Perl と連携したグラフ描画

emathPp.sty ver.0.39

tDB

2005/11/07

概 要

Perl と連携してグラフを描画する手法を提供します。ただし，OS 依存です。

目 次

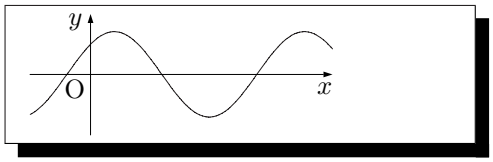
1　なぜ perl か

emathP.sty で $y = f(x)$ のグラフを描画するのに，¥yGurafu というコマンドがあります。
$y = \sin x + \cos x$ のグラフを描いてみましょう。

¥yGurafu

```
¥begin{zahyou}[ul=4mm](-2,8)(-2,2)
¥def¥Fx#1#2{%
  ¥Sin{#1}¥yi
  ¥Cos{#1}¥yii
  ¥Add¥yi¥yii#2}
¥yGurafu¥Fx¥xmin¥xmax
¥end{zahyou}
```

→



関数式の記述が面倒です。sin(x)+cos(x) と簡潔に記述したいのですが，TEX には計算式を解析する機能がありません。それをマクロで実現するのは，所用時間を考えても得策とは思えません。ということで計算を perl というソフトに外注することにしました。その結果，上のリストは

¥YGurafu

```
¥begin{zahyou}[ul=4mm](-2,8)(-2,2)
¥def¥Fx{sin(X)+cos(X)}
¥YGurafu¥Fx¥xmin¥xmax
¥end{zahyou}
```

と簡潔に表すことができます。

ただし，実行するにはこれから述べる準備が必要です。

2　perl とは何か

Larry Wall 氏の開発したテキスト処理用の言語で，計算機能ももっています。元来は，UNIX 用のツールとして開発されましたが，Windows, Macintosh にも移植されています。

3　perl の入手先

UNIX, Mac OS X では，標準配布です。

Windows, Macintosh (Mac OS 9) の場合は，次のサイトからダウンロードして適当なフォルダにインストールします。perl.exe の存在するフォルダに PATH を通して，perl 単体で動作することを確認しておきます。

3.1　Windows

http://www.activestate.com/Products/ActivePerl/

3.2 Macintosh (OS 9)

http://world.std.com/ habilis/MacJP5.j.htm

4 emath で perl を使用する準備

4.1 emath.pl

emath.pl, degsine.pl, nCr.pl というファイルを，perl をインストールしたフォルダの下にある 'lib' フォルダ内に置きます。(Mac OS X では /Library/Perl)

emath.pl などは emath のホームページ,

「その他」

2. perl と連携するための補助ファイル emath.pl など

にあります。

4.2 ¥write18

Windows, Unix 系 OS では，tex から子プロセスを起動することが出来ます。ただし，デフォルトではこの機能は働かないようになっています。

その機能を有効にするためには

```
platex -shell-escape
```

または省略形

```
platex -shell
```

もっと省略して

```
platex -sh
```

とオプションをつけて platex を起動する必要があります。

【注意】この機能を利用するのは，危険を伴います。信用の出来るソースリストに対してのみ，実行してください。

5 perl を用いたグラフ描画

では，先ほどの $y = \sin x + \cos x$ のグラフなどを描画してみましょう。

その前にご注意。この方式では補助の作業用ファイルが作られます。同名のファイルがあると削除されますからご注意ください。そのファイル名は

```
temp.pl
```

タイプセットしているファイルと同名で拡張子が

.d1, .d2, .d3, ... **（ perl を呼び出した回数分だけ）**

5.1 Windows, UNIX, Mac OS X の場合

同梱の ex1.tex をタイプセットしてみましょう。

DOS プロンプトで ex1.tex が存在しているディレクトリを
カレントディレクトリにして

```
platex -shell-escape ex1.tex
```

と打ち込みます。ただし，次のファイルが作成されます。同名のファイルは削除されますからご注意ください。

```
temp.pl
ex1.d1
```

$y = \sin x + \cos x$ のグラフが描画されましたか。エラーが起きる場合は，前節までの準備を再確認してください。

次いで，ex2.tex は $y = \log_2 x$ のグラフを描画します。作業用ファイルは次の 3 個です。

```
temp.pl
ex2.d1, ex2.d2
```

こちらもうまくいったでしょうか。

5.2 Mac OS 9 の場合

5.2.1 一般的な手順

Windows, UNIX, Mac OS X の場合には TEX から perl を起動することができるのですが，Macintosh OS 9 の場合その方法は使えません。

そこで次善の策として次の 3 ステップに分けて処理をすることになります。

手順 1. TEX で perl に引き渡すスクリプトファイルを作成する。

手順 2. 手順 1 で作成されたスクリプトファイルを perl で処理する。

手順 3. 再び TEX で手順 2 で作成された描画データを受け取りグラフをかく。

5.2.2 具体例 — ex1.tex

では，$y = \sin x + \cos x$ のグラフを描く，同梱の ex1.tex を処理してみます。

作業用ファイルとして

```
temp.pl
ex1.d1
```

が作成されます。同名のファイルが存在している場合，削除されてしまいます。ご注意ください。

手順 1. ex1.tex をタイプセットします。グラフは描画されず，座標軸のみが描かれます。

手順 2. 手順 1 の結果，temp.pl という，perl のスクリプトファイルが作成されます。これを perl で処理します。その結果は ex1.d1 なるファイルに書き出され，次の手順で TEX が利用してグラフを描画します。

手順 3. 再度 ex1.tex をタイプセットします。今度はグラフが描かれるはずです。

さていかがでしょうか。うまくいかないときは前節までの準備作業をもう一度見直してください。それでも駄目な場合は，emath の BBS などでお尋ねください。

手順 1 と手順 3 の違いは

```
ex1.d1
```

というファイルが存在しているかいないかで判断されます。

存在していない場合は，手順 1 の段階であると判断され，グラフ描画は行わず，perl 用のスクリプトファイルを作成します。

それに対して，存在している場合は手順 3 の段階であると判断され，グラフ描画に進むことになります。

したがって，いったん ex1.d1 が作成されると手順 1 は行われませんから，このままではグラフの修正ができません。

例えば，ex1.tex で ¥unitlength を 10mm から 8mm に変更したいという場合，ソースリストを書き換えてタイプセットしても，既に作成済みのデータファイル ex1.d1 を使ってグラフ描画が行われますから，グラフは前のままとなってしまいます。

では，どうするかといえば，ex1.d1 を削除した上で，手順 1, 2, 3 を繰り返すということになるのです。

5.2.3 具体例 — ex2.tex

では，次の例に進みましょう。これは面倒の 2 乗 (!?) です。

作業用ファイルは

```
temp.pl
ex2.d1, ex2.d2
```

です。同名のファイルが存在している場合，削除されてしまいます。ご注意ください。

では，はじめましょう。

手順 1. ex2.tex をタイプセットします。なんと！ エラー

```
! #calcval ---> ex2.d1.
```

が発生します。しかし，perl 用のスクリプトファイル temp.pl は作成されていますから，

手順 2. temp.pl を perl で処理します。

エラーが起きた場合は，手順 1, 2 をエラーが発生しなくなるまで繰り返します。すなわち

手順 1′. ex2.tex をタイプセットします。今度はエラーは起きないはずです。

手順 2′. 再度 temp.pl を perl で処理します。

やっと最終手順 3 を実行する段階にたどり着きました。

手順 3. 再度 ex2.tex をタイプセットします。今度はグラフが描かれるはずです。

ex2.tex では，ex1.tex と違って，なぜこんなに面倒になってしまったのかを説明します。
ex2.tex では，perl と連携するコマンドが 2 回呼ばれますが，

```
(1) ¥calcval{1/log(2)}¥lgii
(2) ¥YGurafu¥Fx{0.125}¥xmax
```

(1) は perl に $\frac{1}{\log 2}$ の値を計算させて，その結果を ¥lgii に受け取ります。ところが手順 1 の段階ではその計算をするスクリプトファイル— temp.pl —を作るだけで計算は行われません。つぎの (2) のグラフ描画ではこの計算結果を必要とするのでスクリプトファイルの作りようがありません。そこでエラー終了をさせることにしました。
手順 2 で計算結果が ex2.d1 に記録されます。
手順 1′ では，ex2.d1 が存在していますから，(1) のコマンドで計算結果が ¥lgii に読み込まれ，(2) のグラフ描画用のスクリプトファイル—temp.pl —を作成することができます。
手順 2′ でグラフ描画用データ ex2.d2 が作成され，描画準備が完了し，最終の手順 3 に進むことができる，となるわけです。
おおしんど。
このように，戻り値を必要とするコマンド

```
¥calcval
¥funcval
¥YTen
¥BTen
¥RTen
```

を使用すると，そこでエラー終了します。エラーが出なくなるまで手順 1, 2 を繰り返し，手順 3 に進むことになります。

6　コマンドの紹介

Perl を用いたコマンドの紹介をします。

6.1　関数値の計算

6.1.1　¥calcval

計算式を与えて計算結果を受け取るためのコマンドです。書式は

¥calcval#1#2
　#1 : **計算式（文法は** Perl **に従います。）**
　#2 : **計算結果を受け取る制御綴。**

一例です。

¥calcval

```
¥calcval{exp(-1)}¥x
$¥bunsuu1e$の近似値は ¥x です。
```

→

$\frac{1}{e}$ の近似値は 0.367879 です。

6.1.2　¥funcval

関数式，独立変数の値を与えて関数値を受け取るコマンドです。

¥funcval#1#2#3
　#1 : **関数式** y=f(x)**（文法は** Perl **に従います。）**
　#2 : x **の値**
　#2 : y **の値を受け取る制御綴。**

一例です。

¥funcval

```
¥def¥Fx{log2(10,X)}
¥funcval¥Fx{2}¥xii
¥funcval¥Fx{3}¥xiii
2，3 の常用対数の近似値はそれぞれ
¥xii，¥xiii です。
```

→

2，3 の常用対数の近似値はそれぞれ 0.301030, 0.477121 です。

なお，log2 は Perl にはありませんが，emath.pl で追加してあります。このスクリプトファイルで追加している機能は次のとおりです。

定数：円周率 $pi
関数：正接 tan(X)
対数 log2(a,X) : a **を底とする対数**

6.1.3 戻り値の書式

¥calcval, ¥funcval の戻り値の書式を制御するオプションです。まずは，現状の確認です。

¥calcval

```
¥calcval{68/2}¥y
¥y
```

→ 34.000000

すなわち結果が整数でも，小数点を伴った文字列が戻ってきます。これは

```
printf"%f",....
```

と，Perl の出力フォーマットが ‘f’ 指定になっているからです。
そこで，¥calcval, ¥funcval に [#1] オプションをつけたときは

```
printf"%#1",....
```

の書式で出力することが出来るようにしました。先ほどの出力を整数に指定してみます。

[d] オプション

```
¥calcval[d]{68/2}¥y
¥y
```

→ 34

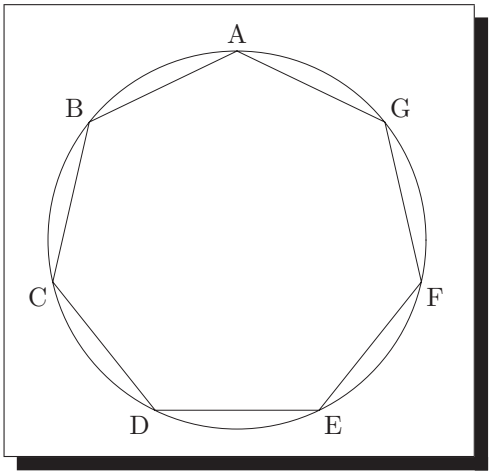
6.1.4 ¥tenretu への応用

この機能を利用して，¥tenretu などの座標に計算式を入れ，perl と連携させることができます。

正七角形

```
¥begin{zahyou*}[ul=25mm]%
      (-1.1,1.1)(-1.1,1.2)
  ¥rtenretu<perl>{%
    A(1,90)n;
    B(1,90+360/7)nw;
    C(1,90+2*360/7)sw;
    D(1,90+3*360/7)sw;
    E(1,90+4*360/7)se;
    F(1,90+5*360/7)se;
    G(1,90+6*360/7)ne}
  ¥Drawline{¥A¥B¥C¥D¥E¥F¥G¥A}
  ¥En¥O{1}%
¥end{zahyou*}
```

