In [2]:
```
import numpy as np
import math
from matplotlib import gridspec
%pylab inline
```

```
Populating the interactive namespace from numpy and matplotlib
```

GPGN 411/511: Advanced Gravity and Magnetic Exploration

LAB 07- Wavenumber Domain Processing: Analytic Signal in 2D Magnetic Data Analysis

Nadima Dwihusna and Felicia Nurindrawati

Due: 11/14/2017

A. Lab Objective and Main Concepts Covered

The main objective of this lab is to explore the properties of the envelope in 2D potential-field data analyses using the magnetic forward modeling code from Lab 03 and the wavenumber-domain processing codes from Lab 06. The main relevant concepts that are covered in the lab are the following:

1. 2D approximation using elongated source bodies.
2. Amplitude of analytic signal and envelope.
3. Magnetic amplitude data.

In class, the analytic signal associated with a profile of magnetic data produced by a 2D source body is discussed. The horizontal and vertical derivatives of the total-field anomaly produced by a 2D source from a Hilbert transform pair, and therefore thay can be used to define an analytic signal of

$$A(x) = \frac{\partial \Delta T}{\partial (x)} + i\frac{\partial \Delta T}{\partial z}$$

where ΔT is the total-field anomaly. This function follows that the amplitude of the analytic signal

$$a(x) = \sqrt{\left(\frac{\partial \Delta T}{\partial (x)}\right)^2 + \left(\frac{\partial \Delta T}{\partial z}\right)^2}$$

is an envelope and is independent of the magnetization direction.

Overall, this lab 07 of wavenumber domain processing analytic signal in 2D magnetic data analysis is designed to:

1. Test the property above by approximating 2D data sets by calculating them using an elongated source body and use the derivative code that have beend developed previously.
2. Explore the similar quantities such as the magnetic amplitude data

$$|B_a| = \sqrt{B_{ax}^2 + B_{az}^2}$$

B. Program Description

Overall, the program below develops a code that is designed to test the property above by approximating the 2D data sets by calculating them using an elongated source body and using the derivative code that was developed in the previous lab. Additionally, the code below was used to explore the similar quantities such as the magnetic amplitude data.

For task 1, the code from Lab 03 was used to compute the total-field magnetic anomaly from a source body elongated in the east-west direction. The parameters and location of the source body along with the gridding parameters are given in this task. Additionally, the inducing field inclination and declination was also given for the three scenerios of Im=90 and Dm=0, Im=45 and Dm=0, and Im=-45 and Dm=0. Using this task 1 code, the total-field anomaly from these three scenarios are computed and plotted in a contour format. In this case, the function for computing and plotting these three scenerios were defined as "task1" function. For each of the scenarios, the different values of Im and Dm pairs are put into the function calling the defined "task1" function. As a result, the contour format of these three scenarios are plotted with minimal change in the east-west direction.

For task 2, the code from Lab 06 was used and modified to add additional derivatice calculations to compute the first vertical derivative and x-derivative (northing direction) of each maps. In this case, the derivatices were used to generate the quantity in the second equation above. The calculation in this task yields to a good approximation to the envelope over the approximate 2D body. In this code, the folding and unfolding functions for 1D and 2D were defined and called throughout. In this case, the total-field anomaly and envelopes along the central north-south profile in the three different magnetization direction cases are ran through separately in order to be plotted and compared to one another in the end.

For task 3, the code from Lab 03 is modified to output the three components directly giving the different Inclination and Declination values for anomaly projection direction that corresponds to verrtical, easting, and northing respectively. The same instruction applies in task 3 as from the previous task 1 and task 2, defining each components to its respective Im and Dm. In the end, the three components are plotted and the changes of the magnetization direction are examined. Additionally, the x-component (northing) and z-component (vertical) are also taken to calculate the magnitude amplitude data for the approximated 2D source.

For task 3 alternative bonus task, using the forward modelling code, the vertical component Baz is computed for each case of the three magnetization directions. Afterwards, the code allows the user to use the component conversion code from Lab 06 to obtain Bax and Bay through the wavenumber-domain operation. In the end, the results of the Bax and Bay componentes throug hthe wavenumber-domain operation are plotted.

C. Task I

Utilize the code from Lab 03, and compute the total-field magnetic anomaly froma source body elongated in east-west direction. Use a source body that is buried at a depth of 100 m andcentered below (0,0) m in easting and northing, hasa width of 300 m in north-south direction, depth extent of 50 m, a length of 10,000 m in the east-west direction, and a magnetization magnitude of 2 A/m. Further assume that the observation grid is the same as that in Lab 03:

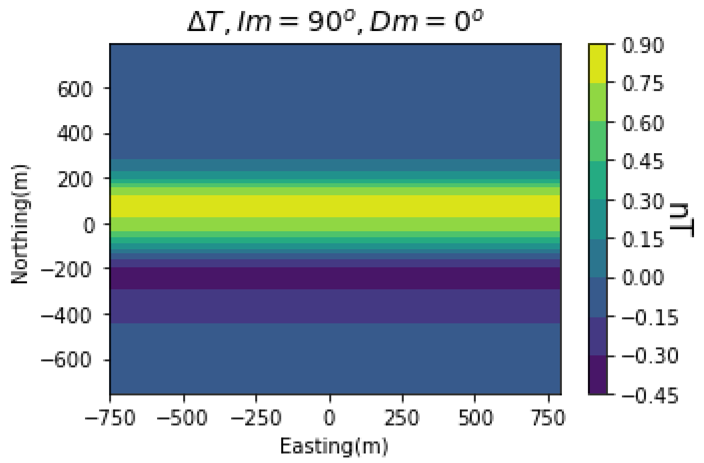
1. Northing:(-775, 800) at 25-m spacing
2. Easting:(-775, 800) at 25-m spacing
3. Elevation of thedata is a constant 0.0

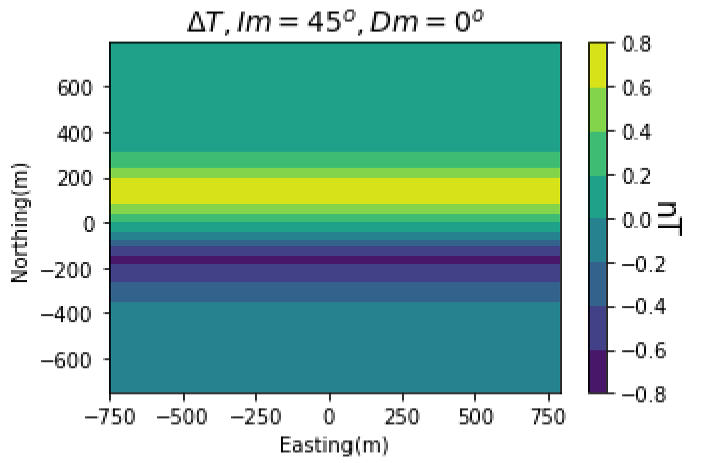
Assuming that the inducing field is in the direction of I=60 deg, D=0 deg, and the magnetization directionhas three different scenarios:

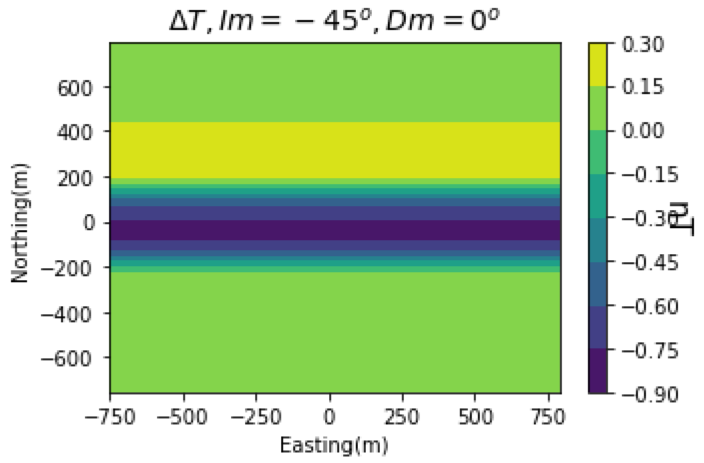
1. Im=90deg, Dm=deg
2. Im=45deg, Dm=0deg
3. Im=-45deg, Dm=0deg

Compute the total-field anomaly from these three scenarios. When plotted in contour format, you should see a minimal change in the east-west direction.

