数理解析研究所講究録
第1955巻 2015年 102-113

Risa/Asirによる曲線と関数グラフの描画
Drawing curves and graphs by Risa/Asir

大島利雄
TOSHIO OSHIMA
城西大学 理学部
FACULTY OF SCIENCE, JOSAI UNIVERSITY

1 はじめに

この数年は複素領域の常微分方程式の研究に興味を持ち，その研究の補助や結果の実現のために数式処理 Ris/Asir を活用してきた（cf. [O1]）．一方，昨年度から大学での初等教育を行う機会が増え，適切な教材作成や成績処理など教育活動においてもパソコンの活用が可能ではないか，と考え，昨年度行った講義の離散数学や複素平面，常微分方程式，初等整数論などには10進 BASIC を活用した．今年度は，数式がそのまま扱える，より高度な機能を持った Risa/Asir を用いることにした．

Risa/Asir において TEX と連携のための関数はいくつか既に作成していたので，それを利用して実際に教材作成を行った．今回はその中で曲線や曲面の描画にかかわって作成した5つの関数について述べる（現在250個以上の関数を作成済み．1年前は150個程度であった．cf. [O1, O2]）．

2 曲線の描画

x-y 平面における $y = f(x)$ のような関数のグラフ，あるいは $(x(t), y(t))$ というようなパラメータ表示をもつ曲線の描画を考察する．

- 少ないデータで綺麗で正確な曲線，すなわち拡大してもジャギーが現れず，指定した点を順に必ず通る．
- C^1 でない点や，不連続点，特異点を許し，必要に応じて局所的に自動的にデータの細かさを変える．
- カラーや塗りつぶしをサポートする．
- TEX で直接扱えるテキストファイルで，さらにエディタでも容易に修正可能な形式の出力をする．
- OS（Unix, Mac, Windows）に依存しない
- 最終形態（ファイルや表示）の汎用性，軽さと高速性．

このような要請を満たすため，XY-pic または TikZ（cf. §3.3）によって曲線を表現した TEX のテキストを Risa/Asir で作成し，最終形態を PDF ファイルとして，Bézier 曲線の描画機能を用いることとした．

2.1 Bézier 曲線

n 次の Bézier 曲線とは，$n+1$ 個の制御点 $B_0, B_1, \ldots, B_n$ で定まる曲線

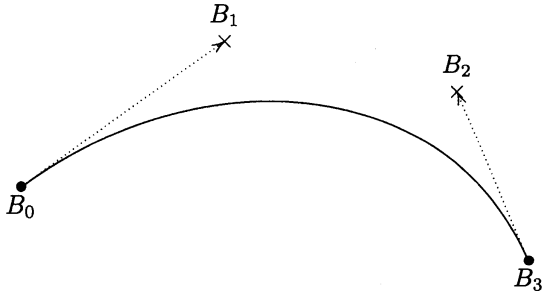
$$P(B_0, B_1, \ldots, B_n; t) = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} B_i \qquad (0 \leq t \leq 1)$$

をいう．この曲線は $B_0 = P(0)$ を始点，$B_n = P(1)$ を終点とするが，途中の制御点 $B_1, \ldots, B_{n-1}$ を一般には通らない．

コンピュータでの曲線描画には4つの制御点 B_0, B_1, B_2, B_3 で定まる3次の Bézier 曲線

$$\begin{aligned} P(t) &= B_0(1-t)^3 + 3B_1t(1-t)^2 + 3B_2t^2(1-t) + B_3t^3 \qquad (0 \le t \le 1) \\ &= (-B_0 + 3B_1 - 3B_2 + B_3)t^3 + (3B_0 - 6B_1 + 3B_2)t^2 + (-3B_0 + 3B_1)t + B_0 \end{aligned}$$

を用いることが多い.



この3次の Bézier 曲線は，始点 B_0 から $\overrightarrow{B_0B_1}$ に接する方向に出て，B_3 には $\overrightarrow{B_2B_3}$ に接する方向に入る.

$$P(B_0, B_1, \ldots, B_n; t) = P(B_0, B_1, \ldots, B_{n-1}; t)(1-t) + P(B_1, B_2, \ldots, B_n; t)t$$

より

$$\begin{aligned} P(t) &= P(P(B_0, B_1, B_2; t), P(B_1, B_2, B_3; t); t) \\ &= P(P(P(B_0, B_1; t), P(B_1, B_2; t); t), P(P(B_1, B_2; t), P(B_2, B_3; t); t); t) \end{aligned}$$

となるが，$P(C, C'; t)$ は線分 CC' を $t : (1-t)$ に内分する点であることに注意すれば，$P(t)$ の図形的な意味（作図法）が分かる．線分 B_0B_1, B_1B_2, B_2B_3 の $t : (1-t)$ の内分点を順に C_0, C_1, C_2 とおき，線分 C_0C_1, C_1C_2 の $t : (1-t)$ の内分点を D_0, D_1 とすると，$P(t)$ は線分 D_0D_1 を $t : (1-t)$ に内分する点である.

なお，Bézier 曲線はアフィン変換で不変なこと，また真の円弧を描けないことを注意しておく.

2.2 通過点を指定した滑らかな曲線

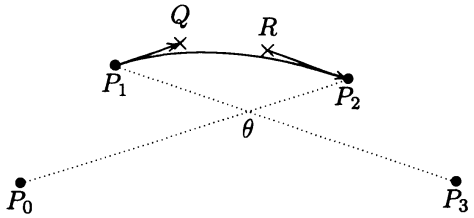
曲線が $(x(t), y(t))$ $(t \in [a, b])$ のようにパラメータ表示されていたとする．パラメータ t の動く範囲 $[a, b]$ を N 等分し，曲線上に点 $P_i = (x(a + \frac{i}{N}(b-a)), y(a + \frac{i}{N}(b-a)))$ を取る．点 $P_0, P_1, \ldots, P_N$ を順に線分結んで出来る折れ線は，元の曲線を近似している．元の曲線の代わりにこの折れ線を用いるとすると，N を十分大きく取る必要があり，また曲線を拡大して表示する場合は，より N を大きく取らないと曲線からのずれが目立ってしまう．一方 N を大きくするほど，曲線を描くためのデータ量が増えてしまう.

