# 7回目 グラフ作成ライブラリmlibの使い方、
# ２次元画像

# 2次元データの表示

1次元データ

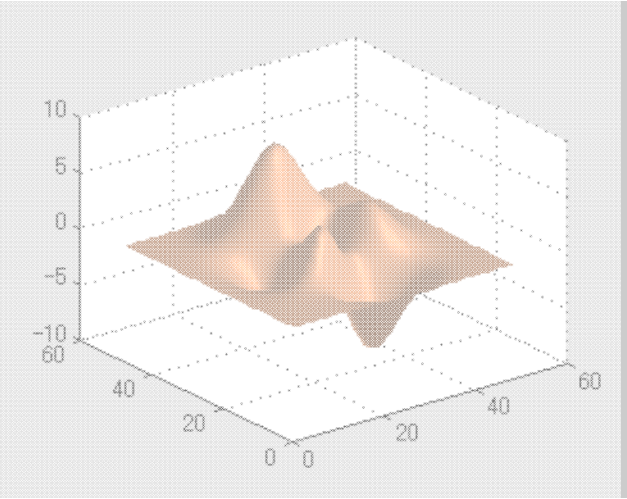
一つのパラメータ変化に着目してある応答（変動）を数値化したもの

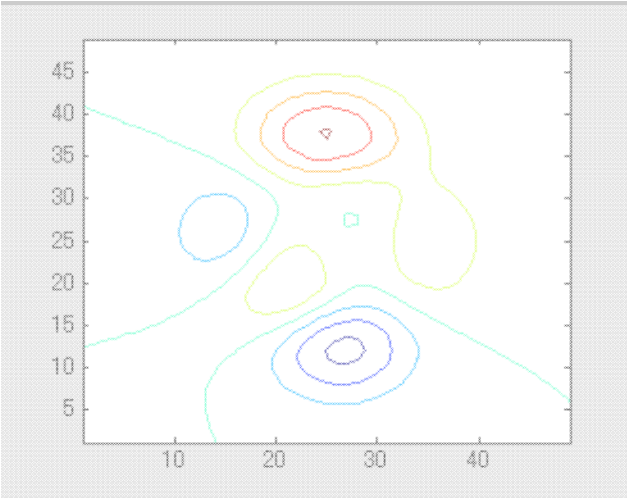
2次元データ

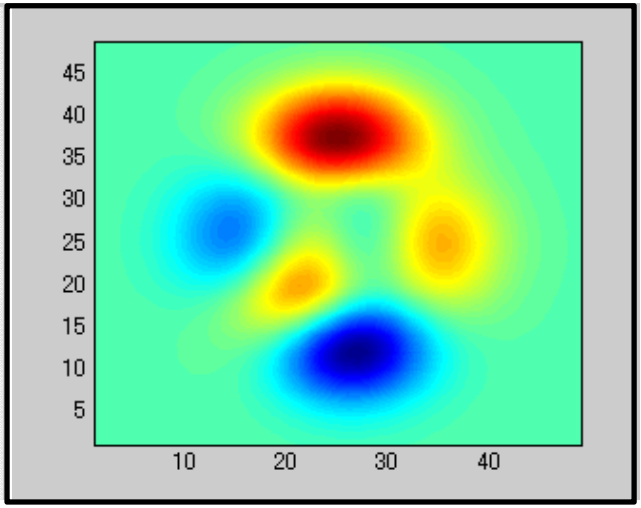
二つの独立なパラメータ変化に着目してある応答（変動）を数値化したもの

画像、1次元データが時間変化する場合
1次元データが場所によって変化する場合

**2次元データは一般に$z = f(x, y)$のように表される**



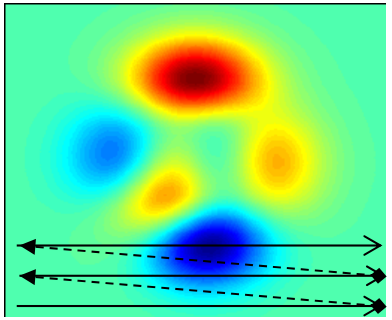



# 2次元データの表示関数のデータ配置

```
void Plot2d(double *yn, int nx, int ny, double cmin, double cmax
                                    ,int cbflug, int plflug, int szflug)
```

