# Observation Plane Transformation for Aperture Field Distribution

# Data Processing for Near Field Measurement

2003/2/24 **平野拓一（東京工業大学）**
Takuichi Hirano (Tokyo Institute of Technology)

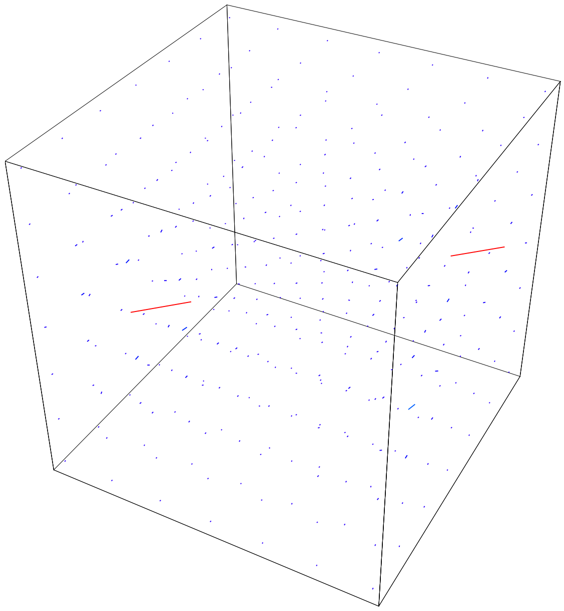
## *Source: Infinitesimal dipole*

$$\sin\theta = \frac{\sqrt{x^2+y^2}}{r}$$
$$\cos\theta = \frac{z}{r}$$
$$\sin\varphi = \frac{y}{\sqrt{x^2+y^2}}$$
$$\cos\varphi = \frac{x}{\sqrt{x^2+y^2}}$$
$$\hat{\varphi} = -\hat{x}\,\sin\varphi + \hat{y}\,\cos\varphi$$

In[8]:=
```
freq = 2.45 * 10^9;
c = 2.998 * 10^8;
λ0 = c / freq;
k0 = (2 * π) / λ0;
η0 = 120. * π;

EFieldInfinitesimalDipole[xo_, yo_, zo_, xs_, ys_, zs_] :=
 Module[{r, x, y, z, sφ, cφ},
  x = xo - xs;
  y = yo - ys;
  z = zo - zs;
  r = Sqrt[x^2 + y^2 + z^2];
  sθ = Sqrt[x^2 + y^2] / r;
  cθ = z / r;
  sφ = y / Sqrt[x^2 + y^2];
  cφ = x / Sqrt[x^2 + y^2];
  (* 単位ベクトル *)
  rhat = {sθ * cφ, sθ * sφ, cθ};
  θhat = {cθ * cφ, cθ * sφ, -sθ};
  φhat = {-sφ, cφ, 0};
  rhat * η0 / (2 * π * r^2) * Exp[-I * k0 * r] / r * (1 + 1 / (I * k0 * r)) * cθ +
   θhat * I * (k0 * η0) / (4 * π * r) * Exp[-I * k0 * r] / r * (1 + 1 / (I * k0 * r) + 1 / (I * k0 * r)^2) * sθ
 ];
EField[xo_, yo_, zo_] := Module[{},
  (* (x̂,ŷ,ẑ)→(ẑ,x̂,ŷ) *)
  e1 = EFieldInfinitesimalDipole[zo, xo, yo, 0, -5 * λ0, -5 * λ0];
  e2 = EFieldInfinitesimalDipole[zo, xo, yo, 0, 5 * λ0, 5 * λ0];
  e3 = EFieldInfinitesimalDipole[zo, xo, yo, 0, 10 * λ0, -5 * λ0];
  e = e1 + e2 + e3;
  {e[[2]], e[[3]], e[[1]]}
 ];
```

# Vector Filed

```
In[15]:= << Graphics`PlotField3D`;
         PlotVectorField3D[Abs[EField[x, y, z]], {x, -1, 1}, {y, -1, 1}, {z, -1, 1},
          ColorFunction → (Hue[-0.7 * (# - 1), 1, 1] &)]
```

