Maxima for the Sciences

Maxima: 理工系ツールとしての

1991 年に翻訳された「Mathematica: 理工系ツールとしての」R. E. クランドール著／伊藤利明／蔡　東生訳は、科学とコンピュータの好きな人の知的好奇心をくすぐる良書でした。残念ながら今は絶版になっています。Mathematica は、 2009 年現在においても 40 万円以上する高額ソフトウエアであり、一般ユーザーはなかなか利用することはできません。一方で、1991 年と現在とでは、フリーソフトウエアの状況が一変しており、Mathematica に肉薄するソフトウエアがいくつか登場しています。中でも Maxima は数式処理とグラフィックスに関して、通常のユーザーが必要とする機能はほとんど備えていると言っていいかと思います。

本稿の目的は、「Mathematica: 理工系ツールとしての」の題材を Maxima で実行しながら学習することです。

本稿の執筆に当たって、 2 つの方針を採用しました。

1. 題材に関する基本的説明を行うこと
2. 利用した Maxima の関数について、マニュアルの意訳をつけること

必ずしも、Maxima の入門を目指してはいません。Maxima 入門としては、

- 「Maxima 入門ノート」中川義行著 http://www.eonet.ne.jp/~kyo-ju/maxima.pdf

をお勧めします。

インストール

Maxima のウェブサイトは http://maxima.sourceforge.net/です。そこから辿れるダウンロードページから、自分のパソコンに合ったバイナリをダウンロードして、インストールします。

本稿は、Maxima 5.18.1 に基づいて作成しました。

基本

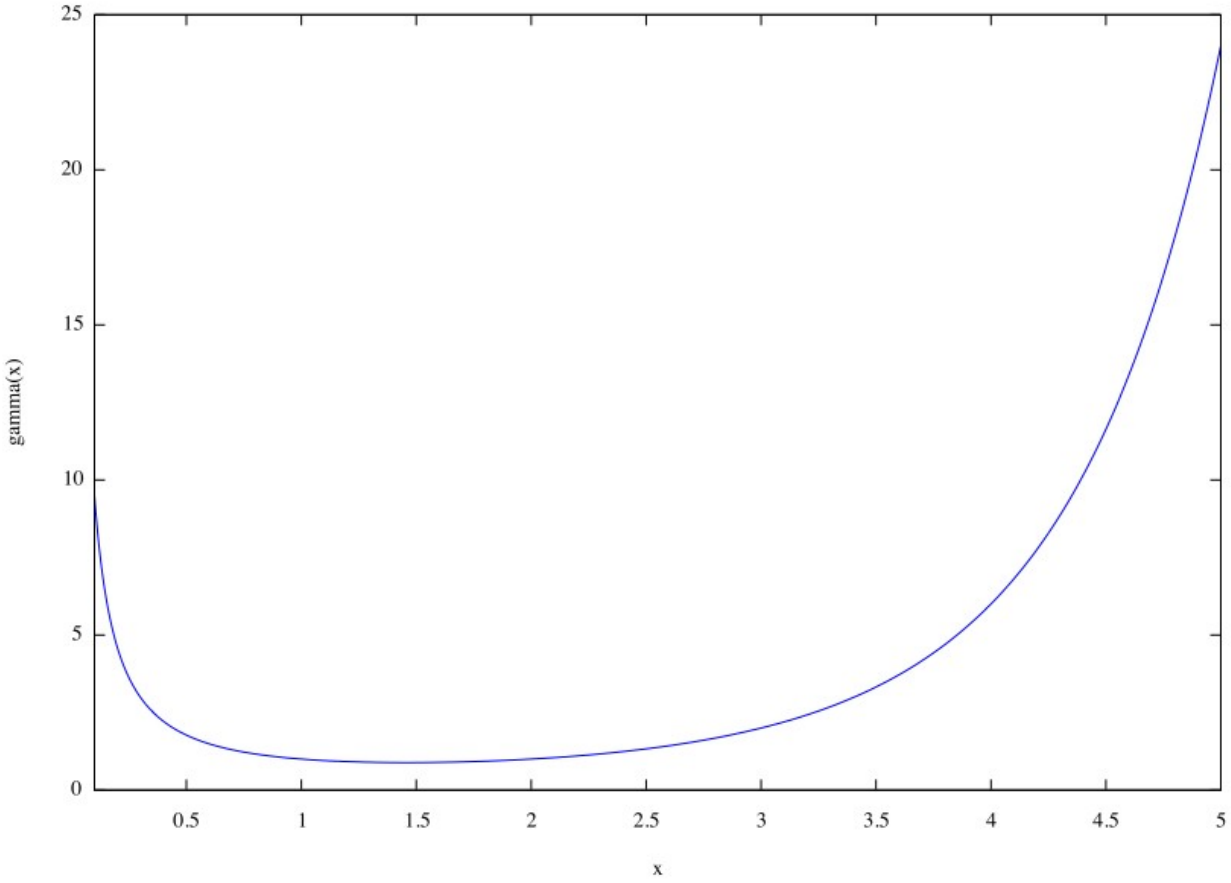
Maxima のほとんどは式です。(*expr_1*, *expr_2*, , *expr_n*)は、expr_1 から順に評価して、expr_n の評価結果を返す式です。

;または$を見つけると、その直前までの式を評価し、;の場合、結果を出力します。$の場合、出力が抑制されます。

グラフィックス

２次元、３次元グラフィックス

```
plot2d(gamma(x), [x, 0.1, 5]);
```



ガンマ関数は、階乗 $n!\equiv 1\cdot 2\cdot 3\cdot \ldots \cdot n$ を連続関数にするために Leonhard Euler(1707/4/15-17839/8)が考えたそうです。ガンマ関数の基本的な定義は、以下の通りです。

$$\Gamma(z)\equiv\int_0^\infty t^{z-1}e^{-t}dt\,(z>0) \quad (1)$$

部分積分を使うと、以下の公式が証明できます。

$$\Gamma(z)=(z-1)\Gamma(z-1) \quad (2)$$

$n!=n\cdot(n-1)!$ なので、階乗の性質とよく似ています。実際、ガンマ関数に自然数を入れると、以下のような形で階乗と等しくなります。

$$\Gamma(n)=(n-1)!$$

$\Gamma(0)=\infty$ なのですが、 $(-1)!=\infty$ ってちょっと不思議ですね。

ガンマ関数の定義(1)の積分は、z>0 だけでなく、z に実部が正の複素数でも収束します。また、定義(1)の積分は、z の実部が 0 以下の複素数では収束しませんが、解析接続によって、z の実部が 0 以下の複素数でも関数を定義することができます。他にもガンマ関数についてたくさんの研究がありますが、この辺りで Maxima に戻りましょう。