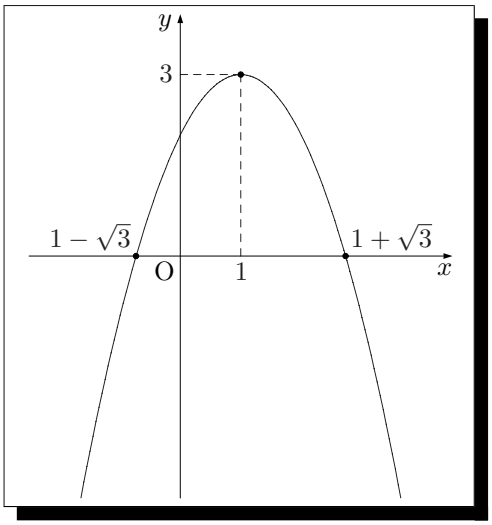
→



二次関数のグラフと x 軸

```
¥begin{zahyou}[ul=8mm]%
      (-2.5,4.5)(-4,4)
  ¥def¥Fx{2+2*X-X*X}
  ¥tenretu*{C(1,3)}
  ¥Put¥C[syaei=xy]{}
  ¥tenretu*<perl>{%
    A(1-sqrt(3),0);B(1+sqrt(3),0)}
  ¥Put¥A[nw]{$1-¥sqrt{3}$}
  ¥Put¥B[ne]{$1+¥sqrt{3}$}
  ¥YGurafu*¥Fx
  ¥kuromaru{¥A;¥B;¥C}
¥end{zahyou}
```

→



6.2 グラフ描画

6.2.1 ¥YGurafu

最初に触れた $y = f(x)$ のグラフを描画します。書式は ¥yGurafu とほとんど同じです。
関数式の記述において，独立変数 x は 'X' を用います。

¥YGurafu(#1)(#2)#3#4#5

#1 : x **の刻み値 デフォルト**=0.05

#2 : **点線で描画するときの描画する部分の** x **のレンジ**

#3 : **関数式** y=f(X) **独立変数は**‘X’**を使用します。**

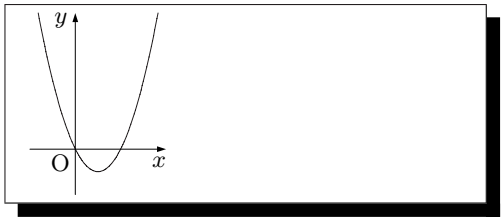
#4 : x **の始め値**

#5 : x **の終り値**

まずは，簡単な 2 次関数のグラフを描画する例です。

```
¥YGurafu
¥unitlength3mm¥small
¥begin{zahyou}(-2,4)(-2,6)
¥YGurafu{X*(X-2)}{¥xmin}{¥xmax}
¥end{zahyou}
```

→

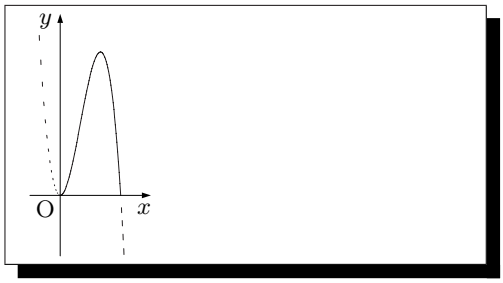


この場合，zahyou 環境における y の範囲が不十分ですから，範囲外の曲線はクリップされてしまいます。

続いて $y = x^2(4 - x)$ のグラフを一部分点線で描画する例です。

```
¥YGurafu
¥unitlength2mm¥small
¥begin{zahyou}(-2,6)(-4,12)
¥def¥Fx{X**2*(4-X)}
¥YGurafu¥Fx{0}{4}
¥YGurafu(.05)(.02)¥Fx{4.05}¥xmax
¥YGurafu(.1)(.02)¥Fx¥xmin{0}
¥end{zahyou}
```

→



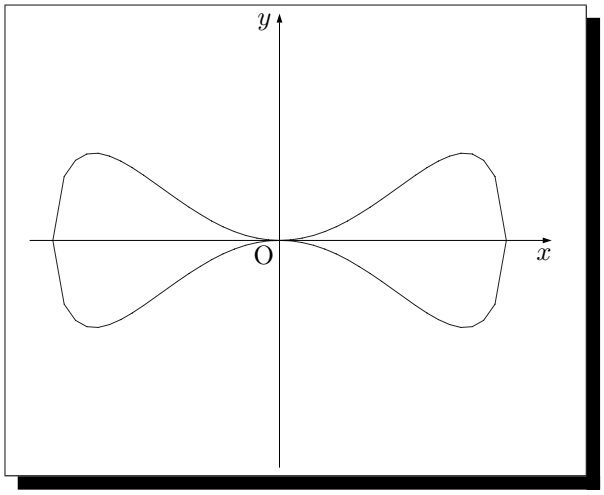
なお，整関数のグラフを描画するには ¥Gurafu コマンドがありますが，¥YGurafu の場合は描画範囲外を自動的にクリップしてくれる点が便利です。

¥YGurafu において，デフォルトの x の刻み値は 0.05¥unitlength となっています。これは ¥unitlength=10mm を想定していますから，[ul=30mm] に対しては 0.15mm となり，少し荒くなります。

デフォルト

```
¥begin{zahyou}[ul=30mm]
  (-1.1,1.2)(-1,1)%
¥def¥Fx{X**2*sqrt(1-X**2)}%
¥def¥Gx{-X**2*sqrt(1-X**2)}%
¥YGurafu¥Fx{-1}{1}
¥YGurafu¥Gx{-1}{1}
¥end{zahyou}
```

→

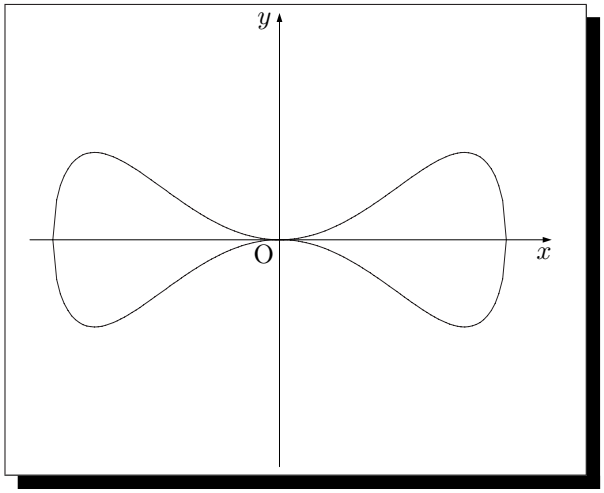


そこで，(0.01667) オプションを附加してみますと

x の刻み値を小さく

```
¥begin{zahyou}[ul=30mm]
  (-1.1,1.2)(-1,1)%
¥def¥Fx{X**2*sqrt(1-X**2)}%
¥def¥Gx{-X**2*sqrt(1-X**2)}%
¥YGurafu(0.01667)¥Fx{-1}{1}
¥YGurafu(0.01667)¥Gx{-1}{1}
¥end{zahyou}
```

→



少しは改善されましたが，$(\pm 1,\, 0)$ のところで折れている感じは残ります。そこで，

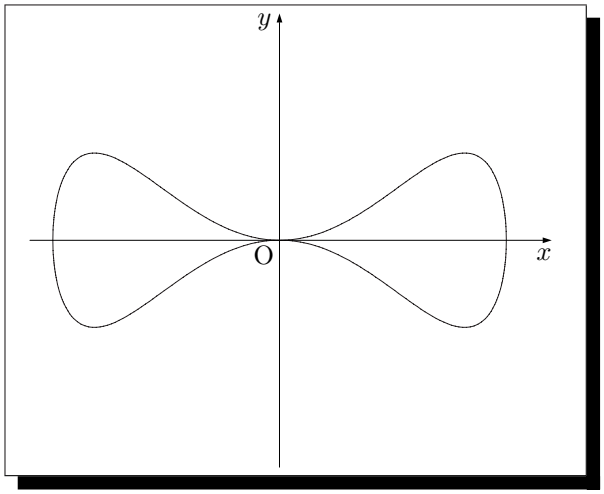
```
emathPp.sty ver. 0.21 2003/11/07
```

において，¥YGurafu に (*) オプションを新設しました。このオプションは，y の変化の大きいところでは，自動的に x の刻みを小さくする機能を実現しています。

新設 (*) オプション

```
¥begin{zahyou}[ul=30mm]
  (-1.1,1.2)(-1,1)%
¥def¥Fx{X**2*sqrt(1-X**2)}%
¥def¥Gx{-X**2*sqrt(1-X**2)}%
¥YGurafu(*)¥Fx{-1}{1}
¥YGurafu(*)¥Gx{-1}{1}
¥end{zahyou}
```

→

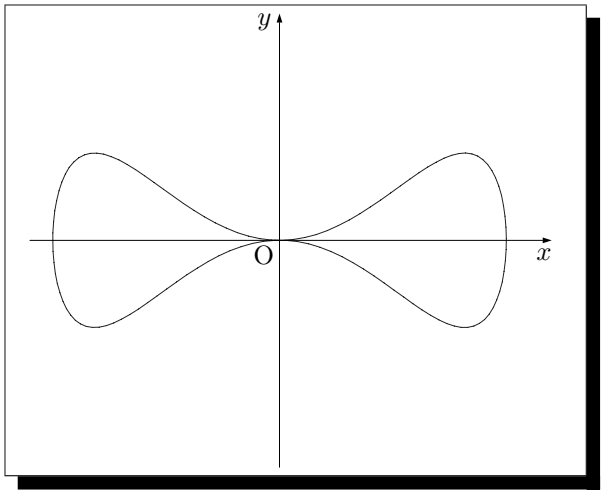


この場面における最善の対応法は，媒介変数表示を用いることであることは論を待ちません。
媒介変数表示がうまく見つからない，
極座標表示も駄目，
というような場合の力技を用意した，ということです。

媒介変数表示

```
¥begin{zahyou}[ul=30mm]
  (-1.1,1.2)(-1,1)%
¥def¥Fx{cos(T)}%
¥def¥Fy{cos(T)**2*sin(T)}%
¥BGurafu¥Fx¥Fy{-$pi}{$pi}
¥end{zahyou}
```

→



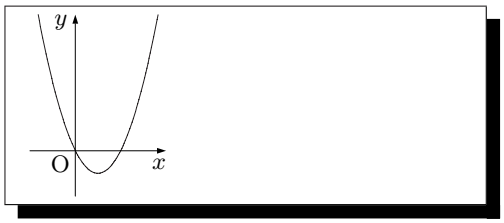
6.2.2 ¥YGurafu*

グラフを描画する x の範囲が {¥xmin}{¥xmax} であるとき，これをいちいち記述するのは面倒だから省略しようというのが ¥YGurafu*です。前節最初の例を ¥YGurafu*を用いると

¥YGurafu*

```
¥unitlength3mm¥small
¥begin{zahyou}(-2,4)(-2,6)
¥YGurafu*{X*(X-2)}
¥end{zahyou}
```

→



¥YGurafu*の書式です。

¥YGurafu*[#1](#2)(#3)#4

#1 : key=val **の形式で**

hidarix=... **描画区間の左端の** x **の値を指定します。**

migix=... **描画区間の右端の** x **の値を指定します。**

#2 : x **の刻み値 デフォルト**=0.05

#3 : **点線で描画するときの描画する部分の** x **のレンジ**

#4 : **関数式** y=f(X) **独立変数は‘X’を使用します。**

備考 このコマンドが実行されると，描画したグラフの

左端点が ¥hidariT

右端点が ¥migiT

と定義される。

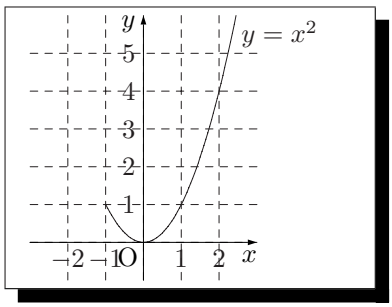
独立変数の範囲を指定する引数がなくなりましたが，[#1] オプションが附加されています。これを利用する例です。

右は描画領域一杯に描きたいが，左は制限したい，などというときは¥YGurafu*の [hidarix=...] オプションを与えます。

[hidarix=...] オプション

```
¥begin{zahyou}[ul=5mm](-3,3)(-1,6)
¥zahyouMemori[g]
¥YGurafu*[hidarix=-1]{X*X}
¥Put¥migiT[se]{$y=x^2$}
¥end{zahyou}
```

→

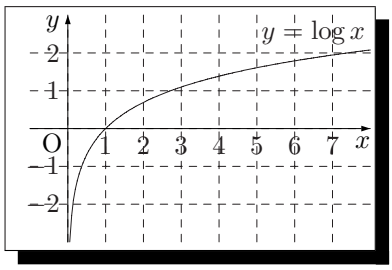


これは，$y = \log x$ などの場合に特に有効です。

$y = \log x$

```
¥begin{zahyou}[ul=5mm](-1,8)(-3,3)
¥zahyouMemori[g]
¥YGurafu*[hidarix=0]{log(X)}
¥Put¥migiT[nw]{$y=¥log x$}
¥end{zahyou}
```

