

## 【2次元離散フーリエ変換 2D Discrete Fourier Transform】

### ■ 2D DFT

$$F(u, v) = \sum_{y=0}^{N_y-1} \sum_{x=0}^{N_x-1} f(x, y) \exp[-j2\pi\{\frac{ux}{N_x} + \frac{vy}{N_y}\}]$$

### ■ 2D IDFT

$$f(x, y) = \frac{1}{N_x N_y} \sum_{v=0}^{N_y-1} \sum_{u=0}^{N_x-1} F(u, v) \exp[j2\pi\{\frac{ux}{N_x} + \frac{vy}{N_y}\}]$$

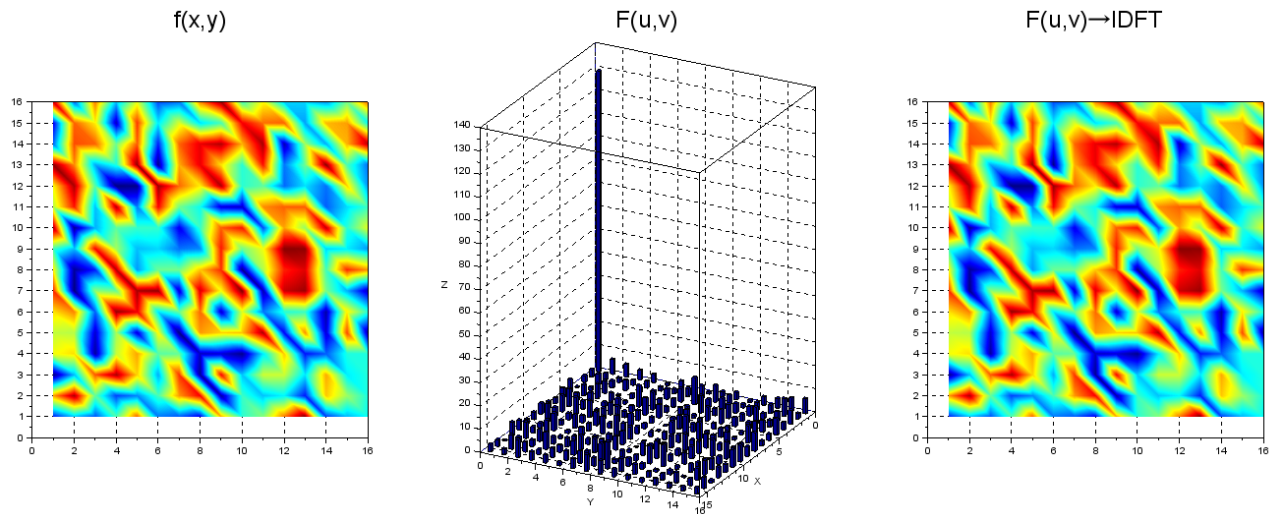


Figure 1: Scilab 実行結果

Source Code 1: Scilab

```

////////////////////////////////////
//      2次元離散フーリエ変換
//      2D Discrete Fourier Transform
//
//                                  M.Tsutsui
////////////////////////////////////

clear all;

pi=%pi;//π
j=%i;//虚数

fxy_table=rand(16,16);//f(x,y)

[size1,size2]=size(fxy_table);

N=size1;

//-----DFT-----//

//----F(u,v)-----//
Fuv=[];
calc=0;

for u=1:1:size1;
    for v=1:1:size1;

```

```

        for x=0:size1-1;
            for y=0:size1-1;
                calc=calc+fx_y_table(x+1,y+1)*exp(j*-2*pi*((u-1)*x/N+(v-1)*y/N));
            end
        end
        Fuv(u,v)=calc;
        calc=0;
    end
end
//-----DFT-----//

//-----IDFT-----//
//----f(x,y)-----//
fxy=[];
calc_i=0;
for y_i=1:1:size1;
    for x_i=1:1:size1;
        for u_i=0:size1-1;
            for v_i=0:size1-1;
                calc_i=calc_i+Fuv(u_i+1,v_i+1)*exp(j*2*pi*(u_i*(x_i-1)/N+v_i*(y_i-1)/N));
            end
        end
        fxy(x_i,y_i)=calc_i;
        calc_i=0;
    end
end
//-----IDFT-----//

//_Plot_//////////
x1_p=linspace(1,N,N);
x2_p=x1_p;

subplot(1,3,1)
Sgrayplot(x1_p,x2_p,abs(fxy_table), strf="041")
set(gcf(),'color_map',jetcolormap(256));
xgrid ();
title("f(x,y)", 'fontsize',5);

subplot(1,3,2)
hist3d(abs(Fuv));
xgrid ();
title("F(u,v)", 'fontsize',5);

subplot(1,3,3)
Sgrayplot(x1_p,x2_p,abs(fxy), strf="041")
set(gcf(),'color_map',jetcolormap(256));
xgrid ();
title("F(u,v)→IDFT", 'fontsize',5);

