In [95]:

```
import numpy as np
# Φορτώνω τα αρχεία από το πρώτο αρχείο για περπάτημα
data = np.load('step_00.npz')
data.files
```

Out[95]:

```
['acc', 'gyr', 'hrm']
```

In [96]:

```
accx=[] #Σε αυτές τις λίστες αποθηκεύω τα δείγματα του αξελερόμετρου για τους 3 άξονες
accy=[]
accz=[]
for x in data['acc'] :
    accx.append(x[0])
    accy.append(x[1])
    accz.append(x[2])

gyrx=[] #Σε αυτές τις λίστες αποθηκεύω τα δείγματα του γυροσκόπειου για τους 3 άξονες
gyry=[]
gyrz=[]
for x in data['gyr'] :
    gyrx.append(x[0])
    gyry.append(x[1])
    gyrz.append(x[2])

hrm=data['hrm']
```

In [11]:

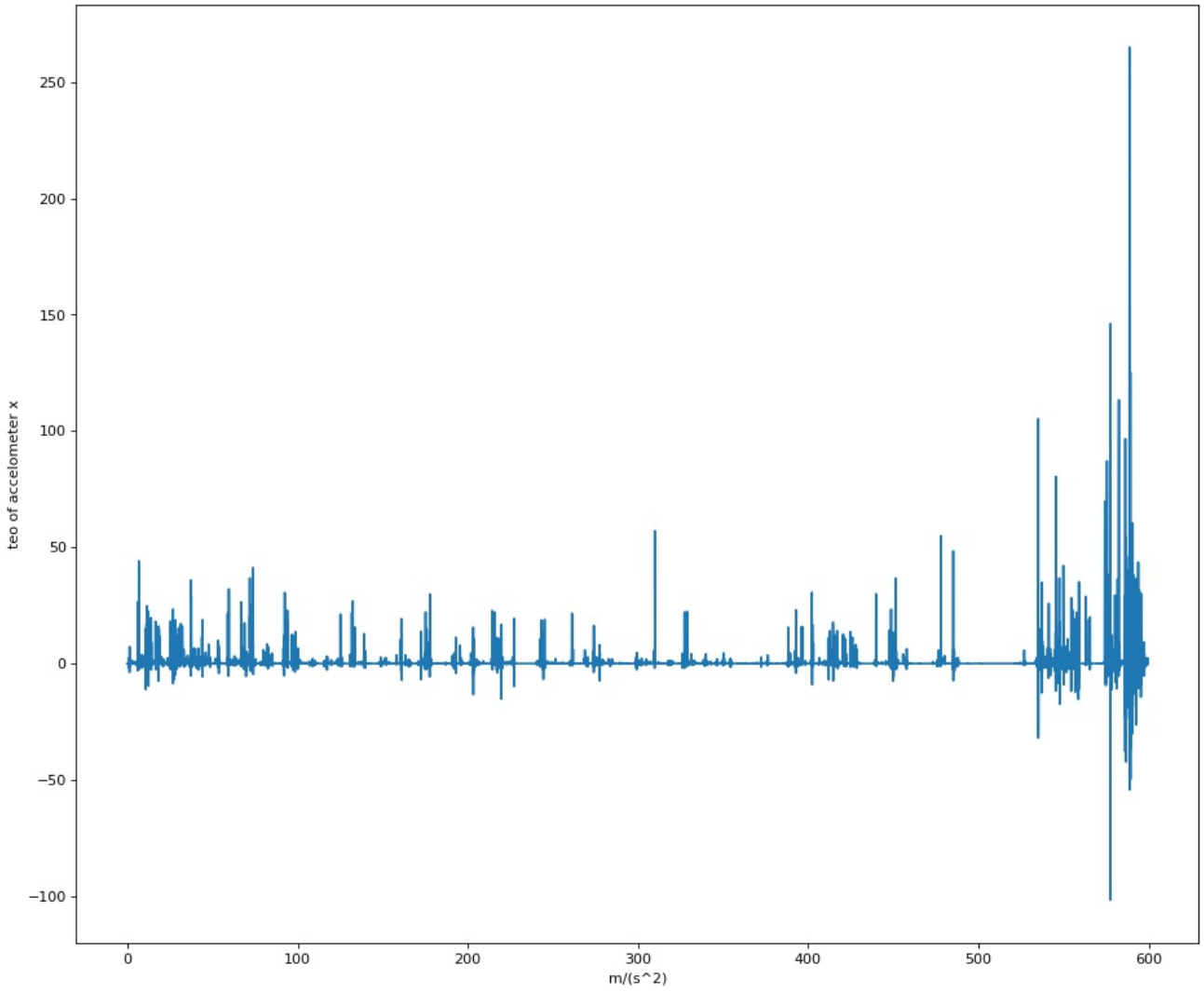
```
def teo_help (signal,n): #Help function that computes TEO operator at points
    if (n==np.size(signal)-1):  #at the end signal(n+1)=signal(n)
        return (((signal[n])**2)-signal[n-1]*signal[n])
    if (n==0): #at start signal(n-1)=signal(n)
        return (((signal[0])**2)-signal[0]*signal[1])
    return ((signal[n])**2 - signal[n-1]*signal[n+1]) #if we are not at the start or at the end that returns TEO
at a mid-point
```

In [12]:

```
def teo (signal): #Teager Energy Operator
    y=[]
    for i in range (0,len(signal)): #Use teo_help len(signal) times
        y.append(teo_help(signal,i))
    return y
```