そこで，曲線の通過点 $P_0, P_1, \ldots, P_N$ を適当な曲線で結んで，元の曲線を近似することを考える．近似曲線は C^1 級で，P_i から P_{i+1} を結ぶ部分の曲線を3次 Bézier 曲線にとることにする．そこで，P_i と P_{i+1} の間の2つの制御点を如何に取るか，ということが問題となる.

デフォルトでは $P_0 \to P_1 \to P_2 \to P_3$ のように点を通過する曲線の P_1 と P_2 を結ぶ3次 Bézier 曲線を次のように制御点 Q, R を選んで定めることにした.

- P_1 における接線は $\overrightarrow{P_0P_2}$ に並行に，P_2 における接線は $\overrightarrow{P_1P_3}$ に並行になるようにする.
  なおこれによって多くの通過点を順に指定して描いた曲線が C^1 級になることが保証される.
- よって $\overrightarrow{P_1Q} = c_1\overrightarrow{P_0P_2}$, $\overrightarrow{RP_2} = c_2\overrightarrow{P_1P_3}$ と表せるが，$c_1 = c_2 = c$ と同じ値にとる.
- $\frac{\overline{P_1Q} + \overline{P_2R}}{\overline{P_1P_2}}$ の値は，$\overrightarrow{P_1Q}$ と $\overrightarrow{RP_2}$ のなす角 θ $(0 \le \theta \le \pi)$ のみに依る.
- P_0, P_1, P_2, P_3 が同一円周上に等間隔で並んだとき，3次 Béziere 曲線はその円の P_1 と P_2 を結ぶ円弧を近似するように c を定める．すなわち P_1 と P_2 を結ぶ3次 Béziere 曲線上の中点（$t = \frac{1}{2}$ に対応する点）が円弧上に位置するように定める.

具体的には以下のようになる.



$$\overrightarrow{P_1Q} = c\overrightarrow{P_0P_2}$$
$$\overrightarrow{RP_2} = c\overrightarrow{P_1P_3}$$
$$\frac{\overline{P_1Q}+\overline{P_2R}}{\overline{P_1P_2}} = \frac{2}{3(1+\sin\frac{\theta}{2})}$$
一直線上に順に並ぶ $\Rightarrow$ $\theta = \pi$

$P_0, P_1, \ldots$ が円周上に等間隔に並んでいるときは，円弧を近似していて，中心からの距離の誤差は

隣り合う 2 点の中心角 $< \frac{\pi}{2} \Rightarrow$ 誤差 $< 0.0068\%$，隣り合う 2 点の中心角 $< \frac{\pi}{8} \Rightarrow$ 誤差 $< 0.0001\%$

Q, R を求める計算は以下のようにできる.

$$\cos\theta = \frac{(\overrightarrow{P_0P_2}, \overrightarrow{P_3P_1})}{\overline{P_0P_2}\cdot\overline{P_1P_3}}, \quad \sin\frac{\theta}{2} = \sqrt{\frac{1-\cos\theta}{2}},$$

より

$$c := \frac{4\overline{P_1P_2}}{3(\overline{P_0P_2}+\overline{P_1P_3})} \frac{1}{1+\sqrt{\frac{1+\frac{(\overrightarrow{P_0P_2},\overrightarrow{P_1P_3})}{\overline{P_0P_2}\cdot\overline{P_1P_3}}}{2}}}$$

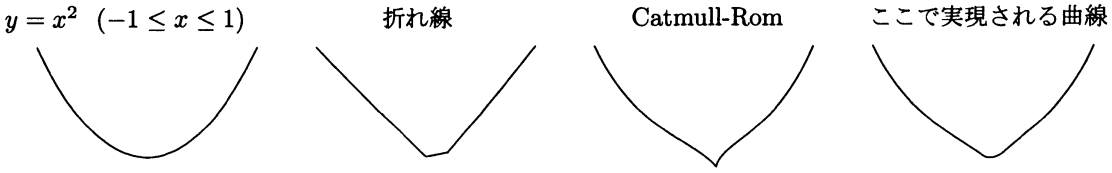
とおくと

$$\overrightarrow{P_1Q} = c\overrightarrow{P_0P_2}, \quad \overrightarrow{P_2R} = c\overrightarrow{P_3P_1}.$$

注意. 調べた範囲では，通過点を指定したときのこのような曲線の描画法は発見できなかった．似たもので Catmull-Rom スプライン曲線があって，それは上で $c = \frac{1}{6}$ と固定したものになっている．これは，上述の曲線において $\overline{P_0P_2} + \overline{P_1P_3} = 4\overline{P_1P_2}$ かつ $\theta = \pi$，すなわち P_0, P_1, P_2, P_3 が円周上に等間隔で並んでいて，円弧の円周角が小さいとき（4 点が直線上に等間隔に並んでいるのに近いとき）に対応している．

曲線上の点を不揃いやランダムに取ると Catmull-Rom スプライン曲線は尖ったりループが出来たりして特異点が生じることがあるが，ここで実現されている曲線描画ではそれを避けるようになっている．

たとえば，$y = x^2 \quad (-1 \le 1 \le 1)$ という放物線の例で見てみる．曲線上の点として $x = -2, -1, 0, 0.2, 1, 2$ における放物線上の 6 点を取り，最初の点と最後の点を制御点として 4 点を結ぶ曲線の描画を見てみると，以下のようになる.



2.3 xylines( )

現在具体的に実現されている Risa/Asir の関数 xylines( ) の仕様を記す.

xylines([[x_1,y_1,s_1],[x_2,y_2,s_2],...]|opt=t,close=1,curve=1,ratio=c,verb=1,scale=r,
dviout=1)

:: XY-pic/TikZ で s_j を (x_j, y_j) に置き，$(x_1, y_1), (x_2, y_2), \ldots$ を線で結ぶ

- close=1 : 最後の点を最初の点と結んで，多角形または滑らかな閉曲線を作る.
- close=-1 : 曲線で結ぶ場合に，最初と最後の点は制御点とし，残りの点を滑らかに結ぶ.
- curve=1 : 曲線（Bézier 曲線を使う）で滑らかに結ぶ.

