# GPGN 411:511 Advanced Gravity and Magnetic Exploration
## Lab Exercise #04: Review of Fourier Transform

**Name: Nadima Dwihusna**
**Lab Date: 10/10/2017**

**Objectives**:
Understanding the Fourier transform of 2D data maps

**Fast Fourier Transform**
We will use MATLAB to explore some of the properties important in the wavenumber domain processing of potential-field maps, and to gain intuitive understanding of the transform results. Provided for this lab exercise are two MATLAB codes for performing and displaying the Fourier transform of real data (non-complex) and a group of simplistic data files with special features.

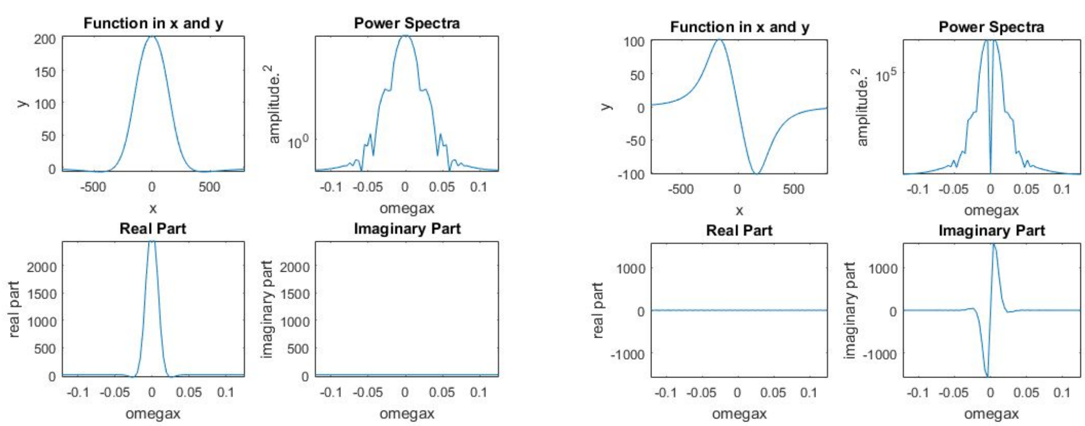
**Code:**
(1) ft1d.m: performs 1D FFT and displays the result.
(2) ft2d.m: performs 2D FFT and displays the result

**Useful observations to make:**

1. The relative magnitude of real and imaginary parts of the transform (especially with even and odd functions). ***(see problem 1 explanation )***
2. Relationship between the width of the box-car function and the "width" of its Fourier transform in terms of the wavenumber band with significant power. ***(see problem 3 explanation)***
3. Directions of elongation in wavenumber domain relative to that in spatial domain. ***(see problem 4 explanation)***

**Tasks**

1. **Run ft1d on even_1d.dat and odd_1d.dat to observe the difference in the corresponding power spectra, real and imaginary components in the Fourier domain for 1D functions.**




(a) even_1d.dat (b) odd_1d.dat

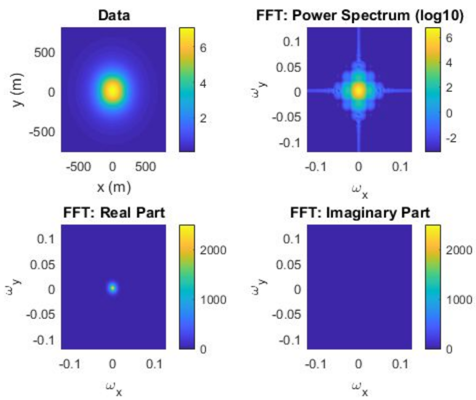
**Figure 1**: Perform 1D FFT on (a) even_1d.dat and (b) odd_1d.dat 1D data profile.

As seen in the figure 1 above, the 1D Fast Fourier Transform is performed to an even and odd ID data profile. There are several differences in the corresponding power spectra, real, and imaginary components in the Fourier domain for the two 1D functions.

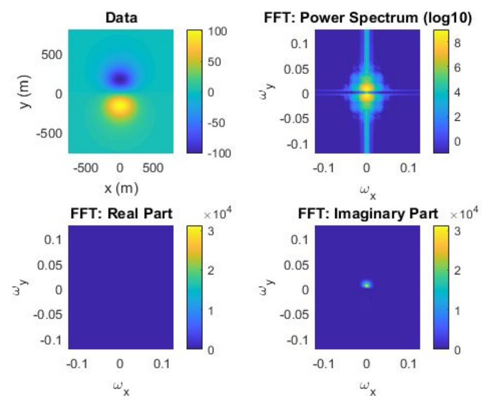
Figure 1 (a) represents a 1D even function with the highest y positive value of 200 at x=0. The power spectra of this even function has the highest amplitude at omega_x = 0 and approaches a very low amplitude of 0 as omega_x increases and decreases. The real part of this even function has a positive high amplitude of approximately 2500 at omega_x = 0 and approaches a very low amplitude of 0 as omega_x increases and decreases. In the real part, the amplitude reaches zero at approximately omega_x=-0.025 and omega_x=0.025. There are no imaginary parts in this even function as the amplitude of imaginary part remains 0 throughout omega_x.

Figure 1(b) represents a 1D odd function with high y amplitude of 100 as x approaches 0 from the negative side, a high negative amplitude of -100 as x approaches 0 from the positive side, and y=0 when x=0. The power spectra of this odd function shows an increase in amplitude as omega_x approaches 0 from the negative and positive direction, although at omega_x=0, the amplitude drastically decreases to 0. There are no real parts in this odd function. The imaginary part shows a high negative amplitude of approximately -1250 as omega_x approaches 0 from the negative side, a high positive amplitude of 1250 as omega_x approaches 0 from a positive part, and an amplitude of 0 at omega_x=0.