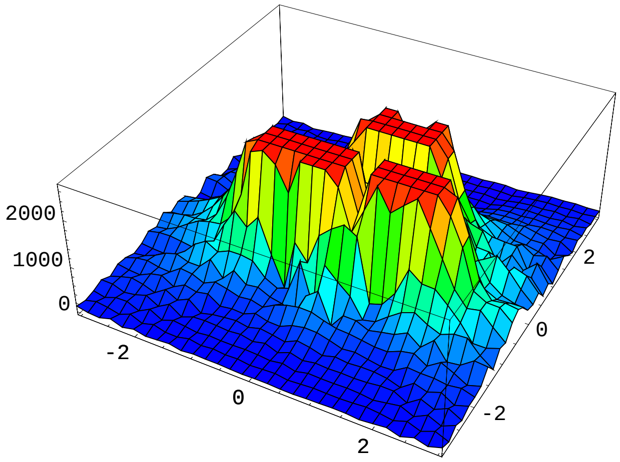


```
Out[16]= - Graphics3D -
```

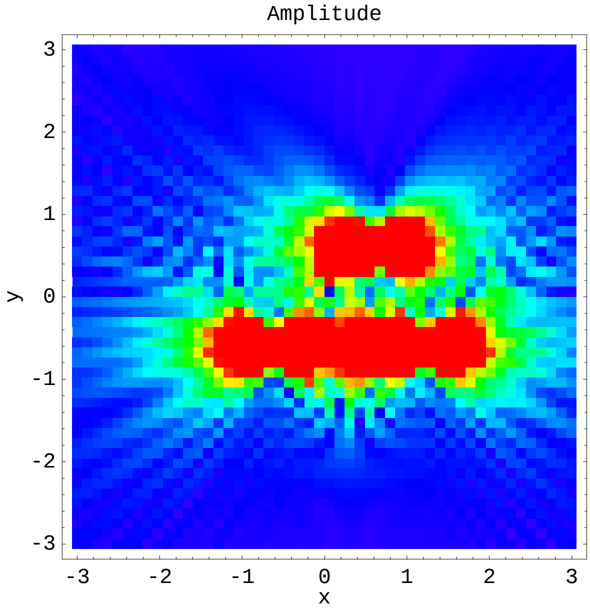
# *Ey Distribution*
# *目的：電界のy成分を描く*

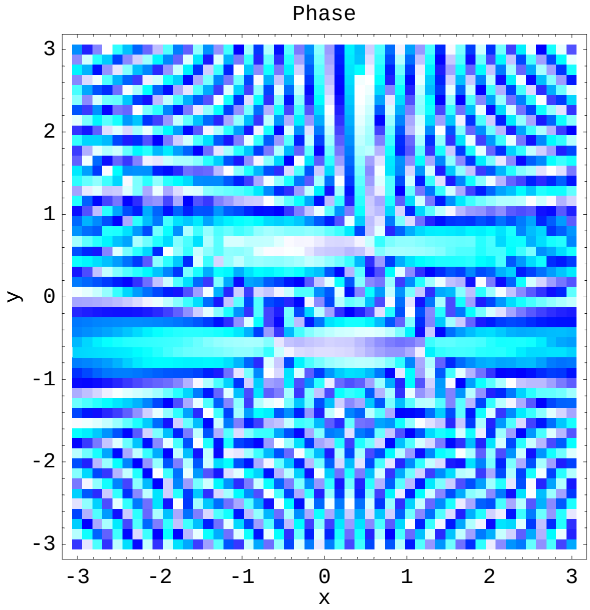
## *関数から分布を描く*

```
In[20]:= RegX = 50. * λ0;
         RegY = 50. * λ0;

         Plot3D[Abs[ EField[x, y, λ0][[2]] ], {x, -RegX / 2, RegX / 2}, {y, -RegY / 2, RegY / 2},
           PlotPoints → 30,
           ColorFunction → (Hue[-0.7 * (# - 1), 1, 1] &)];
```




```
In[28]:= DensityPlot[Abs[ EField[x, y, 0.1 * λ0][[2]] ],
           {x, -RegX / 2, RegX / 2}, {y, -RegY / 2, RegY / 2},
           Mesh → False,
           PlotPoints → 50,
           PlotRange → {0, Automatic},
           PlotLabel → "Amplitude",
           FrameLabel → {"x", "y", "", ""},
           ColorFunction → (Hue[-0.7 * (# - 1), 1, 1] &)];

         colfun[x_] := RGBColor[0, 3 * x, 1] /; (x ≤ 1 / 3);
         colfun[x_] := RGBColor[3 * (x - 1 / 3), 1, 1] /; (1 / 3 < x ≤ 2 / 3);
         colfun[x_] := RGBColor[-3 * (x - 2 / 3) + 1, -3 * (x - 2 / 3) + 1, 1] /; (2 / 3 < x);

         DensityPlot[Arg[ EField[x, y, 0.1 * λ0][[2]] ] * 180/π,
           {x, -RegX / 2, RegX / 2}, {y, -RegY / 2, RegY / 2},
           Mesh → False,
           PlotPoints → 50,
           PlotRange → {-180, 180},
           PlotLabel → "Phase",
           FrameLabel → {"x", "y", "", ""},
           ColorFunction → colfun];
```

