

qbGraphの使い方

Satie "4eyes" Moonlight.

まえがき

`\qbGraph` は \LaTeX で高校の数学で用いる図形やグラフを簡単に描いて利用する為に作ったスタイルファイル, 或いはマクロ (コマンド) 定義群の総称です。

\LaTeX や \TeX については他の解説や用例, ネット上の紹介やコミュニティを読んでみて下さい。

\TeX や \LaTeX のことを最初に見聞きしたのは大学の講義や数学科図書館であったのか, 雑誌であったのか今となっては定かではありません。当時はインターネットのことなど知らず, 良くても大学間のネットワークや, 音響カプラを使ったパーソナルコンピュータネットワーク程度でした。つまり, 情報は本か雑誌, あるいは人から直接得るものだったわけです。

\TeX や \LaTeX のことを最初に印象付けてくれたのは弟です。とても感謝しています。まだフロッピディスクで起動する Macintosh で見せてくれました。兄としては悔しいよりも驚きを覚えていえます。

あれから 20 年が経ちます。先日も「いつごろから使ってられるのですか?」「もう何でもわかりますよね」などと言われましたが, 未だに判らない事だけです。それでも, 何とか便利に使ってきました。

もっと便利にできるような画期的なものが世に出てくるだろうと思いつけて来ましたが, どうやらそんなものは無いようです。そういう意味では Dr.Knuth のこのアイデアと創造の賜物である \TeX は偉大です。

でもそういう事情で仕方なく作成しました。本当に仕方なくです。ですから多々ある不備や誤動作は皆さんで修正してください。そういう意味も含めて, それに マクロの確認や \LaTeX での本の作成の練習も兼ねてここに公開します。

2005.10.20. 鱈本父 (*Mr. Moonlight*)

目次

第 1 章	qbgraph の概要	1
1.1	概要の前説	1
1.2	使用例	2
1.3	必要なファイルと構成	3
第 2 章	qbgraph の実際	4
2.1	早速使うために，最低限のコマンドを	4
2.2	線分を描くマクロ	4
2.3	点を描くマクロ	5
2.4	点を決めるマクロ	6
2.5	点を決めるマクロ群	8
2.6	点の定義，回転点，中点，交点などを使った作図例	9
2.7	点の定義，分点，交点などを使った作図例	10
2.8	点のラベルに関して	11
2.9	線のラベルに関して	12
第 3 章	繰り返し処理と qbgraph	13
3.1	多角形	13
第 4 章	qbgraph で関数を扱う	16
4.1	関数の計算のマクロ化について	16
4.2	2 次関数のグラフ	17
4.3	三角関数のグラフ	19

第 1 章

qbgraph の概要

1.1 概要の前説

`\qbgraph.sty` は、高等学校で数学を教える立場で、 \LaTeX の便利さを実感しながらも、その `picture` 環境の使い勝手の悪さに長年我慢してきた数学教師の勝手マクロ集です。

世の中には \TeX で図形を描いたり使用したりする為に、`metapost`^{*1} や `PSTrick`^{*2}、当に初等中等教育での使用を目的に作られた `emath` シリーズ^{*3} など色々と便利なものが存在しています。

ただ何れも、`Postscript` や `Perl` などの別のスクリプト言語やソフトウェアを必要としたり、`typeset`^{*4} の途中にただでさえログファイルや `DVI` ファイルなどの関連ファイル^{*5} が量産される \TeX で、さらに雑多なファイルをフォルダ内に置いてしまう等、整理が不得手な僕としては、個人的に面倒だなあと感じ、実用には至りませんでした。

ちょっと使う、日常的に使う、週に何枚も作る、そういう使い方だと面倒なわけです。第一、現場で使うプリントは `B4` 一枚が基本だし、手間がかからず散らからないものが良い。できれば、基本的な \LaTeX セットとちょっとしたマクロだけで済ませたい。というわけで、マクロ群 `\qbgraph?.sty` を作成しました。

「車輪の再発明」でなければ良いのですが。

また、指導要領の改訂で、高校数学に中学校で習っていた幾何の一部が廻って来ています。そこで特に点や線、円、角度といった図形を楽に扱いたいというのも強い動機です。

ネット上のあちらこちらに存在する掲示板では、「任意の傾きの直線はどうすれば引けるか？」などの図形に関する質問が散見できます。回答はいずれも、上述の外部依存の解決策です。それはそれで良いのですが、`picture` 環境でも十分じゃないか、ということです。

つまり上記の問には `\qbezier` (始点の座標) (中点の座標) (終点の座標) が答え^{*6} の候

*1 \TeX Wiki の <http://oku.edu.mie-u.ac.jp/~okumura/texwiki/?MetaPost> などを参照のこと

*2 同じく \TeX Wiki の <http://oku.edu.mie-u.ac.jp/~okumura/texwiki/?PSTricks> などを参照のこと

*3 <http://emath.s40.xrea.com/> を参照のこと

*4 \TeX は組版ソフトです。この文書も \TeX で記述されています。ワードや一太郎といったワープロとは違いますし、`Quark` や `InDesign` のような `DTP` ソフトとも違います。近いのはネット上の `html` ファイルでしょうか。ファイル自身はテキストファイル。文章とコマンドの羅列です。それをコンパイルすることで、実際の文書 (`dvi` ファイル) を実現します。 \TeX はこの操作 (コンパイル?) を `typeset` (文字を打つ?) と称します。

*5 `.aux`, `.dvi`, `.log`, `.bak`, などなど、環境や OS によって違うようですが。

*6 もちろん、標準的なオプションでもある、`epic.sty`, `\eeptic.sty` や `exline.tex` のような既成のも

補にあるはずですが。ただこれを実現するには、中点の座標を計算する必要があり、その計算は一人人間がするのか、外部のソフトに任せるのか、 $\text{T}_\text{E}\text{X}$ のマクロに組み込むのかということを見ると、既に在るもので楽に済ませちゃおうってことになるのでしょう。

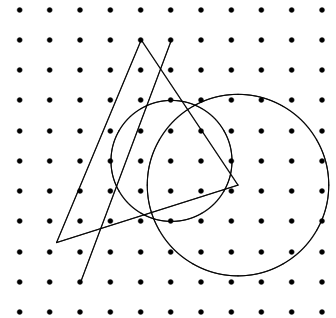
実際私も今回の指導要領改訂までは、良いじゃないかと思い、直線は頭や手や時には電卓や他人の頭で中点の座標を計算して使っていました。でも、今は面倒臭い。楽がしたいわけです。それも気軽に。

