Maximaでお絵描き}

東芝インフォメーションシステムズ株式会社
横田博史

平成19年3月6日(火)

まえがき

数式処理システム Maxima は 1960 年代に開発された MACSYMA を Common Lisp 上に移植したソフトウエアです.Mathematica や Maple 等と比較して, 古色蒼然とした面も否定出来ませんが, 見掛け以上に強力なソフトウエアであり,GPL の下で配布される数式処理システムの中では非常に魅力的な汎用の数式処理ソフトウエアの一つです.

この小冊子は本来は「はじめての Maxima」から抜き出した簡易マニュアルの一部とする予定でした. 但し, 分量が増えた事もあって, 独立させたのが実状です.

時間的な制約もあって, まとまりが何となく良くありませんが, 今後, 改善して行きたいと思っています.

平成 19 年 3 月 3 日 (土) 宇佐子の雛祭　　横田博史

目 次

第1章　Maximaのグラフ表示

1.1　グラフ表示の概要

1.1.1　表示アプリケーションについて

Maxima でグラフ表示を行う場合, 外部のアプリケーションを利用します. Maxima で利用可能な外部アプリケーションで, 代表的なものとしては, gnuplot,openmath と Geomview が挙げられます. gnuplot と openmath は 2D グラフだけではなく,3D グラフの表示も可能ですが, Geomview は三次元グラフ専用です.

gnuplot と openmath を比較すると,3D グラフの場合は openmath の方が綺麗な絵を描き, マウスによる視点変更も容易に行えます. これに対して,gnuplot は Windows 版と UNIX 版で異なります. 先ず,Windows 版ではマウスによる視点の変更が簡単に行え, 曲面も面を張ったもので表示されます.UNIX 版の場合,gnuplot は Maxima とは別個のアプリケーションである為, 個々の環境にインストールされている gnuplot のバージョンに依存します.

先ず,gnuplot のバージョンが 3.8 以降であれば,pm3d を設定する事で比較的綺麗な曲面を描く事が可能です. しかし,UNIX 版の場合,Maxima から描画した絵を直接把持して回転させたり図を拡大する事は出来ません. とは言え,gnuplot から `load 'maxout.gnuplot'` でデータを読込ませて描画した画像に対しては,mouse が on の場合にマウスを使って画像の回転や拡大が可能です. gnuplot の mouse が on かどうかは, show mouse で確認可能で,on でなければ, `set mouse` を実行すれば良いのです.

猶,gnuplot と openmath の画質を比較すると,openmath の方が少し前の Mathematica の様なもので, 全体的に gnuplot よりも綺麗に描きますが,gnuplot の方が何かと機能が高い為, 使い慣れると gnuplot の方が便利でしょう. 又,Maxima 側にも gnuplot の豊富な機能を生かす為の工夫が色々と見られます. その為,gnuplot に慣れておく事が Maxima のグラフ表示を使いこなす上で重要です.

1.1.2　plot2d 函数と plot3d 函数の動作

Maxima のグラフ表示は, 最も機能が高い `plot2d` 函数と `plot3d` を使う事が多くなるでしょう. 基本的に, これらの函数で殆どの事が出来るでしょう.

これらの函数によるグラフ表示の手順について簡単に説明しましょう. 先ず,Maxima 内部で指定された外部プログラム向けのグラフデータを生成します. それから, そのデータをホームディレクトリ上のファイルに出力します. そして最後に外部プログラムにこのファイルを引渡して外部アプリケーションがグラフを表示する仕組みになっています. 猶, このグラフデータは数値データで

あって, 数式ではありません. 又, 函数のオプションや大域変数 plot_options に従った設定がファイルのヘッダ部分に書込まれたものになっています.

ここで, 外部アプリケーションを gnuplot にしている場合には, グラフが表示されているウィンドウを閉じなくても, 次の描画が可能ですが,openmath の場合や Geomview の場合には, これらの外部アプリケーションを終了する迄は Maxima で作業が行えません.

猶,`plot2d` 函数と `plot3d` 函数では, 外部アプリケーションの設定を大域変数 plot_options の plot_format で指定します. 又, 大域変数 plot_option に含まれる run_viewer の設定で, 外部アプリケーションを起動させずに外部アプリケーション向けのデータのみを生成させる事も可能です.

ここで外部アプリケーションに引渡されるファイルの名前は,gnuplot の場合に maxout.gnuplot,openmath の場合には maxout.openmath, そして,Geomview では maxout.geomview と maxout の後に利用するアプリケーション名が置かれたものになります.

1.1.3　フロントエンド

Maxima-5.9.0 までは openmath をデフォルトで用いていましたが, 現在は gnuplot をデフォルトにしています. 猶,Windows 版では gnuplot が標準で同梱されています. その為,gnuplot をインストールしなくても,Maxima をインストールするだけでグラフの表示が行えます.UNIX 版の場合は多少勝手が異なります. 但し,openmath は Maxima のソースファイルに付属しているので, 自力でコンパイルする場合, グラフ表示で openmath を指定すれば問題は無いでしょう. 又,openmath が以前のデフォルトだった事もあり, xmaxima の入力副ウィンドウ内にグラフを表示させる事が出来ます. この様子を図 1.1 に示しておきます.

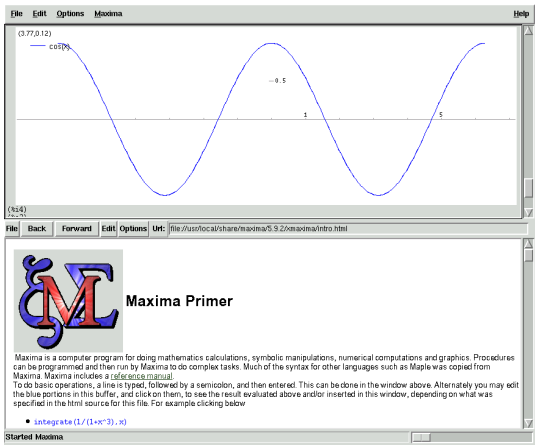


図 1.1: openmath+xmaxima による余弦函数のグラフ

最近は Maxima のフロントエンドとして wxMaxima が主流になっています. 実際,Windows 版では wxMaxima を標準の GUI として採用しています. こちらでは, xmaxima と異なり, グラフは別ウィンドウで表示されます. 猶,wxMaxima を Maxima のフロントエンドとして用いると数式が

綺麗に表示される特徴があります. 更に, 括弧 (を入力すると自動的に) を追加して, 括弧を閉じてくれる機能もあります.

1.1.4 二次元グラフ

では, 最初に幾つかの簡単な例を示しましょう. 先ず, 二次元グラフ表示は, plot2d 函数を用いると, 通常の y=f(x) の函数表示とパラメータ表示の両方で函数が描けます.

例えば, `plot2d(sin(x),[x,-2*%pi,2*%pi]);` と入力すると, xmaxima 上で openmath を利用する時以外は, 別ウィンドウが開かれて, そこにグラフが表示されます.

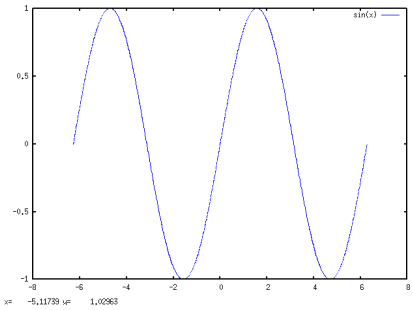


図 1.2: gnuplot による正弦関数の表示

媒介変数を用いた式のグラフ表示を行う事も可能です. ここでは $[\cos(t), \sin(t)\cos(t)]$ のグラフを描いてみましょう. 但し,Maxima で描く曲線は実際の所, 折線で近似する為, 折線の点数が少ないと綺麗な絵が描けない事があります. この点数は大域変数 plot_option の nticks で制御されています. この nticks の初期値は 10 なので, この様な曲線を綺麗に描く為にはもっと多くの点が必要になります. ここでは nticks の値を 100 にしてみましょう. ここで,nticks を一時的に使うだけであれば, plot2d 函数のオプションとして,[nticks,100] を追加します. 又,plot2d 函数を用いる時に, その値を反映させたければ, set_plot_option 函数に nticks を第一成分, 第二成分に割当てたい数値で構成されたリストを引数として引渡します.

ここでは nticks に 100 を設定する例を示しておきましょう.

```
(%i35) set_plot_option([nticks,100]);
(%o35) [[x, - 1.755559702014E+305, 1.755559702014E+305],
[y, - 1.755559702014E+305, 1.755559702014E+305], [t, - 3, 3],
[grid, 30, 30], [view_direction, 1, 1, 1], [colour_z, false],
[transform_xy, false], [run_viewer, true], [plot_format, gnuplot],
[gnuplot_term, default], [gnuplot_out_file, false], [nticks, 100],
[adapt_depth, 10], [gnuplot_pm3d, false], [gnuplot_preamble, ],
[gnuplot_curve_titles, [default]], [gnuplot_curve_styles,
[with lines 3, with lines 1, with lines 2, with lines 5, with lines 4,
with lines 6, with lines 7]], [gnuplot_default_term_command, ],
[gnuplot_dumb_term_command, set term dumb 79 22],
[gnuplot_ps_term_command, set size 1.5, 1.5;
set term postscript eps enhanced color solid 24]]
(%i36) plot2d([parametric,cos(t),cos(t)*sin(t),[t,-5,5]],[x,-3,3]);
```

猶, この設定で `plot2d([parametric,cos(t),cos(t)*sin(t),[t,-5,5]],[x,-3,3]);` を実行した結果が図 1.3 になります.

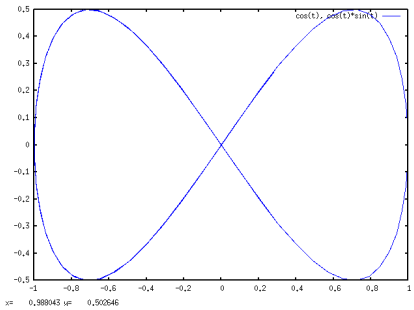


図 1.3: gnuplot による正弦関数の表示

この様な設定を Maxima を立ち上げる時に常に反映させたければ, 1 つの方法は直接ソースファイル (plot.lisp) を修正する方法があります. ただ, これは幾ら何でも大袈裟でしょう. そこで, より一般的な方法としては, カレントディレクトリ上に Maxima の初期化ファイル maxima-init.mac の内部に予め実行したい命令を書込んで置く方法が妥当でしょう.

1.1.5 三次元グラフ表示

三次元グラフ表示は, では plot3d 函数が使えます. ここでは, マニュアルにもあるクラインの壷を gnuplot,openmath や Geomview を使って描いてみましょう. 入力するのは以下の式です.

クラインの壷

```
plot3d(
[5*cos(x)*(cos(x/2)*cos(y)+sin(x/2)*sin(2*y)+3.0)-10.0,
 -5*sin(x)*(cos(x/2)*cos(y)+sin(x/2)*sin(2*y)+3.0),
  5*(-sin(x/2)*cos(y)+cos(x/2)*sin(2*y))],
[x,-%pi,%pi],[y,-%pi,%pi],['grid,40,40]);
```

では,gnuplot,openmath,geomview を使って, クラインの壷がどの様に表示されるか比較してみましょう.

gnuplot によるクラインの壷

gnuplot は Maxima のグラフ表示のデフォルトの外部アプリケーションです. 非常に高機能で, 通常のグラフ表示を行うのであれば最も使えるアプリケーションの 1 つです. 但し,3D グラフの表示はデフォルトではワイヤーフレーム表示のもので, 面を張った曲面を表示する場合には pm3d を指定しなければなりません.pm3d を用いるとそれなりに綺麗な絵が出力されます.gnuplot の描画の終了はグラフィックスのウィンドウ上で q を入力します.

```
set_plot_option([plot_format,gnuplot])
```

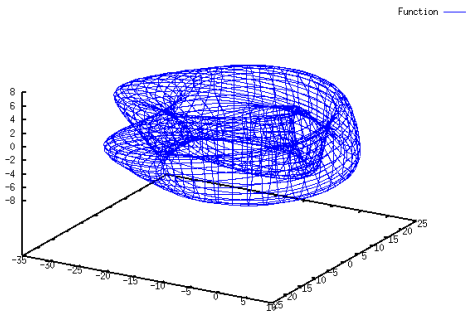


図 1.4: gnuplot によるクラインの壷

openmath によるクラインの壺

openmath は割と高機能で, そこそこ綺麗な描画が出来ます. こちらは標準でパッケージに附属しています. 視点の変更はマウスで直接操作が出来ます.

openmath の GUI は Maxima のバージョンによって異なります.5.9.x の場合,openmath のウィンドウ左上 (立ち上げ時には Menu Here と表示されており, その後は座標値が表示されている個所) にマウスを移動させ, マウスの左ボタンをクリックすると, メニューが出て来ます. 表示を止めたければ, このメニューの最上段の Dismiss を選択します. これに対し,5.10.0 以降はウィンドウの上部に Close,Config,Replot,Zoom, Save,Rotate,Help といったボタンが並ぶ様になっており, 終了する場合には最上段左の Close をクリックします. 猶,gnuplot の場合と異なり,openmath を終了する迄, Maxima の処理は行えません.

```
set_plot_option([plot_format,openmath])
```

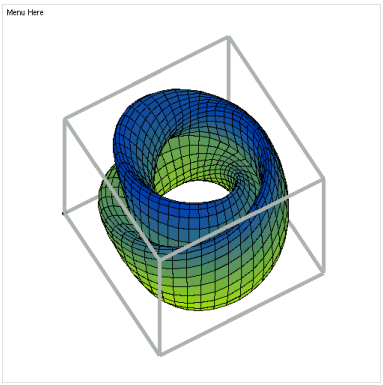


図 1.5: openmath によるクラインの壺 (5.9.x)

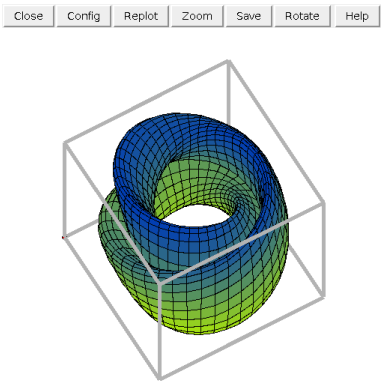


図 1.6: openmath によるクラインの壺 (5.10.0 以降)

Geomview によるクラインの壷

Geomview はミネソタ大学の Geometry Center で開発されたツールで,gnuplot や openmath と比べると, これらの中では最も高機能で描画も美しいものです. 猶,gnuplot と openmath と比べ, 動作環境を選ぶ方で基本的に UNIX 環境で動作します.

描画を開始するとグラフと 2 個の制御パネルが現われ, 視点の変更は長細い制御ウィンドウを用います. 終了は別のウィンドウの File メニューから Exit を選択します.

```
set_plot_option([plot_format,geomview])
```

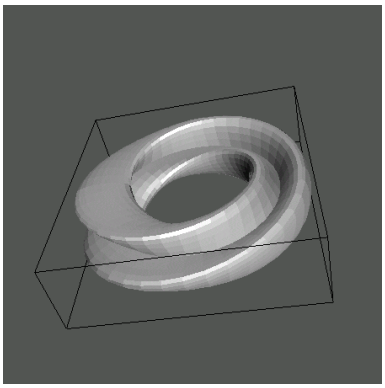


図 1.7: Gemoview によるクラインの壷

他に描画に利用可能なものに,Izic や PostScript があります.

又,Maxima の `system` 函数を利用すれば,Maxima で生成したデータを外部のアプリケーションに引渡す事でグラフを表示させる事も可能です.

以上,Maxima のグラフ表示に関する概要を解説しました. 猶,Maxima は単純に式やデータを引渡してグラフを描く事が容易に出来ますが, タイトルやラベルの設定やグラフ表示用のアプリケーションの設定も色々と行えます.

それでは, 以降の小節で描画函数や関連する大域変数について詳細を述べましょう.

1.2　plot2d 函数と plot3d 函数について

Maxima での描画命令で最も重要な函数が plot2d 函数と plot3d 函数です. 殆どの事はこれらの函数で済ます事が可能です.

これらの函数ではホームディレクトリ上にデータファイルを生成し, そのファイルを外部アプリケーションに引渡して表示する方式を採用しています. その為, これらの函数を実行する為には,Maxima が対応した外部アプリケーション, 例えば,gnuplot,openmath や Geomview 等が必要になります.

現在,Maxima のソースには,openmath が標準で附属しており,Windows 版のバイナリには gnuplot が附属しています. ここでの解説では,openmath か gnuplot を使う事を前提として進めます.

猶, データファイル名は gnuplot 向けは maxout.gnuplot,openmath の場合は, maxout.openmath と maxout. の後にアプリケーション名が続きます. このデータファイルは数値データのみの為, グラフの粗さが目立つ場合は設定を変更して再描画を行う必要があります.

plot2d 函数と plot3d 函数では, 各々オプションを設定する事が可能です. このオプションは大域変数 plot_options に含まれたもので, これらの函数は, 特に指定が無い場合には大域変数 plot_options の設定に従って描画を行います. 従って, 大域変数 plot_options を適切に設定すればオプションを一々設定しなくても済みます. 更に, この大域変数の設定を Maxima の初期化ファイル maxima-init.mac に記述すれば, その設定が起動時から有効にする事が出来ます.

この節では,plot2d 函数と plot3d 函数の構文について解説し, それから, 大域変数 plot_options の詳細を述べる事とします.