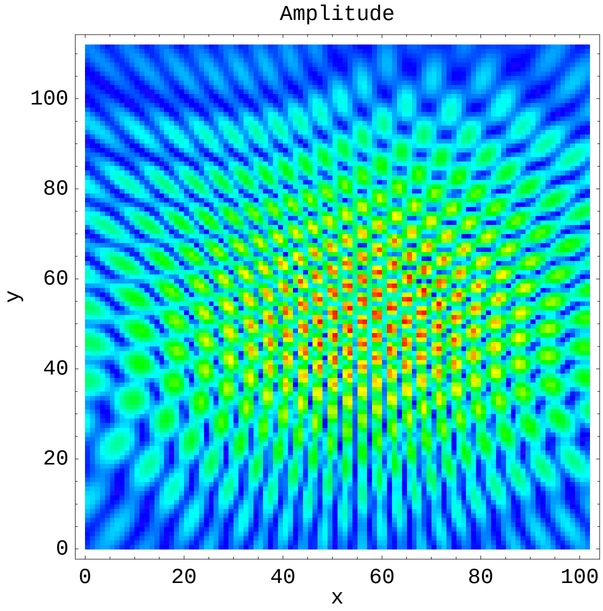

Amplitude
3
2
1
0
-1
-2
-3
y
-3 -2 -1 0 1 2 3
x

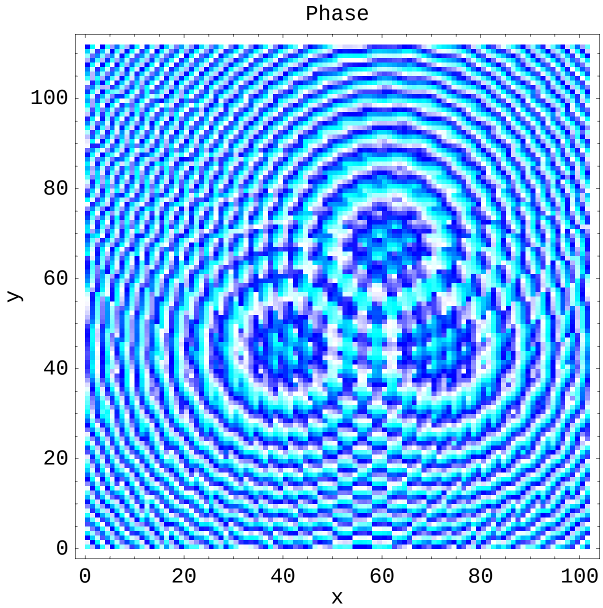

Phase
3
2
1
0
-1
-2
-3
y
-3 -2 -1 0 1 2 3
x

# Make List
# 目的：実験のサンプリングデータをシミュレートするために離散点でサンプリングし、リストを作る

## 一度リストを作ってから分布を描く

```
In[33]:= m = 101;
         n = 111;
         APDist1 = Table[
            Table[
             EField[x, y, 20.0 * λ0][[2]],
             {x, -RegX / 2, RegX / 2, RegX / m}
            ]
            , {y, -RegY / 2, RegY / 2, RegY / n}];

In[36]:= ListDensityPlot[Abs[ APDist1 ],
           Mesh → False,
           PlotRange → {0, Automatic},
           PlotLabel → "Amplitude",
           FrameLabel → {"x", "y", "", ""},
           ColorFunction → (Hue[-0.7 * (# - 1), 1, 1] &)];

         colfun[x_] := RGBColor[0, 3 * x, 1] /; (x ≤ 1 / 3);
         colfun[x_] := RGBColor[3 * (x - 1 / 3), 1, 1] /; (1 / 3 < x ≤ 2 / 3);
         colfun[x_] := RGBColor[-3 * (x - 2 / 3) + 1, -3 * (x - 2 / 3) + 1, 1] /; (2 / 3 < x);