In [5]:
```
Ys1 = -5000
Ys2 = 5000
Xs1 = -150
Xs2 = 150
Zs1 = 100
Zs2 = 150
Xinp = np.arange(-755,800,25)
Yinp = np.arange(-755,800,25)
Zinp = 0

Txx = np.zeros((len(Xinp), len(Yinp)))
Tyy = np.zeros((len(Xinp), len(Yinp)))
Tzz = np.zeros((len(Xinp), len(Yinp)))
Txy = np.zeros((len(Xinp), len(Yinp)))
Tyz = np.zeros((len(Xinp), len(Yinp)))
Txz = np.zeros((len(Xinp), len(Yinp)))
matrixOut = np.zeros((len(Xinp)**2, 9))
count = 0

def task1(Imi, Dmi):
###Scenario-1
    Im = Imi
    Dm = Dmi
    Mm = 2
###### Input Bo information
    I = 60
    D = 0
    muo = 4*np.pi*10**(-7)

    BoX = np.cos(I*np.pi/180)*np.cos(D*np.pi/180)
    BoY = np.cos(I*np.pi/180)*np.sin(D*np.pi/180)
    BoZ = np.sin(I*np.pi/180)
    Bo = [BoX, BoY, BoZ]
    Mx = Mm*np.cos(Im*np.pi/180)*np.cos(Dm*np.pi/180)
    My = Mm*np.cos(Im*np.pi/180)*np.sin(Dm*np.pi/180)
    Mz = Mm*np.sin(Im*np.pi/180)
    M = [Mx, My, Mz]


    for i in range(len(Xinp)):
        for j in range(len(Yinp)):
            Txx[i,j] = math.atan2((Yinp[j]-Ys2)*(Zinp-Zs2),((Xinp[i]-Xs2)*math.sq
            Tyy[i,j] = math.atan2((Xinp[i]-Xs2)*(Zinp-Zs2),((Yinp[j]-Ys2)*math.sq
            Tzz[i,j] = math.atan2((Xinp[i]-Xs2)*(Yinp[j]-Ys2),((Zinp-Zs2)*math.sq
            Txy[i,j] = -math.log(math.sqrt((Xinp[i]-Xs2)**2+(Yinp[j]-Ys2)**2+(Zin
            Txz[i,j] = -math.log(math.sqrt((Xinp[i]-Xs2)**2+(Zinp-Zs2)**2+(Yinp[j
            Tyz[i,j] = -math.log(math.sqrt((Yinp[j]-Ys2)**2+(Zinp-Zs2)**2+(Xinp[i

    Bax = np.zeros((len(Xinp), len(Yinp)))
    Bay = np.zeros((len(Xinp), len(Yinp)))
    Baz = np.zeros((len(Xinp), len(Yinp)))
    deltaT = np.zeros((len(Xinp), len(Yinp)))


    for i in range(len(Xinp)):
        for j in range(len(Yinp)):
            Bax[i,j] = (muo/(np.pi*4))*((Txx[i,j]*Mx) + (Txy[i,j]*My)+(Txz[i,j]*M
            Bay[i,j] = (muo/(np.pi*4))*((Txy[i,j]*Mx) + (Tyy[i,j]*My)+(Tyz[i,j]*M
            Baz[i,j] = (muo/(np.pi*4))*((Txz[i,j]*Mx) + (Tyz[i,j]*My)+(Tzz[i,j]*M
            deltaT[i,j] = ((BoX*Bax[i,j] + BoY*Bay[i,j] + BoZ*Baz[i,j]))*50000*10
    return [Bax,Bay,Baz,deltaT]

matrix = task1(90,0)
Bax = matrix[0]
Bay = matrix[1]
Baz = matrix[2]
deltaT = matrix[3]
gs = gridspec.GridSpec(3, 1)
fig = plt.figure(figsize=(5, 10))
[x,y] = np.meshgrid(Xinp,Yinp)

def plotting(deltaT):


    ax3 = plt.contourf(x, y, deltaT, colormap='hot')
    ax3 = plt.xlabel('Easting(m)')
    ax3 = plt.ylabel('Northing(m)')
    cbar = plt.colorbar()
    cbar.set_label('nT',size = 15, rotation=270)
    return()

plt.subplot(311)
plotting(deltaT)
ax3 = plt.title('$\Delta T, Im=90^{o}, Dm=0^{o}$', size=14)


##scenario2:
matrixb = task1(45,0)
Baxb = matrixb[0]
Bayb = matrixb[1]
Bazb = matrixb[2]
deltaTb = matrixb[3]
plt.subplot(312)
plotting(deltaTb)
ax3 = plt.title('$\Delta T, Im=45^{o}, Dm=0^{o}$', size=14)


##scenario3:
matrixb = task1(-45,0)
Baxc = matrixb[0]
Bayc = matrixb[1]
Bazc = matrixb[2]
deltaTc = matrixb[3]
plt.subplot(313)
plotting(deltaTc)
ax3 = plt.title('$\Delta T, Im=-45^{o}, Dm=0^{o}$', size=14)



plt.tight_layout()
plt.show()
```







Task I - Observation and Conclusion

The total-field magnetic anomaly from a source body elongated in east-west direction were computed. The source body parameters were given with a buried depth of 100 m and centered below (0,0) m in easting and northing with a width of 300 m in the north-south direction, depth extent of 50 m and a length of 10,000 m in the east-wets direction, and a magnetization magnitude of 2 A/m. The three scenerios for the different Inclination and Declination of the magnetization were also given. Using the Task 1 code above, the total-field anomaly from the three scenarios are plotted above. Observing the three scenarios above, there is a higher amplitude approximately near the middle of northing = 0 m and a declining towards a lower constant amplitude throughout the sides. There are very minimal or small changes in the east-west direction. Overall, throughout the three scenarios, the changes of amplitudes appears to be present only in the north-south direction as what is expected from computing and plotting the total-field anomaly of the three given scenarios.

D. Task II

Use your code from Lab 06 (and modify to add additional derivative calculations if needed) to compute the first vertical derivative and x-derivative (northing direction) of each map. Use the derivatives to generate the quantityin equation 2 above. The north-south profile at easting=0 m from each of the calculation yields a good approximation to the envelope over the approximate 2D body.

Please plot and compare the total-field anomalies as well as the envelopes along the central north-south profile in these three cases with different magnetization directions. Please observe the difference among the total-field anomaly and the similarity among the envelopes, and explain you observations.

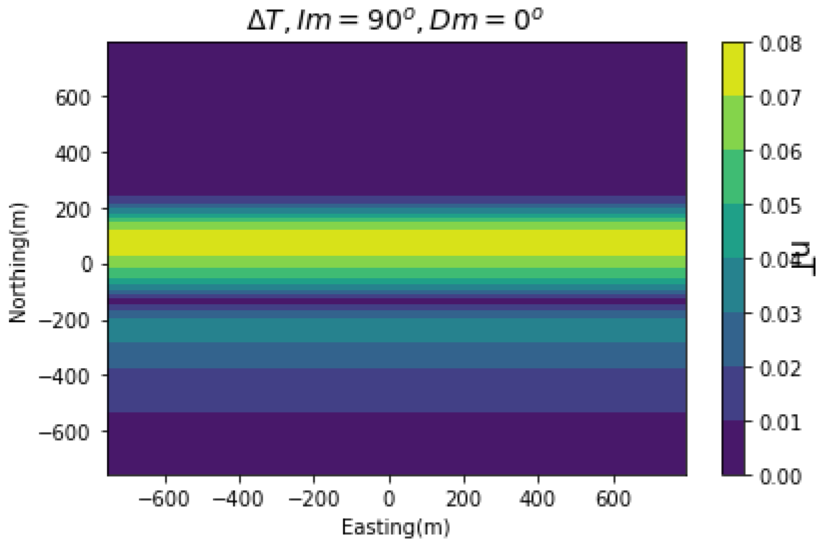
Note: We are focusing on the central profile, but it would also be instructive to examine the maps.

In [8]:
```
####### FUNCTION DEFINITIONS
def foldonce(A,n):
    Ashift = np.zeros(n)
    shift = 1
    for i in range(n):
        temp = A[0]
        for j in range(n):
            if j == n-1:
                Ashift[n-1] = temp
            else:
                Ashift[j] = A[j+shift]
    return Ashift

def foldonce2D(A, n):
    Ashift = np.zeros((n,n),dtype=np.complex_)
    Ashiftshift = np.zeros((n,n),dtype=np.complex_)
    shift = 1
    for i in range(n):
        temp = A[i,0]

        for j in range(n):
            if j == n-1:
                Ashift[i,n-1] = temp
            else:
                Ashift[i,j] = A[i, j+shift]

    for i in range(n):
        temprow = Ashift[0,:]
        if i == n-1:
            Ashiftshift[n-1,:] = temprow
        else:
            Ashiftshift[i,:] = Ashift[i+shift,:]
    return Ashiftshift

def unfold2Donce(A, n):
    Ashift = np.zeros((n,n),dtype=np.complex)
    Ashiftshift = np.zeros((n,n),dtype=np.complex)
    shift = 1
    for i in range(n):
        temp = A[i,n-1]
        for j in range(n):
            if j == 0:
                Ashift[i,0] = temp
            else:
                Ashift[i,j] = A[i, j-shift]

    for i in range(n):
        temprow = Ashift[n-1,:]

        if i == 0:
            Ashiftshift[0,:] = temprow
        else:
            Ashiftshift[i,:] = Ashift[i-shift,:]

    return Ashiftshift
```

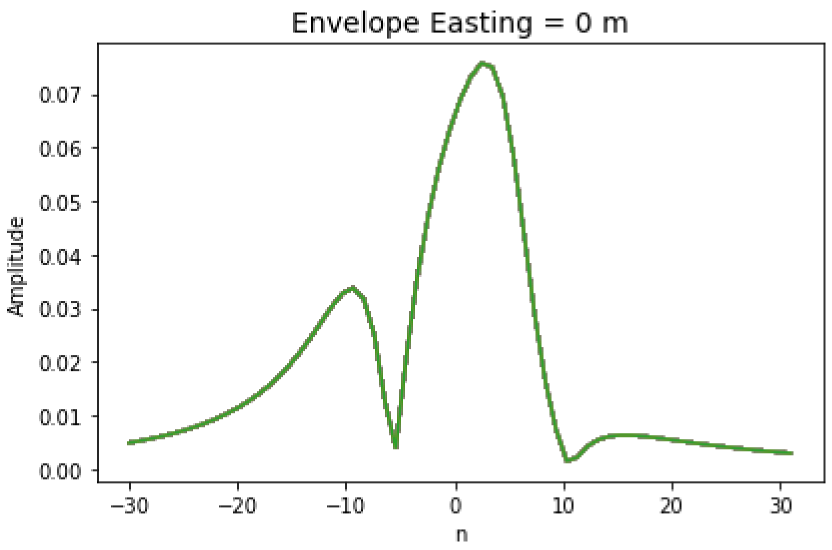
```
######## Unfolding Functions #######################################################

def unfold(s,C3):
    for i in range(int(s-1)):
        ufftR3 = unfold2Donce(C3, n)
        C3 = ufftR3
    return ufftR3


###### Find wx and wy

M = len(Xinp)
N = len(Yinp)
wx = np.zeros((N,N))
wy = np.zeros((M,M))

kx = int(M/2)
ky = int(N/2)

dx = abs(Xinp[0]-Xinp[1])
dy = abs(Yinp[0]-Yinp[1])

m = np.linspace(-kx+1, kx, (M))
n = np.linspace(-ky+1, ky, (N))

wxm = m*2*np.pi/(M*dx)
for i in range(M):
    wx[i,:] = wxm


wym = n*2*np.pi/(N*dy)
for j in range(N):
    wy[j,:] = wym

for i in range(len(wx)):
    for j in range(len(wx)):
        if wx[i,j] == 0:
            wx[i,j] = 0.000000001;
        if wy[i,j] == 0:
            wy[i,j] = 0.00000001
```