使った機能

- **plot2d***(expr, x_range, ..., options, ...)*

plot2d は、式 expr を 1 変数の関数としてプロット表示する関数です。x_range は[variable, min, max]という形のリストで指定します。

plot2d は、他にも色々なプロット表示をすることができます。詳しくは、ヘルプを見てください。

- **gamma***(z)*

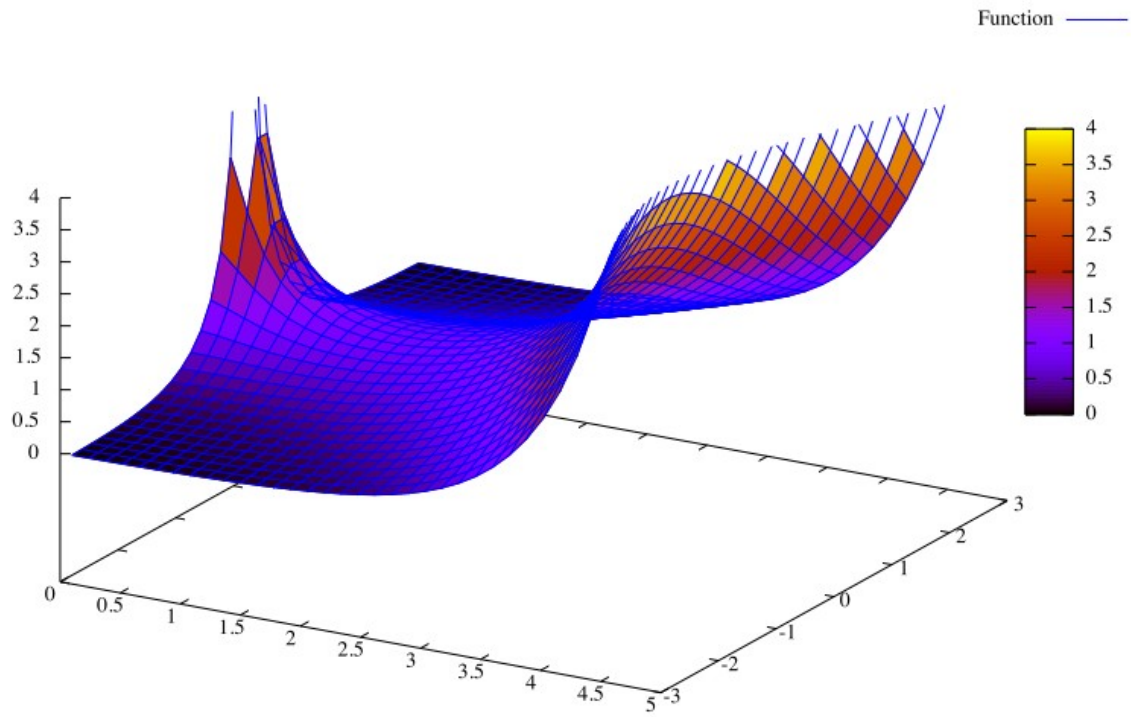
gamma は、z が自然数のとき階乗(z-1)の計算結果を返します。有理数のときは、公式(2)に基づいた、gamma(a) (0<a<1)を使った式を返します。特に a=1/2 のとき $\Gamma(\frac{1}{2})=\sqrt{\pi}$ を使った式を返します。z が浮動小数点表現の複素数の場合、計算結果を浮動小数点表現の複素数で返します。

gamma の振る舞いに Maxima の特徴が出ています。Maxima は数式処理が基本なので、基本的に厳密な式を返そうとします。引数が浮動小数点の場合、引数自体が近似的だと見なされ、近似数値を返します。

演習問題

1. 好きな関数（例えば、 $\sin\theta$ ）や式（例えば、 $(x+1)(x-1)^2$ ）をプロットしてください。
2. gamma(1/2)と gamma(0.5)の結果を比較してください。
3. plot2d のヘルプを読んで、gamma(x)と整数点での(n-1)!をプロットしてください。
4. $\Gamma(z)=(z-1)\Gamma(z-1)$ を証明してください。

```
plot3d(abs(gamma(x + %i*y)), [x, 0.1, 5], [y, -3, 3], [z, 0, 4], [grid, 35, 35], [gnuplot_pm3d, true]);
```



ガンマ関数は、実軸に対して鏡像な関数であり、すなわち、 $\Gamma(\sigma+ti)=\overline{\Gamma(\sigma-ti)}$ が成り立ちます。プロットされたグラフが y 軸に対して対象であることを確認してください。

使った機能

- **plot3d***(expr, x_range, y_range, ..., options, ...)*

plot3d は、式 expr を 2 変数の関数としてプロット表示する関数です。

plot3d は、他にも色々なプロット表示をすることができます。詳しくはヘルプを見てください。

- **abs***(expr)*

expr の絶対値を返します。

- +, -, *, /, ^

それぞれ、加算、減算、乗算、除算、冪乗の算術二項演算子です。

Maxima では、二項演算子は数学と同じ優先順位を持ちます。普通に記述すれば、意図

する式になります。

+, -は符号を示す単項演算子にもなります。

Maximaの内部表現では、+, *はn項演算子であり、-, /は加法、乗法の逆元を作る単項演算子と+または*との組み合わせになります。

数字リテラルに関しては、冪乗を除いて算術演算が実行されます。冪乗演算は、オペランドが浮動小数点の場合、もしくは演算結果が有理数になる場合には実行されます。

算術演算は評価ではなく、簡素化です。クオートされた式の中でも実行されます。

- **%i**

虚数単位です。

- *[***grid***, expr_1, expr_2]*

gridは3次元プロットのオプションの1つで、xy平面上の数値を確定する格子を何点取るかを指定します。 *expr_1* にはx軸方向の格子数、 *expr_2* はy軸方向の格子数を指定します。

