ¥En¥A¥hankeiA
  ¥En¥B¥hankeiB
  ¥CandC¥A¥hankeiA¥B¥hankeiB¥P¥Q
  ¥Put¥A{¥yumigata**<%
    syanurisiteiten=¥Q>¥hankeiA{%
    hazimeten=¥Q}{owariten=¥P}}
  ¥Put¥B{¥yumigata**<%
    syanurisiteiten=¥Q>¥hankeiB{%
    hazimeten=¥P}{owariten=¥Q}}
¥end{zahyou}
```

→



一見つながったように見えますが，よく目を凝らすとちょっとずれています。これが気になれば，斜線塗りする領域の境界線を多角形近似して，多角形を斜線塗りするコマンド ¥Nuritubusi を使うことになりましょう。円弧を折れ線近似するには，

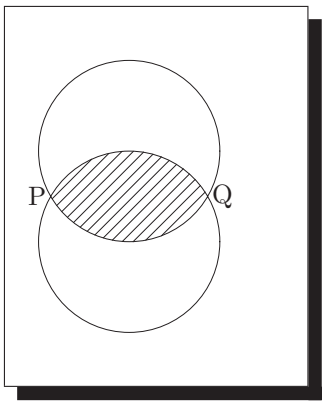
n次関数のグラフを折れ線近似する ¥KinziOresen
一般の関数のグラフを折れ線近似する ¥yKinziOresen
媒介変数表示の曲線を折れ線近似する ¥bKinziOresen

などを用いれば可能ですが，この際，円弧を折れ線近似するコマンドを作ってみました。名付けて ¥KinziEnko 書式はあとで記述することにして，とりあえず使用効果を見てみましょう。

2 円の共通部分の斜線塗り (1)

```
¥unitlength12mm¥drawaxisfalse
¥begin{zahyou}(-1.1,1.1)(-1.5,2.5)
  ¥tenretu*{A(0,0);B(0,1)}
  ¥edef¥hankeiA{1}¥edef¥hankeiB{1}
  ¥En¥A¥hankeiA¥En¥B¥hankeiB
  ¥CandC¥A¥hankeiA¥B¥hankeiB¥P¥Q
  ¥Put¥P[w]{P}¥Put¥Q[e]{Q}
  ¥KinziEnko¥A¥hankeiA{hazimeten=¥Q}{%
    owariten=¥P}¥oresenA
  ¥KinziEnko¥B¥hankeiB{hazimeten=¥P}{%
    owariten=¥Q}¥oresenB
  ¥edef¥oresen{¥Q¥oresenA¥P¥oresenB¥Q}
  ¥Nuritubusi*¥oresen
¥end{zahyou}
```

→



きれいに斜線塗りができたようです。

¥KinziEnko の書式です。

円弧を近似する折れ線

¥KinziEnko<#1>#2#3#4#5#6

#1 : **刻み値**

#2 : **中心**

#3 : **半径を直接与えるか**

tuukaten=xx **として，円弧の周上の一点を与える**

#4 : **始め角を直接与えるか**

hazimeten=xx **として，中心を始点，**xx **を終点とするベクトルの方向角を 始め角とするように指定する。**

#5 : **終り角を直接与えるか**

owariten=xx **として，中心を始点，**xx **を終点とするベクトルの方向角を 終り角とするように指定する。**

#6 : **近似折れ線を受け取る制御綴**

8.7 破線による斜線塗り

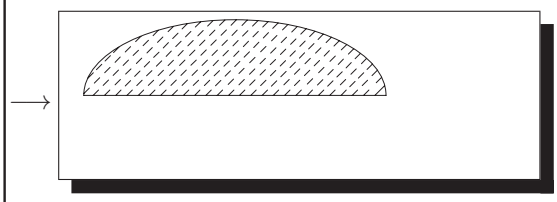
斜線を破線で引くには

```
¥def¥sensyu{¥hasen}%
```

などとします.

破線による斜線塗り

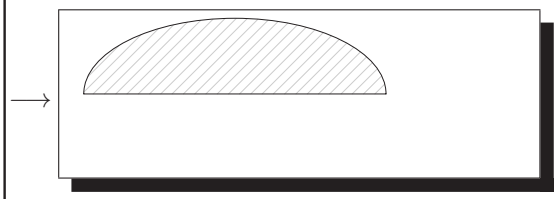
```
¥begin{picture}(4,2)
¥def¥A{(2,1)}%
¥Put¥A{{%
¥def¥sensyu{¥hasen}%
¥Daenko**{2}{1}{0}{180}}%
¥drawline(-2,0)(2,0)%
¥Daenko{2}{1}{0}{180}}%
¥end{picture}
```



なお，斜線を ¥color{lightgray} などとするのも有力な手法です．ただし，gray の濃度はプリンタ環境によって調整する必要があります.

グレースケールによる斜線塗り

```
¥definecolor{%
lightgray}{gray}{.8}%
¥begin{picture}(4,2)
¥def¥A{(2,1)}%
¥Put¥A{{%
¥color{lightgray}%
¥Daenko**{2}{1}{0}{180}}%
¥drawline(-2,0)(2,0)%
¥Daenko{2}{1}{0}{180}}%
¥end{picture}
```



8.8 斜線塗りの制約条項

斜線塗りの制約条項 斜線の方向角を θ として，直線群 $y\cos\theta - x\sin\theta = k$ が領域によってきり取られる線分は単一の連結な線分であること．

駄目な場合の典型は，ドーナッツです．これはどのような方向で切っても中心を通る直線がドーナッツによって切り取られる部分は２個の線分になってしまします．このような領域の斜線塗りはいっぺんにはできません．ドーナッツの場合は，大きい円の内部全部を一旦斜線塗りしたうえで，小さい円の内部を白塗りすることで実現できます．

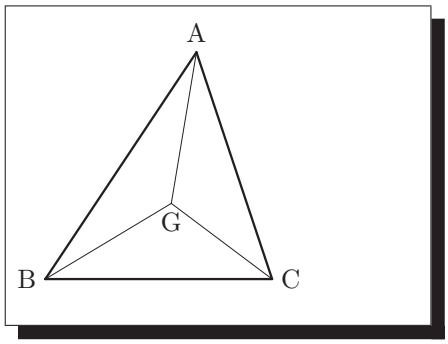
9　三角形の五心

9.1　重心

重心

```
¥begin{picture}%
(3.4,4)(-.2,-.5)%
¥def¥B{(0,0)}%
¥def¥C{(3,0)}%
¥def¥A{(2,3)}%
¥Put¥A{¥makebox(0,0.5){A}}%
¥Put¥B{¥makebox(0,0)[r]{B }}%
¥Put¥C{¥makebox(0,0)[l]{ C}}%
¥Zyuusin¥A¥B¥C¥G
¥Put¥G{¥makebox(0,-0.5){G}}%
¥Drawline{¥A¥G}%
¥Drawline{¥B¥G}%
¥Drawline{¥C¥G}%
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥thinlines
¥end{picture}
```

→



¥Zyuusin#1#2#3#4

#1,#2,#3 を頂点とする三角形の重心を #4 にセットします.

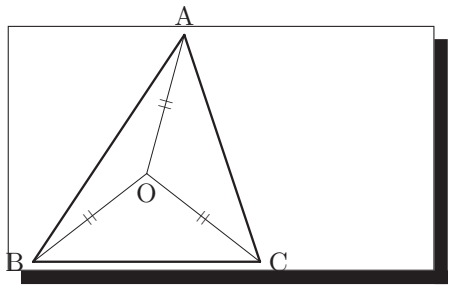
9.2　外心

三角形の外心を求めるコマンド ¥Gaisin です.

外心

```
¥begin{picture}(3,3)%
¥def¥B{(0,0)}%
¥def¥C{(3,0)}%
¥def¥A{(2,3)}%
¥Put¥A{¥makebox(0,0.5){A}}%
¥Put¥B{¥makebox(0,0)[r]{B }}%
¥Put¥C{¥makebox(0,0)[l]{ C}}%
¥Gaisin¥A¥B¥C¥O
¥Put¥O{¥makebox(0,-0.5){O}}%
¥Drawline{¥A¥O}%
¥Drawline{¥B¥O}%
¥Drawline{¥C¥O}%
¥Touhenkigou<2>¥O¥A
¥Touhenkigou<2>¥O¥B
¥Touhenkigou<2>¥O¥C
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥thinlines
¥end{picture}
```

→



¥Gaisin#1#2#3#4

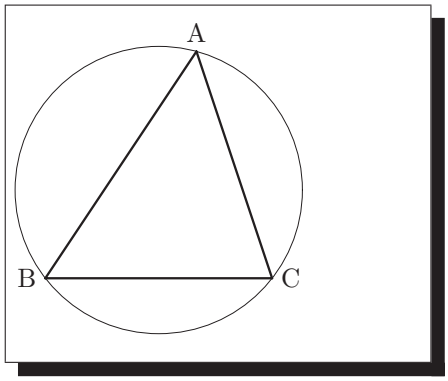
#1,#2,#3 を頂点とする三角形の外心を #4 にセットします.

三角形の外接円を描くコマンド ¥Gaisetuen もあります.

外接円

```
¥begin{picture}%
(3.2,4.5)(-.2,-1)%
¥def¥B{(0,0)}%
¥def¥C{(3,0)}%
¥def¥A{(2,3)}%
¥Put¥A{¥makebox(0,0.5){A}}%
¥Put¥B{¥makebox(0,0)[r]{B }}%
¥Put¥C{¥makebox(0,0)[l]{ C}}%
¥Gaisetuen¥A¥B¥C
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥thinlines
¥end{picture}
```

→



¥Gaisetuen#1#2#3

#1,#2,#3 **を頂点とする三角形の外接円を描画します．**

外心は ¥vGaisin **にセット，半径は** ¥lR

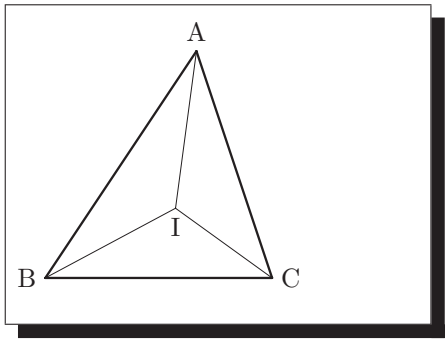
9.3 内心

三角形の内心を求めるコマンド ¥Naisin です．

内心

```
¥begin{picture}(3,4)(-.2,-.5)
¥def¥B{(0,0)}%
¥def¥C{(3,0)}%
¥def¥A{(2,3)}%
¥Put¥A{¥makebox(0,0.5){A}}%
¥Put¥B{¥makebox(0,0)[r]{B }}%
¥Put¥C{¥makebox(0,0)[l]{ C}}%
¥Naisin¥A¥B¥C¥I
¥Put¥I{¥makebox(0,-0.5){I}}%
¥Drawline{¥A¥I}%
¥Drawline{¥B¥I}%
¥Drawline{¥C¥I}%
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥thinlines
¥end{picture}
```

→



¥Naisin#1#2#3#4

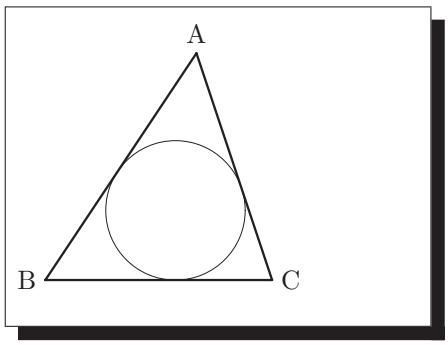
#1,#2,#3 を頂点とする三角形の内心を #4 にセットします.

三角形の内接円を描くコマンド ¥Naisetuen もあります.

内接円

```
¥begin{picture}(3,4)(-.2,-.5)
¥def¥B{(0,0)}%
¥def¥C{(3,0)}%
¥def¥A{(2,3)}%
¥Put¥A{¥makebox(0,0.5){A}}%
¥Put¥B{¥makebox(0,0)[r]{B }}%
¥Put¥C{¥makebox(0,0)[l]{ C}}%
¥Naisetuen¥A¥B¥C
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥thinlines
¥end{picture}
```

→



¥Naisetuen#1#2#3

#1,#2,#3 **を頂点とする三角形の内接円を描画します．**

内心は ¥vNaisin **にセット，半径は** ¥lr

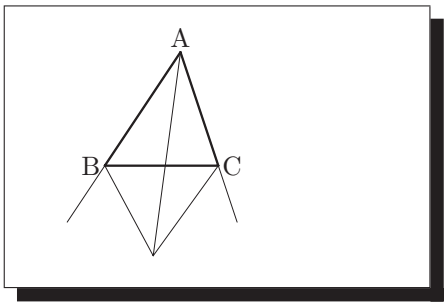
9.4 傍心

三角形の傍心を求めるコマンド ¥Bousin です．

傍心

```
¥begin{zahyou*}[ul=5mm](-2,5)(-3,4)
¥tenretu{A(2,3)n;B(0,0)w;C(3,0)e}
¥Bunten¥A¥B3{-1}¥P
¥Bunten¥A¥C3{-1}¥Q
¥Bousin¥A¥B¥C¥J
¥Drawlines{¥A¥J;¥B¥J;¥C¥J;¥P¥A¥Q}
¥thicklines
¥Takakkei{¥A¥B¥C}%
¥thinlines
¥end{zahyou*}
```

→



¥Bousin#1#2#3#4

#1,#2,#3 **を頂点とする三角形の，**

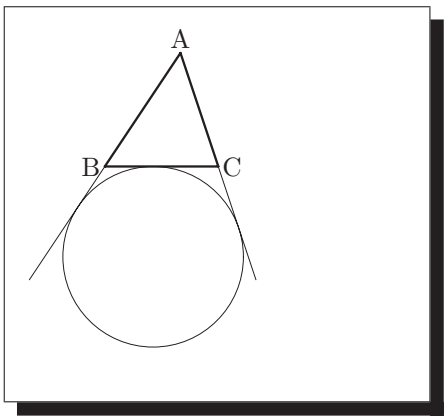
∠ A **内にある傍心を** #4 **にセットします．**

三角形の傍接円を描くコマンド ¥Bousetuen もあります．

傍接円

```
¥begin{zahyou*}[ul=5mm](-2,5)(-6,4)
¥tenretu{A(2,3)n;B(0,0)w;C(3,0)e}
¥Bunten¥A¥B2{-1}¥P
¥Bunten¥A¥C2{-1}¥Q
¥Bousetuen¥A¥B¥C
¥Drawlines{¥P¥A¥Q}
¥thicklines
¥Takakkei{¥A¥B¥C}%
¥thinlines
¥end{zahyou*}
```

→



¥Bousetuen#1#2#3

#1,#2,#3 **を頂点とする三角形の傍接円を描画します．**

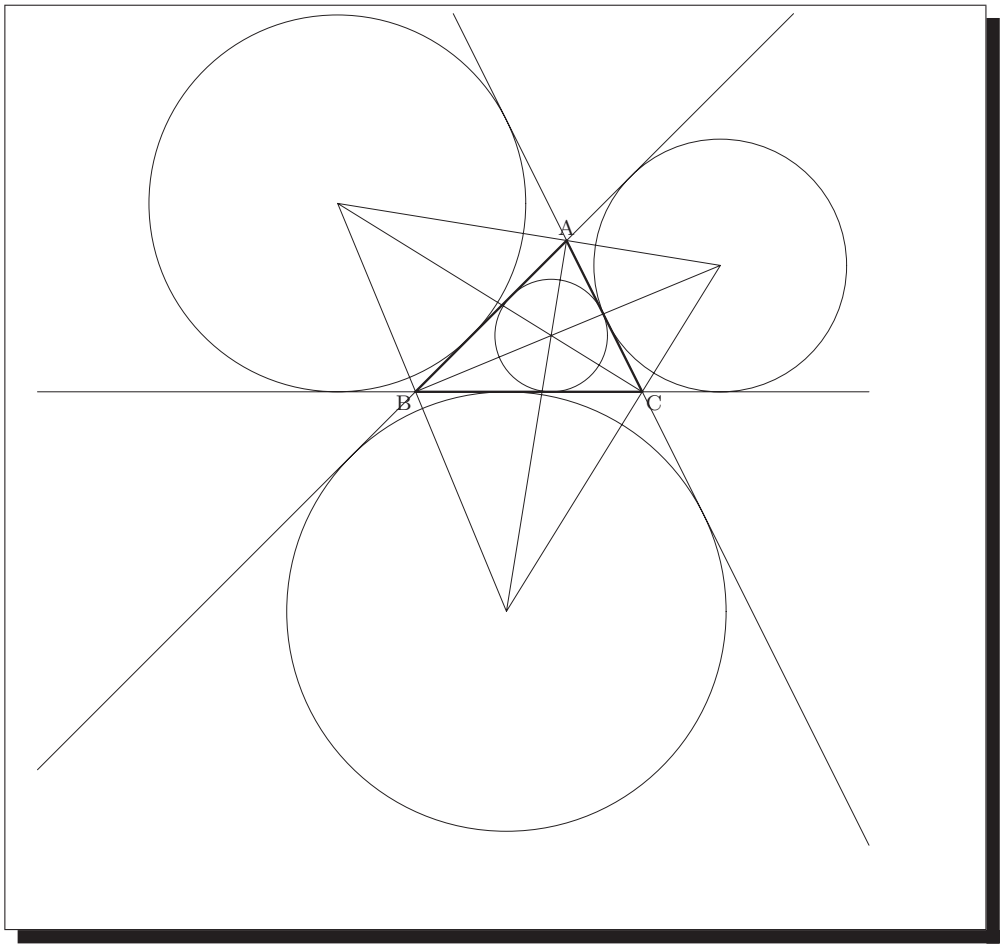
傍心は ¥vBousin **にセット，半径は** ¥BousetuenHankei

三角形の内接円，3 つの傍接円を描画する例です。

内接円と傍接円

```
¥begin{footnotesize}
      ¥begin{zahyou*}[ul=10mm](-5,6)(-7,5)
        ¥tenretu{A(2,2)n;B(0,0)sw;C(3,0)se}
        ¥Naisetuen¥A¥B¥C
        ¥Bousetuen¥A¥B¥C¥edef¥J{¥vBousin}
        ¥Bousetuen¥B¥C¥A¥edef¥K{¥vBousin}
        ¥Bousetuen¥C¥A¥B¥edef¥L{¥vBousin}
        ¥Takakkei{¥J¥K¥L}
        ¥Drawlines{¥J¥A;¥K¥B;¥L¥C}
        ¥Tyokusen¥A¥B{}{}
        ¥Tyokusen¥B¥C{}{}
        ¥Tyokusen¥C¥A{}{}
        ¥thicklines
        ¥Takakkei{¥A¥B¥C}
      ¥end{zahyou*}
¥end{footnotesize}
```

→

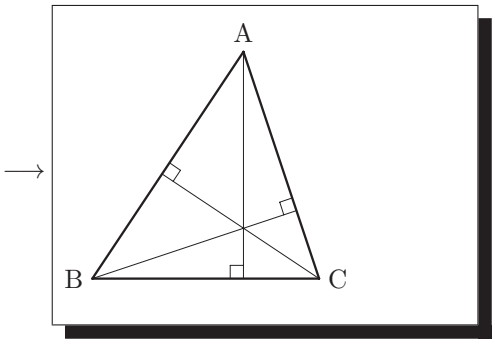


9.5 垂心

三角形の垂心を求めるコマンドは用意してありません．¥Suisen で垂線を引き，¥LandL で交点を求めることで垂心が得られます．

垂心