In [54]:
```
### Fold
s = (M/2) + 1 - 1
num = len(n)
A1 = np.zeros((N,N),dtype=np.complex)
A1 = deltaT
for i in range(int(s-1)):
    fold1 = foldonce2D(A1, num)
    A1 = fold1

### FFT
fft1 = np.fft.fft2(fold1)



### operations
dtdz = np.zeros((num,num),dtype=np.complex)
dtdz = np.zeros((num,num),dtype=np.complex)
dtdz = np.sqrt(1j*wx**2 + 1j*wy**2)*fft1
dtdx = 1j*wx*fft1

##### Should do out sise
#a = np.sqrt(dtdz**2+dtdx**2)



### Ifft both
dtdzi = np.fft.ifft2(dtdz)
dtdxi = np.fft.ifft2(dtdx)

### unfold both
A1 = dtdzi
for i in range(int(s-1)):
    dtdz = unfold2Donce(A1, num)
    A1 = dtdz

A1 = dtdxi
for i in range(int(s-1)):
    dtdx = unfold2Donce(A1, num)
    A1 = dtdx

#### calculate a
a = np.sqrt(dtdz**2+dtdx**2)
plotting(a)
plt.title('$\Delta T, Im=90^{o}, Dm=0^{o}$', size=14)
plt.tight_layout()
plt.show()



count = 0
env1 = np.zeros(N,dtype=np.complex)
for i in range(len(m)):
    for j in range(len(n)):
        if n[i]==0:
            env1[count] = a[i,j]
            count+=1
plt.plot(n,a[0:])
plt.title('Envelope Easting = 0 m', size=14)
plt.xlabel('n')
plt.ylabel('Amplitude')
plt.tight_layout()
plt.show()
```

```
C:\Anaconda3\lib\site-packages\numpy\ma\core.py:2766: ComplexWarning: Casting c
omplex values to real discards the imaginary part
  order=order, subok=True, ndmin=ndmin)
```

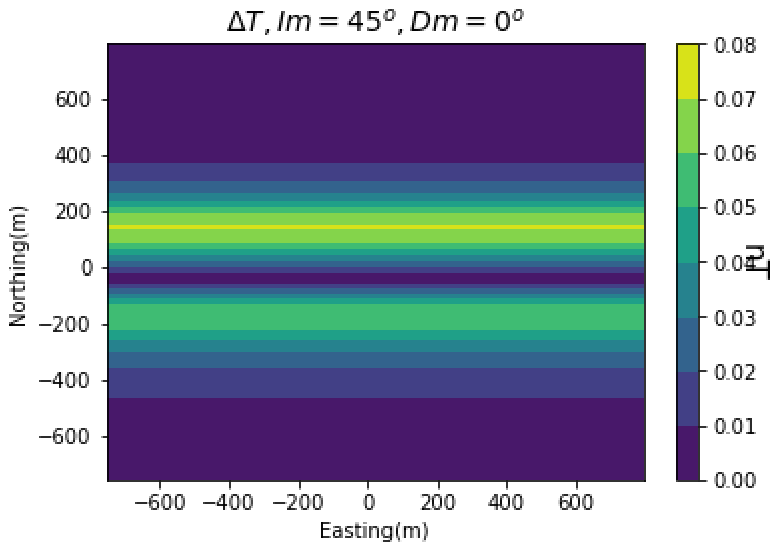


```
C:\Anaconda3\lib\site-packages\numpy\core\numeric.py:531: ComplexWarning: Casti
ng complex values to real discards the imaginary part
  return array(a, dtype, copy=False, order=order)
```

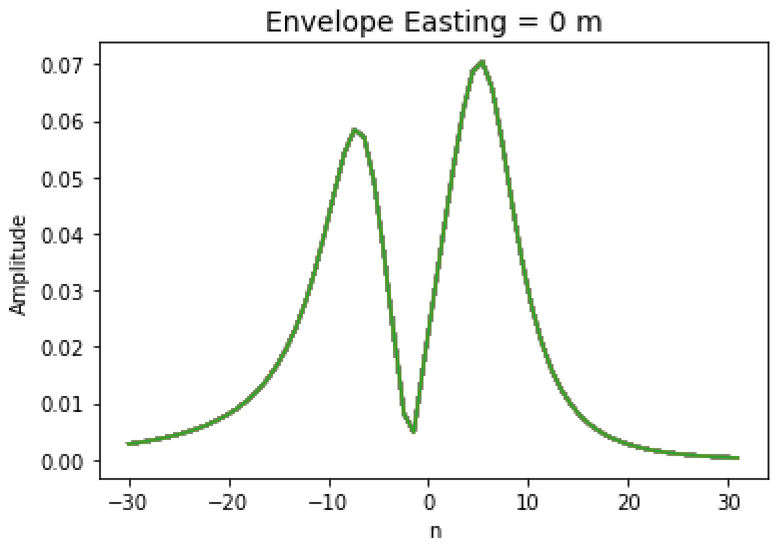


In [55]:
```
### Fold
s = (M/2) + 1 - 1
num = len(n)
A1 = np.zeros((N,N),dtype=np.complex)
A1 = deltaTb
for i in range(int(s-1)):
    fold1 = foldonce2D(A1, num)
    A1 = fold1

### FFT
fft1 = np.fft.fft2(fold1)



### operations
dtdz = np.zeros((num,num),dtype=np.complex)
dtdz = np.zeros((num,num),dtype=np.complex)
dtdz = np.sqrt(1j*wx**2 + 1j*wy**2)*fft1
dtdx = -1j*wx*fft1

##### Should do out sise
#a = np.sqrt(dtdz**2+dtdx**2)



### Ifft both
dtdzi = np.fft.ifft2(dtdz)
dtdxi = np.fft.ifft2(dtdx)
### unfold both
A1 = dtdzi
for i in range(int(s-1)):
    dtdz = unfold2Donce(A1, num)
    A1 = dtdz

A1 = dtdxi
for i in range(int(s-1)):
    dtdx = unfold2Donce(A1, num)
    A1 = dtdx

#### calculate a
a = np.sqrt(dtdz**2+dtdx**2)
plotting(a)
plt.title('$\Delta T, Im=45^{o}, Dm=0^{o}$', size=14)
plt.show()




count = 0
env1 = np.zeros(N,dtype=np.complex)
for i in range(len(m)):
    for j in range(len(n)):
        if n[i]==0:
            env1[count] = a[i,j]
            count+=1
plt.plot(n,a[0:])
plt.title('Envelope Easting = 0 m', size=14)
plt.xlabel('n')
plt.ylabel('Amplitude')
plt.show()
```

```
C:\Anaconda3\lib\site-packages\numpy\ma\core.py:2766: ComplexWarning: Casting c
omplex values to real discards the imaginary part
  order=order, subok=True, ndmin=ndmin)
```

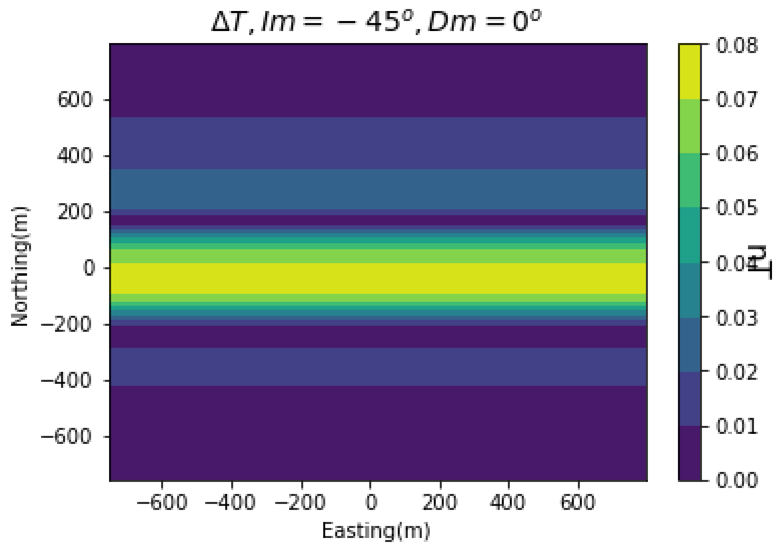


```
C:\Anaconda3\lib\site-packages\numpy\core\numeric.py:531: ComplexWarning: Casti
ng complex values to real discards the imaginary part
  return array(a, dtype, copy=False, order=order)
```