         ListDensityPlot[Arg[APDist1] * 180/π,
          Mesh → False,
          PlotRange → {-180, 180},
          PlotLabel → "Phase",
          FrameLabel → {"x", "y", "", ""},
          ColorFunction → colfun]
```

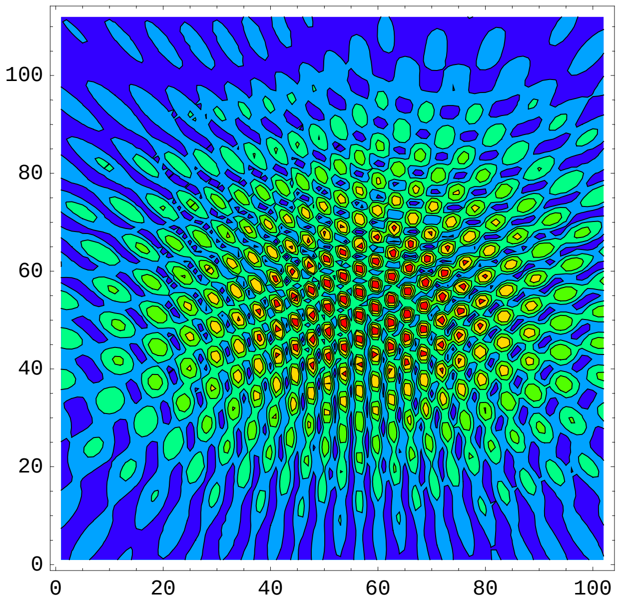


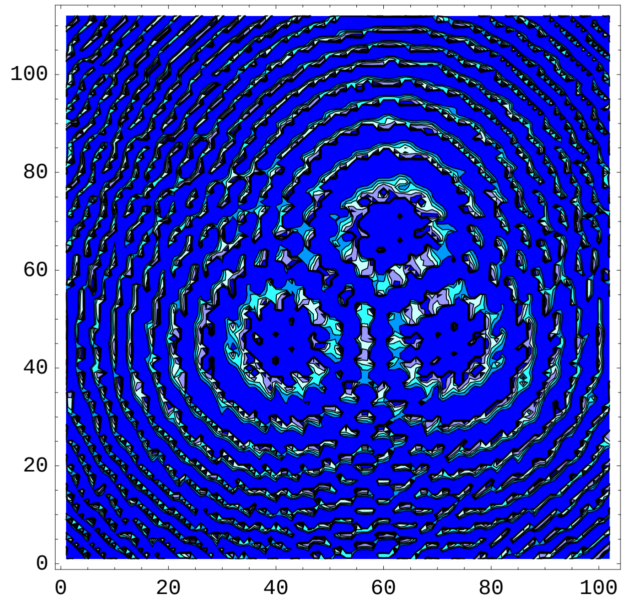




*Out[40]=* - DensityGraphics -

# 等高線図を描く

```
In[41]:= ListContourPlot[Abs[ APDist1 ],
           PlotRange → {0, Automatic},
           Contours → 5,
           ColorFunction → (Hue[-0.7 * (# - 1), 1, 1] &)];

         colfun[x_] := RGBColor[0, 3 * x, 1] /; (x ≤ 1 / 3);
         colfun[x_] := RGBColor[3 * (x - 1 / 3), 1, 1] /; (1 / 3 < x ≤ 2 / 3);
         colfun[x_] := RGBColor[-3 * (x - 2 / 3) + 1, -3 * (x - 2 / 3) + 1, 1] /; (2 / 3 < x);

         ListContourPlot[Arg[APDist1] * 180/π,
          PlotRange → {0, Automatic},
          Contours → 5,
          ColorFunction → colfun]
```