```
¥begin{picture}(3,4)(-.2,-.5)
¥def¥B{(0,0)}%
¥def¥C{(3,0)}%
¥def¥A{(2,3)}%
¥Put¥A{¥makebox(0,0.5){A}}%
¥Put¥B{¥makebox(0,0)[r]{B }}%
¥Put¥C{¥makebox(0,0)[l]{ C}}%
¥Suisen¥B¥C¥A¥E
¥Suisen¥C¥A¥B¥F
¥LandL¥B¥E¥C¥F¥H
¥LandL¥A¥H¥B¥C¥D
¥Drawline{¥A¥D}%
¥Drawline{¥B¥E}%
¥Drawline{¥C¥F}%
¥Tyokkakukigou¥A¥D¥B
¥Tyokkakukigou¥B¥E¥A
¥Tyokkakukigou¥C¥F¥A
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥thinlines
¥end{picture}
```



9.6 角の二等分線

三角形の二等分線と対辺の交点を求めるには，コマンド ¥Nitoubunsen を用います。その書式です。

三角形の二等分線と対辺の交点を求める。

¥Nitoubunsen[#1]#2#3#4#5

角#2#3#4**（**#3 **が角の頂点）を** 2 **等分する直線が辺**#2#4 **と交わる点を**#5 **に与える。**

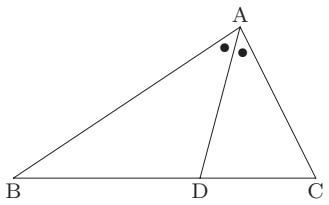
#1 **を与えたときは，外角の** 2 **等分線を**#1 **に与える。**

まずは，内角の二等分線です。

¥Nitoubunsen

```
¥begin{zahyou*}[ul=10mm](-1,5)(-1,3)¥footnotesize
  ¥tenretu{A(3,2)n;B(0,0)s;C(4,0)s}
  ¥Nitoubunsen¥B¥A¥C¥D
  ¥Put¥D[s]{D}
  ¥Drawline{¥A¥B¥C¥A¥D}
  ¥Kakukigou<0>¥B¥A¥D(0,0)[c]{$¥bullet$}
  ¥Kakukigou<0>¥D¥A¥C(0,0)[c]{$¥bullet$}
¥end{zahyou*}
```

↓

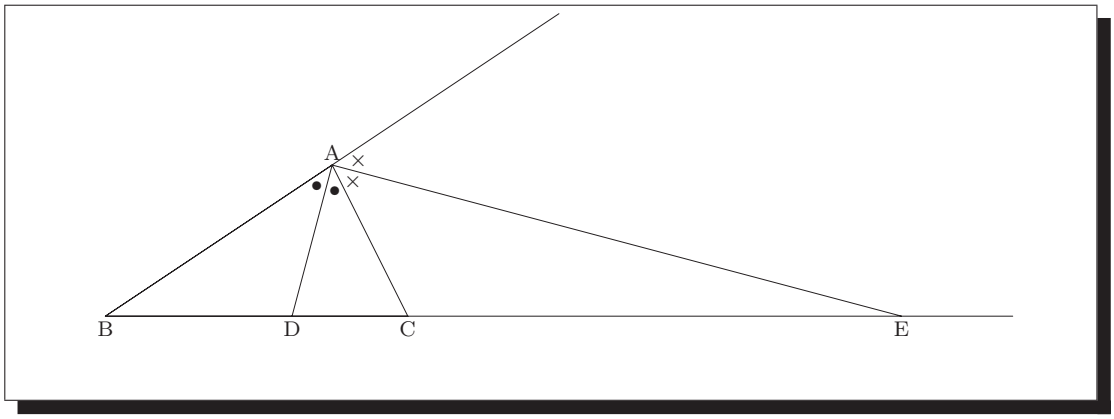


外角の二等分線と対辺（の延長）との交点を得るには，オプション引数 [#1] に交点を受け取る制御綴を与えます。

外角の二等分線

```
¥begin{zahyou*}[ul=10mm](-1,12)(-1,4)¥footnotesize
  ¥tenretu{A(3,2)n;B(0,0)s;C(4,0)s}
  ¥Nitoubunsen[¥E]¥B¥A¥C¥D
  ¥Put¥D[s]{D}
  ¥Put¥E[s]{E}
  ¥Hantyokusen¥B¥C
  ¥Hantyokusen¥B¥A
  ¥Drawlines{¥A¥B¥C¥A¥D;¥A¥E}
  ¥Kakukigou<0>¥B¥A¥D(0,0)[c]{$¥bullet$}
  ¥Kakukigou<0>¥D¥A¥C(0,0)[c]{$¥bullet$}
  ¥Bunten¥A¥B{-1}{2}¥T
  ¥Kakukigou<0>¥C¥A¥E(0,0)[c]{$¥times$}
  ¥Kakukigou<0>¥E¥A¥T(0,0)[c]{$¥times$}
¥end{zahyou*}
```

↓


A
B
D
C
E

10　三角形の決定

10.1　三辺

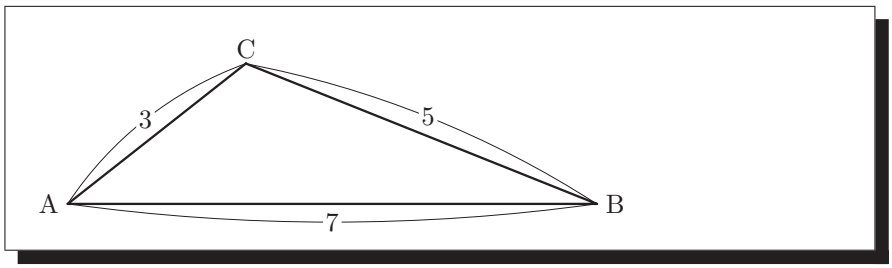
三辺の長さがわかっているときに，その三角形をどのように描画するか，という話です.

具体例として AB = 7, BC = 5, CA = 3 としましょう．まず A(0,0), B(7,0) とします．点 C は，A を中心とする半径 3 の円と，B を中心とする半径 5 の円との交点として求めます.

三角形の決定 (1) 三辺

```
¥begin{picture}(7,3)(-.5,-.5)
¥def¥A{(0,0)}%
¥def¥B{(7,0)}%
¥CandC¥A{3}¥B{5}¥D¥C
¥Put¥A{¥makebox(0,0)[r]{A }}%
¥Put¥B{¥makebox(0,0)[l]{ B}}%
¥Put¥C{¥makebox(0,0.4){C}}%
¥HenKo<.4>¥A¥B{7}%
¥HenKo<.4>¥B¥C{5}%
¥HenKo<.4>¥C¥A{3}%
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥thinlines
¥end{picture}
```

→



10.2　二角夾辺

BC = 5, ∠B = 60°, ∠C = 45° としましょう．これは，B(0,0), C(5,0) とした上で

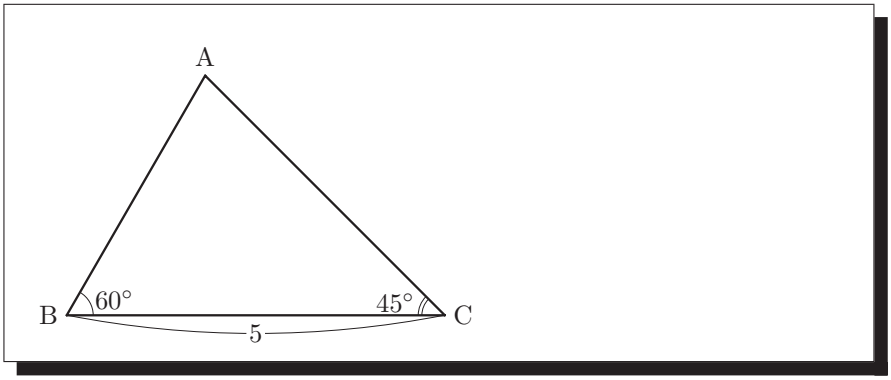
点 B を通り，方向角 60° の直線と
点 C を通り，方向角 135° の直線

の交点として A を求めます.

三角形の決定 (2) 二角夾辺

```
¥begin{picture}(7,4.5)(-0.5,-0.5)%
¥def¥B{(0,0)}%
¥def¥C{(5,0)}%
¥kandk¥B{60}¥C{135}¥A
¥Put¥A{¥makebox(0,0.5){A}}%
¥Put¥B{¥makebox(0,0)[r]{B }}%
¥Put¥C{¥makebox(0,0)[l]{ C}}%
¥HenKo<.4>¥B¥C{5}%
¥Kakukigou¥C¥B¥A(2pt,1pt)[l]{60¥Deg}%
¥Kakukigou<2>¥A¥C¥B(-2pt,1pt)[r]{45¥Deg}%
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥thinlines
¥end{picture}
```

→



10.3 二辺夾角

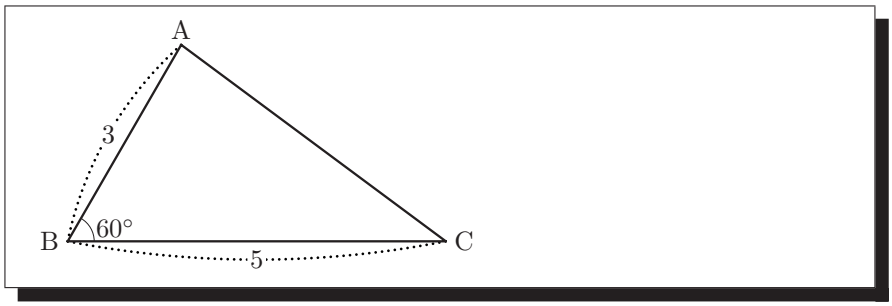
AB = 3, BC = 5, ∠B = 60° としましょう．これは B(0,0), C(5,0) とした上で A の座標を極座標→直交座標変換で求めます．すなわち

```
¥kyokuTyoku(3,60)¥A
```

三角形の決定 (3) 二辺夾角

```
¥unitlength1cm
¥begin{picture}(5.5,3.5)(-0.5,-0.5)
¥def¥B{(0,0)}%
¥def¥C{(5,0)}%
¥kyokuTyoku(3,60)¥A
¥Put¥A{¥makebox(0,0.4){A}}%
¥Put¥B{¥makebox(0,0)[r]{B }}%
¥Put¥C{¥makebox(0,0)[l]{ C}}%
¥HenKo[60]<.4>¥B¥C{5}%
¥HenKo[36]<.4>¥A¥B{3}%
¥Kakukigou¥C¥B¥A(2pt,0)[l]{60¥Deg}%
¥thicklines
¥Drawline{¥A¥B¥C¥A}%
¥thinlines
¥end{picture}
```

$\longrightarrow$



11　正弦定理・余弦定理

11.1　正弦定理

正弦定理を用いて三角形の辺・角，外接円の半径を求めるコマンドを解説します．一例として，

$A = 120°$, $B = 15°$, $c = 10$

である三角形を BC を底辺として描画することを考えてみます．そのために BC $= a$ を正弦定理で求めます．

まずは $C = 45°$ と $c = 10$ から

$$\frac{c}{\sin C} = 2R$$

を用いて外接円の半径を求めます．そのためのコマンドが ¥seigenR です．書式は

¥seigenR#1#2
　#1 : **向かい合った辺・角のうち辺**
　#2 : **角**
　外接円の直径が ¥lRR，**半径が** ¥lR **にセットされる．**

ここでは，¥seigenR{10}{45} とします．

次いで，

$$\frac{a}{\sin A} = 2R$$

を用いて a を求めます．そのためのコマンドが ¥seigen です．

¥seigen#1#2
　#1 : **角**
　#2 : **角と向かい合った辺の長さを受取るコントロールシーケンス**

ここでは ¥seigen{120}¥la として ¥la に a がセットされます．

関連して，逆に辺を与えて角を求めるコマンドが ¥Seigen です．

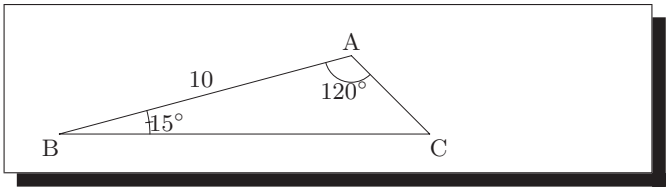
¥Seigen[#1]#2#3
　#2 : a
　#3 : A **を受取るコントロールシーケンス**
　#1 = e **のときは，角** A **（鋭角）を求める．（デフォルト）**
　#1 = d **のときは，角** A **（鈍角）を求める．**

では，この節のはじめに取り上げた三角形を実際に描画してみましょう．

正弦定理

```
{¥unitlength4mm¥small¥drawaxisfalse
¥begin{zahyou}(-1,15)(-1,4)%
¥seigenR{10}{45}%
¥seigen{120}¥la
¥def¥B{(0,0)}%
¥def¥C{(¥la,0)}%
¥kyokuTyoku(10,15)¥A
¥Put¥A(0,2pt)[b]{A}%
¥Put¥B(0,-2pt)[rt]{B}%
¥Put¥C(0,-2pt)[lt]{C}%
¥HenKo[0]¥A¥B{10}%
¥Kakukigou[|]¥C¥B¥A<hankei=3>(0,0)[l]{15¥Deg}%
¥Kakukigou¥B¥A¥C(0,0)[t]{120¥Deg}%
¥Drawline{¥A¥B¥C¥A}%
¥end{zahyou}}
```

⟶



11.2 余弦定理

2 辺夾角がわかっているときに，第 3 辺を求めるコマンドが ¥yogen です．

2 辺夾角から第 3 辺の長さを求める．

¥yogen#1#2#3#4

#1=b，#2=c，#3=A → a を#4 にセットする．

では，このコマンドを用いて

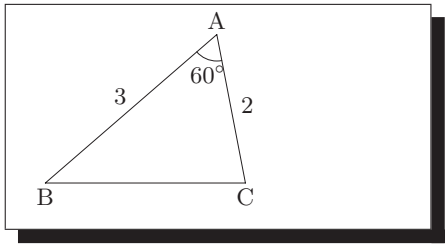
$$b = 2,\ c = 3,\ A = 60^\circ$$

である三角形を BC を底辺として描画してみます．

余弦定理

```
{¥unitlength10mm¥small¥drawaxisfalse
¥begin{zahyou}(-.2,3)(-.5,2.25)%
¥yogen{2}{3}{60}¥la
¥def¥B{(0,0)}%
¥def¥C{(¥la,0)}%
¥CandC¥B{3}¥C{2}¥AA¥A
¥Put¥A(0,2pt)[b]{A}%
¥Put¥B(0,-2pt)[t]{B}%
¥Put¥C(0,-2pt)[t]{C}%
¥HenKo[0]¥C¥A{2}%
¥HenKo[0]¥A¥B{3}%
¥Kakukigou¥B¥A¥C(0,-2pt)[t]{60¥Deg}%
¥Drawline{¥A¥B¥C¥A}%
¥end{zahyou}}%
```

⟶



関連して，3 辺の長さがわかっているとき，角を求めるコマンドが ¥Yogen です.

3 辺の長さから角の余弦を求める.

```
¥Yogen[#1]#2#3#4#5
    cos(A)=(b^2+c^2-a^2)/(2bc)
    #2=a, #3=b, #4=c 結果は #5 にセット
    #1=a のときは角を求める.(単位は度)
```

12　ベクトル

12.1　ベクトル演算

点の座標を位置ベクトルの成分と見て，ベクトル演算をするコマンドを用意しました.

和： ¥Addvec#1#2#3：2 つのベクトル#1，#2 の和ベクトルを#3 に与えます.

差： ¥Subvec#1#2#3：2 つのベクトル#1，#2 の差ベクトルを#3 に与えます.

スカラー倍： ¥Mulvec#1#2#3：ベクトル#2 のスカラー#1 倍を#3 に与えます.

大きさ： ¥Absvec#1#2: ベクトル#1 の大きさを #2 に与えます.

方向角： ¥Argvec#1#2: ベクトルが x 軸の正の向きとなす角を #2 に与えます.

法線ベクトル： ¥Nvec#1#2: ベクトル #1 に垂直な単位ベクトルを #2 に与えます.

回転： ¥Rotvec[#1]<#2>#3#4#5: ベクトル #3 を 角 #4 だけ回転したベクトルを #5 に与えます. [#1] を指定した場合は，長さを指定した値にします. また，<#2> を指定した場合は，長さを元のベクトルの #2 倍にします.

成分： ¥vecXY#1#2#3: ベクトル #1 の x 成分を #2 へ，y 成分を #3 へ抽出します.

12.2　平行四辺形

A(2,3), B(1,1), C(4,1) を頂点とする平行四辺形 ABCD を作図するには 3 点の座標 ¥A, ¥B, ¥C を位置ベクトルと見て

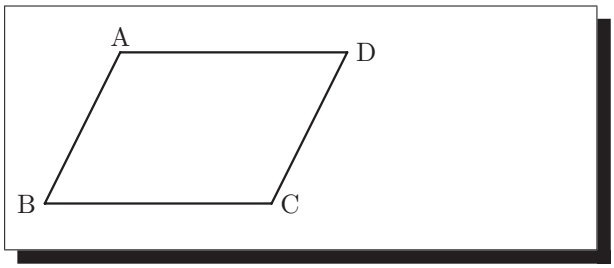
¥A + ¥C − ¥B

として ¥D を求めます.

平行四辺形

```
¥unitlength1cm
¥begin{picture}(5,3)(0.8,0.5)
¥def¥A{(2,3)}%
¥def¥B{(1,1)}%
¥def¥C{(4,1)}%
¥Addvec¥A¥C¥D
¥Subvec¥D¥B¥D
¥Put¥A{¥makebox(0,0.4){A}}%
¥Put¥B{¥makebox(0,0)[r]{B }}%
¥Put¥C{¥makebox(0,0)[l]{ C}}%
¥Put¥D{¥makebox(0,0)[l]{ D}}%
¥thicklines
¥Drawline{¥A¥B¥C¥D¥A}%
¥thinlines
¥end{picture}
```

$\longrightarrow$



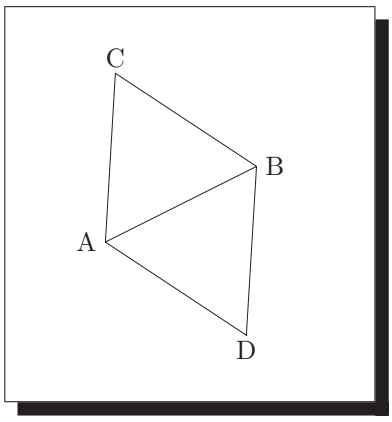
12.3 回転

ベクトルを回転させるコマンド ¥Rotvec を用いて回転を行うことができます．ここではさらに一般化した ¥Kaiten を用いて，指定した 2 点 A, B を結ぶ線分を一辺とする正三角形を作図します．

回転

```
¥begin{picture}(3,5)%
¥def¥B{(3,3)}%
¥def¥A{(1,2)}%
¥Kaiten¥A¥B{60}¥C
¥Kaiten¥A¥B{-60}¥D
¥Put¥A{¥makebox(0,0)[r]{A }}%
¥Put¥B{¥makebox(0,0)[l]{ B}}%
¥Put¥C{¥makebox(0,0.4){C}}%
¥Put¥D{¥makebox(0,-0.4){D}}%
¥Drawline{¥A¥B¥C¥A¥D¥B}%
¥end{picture}
```

$\longrightarrow$



¥Kaiten コマンドの書式です.

¥Kaiten[#1]<#2>#3#4#5#6

#1 : **長さ指定**

#2 : **長さの倍率指定**

#3 : **回転の中心**

#4 : **回転させる点**

#5 : **回転角**

#6 : **結果の座標を受け取るコントロールシーケンス**

13　座標平面

13.1　連立不等式の解を図表示

連立方程式の解を数直線上に図表示する話です。

例 1

連立不等式