さて、念のためですが、本を作ったり論文を書いたりするには、この`\qbgraph.sty`は向かないでしょう。あくまで、一枚から十枚程度のプリントを作るためのものです。

1.2 使用例

では早速どういうことが出来るか見て頂きましょう。

```
\begin{picture}(10,10)(0,0)
  \qbPline(2,1)(5,9) % (2,1) から (5,9) へ線分
  \qbPointDef A(4,9) % 点 A(4,3) を覚えて
  \qbPointDef B(1.2,2.3) % 点 B も
  \qbPointDef C(7.2,4.2) % 点 C も
  \qbGlinesclosed ABC. % 閉じた線分 ABC
  \qbPcirc(5,5)2 % 中心 (5,5) 半径 2 の円
  \qbGcirc C(3) % 中心が C で半径 3 の円
\end{picture}
```



上の図^{*7*}^{*8}は簡単な例ですが、基本的な使い方はこれで十分理解できると思います。

それぞれのマクロは名前の由来^{*9}である`\qbezier`による2次^{*10}bezier曲線^{*11}を用いて線を描きます。そのために、座標を覚えたり、座標で計算したり、その計算結果を利用して描いたり、繰り返したり（円は確か30度ずつ繰り返して描いています。）といったことを実現しています。再利用可能（というよりは部品）なマクロ群ですので、自分で更に特殊なマクロを組上げることはメモリの問題などがなければ大丈夫でしょう。

もっと大事なことは`picture`環境をそのまま使っていますから、いわゆる普通のコマンド（`\put(座標){置くもの},\dots`）が従来通り使えます。あくまでも、`epic.sty`や`eepic.sty`のような`picture`環境の機能拡張マクロなわけです。必要になる実数計算に

のも候補でしょうけど

^{*7} この図では、`\unitlength=5truemm` としていますが煩雑なので本文には入れていません。また、図の範囲が判りやすいように、`\qblatticed{0}{0}{11}{11}` というマクロで格子点を描き加えています。もちろん、実際に使う時は、縦横のサイズ（上の例では(10,10)）や原点の位置（上の例では(0,0)）、`\unitlength`の長さも好きにしてください。個人的には`\unitlength=5truemm`で縦横10なら要するに縦横5cm だあって、図の大きさが把握しやすいので、こう設定してみました。

^{*8} 注意 以下のページでは煩雑さを避ける為、描画環境の宣言`\begin{picture}(##,##)`と終了の宣言`\end{picture}`については省くことにします。真似をする場合は各自で補ってください。

^{*9} `\qbgraph`のマクロの名前は全て（多分）`\qb`で始まります。`\qpline`という感じですね。ちなみに、マクロ中で用いる計算や座標用の変数は`\tmp`で始まるようにしたつもりですが...

^{*10} `\qbezier`の`q`は「2次」という意味の英語quadraticの`q`です。

^{*11} 「bezier曲線とは」という話は、別に譲ります。始点と終点と制御点を幾つか（`\qbezier`では一つ）指定することで、曲線を指定する形式のことです。制御点を中点にすれば、直線となります。

は `eclarith.sty` を利用しています。従って、そこそこの正負の実数の加減乗除と三角関数 (`eclarith.sty` 準拠) が使えます。

1.3 必要なファイルと構成

自分で使うために作り出したので、当初は一つのスタイルファイルにまとめていました。が、エディタ^{*12}が時々 `TEX`, `LaTeX` のコマンドを解釈しようと簡易文法解析?するので、煩くなって分けました。

本体である `\qbgraph.sty` は、基本的にはスタイルファイル群の呼出し役のファイルです。`\qbgrapha~z.sty`^{*13}が中身です。また、`eclarith.sty` を必要としますから、

```
\usepackage{ascmac,mtcastle,eclarith,qbgraph}
```

のように `eclarith.sty` と一緒に `\usepackage{...}` で呼出し指定して下さい。ちなみに、上の例の `mtcastle.sty` はまた別のマクロ集です。気になる方も、気にせず済ませてください。

*12 WinShell というエディタです。

*13 いくつあるかは、依然開発途上状態ですから、この原稿執筆時でもわかりません。それほどに整理下手なんでしょう。

第 2 章

qbgraph の実際

2.1 早速使うために，最低限のコマンドを

さて，先に書いた通り，無手勝流の気楽なマクロ群です。必要なマクロから順に作って構成したので，誰でも改造，改良して好きに作り直すことが出来るでしょう。そこで，基本的な命令（コマンド，マクロ）の仕様と使用を紹介していきます。

2.2 線分を描くマクロ

qbPline, qbPlined
`\qbPline(x1, y1)(x2, y2)` 使用例 `\qbPline(1,2)(3,4)`

という形です。このマクロは実は

qpline, qplined
`\qpline{x1}{y1}{x2}{y2}` 使用例 `\qpline{1}{2}{3}{4}`

`\qpline{x1}{y1}{x2}{y2}` という qbgraph の最初期に作成したマクロを呼んでいます。ずっと L^AT_EX の `\newcommand` でマクロ定義していたので，この形^{*1}なのです。

$$\frac{x_1 + x_2}{2} = \text{\tmpx}$$

のような^{*2}計算で，中点の座標を一時記憶させて，`\qbezier` の中点に使っているわけです。

同じようなマクロですが

qbGline, qbGlined
`\qbGline P点の名前 Q点の名前` 使用例 `\qbGline AB`

は，点の名前と座標を後述のマクロ `\qbPointDef#1(#2,#3)` で前もって定めておけば，線分の名前（例えば，AB とか PM とか）で線分を描画することができるマクロです。

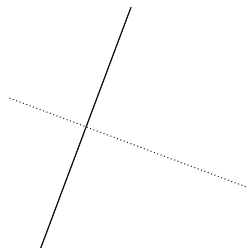
実際に使ってみた様子を観てみましょう。

^{*1} L^AT_EX の `\newcommand` でマクロ定義する場合，複数の引数の一つ一つを `{}` で囲んで並べて記述しなくてはならないようです。違うのかもしれませんが。

^{*2} 実際は `\eclarith.sty` のマクロで `\Add{#1}{#2}{\tmpx}` 足して `\Div{\tmpsx}{2}{\tmpx}` で割るという具合に計算しています。`\tmpx` や `\tmpsx` は所謂マクロ内で使っている代表的な変数格納用のマクロです。T_EX では，変数もコマンドも何もかも `\` で始まる名前と呼ばれるのです。