  P_0, P_1, P_2, P_3 を通る曲線で P_1 と P_2 を結ぶには，前節で述べた c によって

$$\overrightarrow{P_1Q} = c\overrightarrow{P_0P_2},\quad \overrightarrow{P_2R} = c\overrightarrow{P_3P_1}$$

  と与えられる点 Q, R を途中の制御点として定まる 3 次 Bézier 曲線を用いる

  P_0 や P_3 が存在しないときは Q または R を省いて 2 次の Bézier 曲線（または線分）を用いる.

  正 n 角形の頂点を指定し，close=1 を指定すると，デフォルトでは外接円に近い曲線になる.

  実際，外接円の半径を 1 とすると，得られた曲線の中心からの距離と 1 との差の最大値は，$n = 3$ のとき 0.00039, $n = 4$ のとき 0.000068, $n = 6$ のとき 0.000006, $n = 8$ のとき $0.000001 = 10^{-6}$ という程度の小さな値になる.

  なお，c を ratio=c で指定することもできる（$c = \frac{1}{6}$ ならば Catmull-Rom スプライン曲線）.
- curve=2 : B-spline 曲線で結ぶ（途中の点は制御点で通らない）.

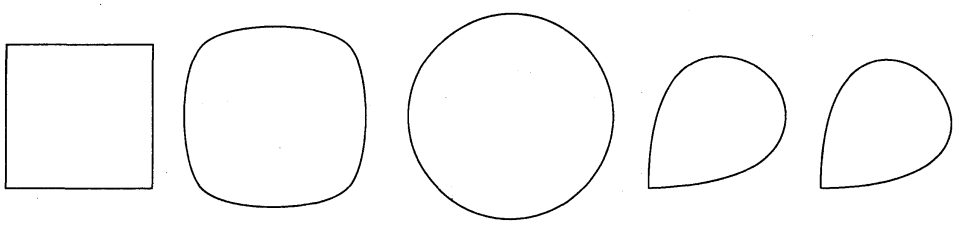
  このときは s_j は最初と最後のみ指定可能．制御点が 5 点以上の場合は Ti*k*Z ではサポートされない.
- $s_1, s_2, \ldots$ は省略可能．また s_j がないときは，[x_j,y_j] はリストでなくてベクトルでもよい.
- 通過点のデータに 0 があるときは，その前後で直線または曲線を切る.
- scale=r : Xy-pic/Ti*k*Z の座標に直すときに r 倍する.
- scale=[r_1,r_2] : Xy-pic/Ti*k*Z の座標に直すときに x 座標を r_1 倍，y 座標を r_2 倍する.

  Xy-pic は座標が mm 単位，Ti*k*Z は cm 単位である.
- opt=t : 線種の指定（マニュアル [O2, os_muldif.pdf] 参照）.
- dviout=1 : 画面で表示する.
- verb=1 : 通過点を •，制御点を × で表示する.

以下、Risa/Asir 上で実行した結果を記す.

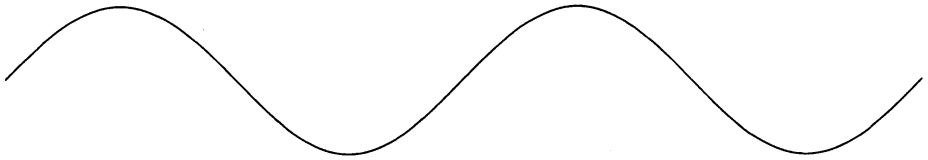
```
[0] L=[[0,0],[20,0],[20,20],[0,20]]$
[1] L0=os_md.xylines(L|close=1);
{(0,0) \ar@{-} (20,0)};
{(20,0) \ar@{-} (20,20)};
{(20,20) \ar@{-} (0,20)};
{(0,20) \ar@{-} (0,0)};
[2] L1=os_md.xylines(L|close=1,curve=1,ratio=1/6)$
[3] L2=os_md.xylines(L|close=1,curve=1)$
[4] L3=os_md.xylines(L|close=1,curve=2)$
[5] L4=os_md.xybezier(append(L,[[0,0]]))$
```

として，L0, L1, L2, L3, L4 を順に表示すると

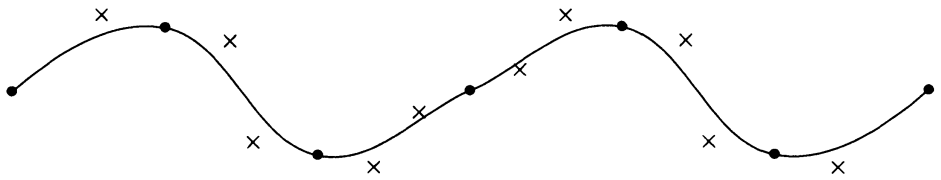


以下はXy-pic の例であるが，TikZ の場合は座標のスケールの違いから，scale=0.1 を指定するとサイズが同じになる．

```
[6] Pi=3.14159265$
[7] for(V=[],I=0;I<=48;I++) V=cons([10*Pi*I/12,dsin(Pi*I/12)*10],V);
[8] os_md.xyproc(os_md.xylines(V|curve=1)|dviout=1)$
```



```
[9] for(V=[],I=0;I<=6;I++) V=cons([10*Pi*I*2/3,dsin(Pi*2*I/3)*10],V);
[10] os_md.xylines(V|curve=1,dviout=1,verb=1);
```



上の [8] では $y=\sin x$ $(0 \le x \le 4\pi)$ のグラフを x の値で 48 等分した曲線上の点を繋いで曲線を描いている．[10] では，6 等分に減らした曲線上の点 • と，その 7 点を元に定めた制御点 × と，それらを元に描いた曲線を示した．

2.4 パラメータで表された曲線の描画

パラメータ表示をもつ曲線に特異点があったり，不連続点があっても，その曲線がうまく描けるようにしたい．曲率が小さくなる点や特異点の近くでは，曲線上の点を密に取ることでそれを実現する．具体的には，折れ線で繋いだときの傾きの変化が一定以上の角度になるときは，その点の両側での細分を再帰的に行う．ある程度細分を続けても曲線上の 2 点間の距離が縮まらないときに不連続点と見なせばよい．

xygraph(f,n,[t_1,t_2],[x_1,x_2],[y_1,y_2]|**opt=**t,**rev=1**,**ax=**[x_0,y_0,s,t,u],**axopt=**[h,w,o,z],
scale=r,**ratio=**c,**raw=1**,**org=**[x_0,y_0],**pt=**[p_1,p_2,...],**verb=1**,**para=1**,**prec=**v,**dviout=1**)
:: 変域の n 等分点での値で関数のグラフを描く（$t_1 \le \mathtt{x} \le t_2$，$(x_1,y_1)$-$(x_2,y_2)$ は表示窓の範囲）