指定されているフィギュアウインドウにdouble型配列 ynの2D画像を描画。

yn ： 1次元配列。二次元データを1次元配列に並び替えたデータとして与える。
nx ： x方向（横軸のデータポイント数）
ny ： y方向（縦軸のデータポイント数）



| | | | | |
|---|---|---|---|---|
| (ny-1)nx | (ny-1)nx+1 | (ny-1)nx+2 | … | (ny)nx -1 |
| … | … | … | … | … |
| 2nx | 2nx+1 | 2nx+2 | … | 3nx-1 |
| nx | nx+1 | nx+2 | … | 2nx-1 |
| 0 | 1 | 2 | … | nx-1 |

上のように見えるべき画像をplot2dで表示させるには、表に書いたような数値の順番で二次元データを1次元に並べ替え、配列変数 yn に格納する。

図の下の行から右方向に向かって行き、右端まで行くと上の行の左端に戻る

# 2次元データの表示色指定

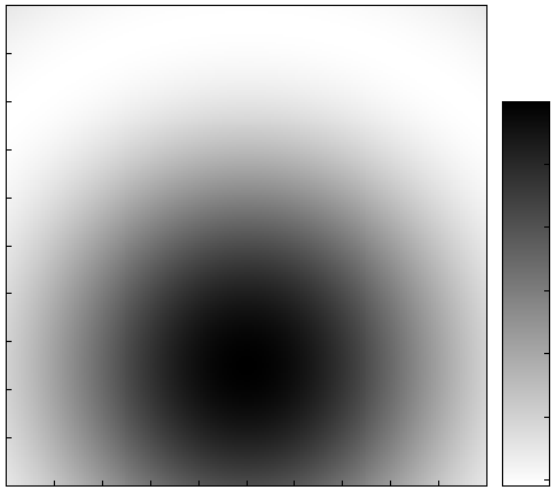
```
void Plot2d(double *yn, int nx, int ny, double cmin, double cmax
                                    ,int cbflug, int plflug, int szflug)
```

cmin ：カラースケールの最小値の指定

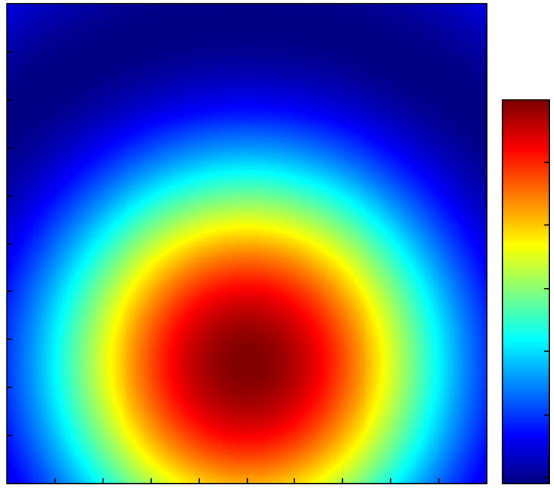
cmax ：カラースケールの最大値の指定、両方とも0のとき最大最小を自動的に探す。