- *[***gnuplot_pm3d***, true]*

gnuplot_pm3dをtrueに設定することで、3次元プロットがパレットマップ表示されます。

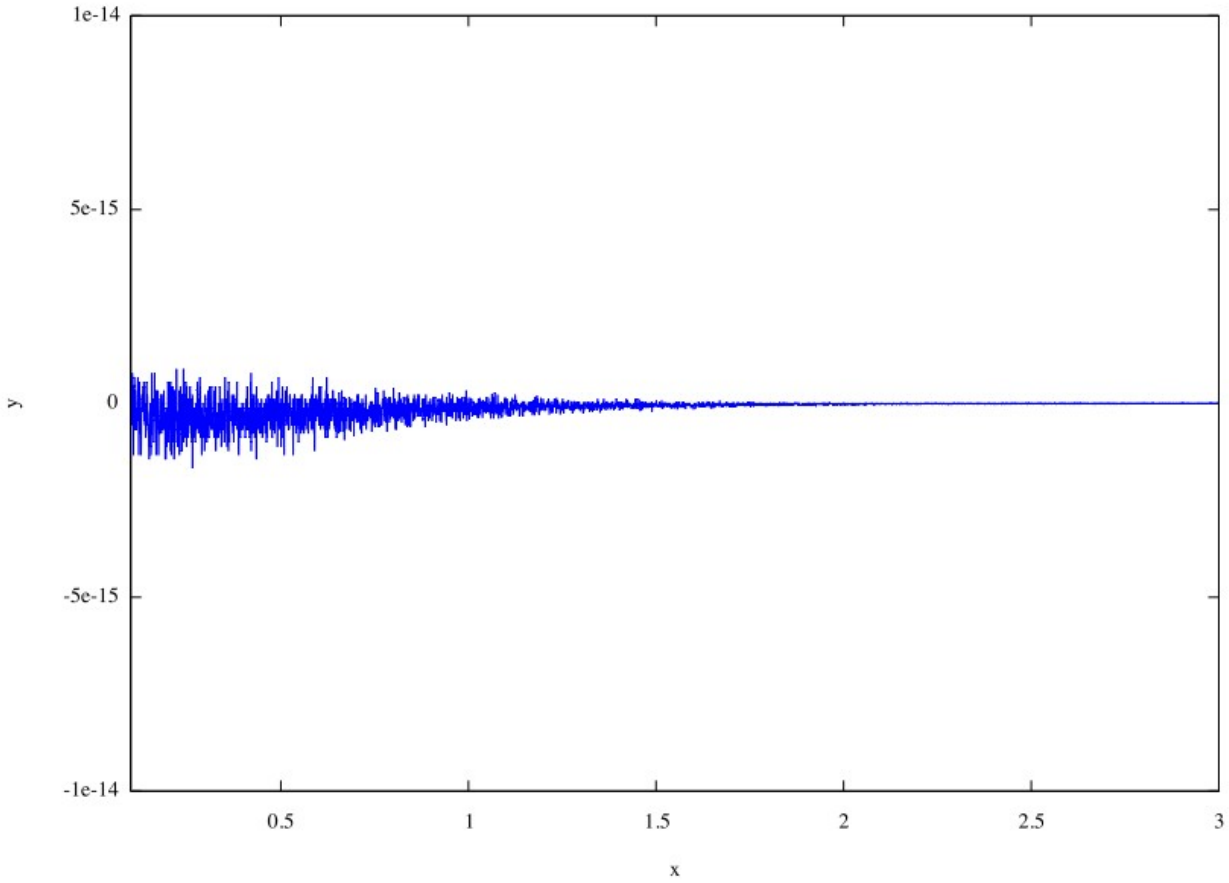
他にどんなプロットオプションが設定できるか見るためには、

plot_options;
を実行してください。

演習問題

1. 好きな2変数関数や式をプロットしてください。
2. $\Gamma(\sigma+it)=\overline{\Gamma(\sigma-it)}$ を証明してください。

```
epsilon : 10^(-14);
plot2d(abs(gamma(1 + %i * x)) ^ 2 - (%pi * x / sinh(%pi * x)), [x, 0.1, 3], [y, -epsilon,
epsilon], [nticks, 1000]);
```



本来、 $|\Gamma(1+iy)|^2=\frac{\pi y}{\sinh \pi y}$ が成り立つので、プロットはすべて0になるべきですが、数値計算の誤差がグラフに現れています。

使った機能

- **:**

割当(assignment)演算子です。

- **%pi**

円周率です。

- **sinh***(x)*

双曲線正弦関数です。 $\sinh(x)=\frac{e^x-e^{-x}}{2}$

- *[***y***, expr_1, expr_2]*

2 次元プロットでの縦軸のレンジの設定。 *expr_1* は最小値、 *expr_2* は最大値を指定し

ます。

- **/nticks**, expr]

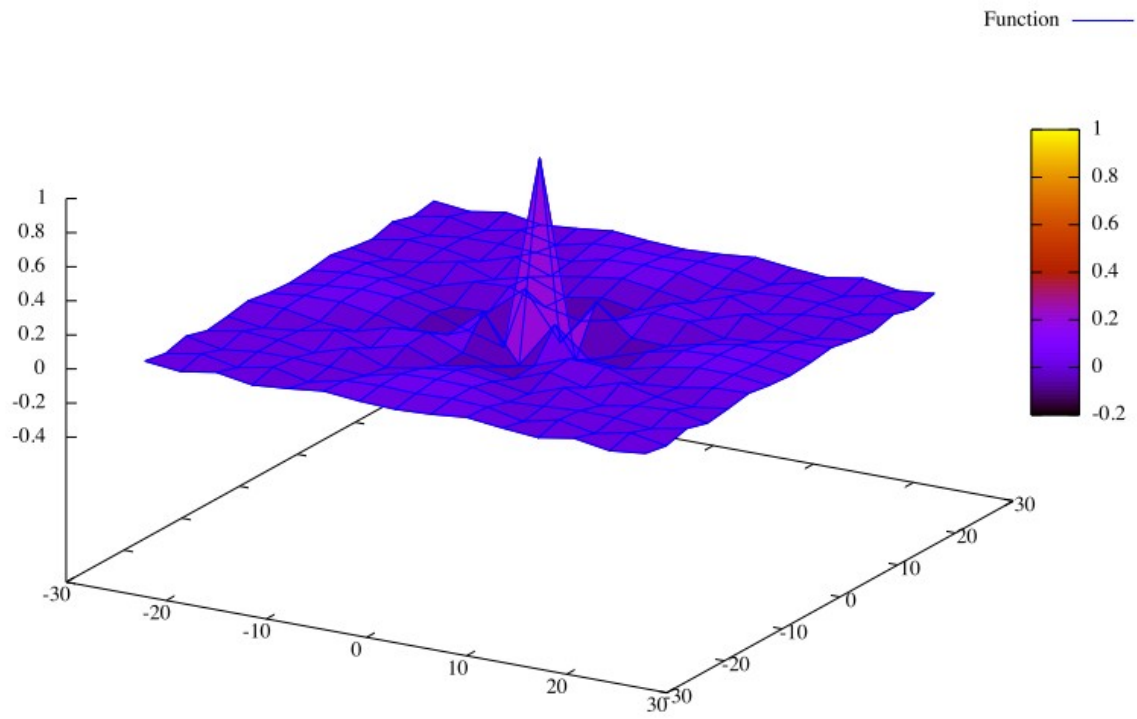
x 軸方向の格子点の数を指定します。

演習問題

1. $|\Gamma(1+iy)|^2=\frac{\pi y}{\sinh \pi y}$ を証明してください。

解像度の重要性

```
sinc(x) := if float(x) = 0.0 then 1.0 else sin(x) / x;
plot3d(sinc(1.5 * sqrt(x ^ 2 + y ^ 2)), [x, -25, 25], [y, -25, 25], [z, -0.5, 1], [grid, 14, 14],
[gnuplot_pm3d, true]);
```



$$\frac{\sin(1.5r)}{1.5r}$$

使った機能

- *f(x_1, ..., x_n)* **:=** *expr*

関数 *f* を定義します。*expr* は評価されません。

- **if** *cond_1* **then** *expr_1* **<elseif** *cond_2* **then** *expr_2* **… else** *expr_0***>**