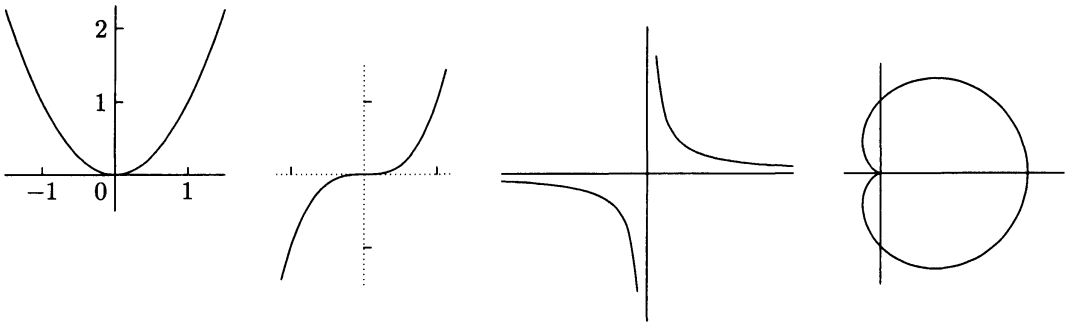
- f は [f_1,f_2] の形のパラメータ表示でもよい．
- f や前項の f_1, f_2 は有理関数や $\sin x$ などの初等関数の他，**mydeval( )** が解釈できる**リスト形式関数** (cf. §3.1) でもよい．よってユーザが定義した関数や，Γ 関数などのように **pari()** 経由の関数も可．
- f_1 が有理式で f_2 がリスト形式関数のときは，**para=1** としてパラメータ表示であることを明示する必要がある．
- 変数を **x** でなくて t とするときは，[t_1,t_2] でなくて [t,t_1,t_2] と指定する．
- **rev=1** : $x=f(y)$ のグラフとみなす．
- (x_1,y_1) と (x_2,y_2) で定まる窓から外れる点は除く．
- $n=0$ と指定したときは，$n=32$ とみなす．
- $n<0$ のときは，$|n|$ 等分した区間の両側の 1 区間外に制御点をとる．

- n 等分点でなくて点を直接指定するときは，[t_1,t_2] でなくて [t_1,t_2,...,t_m] のように指定する（なお，$t_1,\ldots,t_m$ は単調増加な実数列．さらに最初を実数でなく，変数の指定としてもよい）．
  n が負の時は，最初と最後の点 t_1, t_m は制御点とする．
- prec=[v_1,v_2,v_3]：折れ線で繋ぐと角度変化が v_2° 以上の分割点の両側を 2 つに細分することを最大 v_1 ステップ行い，さらに $v_3>0$ のとき，窓の対角線の長さの v_3 分の 1 以上の跳びを不連続点とみなす．
  - $v_2=1$ のときは $v_2=30$ と解釈され，$1<v_2<10$ のときは $v_2=10$ と解釈され，$v_2>120$ のときは $v_2=120$ と解釈される．
  - $v_3=0$ のときは，不連続点はないものと解釈する．
  - $v_3=1$ のときは $v_3=8$ と解釈され，$v_3>16$ のときは $v_3=16$ と解釈される．
  - prec=[v_1,v_2] は，prec=[v_1,v_2,0] と解釈される．
  - prec=v_1 は，$v_1=0$ のとき prec=[4,30,0] と，$v_1>0$ のとき prec=[v_1,30,0] と，$v_1<0$ のとき prec=[$|v_1|$,30,8] と解釈される．
- opt=t：線種の指定など（xylines( ) に渡される）．
- ratio=c：Bézier 曲線を使うときのパラメータ（xylines( ) に渡される）．
- ax=[x_0,y_0]：(x_0,y_0) を原点として x 軸と y 軸を窓内に描く．
  窓内に現れない軸は描かないので，それを利用して一方の軸のみの描画ができる．
- ax=[x_0,y_0,s,t]：s が正数なら，x 軸に原点から s 毎に目盛りをつける．
  目盛りをつける位置を指定するときは s = [s_1,s_2,...] と原点からの x 座標の位置を書く．
  s_j =[$s_{j,0}$,$s_{j,1}$] となっていると $s_{j,0}$ の位置に目盛りをつけて，そこに $s_{j,1}$ を書く．
  t で同様に y 軸の目盛りの指定ができる
- ax=[x_0,y_0,s,t,u]：s, t が数字で u が 1 と 2 とすると，目盛りが ax=[x_0,y_0,s,t] に従ってつけられ，u に従ってそこに数字が書かれる．$u=0$ のとき k 番目の位置の x 軸には $ks+x_0$ が y 軸には $kt+y_0$ が書かれ，$u=1$ のとき k 番目の位置の x 軸には ks が y 軸には kt が書かれ，$u=2$ のとき k 番目の位置に k が書かれる．
- axopt=z：z が文字列の時は x 軸と y 軸の線種指定文字列（デフォルトは実線）
- axopt=h：h が 0 以外の数字の時は，目盛りの軸からの長さを与える．
  値が負の時は軸から負方向（x 軸なら下，y 軸なら左）の目盛り．
- axopt=[h,w,o,z]：軸や目盛りの描き方の指定（マニュアル参照）
- pt=[p_1,p_2,...]：（複数の）点を明示する，または線を描く．仕様は xy2graph( ) の同様なオプションに習う．
- scale=r：Xy-pic/TikZ の座標に直すときに r 倍する．
- scale=[r_1,r_2]：Xy-pic/TikZ の座標に直すときに x 座標を r_1 倍，y 座標を r_2 倍する．
- org=[x_0,y_0]：Xy-pic/TikZ の座標に直すときに (x_0,y_0) をXy-pic/TikZ の原点に対応させる．
  上の両者が指定されたとき，座標 (x,y) は Xy-pic/TikZ の座標 $(r_1(x-x_0),r_2(y-y_0))$ に変換される．
- raw=1：通過点のリストを返す (他のオプションは無視される)．
  これは xylines( ) のデータとなり，また ptaffine( ) でアフィン変換が計算できる．
  なお，範囲外の点は 0 というデータに変換される．
- f =[0,0] のときはグラフを描かない．座標軸のみを描くときに用いる．
- err=c：関数の定義域を外れるエラーが生じるときは，$c=1$, $c=-1$ などとしてこのオプションを指定すると，エラーが解消される可能性がある（c は絶対値があまり大きくない実数）．