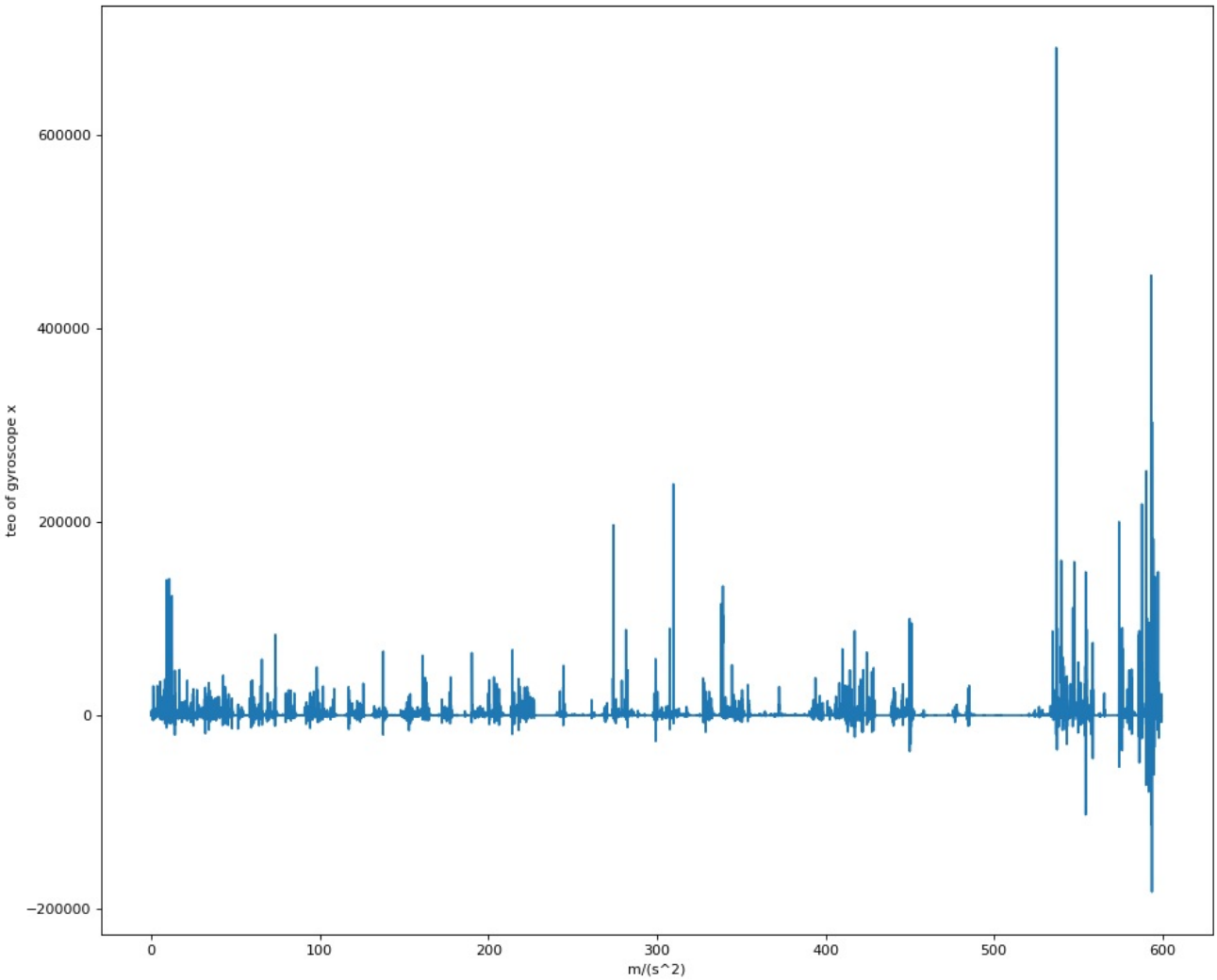
In [13]:

```
import matplotlib.pyplot as plt
#πλοπάρω τον Τελεστή πάνω στο σήμα accX χωρίς να ζητείται απλά για να δω την μορφή του
fig=plt.figure(figsize=(14, 12), dpi= 80, facecolor='w', edgecolor='k')
t20=np.arange(0.,599.6,0.05)
plt.plot(t20,teo(accx[:]))

plt.ylabel('teo of accelometer x')
plt.xlabel('m/(s^2)')
plt.show()
```



In [18]:

```
import matplotlib.pyplot as plt
fig=plt.figure(figsize=(14, 12), dpi= 80, facecolor='w', edgecolor='k')
t20=np.arange(0.,599.6,0.05)
plt.plot(t20,teo(gyrx[:]))

plt.ylabel('teo of gyroscope x')
plt.xlabel('m/(s^2)')
plt.show()
```



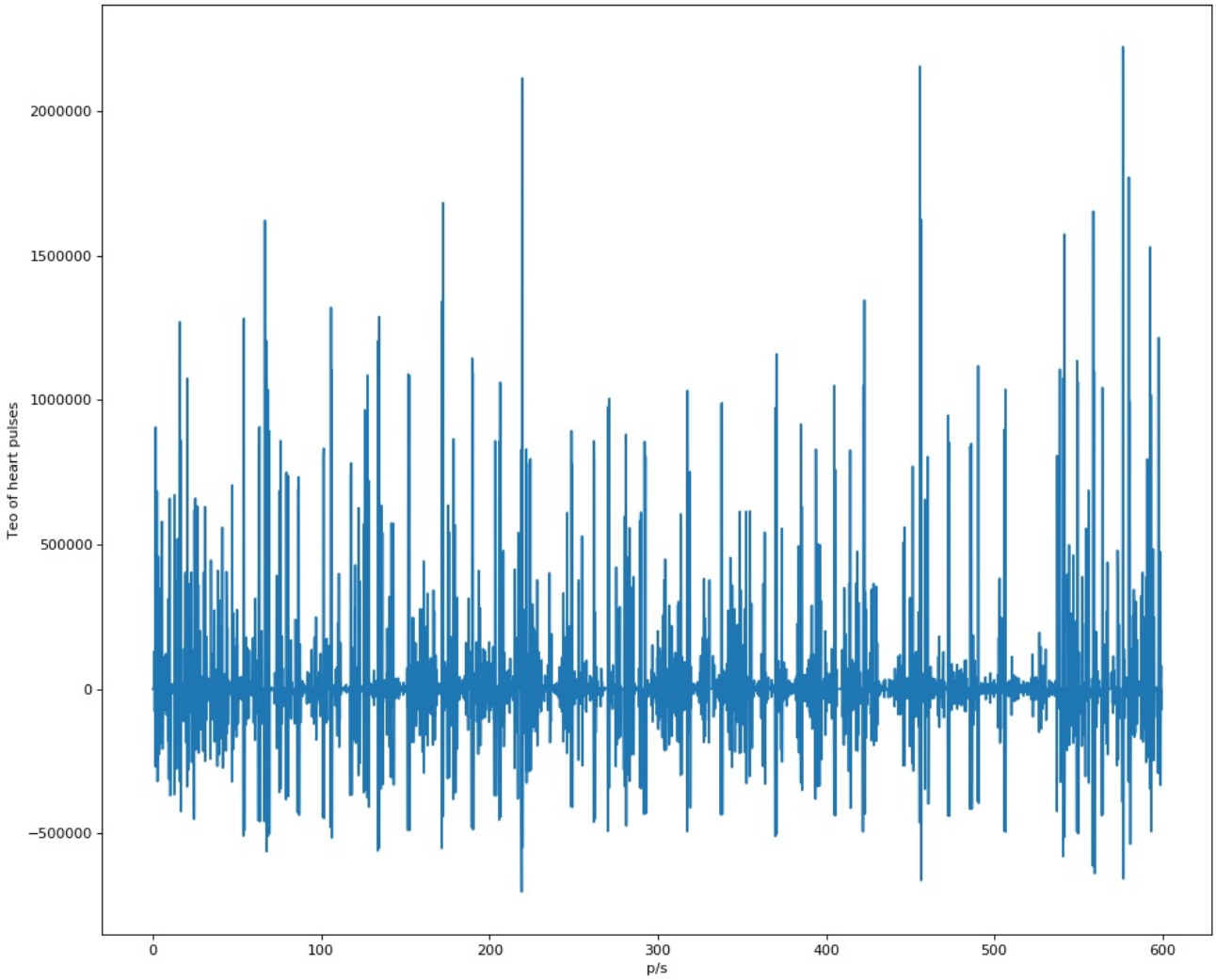
Παρατηρώ ότι υπάρχουν διαστήματα όπου ο Τελεστής παίρνει μηδενικές τιμές.