```
  ¥renritu{&x+1<0 ¥¥ &x^2<4}
```

の解は

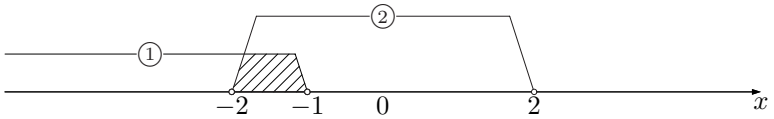
```
¥begin{zahyou*}[ul=10mm](-5,5)(-.5,1.5)%
¥ArrowLine{(¥xmin,0)}{(¥xmax,0)}%
¥Put{(¥xmax,0)}[s]{$x$}%
¥Put{(0,0)}[s]{$0$}%
¥Put{(-1,0)}[s]{$-1$}%
¥Put{(-2,0)}[s]{$-2$}%
¥Put{(2,0)}[s]{$2$}%
¥KTkukan{(,-1);(-2,2)}{(-2,-1)}
¥end{zahyou*}
```

→

連立不等式

$$\begin{cases} x+1<0 & \cdots\cdots\cdots\cdots\cdots\cdots\ ① \\ x^2<4 & \cdots\cdots\cdots\cdots\cdots\cdots\ ② \end{cases}$$

の解は



例 2

連立不等式

```
¥renritu{&x^2>x ¥label{E2-1}¥¥ &x^2¥leqq 4¥label{E2-2}}
```

の解は

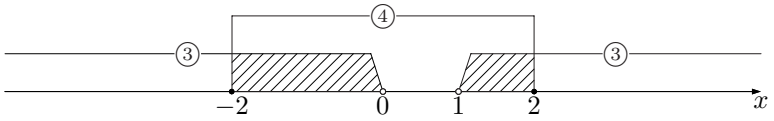
```
¥begin{zahyou*}[10mm](-5,5)(-.5,1.5)%
¥ArrowLine{(¥xmin,0)}{(¥xmax,0)}%
¥Put{(¥xmax,0)}[s]{$x$}%
¥Put{(0,0)}[s]{$0$}%
¥Put{(1,0)}[s]{$1$}%
¥Put{(-2,0)}[s]{$-2$}%
¥Put{(2,0)}[s]{$2$}%
¥KTkukan[E2-1;E2-2]{{(,0)|(1,)};[-2,2]}{[-2,0)|(1,2]}
¥end{zahyou*}
```

→

連立不等式

$$\begin{cases} x^2 > x & \cdots\cdots\cdots\cdots\cdots\cdots\ ③ \\ x^2 \leqq 4 & \cdots\cdots\cdots\cdots\cdots\cdots\ ④ \end{cases}$$

の解は



¥KTkukan の書式です。

¥KTkukan[#1]#2#3

#1 : **各区間のラベル指定オプション**

#2 : **各区間を‘;’区切りで並べる**

#3 : **結果の区間**

区間は

開区間 : (-3,5)

閉区間 : [-3,5]

半開区間 : (-3,5], [-3,5)

無限区間 : (,-3), (5,)

などと表す。

2 つの区間の和集合は (,-3)|(5,) **などのように，‘|’を用いる**

ラベル指定オプションは，

デフォルトは [auto] **で，**¥maru1，¥maru2，..... **が付与される。**

各不等式に ¥label **がついているときは，ラベル名を‘;’区切りで並べる。**

[] **と指定すれば，区間にラベルはつかない。**

下の ¥kukantakasa **を用いて，自由に配置してもよい。**

区間を表す横罫線の y **座標は** ¥kukantakasa **で，そのデフォルト値は** 0.5

層を重ねるときは，その 2，3，... **倍となる。**

13.2 zahyou 環境

座標軸を描きます.
基本的な使用法は

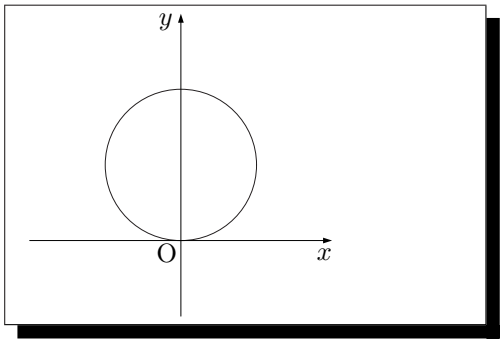
¥begin{zahyou}(x **の下限**，x **の上限**)(y **の下限**，y **の上限**)

例えば,

座標平面

```
¥begin{zahyou}(-2,2)(-1,3)
¥En{(0,1)}{1}%
¥end{zahyou}
```

→



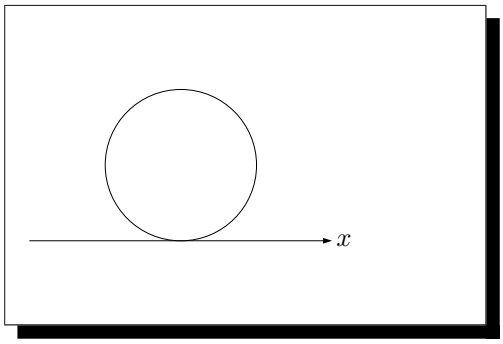
zahyou 環境は，実質 picture 環境です．ここへ，1 次関数，2 次関数のグラフを描こうという魂胆です.

なお，座標軸を描画したくないときは，*付の zahyou*環境を使用します。下の例は，¥drawXaxis で x 軸だけを描画しています。

¥drawaxisfalse

```
¥begin{zahyou*}(-2,2)(-1,3)
¥drawXaxis
¥Put{(¥xmax,0)}[e]{$x$}
¥En{(0,1)}{1}
¥end{zahyou*}
```

→



y 軸のみを描画する ¥drawYaxis も定義されています。

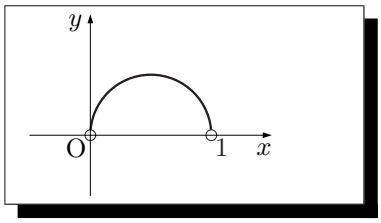
13.3 座標軸描画のタイミング

座標軸を描画するタイミングは，zahyou 環境の最後ですが，座標軸を描画した後に，座標軸上の点に ¥Siromaru で白丸をつけたいときなどはうまくありません。下の図のように，白丸の中を座標軸が突き抜けてしまいます。

座標軸描画のタイミング

```
¥begin{zahyou}[ul=16mm](-.5,1.5)(-.5,1)
  ¥tenretu*{A(0,0);B(1,0);C(.5,0)}
  {¥thicklines¥Enko¥C{.5}{0}{180}}%
  ¥siromaru[2pt]{¥A;¥B}
  ¥Put¥B[se]{1}%
¥end{zahyou}
```

→

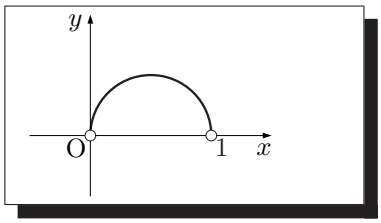


そのようなときは zahyou*環境を用いて，適切なタイミングで ¥drawXYaxis を発行して，座標軸を別途描画します。

¥drawXYaxis

```
¥begin{zahyou*}[ul=16mm](-.5,1.5)(-.5,1)
  ¥tenretu*{A(0,0);B(1,0);C(.5,0)}
  {¥thicklines¥Enko¥C{.5}{0}{180}}%
  ¥drawXYaxis
  ¥siromaru[2pt]{¥A;¥B}
  ¥Put¥B[se]{1}%
¥end{zahyou*}
```

→



13.4 zahyou 環境のオプション

zahyou 環境については，一番初めに紹介しましたが，細かい点を修正するオプションについて説明します。

zahyou 環境の細かい指定をするのに，オプション引数を用意しています。これらは

```
key = val
```

の形式で，コンマで区切ることで複数のオプションを指定できます。

13.4.1 ¥unitlength の指定

picture 環境の単位長 (¥unitlength) は，デフォルトでは 1pt となっています。これを変更するオプションが

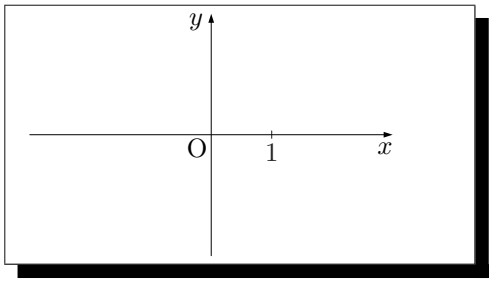
```
ul=...
```

で，右辺値は単位を伴った長さです。

ul=... オプション

```
¥begin{zahyou}[ul=8mm]%
  (-3,3)(-2,2)%
  ¥xmemori{1}%
¥end{zahyou}%
```

→



注 1. TEX が扱える実数値は

```
¥maxdimen=16383.99999pt % the largest legal <dimen>
```

が上限とされています。emathPh における計算もこの制限を受けます。距離計算では平方計算が必要ですが，128^2 でオーバーフローしてしまいます。従って座標の値を大きくしないように，¥unitlength を 1cm 前後にしておく方が良いでしょう。

注 2.

```
¥unitlength8mm
¥begin{zahyou}....

¥end{zahyou}
```

と

```
¥begin{zahyou}[ul=8mm]....

¥end{zahyou}
```

との大きな違いは，¥unitlength 変更の有効範囲です。

前者は zahyou 環境が終わった後でもこの変更が有効（残ってしまう）のに対して，後者はこの変更が当該 zahyou 環境内のみに有効である，すなわちこの zahyou 環境が終われば，¥unitlength は以前の値に戻ることです。

zahyou の [...] オプションによる変更はすべて当該 zahyou 環境内に限定されます。

13.4.2　座標軸の名称変更

デフォルトでは,

原点には O
横軸には x
縦軸には y

が表示されますが，これを変更するオプションが

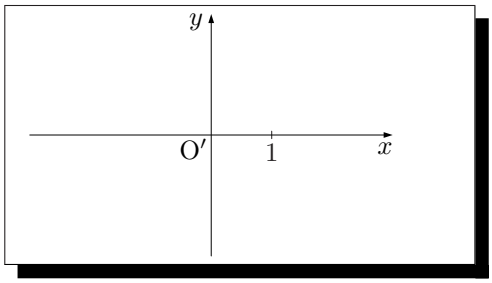
```
gentenkigou=
yokozikukigou=
tatezikukigou=
```

で，右辺値は文字列です。
原点記号を変更します。

原点記号の変更

```
¥begin{zahyou}%
  [%
    ul=8mm,
    gentenkigou=O$'$
  ]%
  (-3,3)(-2,2)%
  ¥xmemori{1}%
¥end{zahyou}%
```

→

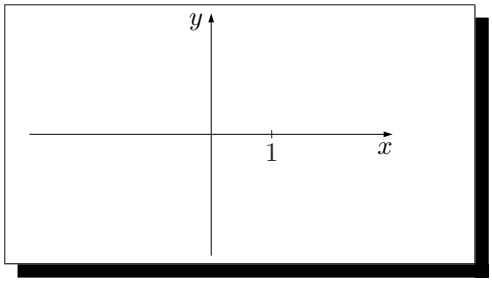


原点記号をつけたくない、という場面なら

原点記号なし

```
¥begin{zahyou}%
  [%
    ul=8mm,
    gentenkigou={}
  ]%
  (-3,3)(-2,2)%
  ¥xmemori{1}%
¥end{zahyou}%
```

→

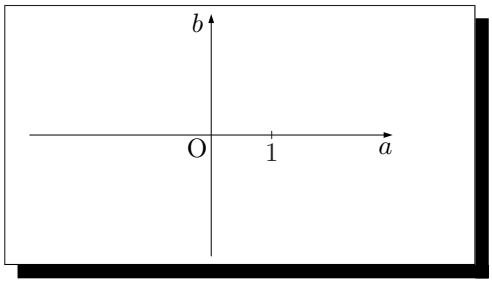


座標軸名称の変更例です。

座標軸名の変更

```
¥begin{zahyou}%
  [%
    ul=8mm,
    yokozikukigou=$a$,
    tatezikukigou=$b$
  ]%
  (-3,3)(-2,2)%
  ¥xmemori{1}%
¥end{zahyou}%
```

→



13.4.3 軸記号の配置オプション

デフォルトでは,

原点記号は 左下
横軸名は 下
縦軸名は 左

に表示されますが，これを変更するオプションが

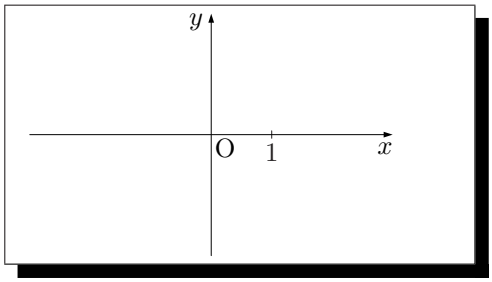
```
gentenhaiti=
yokozikuhaiti=
tatezikuhaiti=
```

で，右辺値は ¥Put のオプションと同形式です。原点記号はデフォルトでは左下に配置されますが，これを右下に変更してみましょう。

原点記号の配置変更

```
¥begin{zahyou}%
  [%
    ul=8mm,
    gentenhaiti={[se]}
  ]%
  (-3,3)(-2,2)%
  ¥xmemori{1}%
¥end{zahyou}%
```

→



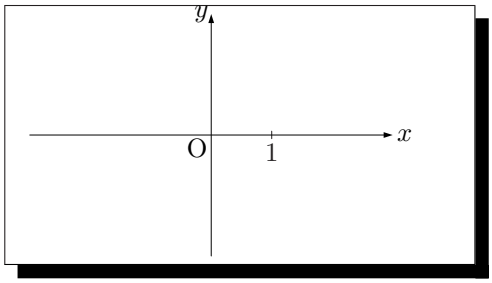
¥Put の配置オプション [se] を gentenhaiti=の右辺値に指定しますが，] が zahyou 環境に対する [...] オプションの終了と解釈されるのを防ぐため{[se]}と，{ } で括っておく必要があります。

次に座標軸名の配置を変更する例です。

座標軸名の配置変更

```
¥begin{zahyou}%
  [%
    ul=8mm,
    yokozikuhaiti={[e]},
    tatezikuhaiti={(-1pt,0)[r]}
  ]%
  (-3,3)(-2,2)%
  ¥xmemori{1}%
¥end{zahyou}%
```

→



この場合，座標軸名 x, y は zahyou 環境の外に飛び出しています。これを防ぐために，周囲に少し余白を付けておきたい，というのが後述の

```
migiyohaku=
hidariyohaku=
ueyohaku=
sitayohaku=
```

オプションです。

13.4.4 矢印のサイズ変更

座標軸の矢印を含め，zahyou 環境内の矢印のサイズを変更するオプションが

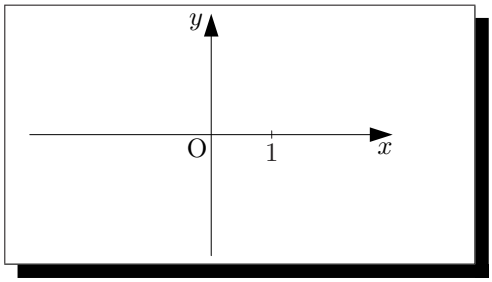
```
arrowheadsize=
```

です。

矢印のサイズ変更

```
¥begin{zahyou}%
  [%
    ul=8mm,
    arrowheadsize=2.5
  ]%
  (-3,3)(-2,2)%
  ¥xmemori{1}%
¥end{zahyou}%
```

→



矢印が大きくなりすぎましたか。arrowheadsize の右辺値を適当に変更してください。この数値は，現在のサイズの何倍にするかを指定するものです。

逆に矢印がいらない場合は，つぎの zikusensyu オプションを用います。

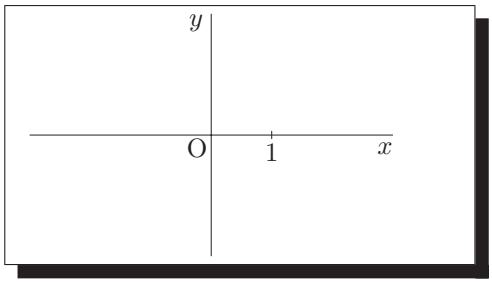
13.4.5　軸の線種変更

座標軸には，デフォルトで矢印が付加されていますが，軸の線種を変更することで矢印を付けないようにすることができます。

軸の矢印なし

```
¥begin{zahyou}%
  [%
    ul=8mm,
    zikusensyu=¥drawline
  ]%
  (-3,3)(-2,2)%
  ¥xmemori{1}%
¥end{zahyou}%
```

→

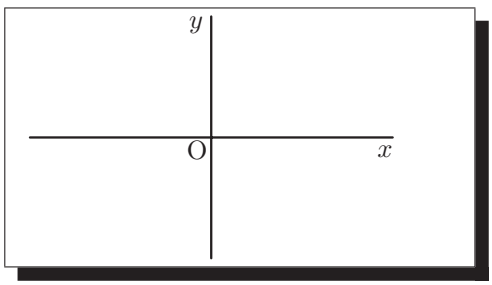


軸を太くしてみましょうか。

軸を太目に

```
¥begin{zahyou}%
  [%
    ul=8mm,
    zikusensyu=¥thicklines¥drawline
  ]%
  (-3,3)(-2,2)%
¥end{zahyou}%
```

→

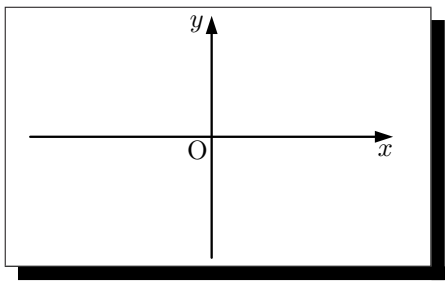


軸を太くし，矢印も付けるには，線種をデフォルトの `¥arrowdrawline` にして `¥thicklines` などをかぶせます。

軸を太目に，矢印も

```
¥begin{zahyou}[%
  ul=8mm,
  zikusensyu=¥thicklines¥arrowdrawline,
  arrowheadsize=2
  ]%
  (-3,3)(-2,2)%
¥end{zahyou}%
```

→



13.4.6　描画領域の周辺に余白

描画領域の周辺に文字を配置するためなどに余白をとっておきたいことがあります。そのためのオプションが

```
migiyohaku
hidariyohaku
```

ueyohaku
sitayohaku

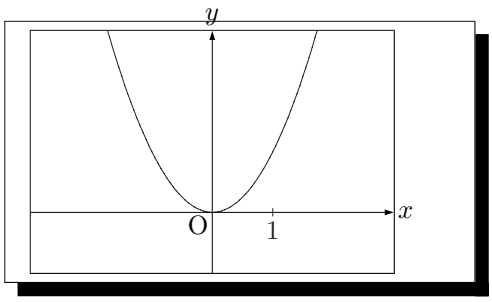
です。

まず基準の確認です。TEX が認識している zahyou 環境を ¥fbox で枠をつけてみます。

基準サイズ

```
¥fboxsep=0pt¥fbox{%
¥begin{zahyou}%
  [%
    ul=8mm,
    yokozikuhaiti={[e]},
    tatezikuhaiti={[n]}
  ]%
  (-3,3)(-1,3)%
  ¥xmemori{1}%
  ¥def¥Fx{X*X}
  ¥YGraph¥Fx
¥end{zahyou}}%
```

→



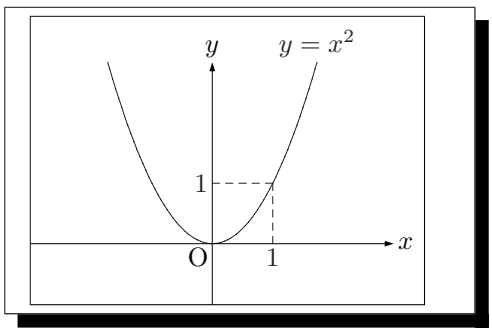
座標軸のラベルが枠外にはみだしています。

上と右に余白を付けて見ます。

上と右に余白