1. 最初に作成した基本マクロ群

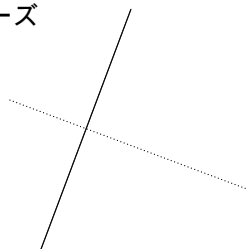
```
\qbline{2}{1}{5}{9}  
\qblined{1}{6}{9}{3}
```



点の各座標を一々{.}で括るのでソースが見辛いし面倒です。

2. 次に\def で座標の形で指定できるようにした\qbP シリーズ

```
\qbPline(2,1)(5,9)  
\qbPlined(1,6)(9,3)
```



新しく定義しなおして座標や点の名前で描けるようにした\qbP で始まるマクロ群です。点の指定を座標表記にしました。

3. 線分 AB のように、点の名前で指定できるようにした\qbG シリーズ

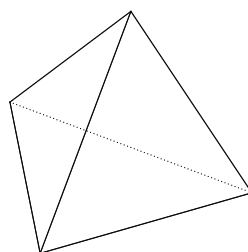
```
\qbPointDef P(2,1)  
\qbPointDef Q(5,9)  
\qbPointDef A(1,6)  
\qbPointDef B(9,3)
```

と点の名前と座標を定義しておけば

```
\qbGline PQ  
\qbGlined AB
```

更に

```
\qbGlinedsclosed APBQ.
```



これらは\qbG の名称で統一された、点の名前で指定して処理するコマンド群です。点の座標と名前を定義すれば、点の名前で指定して線を引きます。特に,\qbGlines,\qbGlineds,\qbGlinedsclosed は引数として点の名前の列を指定できますから便利です。

結果は同じですね。

2.3 点を描くマクロ

さて、線分が簡単に引けるようになれば、ほぼどんな図形でも描けるわけですが、\qbGline のように、点の名前で線分を引こうと思えば、点の名前と座標の定義が必要です。また、初等幾何の問題を描画しようと思えば、線分の交点であるとか中点であると

か、垂線の足であるとか、様々な点が必要になります。円を考える場合は、中心の点も見たい。更には、点に名前があるのなら、描いたときに点の名前も描き込みたい。

そんなこんなのマクロ群です。

<code>\qbPpoint</code>	使用例 <code>\qbPpoint(1,2)</code>
------------------------	---------------------------------

`\qbP` ~ ですから、座標 (#1,#2) を引数にするマクロです。点を描きます。やはり、

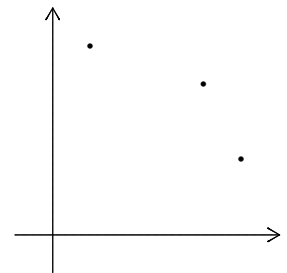
<code>\qbpoint</code>	使用例 <code>\qbpoint{1}{2}</code>
-----------------------	---------------------------------

というマクロを呼び出しています。また、

<code>\qbGpoint</code>	使用例 <code>\qbGline A</code>
------------------------	-----------------------------

と点の名前で描くこともできます。ただし、このマクロの前に `\qbPointDef #1(#2,#3)` などで、点の名前と座標が定められている必要があります。では用例^{*3}です。

```
\begin{picture}(7,7)(-1,-1)
  \qbPpoint(5,2) (5,2) に点
  \qbpoint{1}{5} (1,5) に点
  \qbPointDef A(4,4) 点 A(4,4) を覚えて
  \qbGpoint A 覚えたんだから点 A を描いて
\end{picture}
```



2.4 点を決めるマクロ

次に、点の名前と座標の設定について少し見ておきましょう。既にかいたとおり、当初は判らない事が多くて、マクロの引数を全て、`{#1}{#2}~{#N}`^{*4}のように一々`{}`で括る標記の仕様で作っていました。それが、後になって、 \TeX の `\def` だと、引数の形が自由に設定できることが判ったので、様々な座標を引数にするコマンドについて、座標を `(#1,#2)` の形で指定できるように定義しなおし^{*5}しました。

その過程^{*6}で、マクロの名前を引数で指定できるコマンド `\@namedef` があることがわかりました。

少し難しいですが、例えば点 P の座標を (x,y) としたいときに、点の名前 P をマクロ名に含んだマクロを $P(x,y)$ を引数にすることで定義 (`\def`) することができるわけです。これは凄いことです。これなら、配列は要らないのも当然なのかもしれません。もう少し具体的には、例えば点 A(3,4) を引数にすれば、名前 `\def\qbPtA{A}`, x 座標

^{*3} 位置関係が判りにくいので図には座標軸を入れてあります。

^{*4} といっても確か `\newcommand` で設定できるマクロの引数は 8 個だか 9 個だかまでなので、 $N \leq 9$ 程度です。

^{*5} 所謂、`\qbP` シリーズです。定義しなおしは今のところ `\qbgraphp.sty` 内で行っています。

^{*6} やりたいことを実現する為にネット上を調べたり、尋ねたりして、ある日、良いページを発見しました。

`\def\qbxofPtA{3}`, y 座標 `\def\qbyofPtA{4}` の3つのマクロを自動的に作成する仕様
にしました。実際のコードです。

```
\def\qbPointDef#1(#2,#3){
  \def\tmpDummysname{qbPt#1}\@namedef\tmpDummysname{#1}
  \def\tmpDummysname{qbxofPt#1}\@namedef\tmpDummysname{#2}
  \def\tmpDummysname{qbyofPt#1}\@namedef\tmpDummysname{#3}}
```

`\def\tmpDummysname{qbPt#1}`の部分で点名#1 を含んだ名前をマクロ`\tmpDummysname`
に定義します。それから、その名前を持ったマクロを`\@namedef\tmpDummysname{#1}`で
定義^{*7}しています。

実際のコードと実行例^{*8}です。

```
\qbPointDef G(3,-2)
点の名前は\qbPtG で, $x$座標は$qbxofPtG$で, $y$座標は$qbyofPtG$です。
```

———— typeset ————

点の名前は G で, x 座標は 3 で, y 座標は -2 です。

さて、一々点 G の x 座標を得るのに`\qbxofPtG` と書くぐらいなら、こんなの要
らないじゃないかと思ったあなたは正解です。`\qbPointDef` と対を成すマクロ、
`\qb##ofPointDef` という一連のマクロを作りました。これは、点の名前を引数にして、
別の引数に x 座標や y 座標を返すものです。

```
\def\qbxofPointDef(#1,#2){
  \def\tmpDummysname{qbxofPt#1}\qbdef{\@nameuse\tmpDummysname}{#2}
}
```