1.2.1　plot2d 函数

plot2d 函数は平面曲線の描画を行う函数です. 以下にその構文を示しておきます.

plot2d の構文

plot2d (⟨ 式 ⟩, ⟨ 定義域 ⟩, ⟨ オプション$_1$⟩, $\cdots$, ⟨ オプション$_n$⟩)
plot2d (⟨ 式 ⟩, ⟨ 定義域 ⟩, ⟨ 値域 ⟩, ⟨ オプション$_1$⟩, $\cdots$, ⟨ オプション$_n$⟩)
plot2d (⟨ 媒介変数式 ⟩, ⟨ オプション$_1$⟩, $\cdots$, ⟨ オプション$_n$⟩)
plot2d (⟨ 媒介変数式 ⟩)
plot2d (⟨ 離散式 ⟩, ⟨ オプション$_1$⟩, $\cdots$, ⟨ オプション$_n$⟩)
plot2d (⟨ 離散式 ⟩)
plot2d ([⟨ 式$_1$⟩, $\cdots$, ⟨ 式$_n$⟩], ⟨ 定義域 ⟩, ⟨ 値域, ⟨ オプション$_n$⟩)
plot2d ([⟨ 式$_1$⟩, $\cdots$, ⟨ 式$_n$⟩], ⟨ 定義域 ⟩, ⟨ 値域 ⟩)
plot2d ([⟨ 式$_1$⟩, $\cdots$, ⟨ 式$_n$⟩], ⟨ 定義域 ⟩)

plot2d 函数は二次元グラフの表示を行います.plot2d 函数で描ける式は特に指定しない場合,1 変数を含むの Maxima の式ですが, 利用者が記述した一変数の Maxima の函数, 複素値函数で実部が 1 実数変数を含む式の実部, 函数を媒介変数を用いて記述したもの, そして, 離散的なデータの表示が可能です.

先ず, 通常の Maxima の式を描画する場合, その変数の定義域が必要になります. 定義域は [⟨ 変数 ⟩, ⟨ 下限 ⟩, ⟨ 上限 ⟩] で構成されたリストです. 又, グラフの縦方向の表示範囲を指定する値域も

これと似たリストですが, ここでの変数は y に固定されている為, $[y, \langle$ 下限 $\rangle, \langle$ 上限 $\rangle]$ になります. 但し, 離散式に対する値域の指定は無効になります.

複素函数の実部を表示する場合は大域変数 plot_options の plot_realpart の項目を true に予めしておくか, [plot_realpart,true] をオプションとして引渡します. 猶, 大域変数 plot_options の詳細については後の小節で詳細を述べます.

次に, 媒介変数式の書式は以下の二通りの書式が許容されます.

媒介変数式の書式

[parametric, $\langle X$ 座標 $\rangle, \langle Y$ 座標 $\rangle, \langle$ オプション$_1\rangle, \cdots, \langle$ オプション$_n\rangle$]
[parametric, $\langle X$ 座標 $\rangle, \langle Y$ 座標 $\rangle$]

先ず,X と Y 座標は変数 t を唯一の変数として持つ式でなければなりません. 因に, この書式で媒介変数は t に固定されています. その為, 半径 1 の円を描く場合, [parametric,cos(t),sin(t)] と表記します. オプションは通常の plot2d 函数のオプションが入れられます. 猶, 媒介変数 t の定義域は, 媒介変数式の中に入れても,plot2d 函数で描く式の定義域として与えても構いません. 図 1.8 に plot2d([parametric,cos(2*%pi*t/6),sin(2*%pi*t/6)]) の結果を示しておきましょう.

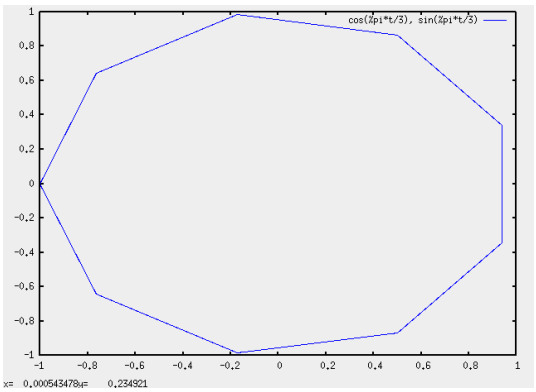


図 1.8: plot2d([parametric,cos(2*%pi*t/6),sin(2*%pi*t/6)]) の結果

猶,t の定義域を省略する事も可能です. これは大域変数 plot_options に予め t の定義域が [t,-3,3] で設定されている為です. ここで t の定義域を変更したい場合は, 大域変数 plot_options の項目 t の定義域する方法と plot2d 函数で定義域を指定する方法があります. 又, 媒介変数式でグラフの粗さが目立つ場合には, 大域変数 plot_options の項目の nticks を 10 よりも大きな値, 例えば,100 にするか, 或いは, [nticks,100] の様に plot2d 函数のオプションとして引渡します.

plot2d 函数は函数だけではなく, 点列の描画も可能です. 点列の場合は, 媒介変数式と似た書式になります. この場合, 二つの書式が可能です.

離散式の書式

[discrete, ⟨X 成分リスト ⟩, ⟨Y 成分リスト ⟩]
[discrete, [[⟨X 成分$_1$⟩, ⟨Y 成分$_1$⟩],$\cdots$, [⟨X 成分$_n$⟩, ⟨Y 成分$_n$⟩]]]

先ず, 双方の書式共に,discrete で開始し, それから,X 成分のリストとそれに対応する Y 成分のリストか,X 成分と Y 成分の対,$[x_i, y_i]$ で構成されるリストの二種類になります.

plot2d 函数は一度に複数の式のグラフを表示する事も可能です. この場合, 表示する複数の式を 1 つのリストで与えます. ここで, リストに含まれる媒介変数式や離散式以外の Maxima の式は, 定義域が全て同じものでなければなりません.

媒介変数式を含む場合, その媒介変数式毎に定義域やその他のオプションが設定可能です. 図 1.9 に媒介変数式を含む式リストの処理結果を示します.

```
plot2d([x^3+2,[parametric,cos(t),cos(t)*sin(t)]],
[x,-3,3],[y,-2,4],[t,-5,5],[nticks,100])
```

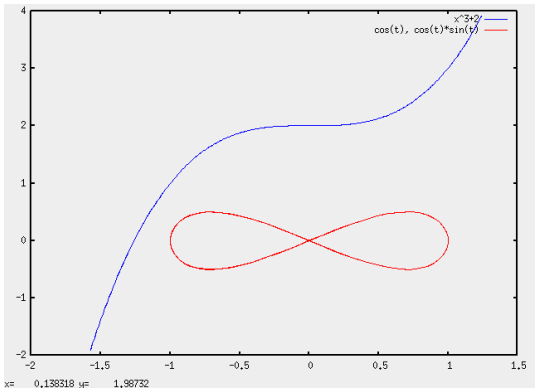


図 1.9: 媒介変数式と通常の式の混在

又, 複数の媒介変数式がある場合には, 媒介変数式内部にオプションを持たせて媒介変数式毎に設定を行う事が可能です.

```
plot2d([[parametric,cos(t),sin(t),[t,-%pi,%pi],[nticks,20]],
[parametric,2*cos(t),sin(-t),[t,-%pi/2,%pi/4],[nticks,5]]]);
```

この例では, 最初の半径 1 の円周の描画を 20 個の点で描画しますが, その次の楕円の描画では,5 点で描画しています. ここでは,nticks の位置に注意して下さい. 即ち, 各々を媒介変数式の中に記述しているので, その影響は個々の媒介変数式にのみ影響されます.

では以下の様に変更するとどうなるでしょうか？

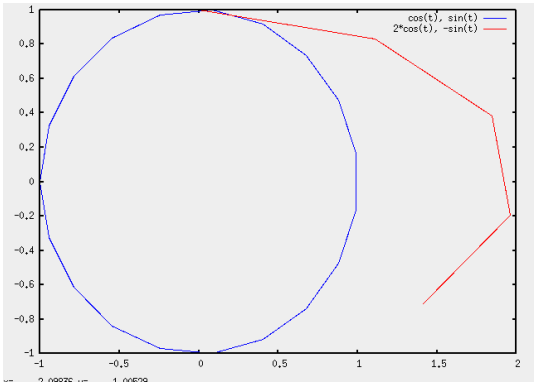


図 1.10: 複数の媒介変数式

```
plot2d([[parametric,cos(t),sin(t),[t,-%pi,%pi],[nticks,20]],
[parametric,2*cos(t),sin(-t),[t,-%pi/2,%pi/4],[nticks,5]]],
[nticks,40]);
```

この場合, 媒介変数式内部の設定の方が優先される為に, 外の `[nticks,40]` は無視されてしまうので, 結果として, 図 1.10 と同じ絵になります.

ところで,plot_format で openmath を指定する場合, 複数の媒介変数式の中に定義域を記述した時に描画が上手く出来ない事があります. その為,`plot2d` 函数を用いる場合は,plot_format をデフォルトの gnuplot にしたままの方が無難でしょう.

さて,`plot2d` 函数のオプションは大域変数 plot_options を構成するリストでもあります. この大域変数 plot_options の諸項目リストを設定する事で, グラフのタイトルやラベル,X,Y 軸のラベル等の細かなグラフの指定が行えます. 特に,gnuplot を用いる場合, 即ち,plot_format が gnuplot, 或いはオプションとして [plot_format,gnuplot] を指定した場合,plot_options の gnuplot_preamble を上手く使う事で,gnuplot の命令文を実行させる事が可能です. この大域変数 plot_options には多くの説明すべき事項がある為, 次の節で詳細を述べる事とします.

1.2.2 plot3d 函数

Maxima では f(x,y) の形式の 2 変数の式の描画が可能です. 更に, 一変数複素数値函数の実部 $f(x+iy)$ のみの表示も可能です.

但し,`x^2+y^2+z^3=1` の様な式のグラフはそのままの形式では描けません.`plot3d` 函数の構文は基本的に `plot2d` 函数の構文を三次元に拡張したもとと言えます.

plot3d の構文

plot3d (⟨ 式 ⟩, ⟨ 定義域$_1$⟩, ⟨ 定義域$_2$⟩, ⟨ オプション$_1$⟩, ··· , ⟨ オプション$_n$⟩)
plot3d (⟨ 式 ⟩, ⟨ 定義域$_1$⟩, ⟨ 定義域$_2$⟩)
plot3d ([⟨ 式$_1$⟩, ⟨ 式$_2$⟩, ⟨ 式$_3$⟩], ⟨ 定義域$_1$⟩, ⟨ 定義域$_2$⟩, ⟨ オプション$_1$⟩, ··· , ⟨ オプション$_n$⟩)
plot3d ([⟨ 式$_1$⟩, ⟨ 式$_2$⟩, ⟨ 式$_3$⟩], ⟨ 定義域$_1$⟩, ⟨ 定義域$_2$⟩)
plot3d ([⟨ 式$_1$⟩, ⟨ 式$_2$⟩, ⟨ 式$_3$⟩], ⟨ 定義域$_1$⟩, ⟨ 定義域$_2$⟩, ⟨ 定義域$_3$⟩, ⟨ オプション$_1$⟩, ··· , ⟨ オプション$_n$⟩, [$transform\_xy$, ⟨ 函数 ⟩])

plot3d 函数は plot2d 函数と似た構文を持っていますが, plot2d と違い, 複数の曲面を同時に描く事は出来ません. 又,gnuplot で曲面を描く場合,pm3d の設定を行うと曲面を張りますが, 無設定の場合はワイヤーフレーム表示になります. 詳細は, 大域変数 plot_options の gnuplot_pm3d の項目を参照して下さい.

plot3d 函数は三次元空間内の曲面だけではなく, 空間曲線も描けます. ここで空間曲線を描く場合は,x,y,z 成分の各成分を,⟨ 定義域$_1$⟩ か ⟨ 定義域$_2$⟩ の変数として記述します.

極座標系の場合も空間曲線を描く方法に似ていますが, この場合には, 大域変数 plot_options の項目 transform_xy に座標変換を行う函数を指定する必要があります. ここで通常のデカルト座標から極座標への変換は単純に polar_to_xy を指定するだけで済みます. 但し, 利用者独自の座標変換函数の指定が必要であれば,make_transform 函数を併用する必要があります. この詳細は, 大域変数 plot_options の transform_xy の個所で述べます.

1.3　大域変数 plot_options

Maxima はグラフ表示に gnuplot,openmath 等の外部アプリケーションを用います. アプリケーションに様々な設定を行いたい場合にはどうすればよいでしょうか？先程の実例に示した様に二つの方法がありました. 先ず, 通常の描画を行う plot2d 函数や plot3d 函数を用いる場合にオプションとして引渡す方法, そして, セッションを通じて設定しておきたい場合に用いる, set_plot_option 函数による大域変数 plot_options の内容変更を行う方法です.

この小節では, 大域変数 plot_options の解説を行いましょう.

そこで, 最初に plot_options の値を見てみましょう. これは直接 plot_options; と入力すれば見る事が出来ます.

```
(%i4) plot_options;
(%o4)[[x, - 1.75555970201398E+305, 1.75555970201398E+305],
[y, - 1.75555970201398E+305, 1.75555970201398E+305], [t, - 3, 3],
[grid, 30, 30], [view_direction, 1, 1, 1], [colour_z, false],
[transform_xy, false], [run_viewer, true], [plot_format, gnuplot],
[gnuplot_term, default], [gnuplot_out_file, false], [nticks, 10],
[adapt_depth, 10], [gnuplot_pm3d, false], [gnuplot_preamble, ],
[gnuplot_curve_titles, [default]], [gnuplot_curve_styles,
[with lines 3, with lines 1, with lines 2, with lines 5, with lines 4,
```

```
with lines 6, with lines 7]], [gnuplot_default_term_command, ],
[gnuplot_dumb_term_command, set term dumb 79 22], [gnuplot_ps_term_command, se\
t size 1.5, 1.5;set term postscript eps enhanced color solid 24],
[logx, false], [logy, false], [plot_realpart, false]]
```

ここで示す様に大域変数 plot_options は二成分リストから構成されたリストです. この大域変数を構成する二成分リストは [plot_format,gnuplot] の様に先頭が plot_options の項目で, その後に項目の値が続きます. 例えば,[plot_format,gnuplot] は外部アプリケーションが gnuplot であり, 生成するファイルも gnuplot のデータファイルになる事を示しています.

この例では,plot_options と入力して全ての設定を表示しています. これでは流石に見難いので,get_plot_option 函数を用いると, 大域変数 plot_options の必要な設定値だけを調べる事が出来ます. ここで,get_plot_option 函数は 1 つの項目だけを引数とし, 項目とそれに対する値の二成分リストを返す函数です.

ここで大域変数 plot_options の各項目に対応する値を変更したい場合, 代入の演算子:を使って, このリストを一々全て記述する事はあまり頭の良い方法とは言えません. 指定した項目を変更する目的の為,set_plot_option 函数が用意されています. この set_plot_option 函数の引数として, 項目とその値で構成されるリストを引渡します. 但し,set_plot_option 函数に指定可能な項目は一件に限られます. これらの函数の構文を以下に纏めておきましょう.

大域変数 plot_options に関連する函数

set_plot_options([⟨ 項目 ⟩, ⟨ 値 ⟩])
get_plot_option(⟨ 項目 ⟩)

では, これらの函数の実例を示しておきましょう. この実例では,plot_format の値を get_plot_option 函数で取出し, この項目の値を set_plot_option 函数で openmath に変更させるというものです.

```
(%i13) get_plot_option(plot_format);
(%o13)                       [plot_format, gnuplot]
(%i14) set_plot_option([plot_format,openmath]);
(%o14) [[x, - 1.75555970201398E+305, 1.75555970201398E+305],
[y, - 1.75555970201398E+305, 1.75555970201398E+305], [t, - 3, 3],
[grid, 30, 30], [view_direction, 1, 1, 1], [colour_z, false],
[transform_xy, false], [run_viewer, true], [plot_format, openmath],
[gnuplot_term, default], [gnuplot_out_file, false], [nticks, 10],
[adapt_depth, 10], [gnuplot_pm3d, true], [gnuplot_preamble, ],
[gnuplot_curve_titles, [default]], [gnuplot_curve_styles,
[with lines 3, with lines 1, with lines 2, with lines 5, with lines 4,
with lines 6, with lines 7]], [gnuplot_default_term_command, ],
[gnuplot_dumb_term_command, set term dumb 79 22], [gnuplot_ps_term_command, se\
t size 1.5, 1.5;set term postscript eps enhanced color solid 24],
[logx, false], [logy, false], [plot_realpart, false]]
```

```
(%i15) get_plot_option(plot_format);
(%o15)                        [plot_format, openmath]
```

この例等で示した様に, 大域変数 plot_options には色々な項目があります. ここでは, 大域変数 plot_options の各項目について解説しましょう. そこで, 外部アプリケーションに依存しない表示設定に関係する項目を以下に示しましょう.

大域変数 plot_options の一般的な項目

項目	初期値	概要
plot_format	[plot_format, gnuplot]	グラフ表示アプリケーションを設定
run_viewer	[run_viewer, true]	true の場合に plot_format で指定したアプリケーションを起動
colour_z	[colour_z, false]	カラーの PS ファイルを出力するかどうかを指定するフラグ
view_direction	[view_direction, 1, 1, 1]	plot_format が ps の場合に三次元グラフの視点を指定
transform_xy	[transform_xy, false]	三次元グラフ表示で座標変換を行うかどうかを指定するフラグ
grid	[grid 30, 30]	三次元グラフの解像度を指定
nticks	[nticks, 10]	二次元グラフの解像度を指定
logx	[logx, false]	二次元グラフでの X 座標の対数目盛フラグ
logy	[logy, false]	二次元グラフでの Y 座標の対数目盛フラグ
plot_realpart	[plot_realpart, false]	複素函数実部の表示フラグ
adapt_depth	[adapt_depth, 10]	
x	[x,-a,a]	a はシステムによって異なる浮動小数点. 表示可能な領域の目安
y	[y,-a,a]	a はシステムによって異なる浮動小数点. 表示可能な領域の目安
t	[t,-3,3]	助変数表示に於ける助変数の定義域

Maxima は現在,gnuplot をデフォルトの描画用のアプリケーションとしています. その為,gnuplot を制御する項目が沢山存在します.

gnuplot に関連する plot_options の項目

項目	初期値	概要
gnuplot_pm3d	[gnuplot_pm3d, true]	gnuplot の PM3D(曲面に面を張るかどうか) のオプション
gnuplot_curve_titles	[gnuplot_curve_titles,[default]]	描画する曲線の表題を設定.
gnuplot_curve_styles	[gnuplot_curve_styles,[with lines 3, with lines 1, with lines 2, with lines 5, with lines 4,with lines 6, with lines 7]]	曲線の様式を指定
gnuplot_out_file	[gnuplot_out_file, false]	gnuplot の出力ファイルを指定.
gnuplot_term	[gnuplot_term, default]	gnuplot の term を設定し, 対応するデータを出力する.
gnuplot_default_term_command	[gnuplot_default_term_command,]	gnuplot で実行する term に関連する命令文を記述する
gnuplot_dumb_term_command	[gnuplot_dumb_term_command, set term dumb 79 22]	term を dumb とした場合の設定
gnuplot_ps_term_command	[gnuplot_ps_term_command, set size 1.5, 1.5; set term postscript eps enhanced color solid 24]	PS ファイルに追加する命令
gnuplot_preamble	[gnuplot_preamble,]	gnuplot に引渡す諸設定を文字列で指定

この plot_options の各成分の値は set_plot_options 函数で設定する事も可能ですが,plot2d や plot3d 函数のオプションとして引渡す事も可能です.

plot_options では, 一般的なグラフ表示の為の設定の他に,gnuplot 専用の設定もあります. その場合には,gnuplot_xxx の様に頭に gnuplot が付くので分り易いかと思います.