→



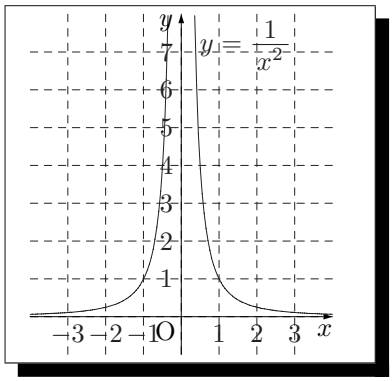
hidarix=0 と与えた値は $y = \log x$ の定義域に入りません。マクロ処理では，与えられた区間の中点における関数値が描画領域に入るか否かを調べる 2 分法で描画すべき左右の端点 ¥hidariT, ¥migiT を求めています。¥YGurafu*コマンドを実行すると，グラフの左右の端点が定義されていますから，これを利用して式を表示したりすることができます。

右端を制限する [migix=...] オプションも使えます。

$y = \dfrac{1}{x^2}$

```
¥begin{zahyou}[ul=5mm](-4,4)(-1,8)
¥zahyouMemori[g]
¥YGurafu*[migix=0]{1/(X*X)}
¥YGurafu*[hidarix=0]{1/(X*X)}
¥Put¥hidariT[se]{$y=¥bunsuu{1}{x^2}$}
¥end{zahyou}
```

→



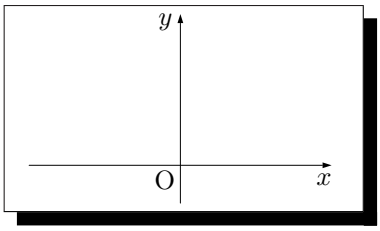
『注意』

¥YGurafu*で描画する関数が描画領域の一部にしか現れない場合，グラフがまったく描画されない場合があります。一例です。

¥YGurafu*が無効な場合

```
¥begin{zahyou}[ul=5mm](-4,4)(-1,4)
  ¥def¥Fx{(X+2)**2+1}
  ¥YGurafu*¥Fx
¥end{zahyou}
```

→



¥YGurafu*は ¥YGurafu の手抜き版ですが，次の条件を満たさなければなりません。

描画区間の中点において，グラフは描画領域に属している。

上の例では，描画区間 $-4 \leqq x \leqq 4$ の中点 $x = 0$ における関数値 $y = 5$ は y の描画区間 $-1 \leqq y \leqq 4$ に属していません。

今回の改定で，このような場合，ログファイルに Warning を出すことにしました。

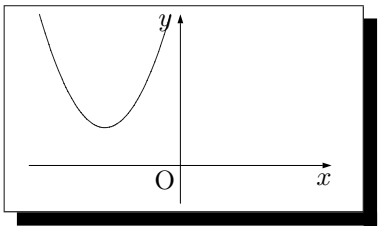
```
LaTeX Warning: YGurafu* : 描画範囲を求めることが出来ませんでした。
  hidarix=, migix= オプションで描画範囲を狭めて下さい。 on input line ***.
```

Warning にあるように，上の条件が満たされるように描画領域を狭めてやります。

修正 1

```
¥begin{zahyou}[ul=5mm](-4,4)(-1,4)
  ¥def¥Fx{(X+2)**2+1}
  ¥YGurafu*[migix=0]¥Fx
¥end{zahyou}
```

→



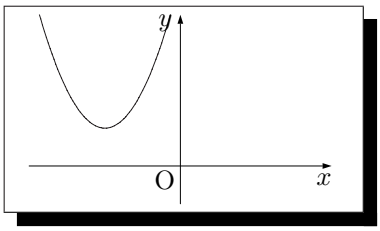
すなわち，オプション [migix=0] で，x の描画区間を $-4 \leqq x \leqq 0$ と狭めてやれば，その中点 $x = -2$ における関数値 $y = 1$ は y の描画領域に属しますから，グラフが描画されることとなりました。

描画区間を狭めるのが面倒なら，¥YGurafu を用いるのもあります。

修正 2

```
¥begin{zahyou}[ul=5mm](-4,4)(-1,4)
  ¥def¥Fx{(X+2)**2+1}
  ¥YGurafu¥Fx{}{}
¥end{zahyou}
```

→



6.2.3 ¥XGurafu

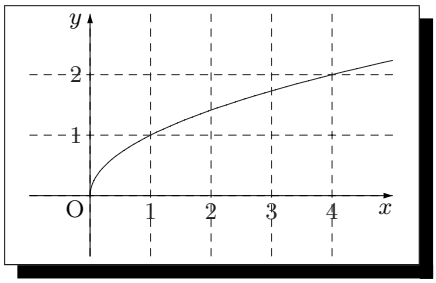
$x = f(y)$ で与えられた曲線を描画します。独立変数は 'Y' または 'X' を用います。

下の例は，$y = \sqrt{x}$ を $x = y^2$ として描画する例です。

¥XGurafu

```
¥begin{zahyou}[ul=8mm](-1,5)(-1,3)
  ¥small¥zahyouMemori[g]
  ¥XGurafu{Y*Y}{0}{¥ymax}
¥end{zahyou}
```

→

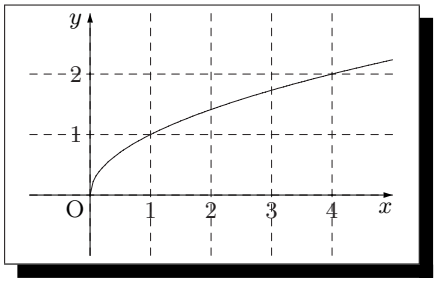


これを $y = \sqrt{x}$ として ¥YGurafu を用いると

$y=\sqrt{x}$

```
¥begin{zahyou}[ul=8mm](-1,5)(-1,3)
  ¥small¥zahyouMemori[g]
  ¥YGurafu{sqrt(X)}{0}{¥xmax}
¥end{zahyou}
```

→



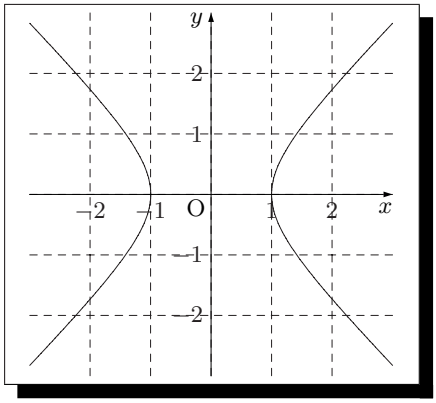
原点の付近がぎごちなくなります。

同様に，双曲線 $x^2-y^2=1$ も $x=\sqrt{1+y^2}$ とする方が滑らかに描けます。¥XGurafu*を用いてみます。

$x^2-y^2=1$

```
¥begin{zahyou}[ul=8mm](-3,3)(-3,3)
  ¥small¥zahyouMemori[g]
  ¥def¥Fx{sqrt(1+Y*Y)}
  ¥XGurafu*{¥Fx}
  ¥XGurafu*{-¥Fx}
¥end{zahyou}
```

→

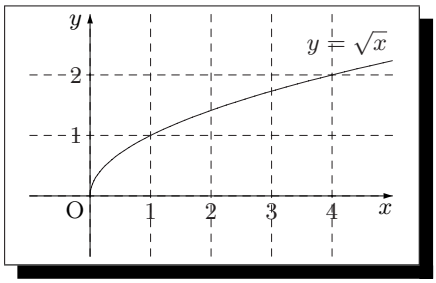


なお，¥XGurafu において，y の範囲の上限，下限を指定する key は uey，sitay です。またグラフの上端，下端が ¥ueT，¥sitaT に定義されています。

¥XGurafu*

```
¥begin{zahyou}[ul=8mm](-1,5)(-1,3)
  ¥small¥zahyouMemori[g]
  ¥XGurafu*[sitay=0]{Y*Y}
  ¥Put¥ueT[nw]{$y=¥sqrt x$}
¥end{zahyou}
```

→



6.2.4 ¥BGurafu

媒介変数表示された曲線を描画するコマンド ¥bGurafu に対しても Perl を使用するコマンド ¥BGurafu を用意しました。媒介変数は 'T' で表します。

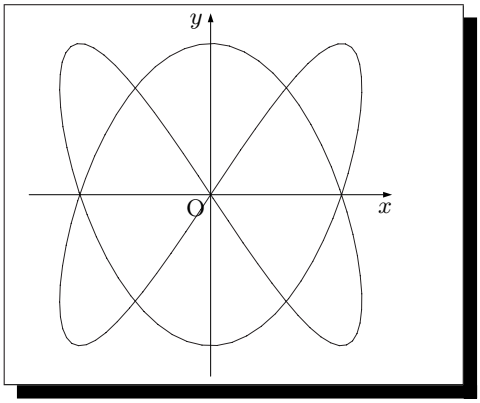
¥BGurafu(#1)(#2)#3#4#5#6
#1 : t の刻み値（デフォルト値は 0.05 ）
#2 : 点線で描画するときの描画する部分の t のレンジ
#3 : x=f(t) 媒介変数は T と表記する。（X, Y も可）
#4 : y=g(t)
#5 : t の始め値
#6 : 終り値

リサージュ曲線の一つです。

¥BGurafu

```
¥unitlength20mm¥small
¥begin{zahyou}(-1.2,1.2)(-1.2,1.2)
¥BGurafu{sin(2*T)}{sin(3*T)}0{2*$pi}
¥end{zahyou}
```

→



6.2.5 ¥RGurafu

極座標平面で極方程式 $r = f(\theta)$ で表される曲線を描画するコマンド ¥rGurafu に対しても Perl を使用するコマンド ¥RGurafu を用意しました。

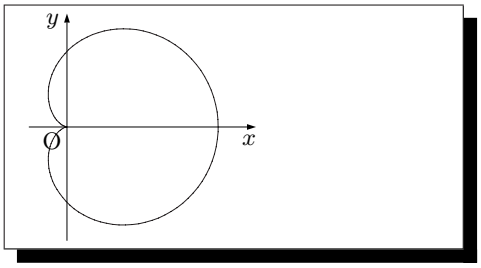
¥RGurafu(#1)(#2)#3#4#5
#1 : t の刻み値（デフォルト値は 0.05 ）
#2 : 点線で描画するときの描画する部分の t のレンジ
#3 : r=f(Θ) : Θも T で記述する。
#4 : Θ の始め値
#5 : 終り値

$r = 1 + \cos\theta$ を描画します。

¥RGurafu

```
¥unitlength10mm¥small
¥begin{zahyou}(-.5,2.5)(-1.5,1.5)
¥RGurafu{1+cos(T)}0{2*$pi}
¥end{zahyou}
```

→



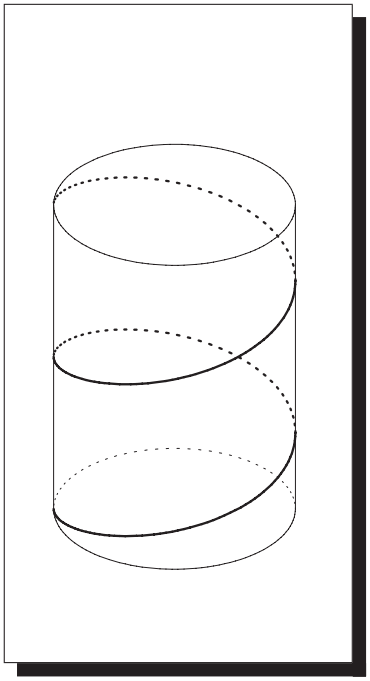
6.2.6 ¥iiiBGurafu

空間座標で媒介変数を用いて表された曲線を描画する ¥iiibGurafu に対しても，Perl と連携する ¥iiiBGurafu を用意しました。

¥iiiBGurafu

```
¥unitlength16mm¥footnotesize
¥Drawaxisfalse
¥begin{Zahyou}[(-1,0)][(0,-.5)][(0,.2)]%
(-1.2,1.2)(-1,2)(-1,18)
¥def¥Fx{cos(T)}
¥def¥Fy{sin(T)}
¥def¥Fz{T}
¥Mul4¥Pie¥tmax
{¥thicklines
¥iiiBGurafu¥Fx¥Fy¥Fz{0}{$pi}
¥iiiBGurafu¥Fx¥Fy¥Fz{2*$pi}{3*$pi}
¥iiiBGurafu(.09)(.02)¥Fx¥Fy¥Fz{$pi}{2*$pi}
¥iiiBGurafu(.09)(.02)¥Fx¥Fy¥Fz{3*$pi}{4*$pi}}
¥iiiBGurafu¥Fx¥Fy{0}{0}{$pi}
¥iiiBGurafu(.09)(.02)¥Fx¥Fy{0}{$pi}{2*$pi}
¥iiiBGurafu¥Fx¥Fy{¥tmax}{0}{2*$pi}
¥iiiDrawline{(1,0,0)(1,0,¥tmax)}
¥iiiDrawline{(-1,0,0)(-1,0,¥tmax)}
¥end{Zahyou}
```