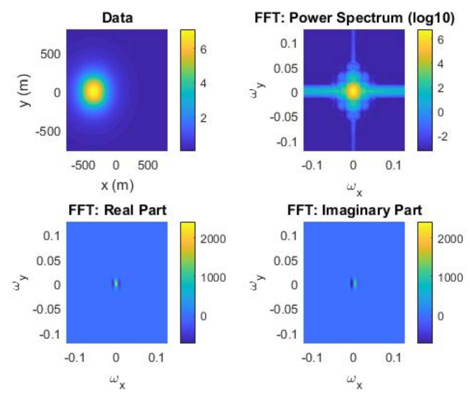
2. **Run ft2d on even.dat, odd.dat, and off.dat to observe the differences in the corresponding power spectra, real and imaginary components for 2D function.**




(a) even.dat




(b) odd.dat




(c) off.dat

**Figure 2:** Perform 2D FFT on (a) even.dat, (b) odd.dat, and (c) off.dat 2D data profile.

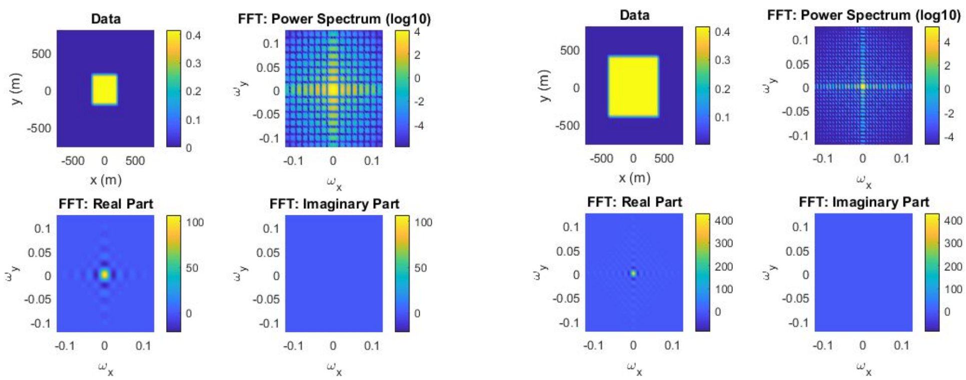
As seen in the figure 2 above, the 2D Fast Fourier Transform is performed to an even, odd, and off 2D function. There are several differences in the corresponding power spectra, real, and imaginary components in the Fourier domain for the two 2D functions.

Figure 2 (a) represents an even 2D data with a high amplitude of approximately 6 at x=0 m and y=0 m while the rest of the data goes to approximately 0 throughout positive and negative x and y. The 2D fast fourier transform power spectra shows a high amplitude of 6 at omega_x=0 m and omega_y=0 m while approaches to zero in all positive and negative direction of omega_x and omega_y. Overall, the power spectra seems very symmetrical in all omega_x and omega_y direction with a slightly higher amplitude of 2 in the omega_x=0 and omega_y=0 direction forming a “plus” like figure. The real part of the fast fourier transform shows a high amplitude of approximately 2000 at omega_x=0 and omega_y=0 and approaches an amplitude of 0 in all positive and negative direction of omega_x and omega_y. There are no imaginary parts of the fast fourier transform of this even data.

Figure 2 (b) represents an odd 2D data with a high positive amplitude of 100 at x=0m and y=-250m, a high negative amplitude of -100 at x=0m and y=250m, and amplitude of 0 at x=0m and y=0m with the rest of the data goes to approximately 0 everywhere else. The 2D power spectra has a high amplitude of 8 right before and after omega_y=0 along the omega_x=0 axis, an amplitude of 0 along omega_y=0, an amplitude of 3 along  omega_x=0, and approaches 0 throughout everywhere else. There are no real parts of the fast fourier transform of this odd data. The imaginary part of the fast fourier transform shows a high amplitude of approximately 30,000 at omega_x=0 and omega_y=0 and approaches an amplitude of 0 in all positive and negative direction of omega_x and omega_y.

Figure 2 (c) represents an offsetted 2D data with a high amplitude of approximately 6 at x=-400m and y=0m. The power spectra has a high positive amplitude of 6 at omega_x=0 and omega_y=0, an amplitude of approximately 2 in between the omega_y=-0.1 and omega_y=0.1 axis, an amplitude of approximately 2 in between the omega_x=0 axis forming a “plus” like figure, and an amplitude of 0 everywhere else. Overall, the power spectra looks like the power spectra of the even function figure 2 (a) although with a thicker band of amplitude 2 along the omega_y axis. Unlike the previous even and odd function, there are both real and imaginary parts present in this fast fourier transform of the offsetted 2D data with the highest amplitude located at omega_x=0 and omega_y=0.

**3. Run ft2d on (sml.dat, lrg.dat) to study the difference in wavenumber content and band width in “anomalies” with different width.**




(a) sml.dat

(b) lrg.dat

**Figure 3:** Perform 2D FFT on (a) sml.dat and (b) lrg.dat  2D data profile.

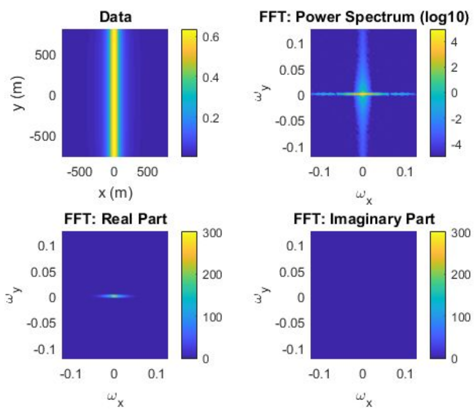
As seen in the figure 3 above, the 2D Fast Fourier Transform is performed to a small and large box car 2D function.

Figure 3 (a) represents a small box-car 2D function with a high amplitude of 4 from x=-250m to x=250m and y=-250m to y=250m. Overall this small box-car has a width/height of 500m. The power spectra of the fast fourier transform of this small box-car function is larger in area than that of the larger box-car function. The power spectra has the highest amplitude of 4 at omega_x=0 and omega_y=0 with a broader symmetrical band forming a “plus” pattern throughout the slightly higher amplitude through the omega_y=0 and omega_x=0 axis. The real part of the fast fourier transform of this small box-car function is also overall broader than that of the larger box-car function. The real part has a highest amplitude of 100 located at omega_x=0 and omega_y=0 and finally approaches to approximately 0 slower at omega_x=-0.05 and lower, omega_x=0.05 and higher, omega_y=-0.05 and lower, and omega_y=0.05 and higher. There are no imaginary parts present for this function.