それでは各成分の意味について解説しましょう.

plot_format

plot_format で Maxima がグラフ表示で用いる外部アプリケーションを指定します. 指定可能な値は gnuplot,openmath,geomview,ps,mgnuplot と zic です.

猶,Maxima-5.9.0 から plot_format の初期値として gnuplot が指定されています. 描画に用いる外部アプリケーションは基本的にここでの値と同じ名前のアプリケーションになりますが,ps を指定した場合だけは ghostview がグラフ表示用の外部アプリケーションとして設定されます.

ここで gnuplot,openmath と geomview については今迄色々と解説しているので問題は無いと思います.mgnuplot は Tcl/Tk で記述したフロントエンドを gnuplot に被せたものです. こちらは三次元空間内の曲面をスライドバーを使って回転させる事が出来ますが, 基本的に gnuplot を操作する plot_option の項目は反映されません. 最後の zic は Izic を外部アプリケーションとするものです. この Izic 自体は Tcl/Tk でフロントエンドを記述した古いアプリケーションの為, 実際に使われる事は無いでしょう.

Maxima には openmath も同梱されているので, 指定すれば使えますが,ps(ghostview), geomview,mgnuplot に Izic になると, 各々のアプリケーションが利用可能な環境でなければ使えません.

ここで Maxima の `plot2d` 函数と `plot3d` 函数は,plot_format で指定したアプリケーション向けのデータファイルを UNIX 環境の場合は, 利用者のホームディレクトリに生成します. それから

run_viewer の値が true であれば, そのファイルを指定した外部アプリケーションに引渡し, この値が false の場合, 外部アプリケーションの立上げを行いません.

猶, ここで生成されるデータファイルの名前は maxout.gnuplot の様に maxout. の後に plot_format の値が付いたファイル名になります. 又, 続けて描画を行う場合, データファイルをそのまま上書きします.

このファイルは各アプリケーションから開く事も可能です. 初期値の gnuplot の場合, データの他に描画命令やオプションの諸設定も記述されたファイルとなっているので, 単純に,gnuplot から `load 'maxout.gnuplot'` で読込むと, グラフの表示を行います. この時, マウスが ON になっていればマウスで直接三次元画像を把持して, 回転や拡大が行えます. もしも, マウスが ON でなければ, `set mouse` で ON に出来ます.

mgnuplot の場合は,gnuplot 向けの曲線, 或いは曲面のデータのみが含まれたファイルになります. その為,plot2d 函数の場合は `plot 'maxout.mgnuplot'` ,plot3d 函数の場合, `splot 'maxout.mgnuplot'` を実行すれば描画を行います.

colour_z

colour_z は, カラーの PostScript ファイルを出力するかどうかを指定する項目です.true の場合にカラーの PostScript ファイル出力を行い,false の場合は白黒の PostScript ファイルを出力します. この colour_z は plot_format として ps が指定された場合のみに有効です.

view_direction

view_direction は視点の位置を指定します. この指定は plot_format が ps の場合の時だけ有効です.

transform_xy と make_transform 函数

transform_xy は通常のデカルト座標系から別の座標への変換を指定します. ここで設定する値は, 通常のデカルト座標系を用いる場合に初期値の false, 極座標系を用いる場合には polar_to_xy を設定します. この transform_xy に利用者定義の座標変換を利用する事も可能です. この場合は,`make_tranform` 函数を用いて指定しなければなりません.

以下に `make_transform` 函数の構文を示しておきます.

`make_transform` 函数の構文

make_transform(⟨ 変数リスト ⟩, ⟨f_x⟩, ⟨f_y⟩, ⟨f_z⟩)

ここで f_x, f_y, f_z は各々デカルト座標系の X,Y,Z 成分となる ⟨ 変数リスト ⟩ に含まれる変数の函数です. 例えば, 通常の円筒座標であれば, `make_transform([r,th,z],r*cos(th),r*sin(th),z)` の様にします. 実際に利用する場合は make_transform 函数を用いた利用者定義の函数を生成し, それを transform_xy に設定するか,`make_transform` 函数をそのまま設定するか何れかが使えます. 但し, 利用者定義の変換函数を用いる場合には polar_to_xy の様に函数名だけを指定する事は出来ず, 変数も一緒に書かなければなりません.

これだけでは分り難いので以下に実例を示しておきます.

```
(%i6) neko(r,th,z):=make_transform([r,th,z],
r*cos(%pi*th/180),r*sin(%pi*th/180),z);
                                                        %pi th
(%o6) neko(r, th, z) := make_transform([r, th, z], r cos(------),
                                                         180
                                                        %pi th
                                                  r sin(------), z)
                                                         180
(%i7) plot3d(r*th^2,[r,1,2],[th,0,360],[gnuplot_pm3d,true],
[transform_xy,neko(r,th,z)]);
```

この例では,plot3d 函数のオプションとして [transform_xy,neko(r,th,z)] を引渡していますが, このオプションを [transform_,make_transform([r,th,z], r*cos(%pi*th/180),r*sin(%pi*th/180),z)] としても同じ結果になります. ここで定義した函数は, 通常の polar_to_xy では角度がラジアンになるのに対し, こちらは度を用いる様にしている点です.

ここで注意点として, `[transform_xy,neko]` の様に利用者定義の変換函数に対し, その函数名だけを記述する事は出来ません. 必ず, `[transform_xy,neko(r,th,z)]` の様に変数も含めて記述しなければなりません. 但し,polar_to_xy に関しては, `[transform_xy,polar_to_xy]` と変数を省略して表記出来ます. これは,polar_to_xy の実体が plot.lisp 内部で定義された LISP の函数の為だからです.

では, 上記の実行結果を以下に示しておきましょう.

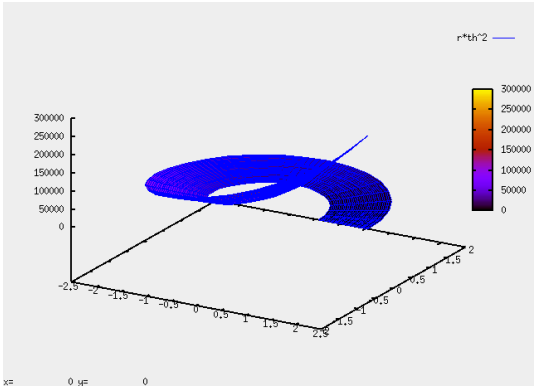


図 1.11: transform_xy と make_transform の組合せ

nticks

nticks は二次元グラフの解像度になります. 初期値値は 10 です. 描いた曲線が粗い場合に, この nticks の値を大きくすると良いでしょう. 特に媒介変数式の表示では初期値を予め大きく指定しなければ綺麗な曲線は描けないでしょう.

grid

grid は三次元グラフの解像度で, 初期値は [30,30] です.grid の値は X 方向と Y 方向の解像度の対で記述し,[grid,50] の様に記述する事は出来ません. 必ず,[grid,50,50] の様に記述します. 描いた曲線が粗い場合, この grid に大き目の整数値を成分に持つリストを設定します.

x と y

x と y に設定される値は Maxima の土台にある Common Lisp 処理系の
`(/ most-positive-double-float 1024)` の処理結果となります. この値は Common Lisp で扱える数値の限界を示すものの為, この値を越える数値は Maxima では扱えないので, 当然, グラフ表示も出来ません. 猶, グラフ表示可能な数値の上限は Maxima で利用可能なグラフ表示アプリケーション毎に異なる為, ここでの設定以下であっても表示が出来ない場合もあります. 例えば,gnuplot では, $\sqrt{x}$ のグラフをこの x の値以下であれば表示が可能ですが,openmath では 10^{203} よりも大きな数値になると表示が出来なくなります.

t

これは函数の媒介変数式で利用する変数 t の定義域を定めます.Maxima では媒介変数式の変数は `t` に固定されています. この定義域を定めておけば, グラフ表示で媒介変数の定義域を省略する事が可能になります.

xlog と ylog

xlog と ylog は各々X 軸と Y 軸の目盛を対数目盛にします. この設定は二次元グラフの場合に有効で, 三次元グラフの場合には無効です.

gnuplot_pm3d

gnuplot_pm3d は gnuplot の pm3d を有効にします. 初期値は false の為,gnuplot で描く曲面はワイヤーフレーム表示となっていますが,true を設定した場合,maxout.gnuplot に `set pm3d` を書込みます. 又, 必要に応じて,b,s,t といった文字で構成された文字列も与えられます. これらの文字にはそれ相当の意味がありますが, この詳細については gnuplot の pm3d の 2.4 節で改めて解説します.

実例を以下に示しておきましょう.

pm3d の例

```
plot3d(sin(x*y),[x,-3,3],[y,-3,3],[gnuplot_pm3d,true],[grid,50,40])
```

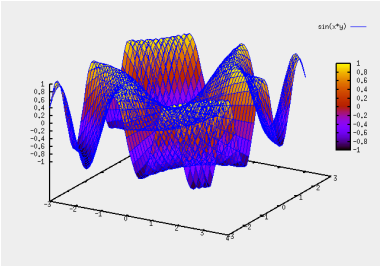


図 1.12: pm3d の例

しかし,gnuplot ではより高度な表示が可能です.`set pm3d` にしても, set pm3d at bs とする事で, 最初に底面への曲面の投影を行い, それから曲面を描くといった指定が行えます. この様な細かな設定は, gnuplot_preamble を用いればより詳細に行えますが,gnuplot_pm3d に引渡す事も可能です. 例えば, 底面への投影を意味する文字 b と曲面の描画を意味する s を組合せた文字の列 bs を Maxima の文字列にせず,`[gnuplot_pm3d,bs]` の様にして引渡せます. 以下に例を示しておきましょう.

gnuplot_pm3d の設定例

```
plot3d(sin(x*y),[x,-3,3],[y,-3,3],
[gnuolot_pm3d,bs],[gnuplot_preamble,"unset surf"],[grid,50,40])
```

この例では,gnuplot の pm3d のオプションとして bs の他に,gnuplot_preamble を用いて,gnuplot に `unset surf` を実行させます. この設定は, 曲面上の網目を消す効果があります. この式を実行して得られた図を図 1.13 に示しておきます.

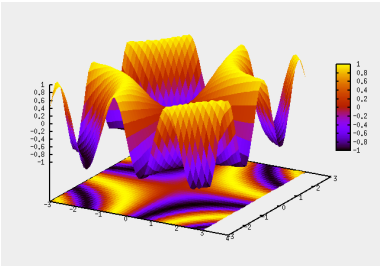


図 1.13: gnuplot_preamble を使った例

猶, 図 1.13 は次の様に gnuplot_preamble をだけを用いても描けます.

gnuplot_preamble を使った例

```
plot3d(sin(x*y),[x,-3,3],[y,-3,3],
[gnuplot_preamble,"set pm3d at bs;unset surf"],[grid,50,40])
```

gnuplot_term

gnuplot_term は gnuplot の terminal を設定します. ここで指定した値を用いて,maxout.gnuplot 内部では,`set term` が記述されます. ここで設定可能な terminal は非常に多い為, ここでは一々説明しません. 大雑把に説明すると,tek410x や x11 の様な仮想端末に関連するものや, tgif の様なアプリケーションに対応したもの, そして,latex,pdf,postscript, png,jpeg や gif の様なデータ書式によるもの等が選べます. この terminal の詳細は参考文献 [11] か, 或いは gnuplot 上で`? term`と入力すれば, オンラインマニュアルと term に設定可能な値の一覧が表示されるので, そちらを参照して下さい.

猶,gnuplot_term に'default' 以外の値を設定した場合,Maxima では gnuplot に引渡すデータファイルの maxout.gnuplot に加えて,gnuplot による maxout.gnuplot の処理結果の出力ファイルの設定が行われます. 例えば,terminal として tgif を設定した場合で解説しましょう.

maxout.gnuplot 内部に `set term tgif` と `set out '/home/yokota/maxplot.tgif/'` (ここで /home/yokota/が利用者のホームディレクトリです) が書込まれます. そして,maxout.gnuplot が gnuplot に引渡されると, ホームディレクトリに maxplot.tgif が書込まれます. 猶,maxplot.tgif を maxplot.obj に変更して tgif に読込ませた結果が図 1.14 に示すものです.

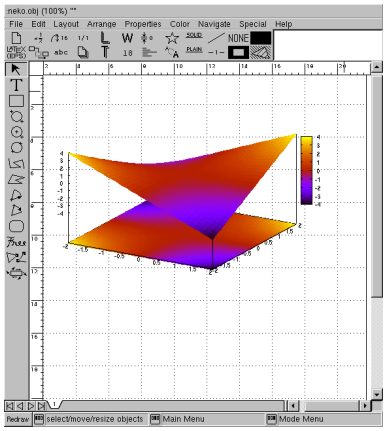
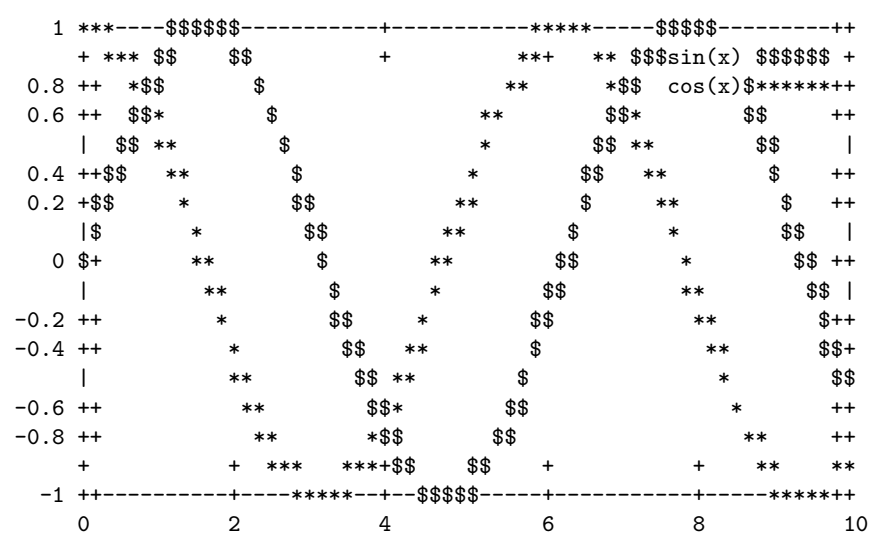


図 1.14: term を tgif にした結果 (maxplot.tgif) を tgif で表示

gnuplot_dumb_term

gnuplot_dumb_term_command は gnuplot の term として dumb を選択した時の設定が行えます. 以下に実例を示しておきます.

```
(%i86) plot2d([sin(x),cos(x)],[x,0,10],[gnuplot_term,dumb],
[gnuplot_dumb_term_command,["set term dumb 70 20"]]);
```




```
Output file "/home/yokota/maxplot.dumb".
(%o86)
```

この例では,dumb 端末を 70 列 20 行として表示しています. 猶,term を dumb にしなければ, この設定は無効です.

gnuplot_curve_titles

gnuplot_curve_titles には描く曲線や曲面の表題をリストで追加します. この時, 構文は gnulot の plot 命令や splot 命令に引渡す表題の設定に準じます. 例えば, 二曲線に A1,A2 と設定したい場合には, `gnuplot_curve_titles,["title 'A1'","title 'A2'"]]` と設定します. この時,maxout.gnuplot には以下の様に,plot 命令の行にリストの内容が追加されます.

maxout.plot の内容 (一部)

```
plot '-' title 'sin(x)' with lines 3,  '-' title 'cos(x)' with lines 1
```

gnuplot_curve_styles

gnuplot_curve_styles に描画する線分の書式を設定します. これは gnuplot の with 命令の文を文字列として引渡す為に用います. 例えば,[gnuplot_curve_styles,["with lines 7","with lines 2"]] の様に行います. 因に,maxout.gnuplot 内部では,gnuplot の描画命令 plot のオプションとして引渡されます. その為,, 後述の gnuplot_preamble で曲線のスタイルを幾ら設定しても, こちらの設定が優先されるので注意が必要です.

gnuplot_preamble

gnuplot_preamble は gnuplot の制御文を,gnuplot に直接引渡す為の仕組みで, gnuplot の制御文の間には; を入れた Maxima の文字列を指定します. この preamble の設定は plot_option で gnuplot に関連する設定と,gnuplot の数値データの間に置かれます. ここで,Maxima で入力した文字列に大文字があった場合, maxout.gnuplot に出力される際に小文字に変換されます.

preamble の内容がどの様に書込まれるか, 以下の描画をさせた場合で観察しましょう.

```
(%i4) nekoneko:"set title 'mike';set xlabel 'X';\
set ylabel 'Y';set zlabel 'height';";
(%o4) set title 'mike';set xlabel 'X';set ylabel 'Y';set zlabel 'height';
(%i5) plot3d(sin(x*y),[x,0,10],[y,0,10],
[gnuplot_pm3d,true],[gnuplot_preamble,nekoneko]);
```

この場合,gnuplot に引渡されるデータファイル maxout.gnuplot の内容はどの様になっているでしょうか. そのファイルの頭の部分を以下に示しておきます.

gnuplot_preamble を反映した maxout.gnuplot のヘッダ部分

```
set pm3d

set title 'mike';set xlabel 'X';set ylabel 'Y';set zlabel 'height';
splot '-' title 'sin(x*y)' with lines 3
0.0     0.0     0.0
0.3333333333333333     0.0     0.0
0.6666666666666666     0.0     0.0
1.     0.0     0.0
```

この様に gnuplot の splot 命令の直前に gnuplot_preamble の内容が書込まれている事が分りますね. この事を利用すれば gnuplot の諸設定を行う事が可能になるのです.