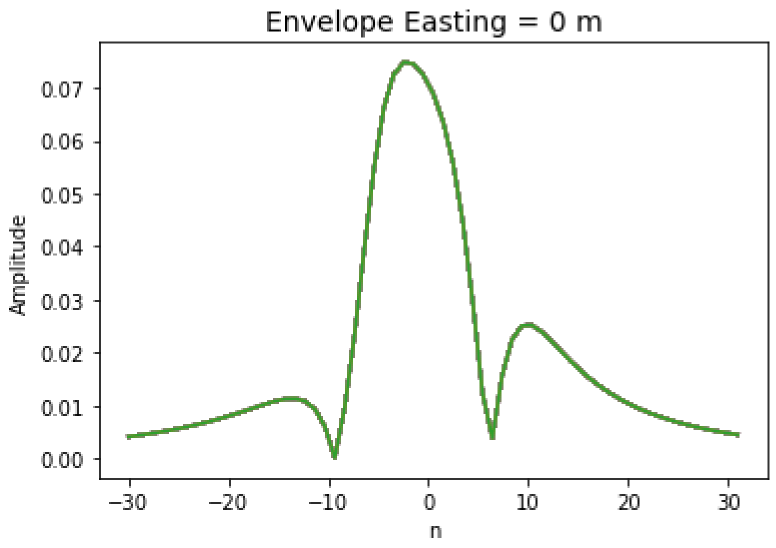


In [56]:
```
### Fold
s = (M/2) + 1 - 1
num = len(n)
A1 = np.zeros((N,N),dtype=np.complex)
A1 = deltaTc
for i in range(int(s-1)):
    fold1 = foldonce2D(A1, num)
    A1 = fold1

### FFT
fft1 = np.fft.fft2(fold1)



### operations
dtdz = np.zeros((num,num),dtype=np.complex)
dtdz = np.zeros((num,num),dtype=np.complex)
dtdz = np.sqrt(1j*wx**2 + 1j*wy**2)*fft1
dtdx = -1j*wx*fft1

##### Should do out sise
#a = np.sqrt(dtdz**2+dtdx**2)



### Ifft both
dtdzi = np.fft.ifft2(dtdz)
dtdxi = np.fft.ifft2(dtdx)
### unfold both
A1 = dtdzi
for i in range(int(s-1)):
    dtdz = unfold2Donce(A1, num)
    A1 = dtdz

A1 = dtdxi
for i in range(int(s-1)):
    dtdx = unfold2Donce(A1, num)
    A1 = dtdx

#### calculate a
a = np.sqrt(dtdz**2+dtdx**2)
plotting(a)
plt.title('$\Delta T, Im=-45^{o}, Dm=0^{o}$', size=14)
plt.show()




count = 0
env1 = np.zeros(N,dtype=np.complex)
for i in range(len(m)):
    for j in range(len(n)):
        if n[i]==0:
            env1[count] = a[i,j]
            count+=1
plt.plot(n,a[0:])
plt.title('Envelope Easting = 0 m', size=14)
plt.xlabel('n')
plt.ylabel('Amplitude')
plt.show()
```

```
C:\Anaconda3\lib\site-packages\numpy\ma\core.py:2766: ComplexWarning: Casting c
omplex values to real discards the imaginary part
  order=order, subok=True, ndmin=ndmin)
```



```
C:\Anaconda3\lib\site-packages\numpy\core\numeric.py:531: ComplexWarning: Casti
ng complex values to real discards the imaginary part
  return array(a, dtype, copy=False, order=order)
```



Task II - Observation and Conclusion

In this taske 2, the total field anomalies as well as the envelopes along the central north-south profile in the three cases with different magnetization directions of Im and Dm are computed and plotted. There are very slight difference among the total field anomaly and envelopes at easting = 0m. As seen in the three plots above, the main differences between the three scenarios of different Im and Dm are the location of the each peaks when the envelope of easting=0 m are plotted. Although overall, all three scenarios above have a higher amplitude approximately near the middle of northing =0 m and a declining towards a lower constant amplitude throughout the sides. There are very minimal or small changes in the east-west direction. Overall, throughout the three scenarios, the changes of amplitudes appear to be present only in the north-south direction as what is expected from computing and plotting the three scenarios. Throughout the three scenarios, there are very minimal differences in between them as the only difference are the location of the main peaks throughout the envelope of easting = 0m.

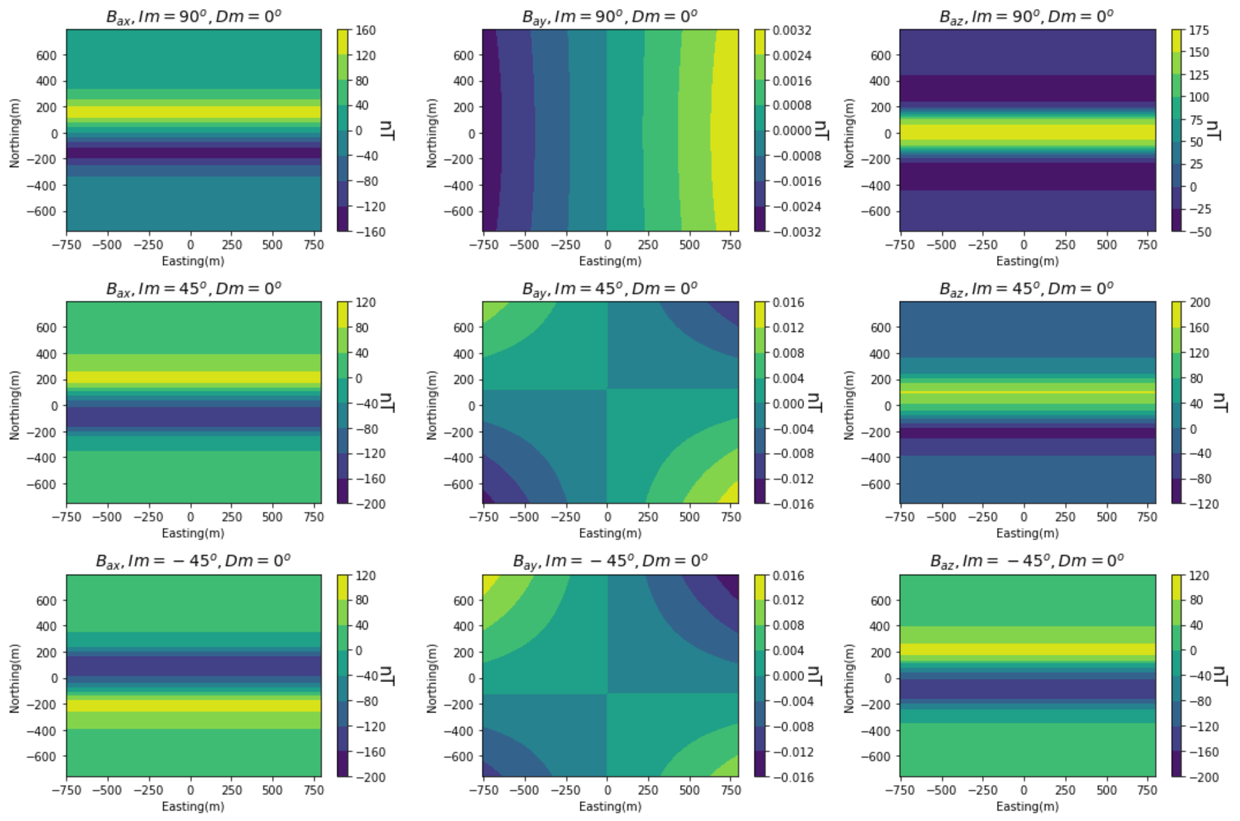
E. Task III

Take the same source body in Task-I and calculate the three components of the magnetic anomaly for each of the three different magnetization directions. For this, you can either modify the Lab#03code to output the three components directly or by “tricking” the code by giving it different I and D values for anomaly projection direction that corresponds to vertical, easting, and northing, respectively.

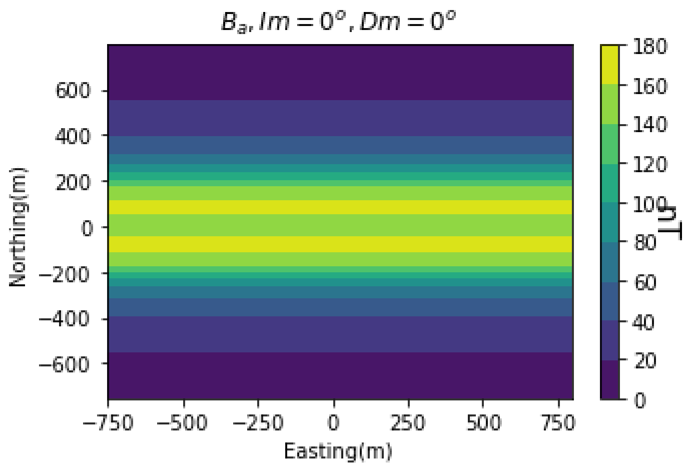
Plot the three components for your own reference and examine the changeswith the magnetization direction.

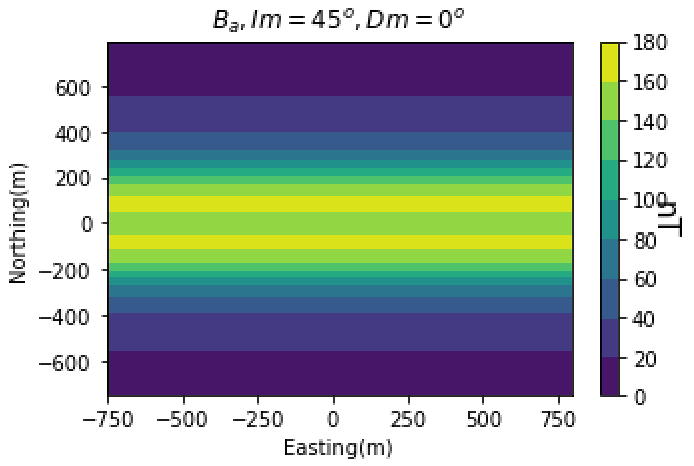
Then take the x-component (northing) and z-component (vertical) to calculate the magnetic amplitude data(equation 3) for the approximate 2D source. Compare the magnetic amplitude anomalies along the central profile at Easting=0 m. Do you observe a similar variation (or lack of it) with the change in the magnetization direction? What conclusion can you draw from the results?