In [15]:

```
fig=plt.figure(figsize=(14, 12), dpi= 80, facecolor='w', edgecolor='k')
t5=np.arange(0.,599.6,0.2)
plt.plot(t5,teo(hrm[:]))
plt.ylabel('Teo of heart pulses')
plt.xlabel('p/s')
```

Out[15]:

```
Text(0.5, 0, 'p/s')
```



In [19]:

```
def Gabor_help (n,fc,a,fs): #help function that computes Gabor fun at points
    b=a/fs
    Omega_c=(2*(np.pi))*(fc/fs)
    b_sq=b**2
    return np.exp(-b_sq*(n**2))*np.cos(Omega_c*n)
```

In [20]:

```
def Gabor (fc,a,fs): #Help function that computes Gabor h(n) function
    b=a/fs
    N=int ((3/b)+1)
    y=[]
    for i in range(-N,N+1):
        y.append(Gabor_help(i,fc,a,fs))
    return y
```

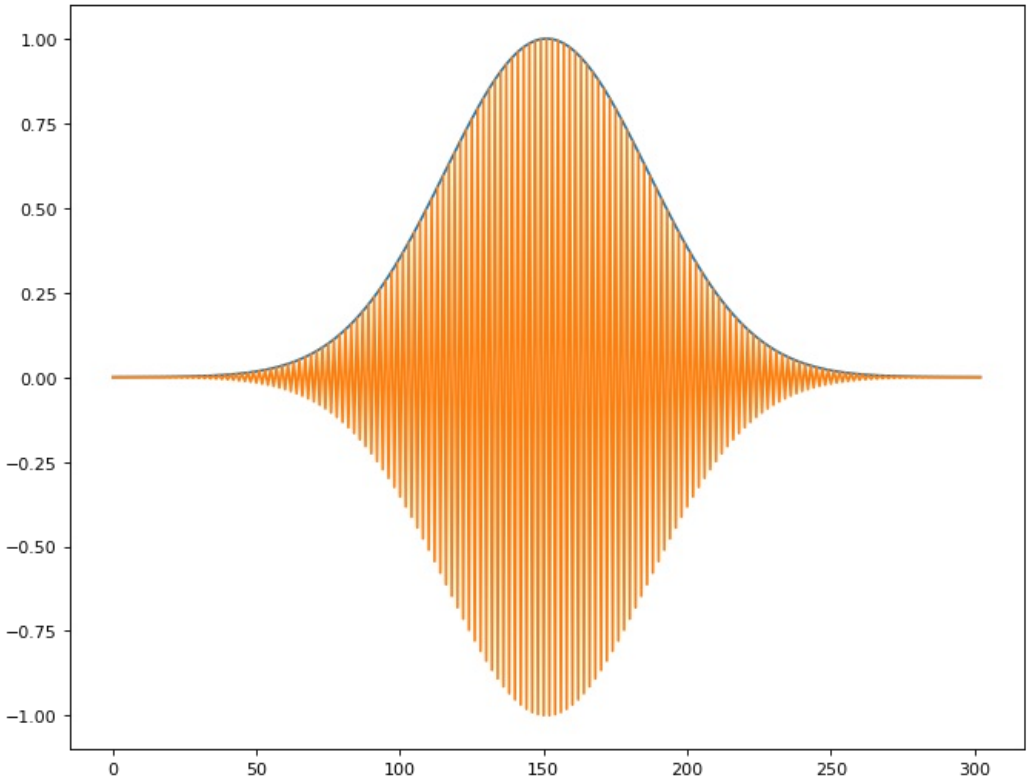
Πλοτάρω τα φίλτρα που δημιούργησα γύρω από μία συγκεκριμένη συχνότητα να δω την μορφή τους .

In [21]:

```
fig=plt.figure(figsize=(10, 8), dpi= 80)
plt.plot(np.abs(Gabor(10,0.4,20)))
plt.plot((Gabor(10,0.4,20)))

plt.figure()
```

Out[21]:

```
<Figure size 432x288 with 0 Axes>
```



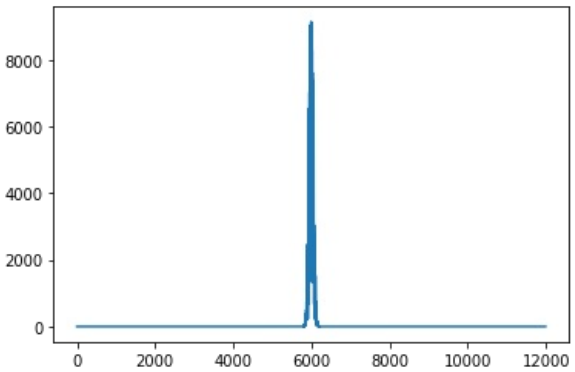

```
<Figure size 432x288 with 0 Axes>
```

In [22]:

```
#Το φάσμα του Gabor φίλτρου
plt.plot(np.abs(np.fft.fft(accx)*np.fft.fft(Gabor(10,.4,20),np.size(accx))))
```

Out[22]:

```
[<matplotlib.lines.Line2D at 0x1146cc650>]
```

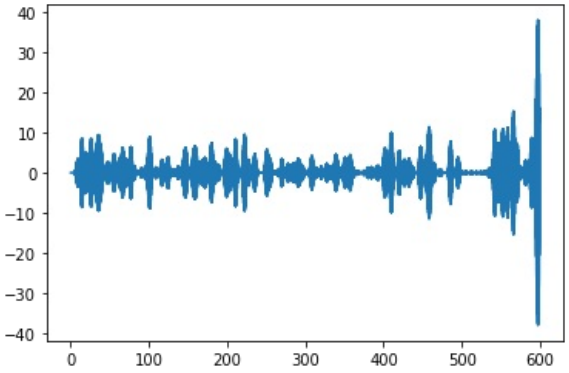


In [23]:

```
import scipy as sp
from scipy import signal
from scipy.signal import lfilter
zi = signal.lfilter_zi(Gabor(10,.4,20), [1])
z, _ = signal.lfilter(Gabor(10,.4,20),[1],accx,zi=zi*accx[0])
plt.plot(t20,z)
```

Out[23]:

```
[<matplotlib.lines.Line2D at 0x1130a6910>]
```



In [99]:

```
import scipy as sp
from scipy import signal
from scipy.signal import lfilter
def gaborfilt(shma,fc,a,fs): #function that filters a signal with Gabor filter
    return scipy.signal.lfilter(Gabor(fc,a,fs),1,shma)
```

In [57]:

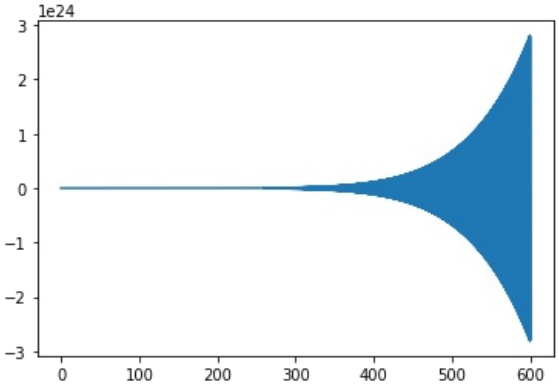
```
def filtering (shma,fs): #Functions that filters a signal with 25 Gabor filter at different frequencies
    K=25
    a=fs/(2*K)
    fcmin=0 #starting frequency
    fcmax=(fs/2) #final frequency
    d=fcmax-fcmin # subspace between starting and final frequency
    sub_of_filts=d/25 #subspace between two consecutive frequencies
    ans=[] #list of the 25 filtered signals
    for j in range(0,25):
        y=[]
        fc_pr=fcmin+(j*sub_of_filts) #Central frequency in filtering the signal
        y.append(gaborfilt(shma,fc_pr,a,fs))
        ans.append(list(y))
    return ans #return 25 lists => 25 times Gabor filtered signal at different frequencies
```