```

## Source Code 2: Python

```

#-----
# Module Name:2D Discrete Fourier Transform
# Author: m_tsutsui
#-----

#Library_Import#####
from numpy import*
import math, numpy as np
import matplotlib.pyplot as plt
#Library_Import_end#####

```

```

def dft_2d(u,v): #DFT function
    loop=[]
    for y in np.arange(0,size1,1):
        for x in np.arange(0,size1,1):
            loop.append(exp(-2j*pi*(u*x/N+v*y/N)))

    squ_mr=[]

    for K in np.arange(1,N+1,1):
        squ_mr.append(loop[N*K-N:N*K])

    F_u_v=np.sum(array(squ_mr)*fxy_table)

    return F_u_v

def idft_2d(x,y): #IDFT function

    loop_i=[]
    for v in np.arange(0,size1,1):
        for u in np.arange(0,size1,1):
            loop_i.append(exp(2j*pi*(u*x/N+v*y/N)))

    squ_mr_i=[]
    for K_i in np.arange(1,N+1,1):
        squ_mr_i.append(loop_i[N*K_i-N:N*K_i])

    f_xy=np.sum(array(squ_mr_i)*Fuv)

    return f_xy

if __name__ == '__main__':

    """_8*_f(x,y)_table_comment_out
    #_f(x.y)_#####
    f00=61; f10=40; f20=42; f30=55; f40=49; f50=39; f60=44; f70=51;
    f01=60; f11=51; f21=51; f31=50; f41=44; f51=47; f61=53; f71=48;
    f02=55; f12=59; f22=60; f32=46; f42=37; f52=50; f62=57; f72=43;
    f03=46; f13=57; f23=65; f33=52; f43=37; f53=44; f63=52; f73=44;
    f04=41; f14=49; f24=66; f34=67; f44=46; f54=36; f64=45; f74=51;
    f05=43; f15=43; f25=63; f35=77; f45=58; f55=39; f65=45; f75=59;
    f06=45; f16=39; f26=55; f36=73; f46=63; f56=49; f66=51; f76=57;
    f07=46; f17=37; f27=47; f37=62; f47=61; f57=56; f67=55; f77=51;

    #_f(x.y)_Table_#####
    fxy_table=array([[f00, f10, f20, f30, f40, f50, f60, f70]
    [f01, f11, f21, f31, f41, f51, f61, f71]
    [f02, f12, f22, f32, f42, f52, f62, f72]
    [f03, f13, f23, f33, f43, f53, f63, f73]
    [f04, f14, f24, f34, f44, f54, f64, f74]
    [f05, f15, f25, f35, f45, f55, f65, f75]
    [f06, f16, f26, f36, f46, f56, f66, f76]
    [f07, f17, f27, f37, f47, f57, f67, f77]])
    """
    fxy_table=np.random.rand(16,16) #16*16 random matrix

    N=np.size(fxy_table,0)

    size1=np.size(fxy_table,0)

    Fuv_buf=[]

```

```

for j in np.arange(0,size1,1):
    for i in np.arange(0,size1,1):
        Fuv_buf.append(dft_2d(i,j)) # i,j Function Parameters

Fuv=[]
for cnt in np.arange(1,N+1,1): #square matrix
    Fuv.append(Fuv_buf[N*cnt-N:N*cnt])

print(Fuv) #F(u,v) print

fxy_buf=[]
for j in np.arange(0,size1,1):
    for i in np.arange(0,size1,1):
        fxy_buf.append(idft_2d(i,j)) # i,j Function Parameters

fxy=[]
for cnt in np.arange(1,N+1,1): #square matrix
    fxy.append(fxy_buf[N*cnt-N:N*cnt])

print((1/N**2)*np.abs(fxy)) #f(x,y) print

#plot_command#####
x1_p=np.linspace(0,N,N)
x1,x2=np.meshgrid(x1_p,x1_p)

plt.figure(facecolor='w')
plt.subplot(131)
plt.contourf(x1,x2,fxy_table,100)
plt.grid()
plt.title('f(x,y)')

plt.subplot(132)
plt.contourf(x1, x2, Fuv,100)
plt.grid()
plt.title('F(u,v)')

plt.subplot(133)
plt.contourf(x1,x2,fxy,100)
plt.grid()
plt.title('F(u,v)-->f(x,y)')
plt.show()
#end#####

```