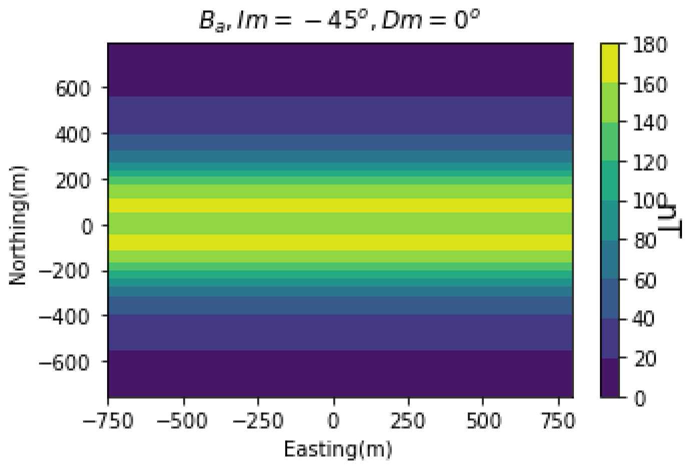
In [57]:
```
###1
plt.figure(figsize=(15,10))
gs = gridspec.GridSpec(3, 3)

#ax3 = fig.add_subplot(gs[0,0])
plt.subplot(331)
plotting(Bax)
ax3 = plt.title('$B_{ax}, Im=90^{o}, Dm=0^{o}$', size=14)


#ax3 = fig.add_subplot(gs[1,0])
plt.subplot(332)
plotting(Bay)
ax3 = plt.title('$B_{ay},  Im=90^{o}, Dm=0^{o}$', size=14)

#ax3 = fig.add_subplot(gs[2,0])
plt.subplot(333)
plotting(Baz)
ax3 = plt.title('$B_{az},  Im=90^{o}, Dm=0^{o}$', size=14)

#ax3 = fig.add_subplot(gs[0,1])
plt.subplot(334)
plotting(Baxb)
ax3 = plt.title('$B_{ax},  Im=45^{o}, Dm=0^{o}$', size=14)


#ax3 = fig.add_subplot(gs[1,1])
plt.subplot(335)
plotting(Bayb)
ax3 = plt.title('$B_{ay}, Im=45^{o}, Dm=0^{o}$', size=14)

#ax3 = fig.add_subplot(gs[2,1])
plt.subplot(336)
plotting(Bazb)
ax3 = plt.title('$B_{az}, Im=45^{o}, Dm=0^{o}$', size=14)


#ax3 = fig.add_subplot(gs[0,2])
plt.subplot(337)
plotting(Baxc)
ax3 = plt.title('$B_{ax}, Im=-45^{o}, Dm=0^{o}$', size=14)

plt.subplot(338)
#ax3 = fig.add_subplot(gs[1,2])
plotting(Bayc)
ax3 = plt.title('$B_{ay}, Im=-45^{o}, Dm=0^{o}$', size=14)

#ax3 = fig.add_subplot(gs[2,2])
plt.subplot(339)
plotting(Bazc)
ax3 = plt.title('$B_{az}, Im=-45^{o}, Dm=0^{o}$', size=14)

plt.tight_layout()
plt.show()
```


$B_{ax}, Im = 90^o, Dm = 0^o$
$B_{ay}, Im = 90^o, Dm = 0^o$
$B_{az}, Im = 90^o, Dm = 0^o$
$B_{ax}, Im = 45^o, Dm = 0^o$
$B_{ay}, Im = 45^o, Dm = 0^o$
$B_{az}, Im = 45^o, Dm = 0^o$
$B_{ax}, Im = -45^o, Dm = 0^o$
$B_{ay}, Im = -45^o, Dm = 0^o$
$B_{az}, Im = -45^o, Dm = 0^o$
Northing(m)
Easting(m)
nT

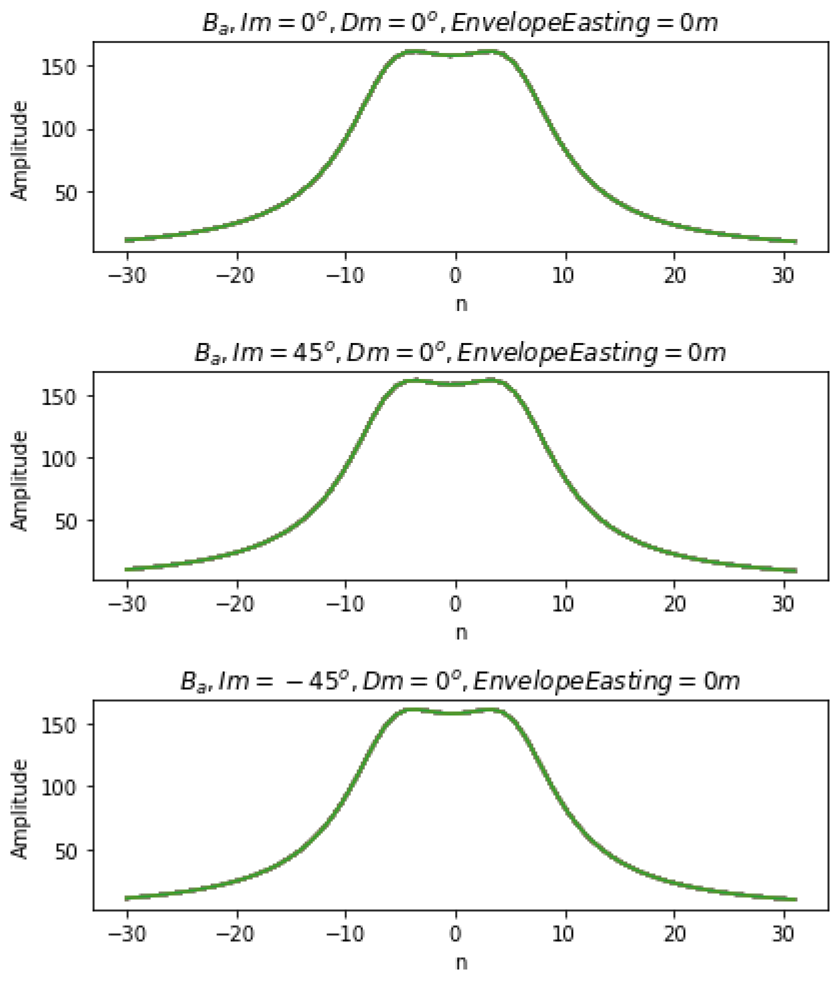
In [58]:
```
plt.figure(figsize=(5,10))
Ba1 = np.sqrt(Bax**2+Baz**2) ## 90 0
Ba2 = np.sqrt(Baxb**2+Bazb**2)## 45 0
Ba3 = np.sqrt(Baxc**2+Bazc**2) ## -45 0

plt.subplot(311)
plotting(Ba1)
plt.title("$B_a , Im=0^{o}, Dm=0^{o}$")

plt.subplot(312)
plotting(Ba2)
plt.title("$B_a , Im=45^{o}, Dm=0^{o}$")

plt.subplot(313)
plotting(Ba3)
plt.title("$B_a , Im=-45^{o}, Dm=0^{o}$")

plt.tight_layout()
plt.show()
```


$B_a, Im = 0^o, Dm = 0^o$
Northing(m)
Easting(m)
nT


$B_a, Im = 45^o, Dm = 0^o$
Northing(m)
Easting(m)
nT


$B_a, Im = -45^o, Dm = 0^o$
Northing(m)
Easting(m)
nT

```
In [59]: plt.figure(figsize=(6,7))
         plt.subplot(311)
         plt.plot(n,Ba1[0:])
         plt.title("$B_a , Im=0^{o}, Dm=0^{o} , Envelope Easting =0 m $")
         plt.xlabel('n')
         plt.ylabel('Amplitude')

         plt.subplot(312)
         plt.plot(n,Ba2[0:])
         plt.title("$B_a , Im=45^{o}, Dm=0^{o} , Envelope Easting =0 m$")
         plt.xlabel('n')
         plt.ylabel('Amplitude')

         plt.subplot(313)
         plt.plot(n,Ba3[0:])
         plt.title("$B_a , Im=-45^{o}, Dm=0^{o} , Envelope Easting =0 m$")
         plt.xlabel('n')
         plt.ylabel('Amplitude')

         plt.tight_layout()
         plt.show()
```



Task III - Observation and Conclusion

In this task 3, the source body in task 1 is taken and the three components of the magnetic anomlay for each of the three different magnetization directions are calculated. The three components are then plotted and the changes within the magnetization direction of Im and Dm are to be compared and examined. Additionally, the x-component (northing) and z-component (vertical) are taken to calculate the magnetix amplitude data for the approximated 2D source. Comparing the magnetic amplitude anomalies along the central profile at Easting =0m above, the amplitudes appear to be the same for all the different scenerios. Observing the above figures of the three components along with its respective envelope of easting = 0m, there is a high amount of similarity in between the changes of magnetization direction. As seen in the enveolope figures above, overall, all three scenarios above have a higher amplitude approximately near the middle of northing =0 m and a declining towards a lower constant amplitude throughout the sides. There are very minimal or small changes in the east-west direction. Additionally, throughout the three scenarios, the changes of amplitudes appear to be present only in the north-south direction as what is expected from computing and plotting the three scenarios. Throughout the three scenarios, there are very minimal differences in between them as the only difference are the location of the main peaks throughout the envelope of easting = 0m. From this obeservation, one can conlude that there are very little changes in the amplitude and overall structure of the envelope of easting = 0m with the changes in magnetization directions of Im and Dm.

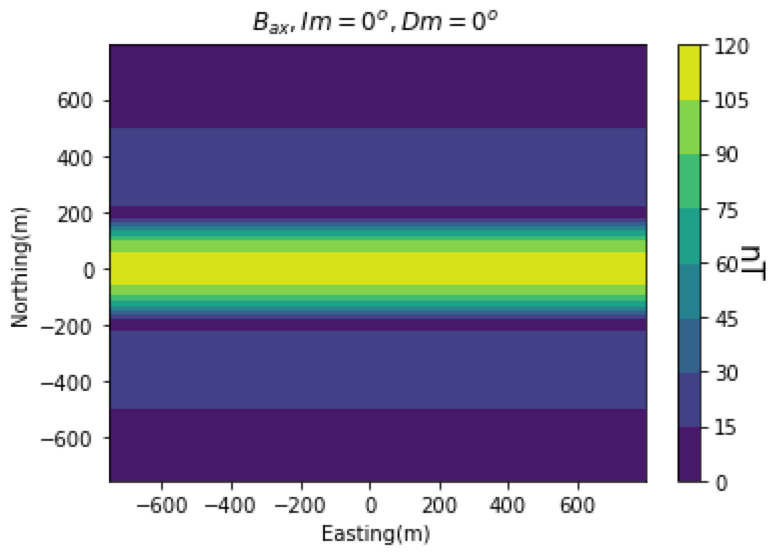
F. TASK III ALTERNATIVE (25% Bonus)

Instead of calculating the different components of the magnetic anomaly in Task-III above using the forward modeling code, you can first compute the vertical component Bazfor each case of the three magnetization directions, and then using the component conversion code from Lab#06to obtain Baxand Baythrough the wavenumber-domain operations!