→



6.3 波線

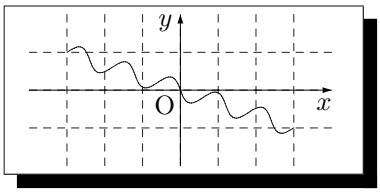
6.3.1 ¥namisen

指定した 2 点を正弦曲線で結ぶコマンドが ¥namisen です。

¥namisen

```
¥begin{zahyou}[ul=5mm](-4,4)(-2,2)
¥zahyouMemori[g][n]
¥namisen{(-3,1)}{(3,-1)}
¥end{zahyou}
```

→



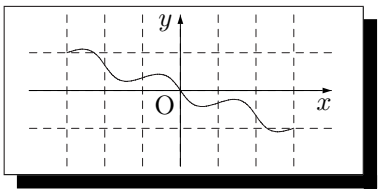
6.3.2 波の数

デフォルトでは，指定した 2 点の間を 5 周期分の正弦曲線を描きます。これを変更するには

namisuu=...

```
¥begin{zahyou}[ul=5mm](-4,4)(-2,2)
¥zahyouMemori[g][n]
¥namisen[namisuu=3]{(-3,1)}{(3,-1)}
¥end{zahyou}
```

→



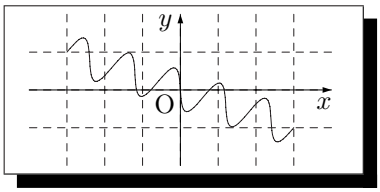
6.3.3 波の高さ

波の高さを変更するオプションは

namitakasa=..

```
¥begin{zahyou}[ul=5mm](-4,4)(-2,2)
¥zahyouMemori[g][n]
¥namisen[namitakasa=.5]{(-3,1)}{(3,-1)}
¥end{zahyou}
```

→



デフォルト値は 0.25 です。

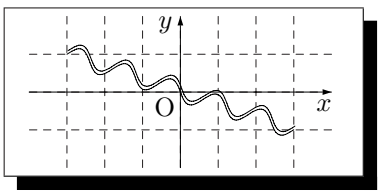
6.3.4 ¥namisen*

アスタリスク付のコマンド ¥namisen*は，2 本の平行な波線を描画し，それによって挟まれた部分を白塗りします。

¥namisen*

```
¥begin{zahyou}[ul=5mm](-4,4)(-2,2)
¥zahyouMemori[g][n]
¥namisen*{(-3,1)}{(3,-1)}
¥end{zahyou}
```

→



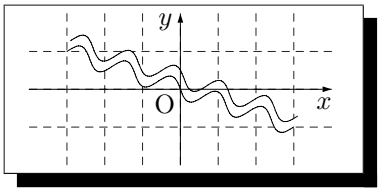
6.3.5　波線の間隔

波線の間隔はデフォルトでは 0.1 となっていますが，これを変更するオプションです。

namikankaku=..

```
¥begin{zahyou}[ul=5mm](-4,4)(-2,2)
¥zahyouMemori[g][n]
¥namisen*[namikankaku=.3]{(-3,1)}{(3,-1)}
¥end{zahyou}
```

→



6.3.6　hyou 環境・縦罫線に波線

表の縦罫線に二重の波線を描いてみましょう。

出発点

```
$
¥begin{hyou}{*4{|C{1zw}}C{1zw}|} ¥hline
    & A & B & C & 残 ¥¥¥hline
  1 & R & B & R & R ¥¥¥hline
  2 & R & R & B & R ¥¥¥hline
  3 & R & R & R & B ¥¥¥hline
¥end{hyou}
$
```

→

	A	B	C	残
1	R	B	R	R
2	R	R	B	R
3	R	R	R	B

この表で，第 4 列と第 5 列の間に波線を入れることを目的とします。

結果を先にご覧頂きましょう。

結果

```
$
¥begin{hyou}{*4{|C{1zw}}C{1zw}|} ¥hline
    & A & B & C & 残 ¥¥¥hline
  1 & R & B & R & R ¥¥¥hline
  2 & R & R & B & R ¥¥¥hline
  3 & R & R & R &
     ¥sya(S=B)[o]<%
      ¥getrowHD*{}¥rowH¥rowD
      ¥mydimena=¥rowH¥advance¥mydimena¥rowD
      ¥setlength{¥mydimenb}{%
                  4¥mydimena+3¥arrayrulewidth}%
      ¥edef¥Lw{¥mumeisuu¥mydimenb}%
      ¥Put¥LB{%
       ¥begin{zahyou*}(0,0)(0,¥Lw)%
        ¥namisen[namitakasa=1]{(-1,0)}{(-1,¥Lw)}%
        ¥namisen[namitakasa=1]{(1,0)}{(1,¥Lw)}%
       ¥end{zahyou*}%
      }%
      >
    ¥¥¥hline
¥end{hyou}
$
```

→

	A	B	C	残
1	R	B	R	R
2	R	R	B	R
3	R	R	R	B

波線は，zahyou*環境で ¥namisen を使用しましょう。そのためには，表の横幅を知らねばなりません。

表の横幅

```
¥dimenref{Whyou1}¥yokohaba
下の表の横幅は ¥yokohaba です。

¥getWHD{hyou1}{%
  $¥begin{hyou}{|c|r|r|}¥hline
    A & 0 & 1 ¥¥¥hline
    B & 2 & 3 ¥¥
    Z & 100 & 300 ¥¥¥hline
  ¥end{hyou}$%
}
```

→

下の表の横幅は 69.6869pt です。

A	0	1
B	2	3
Z	100	300

すなわち，¥getWHD と ¥dimenref の組合せで表の横幅を知ることができます。

波線を引くには，波線のすぐ上の行の左端の列に ¥sya を適用します。¥sya の<..>オプション内は picture 環境になっていますから，そこに ¥namisen を入れましょう。

横罫線を波線で

```
¥dimenref{Whyou2}¥yokohaba
¥setlength{¥mydimena}{¥yokohaba-2¥arrayrulewidth}%
¥edef¥Lw{¥mumeisuu¥mydimena}%
¥getWHD{hyou2}{%
  $¥begin{hyou}{|c|r|r|}¥hline
    A & 0 & 1 ¥¥ ¥hline
    ¥sya(S=B)[o]<%
      ¥Put¥LB{%
        ¥begin{zahyou*}(0,¥Lw)(0,0)
          ¥namisen[namitakasa=1]{(0,-1)}{(¥Lw,-1)}
        ¥end{zahyou*}
      }%
    > & 2 & 3 ¥¥[2pt]
    Z & 100 & 300 ¥¥ ¥hline
  ¥end{hyou}$%
}
```

→

A	0	1
B	2	3
Z	100	300

縦の波線も同様です。

縦罫線を波線で

```
¥dimenref{Thyou3}¥tatehaba
¥setlength{¥mydimena}{¥tatehaba-2¥arrayrulewidth}%
¥edef¥Lw{¥mumeisuu¥mydimena}%
¥getWHD{hyou3}{%
  $¥begin{hyou}{|c|*2{R{2zw}}|}¥hline
    A & 0 & 1 ¥¥ ¥hline
    B & 2 & 3 ¥¥ ¥hline
    Z & 100 &
      ¥sya(S=¥hfill 300)[o]<%
      ¥Put¥LB{%
        ¥begin{zahyou*}(0,0)(0,¥Lw)
          ¥namisen[namitakasa=2]{(2,0)}{(2,¥Lw)}
        ¥end{zahyou*}
      }%
    > ¥¥ ¥hline
  ¥end{hyou}$%
}
```

→

A	0	1
B	2	3
Z	100	300

波線を二重にしてみます。

二重波線

```
¥dimenref{Thyou4}¥tatehaba
¥setlength{¥mydimena}{¥tatehaba-2¥arrayrulewidth}%
¥edef¥Lw{¥mumeisuu¥mydimena}%
¥getWHD{hyou4}{%
  $¥begin{hyou}{|c|*2{R{2zw}}|}¥hline
    A & 0 & 1 ¥¥ ¥hline
    B & 2 & 3 ¥¥ ¥hline
    Z & 100 &
      ¥sya(S=¥hfill 300)[o]<%
      ¥Put¥LB{%
        ¥begin{zahyou*}(0,0)(0,¥Lw)
          ¥namisen*[namitakasa=2,namikankaku=2,dx=0.175]%
              {(2,0)}{(2,¥Lw)}
        ¥end{zahyou*}
      }%
    > ¥¥ ¥hline
  ¥end{hyou}$%
}
```

→

A	0	1
B	2	3
Z	100	300

波線の形状は `¥namisen` のオプションでいろいろ変えることができます。

6.4 曲線上の点

6.4.1 ¥YTen

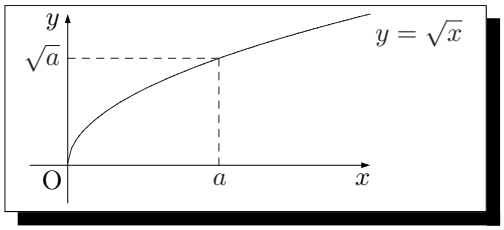
$y = f(x)$ で与えられた曲線において，x の値を与えて点 $(x,\ f(x))$ を求めるコマンドが ¥YTen です。

例えば，

¥YTen

```
¥begin{zahyou}[ul=10mm]%
(-.5,4)(-.5,2)
  ¥def¥Fx{sqrt(X)}
  ¥def¥aval{2}
  ¥YGurafu*[hidarix=0]¥Fx
  ¥Put¥migiT[se]{$y=¥sqrt{x}$}
  ¥YTen¥Fx¥aval¥A
  ¥Put¥A[syaei=xy,xlabel=a,%
    ylabel=¥sqrt{a}]{}
¥end{zahyou}
```

→



¥YTen の書式です。

y=f(x) **上の点の座標を求める。**

¥YTen[#1]#2#3#4

- #1 : **戻り値のフォーマット制御をするオプション**
  - key=val **の形式**
  - xformat=.. : x **座標のフォーマット指定（デフォルトは** ‘s’**）**
  - yformat=.. : y **座標のフォーマット指定（デフォルトは** ‘f’**）**
- #2 : **関数式**
- #3 : x **の値**
- #4 : **点の座標を受け取る制御綴**

¥YTen は，点の座標を戻しますが，そのフォーマットは

x **座標は** ‘s’**，すなわち**#3 **で与えたものが文字列としてそのまま戻ります。**
y **座標は** ‘f’**，通常の小数表示**

x の値を計算式で与えるときは，xformat=f と指定する必要があります。

xformat=.. オプション

```
¥def¥Fx{X*X}
¥YTen¥Fx{1/2}¥P
デフォルトの戻り値は
¥[ ¥P ¥]

¥YTen[xformat=f]¥Fx{1/2}¥P
¥verb+xformat=f+を指定したときの戻
り値は
¥[ ¥P ¥]
```

→

デフォルトの戻り値は

$$(1/2, 0.250000)$$

xformat=f を指定したときの戻り値は

$$(0.500000, 0.250000)$$

y 座標が整数となるときは，yformat=d オプションを与えたほうがよいでしょうか。

yformat=.. オプション

```
¥def¥Fx{X*X}
¥YTen¥Fx{3}¥P
デフォルトでは，結果が整数であっても
¥[ ¥P ¥]

¥YTen[yformat=d]¥Fx{3}¥P
¥verb+yformat=d+を与えれば
¥[ ¥P ¥]
```

→

デフォルトでは，結果が整数であっても

$$(3, 9.000000)$$

yformat=d を与えれば

$$(3, 9)$$

同様に，$x = g(y)$ で与えられた曲線において，y の値を指定して点 $(g(y), y)$ を求めるコマンド ¥XTen もあります。

6.4.2 ¥BTen

曲線が媒介変数表示

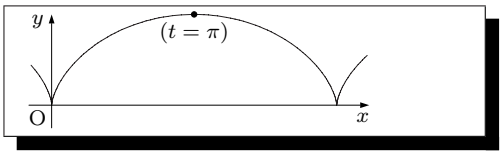
$$\begin{cases} x = f(t) \\ y = g(t) \end{cases}$$

で与えられている場合に，媒介変数 t の値を与えて曲線上の点を求めるコマンドが ¥BTen です。

¥Bten

```
¥footnotesize
¥begin{zahyou}[ul=6mm]%
  (-.5,7)(-.5,2)
  ¥def¥Ft{T-sin(T)}
  ¥def¥Gt{1-cos(T)}
  ¥BGurafu¥Ft¥Gt{-3}{9}
  ¥BTen¥Ft¥Gt{$pi}¥P
  ¥Kuromaru¥P
  ¥Put¥P[s]{($t=¥pi$)}
¥end{zahyou}
```