cbflug ： 0　グレースケール
　　1　青→水色→黄色→オレンジ→赤
　　2　黒→青→白→赤→黒の循環色（位相の表示等に）

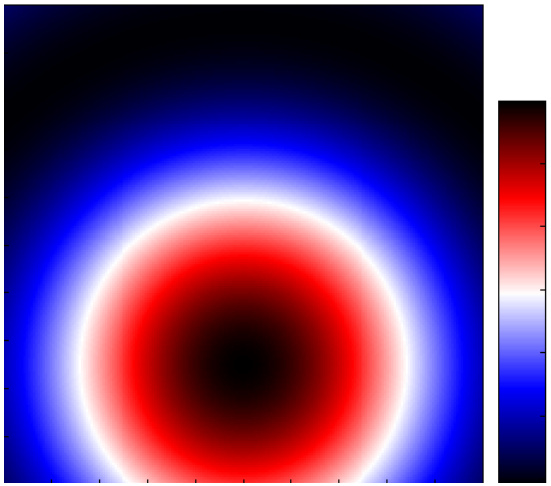
plflug ： 0　カラーバーなし
　　1　カラーバー表示



cbflug=0



cbflug=1



cbflug=2

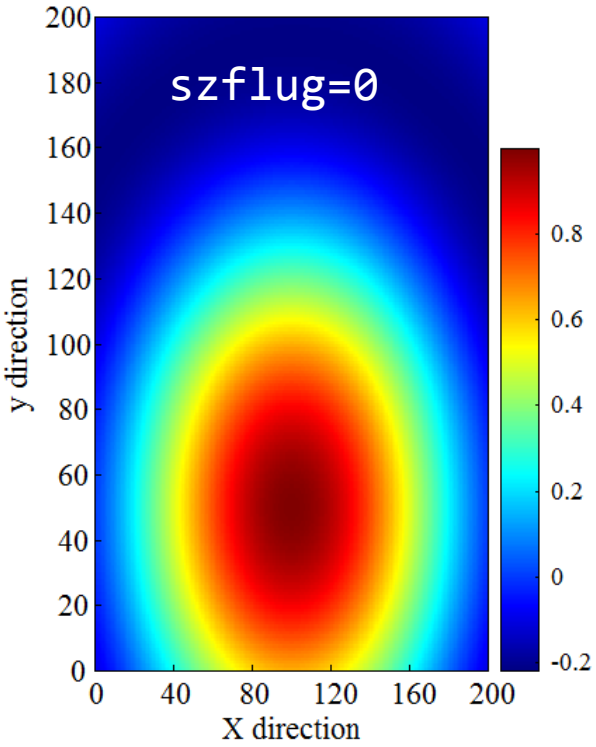
# 2次元データの表示サイズ指定

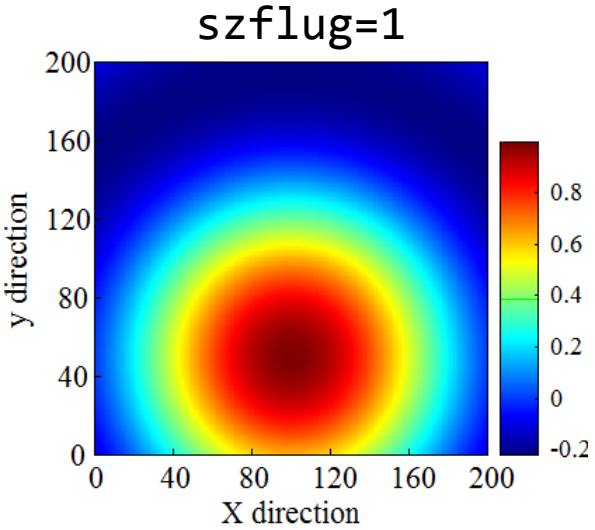
```
void Plot2d(double *yn, int nx, int ny, double cmin, double cmax
                                    ,int cbflug, int plflug, int szflug)
```

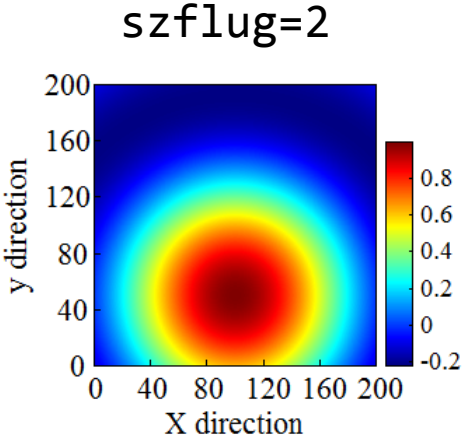
szflug : 0 画像サイズに関係なくフィギュアウインドウの規定サイズに拡大して描画（データの縦横サイズがウインドウサイズより大きいと画像が乱れるので注意）