条件評価の構文です。

- **sin***(x)*

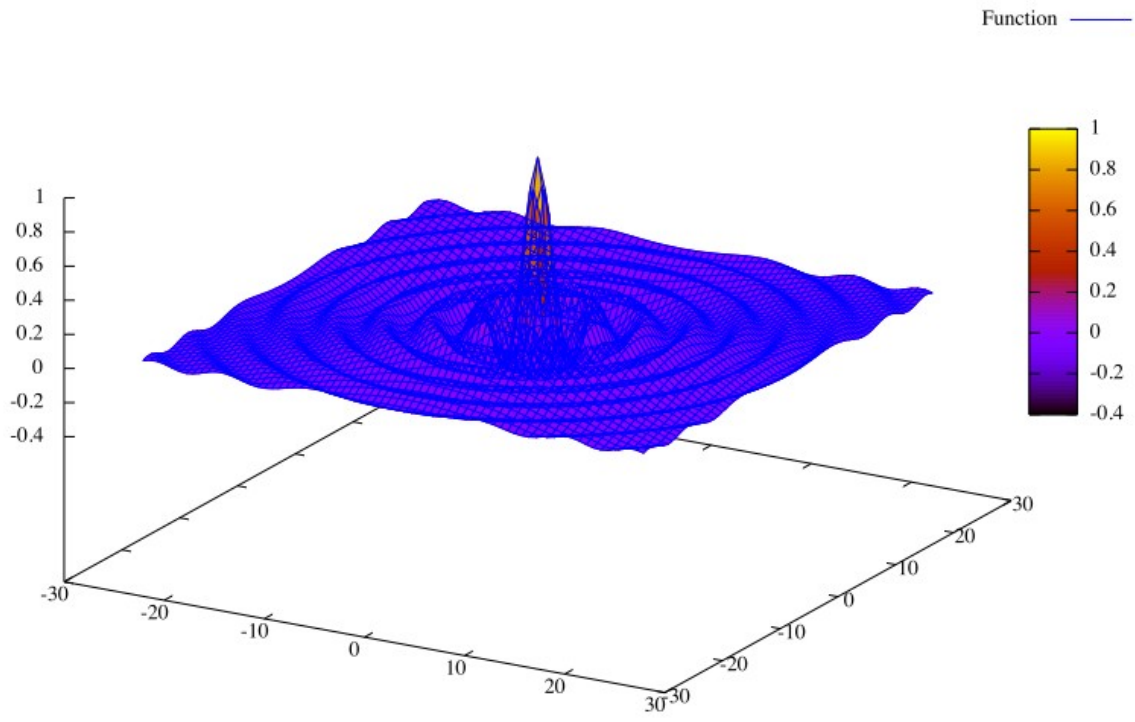
正弦関数です。

- *[***z***, expr_1, expr_2]*

3 次元プロットでの縦軸のレンジの設定。 *expr_1* は最小値、 *expr_2* は最大値を指定し

ます。

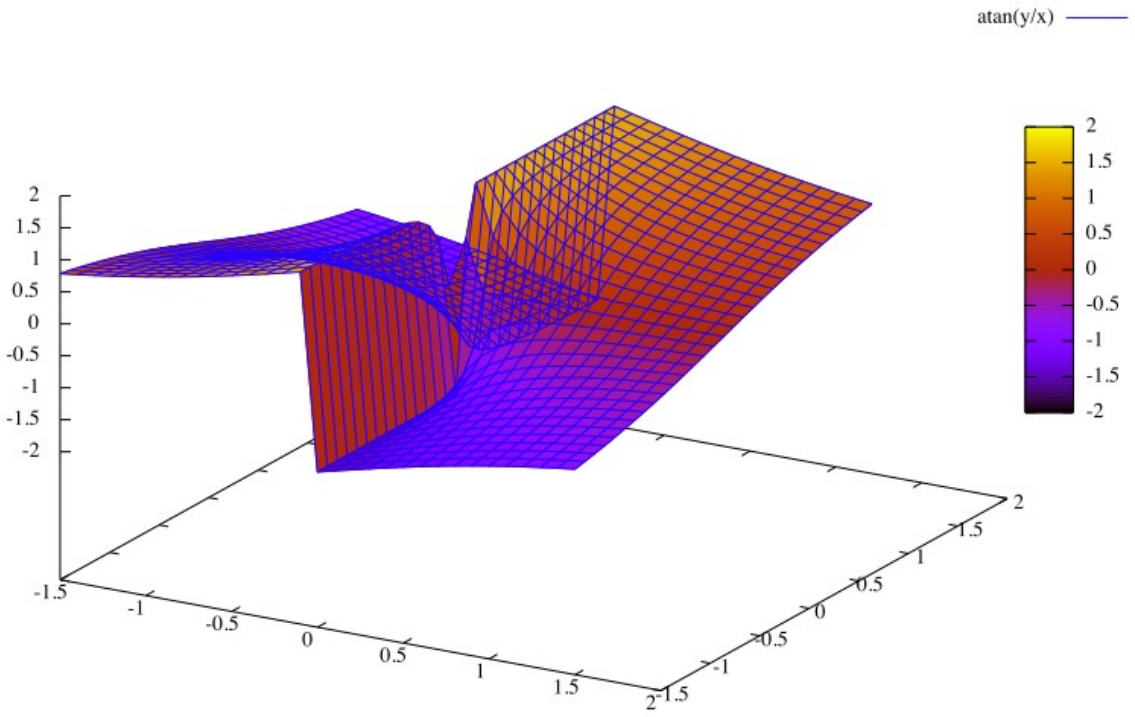
```
plot3d(sinc(1.5 * sqrt(x ^ 2 + y ^ 2)), [x, -25, 25], [y, -25, 25], [z, -0.5, 1], [grid, 70, 70],
[gnuplot_pm3d, true]);
```



grid が変わると滑らかさもずいぶん変わりますね。

等高線表示

```
plot3d(atan(y/x), [x, -1.5, 1.5], [y, -1.5, 1.5], [gnuplot_pm3d, true]);
```