1.4 その他の描画函数

Maxima の描画函数には上述の `plot2d` 函数や `plot3d` 函数の他に, 幾つかの描画函数があります. とは言え, どちらかと言えば補助的な函数が多く, それも openmath 専用や PostScript への出力のみといった機能や出力が限定されたものが殆どです.

最初に, 比較的汎用性がある `openplot_curves` 函数の解説をしましょう.

1.4.1 openplot_curves

`openplot_plot` 函数は大域変数 plot_options の plot_format とは無関係に openmath を用いて与えられた点列の描画を行う函数です. とは言え, オプションを変更する事で, 実数値一変数函数の描画も可能です.

openplot_curves

openplot_curves ([⟨ 点列リスト$_1$⟩, ··· , ⟨ 点列リスト$_n$⟩])
openplot_cur ves ([⟨ オプション$_1$⟩, ⟨ 点列リスト$_1$⟩, ··· , ··· , ⟨ オプション$_n$⟩, ⟨ 点列リスト$_n$⟩)
openplot_curves (⟨ 点列リスト ⟩)
openplot_curves (⟨ オプション ⟩, ⟨ 点列リスト ⟩)
openplot_curves (⟨$xfun$ を含むオプション ⟩)

ここでの点列リストの書式は,$[x_1, y_1, \cdots, x_n, y_n]$ か $[[x_1, y_1], \cdots, [x_n, y_n]]$ の二種類に限定されます.

`openplot_curves` のオプションは plot_options とは無関係で, openmath のメニューからも設定可能な事項になります. 又, 点列リストの直前にあるオプションがその点列の表示で用いられます.

`openplot_curves` の設定項目を以下に示しておきます.

openplot_curves の設定項目

項目	初期値	概要
xfun	なし	指定した変数 x を持つ式を指定
color	blue から開始	曲線の色を指定
plotpoints	0	直線や曲線上に点を打つ為のフラグ. 初期値は 0
linecolors	blue	直線/曲線の色を指定.
pointsize	0	点の大きさを指定
nolines	0	点列を繋ぐ線分表示の為のフラグ
bargraph	0	棒グラフ表示への切替の為のフラグ
xaxislabel	無指定	X 軸のラベルを指定
yaxislabel	無指定	Y 軸のラベルを指定

openplot_curves のオプションには独特の書式があります. これは plot_options に似たもので, `{ 項目 値 }` の書式の文字列を空行で区切って, 全体を引用符で括ったリストになります. 例えば, 各点を表示して, その点の大きさを 6 にする場合には, `["{plotpoints 1} {pointsize 6}"]` をオプションとして与えます.

次の例では, 折線毎に点の大きさと, 折線と点の色を変更し, 各軸にラベルを設定しています.

```
openplot_curves([["{plotpoints 1} {pointsize 6} {color red}"],
[1,2,3,4,5,1],
["{pointsize 10} {xaxislabel time} {yaxislabel neko} {color black}"],
[10,9,9,2,4,2]])
```

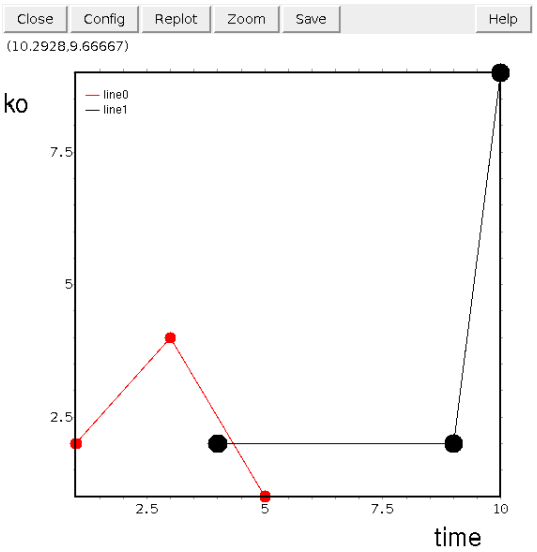


図 1.15: openplot_curves のオプション例

又,xfun を指定する事で,Maxima の式をグラフに追加する事も可能です. 先程の例に正弦函数を追加した例を以下に示しておきましょう.

```
openplot_curves([["{plotpoints 1} {pointsize 6} {color red}"],
[1,2,3,4,5,1],
["{pointsize 10} {xaxislabel time} {yaxislabel neko} {color black}"],
[10,9,9,2,4,2],
["{xfun sin(x)} {color green} {plotpoints 1} {pointsize 1}"]])
```

ここで注意する事は, 描画する式の変数は x に限定される事です. 又, 函数の描画だけを行いたければ, 点列リストを持たないオプションのみの引数を与えます. 但し,openplot_curves 函数で式の定義域を与える事は出来ない様です.

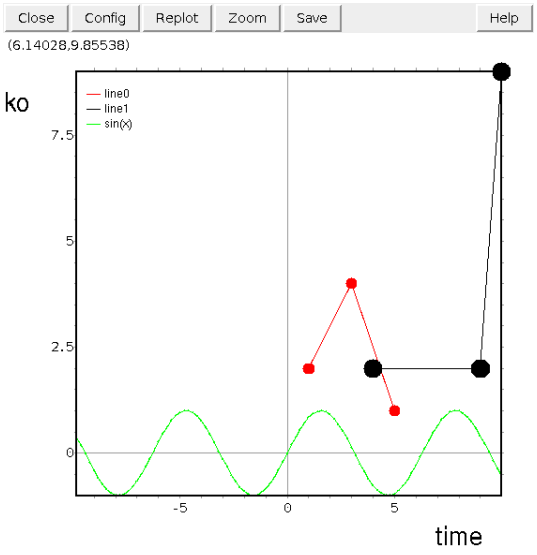


図 1.16: openplot_curves のオプション例 (その 2)

1.4.2 Postscript に関連する函数

Postscript に関連する函数

plot2d_ps(⟨ 式 ⟩,⟨ 定義域 ⟩)
viewps(⟨*PostScript* ファイル名 ⟩)
viewps()
psdraw_curve (⟨ 点列リスト ⟩)
psdraw_points (⟨ 点列リスト ⟩)
pscom(⟨*PostScript* 命令文 ⟩)
closeps()

`plot2d_ps` 函数は `plot2d` 函数に似た構文を持ちますが, デカルト座標系での函数の表示のみが可能です. 具体的には函数と定義域の二つの引数のみを取ります. この函数はカレントディレクトリ上に maxout.ps ファイルを生成し, 内部で `viewps` 函数を呼出して,maxout.ps ファイルの表示を行います.

`viewps` 函数は単純に ghostview に PostScript ファイルを引渡す函数です. ファイル名を指定しない場合には, ホームディレクトリ上の maxout.ps ファイルの表示を行います. 猶,ghostview 命令を持たない環境では, この函数は使えません.

`psdraw_curve` は点列リストのグラフを PostScript 形式で出力します. ここで点列リストの書式は $[x_1, y_1, x_2, y_2, \cdots, x_n, y_n]$ の書式と $[[x_1, y_1], [x_2, y_2], \cdots, [x_n, y_n]]$ の書式の何れかになります. `psdraw_curve` 函数を実行する事でホームディレクトリ上の maxout.ps へのストリームを開きます. このストリームは closeps() で閉じられます.

psdraw_points 函数は点列リストのグラフを PostScript 形式のファイルで出力します. この函数は psdraw_curve 函数に似ていますが, この函数の引数となる点列リストの書式は $[[x_1, y_1], [x_2, y_2], \cdots, [x_n, y_n]]$ のみに限定されます. psdraw_curve 函数と同様に psdraw_points 函数を実行するとホームディレクトリ上のファイル maxout.ps へのストリームが開かれます. 又, closeps() を実行すると, ストリームが閉じられて, maxout.ps ファイルが生成されます.

pscom 函数は与えられた PostScript の命令文をホームディレクトリ上の maxout.ps へのストリームに出力する函数です.

psdraw_curve 函数,psdraw_points 函数と pscom 函数はどちからと言えば原始的な函数で, これらは全てホームディレクトリ上の maxout.ps へのストリームを開き, そのストリームに対して出力を行う函数です. そして,closeps() 函数を実行する事で, ストリームが閉じて maxout.ps が生成されます.

因に, ストリームが開いているかどうかは, 大域変数 pstream で調べられます. ストリームが開いていない場合に,pstream を入力すると結果として false が返り, 開いている場合には, その情報を返します.

実際に,pstream の働きを見ておきましょう.

```
(%i32) psdraw_curve([[1,2],[3,3],[5,1]]);
(%o32)                                false
(%i33) pstream;
(%o33)  #<OUTPUT BUFFERED FILE-STREAM CHARACTER /home/yokota/maxout.ps>
(%i34) closeps();
(%o34)                                true
(%i35) pstream;
(%o35)                                false
```

この例では, 最初に psdraw_curve 函数を実行する事で, ホームディレクトリ (ここでは/home/yokota) 上のファイル maxout.ps へのストリームを開きます. この事は,pstream を見ると,maxout.ps へのストリームが開いている事が分ります. それから closeps() を実行した後に, pstream の値を見ると false になっており, この事からストリームが閉じている事が分ります.

1.5 plot_option 以外の描画に関連する大域変数

Maxima の描画函数で最も重要な大域変数としては plot_options がありました. Maxima にはこの大域変数の他にも多くの大域変数があります. 但し, これらの変数はどちらかと言えば,Maxima の描画函数で内部的に用いる事を想定したものが多く, 用途も特殊なものが多いのが実情です. この様な変数は Maxima のソースファイル plot.lisp の中を見れば色々ある事が分ります.

大域変数

変数名	初期値	概要
in_netmath	false	plot_format の値が openmath の場合に限り,max-out.openmath の内容を画面に表示
show_openplot	false	plot_format が openmath の場合は true, それ以外は false
viewps_command	(ghostview " a")	plot_format を ps にした場合の, PS ファイルの表示命令を設定
ps_scale	[72,72]	PS ファイルの解像度.
window_size	[612.0,792]	PS ファイルのウィンドウの大きさを指定
ps_translate	[0,0]	原点の移動に利用

第2章　gnuplotによる描画について

2.1　maxout.gnuplotの内容

ここではMaximaを使う上で必要なgnuplotの事項のみを記述します. その為,gnuplot全般の使い方や例題を知りたければ,「使いこなすGNUOPLOT」 [11] 等のgnuplotの解説本やgnuplotに付属するマニュアル等の文献を参照して下さい.

先ず,Maximaでplot_formatをgnuplotに指定しておくと,gnuplot向けのデータファイルmaxout.gnuplotがホームディレクトリ上に生成されます. このデータファイルは, 先頭にplot_optionsの各項目に対応するgnuplotの命令が記入されたものです.

但し,gnuplot_curve_]styles と gnuplot_curve_]title は gnuplot の描画命令の plot や splot の命令行にその値が反映されます.

では, 最初に `plot2d` 函数による maxout.gnuplot の内容を解説しておきましょう.

plot2d 函数による maxout.gnuplot

```
set pm3d          /* gnuplot_pm3d が true の場合に記述 */

set log x         /* gnuplot_logx が true の場合に記述 */
set log y         /* gnuplot_logy が true の場合に記述 */

<gnuplot_preamble の内容>
plot '-' <gnuplot_curve_titles の内容> <gnuplot_curve_styles の内容>
<実データ>
```

ここで示す様に,gnuplot_pm3d や gnuplot_logx と gnuplot_logy が true の場合,plot 命令の前にこれらの設定が記述されます. 又,plot 命令の直前の行に, gnuplot_preamble の内容が書込まれます.

`plot3d` 函数の maxout.gnuplot も似た書式です.

plot3d 函数による maxout.gnuplot

```
set pm3d          /* gnuplot_pm3d が true の場合に記述 */

<gnuplot_preamble の内容>
splot '-' <gnuplot_curve_titles の内容> <gnuplot_curve_styles の内容>
<実データ>
```

`plot3d` 命令で生成した maxout.gnuplot も描画命令が plot ではなく splot になっている点を除

くと基本的な事は同じです.

これらのファイルの内容で,plot や splot が gnuplot の描画である事が何となく分るかと思います. これらの描画命令の前に,set から開始して,xlog,ylog や pm3d が続く命令文があります. これらは gnuplot で様々な設定を行う命令文で, set 命令で様々な設定を行い,show 命令でその設定内容を表示し,unset 命令で設定内容を無効にします.

これらの命令は非常に重要な命令です. 以下に,set 命令,show 命令と unset 命令の基本的な構文を示しておきます.

set 命令, show 命令, unset 命令

set ⟨ 項目 ⟩	⟨ 項目 ⟩ を有効にする
set ⟨ 項目 ⟩ ⟨ 設定内容 ⟩	⟨ 項目 ⟩ に設定を行う
show ⟨ 項目 ⟩	⟨ 項目 ⟩ の内容を表示
unset ⟨ 項目 ⟩	⟨ 項目 ⟩ を無効にする

set ⟨ 項目 ⟩ で ⟨ 項目 ⟩ と unset ⟨ 項目 ⟩ は互いに逆操作になります.

又, ⟨ 項目 ⟩ に色々な設定を行う場合は, その項目の設定内容によって構文が色々違います. この点については後の節で色々と紹介したいと思います.

さて,Maxima から単純に `plot2d` 函数や `plot3d` 函数を用いて maxout.gnuplot を生成しても, `set pm3d at bs` の様な gnuplot の命令文を埋込む事は出来ません.gnuplot から直接描画する場合は, この様な命令文を入れて replot 命令で再描画すれば良いのですが,Maxima から操作する場合は, この様な身軽な操作は出来ません.

その為, より高度なグラフ表示を行う為には,maxout.gnuplot に直接 gnuplot の命令文が書込まれる gnuplot_preamble に色々と設定する事になります. この事から,gnuplot_preamble の使いこなしが非常に重要である事が理解出来るかと思います.

この小節では,gnuplot_preamble の使い方について実例を示す事を目的としていますが, この項目に設定する値は gnuplot の命令文そのものである為, この項目を有効に生かす為には gnuplot の事を或る程度知っておく必要があります.

そこで, この小節では gnuplot の描画命令である plot,splot の解説を行い, それからグラフの諸設定に必要な事項, 基本的には set 命令でどの様な設定を行えば良いのかを幾つか解説しきたいと思います.

2.2 plot 命令による曲線の表示

plot 命令の基本的な構文

plot ⟨ 式 ⟩ axes ⟨ 軸のラベルを設定 ⟩ title ⟨ 文字列 ⟩ with ⟨ 曲線のスタイル ⟩
plot ⟨ 式 ⟩ title ⟨ 文字列 ⟩
plot ⟨ 式 ⟩
plot ⟨ データファイル名 ⟩ title ⟨ 文字列 ⟩ with *langle* 曲線のスタイル ⟩
plot ⟨ データファイル名 ⟩ title ⟨ 文字列 ⟩
plot ⟨ データファイル名 ⟩

plot の第一引数には gnuplot で表記可能な数式, 或いは表示すべきデータが含まれているデータファイル名を設定します.

ここで plot 命令に与える数式は基本的に変数 x の gnuplot の式に限定されます. 複数の式を同時に表示させる場合, 式をコンマで区切ったものを与えます. 例えば, 正弦函数と余弦函数を同時に表示したい場合, 式として, `sin(x),cos(x)` を与えます.

データファイル名は引用符, 或いは二重引用符で括られた文字列として与えます. 猶,Maxima で gnuplot で描画させるファイル名やラベルの指定の為に文字列を gnuplot の命令文に組込む場合,Maxima の文字列は二重引用符に限定される為, ファイル名やラベル等の gnuplot の命令文中の文字列は単引用符で括ったものを用います. 具体的には, 曲線の表題を `test` にしたければ,Maxima の大域変数 plot_options の項目 gnuplot_curve_titles への設定を `[gnuplot_curve_titles,"title 'test'"]` で行います.

あるファイルに plot 命令とデータ本体を記載する場合,plot 命令で指定するファイル名の箇所を `'-'` と記述すれば, plot 命令の次の行から開始するデータ列の読込を行い, ファイルの EOF を検出した時点で描画を行います. この方法が maxout.gnuplot で採用されています.

gnuplot の式とデータファイルの内容を混在して表示する事も可能です. この場合, 引数として, 数式とデータファイルをコンマで区切ったものを第一引数として引き渡します. 例えば, 先程の正弦函数と余弦函数に加え, データファイル test1 のデータを表示させたい場合, `plot sin(x),cos(x),'test1'` の様に入力します.

グラフの表題は title の後に文字列を指定します.Maxima では大域変数 plot_options の項目 gnuplot_curve_titles の内容がここに記入されます.

猶, この gnuplot_curve_titles の内容は標準入力を示す記号`'-'` の直後に置かれるので, 阿漕な使い方として, グラフデータを指定したり,gnuplot の式を入れる事が可能です. この事を試しに実行してみましょう.

```
plot2d(1/x,[x,1.0e-8,1.0e-5],
[gnuplot_curve_titles,"title 'Maxima',1/x title 'GNUPlot' "]);
```

この例では, $\frac{1}{x}$ のグラフを描きますが, 一方は Maxima で描き, もう一方は gnuplot の式として gnuplot に計算させて描画するものです. その結果を図 2.1 に示しておきます.