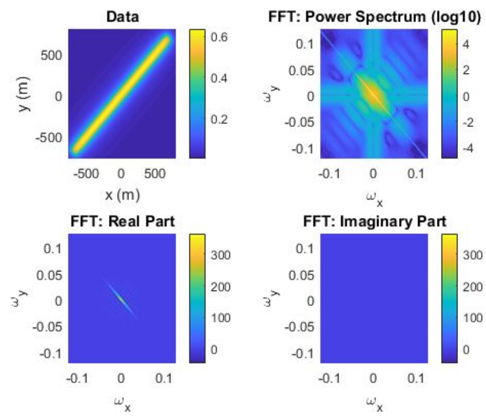
Figure 3 (b) represents a large box-car 2D function with a high amplitude of 4 from x=-500m to x=500m and y=-500m to y=500m. Overall this large box-car has a width/height of 1000m. The power spectra of the fast fourier transform of this large box-car function is overall narrower than that of the smaller box-car function. The power spectra has the same highest amplitude of 4 at omega_x=0 and omega_y=0 but with a narrower symmetrical bad forming a “plus” pattern throughout with a higher amplitude through the omega_y=0 and omega_x=0 axis. The real part of the fast fourier transform of this large box-car function is also overall narrower than that of the smaller box-car function. The real part has the same high amplitude of 400 located at omega_x=0 and omega_y=0 and approaches to approximately 0 faster at omega_x=-0.025 and lower, omega_x=0.025 and higher, omega_y=-0.025 and lower, and omega_y=0.025 and higher. There are no imaginary parts present for this function.

Overall, there is a relationship between the width of the box-car function and the “width” of its fourier transform in terms of the wavenumber band with significant power. As the width of the box-car function gets bigger, the power spectra has the same amplitude but a narrower band of the high amplitude power spectra. On the other hand, with a narrower box-car function, the power spectra presents the same amplitude but significantly wider band.

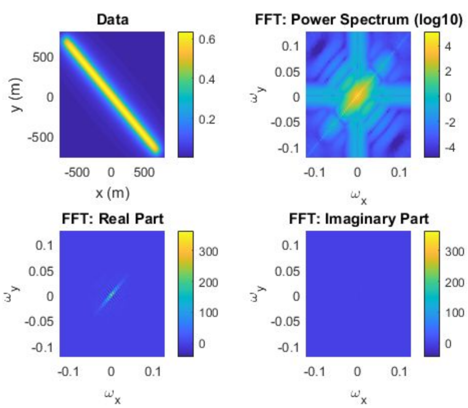
**4. Run ft2d on (nn.dat, ne.dat, nw.dat, ee.dat) to explore the Fourier spectrum of elongated feature.**



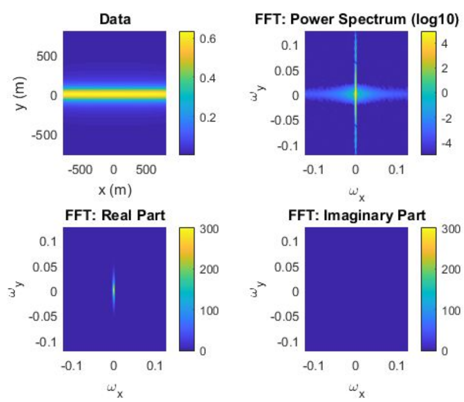

(a) nn.dat




(b) ne.dat




c) nw.dat




(d) ee.dat

**Figure 4:** Perform 2D FFT on (a) nn.dat, (b) ne.dat, (c) nw.dat, and (d) ee.dat 2D data profile.

As seen in the figure 4 above, the 2D Fast Fourier Transform is performed to a an elongated body 2D function located at different directions in the 2D plane.

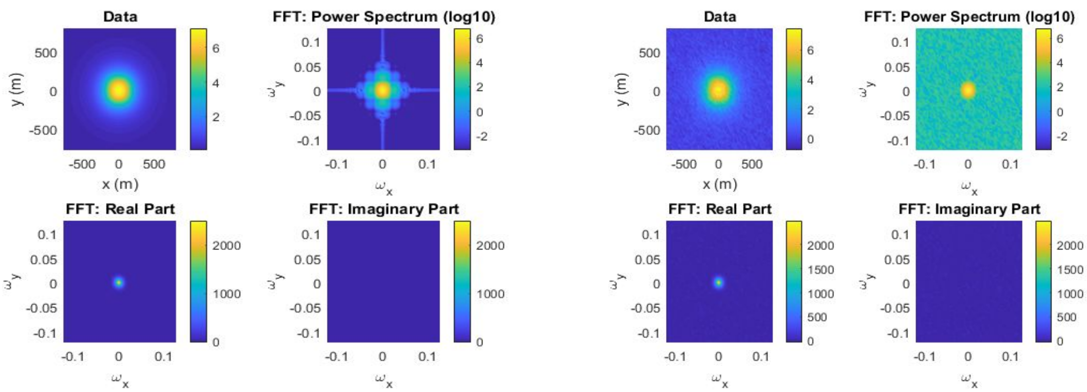
Figure 4 (a) represents a fast fourier transform of an elongated body with a high amplitude of 0.8 located along the x=0m axis in a northward direction. The power spectra of this function has the highest amplitude of 4 in omega_x=0 and omega_y=0, and  an overall “plus” symmetrical pattern with a narrower higher amplitude along the omega_y=0 axis and a broader higher amplitude along the omega_x=0 axis. The real part shows a small narrow band of high amplitude along the omega_y=0 axis. There are no imaginary parts.

Figure 4 (b) represents a fast fourier transform of an elongated body with a high amplitude of 0.8 located diagonally in a positive x and positive y or in a northeastern direction. The power spectra shows an overall “plus” pattern of high amplitudes along the omega_x=0 and omega_y=0 axis, an overall linear diagonal patterns of high amplitudes along the positive omega_y and negative omega_x direction, and the highest amplitude of 4 located in omega_x=0 and omega_y=0. The real part shows a small narrow band of high amplitude along the diagonal positive omega_y and negative omega_x direction. There are no imaginary parts.