上手く動いてくれるようにした結果がこうです。`@nameuse` というマクロが引数を名
前に持つマクロを実行してくれます。マクロを実行するといってもこの場合は登録
されている点#1 の x 座標を返します。その値を`\qbGraph` の代入マクロ`\qbdef{代入
するもの}{代入されるもの}`で#2 に受け渡すわけです。つまり、上の例で定義される
`\qbxofPointDef(#1,#2)` は点の名前#1 を引数にして、その点の x 座標を#2 に戻し
ます。

この項の最初に紹介した、`\qbGpoint` の場合、

```
\def\qbGpoint#1{
  \qbxofPointDef(#1,\tmpSx,\tmpSy)
  \qbPpoint(\tmpSx,\tmpSy)
}
```

^{*7} ここに現れるマクロはいずれも、何かを「する」ものではなく、単なる値の「入れ物」的な扱いです。従っ
て、マクロ定義の内容部分である`\def`の最後の`{}`内はいずれも拍子抜けするほど単純です。判りたい人
はよく読んで下さい。

^{*8} 最初の`\qbPointDef G(3,-2)`は、定義するだけで表面上は何もしません。この定義により産まれたマク
ロ`\qbPtG`、`\qbxofPtG`、`\qbyofPtG`はそれぞれ点の名前 G、 x 座標 3、 y 座標 -2 を返しています。

という具合に、`\qbxyofPointDef` で名前#1 の点の x, y 座標を `\tmpSx, \tmpSy` に返してもらって、それを引数に、`\qbPpoint` で点を描くという仕様になっていますね。とって単純で見易いソースコードではないでしょうか。

ただし、案の定というか何というか、点の名前が定義されていない場合の処理^{*9}は想定外です。正しく使って下さい。

では、先程の例をもう一度みて、ソースと結果を見比べてみましょう。

`\qbPointDef G(3,-2)` 点の名前は `\qbPtG` で、 x 座標は `\qbxofPtG` で、 y 座標は `\qbyofPtG` です。

typeset

点の名前は G で、 x 座標は 3 で、 y 座標は -2 です。

```
\qbPointDef G(3,-2)
点の名前は G で、\qbxyofPointDef(G,\tmpx,\tmpy)%
 $x$ 座標は  $\tmpx$  で、 $y$ 座標は  $\tmpy$  です。
```

typeset

点の名前は G で、 x 座標は 3.0 で、 y 座標は -2.0 です。

違いが判るでしょうか。`\qbxyofPointDef` を実行しているところに、不用なスペース^{*10}が入っていますね。また、 x 座標は、 y 座標の `\tmpx, \tmpy` の値が実数仕様^{*11}になっていますね。

2.5 点を決めるマクロ群

`\qbG` シリーズを使う為には、点の定義をするマクロが必須です。

```
\qbPointDef
\qbPointDef P点の名前( $x$ 座標,  $y$ 座標)          使用例 \qbPointDef A(2,-3)
```

です。

また、ある程度点が指定できてから使うものとして、

```
中点 qbGmidPoint
\qbGmidPoint AB線分 M中点名          使用例 \qbGmidPoint PQ N
```

```
交点 qbGcrossPoint
\qbGcrossPoint AB線分 CD線分 P交点名  使用例 \qbGcrossPoint AM BN G
```

^{*9} いわゆるエラー処理などと呼ばれているプログラミングでは大切で手間の掛る部分です。

^{*10} 何故なのか原因がわかりません。おそらく、`@nameuse` 等に問題があるのでしょうか。誰か修正して下さい。

^{*11} こちらは計算用に用意した数値定義の `\qbdef` で実数値扱いにしています。ですから当然といえば当然の結果です。所謂単位抜きの寸法ということです。

回転点 qbGrotPoint

`\qbGrotPoint AB線分 θ 回転角度 P点名` 使用例 `\qbGrotPoint PQ 45 R`
A を回転の中心にして、線分 AB を θ (度数法で指定) 回転したときの、B の位置を P とする。

分点 qbGdivPoint

`\qbGdivPoint AB線分 m:n分比 P分点名` 使用例 `\qbGdivPoint BC 5:3 L`

垂線の足 qbGverticalPoint

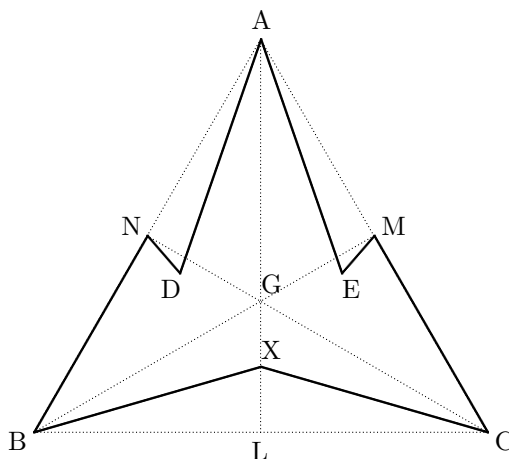
`\qbGverticalPoint A垂線を下す始点 BC垂線を下す線分 H垂線の足` 使用例
`\qbGverticalPoint A BC H`

などがあります。

2.6 点の定義，回転点，中点，交点などを使った作図例

某学校の校章の作図を具体例に

```
\qbPointDef B(0,0)      \qbPointDef C(5,0)
\qbGrotPoint BC 60 A
\qbGlined AB    \qbGlined BC    \qbGlined CA
\qbGverticalPoint A BC L
\qbGverticalPoint B CA M
\qbGverticalPoint C AB N
\qbGlined AL    \qbGlined CN    \qbGlined BM
\qbGcrossPoint AL BM G
\qbGmidPoint GL X
\qbGmidPoint GB Y
\qbGmidPoint GC Z
\qbGcrossPoint AY NX D
\qbGcrossPoint AZ MX E
\thicklines
\qbGlinesclosed AEMCXBND.
```



点 B を原点にとり，点 C を (5, 0) にとり，線分 BC を B を中心に 60° 回転させた (`\qbGrotPoint`) 先の点を A として，正三角形 ABC を設定します。`\qbGlined` で点線で描きます。

`\qbGverticalPoint` で各頂点から対辺への垂線の足を決め、垂線を点線で描きます。
`\qbGCrossPoint` で垂線の交点を G と決めます。`\qbGmidPoint` で GL, GB, GC の中点を X, Y, Z として、再び`\qbGCrossPoint` で AY と NX の交点を D 、 AZ と MX の交点を E とします。