ここで,gnuplot の式は `[gnuplot_curve_titles,"title 'Maxima',1/x title 'GNUPlot'"]`

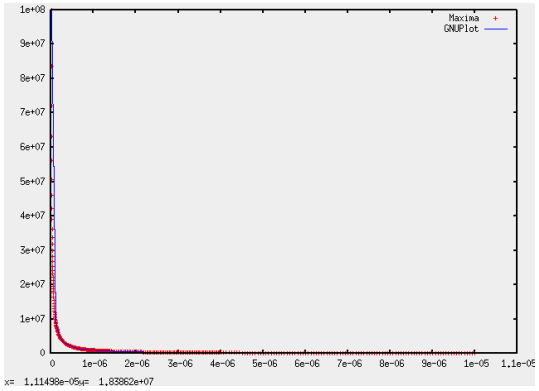


図 2.1: gnuplot_curve_titles の応用で gnuplot の式も描く

で与えている事に注目して下さい. この処理で生成された maxout.gnuplot のヘッダの部分を以下に示しておきますが, 上手く埋め込まれている事が分ります.

maxout.gnuplot の様子

```
plot '-' title 'Maxima',1/x title 'GNUPlot'  with lines 3
1.00000000E-8 100000000.
1.195117187500000100E-8 83673802.90897205
1.390234375000000400000000E-8 71930317.5049171
1.585351562500000600000000E-8 63077491.68411973
1.780468750000000500000000E-8 56164984.642386995
1.975585937500000700000000E-8 50617894.216510125
2.170703125000000600000000E-8 46068022.3141983
2.365820312500000500000000E-8 42268637.001568556
2.560937500000000600000000E-8 39048200.12202562
---- 以下略 ----
```

最後に, 曲線の書式は with の後に記入します.Maxima では gnuplot_curve_styles に with から開始する文字列を設定しておけば,maxout.gnuplot にその文字列が記入されます.

2.3 splot 命令による曲面の表示

splot 命令はデータや式が表現する曲面を描く命令です. 基本的な構文は plot 命令と同じものです.

splot 命令の基本的な構文

splot ⟨ 式 ⟩ title ⟨ 文字列 ⟩ with *langle* 曲面の様式 ⟩
splot ⟨ 式 ⟩ title ⟨ 文字列 ⟩
splot ⟨ 式 ⟩
splot ⟨ データファイル名 ⟩ title ⟨ 文字列 ⟩ with *langle* 曲面の様式 ⟩
splot ⟨ データファイル名 ⟩ title ⟨ 文字列 ⟩
splot ⟨ データファイル名 ⟩

この splot 命令でも plot 命令と同様に, 曲面の表題 (title で指定), そして曲面の様式 (実際はワイヤーフレームを構成する線分の様式に対応) の指定が行えます. ファイル名や文字列に関しては,plot 命令と違いはありません.

猶, 曲面の様式には,lines,point,linespoints,dots,impulses があります. この曲面の様式は曲面上に表示される網目に対するもので, 後述の pm3d に関連する項目には影響を与えません.

Maxima で扱う場合は上述の plot 命令と違いを意識する必要はあまり無いでしょう. 但し,gnuplot の 3.8 版より pm3d が標準になった事で,Maxima から利用では, この pm3d の様々な設定を色々行う必要が出てくるでしょう. 次に, この pm3d の基本的な設定について述べる事とします.

2.4 pm3d

gnuplot では pm3d を有効にする事によって, 面を貼った曲面表示が行えます. この pm3d は従来は gnuplot のパッチとして配布されていましたが,3.8 以降から標準で付属しています.

pm3d の設定

set pm3d set pm3d at b
set pm3d at s
set pm3d at t
set pm3d map
unset pm3d
show pm3d

最初の `set pm3d` で pm3d を有効にします.Maxima の [gnuplot_pm3d,true] がこの設定に対応します.Maxima の大域変数 plot_options には, その他に pm3d に関係する項目はありませんが, 他にも細かな指定が出来ます. 例えば, 面を張った曲面の表示だけではなく, 曲面と底面や上部への投影と一緒に表示したりする事が可能です. この投影の設定は pm3d の設定で,at の後の文字列で指定を行います.

先ず,`set pm3d at b` で曲面を底面に投影します. 後述の map に似ていますが, こちらは斜め上から曲面と投影の両方を眺める形になります. 猶, 曲面はワイヤーフレーム表示になります.

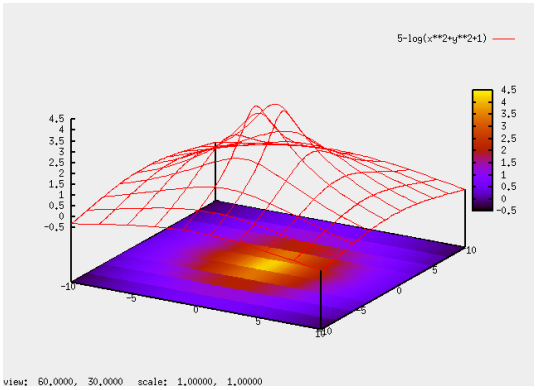


図 2.2: set pm3d at b の場合

これに対し,`set pm3d at s`を実行すると, 今度は曲面に面を張り, この面には等高線に沿って連続的に変化する色彩が付けられたものが表示されます. この場合, 曲面の射影は実行されません. 又,gnuplot で hidden3d を有効にしたり,surface を無効にしていなければ, 曲面とワイヤーフレームが同時に表示されています.

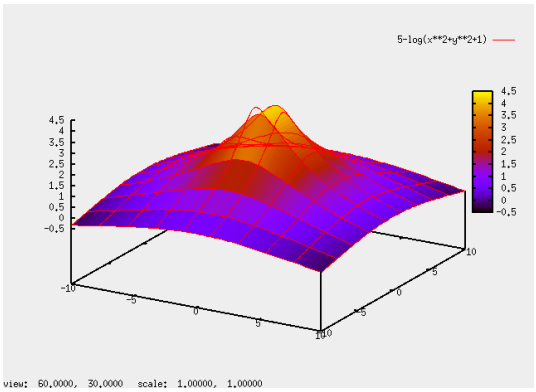


図 2.3: set pm3d at s の場合

それから,`set pm3d at t`で曲面の等高線表示の射影を今度は上面に行います. 射影が上側に行われる事を除くと,b の場合と違いはありません.

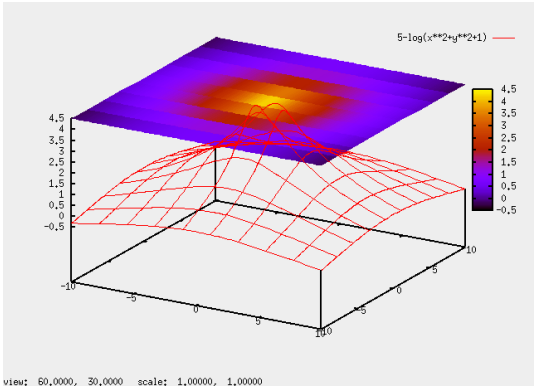


図 2.4: set pm3d at t の場合

最後に,`set pm3d map`にした場合を示しましょう.

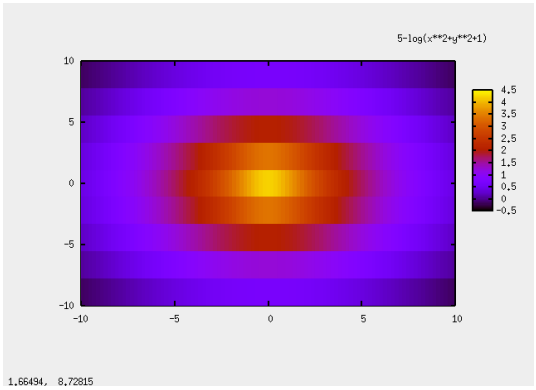


図 2.5: set pm3d map の場合

これらの b,s,t を組合せる事も可能です. 例えば, set pm3d at bs の様に設定します. この場合, 最初に底面への射影を描き, 次に曲面を描く事を意味します. これは,at の後の文字列の左から指定された処理を実行する為です.

それでは gnuplot で splot x*y の描画を, set pm3d at bs を行った場合を図 2.6 に示し, set pm3d at sb を行った場合の結果を図 2.7 に示しておきましょう.

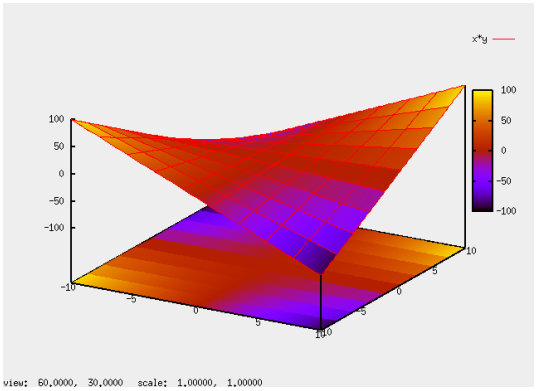


図 2.6: set pm3d at bs の場合

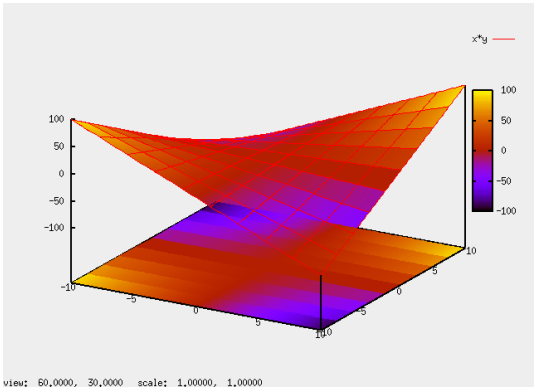


図 2.7: set pm3d at sb の場合

これらの絵から分る様に, 図 2.7 の図は底部の絵の一部が上にある筈の曲面に覆い被さっている事が分ります. これは前述の様に,bs の場合は底面から描く為に, 曲面が底面を覆う事になり,sb の場合は, 逆に曲面から描く事になるので, 重なる部分は, 底面が覆ってしまうからです. この様に, 射影を描いた時点で覆い被さりが気になる場合は, 設定の順序を間違えていないか確認すると良いでしょう.

この底面や上面への曲面の投影図の位置を調整する事も可能です. この場合, ticslevel で数値を指定して行います.

投影面の位置の変更

set ticslevel ⟨ 数値 ⟩
show ticslevel

ここでの数値は $\frac{\text{投影面の } Z \text{ 座標} - \text{曲面の最高値}}{\text{曲面の最低値} - \text{曲面の最高値}}$ に等しくなります. 則ち,0 を指定した場合, 投影面は曲面の底辺にあり,-1 を指定すると投影面は曲面の頂部に配置される事になります.

これを確認する為に, 以下に $1 - \log\left(x^2 + y^2\right)$ のグラフを描いた例を示しておきます.

ここで図 2.8 では set pm3d at bs として曲面が投影面を覆う様にし, 図 2.9 では set pm3d at sb として投影面が曲面を覆う様に設定しています.

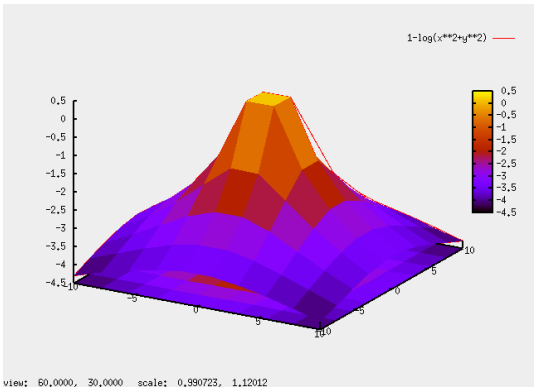


図 2.8: ticslevel が 0 の場合

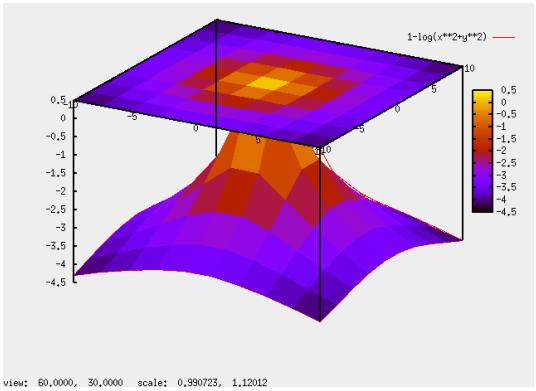


図 2.9: ticslevel が-1 の場合

2.5　等高線の階調の変更

この pm3d を有効にしていると, 曲面は等高線状に色付けられています. この階調は変更が可能です. この階調の変更は palette で行います.

palette の変更

set palette color
set palette color positive
set palette color nagative
set palette gray
set palette gray positive
set palette gray nagative
set palette rgbfomulae ⟨ 数値$_1$⟩, ⟨ 数値$_2$⟩, ⟨ 数値$_3$⟩
show palette

ここでは以下の命令文を実行した結果を,palette の指定の違いで比較しましょう.

```
set isosample 50;
set hidden3d;
splot 10-log(sqrt(x**2+y**2+1));
```

ここでは hidden3d を設定する事で曲面上の網目を非表示にし,isosample を 50 とする事で,X 軸,Y 軸側の分割を各々50 にして曲面の解像度を上げています. 因に,gnuplot では描画した曲線や曲面の設定を変更しただけであれば, replot 命令を実行するだけで再描画を行います.

先ず, set palette color positive の設定を行って再描画したものが図 2.10 に示すグラフになります. この設定が通常の等高線の階調になります.

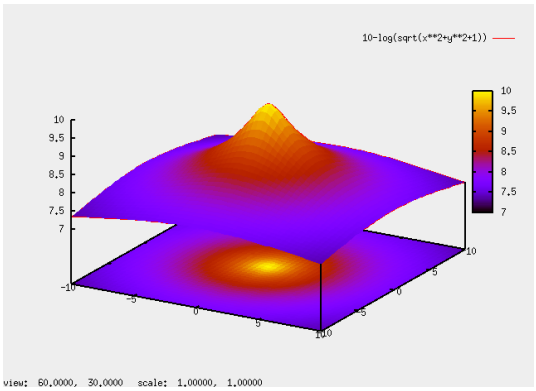


図 2.10: set palette color positive;replot

この図に示す様に, 明い色の方が Z 座標が高くなります. もしも, 階調を逆にしたければ, 今度は color の後に negative を指定して palette の再設定を行い, それから描画を行えば良いのです. 図

2.11 に階調を逆に設定した場合の再描画の様子を示しています.

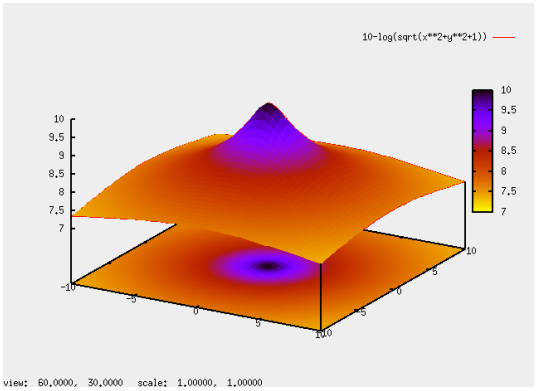


図 2.11: set palette color negative;replot

階調には通常の color の他に白黒の gray 等があります.gray を指定した場合の様子を以下の図 2.12 に示しておきます.

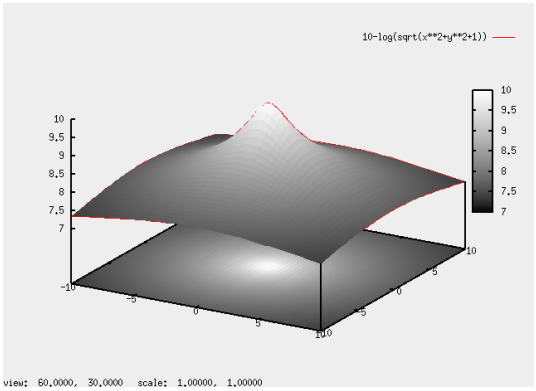


図 2.12: set palette gray positive;replot

gray の場合も, 逆の階調にする場合に gray の後に negative を指定して, palette を設定するだけで済みます,

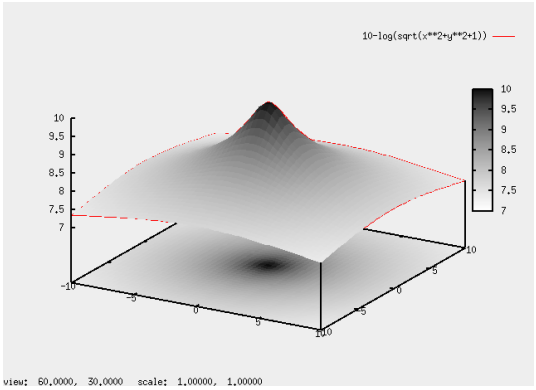


図 2.13: set palette gray negative;replot

この palette は他に色々と指定が可能です. 例えば,rgbfomulae を使って次の図 2.14 に示す様な階調も得られます.

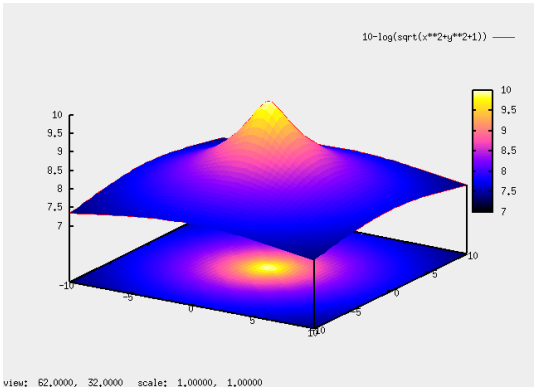


図 2.14: set palette rgbformulae 30,31,32;replot

2.6 視点の変更

gnuplot で視点は view で設定されます.

view の設定

set view ⟨X 軸の回転角 ⟩,⟨Z 軸の回転角 ⟩ ⟨ 拡大率 ⟩,⟨Z 軸方向の拡大率 ⟩
set view ⟨X 軸の回転角 ⟩,⟨Z 軸の回転角 ⟩
set view map
show view

ここで,view の初期値は X 軸回りの回転角が 60 度,Z 軸回りが 30 度で, 各軸方向の拡大率は 1 になっています.

又,view には map があります. こちらは効果としては,`set pm3d map` と同じで, 曲面を真上から眺める形になります. これは X 軸回りと Z 軸回りの回転角が 0 度の場合に一致します.