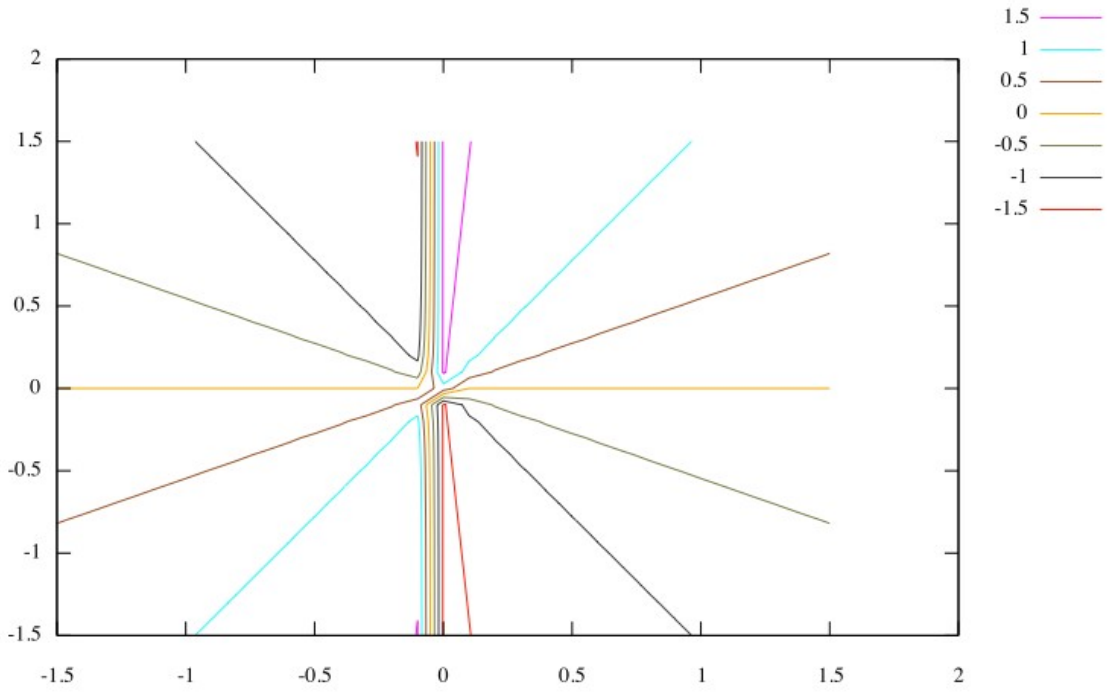
このグラフの等高線を作成します。

使った機能

- **atan***(x)*

逆正接関数です。

```
contour_plot(atan(y / x), [x, -1.5, 1.5], [y, -1.5, 1.5], [gnuplot_preamble, "set cntrparam levels 6"]);
```



使った機能

- **contour_plot***(expr, x_range, y_range, options, ...)*

等高線をプロットします。オプションは plot3d と同じです。

- *[***gnuplot_preamble***, "***set cntrparam levels** *6"]*

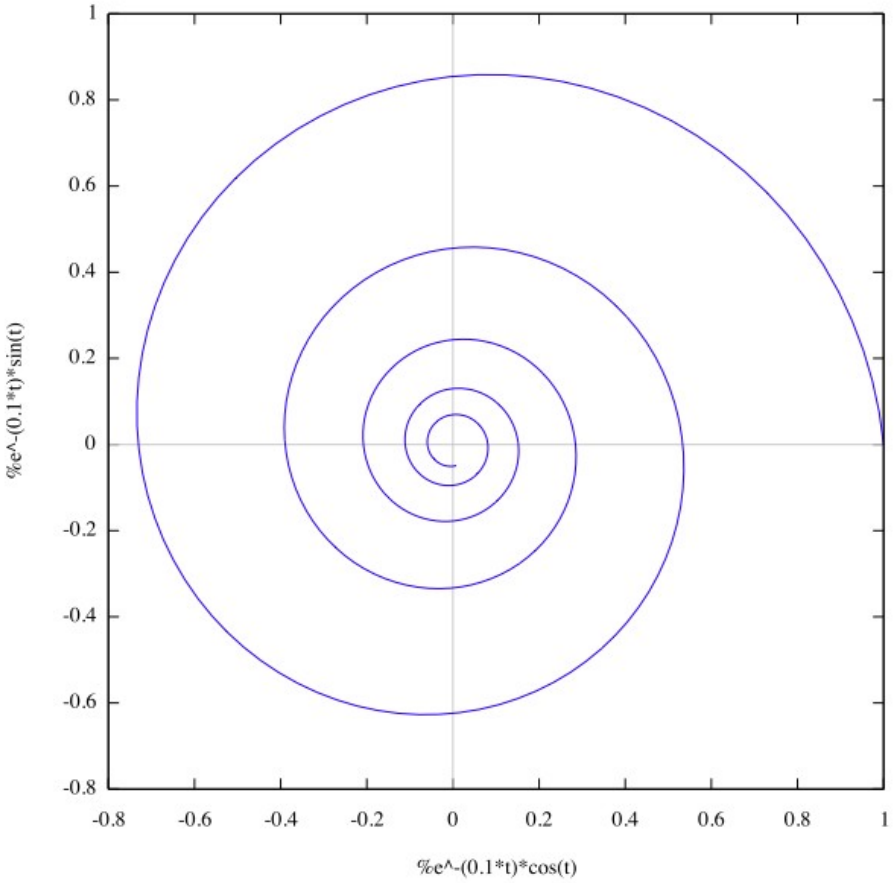
オプション gnuplot_preamble を使うと、gnuplot にデータを渡す前に、gnuplot のコマンドを渡すことができます。cntrparam は gnuplot の等高線用パラメータです。Levels は等高線の段階数による設定を示します。

- *[***gnuplot_pm3d***, false]*

gnuplot のカラーマップ表示をオフにします。Maxima では、デフォルトでカラーマップ表示はオフですが、Mathematica との互換性を近づけるべく、初期設定でオンにしています。なので、ここではオフにしました。

パラメトリックプロット

```
r(t) := exp(-0.1 * t);
plot2d([parametric, r(t) * cos(t), r(t) * sin(t), [t, 0, 30], [nticks, 400]], [gnuplot_preamble, “set size square”]);
```



対数螺旋曲線です。等角螺旋とも呼ばれます。
y座標がx座標の 1 価関数でないとき、2 次元パラメトリックプロットが必要となります。

使った機能

- **exp***(x)*

指数関数です。%e^x に変換されます。

- *[***parametric***, x_expr, y_expr, t_range, options]*

パラメトリックプロットの指定です。

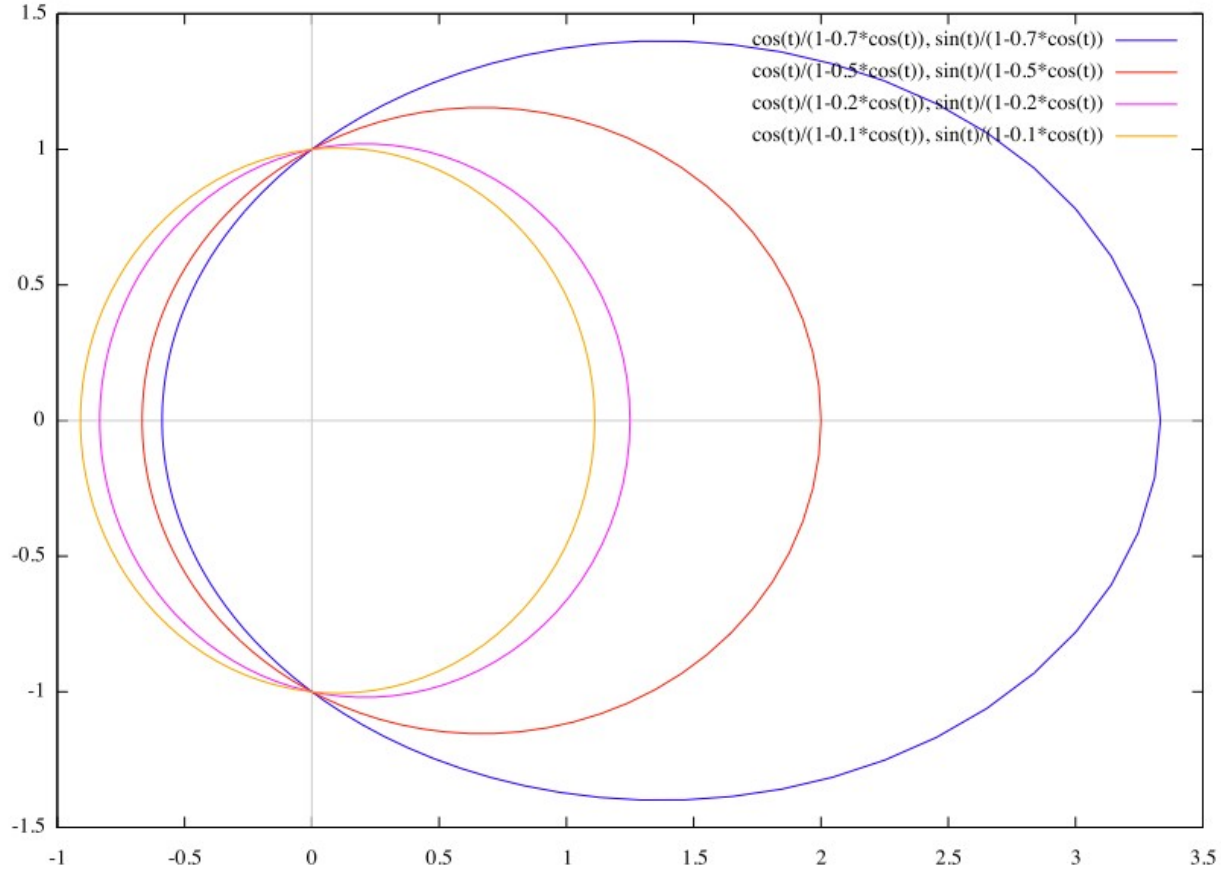
- *[gnuplot_preamble, “***set size square 1 1***”]*