最後に、`\thicklines` で太線に設定して、`AEMCXBND` と結んで出来上がりです。
 図には判りやすいように、`\qbGplabel` で点の名前 (Label) を入れています。
 マクロの名前や引数の形式が組上げた本人も怪しいのですが、うまく使えば、この通り、作図作法 (描き方) 通りに図をが描けます。
 勿論、作図の過程や計算の結果を全て見せる必要は無いので、先の校章の場合でも、



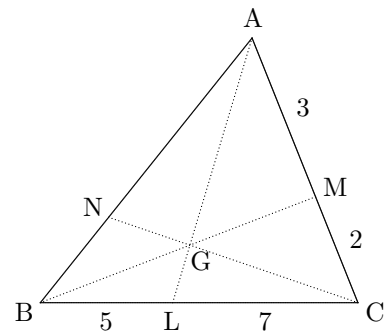
のように結果だけ、或いは必要な部分だけ表示すれば良いわけです。

2.7 点の定義，分点，交点などを使った作図例

次はもう少し高等学校数学用に実用的な例を見ましょう。チェバの定理やメネラウスの定理の問題で使用する図です。

問題 三角形 ABC で、辺 BC を $5:7$ に、辺 AC を $3:2$ に内分する点を L, M とする。 AL, BM の交点を G とし、 CG と AB の交点を N とするとき、 $AG:GL, AN:NB$ を求めなさい。

```
\qbPointDef A(4,5) \qbPointDef B(0,0)
\qbPointDef C(5,0)
\qbGline AB \qbGline BC \qbGline CA
\qbGdivPoint BC 5:7 L
\qbGdivPoint AC 3:2 M
\qbGlined AL \qbGlined BM
\qbGCrossPoint AL BM G
\qbGCrossPoint CG AB N
\qbGlined CN
```

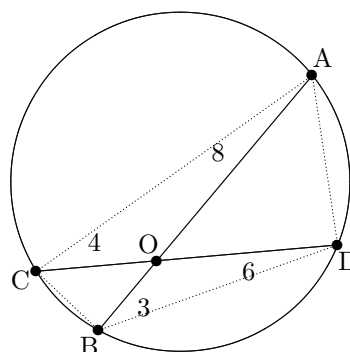
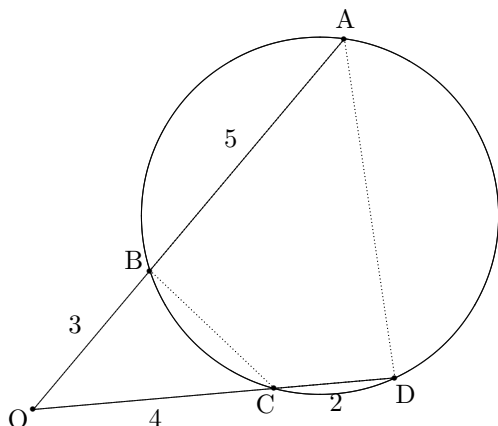


次は、方べきの定理^{*12}の図を描いてみましょう。

方べきの定理 線分 AB と CD の交点を O とし、4 点 ABCD が同一円周上にあるとき、

$$OA \cdot OB = OC \cdot OD$$

```
\qbPointDef O(0,0) % 原点を 0 として
\qbPointDef A(8,0)\qbGrotPoint OA 50 A % 数直線上に 8x3=4x6 となるように
\qbPointDef B(3,0)\qbGrotPoint OB 50 B % 4 点 ABCD を定めて
\qbPointDef D(6,0)\qbGrotPoint OD 5 D % AB は 50 度, CD は 5 度回転させて
\qbPointDef C(4,0)\qbGrotPoint OC 5 C % 四角形 ABCD を構成
\qbGlinedsclosed ABCD.\qbGlines AOD. % 閉折れ線と折れ線を描きます
\qbGpoints ABCD0. % 点列を描きます
\qbGmidPoint AB M \qbGrotPoint MB 90 E % 円を描く為に中心が必要ですから
\qbGmidPoint CD N \qbGrotPoint ND 90 F % 辺の垂直二等分線上の点を用意して
\qbGcrossPoint ME NF P \qbGcirc1 PA % 交点で中心を決定して, 円を描く
```



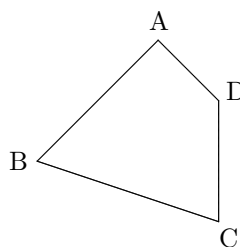
2.8 点のラベルに関して

これまでのところを見てきて、作図については判ったから、点のラベル(名前)をどうつけるのが知りたいと、思っているのではないのでしょうか。

点のラベル `\qbGptlabel`

`\qbGptlabel A点の名前(0~11表示位置)` 使用例 `\qbGptlabel P(2)`
 点 A の名前 A を点 A を中心とした一定半径円周上 12 分割した位置に表示する。

```
\qbPointDef A(3,4) \qbPointDef B(1,2)
\qbPointDef C(4,1) \qbPointDef D(4,3)
\qbGlinesclosed ABCD.
\qbGptlabel A(3) \qbGptlabel B(6)
\qbGptlabel C(10) \qbGptlabel D(1)
```



^{*12} この図では、 $OA = 8, OB = 3, OC = 4, OD = 6$ として描いています。

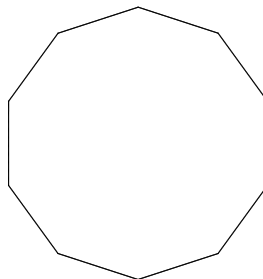
第3章

繰り返し処理と qbgraph

3.1 多角形

高等学校の数学の順列組合せでは、多角形の対角線の本数を計算させますが、

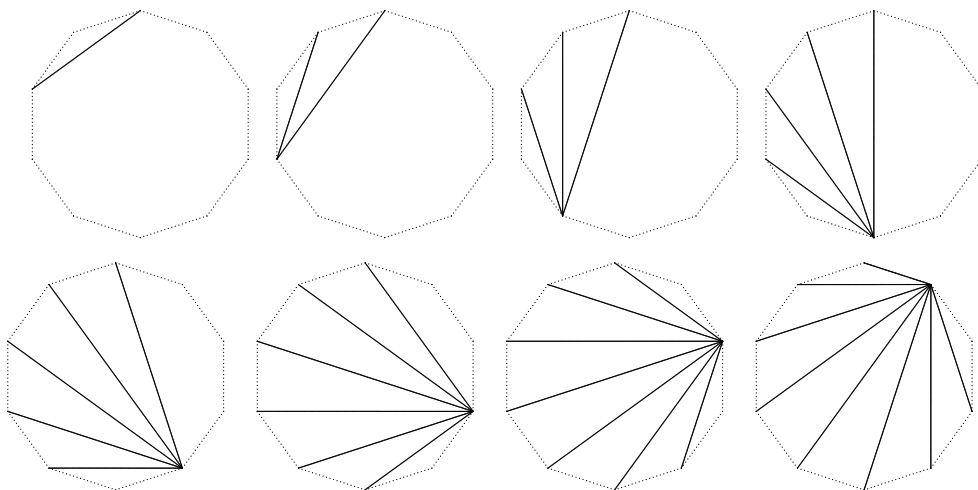
問題 10 角形の対角線の本数を数えなさい。



この数え方は組合せを用いて、

$${}_{10}C_2 - 10 = \frac{10 \cdot 9}{2 \cdot 1} - 10 = 45 - 10 = 35 \text{ (本)}$$