In [59]:

```
plt.plot(t20,gaborfilt(accx,0.2,0.4,20))
#πλοτάρρω δοκιμαστικά ένα Gabor φιλτραρισμένο σήμα για να δω την μορφή του
```

Out[59]:

```
[<matplotlib.lines.Line2D at 0x11368f290>]
```



In [60]:

```
sig=[accx,gyrx]
a=20/(2*25)
fcmin=a/2
fcmax=(20-a)/2
d=fcmax-fcmin
sub_of_filts=d//25
y20=[]
for i in  sig :
    for j in range(0,25):
        fc_pr=fcmin+(j*sub_of_filts)
        y20.append(gaborfilt(i,fc_pr,a,20))
```

In [61]:

```
#Εναλλακτικά
y20=[]
y20.append(filtering(accx,20))
y20.append(filtering(gyrx,20))
```

In [62]:

```
np.shape(y20)
```

Out[62]:

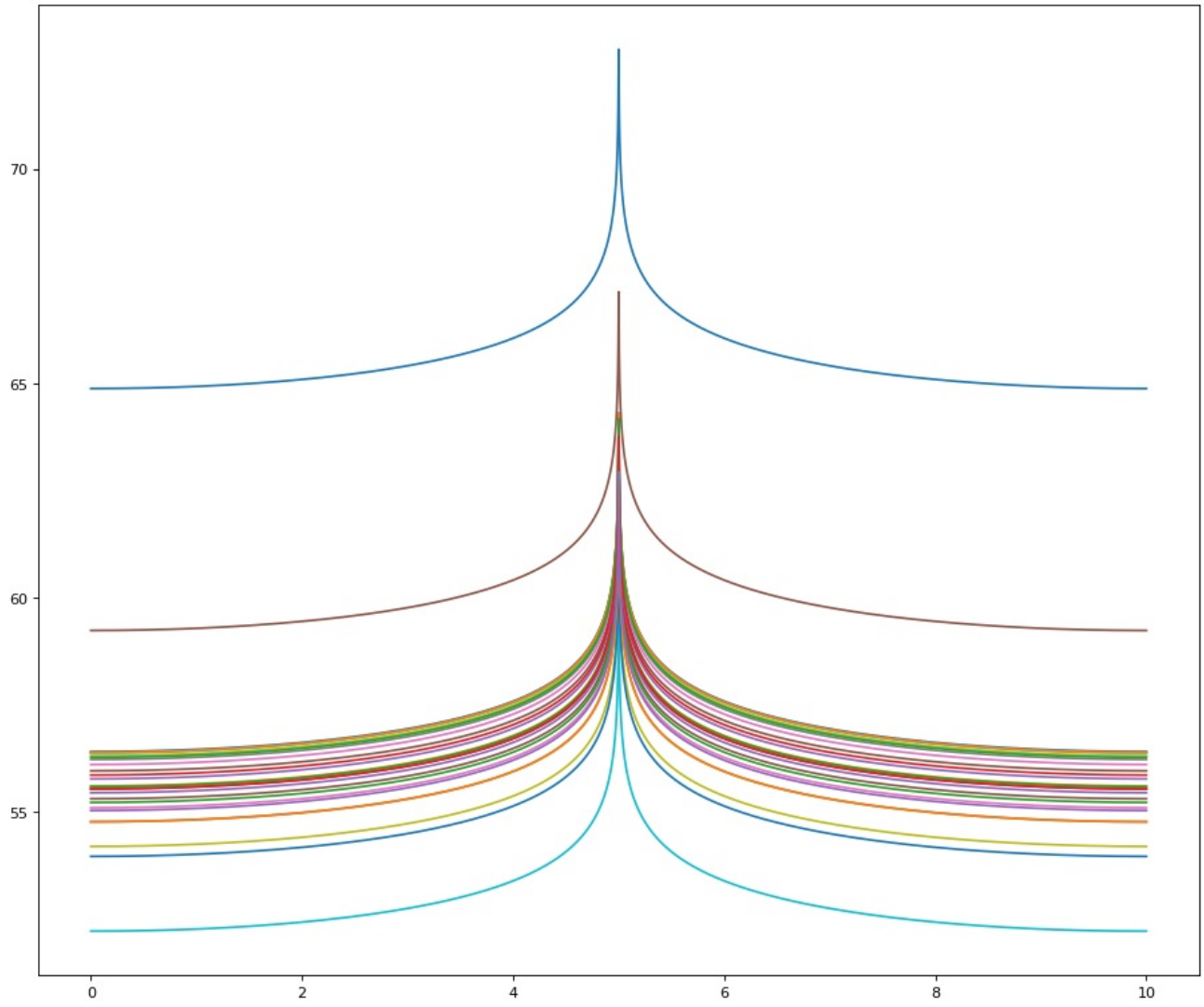
```
(2, 25, 1, 11992)
```

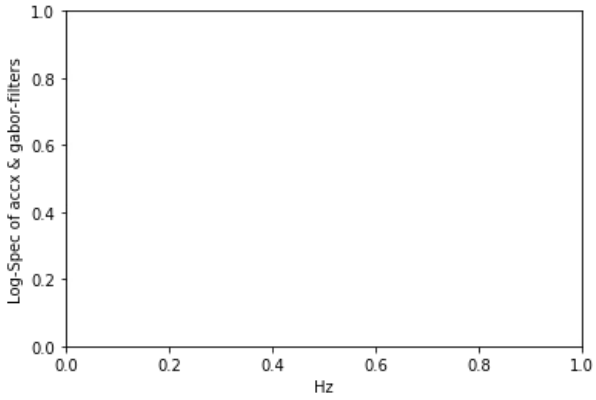
In [63]:

```
fig=plt.figure(figsize=(14, 12), dpi= 80, facecolor='w', edgecolor='k')
f20=np.linspace(0,10,np.size(np.fft.fft(accx)))
plt.plot(f20,np.log(np.abs(np.fft.fft(accx))))
for i in range(0,25):
    plt.plot(f20,np.log(np.abs(np.fft.fft(y20[0][i][0]))))
plt.figure()
plt.ylabel('Log-Spec of accx & gabor-filters')
plt.xlabel('Hz')
```