→



¥BTen の書式です。

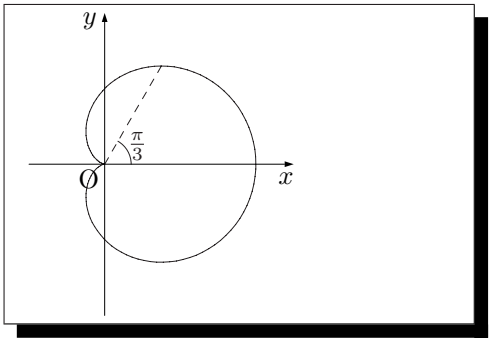
¥BTen#1#2#3#4
媒介変数曲線上の 1 点の座標を求める。
#1 : x=f(t)
#2 : y=g(t)
#3 : t **の値**
#4 : **結果を受け取る制御綴**

6.4.3 ¥RTen

極方程式 $r = f(\theta)$ で与えられる曲線上の点を求めるコマンドが ¥RTen です。

¥RTen
```
¥begin{zahyou}[ul=10mm]%
  (-1,2.5)(-2,2)
  ¥def¥Ft{1+cos(T)}
  ¥RGurafu¥Ft{0}{2*$pi}
  ¥RTen¥Ft{$pi/3}¥A
  ¥Hasen{¥O¥A}
  ¥Kakukigou¥XMAX¥O¥A{$¥frac¥pi3$}
¥end{zahyou}
```

→



6.5 $y = f(x)$ と $y = g(x)$ の交点

2 つのグラフ $y = f(x)$, $y = g(x)$ の交点を求めるコマンドが ¥YKouten です（もちろん近似値です)。その書式です。

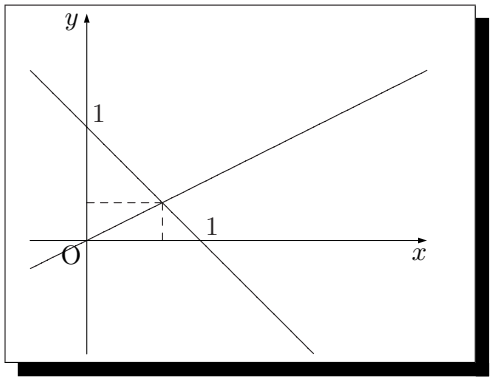
¥def¥YKouten#1#2#3#4#5#6
#1 : f(x)
#2 : g(x)
#3 : **区間左端点（省略時は** ¥xmin)
#4 : **区間右端点（省略時は** ¥xmax)
#5 : **交点の** x **座標を受け取る制御綴**
#6 : **交点の座標を受け取る制御綴**

まずは，2 直線 $y = 1 - x$, $y = \dfrac{1}{2}x$ の交点を求めてみます。

2 直線の交点

```
¥begin{zahyou}[ul=15mm](-.5,3)(-1,2)
  ¥def¥Fx{1-X}
  ¥def¥Gx{X/2}
  ¥YKouten¥Fx¥Gx{}{}¥x¥P
  ¥YGurafu*¥Fx
  ¥YGurafu*¥Gx
  ¥Put{(1,0)}[ne]{1}
  ¥Put{(0,1)}[ne]{1}
  ¥Put¥P[syaei=xy,xlabel=,ylabel=]{}
¥end{zahyou}
```

→

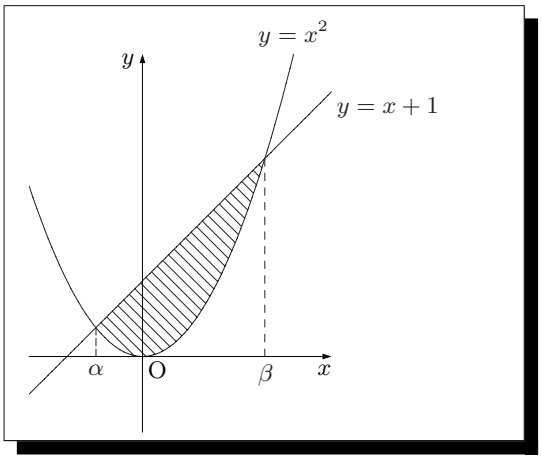


次は，放物線 $y = x^2$ と直線 $y = x + 1$ との交点です。今度は交点が 2 個ありますから，それぞれの交点を含む x の範囲を明示しておく必要があります。

放物線と直線の交点

```
¥begin{zahyou}
[%
  ul=10mm,Ueyohaku=15pt,%
  gentenhaiti={[se]},
  Migiyohaku=45pt%
](-1.5,2.5)(-1,4)
¥small
¥def¥Fx{X*X}
¥def¥Gx{X+1}
¥YKouten¥Fx¥Gx{}{1}¥tmpxi¥P
¥YKouten¥Fx¥Gx{1}{}¥tmpxii¥Q
¥Put¥P[syaei=x,xlabel=¥alpha]{}
¥Put¥Q[syaei=x,xlabel=¥beta]{}
¥YNurii*[-45]¥Fx¥Gx¥tmpxi¥tmpxii
¥YGurafu*¥Fx
¥Put¥migiT[n]{$y=x^2$}
¥YGurafu*¥Gx
¥Put¥migiT[se]{$y=x+1$}
¥end{zahyou}
```

→

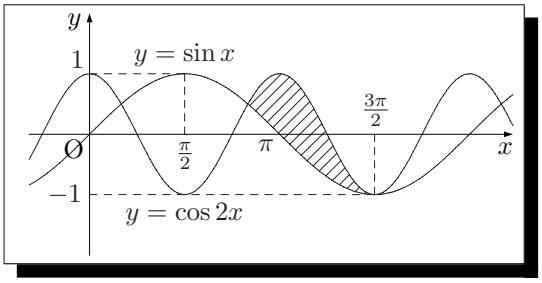


続いて，三角関数のグラフの交点を求めます。今度は接点も含まれますが，うまく求められるでしょうか。

超越曲線の交点

```
¥begin{zahyou}%
[ul=8mm](-1,7)(-2,2)
¥def¥Fx{sin(X)}
¥def¥Gx{cos(2*X)}
¥YKouten¥Fx¥Gx¥Pih¥Pie¥xi¥dmy
¥YKouten¥Fx¥Gx{¥Pie}{¥Pii}¥xii¥dmy
¥YNurii*¥Fx¥Gx¥xi¥xii
¥YGurafu*¥Fx
¥YGurafu*¥Gx
¥Put{(¥Pie,0)}[sw]{$¥pi$}
¥Put{(¥Pihiii,-1)}[syaei=xy,
  xlabel=¥frac{3¥pi}{2}]{}
¥Put{(¥Pih,1)}[houi=n,syaei=xy,
  xlabel=¥frac{¥pi}{2},ylabel=]{%
  $y=¥sin x$}
¥Put{(¥Pih,-1)}[s]{$y=¥cos 2x$}
¥Put{(0,1)}[nw]{1}
¥end{zahyou}
```

→



6.6 塗りつぶし

Perl を用いる手法は，塗りつぶしコマンドにも適用されます。

```
¥YNuri[濃さ]<x の刻み値>{関数}{x1}{x2}
¥YNurii[濃さ]<x の刻み値>{関数 1}{関数 2}{x1}{x2}
¥YNuri*[斜線方向角]<斜線間隔>(x の刻み値){関数}{x1}{x2}
¥YNurii*[斜線方向角]<斜線間隔>(x の刻み値){関数 1}{関数 2}{x1}{x2}
```

いずれも ¥yNuri... と同様です。詳しくは，sampleP.tex をご覧ください。

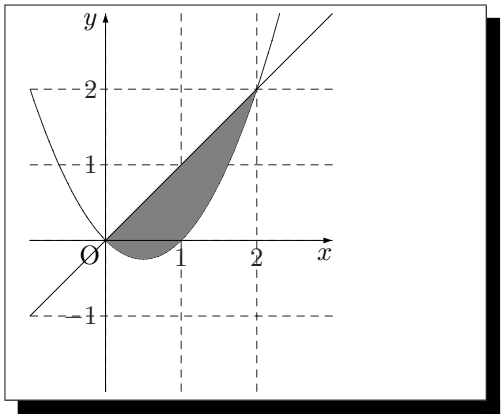
6.6.1 ¥YNuri

まずは $y = f(x)$ と，その上の 2 点を結ぶ弦とで囲まれる図形の塗りつぶしです。

¥YNuri

```
¥begin{zahyou}[ul=10mm](-1,3)(-2,3)
  ¥zahyouMemori[g]
  ¥def¥Fx{X*(X-1)}
  ¥YGurafu*¥Fx
  ¥YNuri¥Fx{0}{2}
  ¥YGurafu*{X}
¥end{zahyou}
```

→



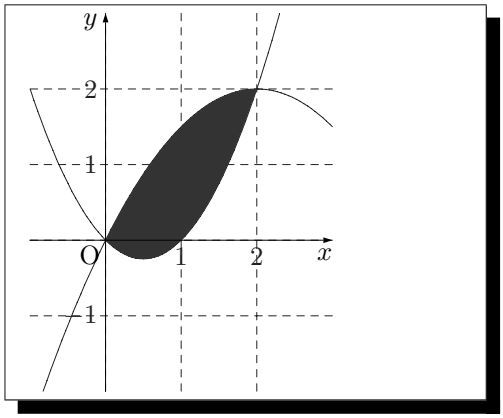
6.6.2 ¥YNurii

続いて，二つの曲線 $y = f(x)$ と $y = g(x)$ で囲まれる領域です。

¥YNurii

```
¥begin{zahyou}[ul=10mm](-1,3)(-2,3)
  ¥zahyouMemori[g]
  ¥def¥Fx{X*(X-1)}
  ¥def¥Gx{2-.5*(X-2)**2}
  ¥YGurafu*¥Fx
  ¥YGurafu*¥Gx
  ¥YNurii[.8]¥Fx¥Gx{0}{2}
¥end{zahyou}
```

→



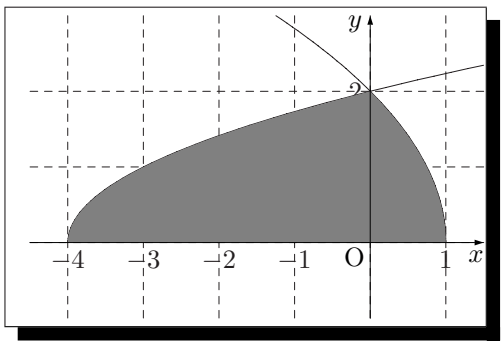
6.6.3 ¥XNuri, ¥XNurii

¥XNuri, ¥XNurii もあります。

¥XNurii

```
¥begin{zahyou}[ul=10mm](-4.5,1.5)(-1,3)
  ¥zahyouMemori[g]
  ¥def¥Fy{Y*Y-4}
  ¥def¥Gy{1-Y*Y/4}
  ¥XGurafu*[sitay=0]¥Fy
  ¥XGurafu*[sitay=0]¥Gy
  ¥XNurii¥Fy¥Gy{0}{2}
¥end{zahyou}
```

→



6.6.4 ¥BNuri

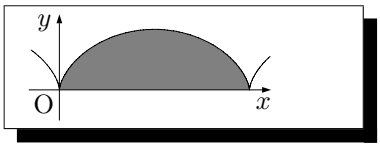
¥BNuri[**濃さ**]<t **の刻み値**>{f(t)}{g(t)}{t1}{t2}

は，媒介変数 $x = f(t)$, $y = g(t)$ で表された曲線の $t_1 \leqq t \leqq t_2$ の部分と両端を結ぶ弦で囲まれた領域を塗りつぶします。

¥BNuri

```
¥def¥Ft{T-sin(T)}
¥def¥Gt{1-cos(T)}
¥begin{zahyou}[ul=4mm](-1,7)(-1,2.5)
  ¥BNuri¥Ft¥Gt{0}{2*$pi}
  ¥BGurafu¥Ft¥Gt{-$pi}{3*$pi}
¥end{zahyou}
```

→

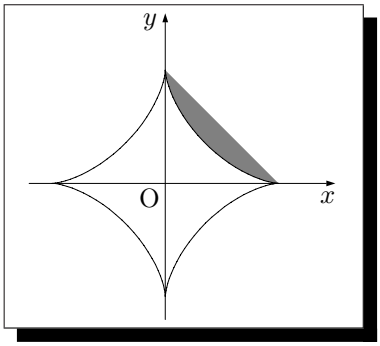


はよいとして

¥BNuri

```
¥def¥Ft{cos(T)**3}
¥def¥Gt{sin(T)**3}
¥begin{zahyou}[ul=15mm](-1.2,1.5)(-1.2,1.5)
  ¥BNuri¥Ft¥Gt{0}{$pi/2}
  ¥BGurafu¥Ft¥Gt{0}{2*$pi}
¥end{zahyou}
```

→



は的外れでしょうね。¥BNuri は，曲線の弧とその両端を結ぶ弦で囲まれた図形を塗りつぶしますから，上の図は指令どおりになっているのですが.....