等と教わり教える訳ですが、次のように数えることもできます。



最後の一本は余分ですから、計算としては、

$$1 + 2 + 3 + \dots + 7 + 7 = 1 + \dots + 8 - 1 = \frac{(1+8) \times 8}{2} - 1 = 36 - 1 = 35 \text{ (本)}$$

ですね。

さて、このような説明で図を使う場合、何度も同じ事、あるいは同じようなことを繰り返すこととなります。この問題の図の場合では、多角形を描くというレベルで、「点を回転させながら線分を引く」ということをしていますし、「あるパターンで対角線を引いた図を次々に描く」というのも当然に繰り返しです。このような繰り返しは $\text{T}_{\text{E}}\text{X}$ では `\loop.. \if___.. \repeat` という命令で実行できます。

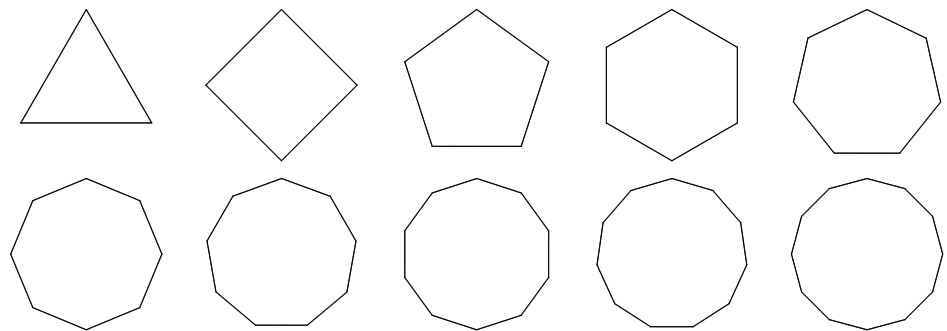
では実際に先の図を描くテキストを見てください。

```
\begin{picture}(##,##)(##,##)
\qbPointDef 0(0,0)      % 原点 0 (多角形の中心) を決めます
\qbPointDef A(0,1)     % 多角形を描き始める頂点 A (今回は上の点) を決めます
\newcount\tmpT \tmpT=0 % 繰り返しのカウンタのために、\tmpT という変数を用意
\loop                  % 繰り返しを始め (\loop) ます。
\ifnum \tmpT <10      % 繰り返しは条件 \tmpT<10 を満たす場合続けます
  \qbGrotPoint OA 36 B % 0 を基準に A を 36 度回転させた点を B とします
  \qbGline AB         % 線分 AB を描きます
  \qbGrotPoint OA 36 A % 0 を基準に A を 36 度回転させた点を次の A とします
  \advance \tmpT by 1 % 繰り返しのカウンタ \tmpT の値を 1 増やします
\repeat               % ここまでが繰り返しです
\end{picture}
```

ここでは 10 角形限定版ということで、36 度回転や 10 回という値を直接頭で計算して込めてありますが、もちろんこれも変数化、計算値として扱うようにすれば、汎用化できます。

```
\qbGregpolygon P点の名前 Q点の名前(角数) 使用例 \qbGregpolygon OA(7) 七角形
```

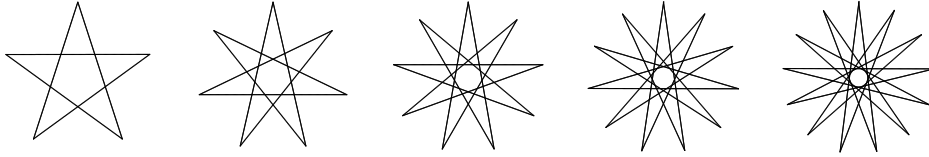
というわけで勿論用意してあるわけです。



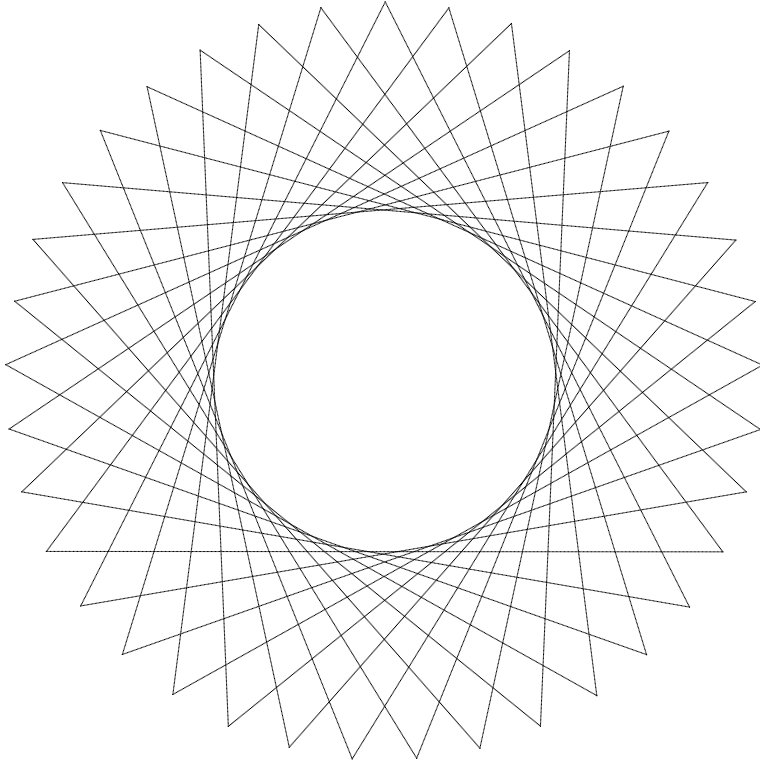
ついでに、

```
\qbGregpolygonn P点の名前 Q点の名前(角数) 頂点を飛ばす数.
使用例 \qbGregpolygonn OA(7) 七角形
```