```
¥fboxsep=0pt¥fbox{%
¥begin{zahyou}%
  [%
    ul=8mm,
    yokozikuhaiti={[e]},
    tatezikuhaiti={[n]},
    ueyohaku=.75,
    migiyohaku=.5
  ]%
  (-3,3)(-1,3)%
  ¥def¥Fx{X*X}
  ¥Put{(1,1)}[syaei=xy]{}%
  ¥YGraph¥Fx
  ¥Put{(1.732,3)}[n]{$y=x^2$}%
¥end{zahyou}}%
```

→



枠内に納まりましたね。

ueyohaku などの右辺値は，無名数で，単位は ¥unitlength です。

これに対して

Migiyohaku=
Hidariyohaku=

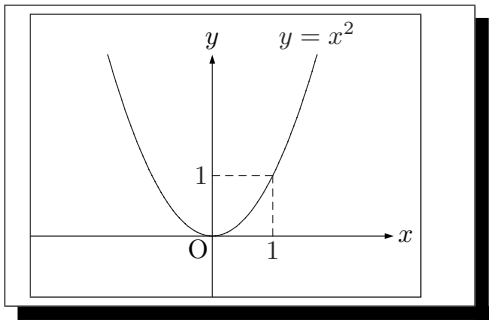
```
Ueyohaku=
Sitayohaku=
Yohaku=
```

は，右辺値に単位を伴う長さを与えます。

¥Ueyohaku

```
¥fboxsep=0pt¥fbox{%
¥begin{zahyou}%
  [%
    ul=8mm,
    yokozikuhaiti={[e]},
    tatezikuhaiti={[n]},
    Ueyohaku=15pt,
    Migiyohaku=10pt
  ]%
  (-3,3)(-1,3)%
  ¥def¥Fx{X*X}
  ¥Put{(1,1)}[syaei=xy]{}%
  ¥YGraph¥Fx
  ¥Put{(1.732,3)}[n]{$y=x^2$}%
¥end{zahyou}}%
```

→



（注）単位付の余白設定オプションは，内部では ¥unitlength 単位に換算されます。従って，[ul=..] と併用するときは，ul=.. が先行する必要があります。

13.4.7 縦横比の変更

zahyou 環境では，縦と横の単位長は同一になっています。これを別々に設定するためのオプション

```
yscale, xscale
```

があります。

ただし，この機能は emathPxy.sty で定義されていますから，

```
¥usepackage{emathPxy}
```

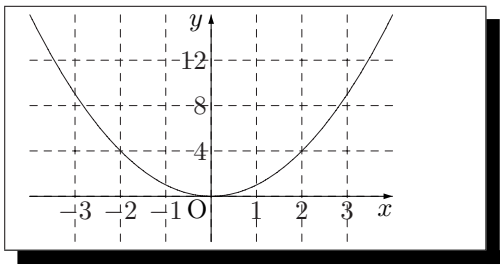
としておく必要があります。

次のリストは yscale=.25 として，y 軸方向を $1/4$ に縮め，$y = x^2$ $(-4 < x < 4)$ のグラフを描画しています。

<yscale=...>オプション

```
¥begin{zahyou}%
  [ul=6mm,yscale=.25]%
  (-4,4)(-4,16)
¥def¥Fx{X*X}
¥YGraph¥Fx
¥zahyouMemori[g]<dx=1,dy=4>
¥end{zahyou}
```

→



グラフ描画コマンド ¥YGraph については後述します。

次は横長の格子を描く例です。

```
ul=3mm,xscale=3,yscale=2
```

というオプションで

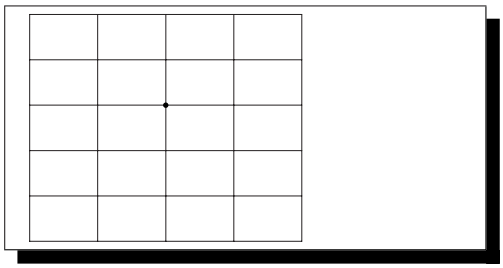
ul=3mm **で** ¥unitlength **は** 3mm
x **軸方向の単位長は** ul × xscale 9mm **で** ¥xunitlength **が** 9mm
y **軸方向の単位長は** ul × yscale 6mm **で** ¥yunitlength **が** 6mm

に設定されます。

横長の格子

```
¥begin{zahyou*}[%
    ul=3mm,xscale=3,%
    yscale=2](0,4)(0,5)%
  ¥Put¥O{¥kousi{4}{5}}%
  ¥Kuromaru{(2,3)}%
¥end{zahyou*}%
```

→



13.5 zahyou 環境の縦方向配置

まずは,

picture(zahyou) 環境で作成した図,
tabular(array) 環境で作成した表

を横に並べたときの縦位置に関する話から始めます。

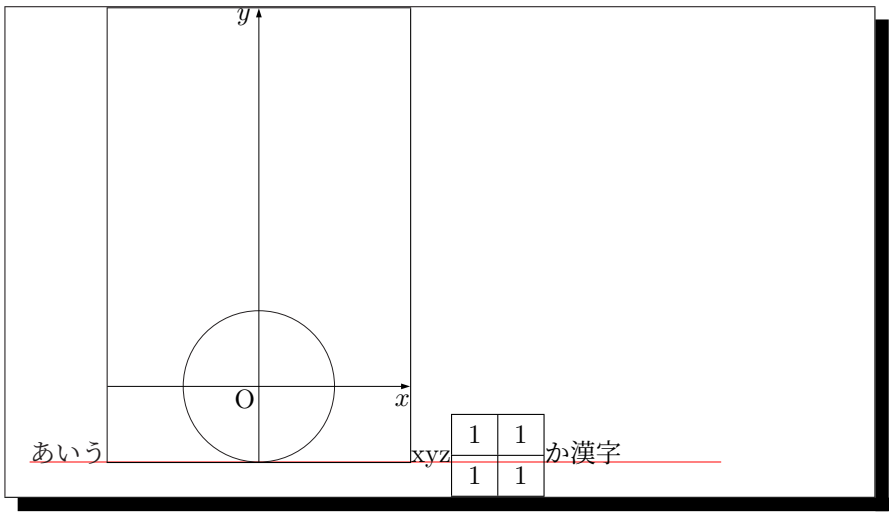
13.5.1 デフォルトの確認

まずはデフォルトの状態の確認です。

デフォルト

```
¥kizyunsen
あいう%
¥fbox{¥begin{zahyou}[ul=10mm](-2,2)(-1,5)
  ¥En¥O{1}
¥end{zahyou}}%
xyz%
¥begin{tabular}{|c|c|}¥hline
  1 & 1 ¥¥¥hline
  1 & 1 ¥¥¥hline
¥end{tabular}%
か漢字%
```

→



赤の横線が基準線（ベースライン）です。TEX は，この線を文字通り基準として文字，表，図などボックスを配置して行きます。まずは，文字にご注目ください。‘y’ は基準線から下にはみ出す形で配置されています。すなわち「深さ」をもっています。全角文字も良く見ると一部基準線から下にはみ出しています。

さて，図ですが，picture 環境の下辺を基準線に重ねる，というのが LATEX の仕様です。

表のほうは，デフォルトでは，縦方向中央揃えとなります。デフォルトでは，といったのは，オプションで変更可能ということで，次節でそれを見ていきます。

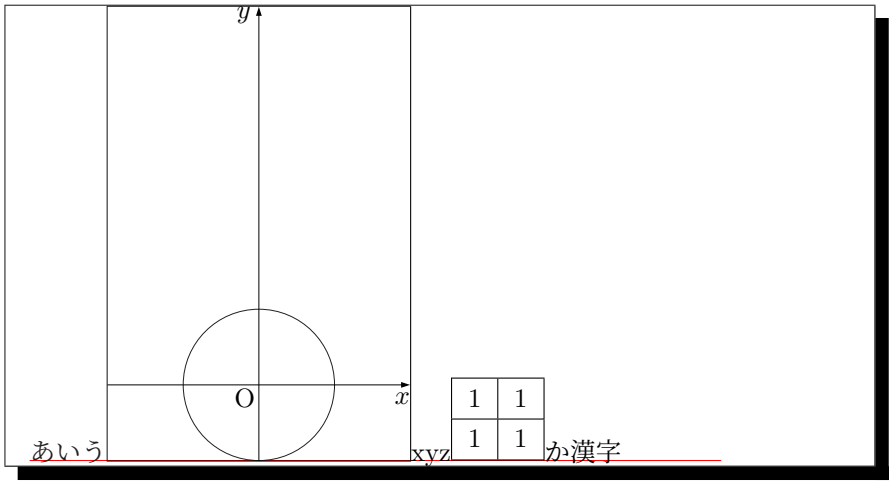
13.5.2 下揃え

表の下辺を基準線に重ねるのが，tabular, array の [b] オプションです。

下揃え

```
¥begin{tabular}[b]{|c|c|}¥hline
```

→



ソースリストは，デフォルトのものから変更した行だけを記します。

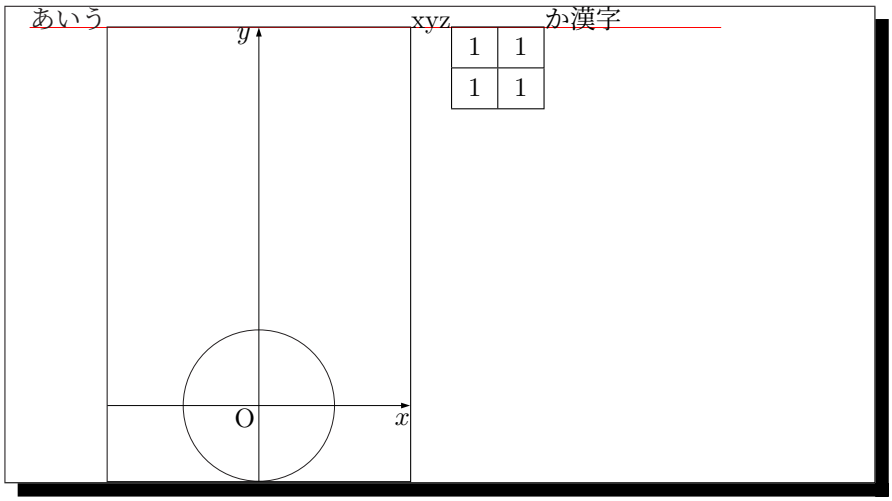
13.5.3 上揃え

逆に，表の上辺を基準線に重ねるのが，tabular, array の [t] オプションです。zahyou 環境にも [haiti=t] オプションがあります。

上揃え

```
¥fbox{¥begin{zahyou}[ul=10mm,haiti=t](-2,2)(-1,5)
¥begin{tabular}[t]{|c|c|}¥hline
```

→

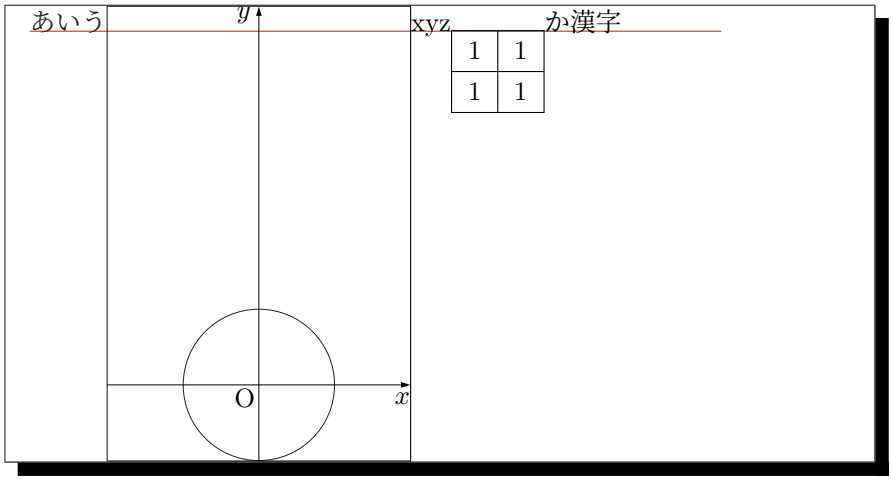


zahyou 環境の上辺をさらに上に引っ張って文字の高さと揃えたい，というご要望もありそうです。そのために haiti=t/c/b オプションには，そのあとに補正量を与えることができるようにしてあります。

上揃え–さらに調整

```
¥fbox{¥begin{zahyou}[ul=10mm,haiti=t+1zh](-2,2)(-1,5)
```

→



正の補正量で上方に，負の補正量で下方に移動します。

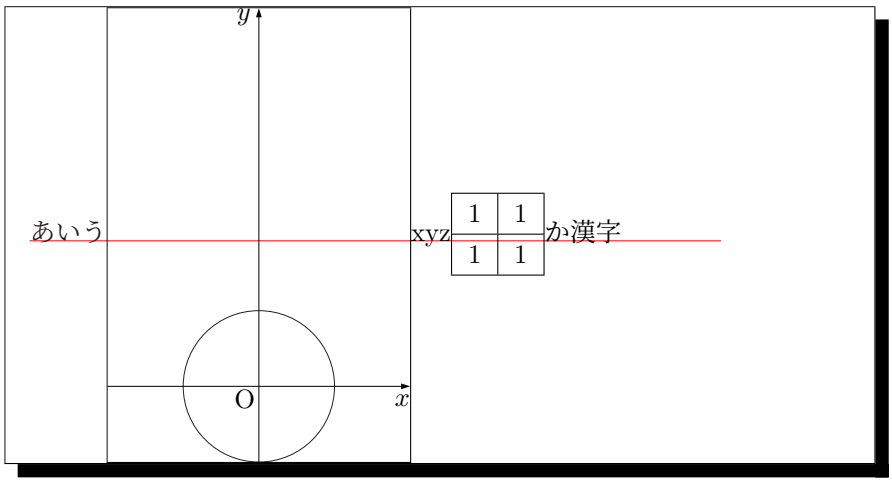
13.5.4 中央揃え

表を縦方向センタリングするのが，tabular, array の [c] オプションです。
zahyou 環境にも [haiti=c] オプションがあります。

中央揃え

```
¥fbox{¥begin{zahyou}[ul=10mm,haiti=c](-2,2)(-1,5)
¥begin{tabular}[c]{|c|c|}¥hline
```

→

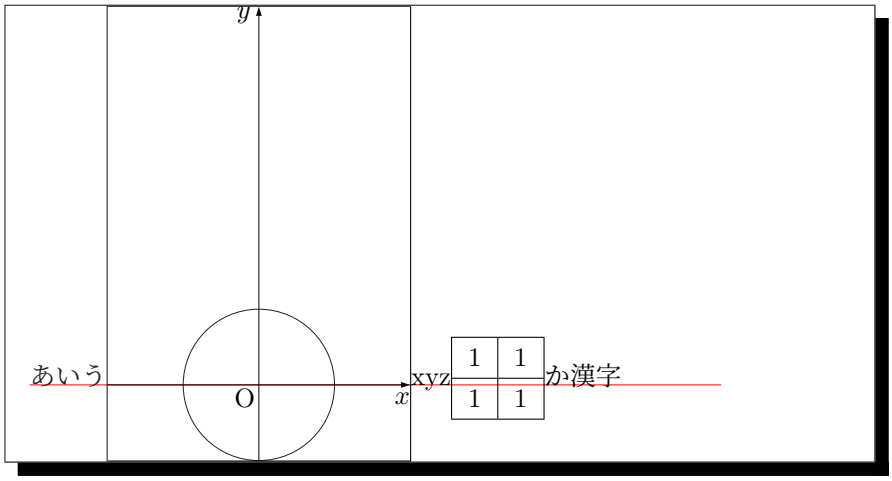


zahyou 環境には，[haiti=x] オプションもあります。これは x 軸を基準線に重ねます。

横軸揃え

```
¥fbox{¥begin{zahyou}[ul=10mm,haiti=x](-2,2)(-1,5)
¥begin{tabular}[c]{|c|c|}¥hline
```

→



emathPh.sty v 1.70 までは，[haiti=c] オプションで，x 軸を基準線に重ねていましたが，v 1.71 で上記のように修正いたしました。

13.6 zahyou 環境の書式

¥begin{zahyou}[#1](#2,#3)(#4,#5)
  #1 : key=val の形式で，key には次のものが使えます。
    yokozikukigou デフォルトは x
    tatezikukigou デフォルトは y
    gentenkigou デフォルトは O
    yokozikuhaiti デフォルトは (0,-3pt)[rt]
    tatezikuhaiti デフォルトは (-3pt,0)[rt]
    gentenhaiti デフォルトは [sw]
    ul ¥unitlength を変更します。デフォルトは 1pt
    yscale デフォルトは 1 --> emathPxy.sty
    xscale デフォルトは 1 --> emathPxy.sty
    arrowheadsize 矢印のサイズ（その時点のものに対する比率）
    zikusensyu 座標軸の線種
    デフォルトは ¥arrowdrawline
    ueyohaku 右辺値は無名数（単位は ¥unitlength）
    sitayohaku
    migiyohaku
    hidariyohaku
    Ueyohaku 右辺値は単位を伴う長さ
    Sitayohaku
    Migiyohaku
    Hidariyohaku
    haiti t/c/b/x
  (#2,#3) : x の範囲
  (#4,#5) : y の範囲

13.7 目盛り

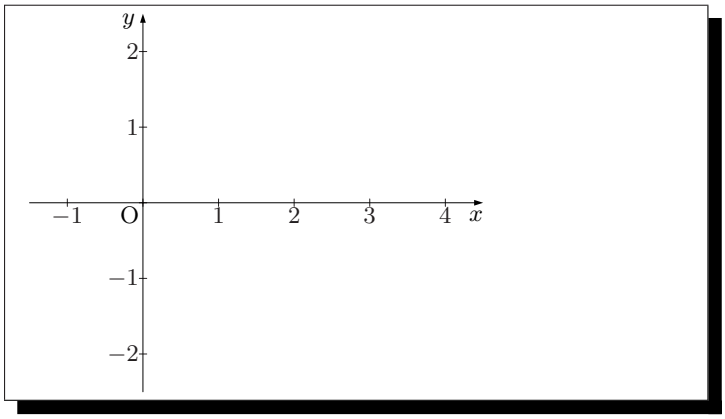
13.7.1 座標軸上に等間隔の目盛り

座標軸に等間隔に目盛りを打つコマンドが ¥zahyouMemori です.

座標目盛り

```
¥begin{zahyou}(-1.5,4.5)(-2.5,2.5)¥small
¥zahyouMemori%
¥end{zahyou}
```

→



¥zahyouMemori[#1][#2]<#3>

#1 : g : **グリッド**

+ : **格子点に +マーク**

o : **格子点に黒丸**

z : **座標軸上の格子点に +マーク**

#2 : n : **グリッドのみで，目盛り数値を打たない．**

#3 : **刻み**

key=val **の形式** ---> emathPxy.sty

key **は**

dx= x **の刻み値**

dy= y **の刻み値**

xo= x **の基準値**

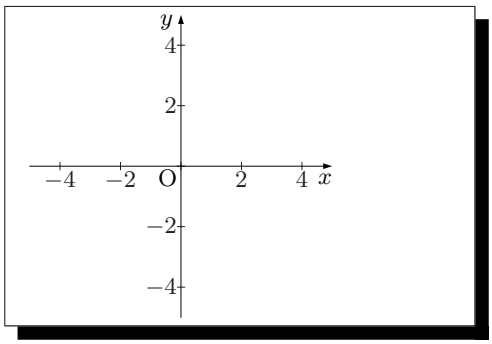
yo= y **の基準値**

目盛りの間隔はデフォルトでは 1 ですが，これを 2 に変更してみましょう。

間隔 2 の目盛線

```
¥small
¥begin{zahyou}[ul=4mm](-5,5)(-5,5)
¥zahyouMemori<2>
¥end{zahyou}
```

→

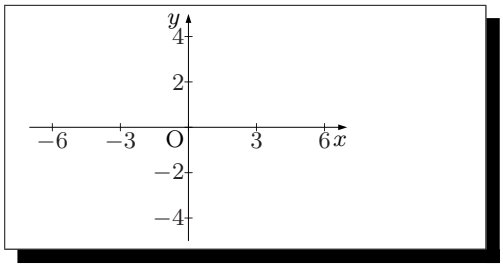


x 軸と y 軸とで間隔を変えることもできます。

<dx=..,dy=..>オプション