- verb=1：グラフ上の通過点を ●，制御点を × で表示する．
- dviout=1：グラフを画面表示する．

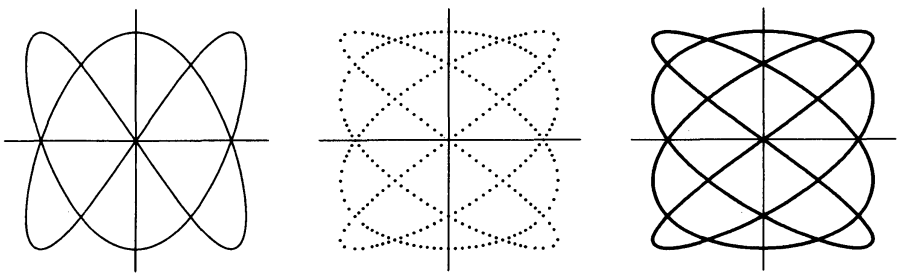
Xy-pic のときは

```
[0] os_md.xygraph(x^2,0,[-1.5,1.5],[-1.5,1.5],[-0.5,2.3]|dviout=1,ax
    =[0,0,1,1,1],scale=10);
[1] os_md.xygraph(x^3,0,[-1.2,1.2],[-1.2,1.2],[-1.5,1.5]|dviout=1,ax
    =[0,0,1,1],axopt="@{.}",scale=10);
[2] os_md.xygraph(1/x,0,[-3,3],[-3,3],[-3,3]|dviout=1,ax=[0,0],scale=5);
[3] F=[(1+cos(x))*cos(x),(1+cos(x))*sin(x)]$
[4] os_md.xygraph(F,0,[-@pi,@pi],[-0.5,2.5],[-1.5,1.5]|dviout=1,scale=10,
    ax=[0,0]);
```



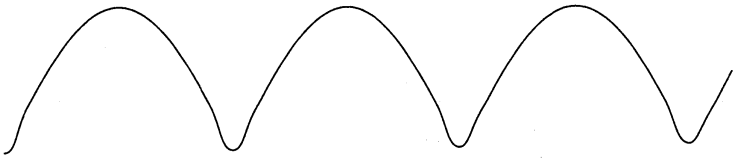

```
[5] F1=[sin(2*x),sin(3*x)]$
[6] os_md.xygraph(F1,-48,[-@pi,@pi],[-1.2,1.2],[-1.2,1.2]|dviout=1,scale=15,
    ax=[0,0])$
[7] F2=[sin(4*x),sin(3*x)]$
[8] os_md.xygraph(F2,-48,[-@pi,@pi],[-1.2,1.2],[-1.2,1.2]|dviout=1,scale=15,
    ax=[0,0],opt="~*=<3pt>{.}")$
[9] os_md.xygraph(F2,-48,[-@pi,@pi],[-1.2,1.2],[-1.2,1.2]|dviout=1,scale=15,
    ax=[0,0],opt="~*={.}")$
```

TikZ の場合は（前者は mm 単位，後者は cm 単位なので）scale= の値を十分の一に，[9] の opt="~*=<3pt>{.}" を opt="dotted" に，[10] の opt="~*={.}" を opt="very thick" などに変更．

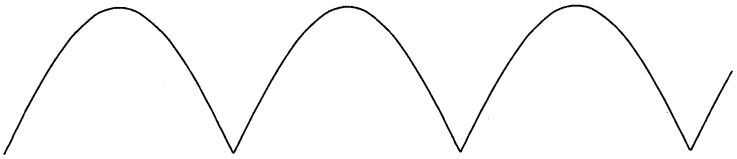


$y = |\sin x|$ $(0 \leq x \leq 10)$ のような微分不可能点をもつ曲線を描くにはオプション prec を用いる．

```
[10] F=[u,[v,dsin,x],[u,os_md.abs,v]]$
[11] os_md.xygraph(F,-32,[0,10],[0,10],[0,1]|dviout=1,scale=[15,25])$
[12] os_md.xygraph(F,-32,[0,10],[0,10],[0,1]|dviout=1,scale=[15,25],prec=0)$
```



微分可能でない点の近くでは不正確であるが，[12] の prec=0 の指定で以下のようにより正確になる：

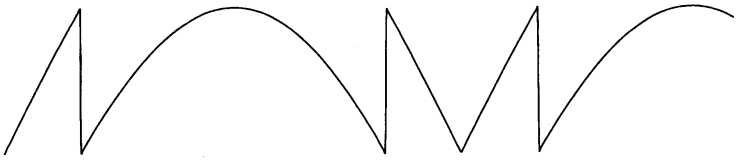


$y = |2\sin x| - [|2\sin x|]$ $(0 \le x \le 5)$ のような不連続点をもつ曲線を描くにもオプション prec を用いる．この不連続関数を描画してみよう．なお $[t]$ は t を越えない最大整数を表す．

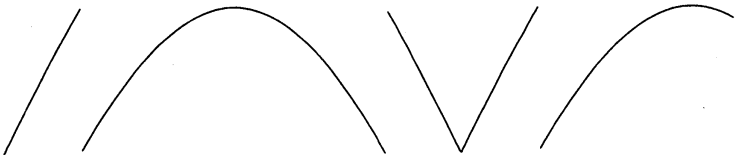
```
[13] G=[u,[v,dsin,x],[w,os_md.abs,2*v],[z,dfloor,w],[u,0,-z+w]]$
[14] os_md.xygraph(G,-32,[0,5],[0,5],[0,1]|dviout=1,scale=20)$
[15] os_md.xygraph(G,-32,[0,5],[0,5],[0,1]|dviout=1,scale=20,prec=0)$
[16] os_md.xygraph(G,-32,[0,5],[0,5],[0,1]|dviout=1,scale=20,prec=[4,0,1])$
```