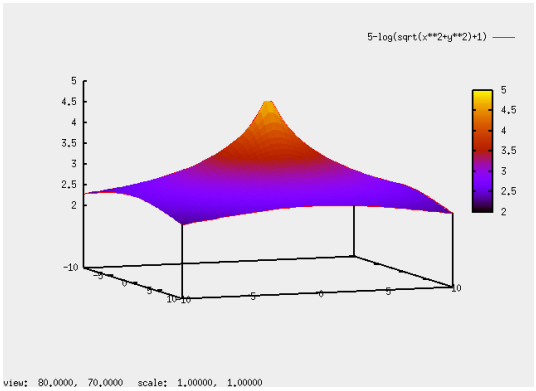


図 2.15: set view 80,70,1,1

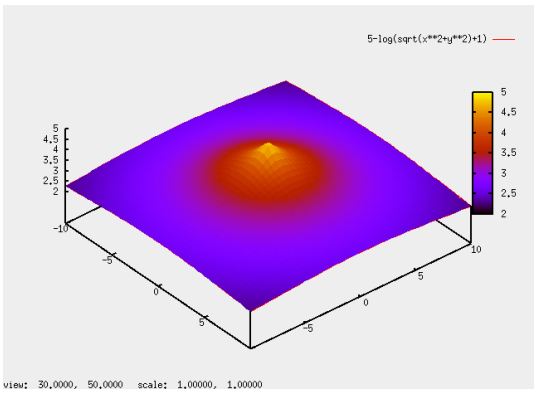


図 2.16: set view 30,50,1,1

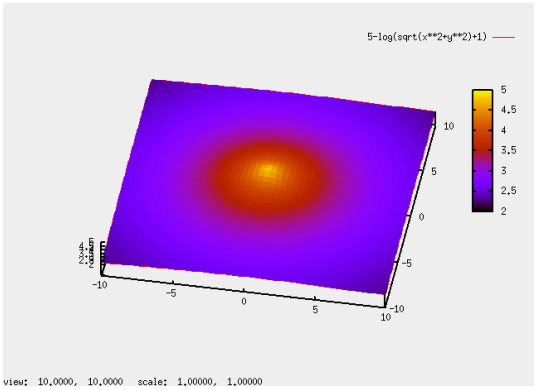


図 2.17: set view 10,10,1,1

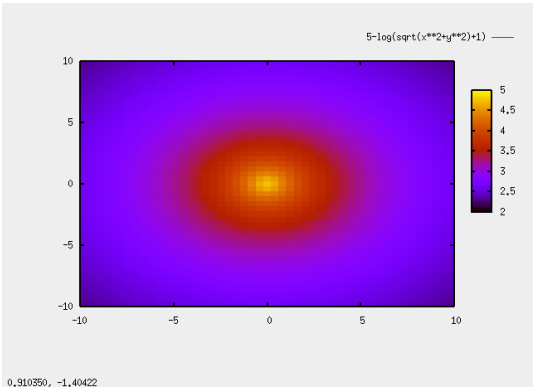


図 2.18: set view map

2.7 cbrange と cblabel

pm3d を有効にしていると, 右側に曲面の階調を示す箱が表示されています. この階調の調整は cbrange で行いますが, その構文は xrange と同じです. 先ず,chrange は曲面の高さと階調の対応関係を指定するものです. 曲面の最低値と最高値が, この cbrange の下端と上端であれば, きめ細かな階調表示になりますが, この cbrange の幅が曲面の高低差と比較して大き過ぎたりすれば, 曲面の階調が殆ど出ない, のっぺら坊な曲面になってしまいます.

cbrange の設定

set cbrange [⟨ 下限 ⟩ : ⟨ 上限 ⟩]
unset cbrange
show cbrange

この cbrange の設定で, 階調の範囲を指定する方法は,MATLAB のベクトルの表記の様な [a:b] を用います. これは通常のリスト [a,b] とは異なるので注意が必要です.

更に, 図の右側に通常出現しているこの階調ボックスにラベルを表示させる事も可能です. このラベルの表示も X 軸等の各軸にラベルを配置する xlabel 等と同様の構文になります.

clabel の設定

set cblabel ⟨ 文字列 ⟩ ⟨x 座標 ⟩, ⟨y 座標 ⟩
set cblabel ⟨ 文字列 ⟩
unset cblabel
show cblabel

ここで, ⟨x 座標 ⟩, ⟨y 座標 ⟩ は何も指定しない位置を基準とした文字列表示位置を指定します. ここでの座標は縦上方向を Y 軸正方向, 横軸右向きを X 軸の正方向とするもので,1 で一文字分になりますが, この値は表示フォントやターミナル等の環境で異なります.

実例として先程の図 2.14 に対して以下の処理を行ったものを図 2.19 に示しておきます.

```
set cblabel 'Height' -13,6;
set cbrange [8:10];
replot
```

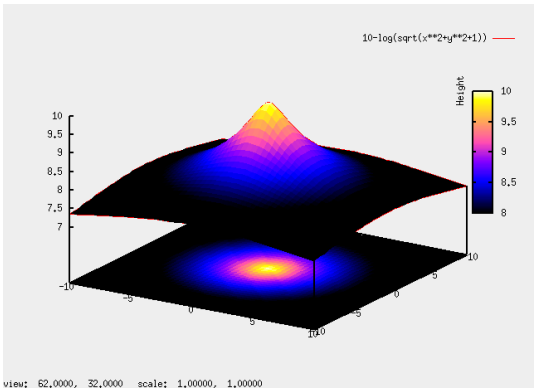


図 2.19: set cbrange [8:10];replot

ここでは cbrange を狭くした為, 高さが 8 以下は真っ黒になっています.
次に, 以下の処理を行ったものを図 2.20 に示します.

```
set cblabel 'Height' -13,6;
set cbrange [5:10];
replot
```

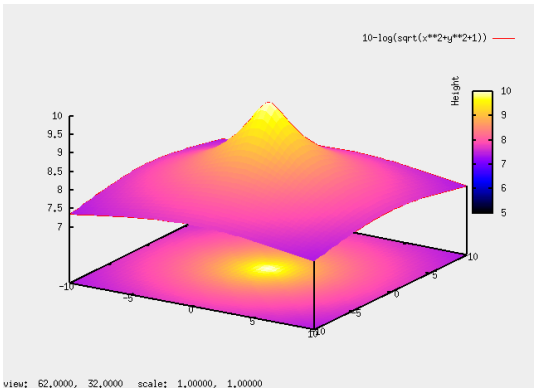


図 2.20: set cbrange [5:10];replot

今度は下に長く取った為に, 曲面全体が明るくなっている事が分ります.

2.8 陰線処理

gnuplot で三次元グラフを行うと, 通常は陰線処理を行っていません. 陰線処理を行う為には,hidden3d を有効にします. 因に, この hidden3d は pm3d が無効でも, 有効になっていても無関係に使えます.

又,pm3d が有効の場合, unset surface を実行する事で pm3d の曲面上の網目を非表示にする事が可能です. 但し, こちらは pm3d の曲面の設定を無効にする副作用があります.

ここでは set hidden3d と unset surface の違いを確認しましょう. 比較で用いるグラフは以下の設定とします.

```
set pm3d at bs
set cbrange [-0.05:1]
splot 1/(x**2+y**2+1)
```

この処理結果を初期状態とし, 図 2.21 にそのグラフを示しておきます.

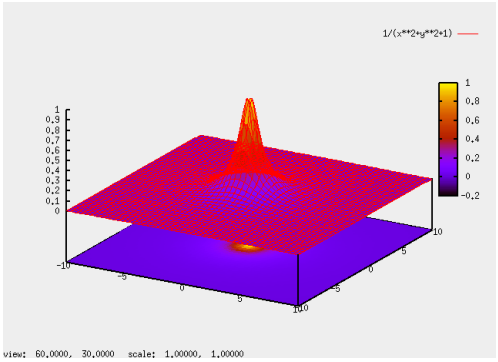


図 2.21: 初期状態

次に,set hidden3d を実行した後に replot した結果を図 2.22 に示します.

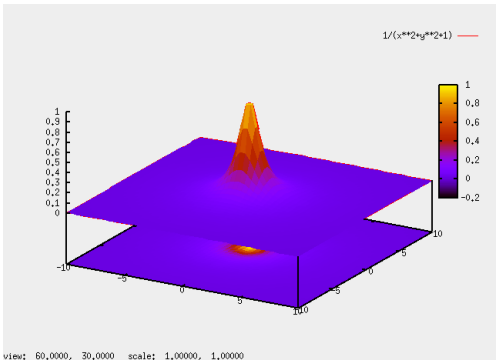


図 2.22: set hidden3d

最後に,unset surface を実行して replot した結果を図 2.23 に示しておきます.

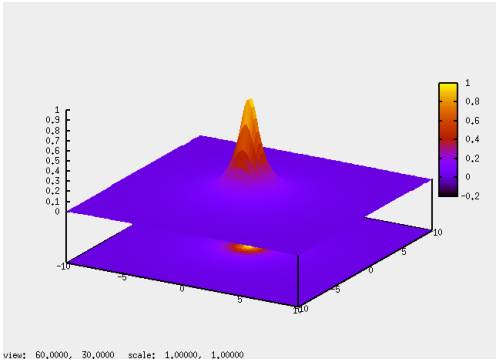


図 2.23: unset surface

hidden3d と unset surface の違いは hidden3d では稜線や境界に線分が残っていますが,unset surf では一切の線分が消えている事です.

猶,gnuplot の式を描画する場合, 陰線処理を有効にした場合, 解像度の問題から, 極が埋没する傾向が強くあります. これは Maxima の様に高精度で計算したデータを用いる場合ではグラフの形が大きく異なるので問題が良く分ります.

実際に確認してみましょう. ここで, 描く函数は $\log\left(x^2y^2\right)$ とします.

最初に gnuplot で安易に `splot log(x**2*y**2)` を実行した結果を図 2.24 に示します. こう見ると結構綺麗な曲面ですね.

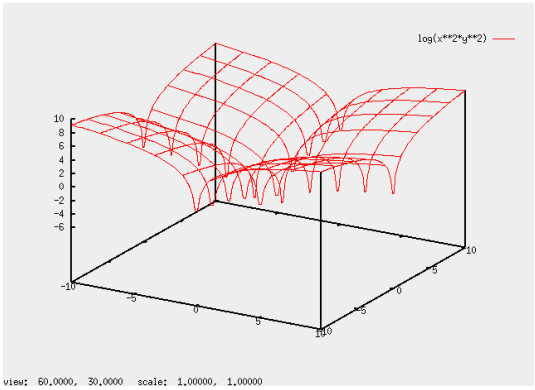


図 2.24: splot log(x**2*y**2)(その 1)

次に set pm3d at s で splot log(x**2*y**2) を実行した結果を図 2.25 に示しておきます. この図からは, 面を張ったが為に, 粗さが目立っています.

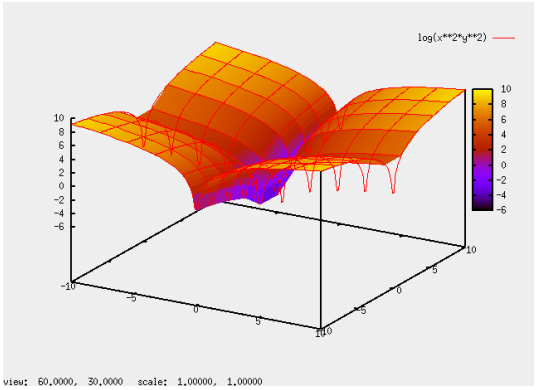


図 2.25: splot log(x**2*y**2)(その 2)

そこで, set isosample 100 に変更して, より細かな分割で曲面を replot した結果を図 2.26 に示します.

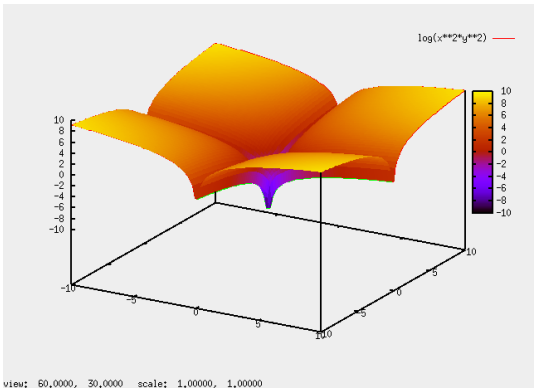


図 2.26: splot log(x**2*y**2)(その 3)

こう見ると, 原点付近が極みたいですね.

では, この函数を Maxima から表示してみましょう.gnuplot と同様の細かさを得る為に, ここでは以下の処理を Maxima に実行させます.

Maxima での描画処理

```
plot3d(realpart(log(x^2*y^2)),[x,-10,10],[y,-10,10],
[grid,100,100],[gnuplot_preamble,"set pm3d at s;unset surf"]);
```

この処理結果を図 2.27 に示しておきます.

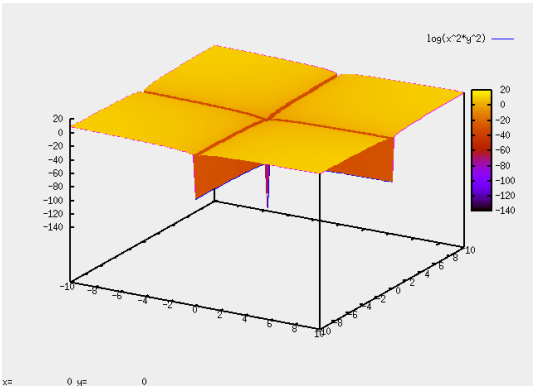


図 2.27: Maxima から $\log\left(x^2y^2\right)$ の描画

この図になると, 今迄の gnuplot の式のものと比べて谷と山が遥かに深くなっています. 何故, 同じ gnuplot でも Maxima から描くと違った形の絵になるのでしょうか？単純にスケールの違いでしょうか？

これは,Maxima では maxout.gnuplot を生成する為に数値データを生成している為に, splot で式を記述する方法よりも良好な結果が得られているのです.

実は式を良く見れば分る簡単な事ですが,X 軸上の点と Y 軸上の点が函数の極になっています. 即ち,gnuplot の表示は大人し過ぎるのです. この意味では,Maxima の処理の方がまだ雰囲気が良く出ていると言えます.

さて, この式を Maxima で描画する際には,Maxima から色々質問が出ています.

$\log\left(x^2y^2\right)$ の Maxima での描画処理

```
(%i6) plot3d(realpart(log(abs(x))+log(abs(y))),[x,-10,10],[y,-10,10],
[grid,100,100],[gnuplot_preamble,"set pm3d at s;set hidden3d;"]);
Is  x  zero or nonzero?

nonzero;
Is  y  zero or nonzero?

nonzero;
(%o6)
```

ここで Maxima が聞いている事は,x と y が零か零でない数値であるかを尋ねているのです. この様な事を Maxima が聞いて来る時の多くは,x や y の何れかが零になると非常に都合の悪い事が生じる可能性がある事を示唆しています. この様な質問が出た場合には, 函数の定義域に関して吟味する必要があるのです. この点から,Maxima の plot.lisp で記述されている内容の方が gnuplot よりも遥かに安全であると言えます.

2.9 曲線と曲面の細かさの指定

gnuplot の式を表示して粗さが目立つ場合, 曲線の場合は sample , 曲面の場合は isosamples の設定を大きなものに変更します.

sample と isosamples の設定

set isosample ⟨ 数値$_1$⟩,⟨ 数値$_2$⟩
set sample ⟨ 数値$_1$⟩
set isosamples ⟨ 数値$_1$⟩,⟨ 数値$_2$⟩
set isosamples ⟨ 数値$_1$⟩
show isosamples

sample と isosamples には二つの数値か一つの数値を指定します.sample の初期値は 100 で, isosamples の初期値は 10 です.

sample の場合,X 軸上の総点数がこの sample で指定した値になります. 実例として, 以下で描画した $\sin\left(\frac{1}{x}\right)$ のグラフの sample を変更して replot したものを示します.

```
set xrange [-1:1]
plot sin(1/x)
```

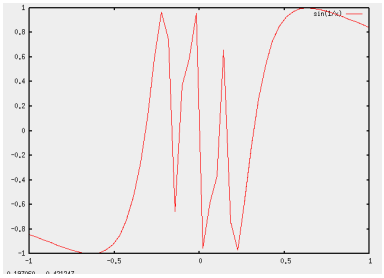


図 2.28: set sample 50 の場合

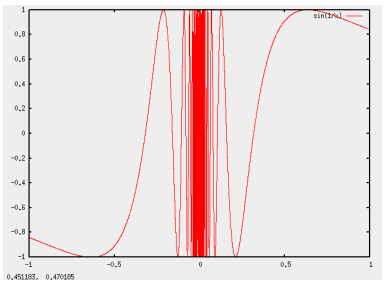


図 2.29: set sample 10000 の場合

isosamples の場合, ⟨ 数値$_1$⟩,⟨ 数値$_2$⟩ で X 軸と Y 軸の分割数を定め,⟨ 数値$_1$⟩ だけが指定された場合には,X 軸も Y 軸も, この分割数になります.

但し,Maxima から曲線や曲面を描画する場合,`plot2d` 函数や `plot3d` 函数に与える Maxima の式から gnuplot の描画に必要な数値データを計算してしますが, この計算の際に大域変数 plot_options の nticks や grid に設定した値が反映されます. その為,gnuplot_preamble で sample や isosample を指定しても, この設定は gnuplot の式の描画にだけ有効な為, 直接役には立ちません. とは言え,gnuplot_curves_titles を使って,gnuplot の式を埋込む場合には有効です.

2.10　描画の領域設定

plot 命令や splot 命令で描画する gnuplot の式に対し, その描画する範囲は xrange, yrange や zrange で指定します.