Out[63]:

```
Text(0.5, 0, 'Hz')
```



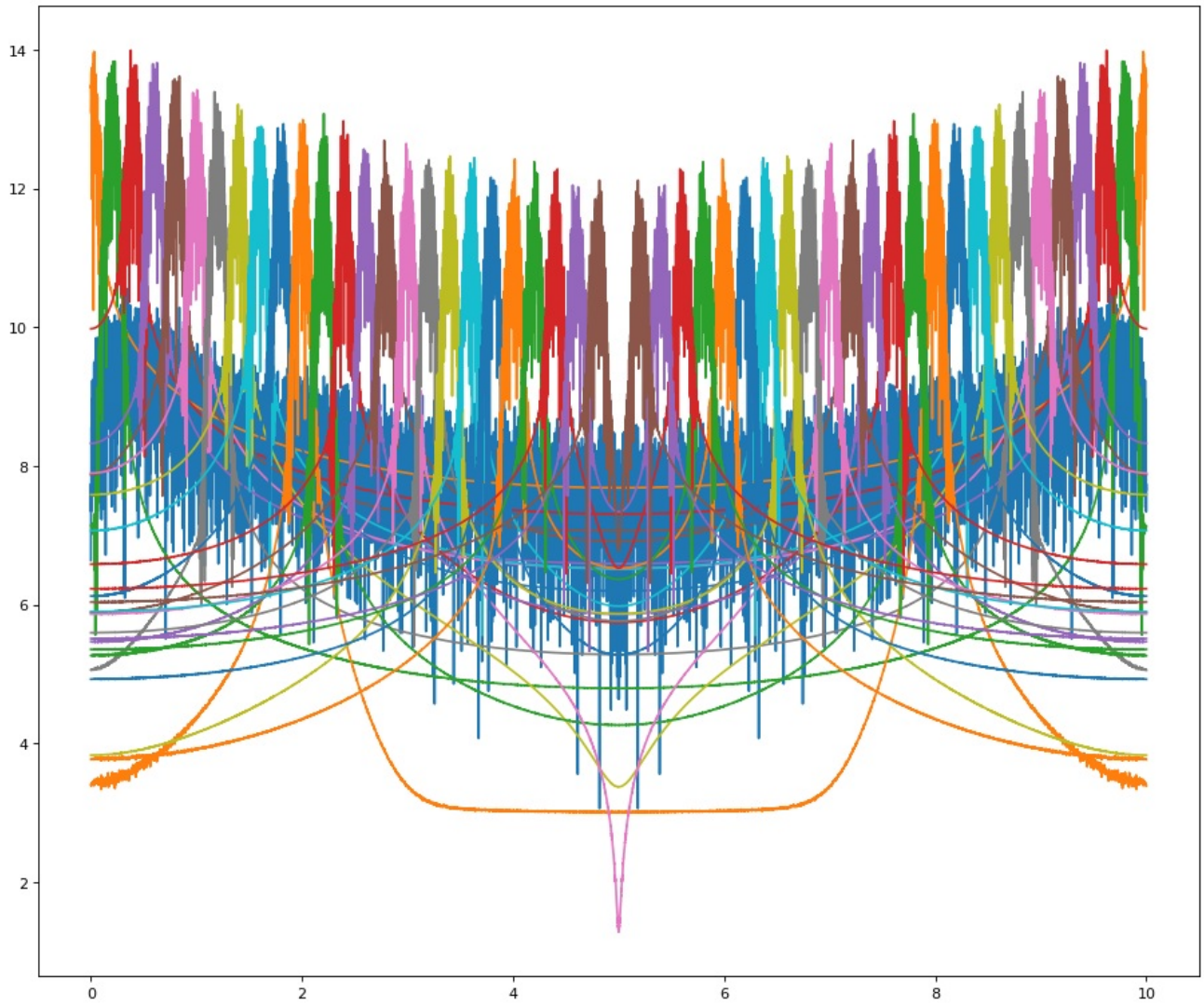


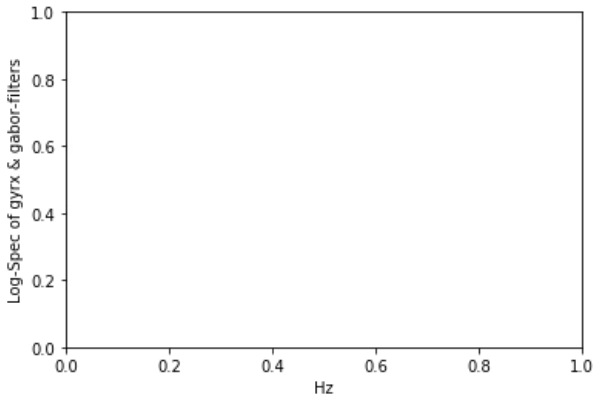
In [66]:

```
fig=plt.figure(figsize=(14, 12), dpi= 80, facecolor='w', edgecolor='k')
f20=np.linspace(0,10,np.size(np.fft.fft(gyrx)))
plt.plot(f20,np.log(np.abs(np.fft.fft(gyrx))))
for i in range(0,25):
    plt.plot(f20,np.log(np.abs(np.fft.fft(y20[1][i][0]))))
plt.figure()
plt.ylabel('Log-Spec of gyrx & gabor-filters')
plt.xlabel('Hz')
```

Out[66]:

```
Text(0.5, 0, 'Hz')
```





In [32]:

```
a=5/(2*25)
fcmin=a/2
fcmax=(20-a)/2
d=fcmax-fcmin
sub_of_filts=d//25
y5=[]
for i in  sig :
    for j in range(0,25):
        fc_pr=fcmin+(j*sub_of_filts)
        y5.append(gaborfilt(hrm,fc_pr,a,20))
```

In [68]:

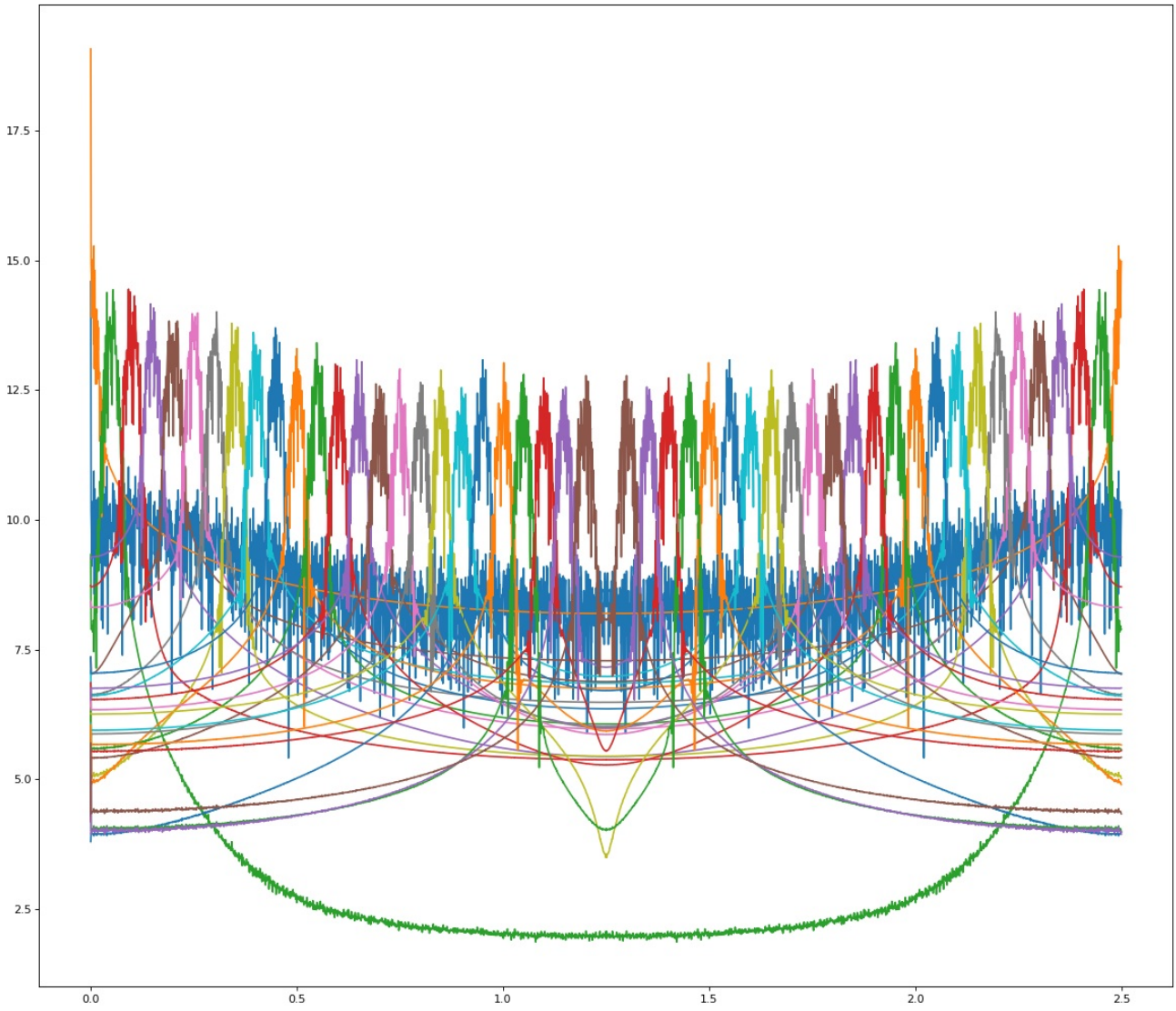
```
#Εναλλακτικά
y5=[]
y5.append(filtering(hrm,5))
```