Figure 4 (c) represents a fast fourier transform of an elongated body with a high amplitude of 0.8 located diagonally in a positive y and negative x or in a northwestern direction. The power spectra shows an overall “plus” pattern of high amplitudes along the omega_x=0 and omega_y=0 axis, an overall linear diagonal patterns of high amplitudes along the positive omega_y and positive omega_x direction, and the highest amplitude of 4 located in omega_x=0 and omega_y=0. The real part shows a small narrow band of high amplitude along the diagonal positive omega_y and positive omega_x direction. There are no imaginary parts.

Figure 4 (d) represents a fast fourier transform of an elongated body with a high amplitude of 0.8 located along the y=0m axis in an easterly direction. The power spectra of this function has the highest amplitude of 4 in omega_x=0 and omega_y=0, and an overall “plus” symmetrical pattern with a narrower higher amplitude along the omega_x=0 axis and a broader higher amplitude along the omega_y=0 axis. The real part shows a small narrow band of high amplitude along the omega_x=0 axis. There are no imaginary parts.

**5. Run ft2d on (even.dat, noisy.dat) to examine the power spectra of accurate and noisy data**




(a) even.dat

(b) noisy.dat

**Figure 5:** Perform 2D FFT on (a) even.dat and (b) noisy.dat 2D data profile.

As seen in the figure 5 above, the 2D Fast Fourier Transform is performed to a clear even 2D function and a noisy even 2D function. There are several differences in the corresponding power spectra, real, and imaginary components in the Fourier domain for the two 2D functions.

Figure 5 (a) represents a very clear and well-defined even 2D data with a high amplitude of approximately 6 at x=0 m and y=0 m while the rest of the data goes to approximately 0 throughout positive and negative x and y. The 2D fast fourier transform power spectra shows a high amplitude of 6 at omega_x=0 m and omega_y=0 m forming a “plus” symmetrical structure, and an amplitude approaching to zero in all positive and negative direction of omega_x and omega_y. Overall, the power spectra seems very clear and symmetrical in all omega_x and omega_y direction with a slightly higher amplitude of 2 in the omega_x=0 and omega_y=0 direction. The real part of the fast fourier transform shows a high amplitude of approximately 2000 at omega_x=0 and omega_y=0 and approaches an amplitude of 0 in all positive and negative direction of omega_x and omega_y. There are no imaginary parts of the fast fourier transform of this even data. This function is the same as in figure 2(a).

On the other hand, figure 5 (b) represents a noisy data of the same even 2D data in figure 5(a). The power spectra for the noisy even data is very different from the original clear even 2D data. The power spectra of the fast fourier transform of the noisy even 2D data has a high amplitude of 6 at omega_x=0 and omega_y=0 and a general positive amplitude of approximately 2 with specs of negative amplitudes throughout. There are also no general “plus” symmetrical pattern that is seen in the power spectra of the fast fourier transform of the clear even 2D function. Additionally, the real and imaginary parts of the fast fourier transform of the noisy even 2D function is very similar with the fast fourier transform of the clear 2D functions. There is a high amplitude of about 2000 at omega_x=0 and omega_y=0 in the real part, and a general amplitude 0 throughout omega_x and omega_y. The difference is that there are specs of different levels of amplitudes throughout the real and imaginary parts of the fast fourier transform of the noisy even 2D function. The presence of these differences of amplitudes represented by the specs represent the noise in the overall even 2D functions.

**Given Matlab Scripts:**

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function ft1d(datf)
%
% function ft1d(datf)
%
% Perform 1D FFT on a 1D data profile and display
% resulting quantities for understanding the
% Fourier transform of different data types.
%
% Input:  datf: name of the data file (string variable)
%
% Requires the input data file in the following format:
%
%-----------------------------------------------------
%
%N
%x(1),       y(1)
%x(2),       y(1)
%..
%x(N),       z(N)
%-------------------------------------------------------

if nargin <1,
   datf=input('  Name of data file: ');
end;

%===================================
% Step-0: Input the data
%===================================
fid = fopen(datf,'rt');
nx = fscanf(fid,'%d',1);
for i=1:nx,
   x(i) = fscanf(fid,'%f',1);
   y(i) = fscanf(fid,'%E',1);
end
fclose(fid);

%===================================
% Step-2: Expand the grid to the nearest
%         power of 2
%===================================
%n2x = pow2(ceil(log2(nx)));
%n2y = pow2(ceil(log2(ny)));
n2x=nx;
%
%
% calculate angular frequencies:
% Note: the frequencies are ordered from 0 to kx (Nyquist),
%       then from -kx+1 to -1
%       dxo and dyo are the frequency intervals
%
% Nyquist indices
kx = n2x/2;

% frequencies
dx=x(2)-x(1);
dxo=2*pi/(dx*n2x);
for ii=-kx+1:kx,
   omegax(ii+kx)=ii*dxo;
end;
%
%==============================
% Step-2: apply FFT
%==============================
tmp=foldfft(y);
zft=fft(tmp);
zft=unfoldfft(zft);
%
amplitude=abs(zft);
real_part=real(zft);
imag_part=imag(zft);
phase=angle(zft);

%===============================
% plot the data:
%================================
figure(1);
subplot(2,2,1),
plot(x, y);
axis tight;

subplot(2,2,2),
semilogy(omegax,amplitude.^2);
axis tight;

subplot(2,2,3),
plot(omegax, real_part);
axis([omegax(1) omegax(nx) min(min(real_part),min(imag_part))
max(amplitude) ]);


subplot(2,2,4),
plot(omegax, imag_part);
axis([omegax(1) omegax(nx) min(min(real_part),min(imag_part))
max(amplitude) ]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function ft2d(datf)
%
% function ft2d(datf)
%
% Perform 2D FFT on a 2D data map and display
% resulting quantities for understanding the
% Fourier transform of different data types.
%
% Input:  datf: name of the data file (string variable)
%
% Requires the input data file in the following format:
%
%-----------------------------------------------------
%
%N_east, N_north
%y(1), x(1),       z(1,1)
%y(1), x(2),       z(1,2)
%..
%y(1), x(N_north), z(1, N_north)
%y(2), x(1),       z(2,1)
%...
%y(N_east), x(N_north), z(N_east, N_north)
%-------------------------------------------------------
%
%
%-------------------------------------------------------

if nargin <1,
   datf=input('  Name of data file: ');
end;