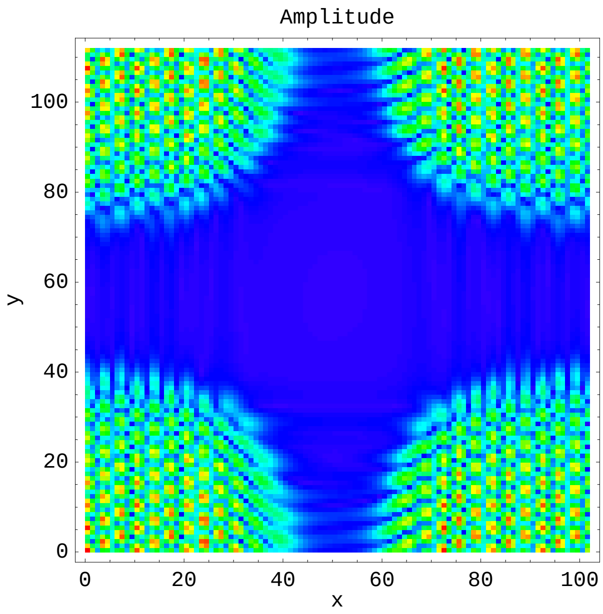


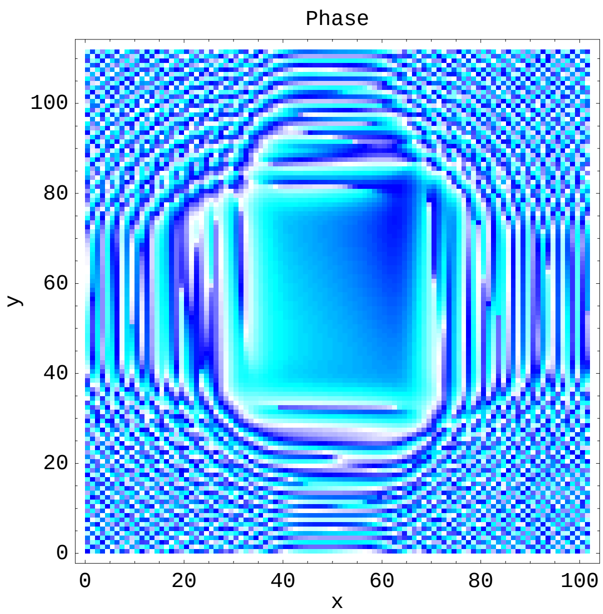



Out[45]= - ContourGraphics -

## DFT
## 目的：離散フーリエ変換してスペクトルを描く

In[52]:=
```
FAPDist1 = Fourier[APDist1];

ListDensityPlot[Abs[ FAPDist1 ],
  Mesh → False,
  PlotRange → {0, Automatic},
  PlotLabel → "Amplitude",
  FrameLabel → {"x", "y", "", ""},
  ColorFunction → (Hue[-0.7 * (# - 1), 1, 1] &)];

colfun[x_] := RGBColor[0, 3 * x, 1] /; (x ≤ 1 / 3);
colfun[x_] := RGBColor[3 * (x - 1 / 3), 1, 1] /; (1 / 3 < x ≤ 2 / 3);
colfun[x_] := RGBColor[-3 * (x - 2 / 3) + 1, -3 * (x - 2 / 3) + 1, 1] /; (2 / 3 < x);

ListDensityPlot[Arg[FAPDist1] * 180/π,
 Mesh → False,
 PlotRange → {-180, 180},
 PlotLabel → "Phase",
 FrameLabel → {"x", "y", "", ""},
 ColorFunction → colfun]
```