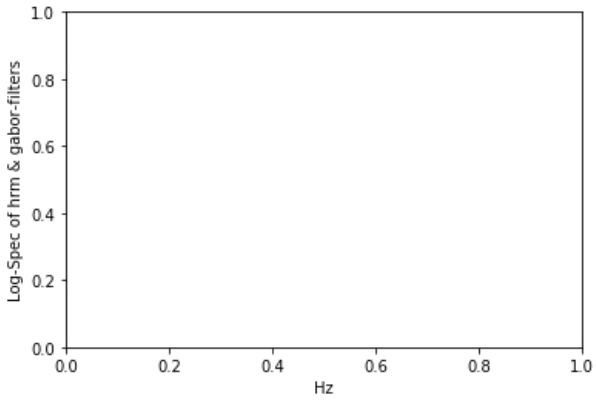
In [69]:

```
fig=plt.figure(figsize=(18, 16), dpi= 80, facecolor='w', edgecolor='k')
f5=np.linspace(0,2.5,np.size(np.fft.fft(hrm)))
plt.plot(f5,np.log(np.abs(np.fft.fft(hrm))))
for i in range(0,25):
    plt.plot(f5,np.log(np.abs(np.fft.fft(y5[0][i][0]))))
plt.figure()
plt.ylabel('Log-Spec of hrm & gabor-filters')
plt.xlabel('Hz')
```

Out[69]:

```
Text(0.5, 0, 'Hz')
```





Υλοποίηση του binomial φίλτρου

In [70]:

```
def binomial (shma):
    return signal.lfilter ([0, 1, 0], [.25, .5 ,.25],shma)
```

In [71]:

```
accx=binomial(binomial(accx)) #περνάω δύο φορές το σήμα από binomial φίλτρο
print(accx)
```

```
[ 0.00000000e+00  0.00000000e+00  0.00000000e+00 ...  8.95953824e+41
 -8.96844406e+41  8.97735795e+41]
```

Προσθέτω μηδενικά στα σήματα μου ώστε να ταιριάζουν τα μεγέθη με αυτά των παραθύρων στην παραθύρωση. Δηλαδή στο τελευταίο κομμάτι του αρχικού σήματος που μένει στην παραθύρωση αν είναι για παράδειγμα 392 θέσεων δεν μπορώ να το παραθυρώσω με παράθυρο 400 δειγμάτων .

In [72]:

```
accX_pad=np.pad(accx,(0,8),'constant',constant_values=(0,0))
gyrX_pad=np.pad(gyrx,(0,8),'constant',constant_values=(0,0))
hrm_pad=np.pad(hrm,(0,2),'constant',constant_values=(0,0))
#print(np.size(accX_pad))
#print(np.size(hrm_pad))
```

In [104]:

```
def hamming_windowing (shma,fs):
    win=np.hamming(fs*20) #hamming window of length 20 sec
    shift=fs*5  # Shift between two consecutive windows
    lim=np.size(shma)-np.size(win)+1 #max value where we can window the signal
    ap=[] #return list with the winowed signals
    for i in range (0,lim,shift):
           ap.append(shma[i:i+len(win)]*win)
    return ap   #return the windows of the signal
```

Στις παρακάτω μεταβλητές αποθηκεύω τις λίστες που περιέχουν τα παραθυρωμένα διαστήματα του σήματος μου .

In [74]:

```
win_of_accX=hamming_windowing(accX_pad,20)
win_of_gyrX=hamming_windowing(gyrX_pad,20)
win_of_hrm=hamming_windowing(hrm_pad,5)
```

In [105]:

```
def hamming_windowing (shma,fs):
    win=np.hamming(fs*20) #hamming window of length 20 sec
    shift=fs*5  # Shift between two consecutive windows
    lim=np.size(shma)-np.size(win)+1 #max value where we can window the signal
    ap=[] #return list with the winowed signals
    for i in range (0,lim,shift):
           ap.append(shma[i:i+len(win)]*win)
    return ap   #return the windows of the signal
```

In [106]:

```
def hamming_windowing_windows (shma,fs):
    win=np.hamming(fs*20)
    shift=fs*5
    lim=np.size(shma)-np.size(win)
    r=[]
    for i in range (0,lim,shift):
        y=[]
        for j in range (0,(fs*20)):
            y.append(shma[i+j]*win[j])
        r.append(y)
    return r
```

In [107]:

```
a=20/(2*25)
fcmin=0
fcmax=10
d=fcmax-fcmin
sub_of_filts=d/25
accX_25=[]
for j in range(0,25):
    y=[]
    fc_pr=fcmin+(j*sub_of_filts)
    y.append(gaborfilt(win_of_accX,fc_pr,a,20))
    accX_25.append(y)
```

In [108]:

```
a=20/(2*25)
fcmin=0
fcmax=10
d=fcmax-fcmin
sub_of_filts=d/25
gyrX_25=[]
for j in range(0,25):
    y=[]
    fc_pr=fcmin+(j*sub_of_filts)
    y.append(gaborfilt(win_of_gyrX,fc_pr,a,20))
    gyrX_25.append(y)
```

In [109]:

```
a=10/(2*25)
fcmin=0
fcmax=2.5
d=fcmax-fcmin
sub_of_filts=d/25
hrm_25=[]
for j in range(0,25):
    y=[]
    fc_pr=fcmin+(j*sub_of_filts)
    y.append(gaborfilt(win_of_hrm,fc_pr,a,5))
    hrm_25.append(y)
```