gnuplot に縦横比を 1 にするようコマンドを送ります。

演習問題

1. 対数螺旋上の点をPとしたとき、線分OPとPでの螺旋曲線の接線の成す角が一定であることを証明してください。

```
r(t, e) := 1 / (1 - e * cos(t));
list : makelist([parametric, r(t, e) * cos(t), r(t, e) * sin(t), [t, 0, 2 * %pi]], e, [0.7, 0.5, 0.2, 0.1]);
plot2d(list, [nticks, 100]);
```



$\frac{1}{r^2}$ の引力に従った物体の軌道群（楕円）です。

使った機能

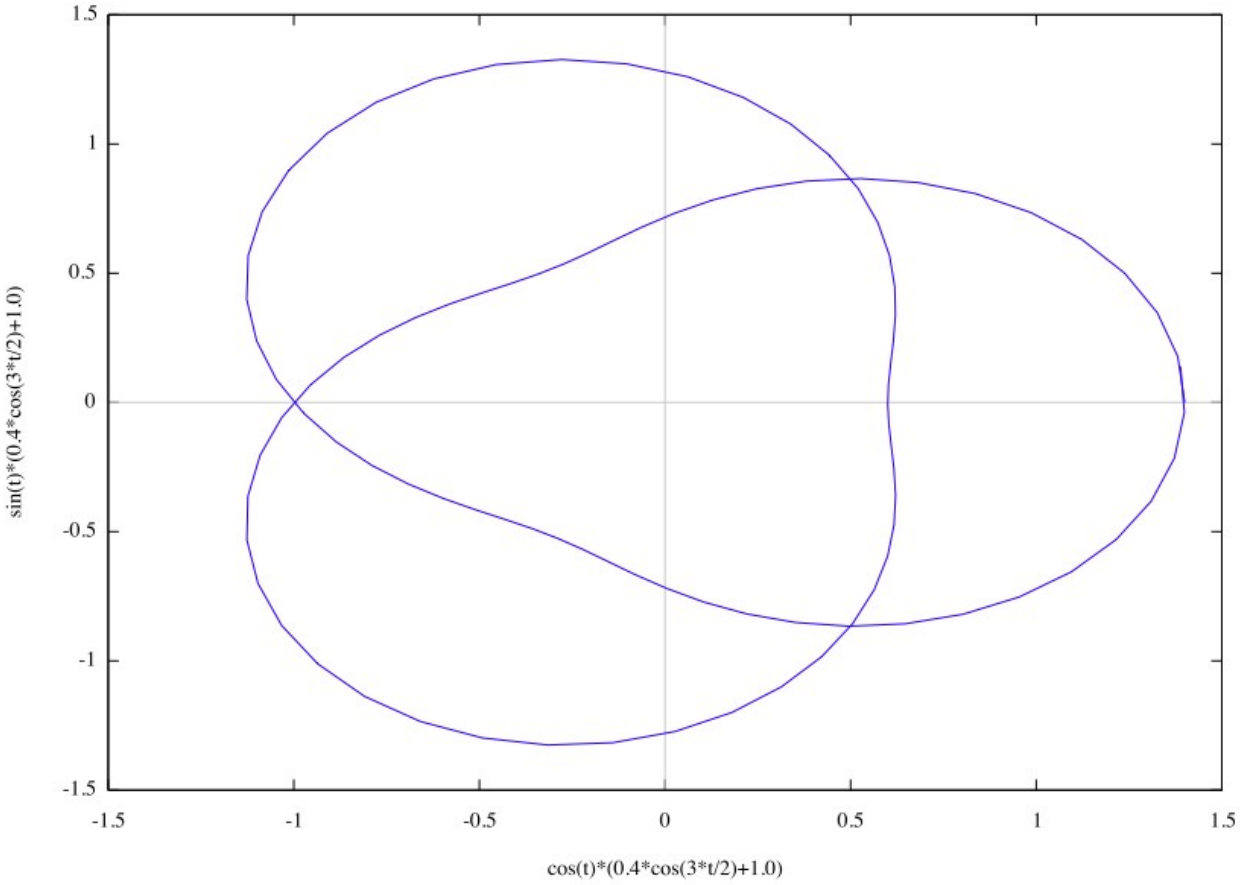
- **makelist***(expr, x, list)*

expr の中の変数 x に list の要素を代入した結果のリストを返します。

makelist は他にも引数の指定もできますので，詳しくは Maxima のヘルプを見てください。また、類似の関数に create_list があります。create_list は makelist の多変数バージョンです。