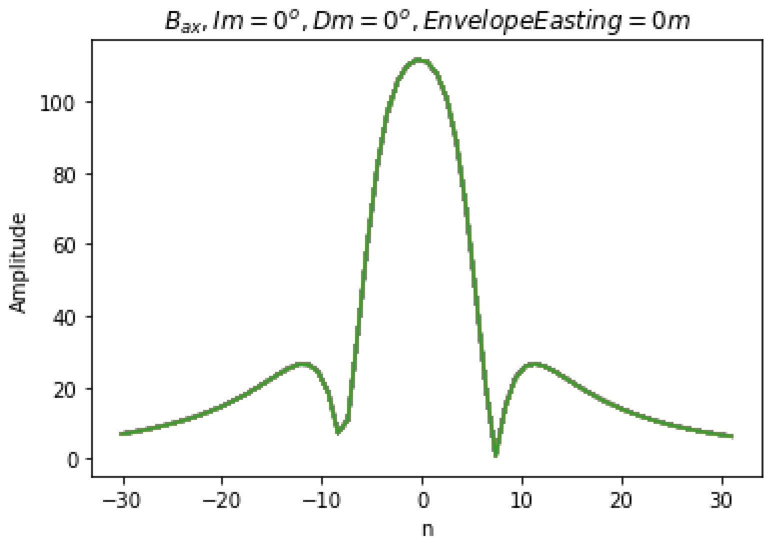
This may be an easier route for you, and any lab report submitted based on this routeinstead of Task-IIIwill receive 25% of bonus point! (Sounds paradoxical?!)

In [60]:
```
### Fold
s = (M/2) + 1 - 1
num = len(n)
A1 = np.zeros((N,N),dtype=np.complex)
A1 = Baz
for i in range(int(s-1)):
    fold1 = foldonce2D(A1, num)
    A1 = fold1

### FFT
fft1 = np.fft.fft2(fold1)



### operations
Bay1 = np.zeros((num,num),dtype=np.complex)
Bax1 = np.zeros((num,num),dtype=np.complex)
Bax1 = (1j*wx/np.sqrt(1j*wx**2 + 1j*wy**2))*fft1
Bay1 = (1j*wy/np.sqrt(1j*wy**2 + 1j*wx**2))*fft1

Ba = np.sqrt(Bax1**2+Bay1**2)

### Ifft
Bay1i = np.fft.ifft2(Bay1)
Bax1i = np.fft.ifft2(Bax1)

### unfold
A1 = Bay1i
for i in range(int(s-1)):
    Bay1 = unfold2Donce(A1, num)
    A1 = Bay1

A1 = Bax1i
for i in range(int(s-1)):
    Bax1 = unfold2Donce(A1, num)
    A1 = Bax1

##calculate Ba
Ba = np.sqrt(Bay1**2+Bax1**2)
plotting(Ba)
plt.title("$B_{ax} , Im=0^{o}, Dm=0^{o}$")
plt.show()
plt.plot(n,Ba[0:])
plt.title("$B_{ax} , Im=0^{o}, Dm=0^{o} , Envelope Easting =0 m$")
plt.xlabel('n')
plt.ylabel('Amplitude')
plt.show()
```

```
C:\Anaconda3\lib\site-packages\numpy\ma\core.py:2766: ComplexWarning: Casting c
omplex values to real discards the imaginary part
  order=order, subok=True, ndmin=ndmin)
```

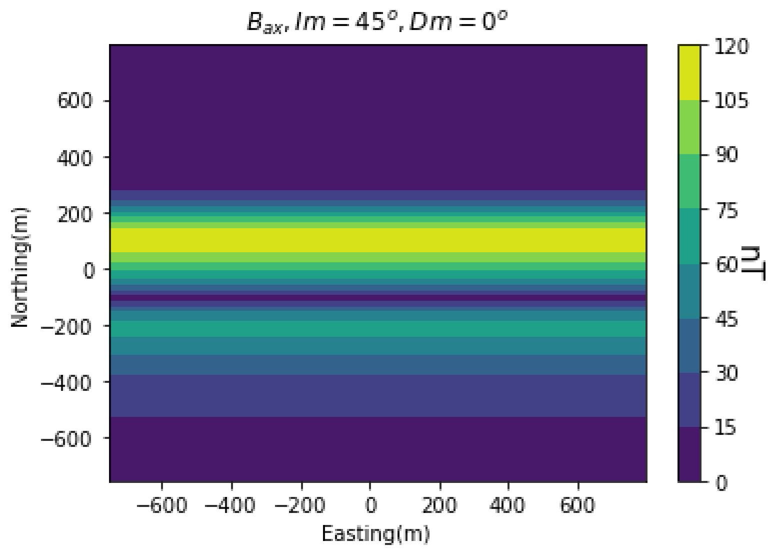


```
C:\Anaconda3\lib\site-packages\numpy\core\numeric.py:531: ComplexWarning: Casti
ng complex values to real discards the imaginary part
  return array(a, dtype, copy=False, order=order)
```

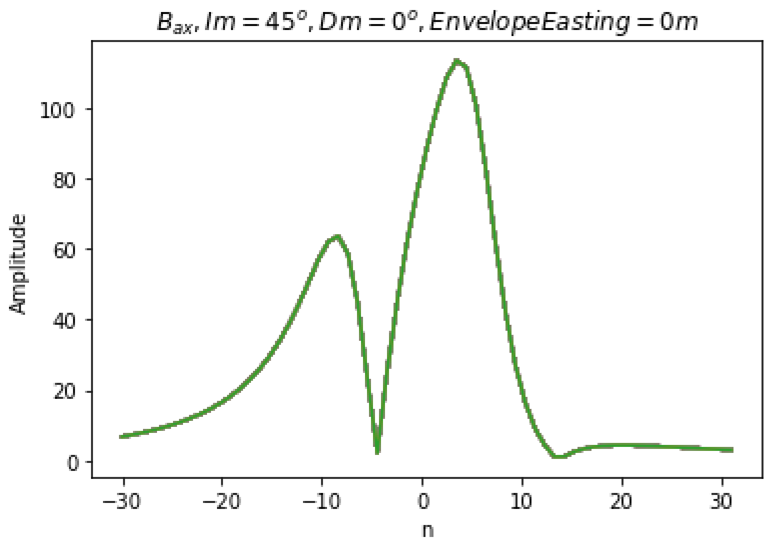


In [61]:
```
### Fold
s = (M/2) + 1 - 1
num = len(n)
A1 = np.zeros((N,N),dtype=np.complex)
A1 = Bazb
for i in range(int(s-1)):
    fold1 = foldonce2D(A1, num)
    A1 = fold1

### FFT
fft1 = np.fft.fft2(fold1)



### operations

Bay1 = np.zeros((num,num),dtype=np.complex)
Bax1 = np.zeros((num,num),dtype=np.complex)
Bax1 = (1j*wx/np.sqrt(1j*wx**2 + 1j*wy**2))*fft1
Bay1 = (1j*wy/np.sqrt(1j*wy**2 + 1j*wx**2))*fft1

Ba = np.sqrt(Bax1**2+Bay1**2)

### Ifft
Bay1i = np.fft.ifft2(Bay1)
Bax1i = np.fft.ifft2(Bax1)

### unfold
A1 = Bay1i
for i in range(int(s-1)):
    Bay1 = unfold2Donce(A1, num)
    A1 = Bay1

A1 = Bax1i
for i in range(int(s-1)):
    Bax1 = unfold2Donce(A1, num)
    A1 = Bax1

##calculate Ba
Ba = np.sqrt(Bay1**2+Bax1**2)
plotting(Ba)
plt.title("$B_{ax} , Im=45^{o}, Dm=0^{o} $")
plt.show()
plt.plot(n,Ba[0:])
plt.title("$B_{ax} , Im=45^{o}, Dm=0^{o} , Envelope Easting =0 m $")
plt.xlabel('n')
plt.ylabel('Amplitude')
plt.show()
```

```
C:\Anaconda3\lib\site-packages\numpy\ma\core.py:2766: ComplexWarning: Casting c
omplex values to real discards the imaginary part
  order=order, subok=True, ndmin=ndmin)
```

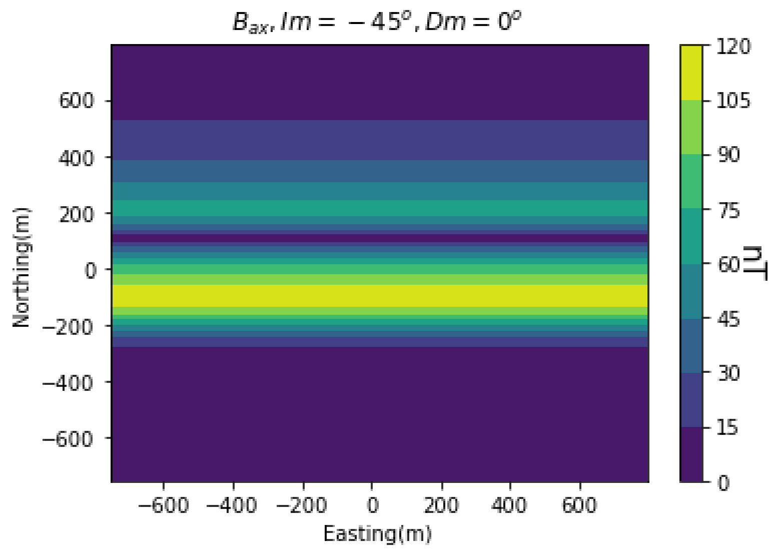


```
C:\Anaconda3\lib\site-packages\numpy\core\numeric.py:531: ComplexWarning: Casti
ng complex values to real discards the imaginary part
  return array(a, dtype, copy=False, order=order)
```