%===================================
% Step-0: Input the data
%===================================
fid = fopen(datf,'rt');
ny = fscanf(fid,'%d',1);
nx = fscanf(fid,'%d',1);
for j=1:ny,
           for i=1:nx,
                      y(i,j) = fscanf(fid,'%f',1);
                      x(i,j) = fscanf(fid,'%f',1);
     z(i,j) = fscanf(fid,'%E',1);
   end
end
fclose(fid);

%===================================
% Step-2: Expand the grid to the nearest
%         power of 2
%===================================
%n2x = pow2(ceil(log2(nx)));
%n2y = pow2(ceil(log2(ny)));
n2x=nx;
n2y=ny;
%
%
% Padding with zero:
```

```
% Create a zero matrix of the appropriate dimensions
% and fill the upper-left portion with the input data
%
%lx1=fix((n2x-nx)/2);
%lx2=n2x-nx-lx1;
%ly1=fix((n2y-ny)/2);
%ly2=n2y-ny-ly1;

%zexpand=zeros(n2x,n2y);
%zexpand(lx1+1:lx1+nx,ly1+1:ly1+ny)=z;
%
% calculate angular frequencies:
% Note: the frequencies are ordered from 0 to kx (Nyquist),
%       then from -kx+1 to -1
%       dxo and dyo are the frequency intervals
%
% Nyquist indices
kx = n2x/2;
ky = n2y/2;

% frequencies
dx=x(2,2)-x(1,1);
dy=y(2,2)-y(1,1);
dxo=2*pi/(dx*n2x);
dyo=2*pi/(dy*n2y);
for ii=-kx+1:kx,
   omegax(ii+kx)=ii*dxo;
end;
for ii=-ky+1:ky,
   omegay(ii+ky)=ii*dyo;
end;
[Omegay,Omegax]=meshgrid(omegay,omegax);
%
%==============================
% Step-2: apply FFT
%==============================
tmp=foldfft(z);
zft=fft2(tmp);
zft=unfoldfft(zft);
%
amplitude=abs(zft);
real_part=real(zft);
imag_part=imag(zft);
phase=angle(zft);

%===============================
% plot the data:
%================================
xmin=min(min(x));
xmax=max(max(x));
ymin=min(min(y));
ymax=max(max(y));

pmax=max(omegax);
pmin=min(omegax);
qmax=max(omegay);
qmin=min(omegay);

tmax=max(max(amplitude));
tminr=min(min(real_part));
tmini=min(min(real_part));
tmin=min(tminr,tmini);

figure(1);
subplot(2,2,1),
pcolor(y, x, z);
xlabel('x (m)');
ylabel('y (m)');
title('Data');
colorbar;
shading interp;
axis tight;

subplot(2,2,2),
pcolor(Omegay, Omegax, log10(amplitude.^2));
xlabel('\omega_x');
ylabel('\omega_y');
title('FFT: Power Spectrum (log10)');
amax=2*log10(tmax);
amin=amax-10.0;
caxis([amin amax]);
colorbar;
shading interp;
axis tight;

subplot(2,2,3),
pcolor(Omegay, Omegax, real_part);
xlabel('\omega_x');
ylabel('\omega_y');
title('FFT: Real Part');
shading interp;
axis tight;
caxis([tmin, tmax]);
colorbar;

subplot(2,2,4),
pcolor(Omegay, Omegax, imag_part);
xlabel('\omega_x');
ylabel('\omega_y');
title('FFT: Imaginary Part');
shading interp;
axis tight;
caxis([tmin, tmax]);
colorbar;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function aout=foldfft(ain)
%
% function aout=fold(ain)
%
% fold Fourier transform to shift
% the DC component to lower-left corner
%
nd = ndims(ain);
idx = cell(1, nd);
for k = 1:nd
    nx = size(ain, k);
    kx = floor(nx/2);
    if kx>1
       idx{k} = [kx:nx 1:kx-1];
    else
       idx{k} = [1];
    end
end
aout = ain(idx{:});

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function prism(inpf)
%
% function prismg(inpf)
%
% function to calculate the gravity field due to one or more
% cuboidal bodies over a regular grid at a given height
%
% Require input file 'prism.inp' in the following format
%------------------------------------------------------
%grav.dat                  <------- File name to output the
anomaly
%Emin Emax De              <------- min, max, interval in
easting
%Nmin Nmax Dn              <------- min, max, interval in
northing
%height1, heights          <------- observation height
%nprism                    <------- number of prisms
%Estrt Eend Nstrt Nend Ztop Zbot rho    <---dimension and density
%...
%Estrt.........
%-------------------------------------------------------
% Requires two user functions:
%
% (1)  grav=gden(x1,x2,y1,y2,z1,z2,rho)
% (2)  corner=fnode(x, y, z, xs, ys, zs)
%
%-------------------------------------------------------
% Author: Yaoguo Li
% Date:   11/10/99
```

```
% Written for the GPGN303 lab exercise #7
%
%--------------------------------------------------------
if nargin <1,
   inpf=input('Input file name: ');
end;
fi=fopen(inpf,'rt');

outf=fgetl(fi);
ymin=fscanf(fi,'%f',1);
ymax=fscanf(fi,'%f',1);
dely=fscanf(fi,'%f',1);
xmin=fscanf(fi,'%f',1);
xmax=fscanf(fi,'%f',1);
delx=fscanf(fi,'%f',1);
height1=fscanf(fi,'%f',1);
height2=fscanf(fi,'%f',1);

% set up the x-location
xp=xmin:delx:xmax;
[ndum,nx]=size(xp);
yp=ymin:dely:ymax;
[ndum,ny]=size(yp);
for i=1:nx,
for j=1:ny,
   x(i,j)=xmin+(i-1)*delx;
   y(i,j)=ymin+(j-1)*dely;
end;
end;

grav1=0.0*x;
grav2=0.0*x;

nprism=fscanf(fi,'%d',1);
for iprism=1:nprism,
   yb=fscanf(fi,'%f',1);
   ye=fscanf(fi,'%f',1);
   xb=fscanf(fi,'%f',1);
   xe=fscanf(fi,'%f',1);
   zb=fscanf(fi,'%f',1);
   ze=fscanf(fi,'%f',1);
   rho=fscanf(fi,'%f',1);

   x1=xb-x;
   x2=xe-x;
   y1=yb-y;
   y2=ye-y;
   z1=0*x+height1+zb;
   z2=0*x+height1+ze;
   grav1=grav1+gden(x1,x2,y1,y2,z1,z2,rho);

   z1=0*x+height2+zb;
   z2=0*x+height2+ze;
   grav2=grav2+gden(x1,x2,y1,y2,z1,z2,rho);
end
fclose(fi);