曲線内部の第 1 象限を塗りつぶしたいとすれば，三角形を塗りつぶして，上図の部分を白塗りするのもありますが，曲線の弧を折れ線近似して多角形塗りつぶしコマンド ¥Nuritubusi を用いるのもあります。

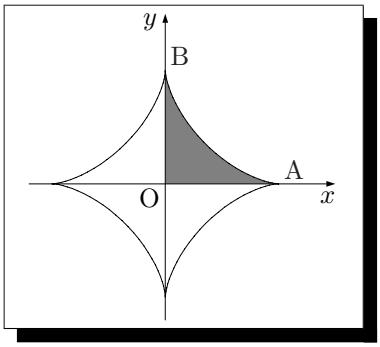
¥BKinziOresen{f(t)}{g(t)}{t1}{t_2}{t **の刻み値**}{**折れ線を受け取る制御綴**}

は，媒介変数表示 $x = f(t)$, $y = g(t)$, $(t_1 \leqq t \leqq t_2)$ された曲線の弧を折れ線で近似します。

¥BKinziOresen

```
¥def¥Ft{cos(T)**3}
¥def¥Gt{sin(T)**3}
¥begin{zahyou}[ul=15mm](-1.2,1.5)(-1.2,1.5)
  ¥tenretu{A(1,0)ne;B(0,1)ne}
  ¥BKinziOresen¥Ft¥Gt{0}{$pi/2}{0.05}¥oresen
  ¥Nuritubusi{¥oresen¥O}
  ¥BGurafu¥Ft¥Gt{0}{2*$pi}
¥end{zahyou}
```

→



曲線の弧 AB を近似した折れ線が ¥oresen に得られます。それに点 ¥O を付加して，¥Nuritubusi コマンドに与えます。

（注 1） 点 ¥O を付加しただけでは，折れ線は (A->B->O) となって，閉じていません。点 ¥A をさらに付加すべきですが，¥Nuritubusi は，始点と終点を強制的に結んで閉じさせた上で塗りつぶす仕様となっていますから，点 ¥A を付加する手間は省略することができます。

（注 2） 曲線 $y = f(x)$ を近似する ¥YKinziOresen も用意はしてあります。

6.6.5 ¥RNuri

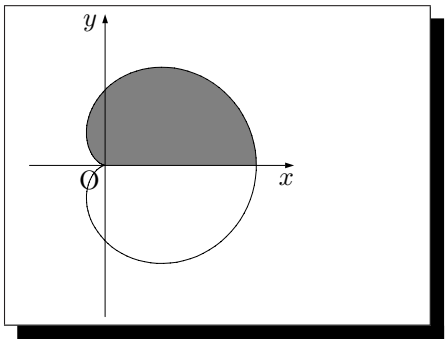
極方程式 $r = f(\theta)$ で与えられた曲線の弧と弦を結ぶ図形を塗りつぶす ¥RNuri もあります。近似折れ線を得る ¥RKinziOresen も使用可能です。

```
¥RNuri[濃さ]<θの刻み値>{f(θ)}{t1}{t2}
¥RKinziOresen{f(θ)}{t1}{t_2}{θの刻み値}{折れ線を受け取る制御綴}
```

¥RNuri

```
¥begin{zahyou}[ul=10mm](-1,2.5)(-2,2)
  ¥def¥Ft{1+cos(T)}
  ¥RNuri¥Ft{0}{$pi}
  ¥RGurafu¥Ft{0}{2*$pi}
¥end{zahyou}
```

→



6.7 斜線塗り

Perl を用いる手法は，斜線塗りコマンドにも適用されます。

¥YNuri*[**斜線方向角**]<**斜線間隔**>(x **の刻み値**){**関数**}{x1}{x2}
¥YNurii*[**斜線方向角**]<**斜線間隔**>(x **の刻み値**){**関数** 1}{**関数** 2}{x1}{x2}

いずれも ¥yNuri*... と同様です。詳しくは，sampleP.tex をご覧ください。

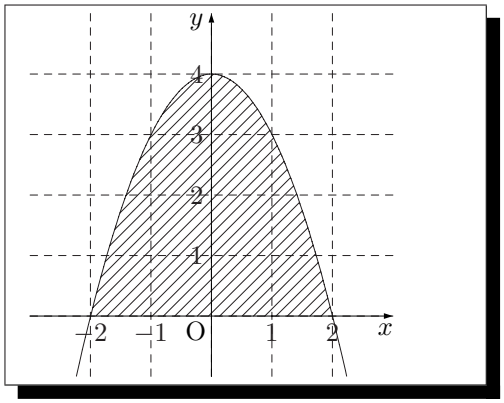
6.7.1 ¥YNuri*

¥YNuri*を用いた例です。

¥YNuri*

```
¥begin{zahyou}[ul=8mm](-3,3)(-1,5)
  ¥zahyouMemori[g]
  ¥def¥Fx{4-X*X}
  ¥YNuri*¥Fx{-2}{2}
  ¥YGurafu*¥Fx
¥end{zahyou}
```

→



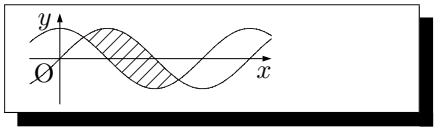
6.7.2 ¥YNurii*

¥YNurii* を用いた例です。

¥YNurii*

```
¥begin{zahyou}[ul=4mm](-1,7)(-1.5,1.5)
¥def¥Fx{sin(X)}
¥def¥Gx{cos(X)}
¥YNurii*<.3>¥Fx¥Gx{¥Piq}{¥Piqv}
¥YGurafu*¥Fx
¥YGurafu*¥Gx
¥end{zahyou}
```

→



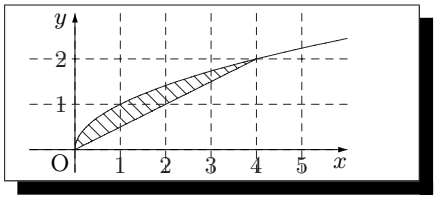
6.7.3 ¥XNuri*

¥XNuri* もあります。

¥XNuri*

```
¥begin{zahyou}[ul=6mm](-1,6)(-.5,3)
¥small¥zahyouMemori[g]
¥def¥Fy{Y*Y}
¥XNuri*[-45]<.2>¥Fy{0}{2}
¥XGurafu*[sitay=0]¥Fy
¥YGurafu{.5*X}{0}{4}
¥end{zahyou}
```

→



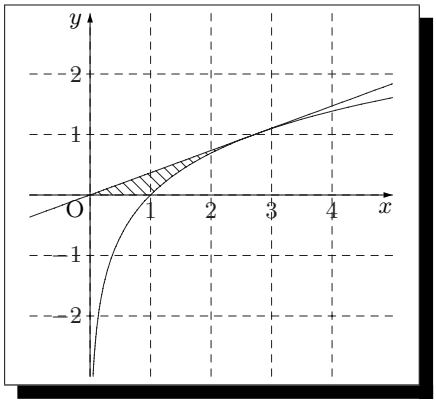
6.7.4 ¥XNurii*

最後に ¥XNurii*で，$y = \log x$ とその接線 $y = ex$ および x 軸で囲まれた領域の斜線塗りです。

¥XNurii*

```
¥begin{zahyou}[ul=8mm](-1,5)(-3,3)
¥small¥zahyouMemori[g]
¥def¥Fy{exp(Y)}
¥def¥Gy{¥Napier*Y}
¥XNurii*[-45]¥Fy¥Gy{0}{1}
¥XGurafu*¥Fy
¥XGurafu*¥Gy
¥end{zahyou}
```

→



6.7.5 ¥BNuri*

媒介変数表示された曲線に対する斜線塗りです。

¥BNuri*[**斜線方向角**]<**斜線間隔**>(t **の刻み値**){f(t)}{g(t)}{t1}{t2}

媒介変数表示

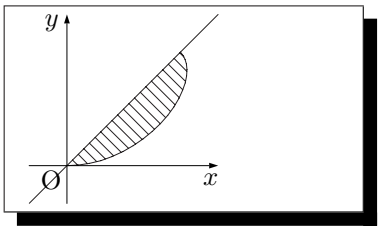
$$x(t) = \frac{3t}{1+t^3},\ y(t) = \frac{3t^2}{1+t^3} \quad (0 \leqq t \leqq 1)$$

で表される曲線 C と直線 $y = x$ で囲まれた領域を斜線塗りします。

¥BNuri*

```
¥begin{zahyou}[ul=10mm](-.5,2)(-.5,2)
  ¥def¥Xt{3*T/(1+T**3)}
  ¥def¥Yt{3*T*T/(1+T**3)}
  ¥BNuri*[-45]¥Xt¥Yt{0}{1}
  ¥BGurafu¥Xt¥Yt{0}{1}
  ¥Drawline{(¥xmin,¥xmin)(¥xmax,¥xmax)}
¥end{zahyou}
```

→

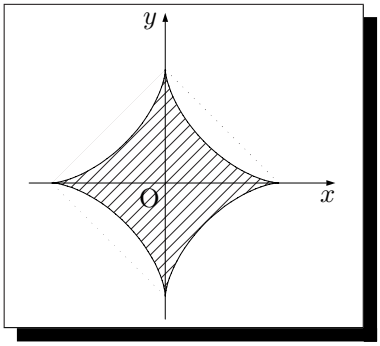


斜線塗りは厄介で，図形が凸でないなど，複雑な場合は工夫が必要です。次の例は，凸でない図形に対し，凸閉包を斜線塗りして，余分なところを白塗りしています。

凸閉包を利用

```
¥begin{zahyou}[ul=15mm](-1.2,1.5)(-1.2,1.5)
  ¥def¥Ft{cos(X)**3}
  ¥def¥Gt{sin(X)**3}
  ¥tenretu*{A(1,0);B(0,1);C(-1,0);D(0,-1)}
  ¥Nuritubusi*{¥A¥B¥C¥D}
  ¥BNuri[0]¥Ft¥Gt{0}{$pi/2}
  ¥BNuri[0]¥Ft¥Gt{$pi/2}{$pi}
  ¥BNuri[0]¥Ft¥Gt{$pi}{1.5*$pi}
  ¥BNuri[0]¥Ft¥Gt{1.5*$pi}{2*$pi}
  ¥BGurafu¥Ft¥Gt{0}{2*$pi}
¥end{zahyou}
```

→

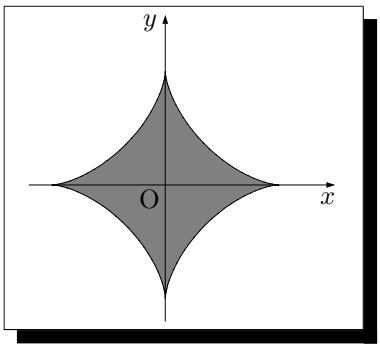


べた塗りなら簡単で

べた塗り

```
¥begin{zahyou}[ul=15mm](-1.2,1.5)(-1.2,1.5)
  ¥def¥Ft{cos(X)**3}
  ¥def¥Gt{sin(X)**3}
  ¥BNuri¥Ft¥Gt{0}{2*$pi}
  ¥BGurafu¥Ft¥Gt{0}{2*$pi}
¥end{zahyou}
```

→



でおしまい。

6.7.6 ¥RNuri*

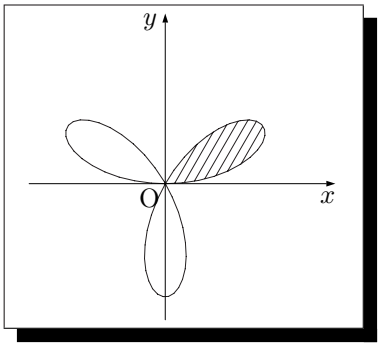
極方程式で与えられた曲線に対する斜線塗りです。

¥RNuri*[**斜線方向角**]<**斜線間隔**>(**θの刻み値**){f(t)}{t1}{t2}
¥RKinziOresen{f(**θ**)}{t1}{t_2}{**θの刻み値**}{**折れ線を受け取る制御綴**}

¥RNuri*

```
¥begin{zahyou}[ul=15mm](-1.2,1.5)(-1.2,1.5)
  ¥def¥Ft{sin(3*T)}
  ¥RNuri*[60]¥Ft{0}{$pi/3}
  ¥RGurafu¥Ft{0}{$pi}
¥end{zahyou}
```

→

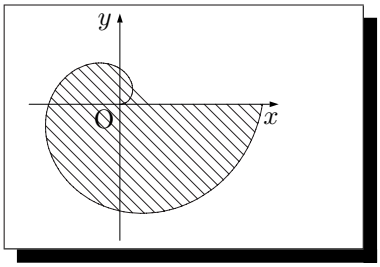


は簡単ですが，図が複雑になると工夫が必要だということは媒介変数のところでも述べました。

¥RNuri*

```
¥begin{zahyou}[ul=6mm](-2,3.5)(-3,2)
  ¥def¥Ft{.5*T}
  ¥RNuri*[-45]¥Ft{0}{2*$pi}
  ¥RGurafu¥Ft{0}{2*$pi}
¥end{zahyou}
```