1 フィギュアウインドウのサイズを正方形にして描画

2 データのサイズに対応したピクセル数で描画（最もきれい）



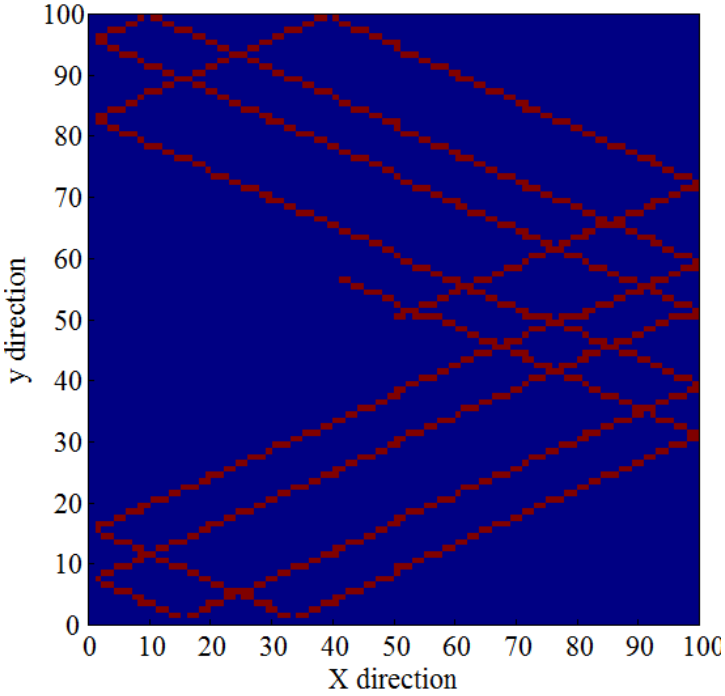
# 総合演習問題B

B-1. 重力や抵抗を無視し、正方形のフィールド内に、質量の無視できる物体に初期移動速度、方向を与え等速直線運動させる。壁で反射係数1で反射するときの物体の動きをアニメーションせよ。ただし、フィールドを100×100の仮想2次元メッシュで表し、Plot_2dを使用して表示する。

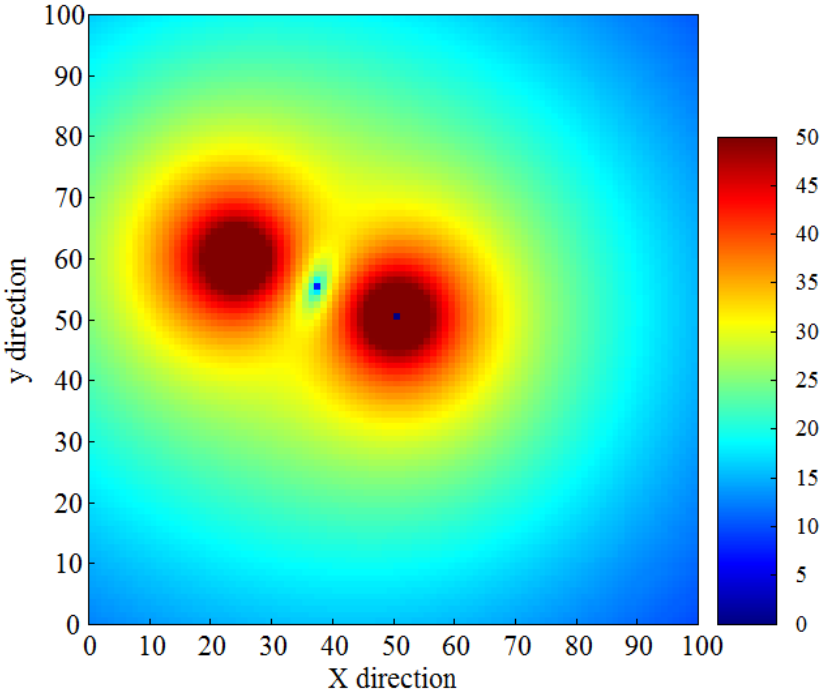
B-2. 中央に+1の電荷があり、+qの電荷がB-1のようにフィールドを動き回るとき、二つの電荷の作る合成電界分布の強さをアニメーションで示せ。電荷qはプログラムで指定、位置 $\vec{r}_0$ の電荷qが $\vec{r}$ の位置に作る電界ベクトル

$$\vec{E} = K\frac{q}{|\vec{r}-\vec{r}_0|^3}(\vec{r}-\vec{r}_0)$$

B-1の例




B-2の例



# B-1 のアルゴリズム

1. 物体 のある時間での位置は式で出すのは難しい → 既知なのは速度ベクトル $\vec{v}(t)$

$$\vec{v}(t) = \frac{d\vec{r}(t)}{dt} = \frac{\vec{r}(t+\Delta t) - \vec{r}(t)}{\Delta t}$$ 運動方程式（微分方程式）

2. 物体の位置の漸化式

運動方程式の時間を離散化 $t = n\Delta t$ → $\vec{r}(t) = \vec{r}_n$

$$\vec{v}_n = \frac{\vec{r}_{n+1} - \vec{r}_n}{\Delta t} = \begin{pmatrix} v_x \\ v_y \end{pmatrix} \Rightarrow \vec{r}_{n+1} = \vec{r}_n + \begin{pmatrix} v_x \\ v_y \end{pmatrix} \Delta t$$