```
¥small
¥begin{zahyou}%
  [ul=3mm](-7,7)(-5,5)
¥zahyouMemori<dx=3,dy=2>
¥end{zahyou}
```

→

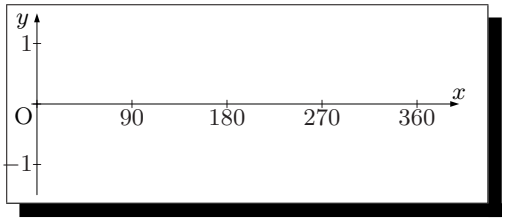


¥xscale, ¥yscale と併用して

¥xscale と併用オプション

```
¥small
¥begin{zahyou}%
  [ul=8mm,yokozikuhaiti={[n]},%
    xscale=0.017453]%
  (-5,400)(-1.5,1.5)
¥zahyouMemori<dx=90>
¥end{zahyou}
```

→

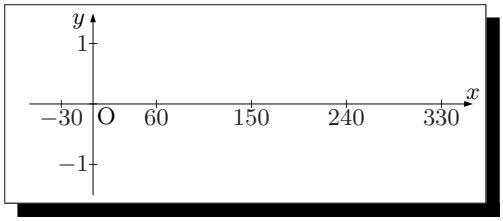


基準点はデフォルトでは原点ですが，これを変更するには，<xo=...,yo=...>オプションを用います。

基準点の変更

```
¥small
¥begin{zahyou}%
  [ul=8mm,%
    yokozikuhaiti={[n]},%
    gentenhaiti={[se]},%
    xscale=0.017453]%
  (-60,360)(-1.5,1.5)
¥zahyouMemori<dx=90,xo=-30>
¥end{zahyou}
```

→



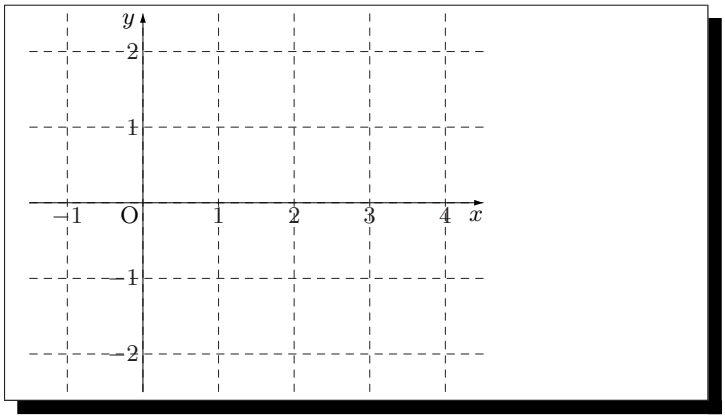
13.7.2 グリッド線

[g] オプションをつけると方眼を描きます.

方眼 & 目盛り

```
¥begin{zahyou}(-1.5,4.5)(-2.5,2.5)¥small
¥zahyouMemori[g]%
¥end{zahyou}
```

$\longrightarrow$

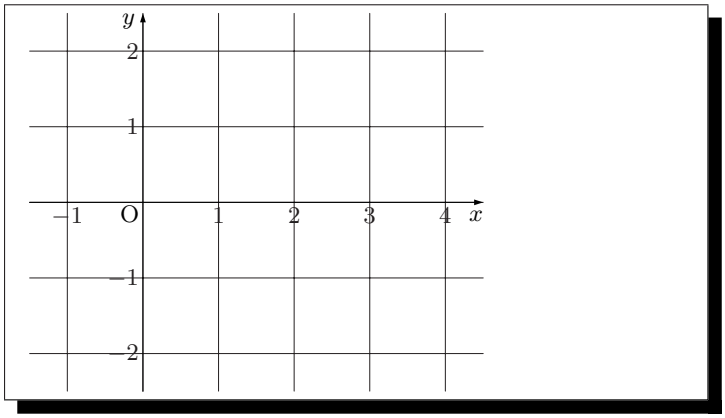


グリッド線の線種を変更するには，<sensyu=...> オプションを用います。

線種の変更

```
¥begin{zahyou}(-1.5,4.5)(-2.5,2.5)¥small
¥zahyouMemori[g]<sensyu=¥drawline>%
¥end{zahyou}
```

$\longrightarrow$

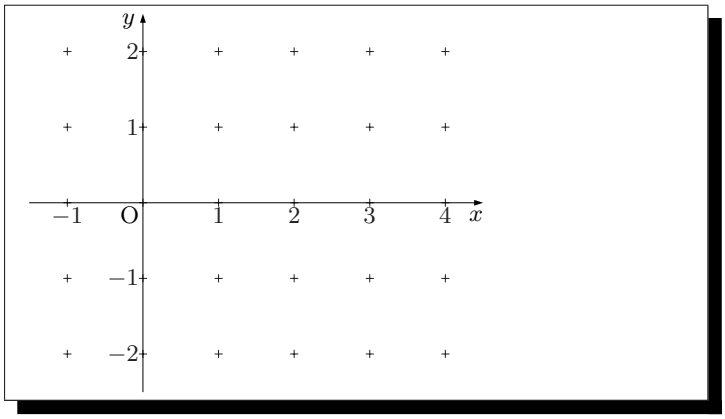


[+] オプションでは，格子点に + マークをつけます。

格子点に +

```
¥begin{zahyou}(-1.5,4.5)(-2.5,2.5)¥small
¥zahyouMemori[+]%
¥end{zahyou}
```

⟶

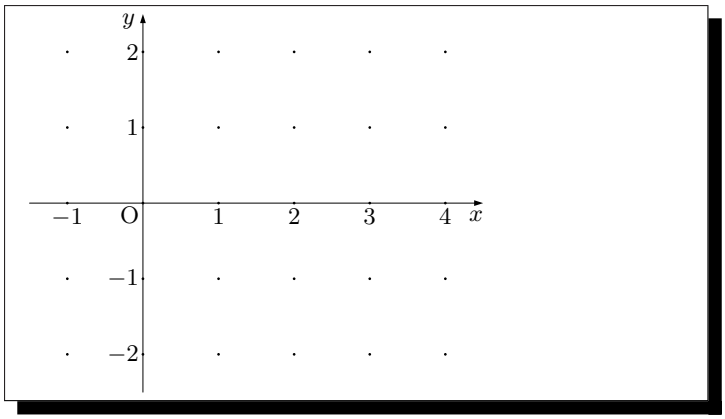


[o] オプションでは，格子点に黒丸をつけます。

格子点に ●

```
¥begin{zahyou}(-1.5,4.5)(-2.5,2.5)¥small
¥zahyouMemori[o]%
¥end{zahyou}
```

⟶

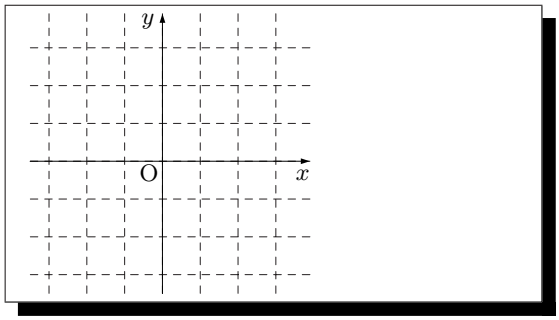


なお，目盛りの数値は要らないというときは第 2 の [n] オプションをつけます.

方眼

```
{¥unitlength5mm¥small
¥begin{zahyou}%
(-3.5,3.9)(-3.5,3.9)%
¥zahyouMemori[g][n]%
¥end{zahyou}}%
```

⟶



13.7.3 軸上に目盛り

また，座標軸上の指定した位置に目盛りを打つコマンドが

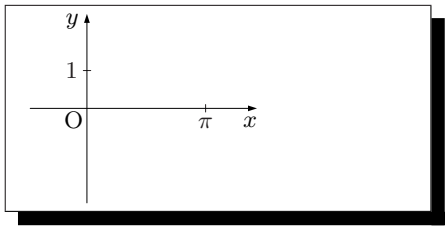
¥xmemori, ¥ymemori

です.

目盛り

```
¥unitlength5mm
¥begin{zahyou}%
(-1.5,4.5)(-2.5,2.5)%
¥small
¥ymemori{1}%
¥xmemori<¥pi>{3.1416}%
¥end{zahyou}
```

→

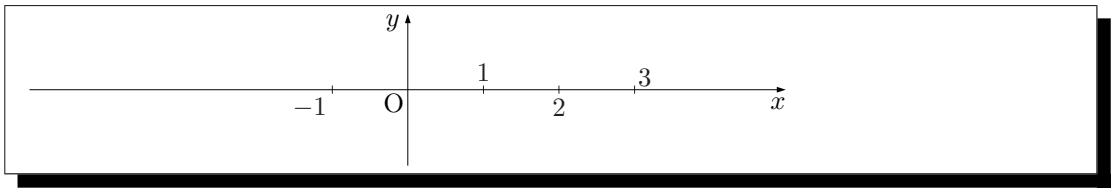


目盛りの位置をずらしたいときは，[..] オプションに ¥emathPut の配置オプションを記述します。

¥xmemori の [...] オプション

```
¥begin{zahyou}[ul=10mm](-5,5)(-1,1)
  ¥xmemori[b]{1}
  ¥xmemori[t]{2}
  ¥xmemori[{[ne]}]{3}
  ¥xmemori[{(-2pt,-3pt)[rt]}]{-1}
¥end{zahyou}
```

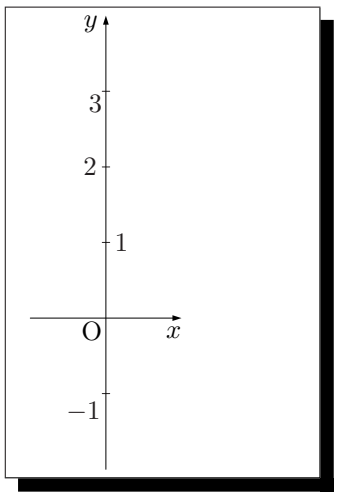
↓



¥ymemori の [...] オプション

```
¥begin{zahyou}[ul=10mm](-1,1)(-2,4)
  ¥ymemori[l]{1}
  ¥ymemori[r]{2}
  ¥ymemori[{[sw]}]{3}
  ¥ymemori[{(-2pt,-3pt)[rt]}]{-1}
¥end{zahyou}
```

→



¥xmemori, ¥ymemori の書式は

座標軸上任意位置に目盛

¥xmemori[#1]<#2>#3

¥ymemori[#1]<#2>#3

#1 : t = **座標軸の下に目盛**

b = **上**

l = **右**

r = **左**

¥emathPut **の配置指定オプション**

#2 : **目盛の文字**

(省略すれば，#3 **に指定したものが代用されます.)**

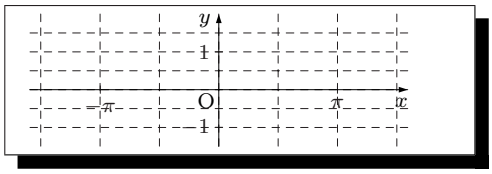
#3 : **目盛の位置**

¥xscale と併用した例です。

¥xscale と併用

```
¥footnotesize
¥begin{zahyou}[ul=5mm](-5,5)(-1.5,2)
¥zahyouMemori[g][n]<dx=1.57,dy=.5>
¥ymemori{1}
¥ymemori{-1}
¥xmemori<¥pi>{3.14}
¥xmemori<-¥pi>{-3.14}
¥end{zahyou}
```

→



13.8 座標軸への垂線

平面上に点をプロットしたとき，その点から座標軸に下ろした垂線を破線で描画したいときがあります。この目的のために ¥Put にオプションを追加しました。

¥emathPut#1[#2].....

#1 : **文字列を置く位置（座標）**

#2 : **方位**

syaei=x|y|xy

houi=n|nw|w|sw|s|se|e|ne

xlabel=..

ylabel=..

xpos=..

ypos=..

syaeisensyu=..

従来，[#2] オプションは，文字列の点に対する方位を指定するものでした。これはそのまま有効ですが，ここに

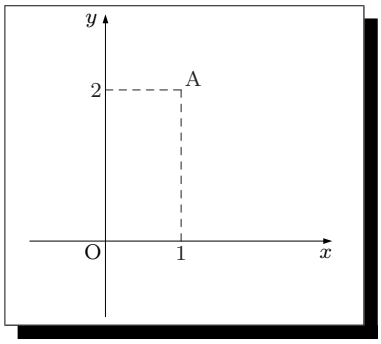
syaei=xy

とすれば，点から両座標軸への垂線を描画することができます。

[syaei=xy]

```
¥unitlength10mm¥footnotesize
¥begin{zahyou}(-1,3)(-1,3)
  ¥tenretu{A(1,2)ne}
  ¥Put¥A[syaei=xy]{}
¥end{zahyou}
```

→



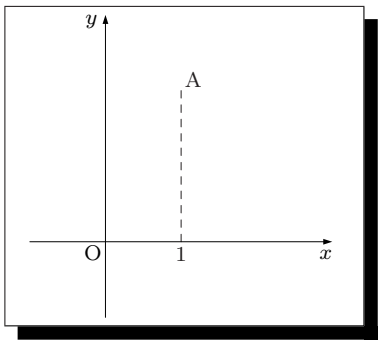
syaei=x

とすれば，x 軸への垂線が描画されます。

[syaei=x]

```
¥unitlength10mm¥footnotesize
¥begin{zahyou}(-1,3)(-1,3)
  ¥tenretu{A(1,2)ne}
  ¥Put¥A[syaei=x]{}
¥end{zahyou}
```

→



従来の方位オプションもここに記述したいときは

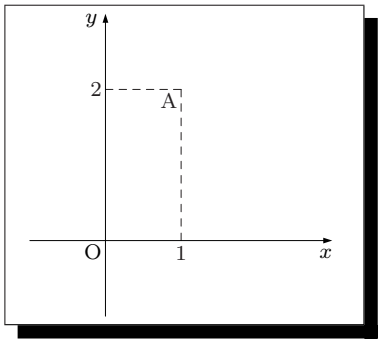
houi=..

と記述します。

[houi=..]

```
¥unitlength10mm¥footnotesize
¥begin{zahyou}(-1,3)(-1,3)
  ¥def¥A{(1,2)}
  ¥Put¥A[syaei=xy,houi=sw]{A}
¥end{zahyou}
```

→



座標軸上のラベルですが，デフォルトでは指定点の x, y 座標が表示されます。これが整数の場合は問題ないのですが，分数，無理数，文字を表示したいときは

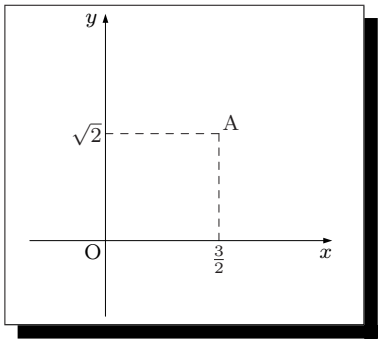
```
xlabel=..
ylabel=..
```

オプションを用います。

[xlabel=..]

```
¥unitlength10mm¥footnotesize
¥begin{zahyou}(-1,3)(-1,3)
  ¥tenretu{A(1.5,1.414)ne}
  ¥Put¥A[syaei=xy,xlabel=¥frac32,
    ylabel=¥sqrt2]{}
¥end{zahyou}
```

→



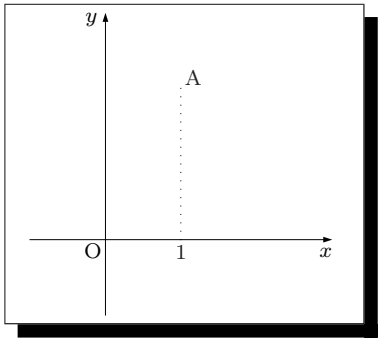
このオプションの右辺値は数式モード内であるという前提です。また `[...,xlabel=]` と右辺値を空にすればラベルは打たれません。

また，垂線の線種を変更するには，`syaeisensyu=..` オプションを用います。

[syaeisensyu=]

```
¥unitlength10mm¥footnotesize
¥begin{zahyou}(-1,3)(-1,3)
  ¥tenretu{A(1,2)ne}
  ¥Put¥A[syaei=x,%
    syaeisensyu=¥protect¥emdottedline]{}
¥end{zahyou}
```

→



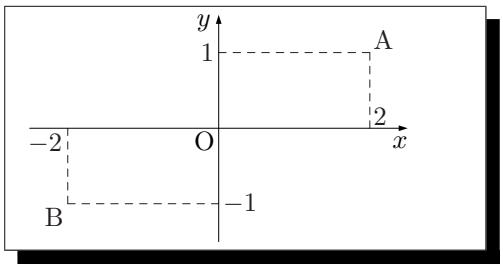
垂線の足につける目盛り位置を修正するときは，`[xpos=..]`, `[ypos=..]` オプションで，右辺値に `¥Put` の配置オプションを記述します。

x 軸に下ろした垂線の足位置につける文字位置は `[xpos=..]` オプションで修正します。

xpos=.. オプション

```
¥begin{zahyou}[ul=10mm]%
    (-2.5,2.5)(-1.5,1.5)
  ¥tenretu{A(2,1)ne;B(-2,-1)sw}
  ¥Put¥A[syaei=xy,xpos={[ne]}]{}
  ¥Put¥B[syaei=xy,%
    xpos={(-2pt,-2pt)[rt]}]{}
¥end{zahyou}
```

→

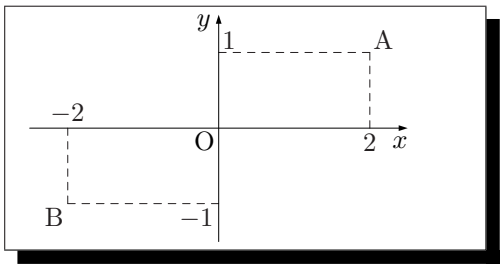


y 軸に下ろした垂線の足位置につける文字位置の調整は [ypos=..] オプションです。

ypos=.. オプション

```
¥begin{zahyou}[ul=10mm]%
    (-2.5,2.5)(-1.5,1.5)
  ¥tenretu{A(2,1)ne;B(-2,-1)sw}
  ¥Put¥A[syaei=xy,ypos={[ne]}]{}
  ¥Put¥B[syaei=xy,%
    ypos={(-2pt,-2pt)[rt]}]{}
¥end{zahyou}
```

→



14　点・直線・円

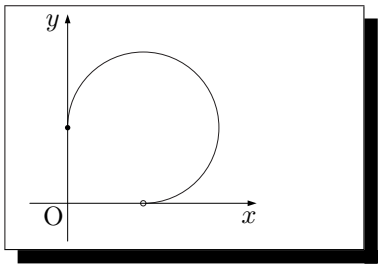
14.1　点の位置に黒丸

軌跡を求める問題などで，端点が含まれるとか含まれないとかを黒丸・中抜きの白丸などで表示することがあります．そのためのコマンドが ¥Kuromaru, ¥Siromaru です．

黒丸・白丸

```
{¥unitlength10mm
¥begin{zahyou}(-.5,2.5)(-.5,2.5)%
¥def¥A{(0,1)}%
¥def¥B{(1,0)}%
¥def¥C{(1,1)}%
¥Put¥C{¥enko{1}{-90}{180}}%
¥Kuromaru¥A
¥Siromaru¥B
¥end{zahyou}}
```

→



その書式は

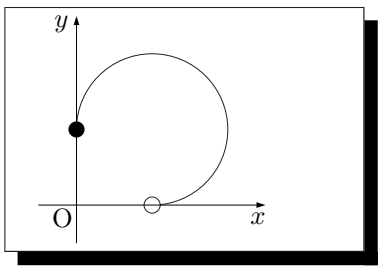
黒丸，白丸
¥Kuromaru[#1]#2
¥Siromaru[#1]#2
　#1 : **丸の半径（単位をつける）デフォルトは** 1pt
　#2 : **位置**

黒丸の大きさを変更したいときは，¥Kuromaru の [#1] オプションを利用すればよいのですが，文書全体で，あるいは広範囲に ¥Kuromaru の大きさを変更したいときは ¥KuromaruHankei コマンドの引数に黒丸の半径（単位つき） を与える方法もあります。

黒丸の大きさ