の実行結果は以下のようになる．すなわちオプション prec を指定しない場合 [14] は

となる．[15] で prec=0 とすると特異点を自動判別し，その近くでより正確な描画

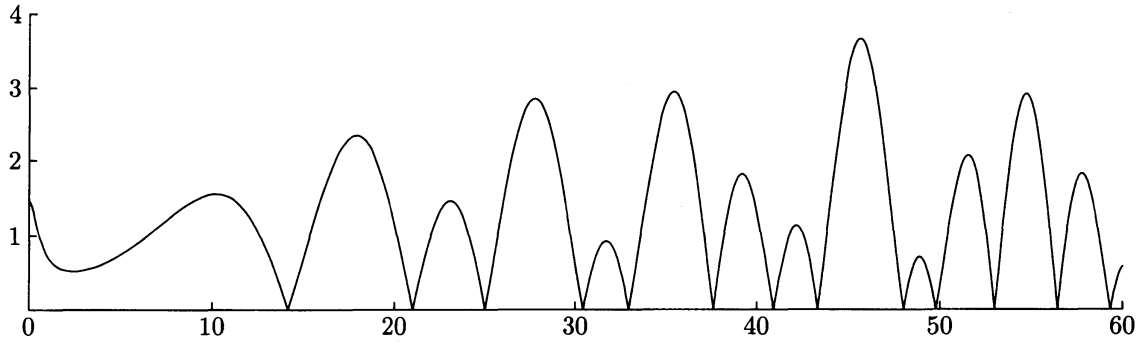


となる．不連続点で繋がってしまうが，prec=-4 または prec=[4,0,1] とした [16] では以下のようになる．



```
[17] H=[w,[z,os_md.zeta,1/2+@i*x],[w,os_md.abs,z]]$
[18] os_md.xygraph(H,-64,[0,60],[0,60],[0,4]|dviout=1,scale=[2.5,10],prec=6,
     ax=[0,0,10,1,1])$
```

とすると（1 秒もかからずに）以下の $|\zeta(\frac{1}{2} + x\sqrt{-1})|$ $(0 \le x \le 60)$ のグラフが得られる：



3 2変数関数の描画

3.1 リスト形式関数

$z = \exp(-x)(\sin x + \cos y)$ のような2変数関数の3Dグラフを描画しようとすると，軸を回転させて 100×100 のメッシュの格子点での関数の値を求めるようなことが必要になる．そのため `deval(subst(sin(x),x,1.234)` などとすると，`sin(1.234)` という不定元が生成され，最終的には $100 \times 100 = 10000$ 個余りの不定元が生成されることになるが，生成された不定元を消す方法は提供されていない．結果として，プログラムが止まってしまうか，動作が異常に遅くなる．

これを避けるために，このような関数は**リスト形式の関数**に変換して上のような不定元の生成が起きないようにしている．なお，リスト形式の関数とは，以下の `mydeval( )` の引数となり得るリストのことである．

`mydeval([`r`,[`x_1`,`f_1`,`v_1`],[`x_2`,`f_2`,`v_2`],...])`
:: `os_md.mydeval(subst([`r`,[`x_2`,`f_2`,`v_2`],...],`x_1`,`f_1`(os_md.mydeval(`v_1`))))` を返す
`os_md.mydeval([r])` は `deval(r)` または `map(deval(r))` を返す．
通常の $\exp(-x)(\sin x + \cos y)$ のような関数をリスト形式の関数に変換するのには `f2df( )` を用いる．

```
[0] os_md.f2df(exp(-x)*(sin(x)+cos(y)));
[z__2*z__+z__2*z__1,[z__,dsin,x],[z__1,dcos,y],[z__2,dexp,-x]]
```

3.2 xy2graph( )

`xy2graph(`f`,`n`,[`x_1`,`x_2`],[`y_1`,`y_2`],[`h_1`,`h_2`],`α`,`β`|opt=`t`,scale=`r`,view=`h`,raw=1,trans=1,dev=`m`,`
`acc=`k`,ax=[`z_1`,`z_2`,`t`],org=[`x_0`,`y_0`,`z_0`],pt=[`p_1`,`p_2`,...], prec=`v`,title=`s`,dviout=1)`
:: x, y 変数の区間を n 等分して曲面 $z = f(x, y)$ の3Dグラフを描く

- x 軸の正方向から y 軸の正方向に α 度だけ回転した（無限）遠方から β 度（$-90 < \beta < 90$）の角度で見下ろす方向に見た曲面 $z = f(x, y)$ $(x_1 \le x \le x_2,\ y_1 \le y \le y_2)$ の3Dグラフを描く．なお，投影した高さ方向の座標（y 座標）が $[h_1, h_2]$ に入る範囲のみ描く．
  より具体的には3次元の点 (x, y, z) は以下のような平面の点に投影される．

$$(x, y, z) \mapsto (-x\sin\alpha^\circ + y\cos\alpha^\circ,\ z\cos\beta^\circ - x\cos\alpha^\circ\sin\beta^\circ - y\sin\alpha^\circ\sin\beta^\circ)$$

  $\alpha = 0$ のときは $\alpha = 60$, $\beta = 0$ のときは $\beta = 15$ と解釈される．
  α は0であるか，または90の整数倍とは5以上離れていることが必要．

通常は h_1 を十分小さく，h_2 を十分大きく取っておけばよい（指定した範囲の曲面全体の表示）．

- 曲面上で x 座標が定数，あるいは y 座標が定数で定まる曲線を（曲面で隠れる部分，すなわち隠線を消して）描くことで曲面を表す．定数は座標の n 等分点と定める．
  $n<0$ のときも $|n|$ 等分点をとるが，その一つ外側を制御点にとる．
- f は有理関数や $\sin x$ などの初等関数に限らず，`mydeval( )` が解釈できるリスト形式関数ならばよい．ユーザが定義した関数でもよい．
- `cpx=1,2,3` を指定するか f に `@i` が含まれていれば，`mydeval()` でなくて `myeval( )` が使われる．
- f が 1 変数 z の有理関数ならば，$z=x+yi$ 変数の複素関数と考え $z=|f(x+iy)|$ のグラフを描く．この場合 f が有理関数であったなら，リスト形式関数

```
[w,[z,0,x+y@i],[w,os_md.abs,f]]
```