描画する領域を指定する場合

set xrange [⟨ 下限 ⟩:⟨ 上限 ⟩ ⟨ オプション$_1$⟩ ⟨ オプション$_2$⟩
set xrange [⟨ 下限 ⟩:⟨ 上限 ⟩ ⟨ オプション$_1$⟩
set xrange [⟨ 下限 ⟩:⟨ 上限 ⟩]
set xrange restore
show xrange
set yrange [⟨ 下限 ⟩:⟨ 上限 ⟩ ⟨ オプション$_1$⟩ ⟨ オプション$_2$⟩
set yrange [⟨ 下限 ⟩:⟨ 上限 ⟩ ⟨ オプション$_1$⟩
set yrange [⟨ 下限 ⟩:⟨ 上限 ⟩]
set yrange restore
show yrange
set zrange [⟨ 下限 ⟩:⟨ 上限 ⟩ ⟨ オプション$_1$⟩ ⟨ オプション$_2$⟩
set zrange [⟨ 下限 ⟩:⟨ 上限 ⟩ ⟨ オプション$_1$⟩
set zrange [⟨ 下限 ⟩:⟨ 上限 ⟩]
set zrange restore
show zrange

cblabel の小節でも解説しましたが, ここで領域の設定は MATLAB のベクトルの定義に似た書式となります. 即ち,[-1:1] と記述し,[-1,1] でない事に注意が必要です. ここで下限の初期値は-10, 上限の初期値は 10 になっています. その為, 何も指定せずに plot 命令や splot 命令で描画する時は, この領域で描画が実行されます.

この領域の設定には, オプションとして軸の向きを逆にする reverse, 元に戻す noreverse があります. 早速,reverse を plot と splot で試してみましょう.

```
set xrange [0:10] reverse
plot cos(x)
```

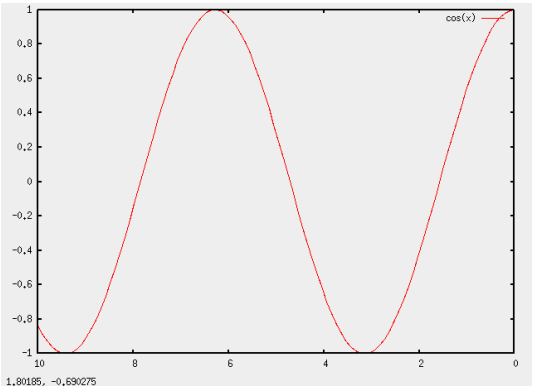


図 2.30: plot cos(x)

```
set zrange [0:10] reverse
splot sqrt(x**2+y**2)
```

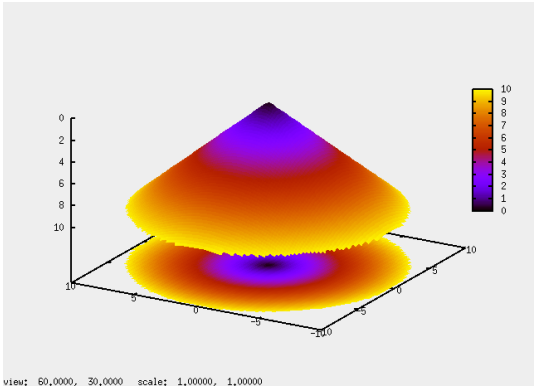


図 2.31: splot sqrt(x**2+y**2)

図 2.30 では X 軸の向きが逆になっていますね. 又, 図 2.31 の場合は,Z 軸が通常の逆になっています.

猶, 領域を初期値に戻したければ,`set xrange restore` の様に区間を記述せずに restore を指定すれば良いのです.

2.11　ラベル表示と注釈に関連する事項

gnuplot の便利の良さの一つに, 曲線や曲面に色々とラベルや注釈を入れる事が容易な事が挙げられるでしょう. 実際, この辺の機能は非常に豊富にあり, この小冊子で把握出来る様なものではありません. その為, ここでは本当に一部の機能の解説に留めておきます. 詳細は文献 [11] を参照して下さい.

では手始めに座標軸のラベルの指定について解説します.

2.11.1　軸のラベル指定

軸のラベルを指定する場合

set xlabel ⟨ 文字列 ⟩ ⟨X 座標 ⟩,⟨Y 座標 ⟩
set xlabel ⟨ 文字列 ⟩
show xlabel
set ylabel ⟨ 文字列 ⟩ ⟨X 座標 ⟩,⟨Y 座標 ⟩
set ylabel ⟨ 文字列 ⟩
show ylabel
set zlabel ⟨ 文字列 ⟩ ⟨X 座標 ⟩,⟨Y 座標 ⟩
set zlabel ⟨ 文字列 ⟩
show zlabel

先ず,xlabel,ylabel,zlabel は X 軸,Y 軸, そして Z 軸のラベルを指定します. 文字列の後に入れる X,Y 座標は, 通常の文字列が表示される位置を基準として, その位置から何文字分移動するかを指定します. 但し, ここでの値はフォントの指定等で異なる値になります.

2.11.2 曲線の表題表示

複数の曲線を表示した際に, 右上側に線分と曲線の表題が並んで表示されています. gnuplot ではこれを key と呼んでおり, この key の位置,key に表示する線分の長さ等の指定が可能です.

最初に key の表示位置を設定する構文を以下に示しておきます.

グラフの key の位置を設定

```
set key right
set key left
set key top
set key top left
set key top right
set key bottom
set key bottom left
set key bottom right
set key outside
set key below
```

これらの設定は,key のグラフ上での位置を指定するものです. 更に,key の表示自体を調整する事も可能です.

グラフの key を設定

```
set key reverse
set key noreverse
set key samplen ⟨ 線分の長さ ⟩
set key box linetype ⟨ 線分の様式 ⟩ linewidth ⟨ 線分の幅 ⟩
unset key box
show key
```

ここで key に対する reverse は線分とその名称の羅列の順序を逆にするものです. noreverse は初期状態に戻します.

samplen は key で表示されている線分の長さの調整に用います.

set key box linetype は,key の線分の様式や幅を変更するものです. 最後に key を表示したくなければ,`unset key box`とします.

では, 各軸のラベルと key の設定例を以下に示しておきます.

```
set xlabel 'Time -sec-'
set ylabel 'Height -mm-' 0,40
set key outside
set key reverse
set key samplen 10
plot sin(x),cos(x),sin(x)/x
```

この例では,X 軸のラベルを'Time -sec-',Y 軸のラベルを'Height -mm-' とし, Y 軸のラベルは 40 文字程上側に配置します. そして,key をグラフの外側に置き, この key での線分の長さを 10 とし, ここでの線分と曲線名の順番を reverse で逆にしています.

この様子を図 2.32 に示しておきます.

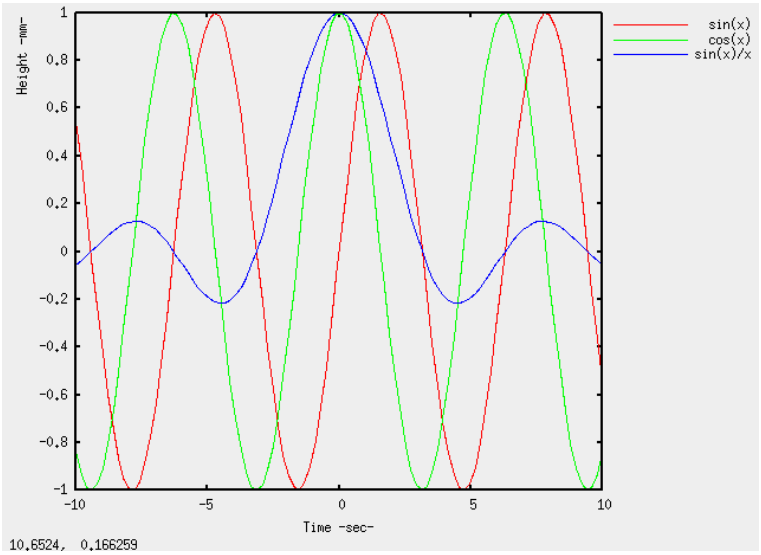


図 2.32: 各軸のラベルと key の設定例

2.11.3 注釈と矢印の追加

gnuplot ではグラフに注釈を入れる事が可能です. この注釈は label で設定します.

註釈の設定

set label ⟨ 文字列 ⟩,⟨ 注釈番号 ⟩ at ⟨X 座標 ⟩,⟨Y 座標 ⟩
set label ⟨ 注釈番号 ⟩,⟨ 文字列 ⟩ at ⟨X 座標 ⟩,⟨Y 座標 ⟩
set label ⟨ 文字列 ⟩
set label ⟨ 注釈番号 ⟩ ⟨ 位置合せ ⟩
set label ⟨ 注釈番号 ⟩ font ⟨ フォント名 ⟩,⟨ 大きさ ⟩
set label ⟨ 注釈番号 ⟩ front
set label ⟨ 注釈番号 ⟩ back
set label ⟨ 注釈番号 ⟩ textcolor ⟨ 表示色 ⟩
set label ⟨ 注釈番号 ⟩ rotate by ⟨ 角度 (deg)⟩
set label ⟨ 注釈番号 ⟩ norotate
set label ⟨ 注釈番号 ⟩ point pointsize ⟨ 数値 ⟩
unset label ⟨ 注釈番号 ⟩
unset label
show label

gnuplot では,set label 文を ⟨ 注釈番号 ⟩ を指定せずに実行すると, 自動的に注釈に適切な番号を割当てます. その後の註釈の回転, 内容やフォントの変更等の処理はこの注釈番号を指定して行わなければなりません. 指定しない場合には新規の註釈が生成されるだけです.

ここで注釈の座標は先程のグラフの点の位置に対応します. 猶, 文字列の左寄, 中寄, 右寄といった位置決めは各々left,center,right を指定する事で行います.

注釈を XY 面内で回転させる事も可能です. この場合,rotate by で行いますが,by の後に指定する角度の単位は度になります.

では実例として, 以下に示す処理を gnuplot でさせてみましょう.

```
set label 1 ’ Comment at Origin’ at 0,0
set label 2 ’ Comment at (1,1)’ at 1,1
set label 1 point pointsize 6
set label 2 point pointsize 2
set label 2 rotate by 90
plot sin(x)*2
```

最初に注釈を原点と (1,1) に配置しています. それから註釈番号 1 の註釈に対して大きさが 6 の点を配置しています. 同様に, 番号 2 の註釈には大きさが 2 の点を配置します. それから番号 2 の註釈を 90 度回転させています. 最後は,plot 命令で $2\sin x$ 函数の描画を行います.

この結果を図 2.33 に示しておきます.

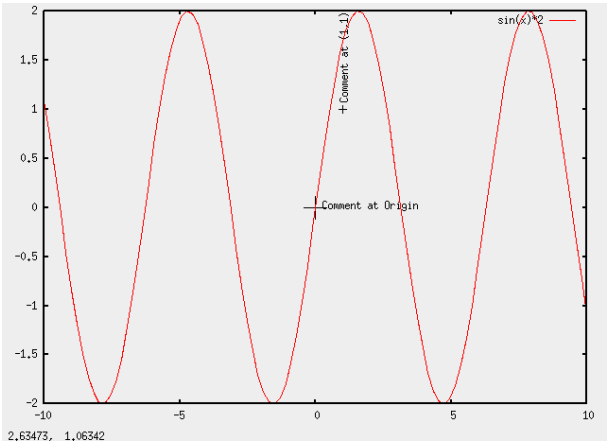


図 2.33: 注釈の設定例 (その 1)

或る注釈を非表示にしたければ, unset label ⟨ 注釈番号 ⟩ と入力する事で対処します. ここでは図 2.33 の例に対し, 番号 2 の註釈を `unset label 2` を実行した後に replot した結果を図 2.34 に示しておきます.

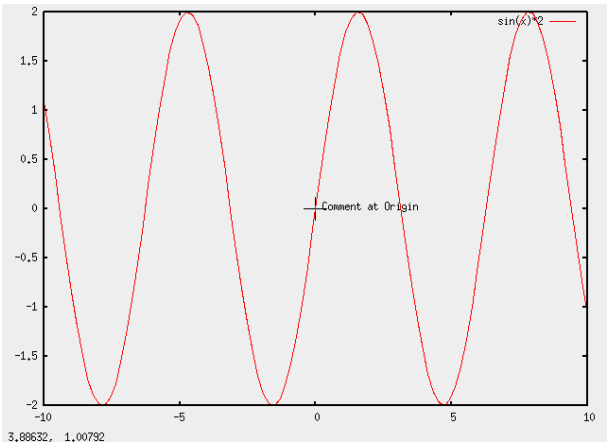


図 2.34: 注釈の設定例 (その 2)

番号 2 の註釈が実際に消えている事が分ります.

2.11.4 矢印の追加

gnuplot では矢印をグラフ中に追加する事も可能です. この矢印は註釈と併用する事で効果が上ります.

arrow による矢印の設定

set arrow ⟨ 番号 ⟩ from ⟨X 座標 ⟩,⟨Y 座標 ⟩ to ⟨X 座標 ⟩,⟨Y 座標 ⟩
set arrow ⟨ 番号 ⟩ from ⟨X 座標 ⟩,⟨Y 座標 ⟩
set arrow ⟨ 番号 ⟩ to ⟨X 座標 ⟩,⟨Y 座標 ⟩
set arrow ⟨ 番号 ⟩ size ⟨ 矢の長さ ⟩,⟨ 矢の角度 ⟩
set arrow ⟨ 番号 ⟩ head
set arrow ⟨ 番号 ⟩ heads
set arrow ⟨ 番号 ⟩ nohead
set arrow ⟨ 番号 ⟩ filled
set arrow ⟨ 番号 ⟩ nofilled
unset arrow
show arrow

やっている事は label を用いた註釈の設定に似ています. 先ず, 図形としての矢印の設定は `set arrow 1 from 0,0 to 1,1` の様に始点と終点を決めるだけで生成されます. この時, 矢印の向きは from から to に向きます. 猶,from の点, 或いは to の点が原点の場合は省略しても構いません.

次に arrowsize で矢印の頭の部分を調整する事が可能です. ここでの ⟨ 矢の長さ ⟩ は矢印の頭の矢の長さで, ⟨ 矢の角度 ⟩ は頭の矢の角度になります. この角度は度数です. filled を指定すると, 矢印の矢の部分が塗り潰されます.

では実際に以下の様に入力してどの様なグラフになるか確認してみましょう.

```
set arrow 1 from 0.4,-1 to 1.1,-1
set arrow 2 from 0.9,1 to 1.1,1
set arrow 3 from  1,-1 to 1,1
set arrow 1 nohead
set arrow 2 nohead
set arrow 3 size 0.1,30 filled heads
set label 1 ’  Width’ at 1.05,0
plot cos(6.283*x);
set xrange [0:2]
set yrange [-2:2]
plot cos(6.283*x);
set arrow 3 filled
plot cos(6.283*x);
```

ここでは最初の三行で矢印を生成しています. 次の三行で各矢印の性質を指定しています. ここ

で, 第三の矢印のみ両端に矢を付け, その他は棒にします. そして, 第三の矢印の矢の部分を大きくして塗り潰してしまいます. この矢印の設定は, この様に一度に設定しなければ, 最後に入力した設定だけが有効になり, 他は初期化されてしまうので注意が必要です.

この結果を図 2.35 に示しておきます.

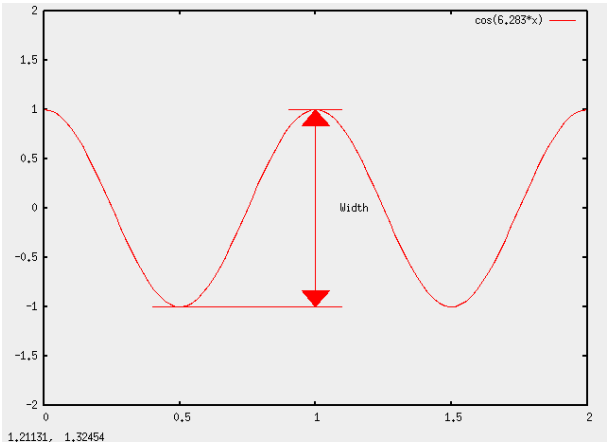


図 2.35: 矢印と註釈の設定例

以上で gnuplot の簡単な使い方の解説を終えます. 以降は,Maxima での使い方について幾つかの実例の解説を行います.

第3章　Maximaからgnuplotを使う

最後に,plot2d 函数と plot3d 函数で gnuplot を用いた描画例を幾つか示しておきます.

Maxima の plot2d 函数や plot3d 函数で用いる大域変数 plot_options には,gnuplot に関連する項目が沢山あります. しかし, これらの項目は決して gnuplot の機能を遍く網羅するものではありません. これは gnuplot が非常に高機能である為でもあり, 実際、その機能に応じた gnuplot の項目を増やしたとすれば, オプションが山の様になってしまい. 結局, 何が何だか分らなくなってしまうでしょう.

gnuplot は非常に機能が高いアプリケーションですが,plot 命令と splot 命令で描画,replot で再描画を行い,set 命令で諸設定を行う仕組みになっています. Maxima から gnuplot を用いて描画する場合, 基本的な事は,plot2d 函数や plot3d で処理してくれるので, 後は,set 命令文をどの様に組込むかという問題に帰着されます.Maxima では大域変数 plot_options の項目の gnuplot_preamble に gnuplot の制御文を設定する方法を採用しています.

さて, 今迄の復習になりますが, グラフの表題を mike,X 軸と Y 軸のラベルを各々X,Y としたい場合,Maxima でどの様に入力すれば良いかを考えましょう.

ここで,gnuplot ではグラフの表題は set title で指定し, X 軸と Y 軸のラベルの設定は, 各々 set xlabel と set ylabel で行います. 最終的には, これらの gnuplot の命令文をセミコロンで区切り, 全体を二重引用符で括ったものを gnuplot_preamble に設定します.

この処理と描画の実行を以下に示しておきます.

```
(%i1) nekoneko:"set title 'mike';set xlabel 'X';set ylabel 'Y';";
(%o1)           set title 'mike';set xlabel 'X';set ylabel 'Y';
(%i2) plot2d(sin(x)/x,[x,0,10],[gnuplot_preamble,nekoneko]);
```

この処理の結果得られるグラフを図 3.1 に示しておきます.