```
{¥unitlength10mm
¥KuromaruHankei{3pt}
¥begin{zahyou}(-.5,2.5)(-.5,2.5)%
¥def¥A{(0,1)}%
¥def¥B{(1,0)}%
¥def¥C{(1,1)}%
¥Put¥C{¥enko{1}{-90}{180}}%
¥Kuromaru¥A
¥Siromaru¥B
¥end{zahyou}}
```

→

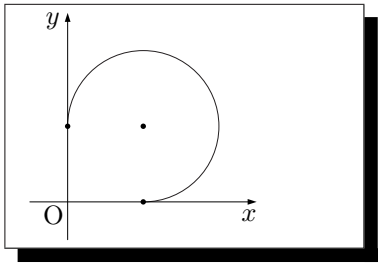


複数の点に黒丸（白丸）をつけるコマンドが ¥kuromaru(¥siromaru) です。複数の点を ';' で区切ります。

複数の点に黒丸

```
{¥unitlength10mm
¥begin{zahyou}(-.5,2.5)(-.5,2.5)%
¥def¥A{(0,1)}%
¥def¥B{(1,0)}%
¥def¥C{(1,1)}%
¥Put¥C{¥enko{1}{-90}{180}}%
¥kuromaru{¥A;¥B;¥C}%
¥end{zahyou}}
```

→



丸のサイズを指定する [..] オプションも ¥Kuromaru などと同様に有効です。

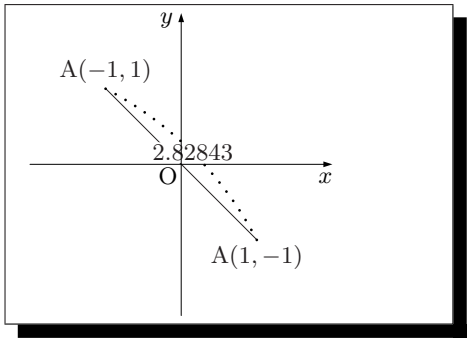
14.2 2 点間の距離

2 点間の距離を求めるには，コマンド ¥Kyori を用います.

¥Kyori

```
{¥unitlength10mm¥small
¥begin{zahyou}(-2,2)(-2,2)%
  ¥def¥A{(-1,1)}%
  ¥def¥B{(1,-1)}%
  ¥Kyori¥A¥B¥kyori
  ¥Put¥A(0,2pt)[b]{A$(-1,1)$}%
  ¥Put¥B(0,-2pt)[t]{A$(1,-1)$}%
  ¥Drawline{¥A¥B}%
  ¥HenKo[20]¥B¥A¥kyori
¥end{zahyou}}%
```

→



¥Kyori の書式です.

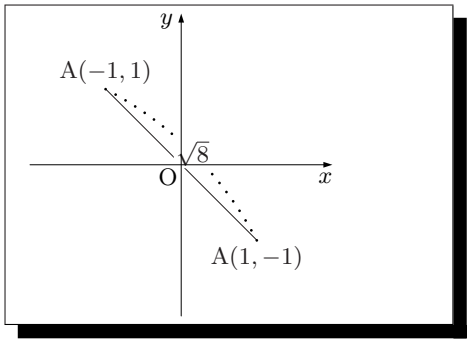
```
% 2 点間の距離
% ¥Kyori#1#2#3
%   2 点 #1, #2 の距離を #3 に代入
```

距離の平方を求めるコマンドが ¥Kyorii です.

¥Kyorii

```
{¥unitlength10mm¥small
¥begin{zahyou}(-2,2)(-2,2)%
  ¥def¥A{(-1,1)}%
  ¥def¥B{(1,-1)}%
  ¥Kyorii¥A¥B¥kyorii
  ¥Put¥A(0,2pt)[b]{A$(-1,1)$}%
  ¥Put¥B(0,-2pt)[t]{A$(1,-1)$}%
  ¥Drawline{¥A¥B}%
  ¥HenKo[20]¥B¥A{%
  $¥sqrt{¥kyorii}$}%
¥end{zahyou}}%
```

→



¥Kyorii の書式です.

```
% 2 点間の距離の平方
% ¥Kyorii#1#2#3
%    2 点 #1, #2 の距離の平方を #3 に代入
```

14.3 直線

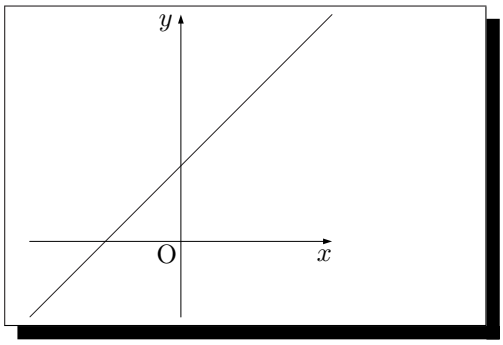
14.3.1 2 点を通る直線

2 点を通る直線を描画するコマンドが ¥Tyokusen です.

2 点を通る直線

```
¥begin{zahyou}(-2,2)(-1,3)%
¥def¥A{(0,1)}%
¥def¥B{(1,2)}%
¥Tyokusen¥A¥B¥xmin¥xmax
¥end{zahyou}%
```

→



¥Tyokusen の書式です.

2 点を与えて直線を描画する．

¥Tyokusen[#1]#2#3[#4]#5[#6]#7

#1 : ¥qbezier **の** [...]
#1 を与えないときは直線を
¥qbezier ではなく，¥drawline で描画する．

#2,3 : **直線上の２点**

#5 : **左端の** x **座標** （#4 **に** [y] **とすれば，**y **座標）**

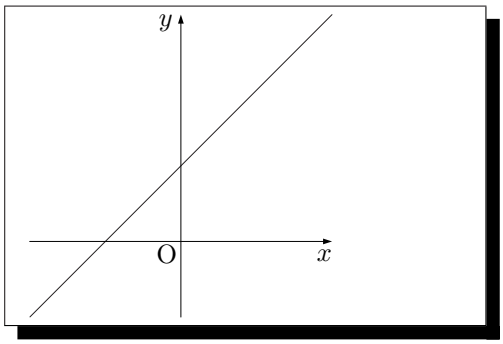
#7 : **右端の** x **座標** （#6 **に** [y] **とすれば，**y **座標）**

直線を描画領域いっぱいにかくときは，#5，#7 を空にすることができます。

2 点を通る直線

```
¥begin{zahyou}(-2,2)(-1,3)%
¥def¥A{(0,1)}%
¥def¥B{(1,2)}%
¥Tyokusen¥A¥B{}{}
¥end{zahyou}%
```

→



この場合，描画した直線の両端点が

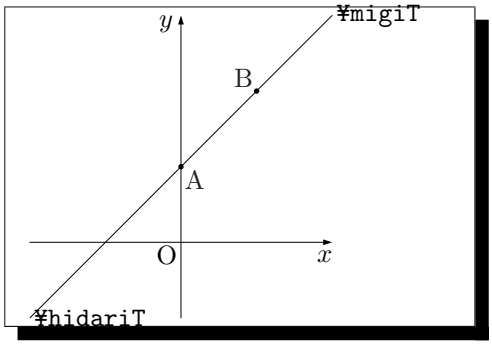
¥hidariT, ¥migiT

に定義されています。ただし，¥hidariT と ¥migiT は，左・右とはなっていません。

¥hidariT(1)

```
¥begin{zahyou}[ul=10mm](-2,2)(-1,3)%
¥tenretu{A(0,1)se;B(1,2)nw}
¥kuromaru{¥A;¥B}
¥Tyokusen¥A¥B{}{}
¥Put¥hidariT[e]{¥cmd{hidariT}}
¥Put¥migiT[e]{¥cmd{migiT}}
¥end{zahyou}%
```

→

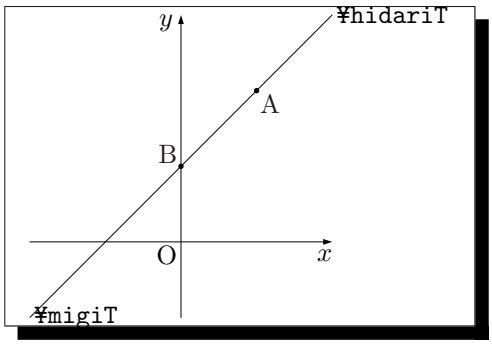


は自然ですが,

¥hidariT(2)

```
¥begin{zahyou}[ul=10mm](-2,2)(-1,3)%
¥tenretu{A(1,2)se;B(0,1)nw}
¥kuromaru{¥A;¥B}
¥Tyokusen¥A¥B{}{}
¥Put¥hidariT[e]{¥cmd{hidariT}}
¥Put¥migiT[e]{¥cmd{migiT}}
¥end{zahyou}%
```

→



は，???ですね。

#1 に近い方が ¥hidariT, #2 に近い方が ¥migiT です。

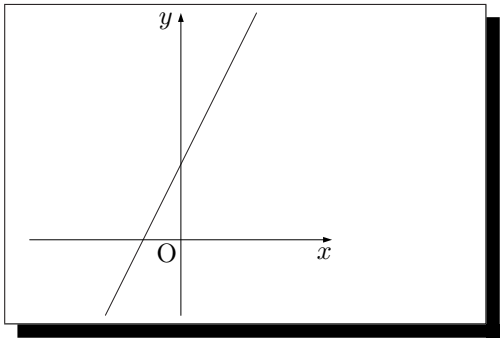
14.3.2 1 点と方向ベクトルによる直線

1 点と方向ベクトルを与えて直線を描画するコマンドが ¥mTyokusen です.

1 点と方向ベクトルを指定

```
¥begin{zahyou}(-2,2)(-1,3)%
¥def¥A{(0,1)}%
¥def¥u{(1,2)}%
¥mTyokusen¥A¥u[y]¥ymin[y]¥ymax
¥end{zahyou}%
```

⟶



その書式は

１点と方向ベクトルを与えて直線を描画する．

¥mTyokusen[#1]#2#3[#4]#5[#6]#7

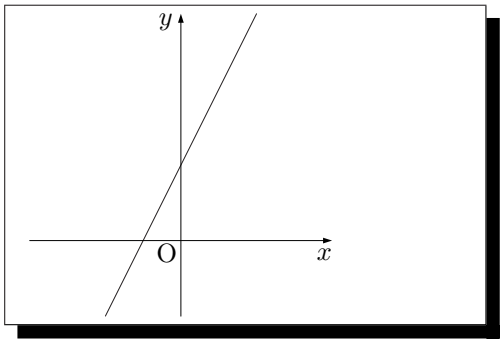
- #1 : ¥qbezier **の** [...]
  - #1 **を与えないときは直線を**
  - ¥qbezier **ではなく，**¥drawline **で描画する．**
- #2 : **直線上の１点**
- #3 : **方向ベクトル**
- #5 : **左端の** x **座標** （#4 **に** [y] **とすれば，**y **座標）**
- #7 : **右端の** x **座標** （#6 **に** [y] **とすれば，**y **座標）**

直線を描画領域いっぱいにかくときは，#5，#7 を空にすることができます。

端点の指定省略

```
¥begin{zahyou}(-2,2)(-1,3)%
¥def¥A{(0,1)}%
¥def¥u{(1,2)}%
¥mTyokusen¥A¥u{}{}
¥end{zahyou}%
```

→



この場合も，両端点が ¥hidariT，¥migiT に定義されています。

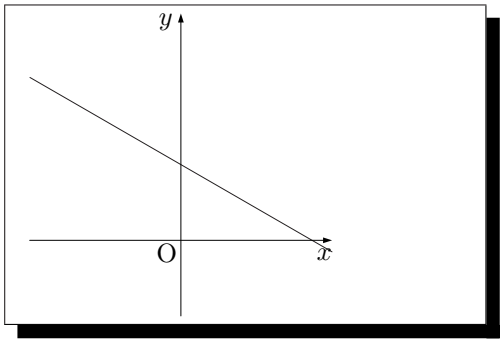
14.3.3 1 点と方向角による直線

1 点と方向角を与えて直線を描画するコマンドが ¥kTyokusen です.

1 点と方向角を指定

```
¥begin{zahyou}(-2,2)(-1,3)%
¥def¥A{(0,1)}%
¥kTyokusen¥A{-30}{}{}
¥end{zahyou}%
```

→



その書式は

１点と方向角を与えて直線を描画する．

¥kTyokusen[#1]#2#3[#4]#5[#6]#7

- #1 : ¥qbezier **の** [...]
  #1 **を与えないときは直線を**
  ¥qbezier **ではなく，**¥drawline **で描画する．**
- #2 : **直線上の１点**
- #3 : **方向角**
- #5 : **左端の** x **座標** （#4 **に** [y] **とすれば，**y **座標）**
- #7 : **右端の** x **座標** （#6 **に** [y] **とすれば，**y **座標）**

直線を描画領域いっぱいにかくときは，#5，#7 を空にすることができます。
この場合も，両端点が ¥hidariT，¥migiT に定義されています。

14.3.4 線種の変更

直線を実線ではなく，点線とか破線で描画するには ¥sensyu を再定義します。デフォルトは

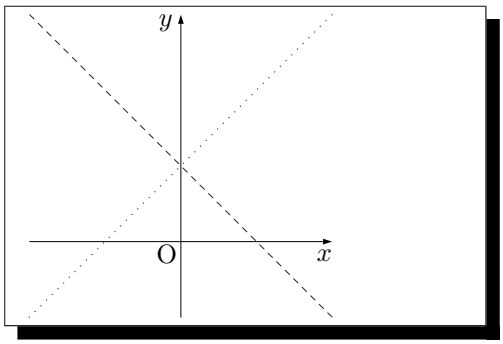
```
¥def¥sensyu{¥drawline}
```

となっています。

線種の変更

```
¥begin{zahyou}(-2,2)(-1,3)%
¥def¥A{(0,1)}%
¥def¥B{(1,2)}%
¥def¥C{(1,0)}%
{¥def¥sensyu{%
  ¥protect¥emdottedline}
  ¥Tyokusen¥A¥B{}{}}
{¥def¥sensyu{%
  ¥dashline[40]{.1}}
  ¥Tyokusen¥A¥C{}{}}
¥end{zahyou}%
```

→



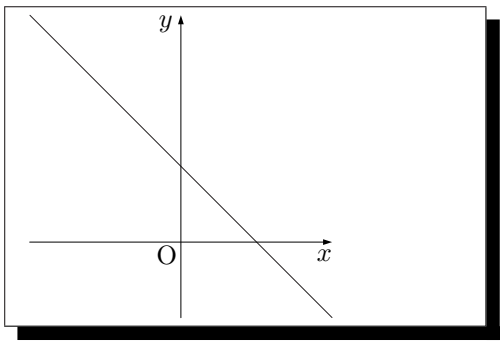
14.3.5 直線 $ax + by + c = 0$

方程式 $ax + by + c = 0$ で与えられた直線を描画する ¥tyokusen です.

方程式の係数指定

```
¥begin{zahyou}(-2,2)(-1,3)%
¥tyokusen{1}{1}{-1}{}{}
¥end{zahyou}%
```

→



その書式は

直線 ax+by+c=0 **を描画する.**

¥tyokusen[#1]#2#3#4[#5]#6[#7]#8

- #1 : ¥qbezier **の** [...]
  #1 **を与えないときは直線を**
  ¥qbezier **ではなく,** ¥drawline **で描画する.**
- #2 : a
- #3 : b
- #4 : c
- #6 : **左端の** x **座標** (#5 **に** [y] **とすれば,** y **座標**)
- #8 : **右端の** x **座標** (#7 **に** [y] **とすれば,** y **座標**)

このコマンドでも，座標平面全体に描画するときは#5，#7 は空でかまいません。

14.3.6 直線 $y = ax + b$

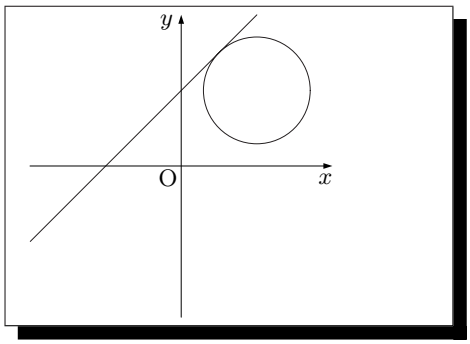
これは，次節の一次関数のグラフ描画コマンドをご利用ください.

14.3.7 点 $\mathrm{P}(x_1, y_1)$ と直線 $ax + by + c = 0$ の距離

点 P と直線 l の距離を求めることにより，点 P を中心として l に接する円を描画することが出来ます.

¥tentotyokusen

```
{¥unitlength10mm¥small
¥begin{zahyou}(-2,2)(-2,2)%
¥def¥C{(1,1)}% 点 C を中心とし
¥tyokusen{1}{-1}{1}{}{}%
%                直線 x-y+1=0 に
¥tentotyokusen¥C{1}{-1}{1}%
    ¥hankei%
¥En¥C¥hankei% 接する円
¥end{zahyou}}%
```



その書式は

点と直線の距離(1)
点 P$(x1,y1)$ **と直線** $ax+by+c=0$ **との距離を求める.**
¥tentotyokusen#1#2#3#4#5
#1 : **点** P
#2 : a
#3 : b
#4 : c
#5 : **点と直線の距離**

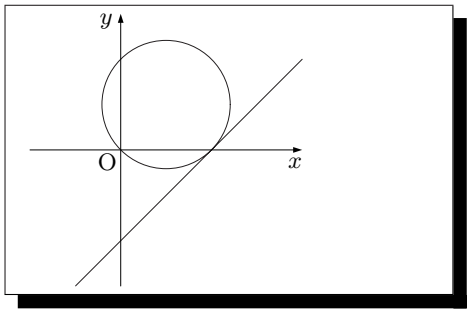
14.3.8 点 $\mathrm{P}(x_1, y_1)$ と 2 点 A,B を通る直線の距離

直線がその上の 2 点で指定された場合です.

¥tentoTyokusen

```
{¥unitlength6mm¥small
¥begin{zahyou}(-2,4)(-3,3)%
¥def¥C{(1,1)}%  点 C を中心とし，
¥def¥A{(0,-2)}%
¥def¥B{(1,-1)}% 直線 AB に
¥tentoTyokusen¥C¥A¥B¥hankei
¥En¥C¥hankei%   接する円
¥Tyokusen¥A¥B{}{}%
¥end{zahyou}}%
```

→



その書式は

点と直線の距離 (2)
点 P¥retu(x1,y1) **と直線** AB **との距離を求める．**
¥tentoTyokusen#1#2#3#4
#1 : **点** P
#2 : **点** A
#3 : **点** B
#4 : **点と直線の距離**

14.4 円の接線

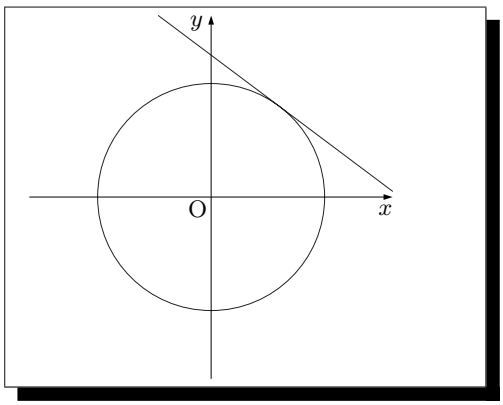
14.4.1 円周上の点における接線

円周上の点における接線を描画するために，その方向ベクトルを求めるコマンドが ¥ennoSessen です．

円周上の点における接線