% write out the data
fo=fopen(outf,'wt');
fprintf(fo,'%6d %6d\n',ny,nx);
fprintf(fo,'\n');
fprintf(fo,'\n');
fprintf(fo,'\n');
for j=1:ny,
   for i=1:nx,
      fprintf(fo,'%8.2f %8.2f
%8.4f\n',y(i,j),x(i,j),grav1(i,j));
   end;
end;
fclose(fo);

% plot the data
zmin=min(min(grav1));
zmax=max(max(grav1));
figure(1);
subplot(2, 2, 1);
%pcolor(yp,xp,grav1);
%colorbar;
%contour(yp,xp,grav,11);
mesh(yp,xp,grav1);
axis([ymin ymax xmin xmax zmin zmax]);
%axis square;
colormap('jet');
title('Data At Height-I');
xlabel('Easting (m)');
ylabel('Northing (m)');
zlabel('Anomaly (mGal)');
subplot(2,2,2);
%pcolor(yp,xp,grav2);
%colorbar;
%contour(yp,xp,grav,11);
mesh(yp,xp,grav2);
axis([ymin ymax xmin xmax zmin zmax]);
%axis square;
colormap('jet');
title('Data At Height-II');
xlabel('Easting (m)');
ylabel('Northing (m)');
zlabel('Anomaly (mGal)');
%
subplot(2, 2, 3);
pcolor(yp,xp,grav1);
shading interp;
colorbar;
axis([ymin ymax xmin xmax]);
axis square;
colormap('jet');
title('Data At Height-I');
xlabel('Easting (m)');
ylabel('Northing (m)');
zlabel('Anomaly (mGal)');
subplot(2,2,4);
pcolor(yp,xp,grav2);
shading interp;
colorbar;
axis([ymin ymax xmin xmax]);
axis square;
colormap('jet');
title('Data At Height-II');
xlabel('Easting (m)');
ylabel('Northing (m)');
zlabel('Anomaly (mGal)');
%end of prism

function grav=gden(x1,x2,y1,y2,z1,z2,rho)
%
% calculate the vertical gravity field due to a uniform
% cuboidal prism
%
gcons=6.672E-03;
%
x1s=x1.*x1;
x2s=x2.*x2;
y1s=y1.*y1;
y2s=y2.*y2;
z1s=z1.*z1;
z2s=z2.*z2;

grav=-(- fnode(x1, y1, z1, x1s, y1s, z1s) ...
       + fnode(x2, y1, z1, x2s, y1s, z1s) ...
       + fnode(x1, y2, z1, x1s, y2s, z1s) ...
       - fnode(x2, y2, z1, x2s, y2s, z1s) ...
       + fnode(x1, y1, z2, x1s, y1s, z2s) ...
       - fnode(x2, y1, z2, x2s, y1s, z2s) ...
       - fnode(x1, y2, z2, x1s, y2s, z2s) ...
       + fnode(x2, y2, z2, x2s, y2s, z2s) )*gcons*rho;

%end of gden

function corner=fnode(x, y, z, xs, ys, zs)

rt=sqrt(xs + ys + zs);
top=x + y + rt;
corner= x.*log(y + rt) + y.*log(x + rt) + 2.0D0*z.*atan2(top, z);

% end of fnode
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function mprism(inpf)
%
% function prismg(inpf)
%
% function to calculate the gravity field due to one or more
% cuboidal bodies over a regular grid at a given height
%
% Require input file 'prism.inp' in the following format
%-----------------------------------------------------
%mag.dat                  <------- File name to output the
anomaly
%inc, dec, geomag         <------- inducing field parameter
%ainc,adec                <------- anomaly projection direction
%Emin Emax De             <------- min, max, interval in
easting
%Nmin Nmax Dn             <------- min, max, interval in
northing
%height                   <------- observation height
%nprism                   <------- number of prisms
%Estrt Eend Nstrt Nend Ztop Zbot sus    <---dimension and density
%...
%Estrt.........
%------------------------------------------------------
% Requires one user function:
% mag=gsus(x1,x2,y1,y2,z1,z2,sus)
%------------------------------------------------------
% Author: Yaoguo Li
% Date:   14/11/99
% Written for the GPGN303 lab exercise #11
%------------------------------------------------------
if nargin <1,
   inpf=input('Input file name: ');
end;
fi=fopen(inpf,'rt');

%input the output file name
outf=fgetl(fi);

%inducing field direction and strength
dincl=fscanf(fi,'%f',1);
ddecl=fscanf(fi,'%f',1);
geomag=fscanf(fi,'%f',1);

%anomaly projection direction
aincl=fscanf(fi,'%f',1);
adecl=fscanf(fi,'%f',1);

%observation grid and height
ymin=fscanf(fi,'%f',1);
ymax=fscanf(fi,'%f',1);
dely=fscanf(fi,'%f',1);
xmin=fscanf(fi,'%f',1);
xmax=fscanf(fi,'%f',1);
delx=fscanf(fi,'%f',1);
height=fscanf(fi,'%f',1);

% set up the grid
xp=xmin:delx:xmax;
yp=ymin:dely:ymax;
[y,x]=meshgrid(yp,xp);
z=0.0*x-height;
mag=0.0*x;

nprism=fscanf(fi,'%d',1);
for iprism=1:nprism,
   yb=fscanf(fi,'%f',1);
   ye=fscanf(fi,'%f',1);
   xb=fscanf(fi,'%f',1);
   xe=fscanf(fi,'%f',1);
   zb=fscanf(fi,'%f',1);
   ze=fscanf(fi,'%f',1);
   sus=fscanf(fi,'%f',1);

   mag=mag+gsus(xb,xe,yb,ye,zb,ze,x,y,z,sus,...
                dincl,ddecl,geomag,aincl,adecl);
end
fclose(fi);

