1. スクリプト plot.sh を各行説明しなさい。

1.1 スクリプト : plot.sh

```
1  #!/bin/sh
2  num=e0957XX
3  echo "set terminal png" > $num.gplot
4  echo "set output \"$num.png\"" >> $num.gplot
5  echo "set xrange [-2:2]" >> $num.gplot
6  echo "set yrange [-2:2]" >> $num.gplot
7  echo "set isosample 40,40" >> $num.gplot
8  echo "splot x*x*exp(-x*x)*y*y*exp(-y*y)" >> $num.gplot
9  gnuplot < $num.gplot
```

1.2 解 説

1行目 : #! のあとにシェルのパスを書くことでシェルコマンドとして認識される。

2行目 : 変数num に、e095739 という値を代入する。

3行目 : echo で "set terminal png" をnum.gplot にリダイレクションする。
gplotで描いたグラフをpng形式の画像ファイルにする。

4行目 : echo で"set output \"$num.png\"" をnum.gplot に追加書き込み。
画像ファイルの名前を e095739.png にする。

5行目 : echo "set xrange [-2:2]" をnum.gplot に追加書き込み。
変数xの範囲を -2 ～ 2 とする。

6行目 : echo "set yrange [-2:2]" をnum.gplot に追加書き込み。
変数yの範囲を -2 ～ 2 とする。

7行目 : echo "set isosample 40,40" をnum.gplot に追加書き込み。
メッシュの間隔をx,yそれぞれ40,40とする。

8行目 : echo "splot x*x*exp(-x*x)*y*y*exp(-y*y)" をnum.gplotに追加書き込み。
3次元のグラフのときは splot を使う。

9行目 : gnuplot への入力をnum.gplotとする。

1.3 考 察

このシェルスクリプトを実行することで、e095739.gplot というファイルと、e095739.png という画像ファイルができる。

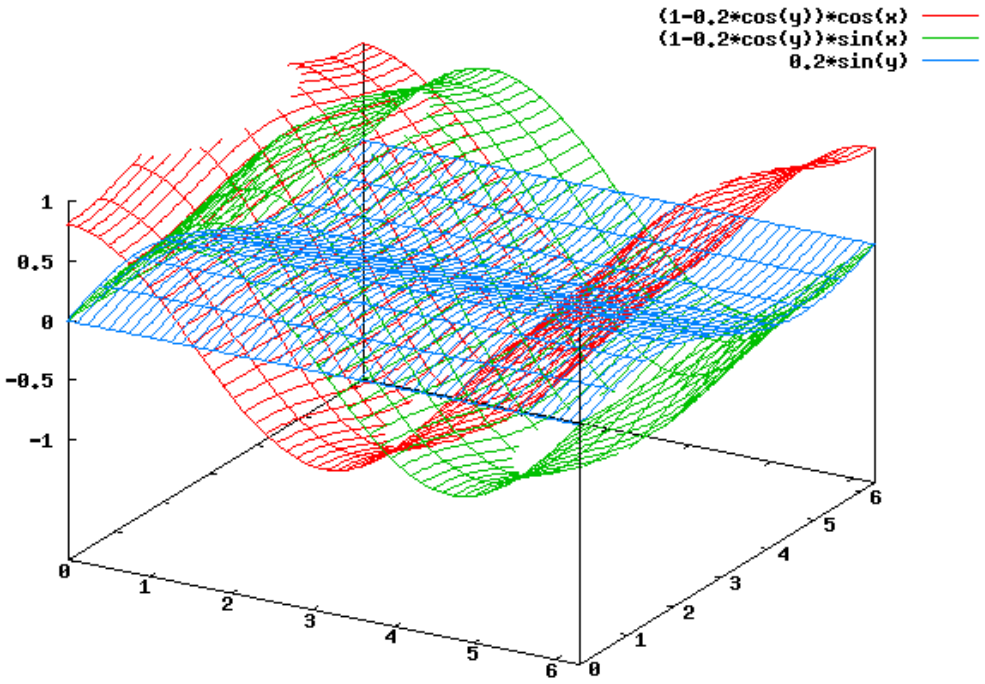
シェルスクリプトは、1行1行逐次解釈しながら実行するインタプリタなプログラムなので、比較的理解し易かった。

2. オリジナルのシェルスクリプト & グラフ

2.1 スクリプト ： plot2.sh

```
 1  #!/bin/sh
 2  num=e095739_1
 3  echo "set terminal png" > $num.gplot
 4  echo "set output \"$num.png\"" >> $num.gplot
 5  echo "set xrange [  0 : 6.28319 ] " >> $num.gplot
 6  echo "set yrange [  0 : 6.28319 ] " >> $num.gplot
 7  echo "set zrange [ -1 : 1 ] " >> $num.gplot
 8  echo "set isosample 50, 10" >> $num.gplot
 9  echo "splot (1-0.2*cos(y))*cos(x),(1-0.2*cos(y))*sin(x),0.2
      *sin(y)" >> $num.gplot
10  gnuplot < $num.gplot
```

2.2 グラフ ： e095739_1.png



2.3 感　想

同じようなスクリプトでも、あたえる式と x, y, z の範囲によって全然雰囲気の違うグラフが出来た。

本当はもっと立体的なグラフを作りたかったのですが、gnuplot 上で直接実行するとグラフが上手く出来たのですが、シェルスクリプトにすると色々エラーがでてきたので、断念しました。

今回はじめて gnuplot を使って結構楽しかったので、もっと色々なグラフを試してみたいと思っています。