```
{¥unitlength3mm¥small
¥begin{zahyou}(-8,8)(-8,8)%
¥def¥O{(0,0)}%
¥def¥T{(3,4)}%
¥ennoSessen¥O¥T¥m
¥mTyokusen¥T¥m{}{}
¥En¥O{5}%
¥end{zahyou}}
```

→



その書式は

円周上の点における接線の単位方向ベクトルを求める．

¥ennoSessen#1#2#3{%

#1 ： **円の中心**

#2 ： **円周上の点**

#3 ： **接線の方向ベクトル（単位ベクトル）**

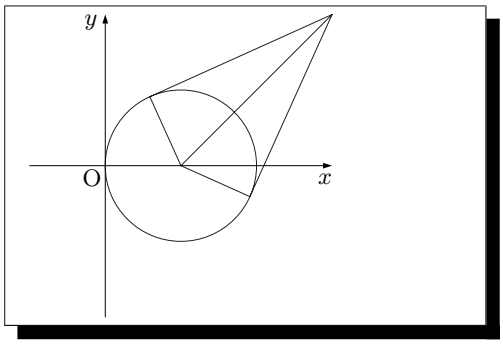
14.4.2　円外の点からの接線

円の外部の点から円に引いた接線の 2 個の接点を求めるコマンドが ¥enniSessen です．

円外の点からの接線

```
{¥unitlength10mm¥small
¥begin{zahyou}(-1,3)(-2,2)%
¥def¥A{(3,2)}%  円外の点
¥def¥C{(1,0)}%  円の中心
¥def¥hankei{1}% 円の半径
¥enniSessen¥C¥hankei¥A¥T¥TT
¥En¥C¥hankei
¥Drawline{¥A¥T¥C¥TT¥A¥C}%
¥end{zahyou}}
```

→



書式は

円の外部の点から円に引いた接線の接点を求める．

¥enniSessen#1#2#3#4#5{%

#1 ： **円の中心**

#2 ： **円の半径**

#3 ： **円の外部の点**

#4 ： **接点** (1)

#5 ： **接点** (2)

2 **つの接点のうち**

x **座標の小さい方**

等しいときは y **座標の小さい方**

が #4 **に入ります．**

14.4.3　二つの円の共通接線

2 つの円の共通外（内）接線を描画するためのコマンドが ¥KTGAISessen, ¥KTNAISessen です。書式は

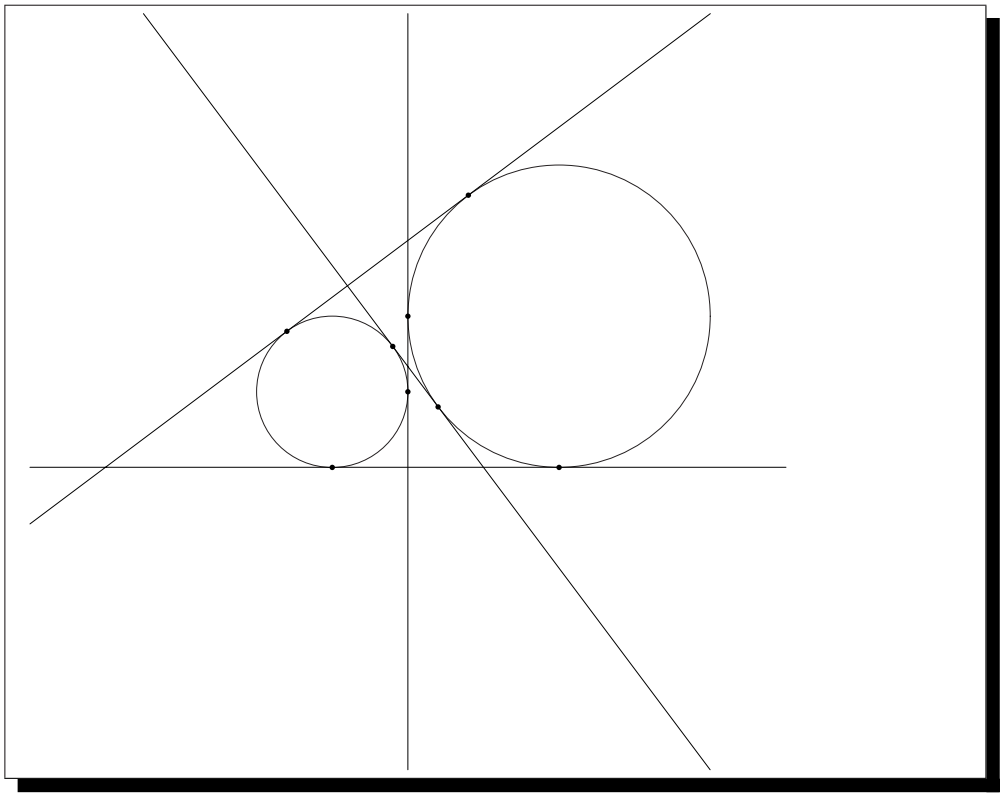
¥KTGAISessen#1#2#3#4#5#6#7#8
¥KTNAISessen#1#2#3#4#5#6#7#8
#1 : **円 1 の中心**
#2 : **円 1 の半径**
#3 : **円 2 の中心**
#4 : **円 2 の半径**
#5 : **共通接線 1 と円 1 の接点**
#6 : **共通接線 1 と円 2 の接点**
#7 : **共通接線 2 と円 1 の接点**
#8 : **共通接線 2 と円 2 の接点**

で，共通接線と円との接点を求め，¥Tyokusen などで共通接線を引きます。その一例です。

共通接線

```
{¥unitlength10mm¥small¥drawaxisfalse
¥begin{zahyou}(-5,5)(-5,5)%
¥def¥Ci{(-1,0)}¥def¥ri{1}%
¥def¥Cii{(2,1)}¥def¥rii{2}%
¥En¥Ci¥ri¥En¥Cii¥rii
¥KTGAISessen¥Ci¥ri¥Cii¥rii¥Ti¥Tii¥Si¥Sii
¥Kuromaru¥Ti¥Kuromaru¥Tii¥Kuromaru¥Si¥Kuromaru¥Sii
¥Tyokusen[l]¥Ti¥Tii{}{}
¥Tyokusen[l]¥Si¥Sii{}{}
¥KTNAISessen¥Ci¥ri¥Cii¥rii¥Ti¥Tii¥Si¥Sii
¥Kuromaru¥Ti¥Kuromaru¥Tii¥Kuromaru¥Si¥Kuromaru¥Sii
¥ennoSessen¥Ci¥Ti¥houkouv
¥mTyokusen[l]¥Ti¥houkouv{}{}
¥ennoSessen¥Ci¥Si¥houkouv
¥mTyokusen[l]¥Si¥houkouv{}{}
¥end{zahyou}}
```

$\longrightarrow$



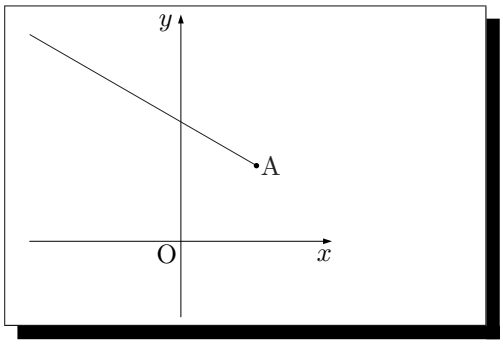
14.5 半直線

端点と方向角を六十分法で指定して半直線を描画するコマンドが ¥kHantyokusen です。

半直線（端点と方向角指定)

```
¥begin{zahyou}(-2,2)(-1,3)%
¥tenretu{A(1,1)e}%
¥def¥kaku{150}%
¥kHantyokusen¥A¥kaku
¥Kuromaru¥A
¥end{zahyou}%
```

$\longrightarrow$

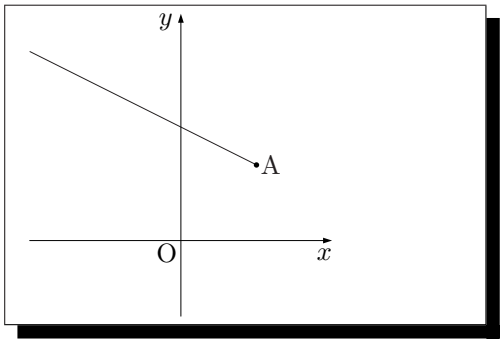


端点と方向ベクトルを指定する ¥mHantyokusen です。

半直線（端点と方向ベクトル指定)

```
¥begin{zahyou}(-2,2)(-1,3)%
¥tenretu{A(1,1)e}%
¥def¥houkou{(-2,1)}%
¥mHantyokusen¥A¥houkou
¥Kuromaru¥A
¥end{zahyou}%
```

$\longrightarrow$

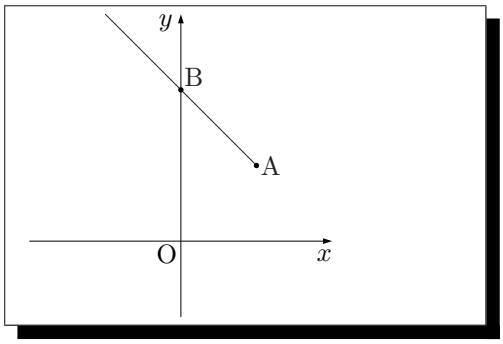


端点と通過する 1 点を指定する ¥Hantyokusen です。

半直線（端点と通過する 1 点指定)

```
¥begin{zahyou}(-2,2)(-1,3)%
¥tenretu{A(1,1)e;B(0,2)ne}%
¥Hantyokusen¥A¥B
¥kuromaru{¥A;¥B}%
¥end{zahyou}%
```

→



これらの半直線の線種を変更するには,

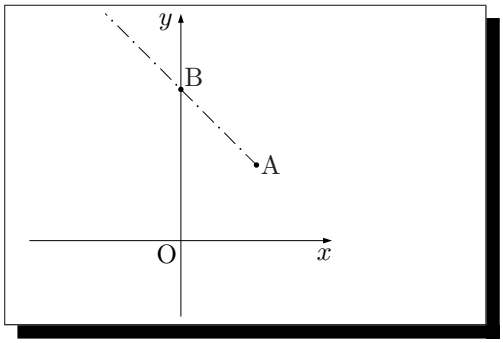
```
<sensyu=....>
```

オプションを利用します。一例です。

鎖線で半直線

```
¥begin{zahyou}(-2,2)(-1,3)%
¥tenretu{A(1,1)e;B(0,2)ne}%
¥Hantyokusen<%
sensyu=¥chainline[.2][.1]>%
¥A¥B
¥kuromaru{¥A;¥B}%
¥end{zahyou}%
```

→

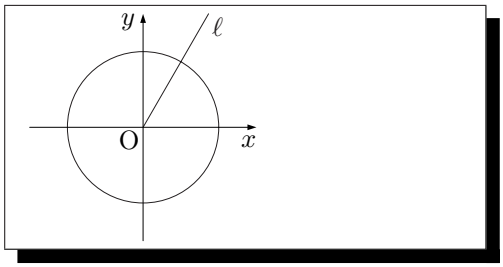


半直線の描画領域における端点を返すコマンドが ¥HtyokuT です。半直線にラベルをつけたりするのに便利でしょう。

¥HtyokuT

```
¥begin{zahyou}[ul=10mm]%
(-1.5,1.5)(-1.5,1.5)
  ¥En¥O{1}
  ¥kHantyokusen¥O{60}
  ¥Put¥HtyokuT[se]{$¥ell$}
¥end{zahyou}
```

→



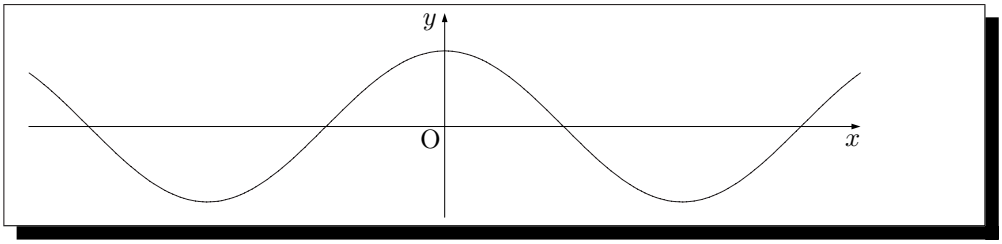
15　関数のグラフ

関数のグラフを描画するのに，Perl を用いる方法があります。

$y = \cos x$

```
¥begin{zahyou}[ul=10mm](-5.5,5.5)(-1.2,1.5)
  ¥def¥Fx{cos(X)}
  ¥YGraph¥Fx
¥end{zahyou}
```

→



関数式の記述が簡単でお勧めですが，機種に依存する面もありますので別ファイルにしてあります。samplePp.tex をご覧ください。

16　空間座標

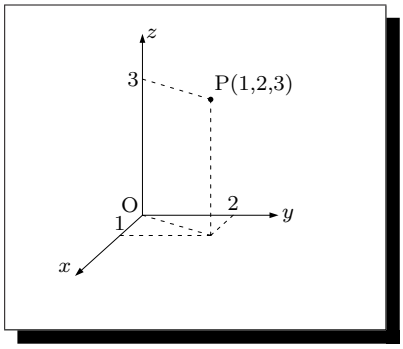
16.1　Zahyou 環境

zahyou 環境が平面座標を扱ったのに対して，Zahyou 環境は空間座標を扱います。
単純な例です。

空間座標

```
{¥unitlength6mm¥footnotesize
¥begin{Zahyou}(-1,3)(-1,3)(-1,4)%
¥def¥O{(0,0,0)}
¥def¥P{(1,2,3)}
¥def¥H{(1,2,0)}
¥def¥A{(1,0,0)}
¥def¥B{(0,2,0)}
¥def¥C{(0,0,3)}
¥iiiKuromaru¥P
¥iiiPut¥P[ne]{P(1,2,3)}
¥iiiDashline[40]{.1}{¥A¥H¥B}
¥iiiDashline[40]{.1}{¥O¥H¥P¥C}
¥iiiPut¥A[n]{1}
¥iiiPut¥B[n]{2}
¥iiiPut¥C[w]{3}
¥iiiPut¥O[nw]{O}
¥iiiPut{(¥Xmax,0,0)}[nw]{$x$}
¥iiiPut{(0,¥Ymax,0)}[e]{$y$}
¥iiiPut{(0,0,¥Zmax)}[e]{$z$}
¥end{Zahyou}}
```

$\longrightarrow$



簡単に解説します。
まず，Zahyou 環境は

x の範囲，y の範囲，z の範囲

を

(x の下限,x の上限),(y の下限,y の上限),(z の下限,z の上限)

の形で与えます。これらの値は，Zahyou 環境内では，順に

¥Xmin，¥Xmax，¥Ymin，¥Ymax，¥Zmin，¥Zmax

で引用することができます。
点は ¥def¥P{(1,2,3)}のように 3 次元ベクトルで与えます。
次いで，¥iiiKuromaru¥P で点 P に黒丸を打ちます。座標平面で定義された多くのコマンドの先頭に ‘iii’ をつけた空間座標用のコマンドが定義されています。現時点で定義されているコマンドは次の通りです。

¥iiiPut, ¥iiiPutStr
¥iiiBunten
¥iiiDrawline, ¥iiiDashline, ¥iiiArrowLine
¥iiiTyokkakukigou, ¥iiiHenKo
¥iiiKuromaru, ¥iiiSiromaru
¥iiiKyori, ¥iiiKyorii
¥iiiAddvec, ¥iiiSubvec, ¥iiiMulvec
¥iiiNuritubusi
¥iiibGurafu

空間特有のコマンドについては，次節以降でとりあげます。

さて，座標軸の向き，単位長の変更についてのオプション機能です。

Zahyou 環境の [...] オプションで x, y, z 軸の単位ベクトルを指定できるようにしてあります。ただし，それらは描画する平面を座標平面と見立てての成分表示です。デフォルトでは

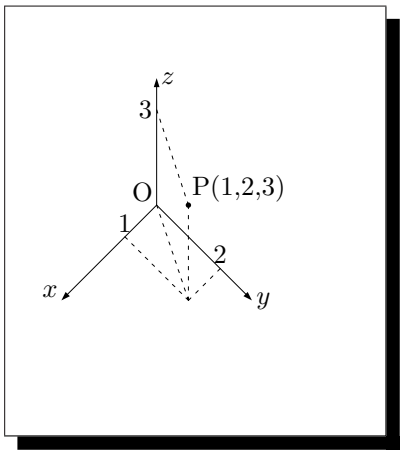
z **軸：**(0, 1)
y **軸：**(1, 0)
x **軸は** ¥kyokuTyoku(.667,-138)

となっています。これを変更してみます。

基本単位ベクトルの変更

```
{¥unitlength6mm
¥begin{Zahyou}%
  [(-.7,-.7)][(.7,-.7)][(0,.7)]%
  (-1,3)(-1,3)(-1,4)%
¥def¥O{(0,0,0)}
¥def¥P{(1,2,3)}
¥def¥H{(1,2,0)}
¥def¥A{(1,0,0)}
¥def¥B{(0,2,0)}
¥def¥C{(0,0,3)}
¥iiiKuromaru¥P
¥iiiPut¥P[ne]{P(1,2,3)}
¥iiiDashline[40]{.1}{¥A¥H¥B}
¥iiiDashline[40]{.1}{¥O¥H¥P¥C}
¥iiiPut¥A[n]{1}
¥iiiPut¥B[n]{2}
¥iiiPut¥C[w]{3}
¥iiiPut¥O[nw]{O}
¥iiiPut{(¥Xmax,0,0)}[nw]{$x$}
¥iiiPut{(0,¥Ymax,0)}[e]{$y$}
¥iiiPut{(0,0,¥Zmax)}[e]{$z$}
¥end{Zahyou}}
```

$\longrightarrow$



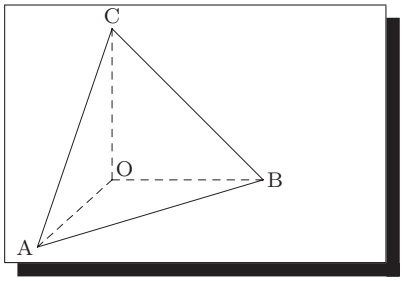
16.2 角錐

角錐を描画するのにコマンド ¥Kakusui を用意しました。O(0, 0, 0), A(1, 0, 0), B(0, 1, 0) を頂点とする三角形 OAB を底面とし，C(0, 0, 1) を頂点とする角錐—四面体を描画してみます。

角錐

```
{¥unitlength20mm¥footnotesize
¥Drawaxisfalse
¥begin{Zahyou}(0,1.1)(0,1)(0,1.1)
  ¥def¥O{(0,0,0)}
  ¥def¥A{(1,0,0)}
  ¥def¥B{(0,1,0)}
  ¥def¥C{(0,0,1)}
  ¥iiiPut¥A[w]{A}
  ¥iiiPut¥B[e]{B}
  ¥iiiPut¥O[ne]{O}
  ¥iiiPut¥C[n]{C}
  ¥Kakusui{AB}{O}{C}
¥end{Zahyou}}
```

→



ここでは，座標軸を描画しないように指定するのに，¥Drawaxisfalse としています。

¥Kakusui の書式です。

¥Kakusui<#1>#2#3#4

- #1 : **角錐台を描画する際の切断面の位置**（0〜1）
- #2 : **見える頂点列**
- #3 : **見えない頂点列**
- #4 : **錐の頂点**

図の見えない部分の点線のスタイルは，デフォルトでは

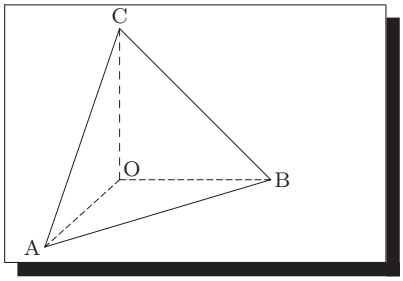
```
¥def¥iiiTensen{¥iiiDashline[40]{.1}}
```

となっていますが，これを再定義してみます。

¥iiiTensen の再定義