で置き換えられる．なお，複素変数の $\sin z$, $\cos z$, $\tan z$, $\mathrm{atan}\, z$, $\mathrm{asin}\, z$, $\mathrm{acos}\, z$, $\sinh z$, $\cosh z$, $\tanh z$, $\exp z$, $\log z$, z^w や，それらを含む合成関数や有理関数もサポートされている．

f が `sin(z^2)+1` のときは

```
[w,[z,0,x+y*@i],[w,os_md.abs,[z__+1,[z__,os_md.sin,z^2]]]]
```

のように置き換えられる．
$|\Gamma(z)|$ のグラフを描くときは，たとえば f を以下のように取ればよい．

```
[w,[z,0,x+y*@i],[u,os_md.gamma,z],[w,os_md.abs,u]]
```

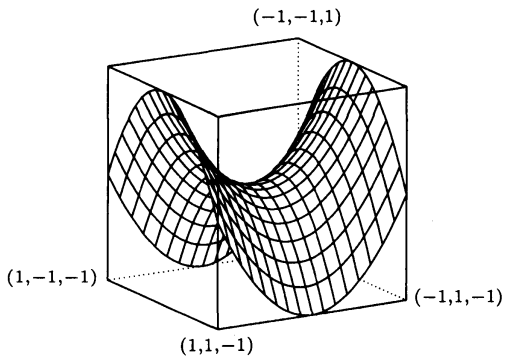
このときタイトルの関数は `title="\\Gamma(z)"` というオプションで表示できる．

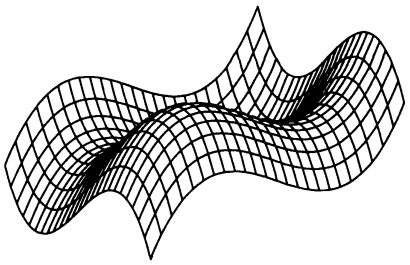
- `scale=`r：表示するためにXy-pic/Ti*k*Z の座標に直すときに r 倍する．
- `scale=[`r_1`,`r_2`]`：元の曲面の z 座標を $\frac{r_2}{r_1}$ 倍したものを平面に投影したあと，表示するためXy-pic/Ti*k*Z の座標に直すときに座標の単位を r_1 倍する．
- `scale=[`r_1`,`r_2`,`r_3`]`：元の曲面の z 座標を $\frac{r_2}{r_1}$ 倍，y 座標を $\frac{r_3}{r_1}$ したものを平面に投影したあと，表示するためXy-pic/Ti*k*Z の座標に直すときに座標の単位を r_1 倍する．
- `org=[`x_0`,`y_0`,`z_0`]`：元の座標の (x_0,y_0,z_0) をXy-pic/Ti*k*Z の座標での原点にする（デフォルトでは原点が原点に対応）．
- $|n|$ が大きいと TEX のソース・ファイルが長大になり，ソース・ファイルから DVI ファイルや PDF ファイルへの変換に時間がかかることがあるので注意（cf. §3.3）．
  $n=-16$ がデフォルト（$n=0$ とするとデフォルト値と解釈される）．
- `view=1`：陰線消去を行わない．　`view=-1`：陰線は点線で表示する．
- `raw=1`：通過点のリストを返す．
- `dev=`m：陰線消去のためのメッシュを，x 変数，y 変数とも $m\times|n|$ 等分したものとする（デフォルトは $m=16$）．ただし `dev=[`m_1`,`m_2`]` とすると，m_2 は出力される x 座標のメッシュの細かさ，m_1 は曲線間にとるメッシュの細かさ，と別に指定することができる．
- $|n|\times m$ を増やすと処理時間が増える．特に f が多項式や有理関数でなくて三角関数や指数関数，対数関数，べき関数などを含むときは注意．なお処理時間は，ほぼ $|n|^2\times m^2$ に比例．
- `acc=`k：描く曲線の等分点の個数を約 k 倍にする（k は実数．曲線の本数は不変）．$k=2$ なら n を 2 倍にして `dev` を半分にした場合の曲線を一本おきに描くことにほぼ等しい（結果のファイル・サイズは，$n^2\times k$ にほぼ比例）．

- err=c：有理関数の分母が 0 になるなどの，関数の定義域を外れるエラーが生じるときは，$c=1, -1$ などとすると，エラーが解消される可能性が大きい（c は絶対値があまり大きくない実数）.
- prec=v：xygraph( ) の同様のパラメータと同じ.
- ax=[z_1,z_2]：x, y, z の座標が (x_2, y_2, z_1), (x_1, y_1, z_2) を対角線の頂点とする直方体の枠を書く，
- ax=[z_1,z_2,t] ：上に加え，一部の頂点の座標を入れる（詳しくはマニュアル参照）.
- 複素変数の実数値関数のときに座標を入れる形式は，$(x, y, z) = (1, 2, 3)$ の場合に
  - cpx=1：$(1+2i, 3)$（デフォルト）　• cpx=2：$(1+2\sqrt{-1}, 3)$　• cpx=3：$(1, 2, 3)$
- title=s：画面表示するとき，関数名が s で表示される（s は TEX の数式モードでの文字列 ）. 関数がリスト形式のときの関数名表示に役立つ.
- pt=[p_1,p_2,...]：（複数の）点を明示する，あるいは線を引く（詳しくはマニュアル参照）.
- dviout=1, 2, 3：画面表示する. 関数式 ($k=2$)，さらに始点の角度と倍率（$k=2$）も表示する
  k が負の時は，dviout=|k| に対応する TEX のコードがリスト形式で出力される.
  リストの最後の成分は xyproc( ) によってグラフ描画の TEX のソースとなる. その前の成分は関数式などを表す TEX のソース，trans=1 を指定した場合はその結果が先頭につく.
- trans=1：もとの (x, y, z) 座標に対応するXy-pic/TikZ の座標の対応 $[X, Y]$ を返す（詳しくはマニュアル参照）.

```
[0] os_md.xy2graph(x^2-y^2,0,[-1,1],[-1,1],[-2,2],0,0|ax=[-1,1,-6],scale=15,
dev=64,dviout=3)$
[1] os_md.xy2graph(-x^3-y^3,-24,[-1,1],[-1,1],[-2,2],60,-35|scale=20,dev=64,
dviout=2)$
```