→



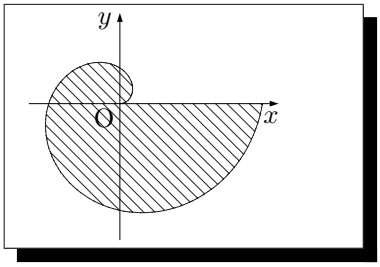
図が凸でないため，原点の右上のところで斜線がはみだしています。斜線の方向角を調整するのもひとつの解決法ですが，あえて別の方法を紹介します。

θ の範囲を分けて，始線の上下に分けて描画してみます。

分割

```
¥begin{zahyou}[ul=6mm](-2,3.5)(-3,2)
  ¥def¥Ft{.5*T}
  ¥RNuri*[-45]¥Ft{0}{$pi}
  ¥RNuri*[-45]¥Ft{$pi}{2*$pi}
  ¥RGurafu¥Ft{0}{2*$pi}
¥end{zahyou}
```

→

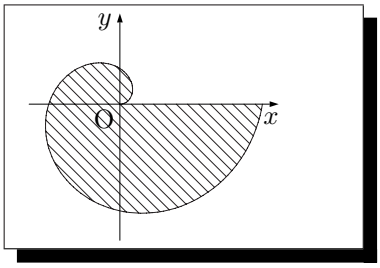


はみ出しはなくなりましたが，分割した上下の斜線がつながっていません。斜線の通過する 1 点を指定しましょう。

通過点指定

```
¥begin{zahyou}[ul=6mm](-2,3.5)(-3,2)
  ¥def¥Ft{.5*T}
  ¥RNuri*[-45]<syanurisiteiten=¥O>¥Ft{0}{$pi}
  ¥RNuri*[-45]<syanurisiteiten=¥O>¥Ft{$pi}{2*$pi}
  ¥RGurafu¥Ft{0}{2*$pi}
¥end{zahyou}
```

→

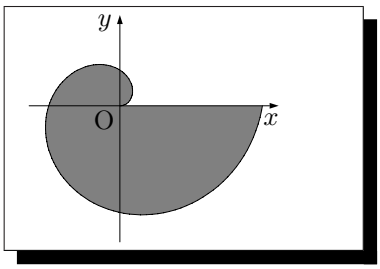


これもべた塗りならどうということはありません。

べた塗り

```
¥begin{zahyou}[ul=6mm](-2,3.5)(-3,2)
  ¥def¥Ft{.5*T}
  ¥RNuri¥Ft{0}{2*$pi}
  ¥RGurafu¥Ft{0}{2*$pi}
¥end{zahyou}
```

→



6.8 開領域の斜線塗り

領域を斜線塗りする際，境界が含まれないことを図示するために，斜線と境界の間を切るという表現法があります。

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）**

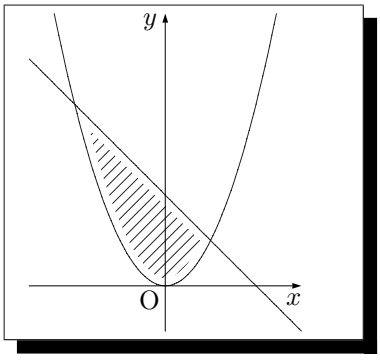
オプションにより白塗りの幅を指定することにしてみました。境界線を中心として，その左右に指定した幅を有する白い曲線を描画します。

（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](-3,3)(-1,6)
  ¥def¥Fx{X*X}
  ¥def¥Gx{2-X}
  ¥YNurii*<syasentanmatu=1mm>¥Fx¥Gx{-2}{1}
  ¥YGurafu*¥Fx
  ¥YGurafu*¥Gx
¥end{zahyou}
```

→

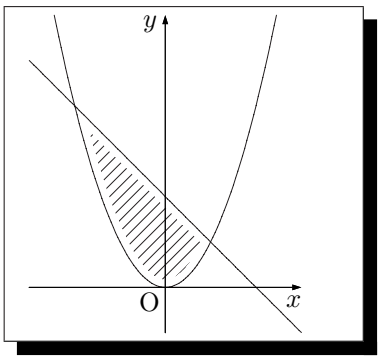


ただし境界線を白塗りする結果，座標軸などが消え，そのお化粧直しが必要となる場合もあります。

お化粧直し

```
¥begin{zahyou}[ul=6mm](-3,3)(-1,6)
  ¥def¥Fx{X*X}
  ¥def¥Gx{2-X}
  ¥YNurii*<syasentanmatu=1mm>¥Fx¥Gx{-2}{1}
  ¥YGurafu*¥Fx
  ¥YGurafu*¥Gx
  ¥drawXaxis¥drawYaxis% お化粧直し
¥end{zahyou}
```

→



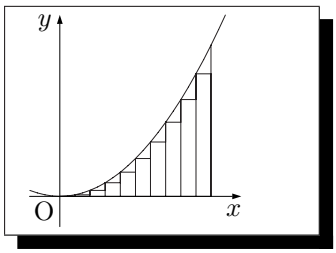
6.9 区分求積の図

6.9.1 短冊のみ

¥kubunkyuusekizu

```
¥begin{zahyou}[ul=20mm](-.2,1.2)(-.2,1.2)
  ¥def¥Fx{X*X}
  ¥YGurafu*¥Fx
  ¥kubunkyuusekizu¥Fx{0}{1}{10}{l}
¥end{zahyou}
```

→



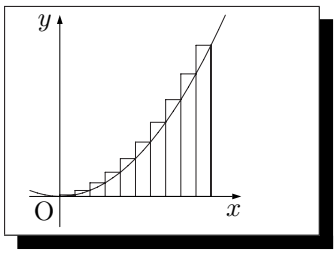
は，関数 ¥Fx の区間 $0 \leqq x \leqq 1$ を 10 等分して，区間の左端を高さとする短冊を寄せ集めた図を作ります。

区間の右端を高さとするときは，最後の引数を{r}とします。

区間の右端を高さとする

```
¥begin{zahyou}[ul=20mm](-.2,1.2)(-.2,1.2)
  ¥def¥Fx{X*X}
  ¥YGurafu*¥Fx
  ¥kubunkyuusekizu¥Fx{0}{1}{10}{r}
¥end{zahyou}
```

→



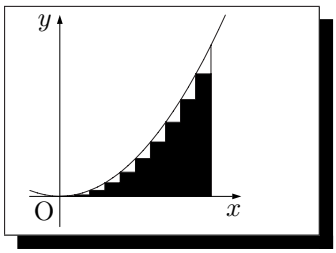
6.9.2 短冊内をベタ塗り

短冊内をベタ塗りするには，*付のコマンドを用います。

¥kubunkyuusekizu*

```
¥begin{zahyou}[ul=20mm](-.2,1.2)(-.2,1.2)
  ¥def¥Fx{X*X}
  ¥YGurafu*¥Fx
  ¥kubunkyuusekizu*¥Fx{0}{1}{10}{l}
¥end{zahyou}
```

→



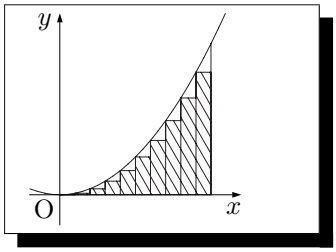
6.9.3　短冊内を斜線塗り

短冊内を斜線塗りするには，**付のコマンドを用います。

¥kubunkyuusekizu**

```
¥begin{zahyou}[ul=20mm](-.2,1.2)(-.2,1.2)
  ¥def¥Fx{X*X}
  ¥YGurafu*¥Fx
  ¥kubunkyuusekizu**¥Fx{0}{1}{10}{l}
¥end{zahyou}
```

→

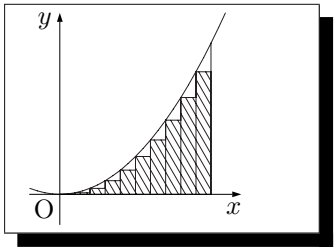


ベタ塗りの濃さ，斜線の方向角・間隔などを変更する方法は ¥Nuritubusi コマンドに準じます。斜線間隔を細かくしてみます。

斜線間隔の調整

```
¥begin{zahyou}[ul=20mm](-.2,1.2)(-.2,1.2)
  ¥def¥Fx{X*X}
  ¥YGurafu*¥Fx
  ¥kubunkyuusekizu**<.04>¥Fx{0}{1}{10}{l}
¥end{zahyou}
```

→



6.9.4　書式

¥kubunkyuusekizu の書式です。