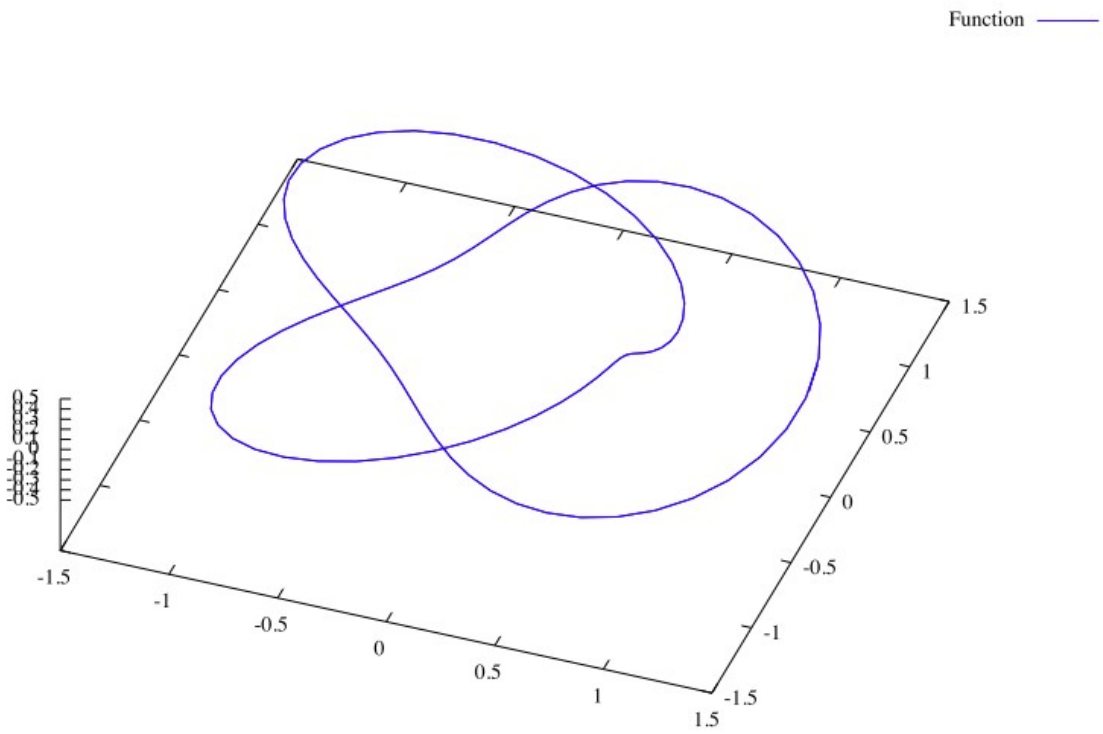
3次元パラメトリックプロット

```
a : 1.0;
b : 0.4;
r(t) := a + b * cos(3 * t / 2);
plot2d([parametric, r(t) * cos(t), r(t) * sin(t), [t, 0, 4 * %pi + 0.1]], [nticks, 100]);
```



これは三つ葉編み紐の投影図です。

```
a : 1.0;
b : 0.4;
c : 0.5;
r(t) := a + b * cos(3 * t / 2);
z(t) := c * sin(3 * t / 2);
plot3d([r(t) * cos(t), r(t) * sin(t), z(t)], [t, 0, 4 * %pi + 0.1], [dummy, 0, 1], [grid, 100, 1],
[gnuplot_preamble, "set view 20,20"]);
```



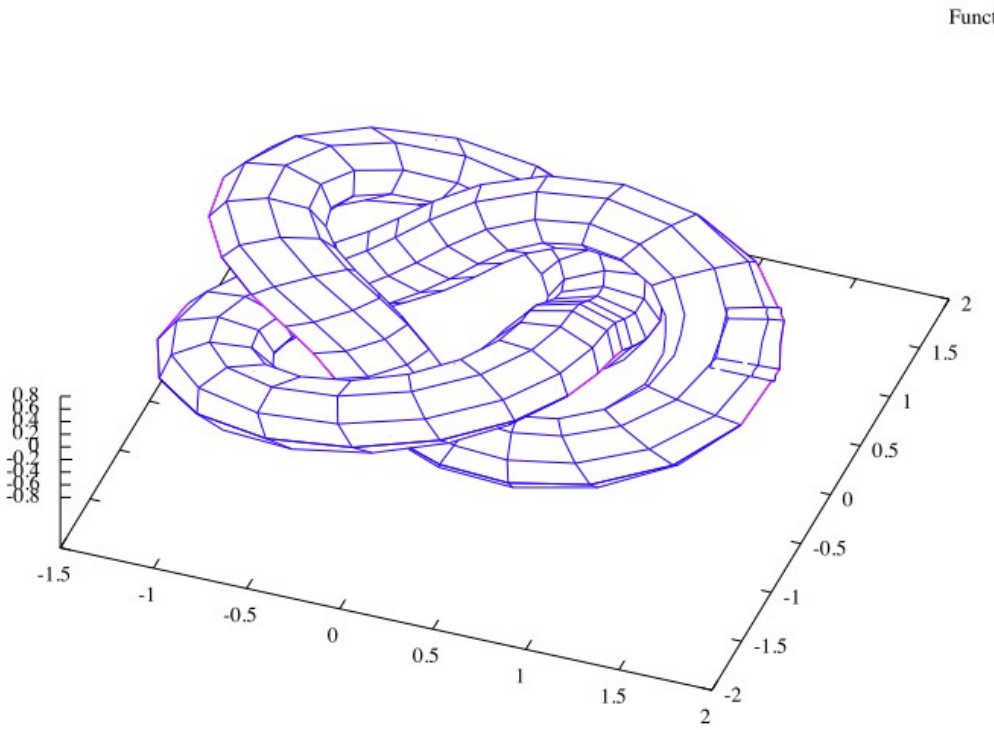
斜めから見てみました。

使った機能

- *[gnuplot_preamble, "set* **view** *rot_x, rot_z"]*

viewは、 3 次元図形をx軸に対してrot_x度回転させて、z軸に対してrot_z度回転させて、表示します。

```
a : 1.0;
b : 0.4;
c : 0.5;
d : 0.3;
r(t) := a + b * cos(1.5 * t);
z(t) := c * sin(1.5 * t);
x(t) := r(t) * cos(t);
y(t) := r(t) * sin(t);
qqx(t) := -((1.0 + 0.3 * cos(1.5 * t)) * sin(t)) - 0.45 * cos(t) * sin(1.5 * t);
qqy(t) := cos(t) * (1.0 + 0.3 * cos(1.5 * t)) - 0.45 * sin(t) * sin(1.5 * t);
qqz(t) := 0.75 * cos(1.5 * t);
norm(t) := sqrt(qqx(t) ^ 2 + qqy(t) ^ 2 + qqz(t) ^ 2);
qx(t) := qqx(t) / norm(t);
qy(t) := qqy(t) / norm(t);
qz(t) := qqz(t) / norm(t);
normv(t) := sqrt(qx(t) ^ 2 + qy(t) ^ 2);
vx(t) := qy(t) / normv(t);
vy(t) := -qx(t) / normv(t);
wx(t) := -qz(t) * vy(t);
wy(t) := qz(t) * vx(t);
wz(t) := qx(t) * vy(t) - vx(t) * qy(t);
xx(t, u) := x(t) + d * (vx(t) * cos(u) + wx(t) * sin(u));
yy(t, u) := y(t) + d * (vy(t) * cos(u) + wy(t) * sin(u));
zz(t, u) := z(t) + d * (wz(t) * sin(u));
plot3d([xx(t, u), yy(t, u), zz(t, u)], [t, 0, 4 * %pi + 0.31], [u, 0, 2 * %pi], [grid, 50, 8],
[gnuplot_preamble, "set view 20,20; set hidden3d"]);
```