$z = x^2 - y^2$ $(-1 \le x \le 1, -1 \le y \le 1)$　　$z = -x^3 - y^3$ $(-1 \le x \le 1, -1 \le y \le 1)$



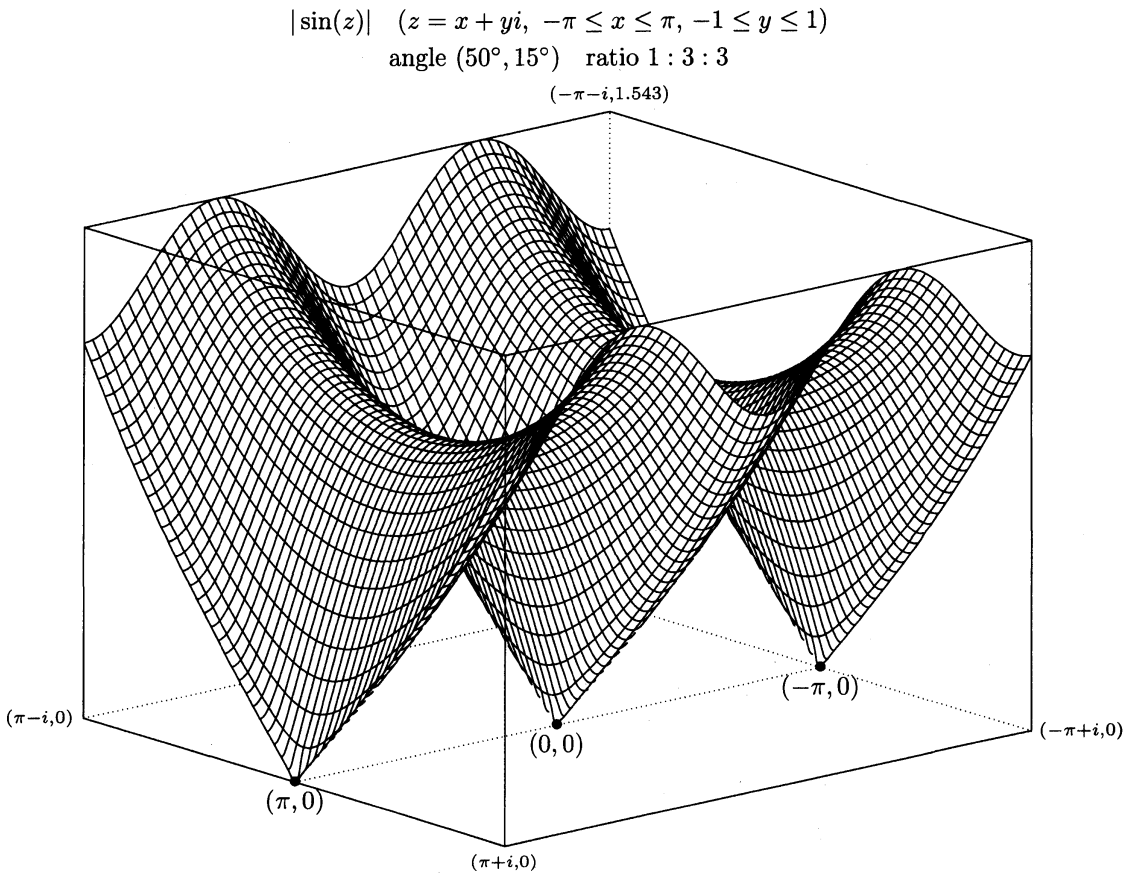


上はXy-pic のときの [0] と [1] の結果で，TikZ を用いる場合は scale の値を $\frac{1}{10}$ 倍にする.

```
[2] S0=[[3.1416,0,0],0,"*+!U{(\\pi,0,0)}"]$
[3] S1=[[0,0,0],0,"*+!U{(0,0,0)}"]$
[4] S2=[[-3.1416,0,0],0,"*+!U{(-\\pi,0,0)}"]$
[5] S3=[[3.1416,0,0],[-3.1416,0,0],2]$
[6] os_md.xy2graph(sin(z),-60,[-@pi,@pi],[-1,1],[-5,8],50,0|
    scale=[15,45,45],ax=[0,1.543,-6],dviout=3,pt=[S0,S1,S2,S3])$
```

これは複素変数関数 $|\sin z|$ のグラフの表示であるが，TikZ のときは，たとえば上の [2] は次のようにする.

```
[2] S0=[[[3.1416,0,0],0,1], [1,["below","$(\\pi,0,0)$"]]]$
```



隠線消去の判定にかかる時間が処理スピードを左右している（現時点では，判定の最適化を行っていない）．

Risa/Asir でのコマンド入力 → 「TEX ソース書き出し → DVI ファイルに変換 → PDF ファイルに変換 → 表示」が自動化され，上の例ではコマンドの入力 [6] から上の $|\sin(z)|$ のグラフの表示までに，30 秒程度かかる（2014 年における一般的なパソコン）．なお，2 変数の有理関数のグラフなら 10 秒程度で終わる．

3.3 補足

XY-pic を用いて PDF ファイルを作成するときは，`\usepackage[pdf,all]{xy}` などと指定して `dvipdfmx` などを用いるように指定する（詳しくはマニュアル [O2, os_muldif.pdf] を参照）．サイズの小さな良質の PDF ファイルが，短い時間で作成できる．また，この指定では複雑な画像でも処理が可能．

描画のカラー化やパターンでの指定領域の塗りつぶしなどは，`\usepackage{TikZ}` などと指定して TikZ を用いることで対応できる（XY-pic との併用も可能．[O2, os_muldif.pdf]）．

参考文献

[O1] 大島利雄，数式処理による数学研究とプレゼンテーション，数式処理とその周辺分野の研究，数理解析研究所講究録 **1907** (2014), 97–109.

[O2] 大島利雄，os_muldif.rr および os_muldif.pdf, a library for computer algebra Risa/Asir, 2007–2015, ftp://akagi.ms.u-tokyo.ac.jp/pub/math/muldif/