```
¥kubunkyuusekizu[#1]<#2>#3#4#5#6#7%      短冊のみ
¥kubunkyuusekizu*[#1]<#2>#3#4#5#6#7%     ベタ塗り
¥kubunkyuusekizu**[#1]<#2>#3#4#5#6#7%    斜線塗り
      #1 : 塗りの濃さ／斜線の方向角
      #2 : 斜線塗りの場合の斜線間隔
      #3 : 関数式
      #4 : 区間の左端
      #5 :       右端
      #6 : 分割数
      #7 : l = 区間の左端を高さ
           r =       右端
```

6.9.5　実例 (1)

区間 $0 \leqq x \leqq 1$ を細分するときだけではない，ということで，無限級数

$$\frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\cdots+\frac{1}{n^2}+\cdots$$

が収束することを示す説明図をご覧頂きましょう。

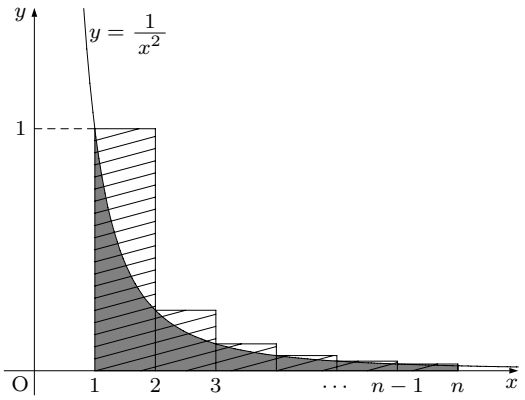
> 次の不等式が成り立つことを証明せよ。ただし，n は 2 以上の自然数とする。
>
> $$1-\frac{1}{n}+\frac{1}{n^2}<\frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\cdots+\frac{1}{n^2}<2-\frac{1}{n}$$

右上図において，曲線 $y=\dfrac{1}{x^2}\quad(1 \leqq x \leqq n)$ と x 軸との間の面積と，斜線部の面積を比較して

$$\int_1^{n+1}\frac{dx}{x^2} < \frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\cdots+\frac{1}{(n-1)^2}$$

定積分を計算し，両辺に $\dfrac{1}{n^2}$ を加えると

$$1-\frac{1}{n}+\frac{1}{n^2} < \frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\cdots+\frac{1}{n^2} \quad \cdots\cdots ①$$

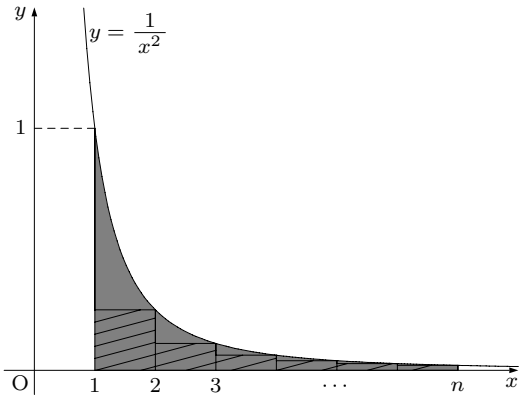


次に，右下図においては

$$\int_1^{n}\frac{dx}{x^2} > \frac{1}{2^2}+\frac{1}{3^2}+\cdots+\frac{1}{n^2}$$

両辺に 1 を加えて

$$2-\frac{1}{n} > \frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\cdots+\frac{1}{n^2} \quad \cdots\cdots ②$$



①, ②より

$$1-\frac{1}{n}+\frac{1}{n^2}<\frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\cdots+\frac{1}{n^2}<2-\frac{1}{n}$$

（証明終り）

6.9.6　実例 (2)

先の ¥namisen*と併用した例です。

球面を輪切りにし，それぞれの部分を円錐の側面の一部分で近似することによって，球の表面積を求めることを考える。

(1) 半径 $r(>0)$ の半円 $S : y = \sqrt{r^2 - x^2}$ と直線 $L_s : y = s$ $(0 < s < r)$ は 2 つの交点をもつ。それぞれの交点における半円 S の 2 本の接線は点 $(0,$ **（ア）**$)$ で交わる。この 2 本の接線と直線 L_s で囲まれた三角形を y 軸のまわりに 1 回転してできる円錐の側面積は **（イ）** である。さらに，この円錐から平面 $y = t$ $(0 < s < t \leqq r)$ より上にある部分を取り除いた立体図形の側面積は $\pi r(t - s) \times$ **（ウ）** である。

(2) n を自然数とし，$k = 1, 2, \cdots, n-1$ とする。(1) で定義された立体図形で，$s = \dfrac{k}{n}r$, $t = \dfrac{k+1}{n}r$ のときの側面積を A_k と表すと，

$$A_k = \frac{2\pi r^2}{n} - \frac{\pi r^2}{n^2} \times \boxed{\text{（エ）}}$$

となる。

(3) $\displaystyle\lim_{n\to\infty} \sum_{k=1}^{n-1} A_k = 2\pi r^2$ を証明し，それを **（オ）** に書きなさい。

2002 慶應義塾大学

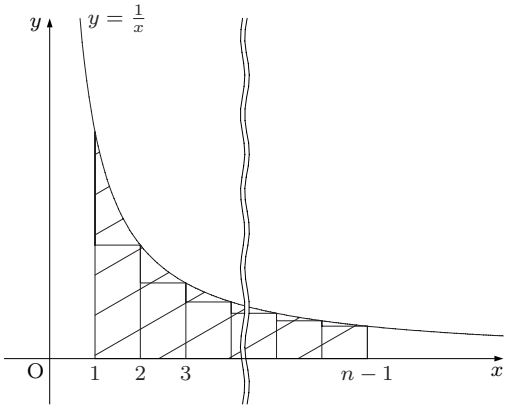
【解答】

（ア）	（イ）	（ウ）	（エ）
$\dfrac{r^2}{S}$	$\dfrac{\pi r(r^2 - s^2)}{s}$	$\dfrac{2r^2 - s^2 - st}{r^2 - s^2}$	$\dfrac{nk}{n^2 - k^2}$

（オ） の部分：

$$\begin{aligned}\lim_{n\to\infty} \sum_{k=1}^{n-1} A_k &= \lim_{n\to\infty} \left(\frac{2\pi(n-1)r^2}{n} - \frac{\pi r^2}{n^2} \times \sum_{k=1}^{n-1} \frac{nk}{n^2 - k^2} \right) \\ &= \lim_{n\to\infty} \left\{ 2\pi \left(1 - \frac{1}{n}\right) r^2 - \frac{\pi r^2}{n} \times \sum_{k=1}^{n-1} \frac{k}{n^2 - k^2} \right\} \qquad \cdots\cdots ①\end{aligned}$$

において



$$\begin{aligned}\sum_{k=1}^{n-1}\frac{k}{n^2-k^2} &= \sum_{k=1}^{n-1}\frac{1}{2}\left(\frac{1}{n-k}-\frac{1}{n+k}\right)\\ &< \frac{1}{2}\sum_{k=1}^{n-1}\frac{1}{n-k}\\ &= \frac{1}{2}\left(1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{n-1}\right)\\ &< \frac{1}{2}\left(1+\int_1^{n-1}\frac{dx}{x}\right)\\ &= \frac{1}{2}\left(1+\Big[\log x\Big]_1^{n-1}\right)\\ &= \frac{1}{2}(1+\log(n-1))\end{aligned}$$

従って

$$0<\frac{1}{n}\sum_{k=1}^{n-1}\frac{k}{n^2-k^2}<\frac{1}{2}\left(\frac{1}{n}+\frac{\log(n-1)}{n}\right)$$

$\lim_{n\to\infty}\dfrac{\log(n-1)}{n}=0$ であるから

$$\lim_{n\to\infty}\frac{1}{n}\sum_{k=1}^{n-1}\frac{k}{n^2-k^2}=0$$

ゆえに①より

$$\lim_{n\to\infty}\sum_{k=1}^{n-1}A_k=2\pi r^2$$

6.10 ¥checkPerl

¥checkPerl は，Perl との連携機能を確認するコマンドです。従来，これがエラーとなるのは，次の 2 つの場合がありました。

1. Perl **との連携ができていない**
2. Perl **との連携はできるていが，**-shell-escape **オプションをつけずに起動している。**

角藤先生が配布されている ptex に，¥write18 が enable か否かを確認するコマンド ¥ifeof18 が用意されましたので，それを利用して，上記 2 種類のエラーを分離することを可能としたのがアスタリスクつきコマンド ¥checkPerl*です。

このコマンドがうまく機能しない場合は，次の 3 つのエラーのどれかが発生します。

(1) Bad number
これは，ptex が古くて新機能 ¥ifeof18 が定義されていません。対応策は，ptex を更新することです。あるいは，¥checkPerl*をあきらめて，従来どおり ¥checkPerl を用います。

(2) Please enable shell escape by --shell option
文字通り，platex の起動において，-shell-escape オプションを付加するのを忘れたエラーです。

(3) Perl **との連携ができていません。**

これは，Perl との連携機能が働いていないことを示しています。次の 3 点をチェックします。

(i) Perl はインストールされているか。

(ii) Perl に PATH が通っているか。

(iii) emath.pl などが所定の場所に存在するか。

6.11 Perl library の追加

perl を利用して，計算結果を受け取る方法において，基本的な関数は用意されていますが，さらに自分で作成したものを追加したい，というご要望がありましたので，¥useperllib コマンドを用意しました。これは，perl のライブラリを作成して，それを利用することになります。以下，2 つの例を紹介します。

6.11.1 degsine.pl

perl の三角関数 sin, cos, tan では，引数はラジアンです。六十分法の数値を引数とする三角関数を用いたいときもあります。そのために用意したのが，perl のライブラリ degsine.pl です。その内容は，次のようになっています。

```
sub degsin{my $x=shift;return sin($pi*$x/180);}
sub degcos{my $x=shift;return cos($pi*$x/180);}
sub degtan{my $x=shift;return tan($pi*$x/180);}
1;
```

やっていることは単純で，六十分法の角をラジアンに変換して，sin, cos, tan に引き渡しているだけです。

簡単な例をご覧頂きましょう。

¥degsine

```
¥calcval{degsin(30)}¥stmp
$¥sin30¥Deg=¥stmp$
```

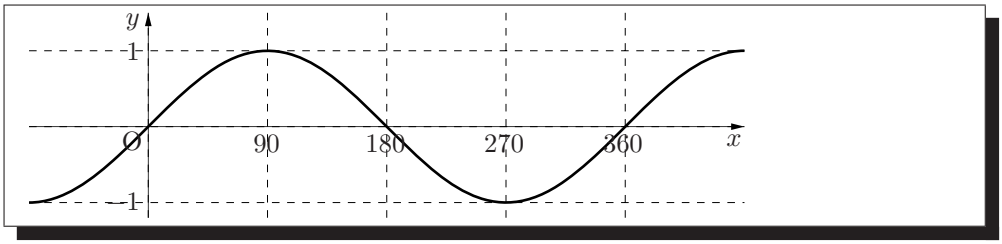
→ $\sin 30^\circ = 0.500000$

$y = \sin x^\circ$ のグラフを描いてみましょう。

$y = \sin x^\circ$ のグラフ

```
¥begin{pszahyou}[ul=10mm,xscale=0.017453294](-90,450)(-1.2,1.5)
  ¥zahyouMemori[g]<dx=90>
  ¥YGurafu*(1){degsin(X)}
¥end{pszahyou}
```

→



6.11.2 nCr.pl

第 2 の例は，$n!$, ${}_nP_r$, ${}_nC_r$ の値を計算するライブラリで，その内容は次のようになっています。

```
sub kaizyou{my $x=shift;
    if ($x<2) {return 1;}
    else {return $x*kaizyou($x-1);}}
sub nPr{my $n=shift; my $r=shift;
    if ($r<1) {return 1;}
    else {return $n*nPr($n-1,$r-1);}}
sub nCr{my $n=shift; my $r=shift;
    if ($r<1) {return 1;}
    else {return $n*nCr($n-1,$r-1)/$r;}}
1;
```

使用例です。

階乗

```
¥calcval[d]{kaizyou(10)}¥kotae
$10!=¥kotae$
```

→ $10! = 3628800$

順列

```
¥calcval[d]{nPr(10,3)}¥ans
$¥zyunretu{10}{3}=¥ans$
```

→ ${}_{10}P_3 = 720$

組合せ

```
¥calcval[d]{nCr(10,3)}¥tmp
$¥kumiawase{10}{3}=¥tmp$
```

→ ${}_{10}C_3 = 120$

注 1. これらの関数を使用するには，プリアンブルに

```
¥useperllib{nCr}
```

と宣言をしておく必要があります。

先の degsine.pl も併用したければ

```
¥useperllib{degsine}
¥useperllib{nCr}
```

と併記することになります。

注 2. degsine.pl, nCr.pl などは，emath.pl と同じディレクトリに置かなければなりません。

6.11.3 ¥useperlpm

perl のモジュールを利用するためのコマンド ¥useperlpm を新設しました。例えば，perl で標準配布されているモジュールの一つ POSIX.pm を使用したいときは，プリアンブルに

```
¥useperlpm{POSIX}
```

を宣言します。このモジュールを用いると

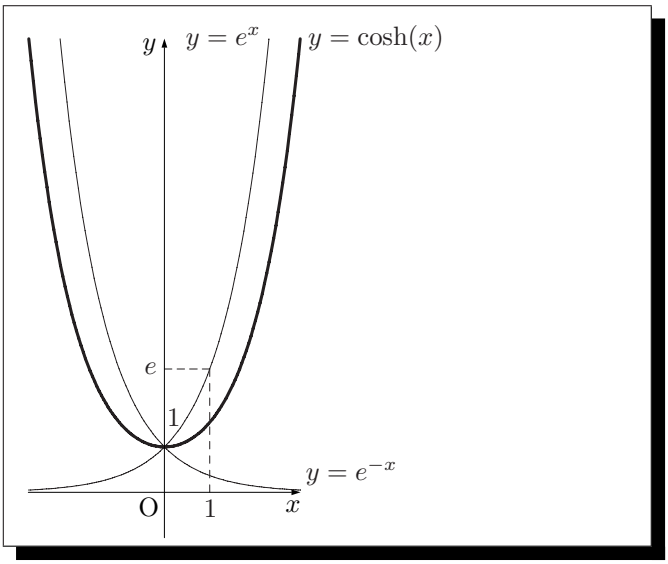
```
acos asin atan ceil cosh fabs floor fmod
frexp ldexp log10 modf pow sinh tan tanh
```

などを使用することが可能となります。cosh を使用した一例です。

$y = \cosh(x)$ のグラフ

```
¥begin{zahyou}[ul=6mm,Ueyohaku=1zh](-3,3)(-1,10)
  ¥def¥E{(1,2.71828)}¥Put¥E[syaei=xy,ylabel=e]{}
  ¥Put{(0,1)}(1pt,8pt)[lb]{1}
  ¥YGurafu*{exp(X)}¥Put¥migiT[w]{$y=e^x$}
  ¥YGurafu*{exp(-X)}¥Put¥migiT[ne]{$y=e^{-x}$}
  {¥Thicklines¥YGurafu*{cosh(X)}¥Put¥migiT[e]{$y=¥cosh(x)$}}
¥end{zahyou}
```

→



6.11.4 emath.pl v0.04

emathPp.sty で使用している Perl 用のライブラリ emath.pl にいくつかの関数を追加しました。

`Int` Perl の基本関数に `int` というのがありますが，これは BASIC の `INT` とは異なります。

int と INT の比較

```
正の数に対しては
¥calcval{int($pi)}¥y
¥[ ¥mathrm{int}(¥pi)=¥y ¥]
負の数に対しては
¥calcval{int(-$pi)}¥y
¥[ ¥mathrm{int}(-¥pi)=¥y ¥]
```

→

正の数に対しては

$$\mathrm{int}(\pi) = 3.000000$$

負の数に対しては

$$\mathrm{int}(-\pi) = -3.000000$$

そこで,「x を超えない最大の整数」を得るための Perl の関数 `Int` を新設しました。

Int

```
正の数に対しては
¥calcval{Int($pi)}¥y
¥[ ¥mathrm{Int}(¥pi)=¥y ¥]
負の数に対しては
¥calcval{Int(-$pi)}¥y
¥[ ¥mathrm{Int}(-¥pi)=¥y ¥]
```

→

正の数に対しては

$$\mathrm{Int}(\pi) = 3.000000$$

負の数に対しては

$$\mathrm{Int}(-\pi) = -4.000000$$

`Degsin`, `Degcos`, `Degtan` perl の三角関数は，単位がラジアンです。六十分法を単位とする三角関数を追加しました。

Degsin など

```
¥calcval{Degsin(60)}¥y
¥[ ¥sin60¥Deg=¥y ¥]
¥calcval{Degcos(60)}¥y
¥[ ¥cos60¥Deg=¥y ¥]
¥calcval{Degtan(60)}¥y
¥[ ¥tan60¥Deg=¥y ¥]
```

→

$$\sin 60^\circ = 0.866025$$

$$\cos 60^\circ = 0.500000$$

$$\tan 60^\circ = 1.732051$$

7 perl の参考文献

perl になじみのない方には

R.L.Schwartz E.Olson T.Phoenix **共著の邦訳**
初めての Perl **第** 3 **版**
発行元 オライリー・ジャパン
定価 3,600**+税**

ISBN:4-87311-126-9
発行年月：2003.5

をお勧めします。