という星型を描くものも用意しました。



なかなか綺麗ですね。



第 4 章

qbgraph で関数を扱う

4.1 関数の計算のマクロ化について

さて、一定図形が描けるようになり、繰り返しも使えるようになれば、やはり関数のグラフも描かせたくなるというものです。ところが、一般のプログラム言語とは違い、 $\text{T}_\text{E}\text{X}$ は本来は浄書の為のソフトウェアであり、計算は何とかできても、計算式の扱いは不得手です。つまり、

$$y=2*x+3 \text{ とか } y=2*x^2+3x-5 \text{ とか } y=\sin(2*x-\text{PI}/3)$$

などという記述は通用しません。

ただ、高等学校の数学で扱う関数は高々知れていますから、何とかするのはないかと思っていたら案の定こんな手^{*1}が使えることが判りました。

```
\def\qbFabc y=#1x^2+#2x+#3|x=#4y=#5{\  
\Mul{#1}{#4}{\tmpFv}\Add{\tmpFv}{#2}{\tmpFv}  
\Mul{\tmpFv}{#4}{\tmpFv}\Add{\tmpFv}{#3}{#5}  
}
```

これは、2 次関数の計算をするマクロです。`\qbFabc -2x^2+-3x+4|x=-1y=\tmpFy*2` のように用いて、 $f(x) = -2x^2 - 3x + 4$ に対して $f(-1)$ の値を計算して `\tmpFy` に返してくれます。書式のイメージは、積分や物理の計算などでよく用いる、

$$y = -2x^2 - 3x + 4 \Big|_{x=-1}$$

の形です。

^{*1} これまで見て来たように、 $\text{T}_\text{E}\text{X}$ では動作や計算を組合せることで `\qbGline` のような新しいコマンドを作るときも、単に `\tmpFx` のような計算に用いるだけの値を持つ入れ物 (変数と呼ぶとややこしや) を作る時も、なんであれ `\def\{名前\}{その内容}` という形でその都度 (名前) と {内容} を指定して新しく作成します。詳しくは知らない (恐ろしいことです。) のですが、それでなんとかやっています。で、変数 (というか参考にするもの? 実際のところ、変数と書いてますが $\text{T}_\text{E}\text{X}$ は基本的に数であれ何であれ全てを文字列として扱っていると捉えると判り易いようです。) を用いる場合は、その名前の部分に #1 (一つ目の変数)、#2 のように入れ込んでいきます。この例では、`\def` と { に挟まれた `\qbFabc y=#1x^2+#2x+#3|x=#4y=#5` が全てこのコマンド (マクロ) の名前というか、パターンになっています。この形 (順序、パターン) で $\text{T}_\text{E}\text{X}$ は動作を判別 (パターンマッチング) して、実際の行動に移るわけです。もう少しだけ判りやすく? 表記してみると、`\qbFabc y=変数 1x^2+ 変数 2x+ 変数 3|x=変数 4y=変数 5` という具合に名前を付けることができるという手なのです。

^{*2} 定型の関数 (Function) で係数 abc を指定するので `\qbFabc...` という命名です。

なあんだ，簡単じゃないかと思ったあなたは幸いです。2次関数や分数関数，無理関数など，定型のパターンを持つ関数であれば，各々をこの形で定義することで計算については解決しそうです。

実際に上のマクロを使って計算してみましょう。

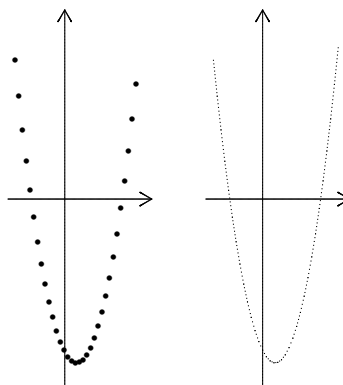
$f(0.0) = 4.0$	$f(1.19998) = -2.47984$
$f(0.2) = 3.32002$	$f(1.39998) = -4.11981$
$f(0.4) = 2.48004$	$f(1.59998) = -5.91977$
$f(0.59999) = 1.48006$	$f(1.79997) = -7.87971$
$f(0.79999) = 0.32008$	
$f(0.99998) = -0.9999$	

何だか微妙に狂っています*3 が，グラフを描くには支障は無さそうですね。

4.2 2次関数のグラフ

では早速このマクロを使って2次関数のグラフを描いてみましょう。

```
\qbParrow(-3,0)(3,0)1
\qbParrow(0,-5)(0,5)1
\qbdef{-1.3}{\tmpFx}
\loop
\ifdim \tmpFx pt<2.0 pt
\qbFabc y=3x^2+-2x+-4 |x=\tmpFx y=\tmpFy
\qbPpoint(\tmpFx,\tmpFy)
\Add{\tmpFx}{0.1}{\tmpFx}
\repeat
```



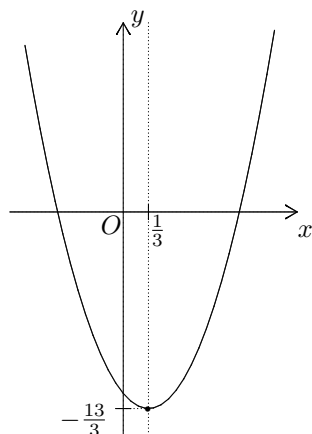
上のソースコードは左側の図のものです。x座標を0.1刻みで-1.3から2まで変えながら代入し，その都度，関数の値，つまりグラフでのy座標を求めて，点をプロットしています。

簡単ですよ。準備は大変でしたが，計算や線引きが用意できれば，ちょっと不便なBASICでプログラムを書く程度の感覚でグラフの描画が実現できます。

右側のもは，直前に計算した点の座標を覚えておいて，\qbPlinedで繋いだものです。

あとは少しの工夫を加えれば，

*3 TeXが根源の問題なのか，\eclarith.styが理由なのか。そのあたりのことが判りません。どなたか原因と解決法を教えてください。



のような使える図が比較的簡単にそこそこ綺麗にできあがります。

何と言っても学校現場での $\text{T}_\text{E}\text{X}$ の有用性の第一は再利用性です。比較的簡単とはいっても、それなりに苦勞するのでしょうけど、一度作ってしまえば、あとは変更して再利用できることが期待できますから... (ぶつぶつ)