In [110]:

```
np.shape(hrm_25)
```

Out[110]:

```
(25, 1, 3000)
```

In [111]:

```
accX_teo=[]
for i in range (0,25):
    y=[]
    y.append(teo(accX_25[i][0]))
    accX_teo.append(y)
```

In [112]:

```
gyrX_teo=[]
for i in range (0,25):
    y=[]
    y.append(teo(gyrX_25[i][0]))
    gyrX_teo.append(y)
```

In [113]:

```
hrm_teo=[]
for i in range (0,25):
    y=[]
    y.append(teo(hrm_25[i][0]))
    hrm_teo.append(y)
```

In [114]:

```
np.shape(hrm_teo)
```

Out[114]:

```
(25, 1, 3000)
```

In [115]:

```
accX_teo_flt=[]
for i in range (0,25):
    y=[]
    y.append(binomial(binomial(accX_teo[i][0])))
    accX_teo_flt.append(y)
```

In [116]:

```
gyrX_teo_flt=[]
for i in range (0,25):
    y=[]
    y.append(binomial(binomial(gyrX_teo[i][0])))
    gyrX_teo_flt.append(y)
```

In [117]:

```
hrm_teo_flt=[]
for i in range (0,25):
    y=[]
    y.append(binomial(binomial(hrm_teo[i][0])))
    hrm_teo_flt.append(y)
```

In [118]:

```
non_zero_accX=[]
non_zero_gyrX=[]
non_zero_hrm=[]
for i in range(0,25):
    non_zero_accX.append(list(filter(lambda num: num != 0, accX_teo_flt[i][0][:])))
    non_zero_gyrX.append(list(filter(lambda num: num != 0, gyrX_teo_flt[i][0][:])))
    non_zero_hrm.append(list(filter(lambda num: num != 0, hrm_teo_flt[i][0][:])))
```

In [119]:

```
import statistics
mean_energy_accX=[]
mean_energy_gyrX=[]
mean_energy_hrm=[]
for i in range (0,25):
    mean_energy_accX.append(statistics.mean(non_zero_accX[i]))
    mean_energy_gyrX.append(statistics.mean(non_zero_gyrX[i]))
    mean_energy_hrm.append(statistics.mean(non_zero_hrm[i]))
```

In [120]:

```
def find_max_and_pos (shma):
    p=shma[0]
    position=0
    current_position=0
    for i in shma:
        if (i>p):
            p=i
            position=current_position
        current_position+=1
    return (p,position)
```

In [121]:

```
print(find_max_and_pos(mean_energy_hrm))
print(find_max_and_pos(mean_energy_accX))
print(find_max_and_pos(mean_energy_gyrX))
```

```
(230008873565.71338, 12)
(1.8811365634043112e+84, 0)
(109098248927.10313, 22)
```

In [124]:

```
def Mean_Teager_Energy(shma): #function that computes mean value of a list only for positive values
    return np.mean(list(filter(lambda num: num > 0, shma)))
```

In [125]:

```
#Εναλλακτικά όλα τα παραπάνω συγκεντρωμένα σε μία συνάρτηση
def Mean_Teager (shma,fs):
    ap=[]#λίστα όπου θα αποθηκεύσουμε την μέση τιμή από κάθε Teager Energy
    a=hamming_windowing(shma,fs) #Στο a θα αποθηκευτούν τα 116 (μετά από δοκιμές παρατήρησα ότι τόσα βγαίνουν) παράθυρα
    A=np.shape(a)[0] #Στο A θα αποθηκευτεί ο #παραθύρων (δηλαδή 116)
    filtra=[] #εδώ θα αποθηκευτούν τα 116 παράθυρα ώστε το καθένα να φιλτραριστεί 25 φορές
    for i in range(0,A): #φιλτράρω το κάθε παράθυρο , αυτό θα μου επιστρέψει λίστα(#παραθύρων X 25 φιλτραρισμένες λίστες)
        filtra.append(filtering(a[i],fs))
    for i in range (0,A): #Εφαρμόζω Teager τελεστή ενέργειας σε κάθενα από τα 25 παράθυρα του σήματος X όλα τα παράθυρα (
σσ.116)
        for j in range (0,25):
            filtra[i][j][0]=teo(filtra[i][j][0])
    for i in range(0,A):
        for j in range (0,25):#Παιρνώ κάθε Teager Energy από binomial φίλτρο 2 φορές
            filtra[i][j][0]=binomial(filtra[i][j][0])
            filtra[i][j][0]=binomial(filtra[i][j][0])
    mesh_Teager_list=[] #λίστα όπου αποθηκεύω την μέση τιμή μη-μηδενικών τιμών Μέσης-Teager Ενέργειας
    max_mean_list=[]
    for i in range (0,A):
        mesh_Teager_list=[] #λίστα όπου αποθηκεύω την μέση τιμή μη-μηδενικών τιμών Μέσης-Teager Ενέργειας
        for j in range (0,25):
            mesh_Teager_list.append(Mean_Teager_Energy(filtra[i][j][0]))
        max_mean_list.append(mesh_Teager_list)
    ap=[]
    for i in max_mean_list:
        ap.append(np.max(i))
    return ap
```