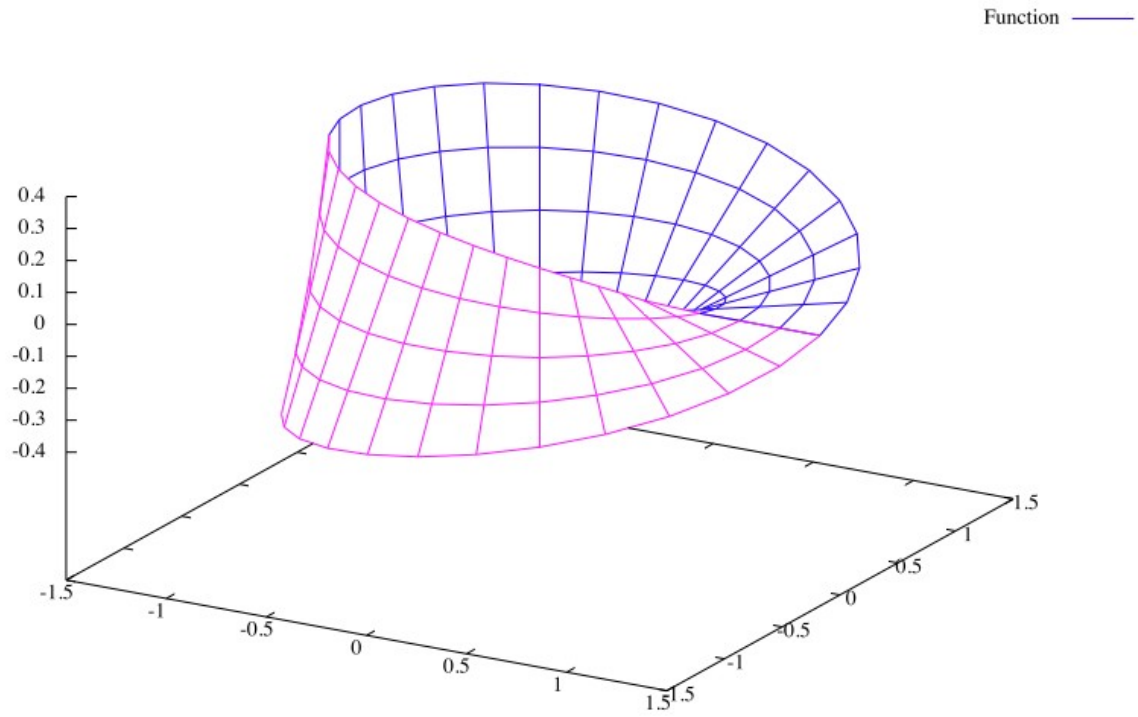
3 次元構造がずいぶんわかりやすくなりました。

使った機能

- *[gnuplot_preamble, "set view 20,20; set* **hidden3d***"]*

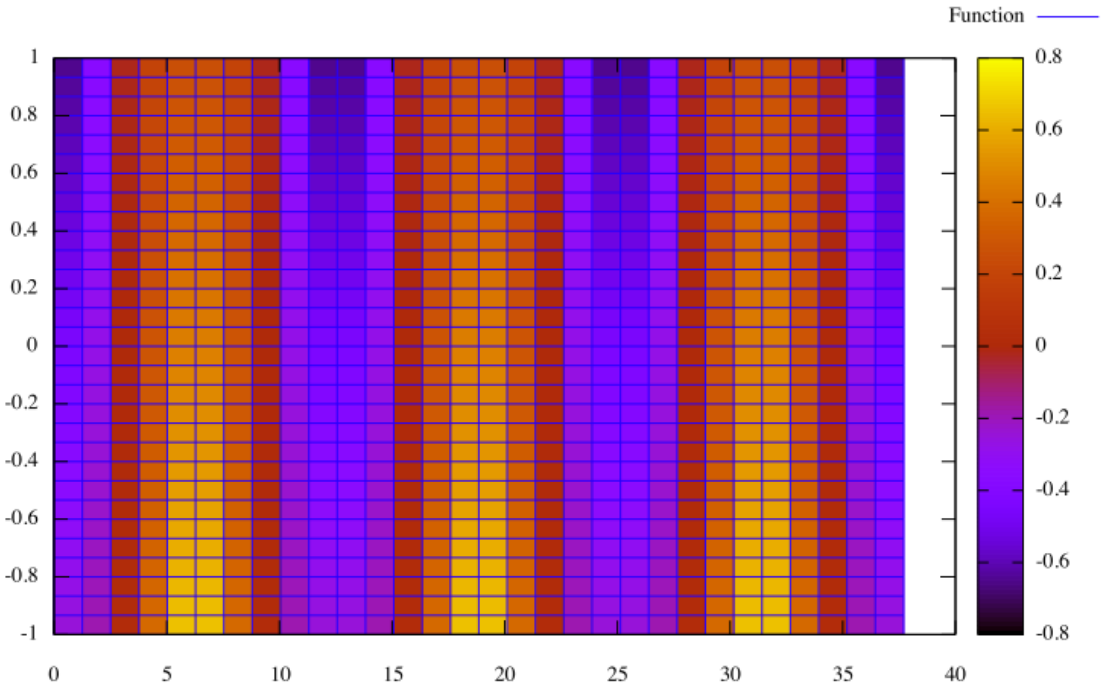
hidden3dを設定すると、陰線処理して表示されます。

```
a : 1.0; b = 0.5;
r(t, v) := a + b * v * cos(t / 2);
x(t, v) := r(t, v) * cos(t);
y(t, v) := r(t, v) * sin(t);
z(t, v) := b * v * sin(t / 2);
plot3d([x(t, v), y(t, v), z(t, v)], [t, 0, 2 * %pi], [v, -1, 1], [grid, 30, 4], [gnuplot_preamble, "set
hidden3d"]);
```



メビウスの帯です。

```
a : 1.0;
b : 0.5;
r(t, v) := a + b * v * cos(t / 2);
x(t, v) := r(t, v) * cos(t);
y(t, v) := r(t, v) * sin(t);
z(t, v) := b * v * sin(t / 2);
cross(c, d) := transpose(adjoint(matrix(c, d, [1, 1, 1])))[3];
tant(t, v) := diff([x(t, v), y(t, v), z(t, v)], t);
tanv(t, v) := diff([x(t, v), y(t, v), z(t, v)], v);
nz(t, v) := cross(tant(t, v), tanv(t, v))[3];
nz(t, v);
plot3d(nz(t, v), [t, 0, 12 * %pi], [v, -1, 1], [gnuplot_pm3d, true], [gnuplot_preamble, "set view
map"]);
```



上の密度プロットは、メビウスの帯の法線ベクトルを帯に従って 6 周回しても、法線ベクトル（のz成分）は 3 周しかしないことを示しています。

使った機能

- **transpose***(M)*

行列 M の転置行列を返します。

- **adjoint***(M)*

行列 M の余因子行列を返します。余因子行列は、逆行列を求める際に行列式の後に出てくるあれです。

- **matrix***(row_1, ..., row_n)*

同じ長さを持つリスト row_1, ..., row_n をそれぞれの行とする行列を返します。

- **diff***(expr, x)*

式 expr を変数 x で微分した結果を返します。

- *[gnuplot_preamble, "***set view map***"]*

set view map は 3 次元の表面プロットを上から投影して表示します。

Maxima にはベクトルの外積を求める関数がないので、transpose と adjoint, matrix を使って外積を返す関数 cross を定義しました。

2 つのパラメータのそれぞれの接線方向のベクトルを微分を使って求め、その外積を計算すると、それは法線ベクトルになります。