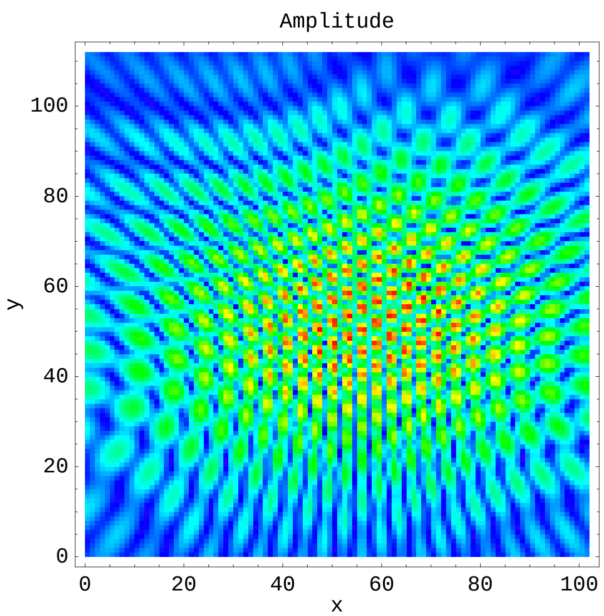


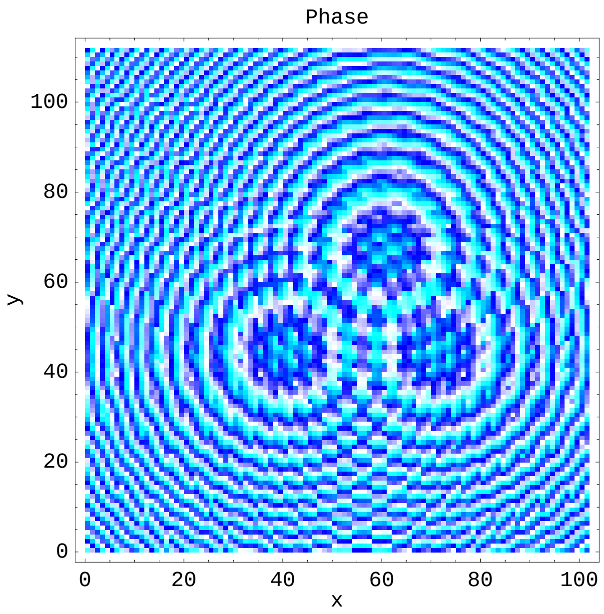




*Out[57]=* - DensityGraphics -

# Inverse DFT
# 目的：単純に逆フーリエ変換して元に戻ることを確認する。

In[64]:=
```
IFAPDist1 = InverseFourier[FAPDist1];

ListDensityPlot[Abs[ IFAPDist1 ],
  Mesh → False,
  PlotRange → {0, Automatic},
  PlotLabel → "Amplitude",
  FrameLabel → {"x", "y", "", ""},
  ColorFunction → (Hue[-0.7 * (# - 1), 1, 1] &)];

colfun[x_] := RGBColor[0, 3 * x, 1] /; (x ≤ 1 / 3);
colfun[x_] := RGBColor[3 * (x - 1 / 3), 1, 1] /; (1 / 3 < x ≤ 2 / 3);
colfun[x_] := RGBColor[-3 * (x - 2 / 3) + 1, -3 * (x - 2 / 3) + 1, 1] /; (2 / 3 < x);

ListDensityPlot[Arg[IFAPDist1] * 180/π,
 Mesh → False,
 PlotRange → {-180, 180},
 PlotLabel → "Phase",
 FrameLabel → {"x", "y", "", ""},
 ColorFunction → colfun]
```