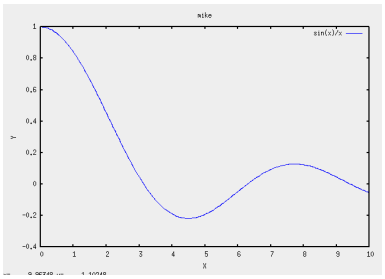


図 3.1: gnuplot_preamble で設定した結果

念の為, この例題の maxout.gnuplot の先頭部分を確認しておきましょう.

maxout.gnuplot のヘッダ部分

```
set title 'mike';set xlabel 'X';set ylabel 'Y';
plot '-' title 'sin(x)/x' with lines 3

2.441406250000E-4 0.9999999900658926
4.88281250000E-4 0.9999999602635706
---省略---
```

gnuplot_preamble の内容は gnuplot の描画命令である plot 命令や splot 命令の直前にそのまま埋込まれます. 従って,plot 命令や splot 命令のオプションとして設定される曲線の表題や曲線の様式を gnuplot_preamble で設定したとしても, 描画の際に,plot 命令や splot 命令のオプションが優先されるので, 苦労して設定しても意味はありません. 曲線の表題や曲線の様式は各々,gnuplot_curve_titles と gnuplot_curve_style に設定すべきです.

では, 以下の様な gnuplot_curve_style への設定を行って描画させてみましょう.

```
plot2d(sin(x),[x,-1,1],[gnuplot_curve_styles,"with impulse"])
```

この時,maxout.gnuplot の plot 命令付近は以下の様になっています.

gnuplot_curve_styles の設定

```
plot '-' title 'sin(x)' with impulse
-1.      -0.8414709848078965
-0.975      -0.8277018881672576
--- 略 ---
```

この結果は図 3.2 の様になります.

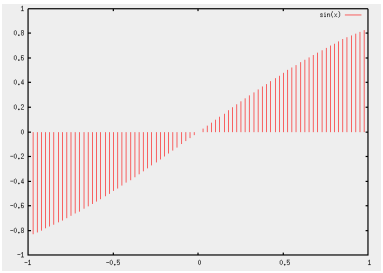


図 3.2: gnuplot_preamble で設定した結果

他に,gnuplot_preamble に入れても効果が無いものとして,sample や isosamples を挙げておきます. これは maxout.gnuplot に引渡されるデータは数値データであって, gnuplot の式ではないからです.

3.1 Maximaのバッチ処理

今度は,Maxima で複数のグラフを描く処理を行う為に,Maxima にバッチ処理を行わせる方法について簡単に解説しましょう.

先ず,gnuplot_preamble に記述する設定の内容は, 高度なグラフ表示を行わなければならなくなれば, 益々, 膨大なものとなるでしょう. これを一々手入力する事は最初の一枚のグラフを得る為に試行錯誤する場合は仕方が無いものとしても, 書式も全て決っており, 決められた書式で出力すれば済む話であるのならば, 自動化を図りたいところです.

ここで Maxima にはバッチ処理機能があります. この場合,Maxima に実行させるファイルを予め準備しておきますが, このファイルの中身は通常の Maxima の式を記述したものです. このファイルを Maxima に引渡すと,Maxima がファイルの内容に従って処理を行うもので, この処理は Maxima の通常の数式や数値の処理だけではなく, 画像の生成にも使えます.

手始めに簡単な入力ファイル A を記述しておきましょう.

ファイル A の内容

```
a:(x+1)^2;
b:expand(a);
plot2d(b,[x,-1,1]);
plot2d(b,[x,-1,1],
[gnuplot_preamble,"set term jpeg;set output 'test.jpeg'"]);
```

これは非常に簡単な処理です. 最初の二つの式は通常の Maxima の処理の式です. それから `plot2d` の処理を行なっていますが, この場合は,gnuplot でグラフを生成し, グラフ表示ウィンドウにグラフを表示してものを出力します. そして, 最後の処理では,gnuplot_preamble を利用して描画を行うものです. ここで gnuplot_preamble には,`set term jpeg` と `set output 'test.jpeg'` といった二つの gnuplot の設定文を纏めたものを入れています. ここで, 最初の命令文は gnuplot の terminal を JPEG に変更するもので, 次の命令文が出力先をファイル'test.jpeg' に切替えています. 即ち, 最後の二つの命令文により,gnuplot はグラフを JPEG データに変換してファイル test.jpeg に書込む事になります.

このファイルは Maxima の中から,`batch(A);` と入力する事でも処理が出来ますが, 一々,Maxima を立ち上げなくても, 以下の構文で Maxima にファイルを引渡す事が可能です.

Maxima によりバッチ処理

maxima -b ⟨ ファイル ⟩
maxima –batch=⟨ ファイル ⟩

猶, この処理は Windows でもコマンドプロンプトで実行する事が可能です. このバッチ処理の計算結果は, 出力先をバッチファイル内部で指定しない限り, 標準出力に出力されます. 重要な計算を行う場合は, リダイレクトを用いると良いでしょう.

又, 一々, コマンドラインに入力するのが面倒であれば, この内容を含むシェルスクリプトを記述しておくのも良いでしょう.

参考迄に `maxima -b A` を実行した様子を以下に示しておきます.

```
yokota@Zuse:~/TEST> maxima -b A
Maxima 5.10.0 http://maxima.sourceforge.net
Using Lisp CLISP 2.37 (2006-01-02)
Distributed under the GNU Public License. See the file COPYING.
Dedicated to the memory of William Schelter.
This is a development version of Maxima. The function bug_report()
provides bug reporting information.
(%i1)                              batch(A)

batching /home/yokota/TEST/A
                                          2
(%i2)                           a : (1 + x)
                                         2
(%o2)                             (x + 1)
(%i3)                           b : expand(a)
                                 2
(%o3)                           x  + 2 x + 1
(%i4)                      plot2d(b, [x, - 1, 1])
(%o4)
(%i5) plot2d(b, [x, - 1, 1], [gnuplot_preamble,
                                       set term jpeg;set output 'test.jpeg'])
(%o5)
```

この処理では,gnuplot のグラフも表示されています. 但し, このグラフは面白味が無いので, ここでは示しません. あしからず.

では, もう少し複雑なグラフを描いてみましょう. その為に, 次の内容のファイル A2 を記述しましょう.

ファイル A2 の内容

```
nekoneko:"set pm3d at bs;set xrange [-2:2];\
set yrange [-2:2];set zrange [-10:20];\
set label 1 'top' at 0,0,20;\
set arrow 1 from 0,0,20 to 0,0,10;\
set arrow 1 size 5,30 filled head;\
set hidden3d;\
set output 'test1.jpeg';set term jpeg;";
plot3d(10*cos(x*y),[x,-2,2],[y,-2,2],[grid,50,50],
[gnuplot_preamble,nekoneko]);
```

このファイル A2 では,gnuplot_preamble に与える文字列が長くなるので, 適宜,\を入れて改行しています.

`maxima -b A2` の結果の text1.jpeg の絵を図 3.3 に示しておきます.

ここで gnuplot で行う処理が複雑になればなる程,gnuplot_preamble の内容は必然的に長くなります. 画像 1 では設定 1, 画像 2 の場合に設定 2 を使う…等々とする場合, その設定を全て gnuplot_preamble に与える文字列として持たせるという方法は, 安易な方法ではありますが, 自動処理の事を考えた場合には, それ程, 効率的ではありません.

寧ろ, 共通の設定 1 があって, 後はケース毎に設定が追加される方が汎用性が高いでしょう. 又,

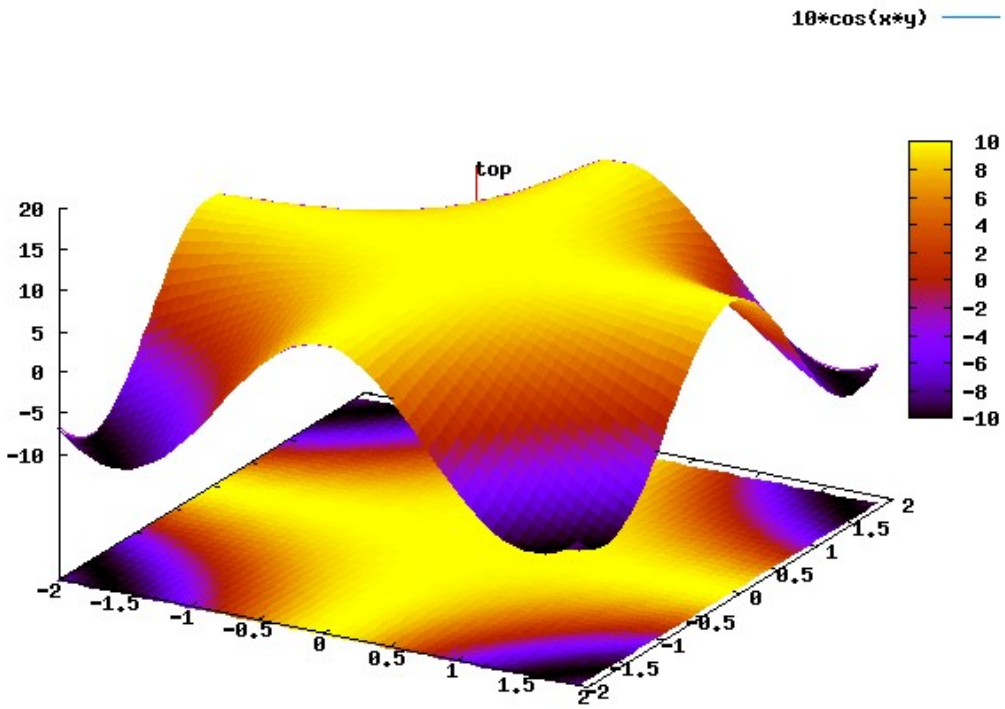


図 3.3: ファイル A2 の実行結果

ケース毎に画像ファイル等の保存先のファイルを切り替える事は普通に行われる事でもあります. ところで,gnuplot_preamble に与えるのは一つの文字列です. そこで,Maxima で複数の文字列を纏めて一つの文字列にする操作を行えば良い事になります.Maxima には, 文字列の結合を行う処理を行う函数として, concat 函数があります. この函数は与えられた複数の文字列を一つの文字列に変換する函数です.

そこで今度は concat 函数を使ってみた例を示しましょう.

ファイル B1 の内容

```
A1:"set pm3d at bs;";
R1:"set xrange [-10:10];\
set yrange [-10:10];set zrange [7:12];";
H1:"set hidden3d;set isosamples 50;";
O1:"set output ";
T1:"set term jpeg;";
Ox:concat(O1,"'Fig1.jpeg'",";");
nekoneko:concat(A1,R1,H1,Ox,T1);
assume(x^2+y^2>0);
fxy:10-realpart(log(sqrt(x^2+y^2+1)));
plot3d(fxy,[x,-10,10],[y,-10,10],[grid,20,20],
[gnuplot_curve_titles,"title 'fxy',5+x*y/10 title 'test'"],
[gnuplot_preamble,nekoneko]);
```

この例では,O1 に set output を割当てており, これを雛形として, 出力先のファイルを O1 に concat 函数で追加した文字列 Ox を利用しています. この方法を使えば, 条件文やバッチファイルを生成するスクリプトで, ファイル名の切替えが容易に行えます.

又, このファイルでは,splot に gnuplot の式を gnuplot_curve_titles を使って組込ませる様にしています.

更に, 今度は Maxima に B1 を処理させる以下に示すシェルスクリプトを記述します.

```
#!/usr/local/bin/maxima

maxima -b B1>B1.log
convert Fig1.jpeg Batch2.eps
display Fig1.jpeg
less B1.log
```

このシェルスクリプトでは, 最初に Maxima でファイル B1 の処理を実行します. その結果, 画像ファイル Fig1.jpeg が生成されますが, 次に convert 命令を使って, この Fig1.jpeg を Batch2.eps に変換して,Fig1.jpeg を display 命令を使って閲覧します. そして最後にバッチ処理の記録ファイルの B1.log を見るというものです.

このバッチ処理で生成された画像ファイルを図 3.4 に示しておきます.

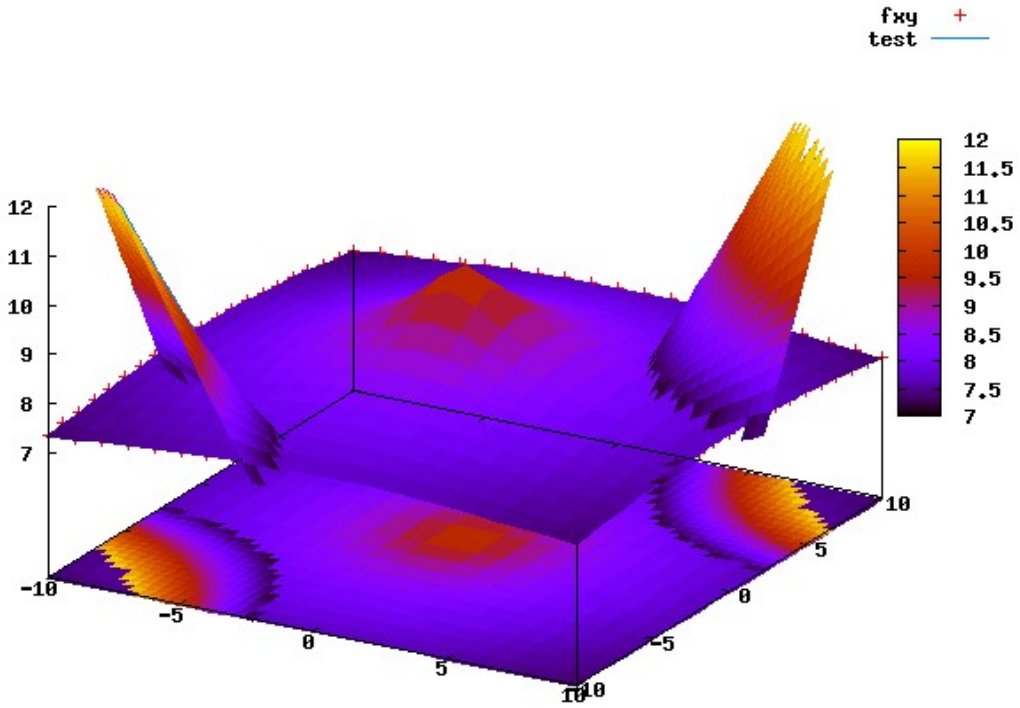


図 3.4: ファイル B1 の実行結果

Maxima のバッチ処理と gnuplot_preamble が揃えば, 後は,awk 等の適切な言語があれば十分複雑な処理が行えるでしょう.

関連図書

[1] J. リヒター-ゲバート/U.H. コルテンカンプ著, 阿原一志訳, シンデレラ 幾何学のためのグラフィックス, シュプリンガー・フェアラーク東京,2001.

[2] 河内明夫編, 結び目理論, シュプリンガーフェラーク東京,1990.

[3] 下地貞夫, 数式処理, 基礎情報工学シリーズ, 森北出版,1991.

[4] クロウエル, フォックス, 結び目理論入門, 現代数学全書, 岩波書店,1989.

[5] ポール・グレアム,ANSI Common Lisp, ピアソン・エデュケーション,2002.

[6] 本間龍雄, 組合せ位相幾何学, 共立出版,1980.

[7] 寺坂英孝編, 現代数学小辞典, ブルーバックス, 講談社,2005.

[8] 丸山茂樹, クレブナー基底とその応用, 共立叢書 現代数学の潮流, 共立出版,2002.

[9] 村上順, 結び目と量子群, 数学の風景 3, 朝倉書店,2000.

[10] 日本数学会編, 数学辞典 第 3 版, 岩波書店,1987.

[11] 矢吹道郎, 大竹敢, 使いこなす GNUPLOT(改訂新版), テクノプレス,2001.

[12] H.Cohen, A Course in Computational Algebraic Number Theory,GTM 138, Springer-Verlag,New York-Berlin,2000.

[13] D.Cox,J.Little and D. O'Shea,Ideals, Varieties, and Algorithms, UTM,Springer-Verlag,New York-Berlin,1992.

[14] Gert-Martin Greuel, Gerhard Pfister, A Singular Introduction to Commutative Algebra, Springer-Verlag,New York-Heiderberg-Berlin,2000.

[15] D.Rolfsen, Knots and Links. Publish or Perish, Inc,1975.

[16] Hal Schenck, Computational Algebraic Geometry London Mathematical Society student texts;58,2003.

[17] Open AXIOM のサイト http://wiki.axiom-developer.org/FrontPage

[18] DERIVE のサイト http://www.derive.com

[19] GAP のサイト http://www-gap.dcs.st-and.ac.uk/

[20] Macaulay2 のサイト http://www.math.uiuc.edu/Macaulay2/

[21] Mathsoft Engineering & Education, Inc. http://www.mathcad.com/

[22] Wolfram Research Inc. http://www.wolfram.com/

[23] Waterloo Maple Inc. http://www.maplesoft.com/

[24] Maxima の SOURCEFORGE のサイト http://maxima.sourceforge.net/

[25] MuPAD のサイト http://www.mupad.de/

[26] PARI/GP のサイト http://pari.math.u-bordeaux.fr/

[27] REDUCE のサイト http://www.zib.de/Symbolik/reduce/

[28] Risa/Asir 神戸版 http://www.math.kobe-u.ac.jp/Asir/asir-ja.html

[29] OpenXM(Open message eXchange for Mathematics)
http://www.math.sci.kobe-u.ac.jp/OpenXM/index-ja.html

[30] SINGULAR のサイト http://www.singular.uni-kl.de/

[31] 株式会社シンプレックスのページ htto://www.simplex-soft.com/

[32] The MathWorks,Inc. http://www.mathworks.com/

[33] Octave WebPage http://bevo.che.wisc.edu/octave/

[34] Scilab のサイト http://www.scilab.org/

[35] Yorick の公式サイト ftp://ftp-icf.llnl.gov/pub/Yorick/
Yorick の非公式サイト http://www.maumae.net/yorick/doc/index.php

[36] R のサイト http://www.r-project.org

[37] Insightful Corporation のページ http://www.insightful.com/

[38] Cinderella のサイト
本家：http://www.cinderella.de/
日本語版ホームページ http://cdyjapan.hp.infoseek.co.jp/

[39] dynagraph のサイト http://www.math.umbc.edu/ rouben/dynagraph

[40] KSEG のサイト http://www.mit.edu/ ibaran/kseg.html

[41] Geomview のサイト http://http.geomview.org/

[42] surf の sourceforge のサイト http://surf.sourceforge.net/

[43] XaoS のサイト http://wmi.math.u-szeged.hu/ kovzol/xaos