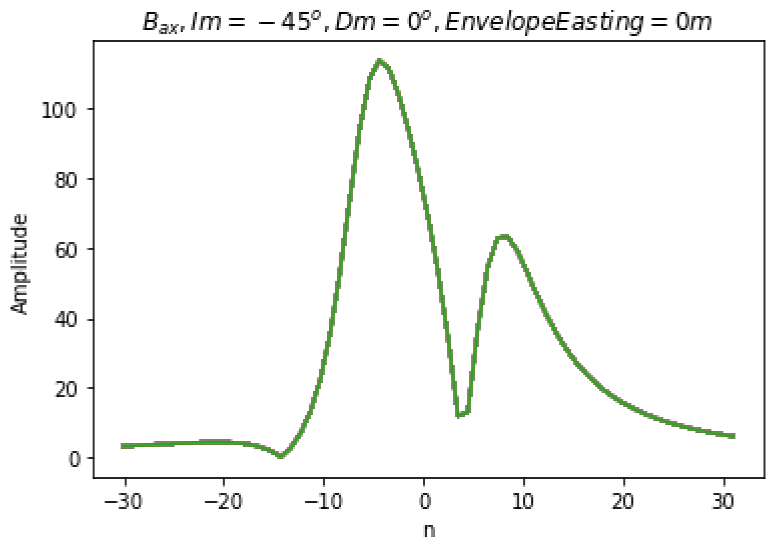


In [62]:
```
### Fold
s = (M/2) + 1 - 1
num = len(n)
A1 = np.zeros((N,N),dtype=np.complex)
A1 = Bazc
for i in range(int(s-1)):
    fold1 = foldonce2D(A1, num)
    A1 = fold1

### FFT
fft1 = np.fft.fft2(fold1)



### operations
Bay1 = np.zeros((num,num),dtype=np.complex)
Bax1 = np.zeros((num,num),dtype=np.complex)
Bax1 = (1j*wx/np.sqrt(1j*wx**2 + 1j*wy**2))*fft1
Bay1 = (1j*wy/np.sqrt(1j*wy**2 + 1j*wx**2))*fft1

Ba = np.sqrt(Bax1**2+Bay1**2)

### Ifft
Bay1i = np.fft.ifft2(Bay1)
Bax1i = np.fft.ifft2(Bax1)

### unfold
A1 = Bay1i
for i in range(int(s-1)):
    Bay1 = unfold2Donce(A1, num)
    A1 = Bay1

A1 = Bax1i
for i in range(int(s-1)):
    Bax1 = unfold2Donce(A1, num)
    A1 = Bax1

##calculate Ba
Ba = np.sqrt(Bay1**2+Bax1**2)
plotting(Ba)
plt.title("$B_{ax} , Im=-45^{o}, Dm=0^{o} $")
plt.show()
plt.plot(n,Ba[0:])
plt.title("$B_{ax} , Im=-45^{o}, Dm=0^{o} , Envelope Easting =0 m$")
plt.xlabel('n')
plt.ylabel('Amplitude')
plt.show()
```

```
C:\Anaconda3\lib\site-packages\numpy\ma\core.py:2766: ComplexWarning: Casting c
omplex values to real discards the imaginary part
  order=order, subok=True, ndmin=ndmin)
```

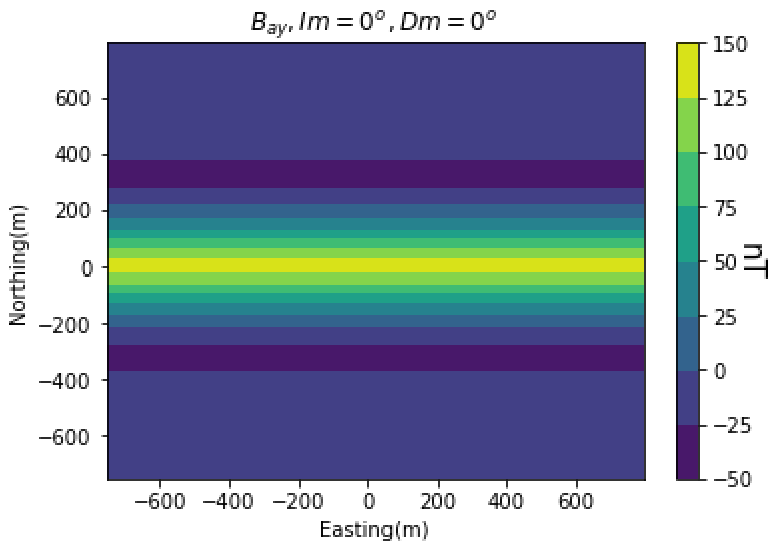


```
C:\Anaconda3\lib\site-packages\numpy\core\numeric.py:531: ComplexWarning: Casti
ng complex values to real discards the imaginary part
  return array(a, dtype, copy=False, order=order)
```

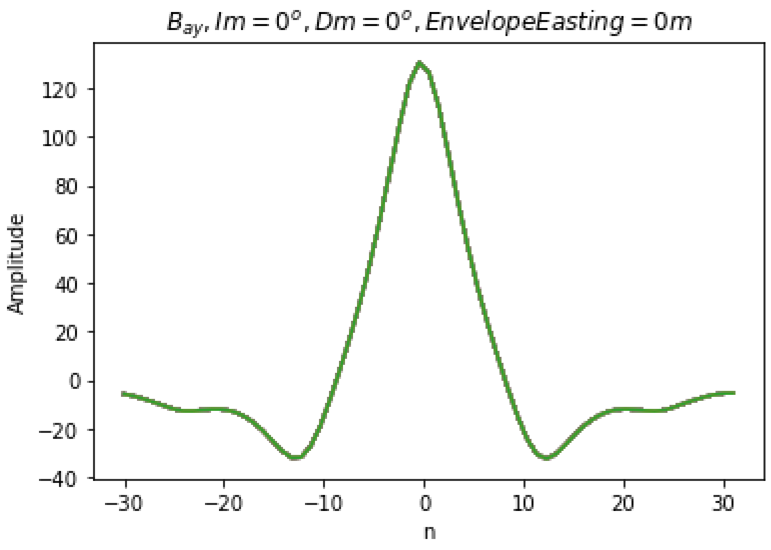


In [63]:
```
### Fold
s = (M/2) + 1 - 1
num = len(n)
A1 = np.zeros((N,N),dtype=np.complex)
A1 = Baz
for i in range(int(s-1)):
    fold1 = foldonce2D(A1, num)
    A1 = fold1

### FFT
fft1 = np.fft.fft2(fold1)

### unfold

A1 = fft1
for i in range(int(s-1)):
    fft1u = unfold2Donce(A1, num)
    A1 = fft1u



### operations
Bay1 = np.zeros((num,num),dtype=np.complex)
Bax1 = np.zeros((num,num),dtype=np.complex)
Bax1 = (1j*wx/np.sqrt(1j*wx**2 + 1j*wy**2))*fft1u
Bay1 = (1j*wy/np.sqrt(1j*wy**2 + 1j*wx**2))*fft1u

Ba = np.sqrt(Bax1**2+Bay1**2)
### fold
A1 = Ba
for i in range(int(s-1)):
    fold1t = foldonce2D(A1, num)
    A1 = fold1t

### Ifft
ai = np.fft.ifft2(fold1t)

### unfold
A1 = ai
for i in range(int(s-1)):
    a1u = unfold2Donce(A1, num)
    A1 = a1u

plotting(a1u)
plt.title("$B_{ay} , Im=0^{o}, Dm=0^{o}$")
plt.show()
plt.plot(n,a1u[0:])
plt.title("$B_{ay} , Im=0^{o}, Dm=0^{o} , Envelope Easting =0 m $")
plt.xlabel('n')
plt.ylabel('Amplitude')
plt.show()
```

```
C:\Anaconda3\lib\site-packages\numpy\ma\core.py:2766: ComplexWarning: Casting c
omplex values to real discards the imaginary part
  order=order, subok=True, ndmin=ndmin)
```

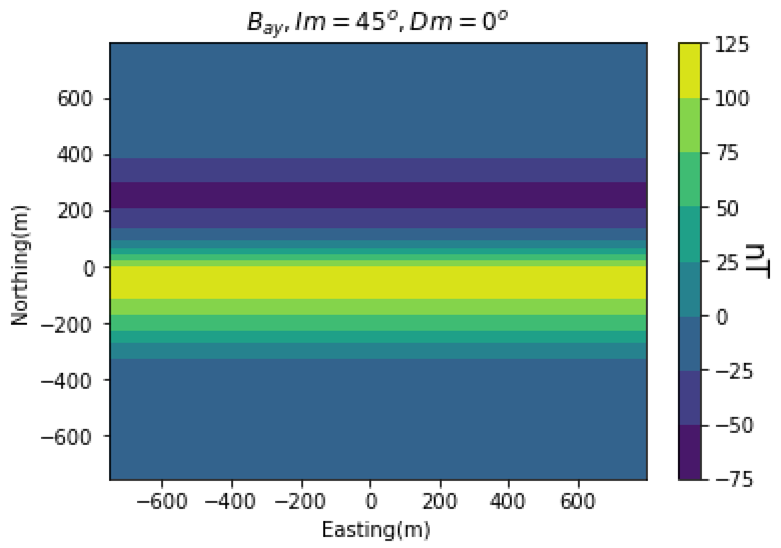


```
C:\Anaconda3\lib\site-packages\numpy\core\numeric.py:531: ComplexWarning: Casti
ng complex values to real discards the imaginary part
  return array(a, dtype, copy=False, order=order)
```