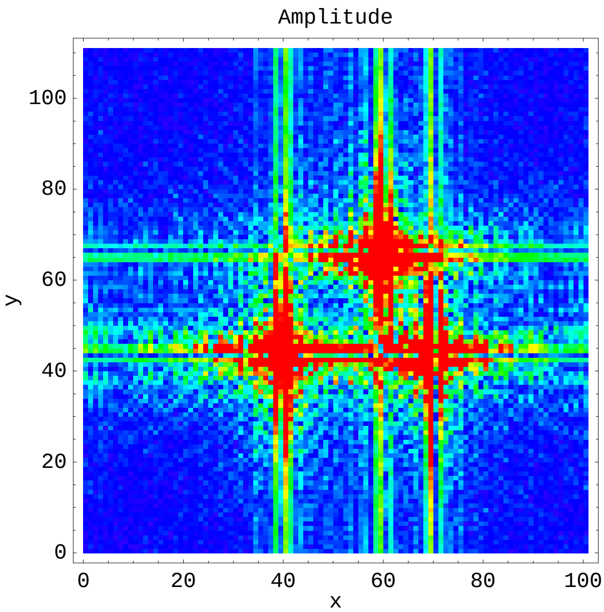


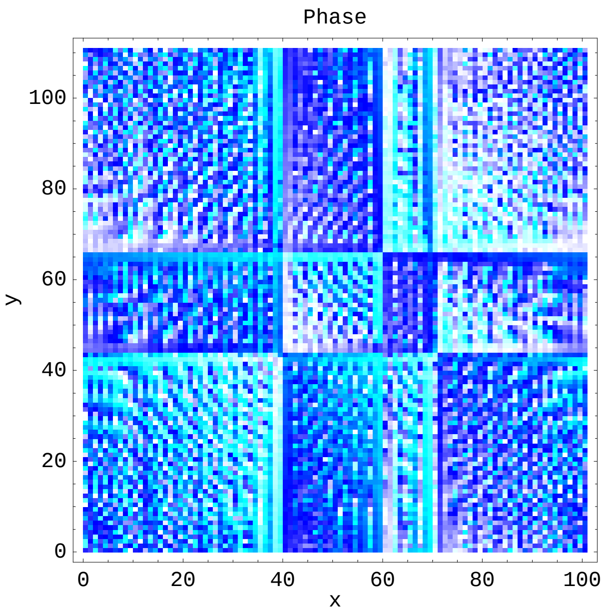

Out[69]= - DensityGraphics -

## *Observation Plane Transformation for Aperture Field Distribution Inverse DFT*
***目的：任意の観測面の開口分布に変換する***
***使い方：元の開口分布がz=0のときのものだったとするとdz でz=0から移動する高さを指定する***

In[73]:=
```
dz = -20. * λ0;
FAPDist2 = Table[
  Module[{},
   kx = (j - 1) * (2 * π)/RegX;
   ky = (i - 1) * (2 * π)/RegY;
   If[i > IntegerPart[n / 2], ky -= n * (2 * π)/RegY];
   If[j > IntegerPart[m / 2], kx -= m * (2 * π)/RegX];
   kz = Sqrt[k0^2 - kx^2 - ky^2];
   If[(Re[kz] ≥ 0) && (Abs[Im[kz]] < 10^-15),
    FAPDist1[[i, j]] * Exp[-I * kz * dz],
    10^-15]
  ]
  , {i, 1, n}
  , {j, 1, m}
 ];
IFAPDist2 = InverseFourier[FAPDist2];
```

```
In[76]:= ListDensityPlot[Abs[ IFAPDist2 ],
           Mesh → False,
           PlotRange → {0, Automatic},
           PlotLabel → "Amplitude",
           FrameLabel → {"x", "y", "", ""},
           ColorFunction → (Hue[-0.7 * (# - 1), 1, 1] &)];

         colfun[x_] := RGBColor[0, 3 * x, 1] /; (x ≤ 1 / 3);
         colfun[x_] := RGBColor[3 * (x - 1 / 3), 1, 1] /; (1 / 3 < x ≤ 2 / 3);
         colfun[x_] := RGBColor[-3 * (x - 2 / 3) + 1, -3 * (x - 2 / 3) + 1, 1] /; (2 / 3 < x);

         ListDensityPlot[Arg[IFAPDist2] * 180/π,
          Mesh → False,
          PlotRange → {-180, 180},
          PlotLabel → "Phase",
          FrameLabel → {"x", "y", "", ""},
          ColorFunction → colfun]
```






Out[80]= - DensityGraphics -