% write out the data
[nx,ny]=size(x);
fo=fopen(outf,'wt');
fprintf(fo,'%6d %6d\n',ny,nx);
fprintf(fo,'\n');
fprintf(fo,'\n');
fprintf(fo,'\n');
for j=1:ny,
   for i=1:nx,
      fprintf(fo,'%8.2f %8.2f  %8.4f\n',y(i,j),x(i,j),mag(i,j));
   end;
end;
fclose(fo);

% plot the data
zmin=min(min(mag));
zmax=max(max(mag));
figure(2);
subplot(2, 2, 1);
mesh(yp,xp,mag);
axis([ymin ymax xmin xmax zmin zmax]);
colormap('jet');
title('Magnetic Anomaly');
xlabel('Easting (m)');
ylabel('Northing (m)');
zlabel('Anomaly (nT)');
subplot(2,2,2);
pcolor(yp,xp,mag);
shading interp;
colorbar;
axis([ymin ymax xmin xmax]);
colormap('jet');
title('Magnetic Anomaly');
xlabel('Easting (m)');
ylabel('Northing (m)');
%end of mprism


function mag=gsus(xb,xe,yb,ye,zb,ze,x,y,z,...
                  sus,dincl,ddecl,geomag,aincl,adecl)
%
% function mag=gsus(xb,xe,yb,ye,zb,ze,sus,x,y,z,...
%                  dinc,ddec,geomag,ainc,adec)
%
% computes the magnetic field due to a cuboidal prism
% with a constant susceptibility
%
%----------------------------------------------------

% calculate the direction cosine of the geomagnetic field
di=pi*dincl/180.0;
dd=pi*ddecl/180.0;

cx=cos(di)*cos(dd);
cy=cos(di)*sin(dd);
cz=sin(di);

% calculate the direction cosines of the direction at
% which to project the anomaly

ai=pi*aincl/180.0;
ad=pi*adecl/180.0;

cxp=cos(ai)*cos(ad);
cyp=cos(ai)*sin(ad);
czp=sin(ai);

gxy= cx*cyp + cy*cxp;
gxz= cx*czp + cz*cxp;
gyz= cy*czp + cz*cyp;

gxx= cx*cxp;
gyy= cy*cyp;
gzz= cz*czp;

fact=geomag*sus/(4.0*pi);
```

```
% begin the calculation
a1=xb-x;
a2=xe-x;
b1=yb-y;
b2=ye-y;
h1=zb-z;
h2=ze-z;

r111=sqrt(a1.^2 + b1.^2 + h1.^2);
r112=sqrt(a1.^2 + b1.^2 + h2.^2);

r211=sqrt(a2.^2 + b1.^2 + h1.^2);
r212=sqrt(a2.^2 + b1.^2 + h2.^2);

r121=sqrt(a1.^2 + b2.^2 + h1.^2);
r122=sqrt(a1.^2 + b2.^2 + h2.^2);

r221=sqrt(a2.^2 + b2.^2 + h1.^2);
r222=sqrt(a2.^2 + b2.^2 + h2.^2);

%----1/(DyDz) term
top1=(a2 + r222);
bot1=(a1 + r122);

top2=(a1 + r121);
bot2=(a2 + r221);

top3=(a1 + r112);
bot3=(a2 + r212);

top4=(a2 + r211);
bot4=(a1 + r111);

top=top1.*top2.*top3.*top4;
bot=bot1.*bot2.*bot3.*bot4;

tyz=log(top./bot);

%----1/(DxDz) term
top1=(b2 + r222);
bot1=(b1 + r212);

top2=(b1 + r211);
bot2=(b2 + r221);

top3=(b1 + r112);
bot3=(b2 + r122);

top4=(b2 + r121);
bot4=(b1 + r111);

top=top1.*top2.*top3.*top4;
bot=bot1.*bot2.*bot3.*bot4;

txz=log(top./bot);

%----1/(DxDy) term
top1=(h2 + r222);
bot1=(h1 + r221);

top2=(h1 + r211);
bot2=(h2 + r212);

top3=(h1 + r121);
bot3=(h2 + r122);

top4=(h2 + r112);
bot4=(h1 + r111);

top=top1.*top2.*top3.*top4;
bot=bot1.*bot2.*bot3.*bot4;

txy=log(top./bot);

tol=0.0*a1 + 1.0E-10;

%----1/(DxDx) term
del=sign(a1).*max(abs(a1),tol) + (1-abs(sign(a1))).*tol;
txx=    - atan(b1.*h2./(r112.*del)) + atan(b1.*h1./(r111.*del));
txx=txx + atan(b2.*h2./(r122.*del)) - atan(b2.*h1./(r121.*del));

del=sign(a2).*max(abs(a2),tol) + (1-abs(sign(a2))).*tol;
txx=txx + atan(b1.*h2./(r212.*del)) - atan(b1.*h1./(r211.*del));
txx=txx - atan(b2.*h2./(r222.*del)) + atan(b2.*h1./(r221.*del));

%----1/(DyDy) term
del=sign(b1).*max(abs(b1),tol) + (1-abs(sign(b1))).*tol;
tyy=    - atan(a1.*h2./(r112.*del)) + atan(a1.*h1./(r111.*del));
tyy=tyy + atan(a2.*h2./(r212.*del)) - atan(a2.*h1./(r211.*del));

del=sign(b2).*max(abs(b2),tol) + (1-abs(sign(b2))).*tol;
tyy=tyy + atan(a1.*h2./(r122.*del)) - atan(a1.*h1./(r121.*del));
tyy=tyy - atan(a2.*h2./(r222.*del)) + atan(a2.*h1./(r221.*del));