では、少し長いですが、整形版のソースです。

```

\def\qbFabc r y=#1x^2+#2x+#3|x=#4y=#5(#6){
\Mul{#1}{#4}{\tmpFv}\Add{\tmpFv}{#2}{\tmpFv}
\Mul{\tmpFv}{#4}{\tmpFv}\Add{\tmpFv}{#3}{\tmpFv}
\Mul{\tmpFv}{#6}{#5}
}
%
\qbdef{0.6}{\tmpFr}
\newcount\tmpFNa \tmpFNa=3
\newcount\tmpFNb \tmpFNb=-2
\newcount\tmpFNC \tmpFNC=-4
%
\unitlength=1truecm
\begin{picture}(5,6)(-2.5,-3)
% 座標軸系
\qbParrow(-1.5,0)(2.3,0)\qbParrow(0,-3)(0,2.5)1
\qbPptlabel(2.3,0)$x$(10)
\qbPptlabel(0,2.5)$y$(1)
\qbPptlabel(0,0)$O$(7.5)
% 頂点系
\Mul{\tmpFNa}{-2}{\tmpFtx}
\Div{\tmpFNb}{\tmpFtx}{\tmpFtx}
\qbFabc r y=\tmpFNa x^2+\tmpFNb x+\tmpFNC |x=\tmpFtx y=\tmpFty(\tmpFr)
\qbPpoint(\tmpFtx,\tmpFty)
\qbPlined(\tmpFtx,-3)(\tmpFtx,2.5)
\qbPlined(0,\tmpFty)(\tmpFtx,\tmpFty)

```

```

\qbPline(\tmpFtx,-0.1)(\tmpFtx,0.1)
\qbPline(-0.1,\tmpFty)(0.1,\tmpFty)
\qbPptlabel(\tmpFtx,0)\frac{1}{3}(10)
\qbPptlabel(-0.5,\tmpFty)-\frac{13}{3}(7)
%
\qbdef{-1.3}{\tmpFx}
\qbFabcr y=\tmpFNa x^2+\tmpFNb x+\tmpFNC |x=\tmpFx y=\tmpFy(\tmpFr)
\loop
\ifdim \tmpFx pt<2.0 pt
\Add{\tmpFx}{0.1}{\tmpFnx}
\qbFabcr y=\tmpFNa x^2+\tmpFNb x+\tmpFNC |x=\tmpFnx y=\tmpFny(\tmpFr)
\qbPline(\tmpFx,\tmpFy)(\tmpFnx,\tmpFny)
\qbdefp{\tmpFnx}{\tmpFny}{\tmpFx}{\tmpFy}
\repeat
\end{picture}

```

4.3 三角関数のグラフ

次は、三角関数のグラフを見てみましょう。その前に三角関数の値の計算のためにマクロを作っておきましょう。先程と同様です。

```

\def\qbFabcsin y=#1sin(#2(x-#3/#4pi))|x=#5/#6pi y=#7{
\Mul{#5}{180}{\tmpFvv}\Div{\tmpFvv}{#6}{\tmpFvv}
\Mul{#3}{180}{\tmpFv}\Div{\tmpFv}{#4}{\tmpFv}
\Sub{\tmpFvv}{\tmpFv}{\tmpFv}
\Mul{\tmpFv}{#2}{\tmpFv}
\qbsin{\tmpFv}{\tmpFv}
\Mul{\tmpFv}{#1}{#7}
}

```

やはり同様に、

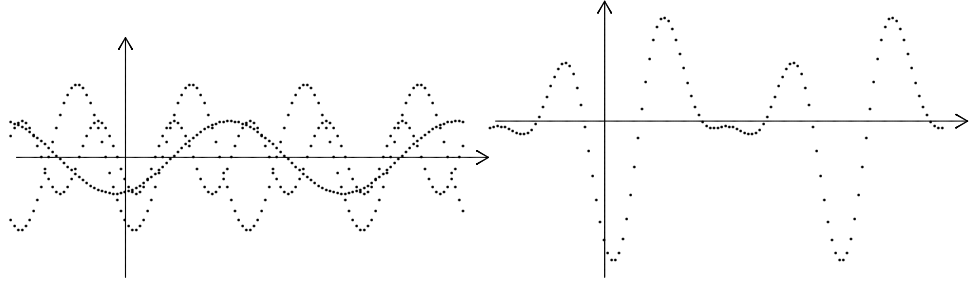
$$y = a \sin b \left(x - \frac{c}{d} \pi \right) \Big|_{x = \frac{e}{f} \pi}$$

という形です。 x への代入値は、高校現場での使用を前提に $\frac{e}{f} \pi$ の形で指示を出すようにしました。

実際の三角関数の値の計算は`\qbsin`等で行いますが、この`\qbsin`は弧度法(ラジアン)ではなく度数法(角度)で計算しますので、まず、弧度法で示される角の大きさを角度に直してから、`\qbsin`に代入する形をとっています。^{*4}

では、実際のグラフです。

^{*4} 先程の2次関数もですが、この値が、分数やルートを使って返ってくれば大変難しい楽なのですが、まだそこまでは用意してません。きっとまあ比較的簡単に実現できる筈です。皆さん頑張って下さい。



左側は，

$$y = 2 \sin 2 \left(x - \frac{\pi}{3} \right), y = \sin 3 \left(x + \frac{2}{5} \pi \right), y = \sin -1 \left(x + \frac{7}{12} \pi \right)$$

という三つの三角関数のグラフです。右側は，

$$y = 2 \sin 2 \left(x - \frac{\pi}{3} \right) + \sin 3 \left(x + \frac{2}{5} \pi \right) + \sin -1 \left(x + \frac{7}{12} \pi \right),$$

その三つを足した三角関数のグラフです。

では右側のグラフのソーステキストです。

```

\begin{picture}(13,8)(-3,-4.5)
\qbParrow(-3,0)(10,0)1
\qbParrow(0,-4.3)(0,3.3)1
%
\newcount\tmpN
\tmpN=-30
\loop
\ifnum \tmpN < 90
\qbFabcsin y=2sin(2(x-1/3pi))|x=\tmpN/30pi y=\tmpFy
\qbFabcsin y=1sin(3(x--2/5pi))|x=\tmpN/30pi y=\tmpFz
\Add{\tmpFy}{\tmpFz}{\tmpFy}
\qbFabcsin y=1sin(-1(x--7/12pi))|x=\tmpN/30pi y=\tmpFz
\Add{\tmpFy}{\tmpFz}{\tmpFy}
\Mul{3.1416}{\tmpN}{\tmpFx} \Div{\tmpFx}{30}{\tmpFx}
\qbPpointd(\tmpFx,\tmpFy)
\advance\tmpN by 1
\repeat
\end{picture}

```