```
{¥unitlength20mm¥footnotesize
¥Drawaxisfalse
¥def¥iiiTensen{¥iiiDashline[80]{.05}}
¥begin{Zahyou}(0,1.1)(0,1)(0,1.1)
  ¥def¥O{(0,0,0)}
  ¥def¥A{(1,0,0)}
  ¥def¥B{(0,1,0)}
  ¥def¥C{(0,0,1)}
  ¥iiiPut¥A[w]{A}
  ¥iiiPut¥B[e]{B}
  ¥iiiPut¥O[ne]{O}
  ¥iiiPut¥C[n]{C}
  ¥Kakusui{AB}{O}{C}
¥end{Zahyou}}
```

→



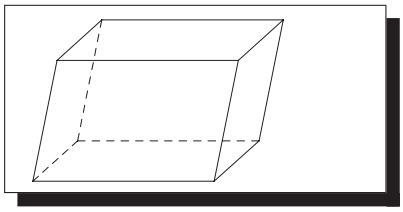
16.3 角柱

立方体，直方体，平行六面体を描画するために ¥Kakutyuu を用意してあります。平行六面体を描画してみます。

角柱

```
{¥unitlength8mm¥footnotesize
¥Drawaxisfalse
¥begin{Zahyou}[][][(.2,1)]%
(0,1.6)(0,3.1)(0,2.1)
  ¥def¥O{(0,0,0)}
  ¥def¥A{(1.5,0,0)}
  ¥def¥C{(0,3,0)}
  ¥def¥D{(0,0,2)}
  ¥iiiAddvec¥A¥C¥B
  ¥Kakutyuu{ABC}{O}{D}
¥end{Zahyou}}
```

→



直方体ではないことを示すため，z 軸方向の単位ベクトルを変更した斜交座標系を用いています。

¥Kakutyuu の書式です。

¥Kakutyuu#1#2#3
#1 : **見える頂点列**
#2 : **見えない頂点列**
#3 : **高さベクトル**

16.4 直線と平面の交点

直線と平面の交点を求めるコマンドは，次の 4 つを用意してあります。

¥PandL, ¥Pandl, ¥pandL, ¥pandl

ここで,

P は 3 点を指定した平面
p は点と法線ベクトルを指定した平面
L は 2 点を指定した直線
l は点と方向ベクトルを指定した直線

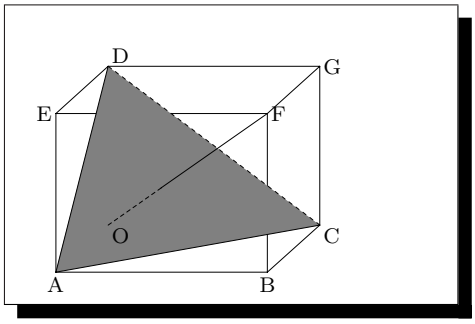
を意味します。

一例として直方体 OABC-DEFG の対角線 OF と 3 点 A, C, D を含む平面との交点 P を描画してみます。

直線と平面

```
{¥unitlength7mm¥footnotesize
¥Drawaxisfalse
¥begin{Zahyou}(0,3)(0,6)(0,4)
¥def¥O{(0,0,0)}
¥def¥A{(2,0,0)}
¥def¥C{(0,4,0)}
¥def¥D{(0,0,3)}
¥iiiAddvec¥A¥C¥B
¥iiiAddvec¥A¥D¥E
¥iiiAddvec¥C¥D¥G
¥iiiAddvec¥B¥D¥F
¥PandL¥A¥C¥D¥O¥F¥P
¥iiiKuromaru¥P
¥Kakutyuu{ABC}{O}{D}
¥iiiNuritubusi{¥A¥C¥D¥A}
¥iiiDrawline{¥C¥A¥D}
¥iiiDrawline{¥F¥P}
¥iiiDashline[60]{.1}{¥O¥P}
¥iiiDashline[60]{.1}{¥C¥D}
¥iiiPut¥O[se]{O}
¥iiiPut¥A[s]{A}
¥iiiPut¥B[s]{B}
¥iiiPut¥C[se]{C}
¥iiiPut¥D[ne]{D}
¥iiiPut¥E[w]{E}
¥iiiPut¥F[e]{F}
¥iiiPut¥G[e]{G}
¥end{Zahyou}}
```

$\longrightarrow$



書式です。

¥PandL#1#2#3#4#5#6
　3 点#1，#2，#3 を通る平面と，
　2 点#4，#5 を通る直線との交点を#6 に
¥Pandl#1#2#3#4#5#6
　3 点#1，#2，#3 を通る平面と，
　点#4 を通り方向ベクトルが#5 の直線との交点を#6 に
¥pandL#1#2#3#4#5
　点#1 を通り法線ベクトルが#2 の平面と，
　2 点#3，#4 を通る直線との交点を#5 に
¥pandl#1#2#3#4#5
　点#1 を通り法線ベクトルが#2 の平面と，
　点#3 を通り方向ベクトルが#4 の直線との交点を#5 に

16.5　垂線

座標平面で，点から直線に下した垂線の足を求めるコマンド ¥Suisen の 3 次元版の話しです。書式です。

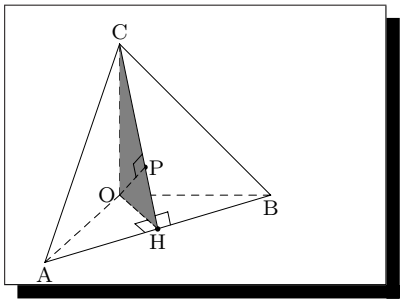
¥LSuisen#1#2#3#4
　点 #1 から直線 #2#3 へ下ろした垂線の足を #4 にセット
¥lSuisen#1#2#3#4
　点#1 から，点#2 を通り，方向ベクトルが#3 の直線への垂線
　の足を #4 にセット
¥pSuisen#1#2#3#4
　点#1 から，#2 を通り法線ベクトルが#3 である平面
　に下ろした垂線の足を#4 に与える。
¥PSuisen#1#2#3#4#5
　点#1 から，三点#2,#3,#4 を通る平面
　に下ろした垂線の足を#5 に与える。

例として，四面体 OABC の頂点 O から底面 ABC に下した垂線 OP と，O から線分 AB に下した垂線 OH を作図します。

垂線

```
{¥unitlength20mm¥footnotesize
¥Drawaxisfalse
¥begin{Zahyou}(0,1.2)(0,1.2)(0,1.2)
¥def¥O{(0,0,0)}
¥def¥A{(1,0,0)}
¥def¥B{(0,1,0)}
¥def¥C{(0,0,1)}
¥PSuisen¥O¥A¥B¥C¥P
¥LSuisen¥O¥A¥B¥H
¥iiiKuromaru¥P
¥iiiKuromaru¥H
¥Kakusui{AB}{O}{C}
¥iiiNuritubusi{¥O¥C¥H¥O}
¥iiiDrawline{¥C¥H}
¥iiiDashline[80]{0.05}{¥H¥O¥P}
¥iiiTyokkaku¥P[¥O]¥C
¥iiiTyokkaku¥H[¥O]¥A
¥iiiTyokkaku¥H[¥C]¥B
¥iiiPut¥A[s]{A}
¥iiiPut¥B[s]{B}
¥iiiPut¥C[n]{C}
¥iiiPut¥O[w]{O}
¥iiiPut¥P[e]{P}
¥iiiPut¥H[s]{H}
¥end{Zahyou}}
```

→



16.6 空間曲線

座標平面で媒介変数表示された曲線を描画するコマンド ¥bGurafu の 3 次元版が ¥iiibGurafu です。書式は

¥iiibGurafu(#1)(#2)#3#4#5#6#7

#1 : t **の刻み値（デフォルト値は** 0.05 **）**

#2 : **点線で描画するときの描画する部分の** t **のレンジ**

#3 : x=f(t)

#4 : y=g(t)

#5 : z=h(t)

#6 : t **の始め値**

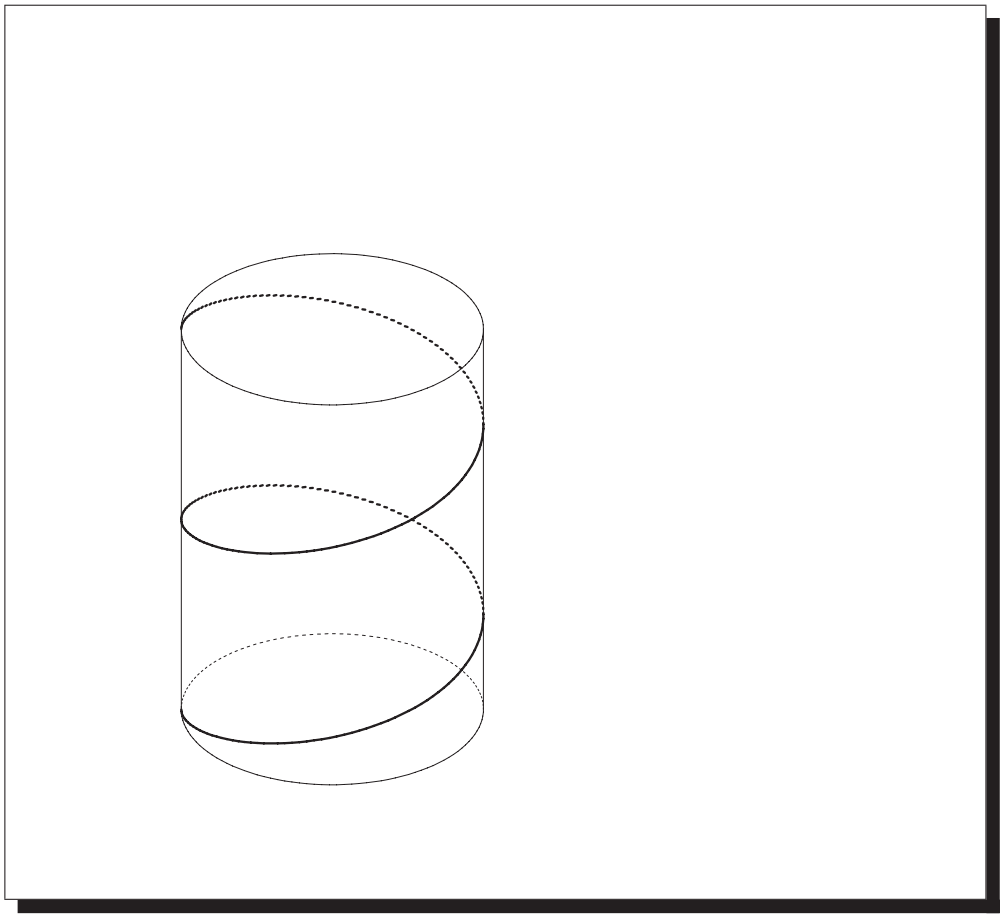
#7 : **終り値**

具体例として，円柱螺旋を描画してみましょう。

円柱螺旋

```
{¥unitlength20mm¥footnotesize
¥Drawaxisfalse
¥begin{Zahyou}[(-1,0)][(0,-.5)][(0,.2)]%
(-2,2)(-2,2)(-1,18)
¥def¥Fx#1#2{¥Cos{#1}#2}
¥def¥Fy#1#2{¥Sin{#1}#2}
¥def¥Fz#1#2{¥edef#2{#1}}
¥def¥Fo#1#2{¥edef#2{0}}
¥def¥Fi#1#2{¥edef#2{¥tmax}}
¥Mul3¥Pie¥Piii
¥Mul4¥Pie¥tmax
{¥thicklines
¥iiibGurafu¥Fx¥Fy¥Fz{0}{¥Pie}
¥iiibGurafu¥Fx¥Fy¥Fz{¥Pii}{¥Piii}
¥iiibGurafu(.05)(.02)¥Fx¥Fy¥Fz{¥Pie}{¥Pii}
¥iiibGurafu(.05)(.02)¥Fx¥Fy¥Fz{¥Piii}{¥tmax}}
¥iiibGurafu¥Fx¥Fy¥Fo{0}{¥Pie}
¥iiibGurafu(.05)(.02)¥Fx¥Fy¥Fo{¥Pie}{¥Pii}
¥iiibGurafu¥Fx¥Fy¥Fi{0}{¥Pii}
¥iiiDrawline{(1,0,0)(1,0,¥tmax)}
¥iiiDrawline{(-1,0,0)(-1,0,¥tmax)}
¥end{Zahyou}}
```

→



なお，空間曲線を近似する折れ線を得るコマンド ¥iiiBKinziOresen もありますが，これは perl との連携機能を必要とします。

17　作表

この節の環境，コマンド類は emathT.sty で定義されています。したがって，プリアンブルで

```
¥usepackage{emathT}
```

を宣言しておく必要があります.

17.1　列幅指定

表を作成するには，tabular 環境，array 環境があります．これらの環境で作成される表の列幅は，列の中に置かれる内容によって自動的に定まります.

表の幅

```
¥begin{tabular}{|*{4}{c|}}¥hline
     & かみ & いし & はさみ ¥¥¥hline
かみ &      & ○   & ×      ¥¥¥hline
いし & ×    &      & ○      ¥¥¥hline
はさみ & ○ & × &           ¥¥¥hline
¥end{tabular}
```

→

	かみ	いし	はさみ
かみ		○	×
いし	×		○
はさみ	○	×	

これは便利ですが，ときには列幅を指定したいこともあります．欄指定子 l/c/r を発展させて L/C/R を用意しました．これらは，列の幅を引数に取ります．その一例です.

列幅一定の表

```
¥begin{tabular}{|*{4}{C{4zw}|}}¥hline
     & かみ & いし & はさみ ¥¥¥hline
かみ &      & ○   & ×      ¥¥¥hline
いし & ×    &      & ○      ¥¥¥hline
はさみ& ○   & ×    &       ¥¥¥hline
¥end{tabular}
```

→

	かみ	いし	はさみ
かみ		○	×
いし	×		○
はさみ	○	×	

17.2 表の罫線を太く

17.2.1 ¥arrayrulewidth

表の罫線全部を太くするのは，LaTeX で用意されている ¥arrayrulewidth の値を変更することで実現できます。

¥arrayrulewidth の変更

```
¥arrayrulewidth1pt¥relax
¥begin{tabular}{|c|c|c|c|}¥hline
  A & B & C & D ¥¥¥hline
  1 & 2 & 3 & 4 ¥¥¥hline
  a & b & c & d ¥¥¥hline
¥end{tabular}
```

→

A	B	C	D
1	2	3	4
a	b	c	d

17.2.2 外枠のみを太く

外枠だけを太くしたい，など一部の罫線を太くするには，面倒な手順を踏まねばなりませんので，マクロ化することにしました。

横罫線を太くするために

```
¥hline, ¥cline
```

にかえて，それぞれ

```
¥hlineb, ¥clineb
```

を新設しました。これらの罫線の太さは ¥arrarulewidthb で指定します。デフォルトは 1pt としてあります。縦罫線を太くする位置には

```
|
```

にかえて

```
I
```

を用います。

では，これらを用いて外枠だけを太くしてみましょう。

外枠を太く

```
¥begin{tabular}{Ic|c|c|cI}¥hlineb
  A & B & C & D ¥¥¥hline
  1 & 2 & 3 & 4 ¥¥¥hline
  a & b & c & d ¥¥¥hlineb
¥end{tabular}
```

→

A	B	C	D
1	2	3	4
a	b	c	d

17.2.3　二重罫線との併用

さらに，見出し行・列と表の内容との境界を二重線にすることもできます。この部分は hhline.sty の一部を修正しています。なお，emathT.sty はその中で，array.sty と hhline.sty を読み込んでいます。

二重罫線も

```
¥begin{tabular}{Ic||c|c|cI}¥hlineb
  A & B & C & D ¥¥¥hhline{I=#=|=|=I}
  1 & 2 & 3 & 4 ¥¥¥hline
  a & b & c & d ¥¥¥hlineb
¥end{tabular}
```

→

A	B	C	D
1	2	3	4
a	b	c	d

17.2.4　太罫線の太さ

太罫線の太さは ¥arrayrulewidthb で決まりますから，もっと太くしたければ

¥arrayrulewidthb

```
¥arrayrulewidthb=2pt¥relax
¥begin{tabular}{Ic||c|c|cI}¥hlineb
  A & B & C & D ¥¥¥hhline{I=#=|=|=I}
  1 & 2 & 3 & 4 ¥¥¥hline
  a & b & c & d ¥¥¥hlineb
¥end{tabular}
```

→

A	B	C	D
1	2	3	4
a	b	c	d

17.2.5　特定のブロック枠を太く

一部分を太罫線で

```
¥begin{tabular}{|c|c|c|c|}¥hlineb
  A & B & C & D ¥¥¥hline
  ¥noalign{¥vskip-¥arrayrulewidth}
  ¥clineb{2-3}
  ¥multicolumn{1}{|cI}{1}
    & ¥multicolumn{1}{c|}{2}
    & ¥multicolumn{1}{cI}{3}
    & 4 ¥¥
  ¥noalign{¥vskip-¥arrayrulewidthb
  ¥vskip¥arrayrulewidth}
  ¥clineb{2-3}
  ¥noalign{¥vskip¥arrayrulewidthb
  ¥vskip-¥arrayrulewidth}¥hline
  a & b & c & d ¥¥¥hlineb
¥end{tabular}
```

→

A	B	C	D
1	2	3	4
a	b	c	d

17.3 罫線を点線で

罫線を点線で引くには，arydshln.sty を用いることが出来ます。しかし，このスタイルファイルは hhline.sty と相性が悪いので，emathT.sty との併用については，emathWiki の arydshln-LO-T なるページをご覧ください。

17.4 カラムに斜線

また，この環境内では，¥emTsya コマンドで欄に斜線を引くことができます.

カラムに斜線

```
¥begin{tabular}{|L{6zw}|*{3}{C{4zw}|}}¥hline
¥emTsya[r]<¥hyoumidasi{甲}{乙}> & かみ & いし & はさみ ¥¥¥hline
かみ & ¥emTsya[r] & ○ & × ¥¥¥hline
いし & × & ¥emTsya[r] & ○ ¥¥¥hline
はさみ & ○ & × & ¥emTsya[r] ¥¥¥hline
¥end{tabular}
```

→

甲＼乙	かみ	いし	はさみ
かみ		○	×
いし	×		○
はさみ	○	×	

詳しくは emathWiki の emTsya なるページをご覧ください。

18 囲み

18.1 rectbox 環境

この節で紹介する rectbox 環境は emathPb.sty で定義されています。
版面，左右いっぱいに拡がる罫線囲みです。

rectbox 環境

```
¥begin{rectbox}[item=~見出し~]
ああああああああああああああああああ
ああああああああああああああああああ
ああああああああああああああああああ

いいいいいいいいいいいいいいいいいい
いいいいいいいいいいいいいいいいいい
いいいいいいいいいいいいいいいいいい
いいいいいいいいいいいいいいいいいい
¥end{rectbox}
```

↓

見出し

ああ

いいい

詳しくは，emathWiki の「枠囲み」のページをご覧ください。

19　その他

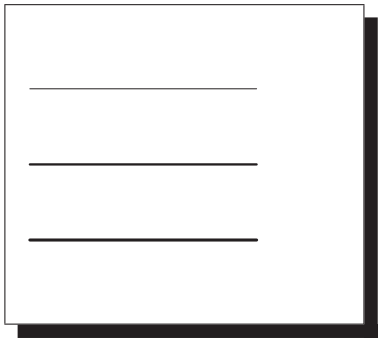
19.1　線の太さ

線の太さは 3 種類用意されています。（デフォルトは ¥thinlines です。）

線の太さ

```
¥begin{picture}(3,4)%
¥thinlines
¥put(0,3){¥drawline(0,0)(3,0)}%
{¥thicklines
¥put(0,2){¥drawline(0,0)(3,0)}}%
{¥Thicklines
¥put(0,1){¥drawline(0,0)(3,0)}}%
¥end{picture}
```

⟶



もっと太くしたい，というときは ¥allinethickness というコマンドも eepic.sty で用意されています。引数に太さを与えます。

¥allinethickness

```
¥begin{picture}(3,2)%
{¥allinethickness{2pt}%
¥put(0,1){¥drawline(0,0)(3,0)}}%
¥end{picture}
```

⟶

索 引