%---1/(DzDz) term
tzz= - (txx + tyy);

wk=gxx*txx + gyy*tyy + gzz*tzz +  gxy*txy + gxz*txz + gyz*tyz;

mag=fact*wk;
% end of function gsus
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function prism(inpf)
%
% function prismg(inpf)
%
% function to calculate the gravity field due to one or more
% cuboidal bodies over a regular grid at a given height
%
% Require input file 'prism.inp' in the following format
%-------------------------------------------------------
%grav.dat                   <------- File name to output the
anomaly
%Emin Emax De               <------- min, max, interval in
easting
%Nmin Nmax Dn               <------- min, max, interval in
northing
%theta
%height1, heights           <------- observation height
%nprism                     <------- number of prisms
%Estrt Eend Nstrt Nend Ztop Zbot rho    <---dimension and density
%...
%Estrt.........
%--------------------------------------------------------
% Requires two user functions:
%
% (1)  grav=gden(x1,x2,y1,y2,z1,z2,rho)
% (2)  corner=fnode(x, y, z, xs, ys, zs)
%
%--------------------------------------------------------
% Author: Yaoguo Li
% Date:   11/10/99
% Written for the GPGN303 lab exercise #7
%
%--------------------------------------------------------
if nargin <1,
   inpf=input('Input file name: ');
end;
fi=fopen(inpf,'rt');

outf=fgetl(fi);
ymin=fscanf(fi,'%f',1);
ymax=fscanf(fi,'%f',1);
dely=fscanf(fi,'%f',1);
xmin=fscanf(fi,'%f',1);
xmax=fscanf(fi,'%f',1);
delx=fscanf(fi,'%f',1);
height1=fscanf(fi,'%f',1);
height2=fscanf(fi,'%f',1);

% set up the x-location
xp=xmin:delx:xmax;
[ndum,nx]=size(xp);
yp=ymin:dely:ymax;
[ndum,ny]=size(yp);
for i=1:nx,
```

```
for j=1:ny,
   xin(i,j)=xmin+(i-1)*delx;
   yin(i,j)=ymin+(j-1)*dely;
end;
end;

grav1=0.0*xin;
grav2=0.0*xin;
nprism=fscanf(fi,'%d',1);
for iprism=1:nprism,
   yb=fscanf(fi,'%f',1);
   ye=fscanf(fi,'%f',1);
   xb=fscanf(fi,'%f',1);
   xe=fscanf(fi,'%f',1);
   zb=fscanf(fi,'%f',1);
   ze=fscanf(fi,'%f',1);
   rho=fscanf(fi,'%f',1);
   theta=fscanf(fi,'%f',1);

   [th,r]=cart2pol(xin,yin);
   th=th+theta*pi/180.0;
   [x,y]=pol2cart(th,r);
   x1=xb-x;
   x2=xe-x;
   y1=yb-y;
   y2=ye-y;
   z1=0*x+height1+zb;
   z2=0*x+height1+ze;
   grav1=grav1+gden(x1,x2,y1,y2,z1,z2,rho);

   z1=0*x+height2+zb;
   z2=0*x+height2+ze;
   grav2=grav2+gden(x1,x2,y1,y2,z1,z2,rho);
end
fclose(fi);
x=xin;
y=yin;
% write out the data
grav1=grav1+randn(size(grav1))*0.2;
fo=fopen(outf,'wt');
fprintf(fo,'%6d %6d\n',ny,nx);
fprintf(fo,'\n');
fprintf(fo,'\n');
fprintf(fo,'\n');
for j=1:ny,
   for i=1:nx,
      fprintf(fo,'%8.2f %8.2f
%8.4f\n',y(i,j),x(i,j),grav1(i,j));
   end;
end;
fclose(fo);

% plot the data
zmin=min(min(grav1));
zmax=max(max(grav1));
figure(1);
subplot(2, 2, 1);
%pcolor(yp,xp,grav1);
%colorbar;
%contour(yp,xp,grav,11);
mesh(yp,xp,grav1);
axis([ymin ymax xmin xmax zmin zmax]);
%axis square;
colormap('jet');
title('Data At Height-I');
xlabel('Easting (m)');
ylabel('Northing (m)');
zlabel('Anomaly (mGal)');
subplot(2,2,2);
%pcolor(yp,xp,grav2);
%colorbar;
%contour(yp,xp,grav,11);
mesh(yp,xp,grav2);
axis([ymin ymax xmin xmax zmin zmax]);
%axis square;
colormap('jet');
title('Data At Height-II');
xlabel('Easting (m)');
ylabel('Northing (m)');
zlabel('Anomaly (mGal)');
%
subplot(2, 2, 3);
pcolor(yp,xp,grav1);
shading interp;
colorbar;
axis([ymin ymax xmin xmax]);
axis square;
colormap('jet');
title('Data At Height-I');
xlabel('Easting (m)');
ylabel('Northing (m)');
zlabel('Anomaly (mGal)');
subplot(2,2,4);
pcolor(yp,xp,grav2);
shading interp;
colorbar;
axis([ymin ymax xmin xmax]);
axis square;
colormap('jet');
title('Data At Height-II');
xlabel('Easting (m)');
ylabel('Northing (m)');
zlabel('Anomaly (mGal)');
%end of prism

function grav=gden(x1,x2,y1,y2,z1,z2,rho)
%
% calculate the vertical gravity field due to a uniform
% cuboidal prism
%
gcons=6.672E-03;
x1s=x1.*x1;
x2s=x2.*x2;
y1s=y1.*y1;
y2s=y2.*y2;
z1s=z1.*z1;
z2s=z2.*z2;
grav=-(- fnode(x1, y1, z1, x1s, y1s, z1s) ...
       + fnode(x2, y1, z1, x2s, y1s, z1s) ...
       + fnode(x1, y2, z1, x1s, y2s, z1s) ...
       - fnode(x2, y2, z1, x2s, y2s, z1s) ...
       + fnode(x1, y1, z2, x1s, y1s, z2s) ...
       - fnode(x2, y1, z2, x2s, y1s, z2s) ...
       - fnode(x1, y2, z2, x1s, y2s, z2s) ...
       + fnode(x2, y2, z2, x2s, y2s, z2s) )*gcons*rho;

%end of gden

function corner=fnode(x, y, z, xs, ys, zs)

rt=sqrt(xs + ys + zs);
top=x + y + rt;
corner= x.*log(y + rt) + y.*log(x + rt) + 2.0D0*z.*atan2(top, z);
% end of fnode

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function aout=unfoldfft(ain)
%
% function aout=fold(ain)
%
% unfold Fourier transform to shift
% the DC component to centre
%
nd = ndims(ain);
idx = cell(1, nd);
for k = 1:nd
    nx = size(ain, k);
    kx = floor(nx/2);
    if kx>1
       idx{k}=[kx+2:nx 1:kx+1];
    else
       idx{k}=[1];
    end
end
aout = ain(idx{:});
```