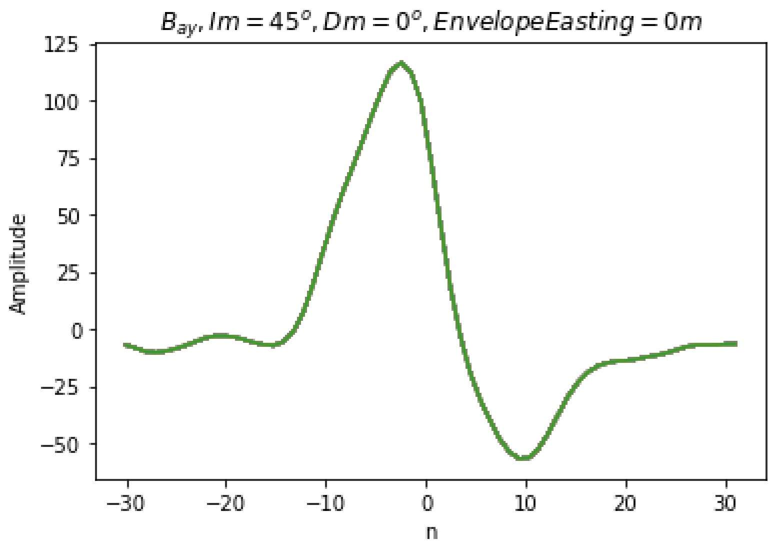


In [64]:
```
### Fold
s = (M/2) + 1 - 1
num = len(n)
A1 = np.zeros((N,N),dtype=np.complex)
A1 = Bazb
for i in range(int(s-1)):
    fold1 = foldonce2D(A1, num)
    A1 = fold1

### FFT
fft1 = np.fft.fft2(fold1)

### unfold

A1 = fft1
for i in range(int(s-1)):
    fft1u = unfold2Donce(A1, num)
    A1 = fft1u


### operations
Bay1 = np.zeros((num,num),dtype=np.complex)
Bax1 = np.zeros((num,num),dtype=np.complex)
Bax1 = (1j*wx/np.sqrt(1j*wx**2 + 1j*wy**2))*fft1u
Bay1 = (1j*wy/np.sqrt(1j*wy**2 + 1j*wx**2))*fft1u

Ba = np.sqrt(Bax1**2+Bay1**2)
### fold
A1 = Ba
for i in range(int(s-1)):
    fold1t = foldonce2D(A1, num)
    A1 = fold1t

### Ifft
ai = np.fft.ifft2(fold1t)

### unfold
A1 = ai
for i in range(int(s-1)):
    a1u = unfold2Donce(A1, num)
    A1 = a1u

plotting(a1u)
plt.title("$B_{ay} , Im=45^{o}, Dm=0^{o} $")
plt.show()
plt.plot(n,a1u[0:])
plt.title("$B_{ay} , Im=45^{o}, Dm=0^{o} , Envelope Easting =0 m$")
plt.xlabel('n')
plt.ylabel('Amplitude')
plt.show()
```

```
C:\Anaconda3\lib\site-packages\numpy\ma\core.py:2766: ComplexWarning: Casting c
omplex values to real discards the imaginary part
  order=order, subok=True, ndmin=ndmin)
```

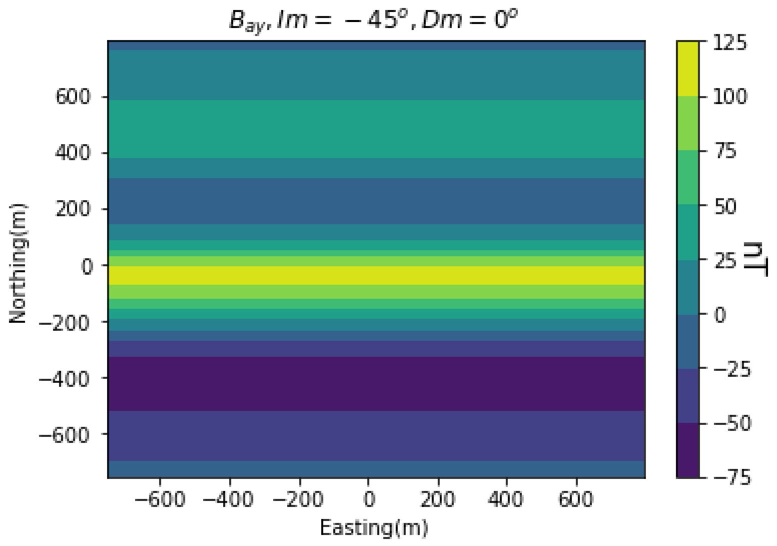


```
C:\Anaconda3\lib\site-packages\numpy\core\numeric.py:531: ComplexWarning: Casti
ng complex values to real discards the imaginary part
  return array(a, dtype, copy=False, order=order)
```

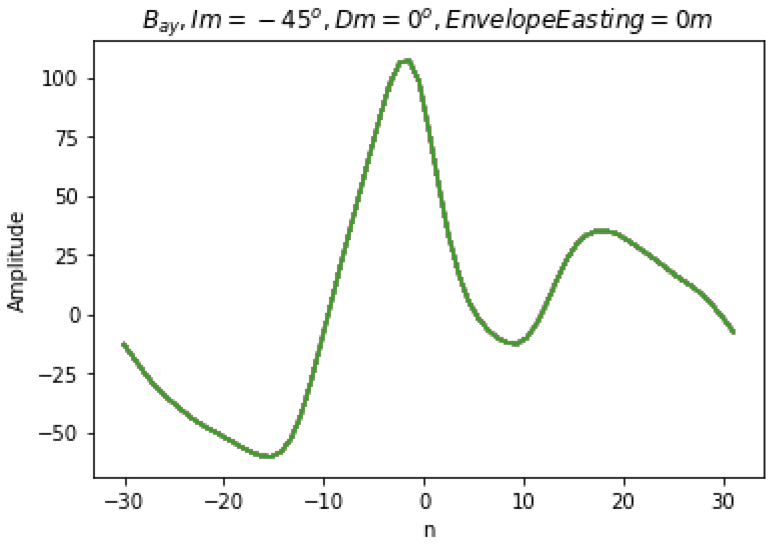


In [65]:
```
### Fold
s = (M/2) + 1 - 1
num = len(n)
A1 = np.zeros((N,N),dtype=np.complex)
A1 = Bazc
for i in range(int(s-1)):
    fold1 = foldonce2D(A1, num)
    A1 = fold1

### FFT
fft1 = np.fft.fft2(fold1)

### unfold

A1 = fft1
for i in range(int(s-1)):
    fft1u = unfold2Donce(A1, num)
    A1 = fft1u


### operations
Bay1 = np.zeros((num,num),dtype=np.complex)
Bax1 = np.zeros((num,num),dtype=np.complex)
Bax1 = (1j*wx/np.sqrt(1j*wx**2 + 1j*wy**2))*fft1u
Bay1 = (1j*wy/np.sqrt(1j*wy**2 + 1j*wx**2))*fft1u

Ba = np.sqrt(Bax1**2+Bay1**2)
### fold
A1 = Ba
for i in range(int(s-1)):
    fold1t = foldonce2D(A1, num)
    A1 = fold1t

### Ifft
ai = np.fft.ifft2(fold1t)

### unfold
A1 = ai
for i in range(int(s-1)):
    a1u = unfold2Donce(A1, num)
    A1 = a1u

plotting(a1u)
plt.title("$B_{ay} , Im=-45^{o}, Dm=0^{o}$")
plt.show()

plt.plot(n,a1u[0:])
plt.title("$B_{ay} , Im=-45^{o}, Dm=0^{o} , Envelope Easting =0 m $")
plt.xlabel('n')
plt.ylabel('Amplitude')
plt.show()
```

```
C:\Anaconda3\lib\site-packages\numpy\ma\core.py:2766: ComplexWarning: Casting c
omplex values to real discards the imaginary part
  order=order, subok=True, ndmin=ndmin)
```



```
C:\Anaconda3\lib\site-packages\numpy\core\numeric.py:531: ComplexWarning: Casti
ng complex values to real discards the imaginary part
  return array(a, dtype, copy=False, order=order)
```



Task III Alternative - Observation and Conclusion

In the alternative bonus task 3, instead of calculating the different components of the magnetic anomaly in the previous task 3 above using the forward modeling code, the vertical component Baz is computed first for each case of the three magnetization directions. Afterwards, using the component conversion code from Lab 06, the Bax and Bay are obtained through the wavenumber domain operations as shown above. Observing the results above, there are very minimal or small changes in the east-west direction but high changes in the north-south direction. Additionally the results are also different for each of the different magnetization direction or changes in Im and Dm scenerios as seen in each of the envelopes when easting = 0m.

G. Lab Conclusion

In conclusion, the lab gave the students a better understanding on wavenumber domain processing of the analytic signal in 2D magnetix data analyses. Overall, the properties of the envelope in 2D potential-field data analyses using the magnetic forward modeling code from Lab 03 and the wavenumber-domain processing codes from Lab 06 was explored. Additionally, the relevant concept of 2D approximation using elongated source bodies, amplitude of analytic signal and envelope, and magnetic amplitude data was understood. The code has tested the property of the horizontal and vertical derivatives of the total-field anomaly produced by a 2D source from Hilbert transform pair following the amplitude of analytic signal as an envelope and as an independent factor of the magnetiation direction. In the end, the code allows the user to test the property by approximating the 2D data sets by calculating them using an elongated source body and using the derivative code that was derived in the previous lab. In addition, the lab allows the students to understand the similar quantities of the magnetic amplitude data.

```
In [ ]: 
```