3. 壁による跳ね返り

・壁面は完全反射するので、壁面の法線方向の速度成分の向きのみが逆転

・壁面の接戦方向の速度成分は変化しない

・$x$ 、$y$ 方向のボールの向きの情報を持つ $p$ 、$q$を（+1 or -1 ）考える

# B-1 のプログラム例

4．物体の当たり判定

・当たり判定の必要性　物体と壁

・壁は固定、物体の座標は既知、大きさはなし

・物体がフィールドから出たかどうかは座標で判定可

5．位置の更新

・位置ベクトルは　$x_{n+1} = x_n + p_n v_x \Delta t$
$y_{n+1} = y_n + q_n v_y \Delta t$

・物体が左右の壁から外へ出たとき　$v_x = -v_x$

・物体が上下の壁から外へ出たとき　$v_y = -v_y$

6．座標系

・物体の実フィールド範囲を -1<x , y <1 に仮定

・物体の表示上の座標を 0<= nx , ny <100 の仮想2次元座標（整数値に離散化）に変換

・Plot2Dで表示するために、2次元空間の物体位置を np=ny*100+nx で1次元配列の要素に変換

・物体が存在する配列の要素位置 E[np] に1を代入

```
int i,m,nx,ny,np;
double v,th,pi=3.141,E[N*N]={0};
double x=0,y=0,dt=0.01,vx,vy;

v=Get_double(0);
th=Get_double(1)/180*pi;
vx=v*cos(th);   //速度ベクトルx成分
vy=v*sin(th);   //速度ベクトルy成分

Set_figure(1,1,1);
for(m=1;m<=1000;m++){
 x=x+vx*dt;  //x座標更新
 y=y+vy*dt;  //y座標更新
 if (abs(x)>0.5)  vx=-vx;  //左右壁
 else if (abs(y)>0.5)  vy=-vy;  //上下壁
 else {
   nx=(int)(x*N)+N/2;  // 仮想座標へ変換
   ny=(int)(y*N)+N/2;  // 仮想座標へ変換
   np=(ny)*N+(nx);  // 1次元座標へ変換
   E[np]=1;
   Plot2d(E,N,N,0,0,1,0,1);
  }
  UpdateWindow(hWnd);
}
```

# B-2 のプログラム例

```
int i,m,nx,ny,np;
double v,th,pi=3.141,E[N*N]={0};
double x=0,y=0,dt=0.01,vx,vy;

v=Get_double(0);
th=Get_double(1)/180*pi;
vx=v*cos(th);   //速度ベクトルx成分
vy=v*sin(th);   //速度ベクトルy成分

Set_figure(1,1,1);
for(m=1;m<=1000;m++){
 x=x+vx*dt;  //x座標更新
 y=y+vy*dt;  //y座標更新
 if (abs(x)>0.5)  vx=-vx;  //左右壁
 else if (abs(y)>0.5)  vy=-vy;  //上下壁
 else {
   nx=(int)(x*N)+N/2;  // 仮想座標へ変換
   ny=(int)(y*N)+N/2;  // 仮想座標へ変換
   np=(ny)*N+(nx);  // 1次元座標へ変換
   E[np]=1;
   Plot2d(E,N,N,0,0,1,0,1);
  }
  UpdateWindow(hWnd);
}
```

配列Eが電界の大きさの2次元マップに対応

電荷の位置ベクトル $\vec{r}_0$、$\vec{r}$ は既知

100*100のグリッドの全ての位置を表す$\vec{p}$を考える

$\vec{p}$と$\vec{r}_0$間の電界$\vec{E}_0$は$\vec{s}_0 = \vec{p} - \vec{r}_0$とすると

$$\vec{E}_0 = K\frac{1}{|\vec{s}_0|^3}\vec{s}_0$$

$\vec{p}$ と$\vec{r}$間の電解$\vec{E}$ は$\vec{s} = \vec{p} - \vec{r}$ とすると

$$\vec{E} = K\frac{q}{|\vec{s}|^3}\vec{s}$$

点 $\vec{p}$での二つの電荷による合成電界$\vec{E}_s$は$\vec{E} + \vec{E}_0$

配列Ex[]、Ey[]を定義し、$\vec{E}_s$ のx成分,y成分を計算して代入

Ex[]、Ey[]から得られるベクトル成分から、電界の大きさを求め、配列列E[]に入れる