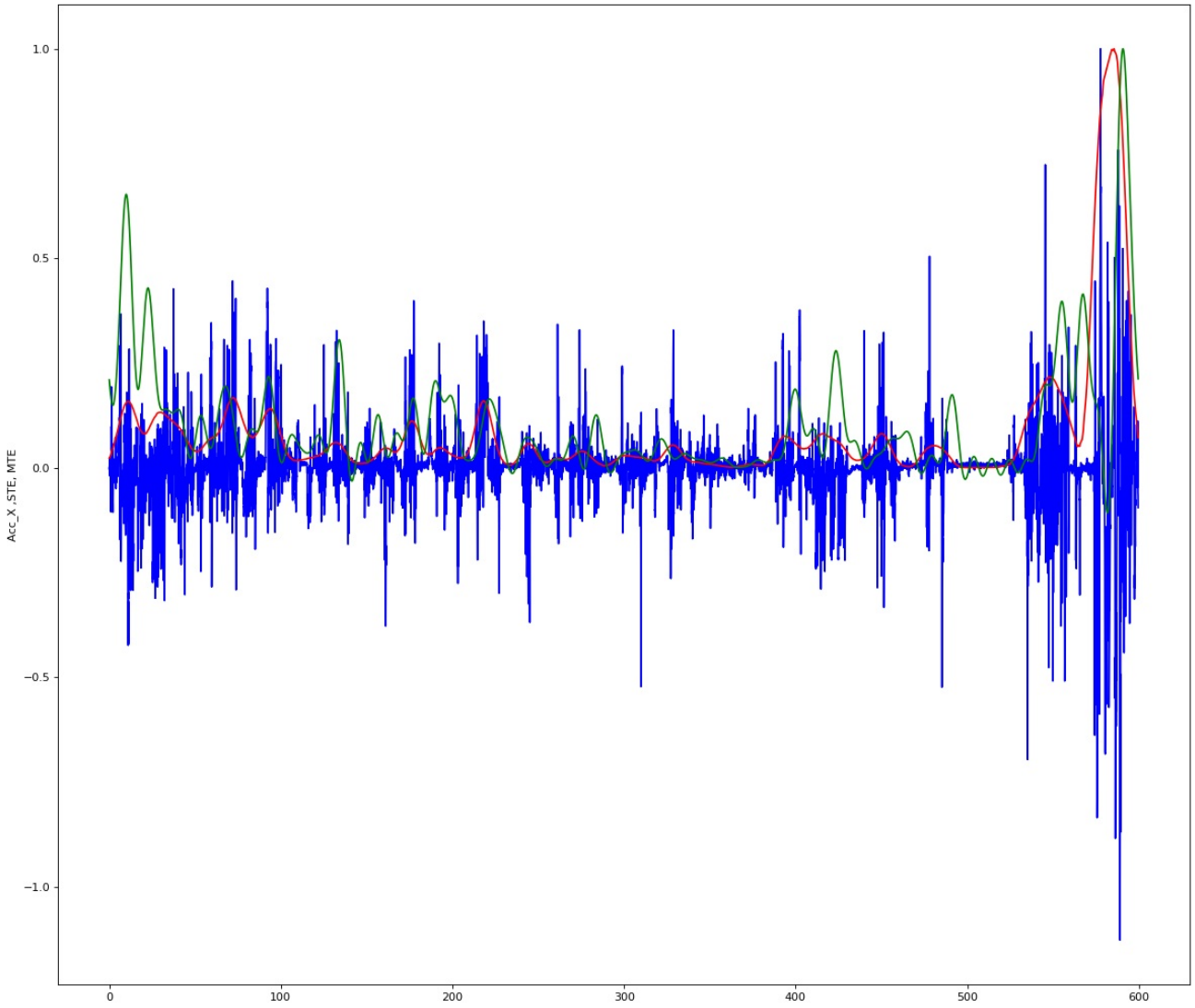
Για το πλοτάρισμα θα πρέπει η ενέργεια βραχέως χρόνου και η μέγιστη μέση Teager Ενέργεια να έχουν το ίδιο μέγεθος άρα κάνουμε τα εξής:

In [129]:

```
import scipy.signal
def square(list):
    return [i ** 2 for i in list] #square every element in a list

def ste(x,fs): #Compute short-time energy.
    win=np.hamming(20*fs) #Declaration of a hamming window of length 20 sec
    return scipy.signal.convolve(square(np.abs(x)), win, mode="same")#Returns convolution with the hamming window
of length 20s
```

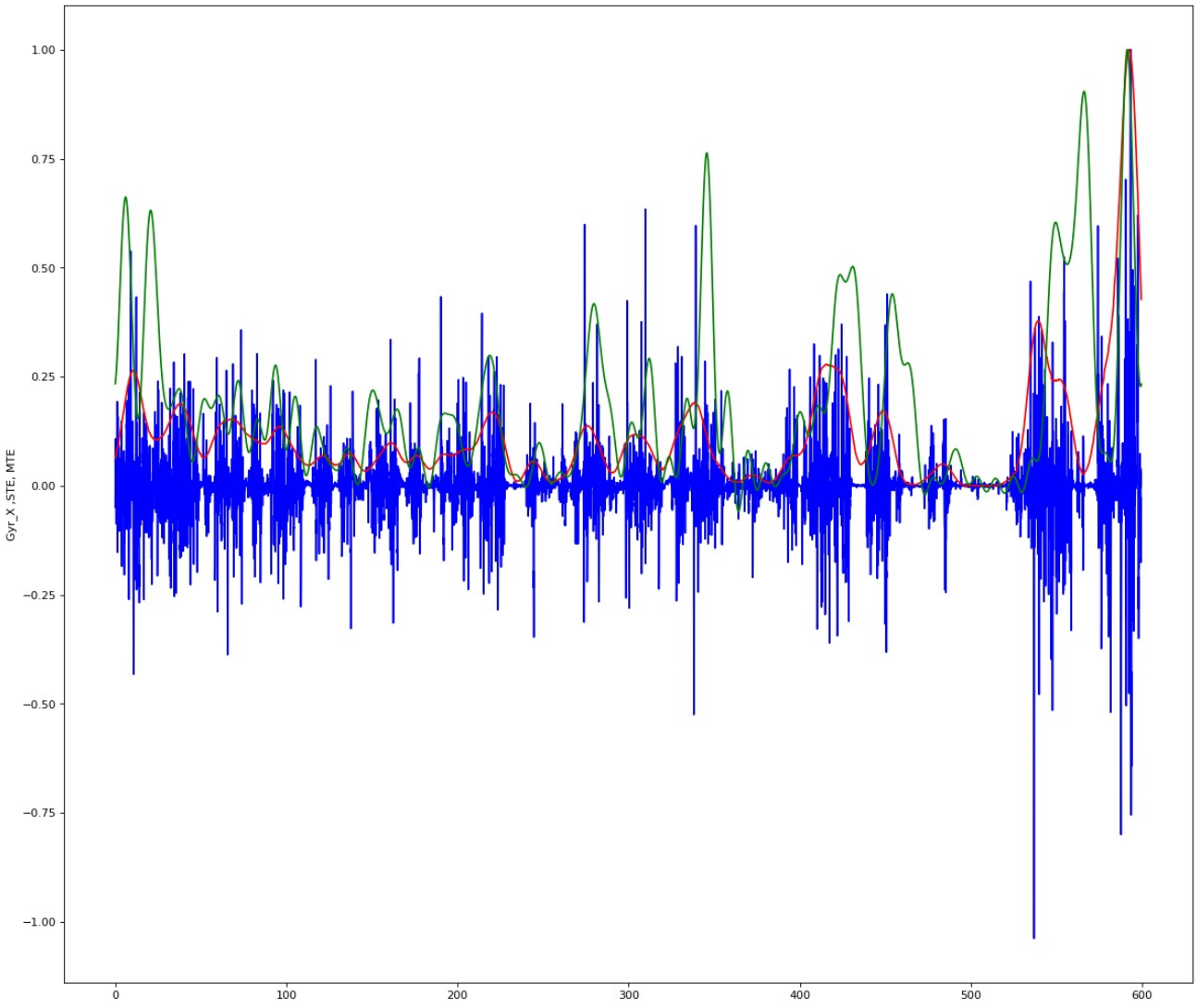
In [136]:

```
import scipy
from scipy import signal
ACC_X=scipy.signal.resample(Mean_Teager(accx,20), 11992)
GYR_X=scipy.signal.resample(Mean_Teager(gyrx,20), 11992)
HRM=scipy.signal.resample(Mean_Teager(hrm,5), 2998)
#Short time energy compute
ste_accx=ste(accx,20)
ste_gyrx=ste(gyrx,20)
ste_hrm=ste(hrm,5)
#let's plot
#plot for accx
fig=plt.figure(figsize=(18, 16), dpi= 80, facecolor='w', edgecolor='k')
plt.plot(t20, accx/max(accx), color = 'blue') #το μαξ για κανονικοποιήση
plt.plot(t20, ste_accx/max(ste_accx), color = 'red')
plt.plot(t20, ACC_X/max(ACC_X), color = 'green')
plt.ylabel('Acc_X ,STE, MTE')
plt.show()
```



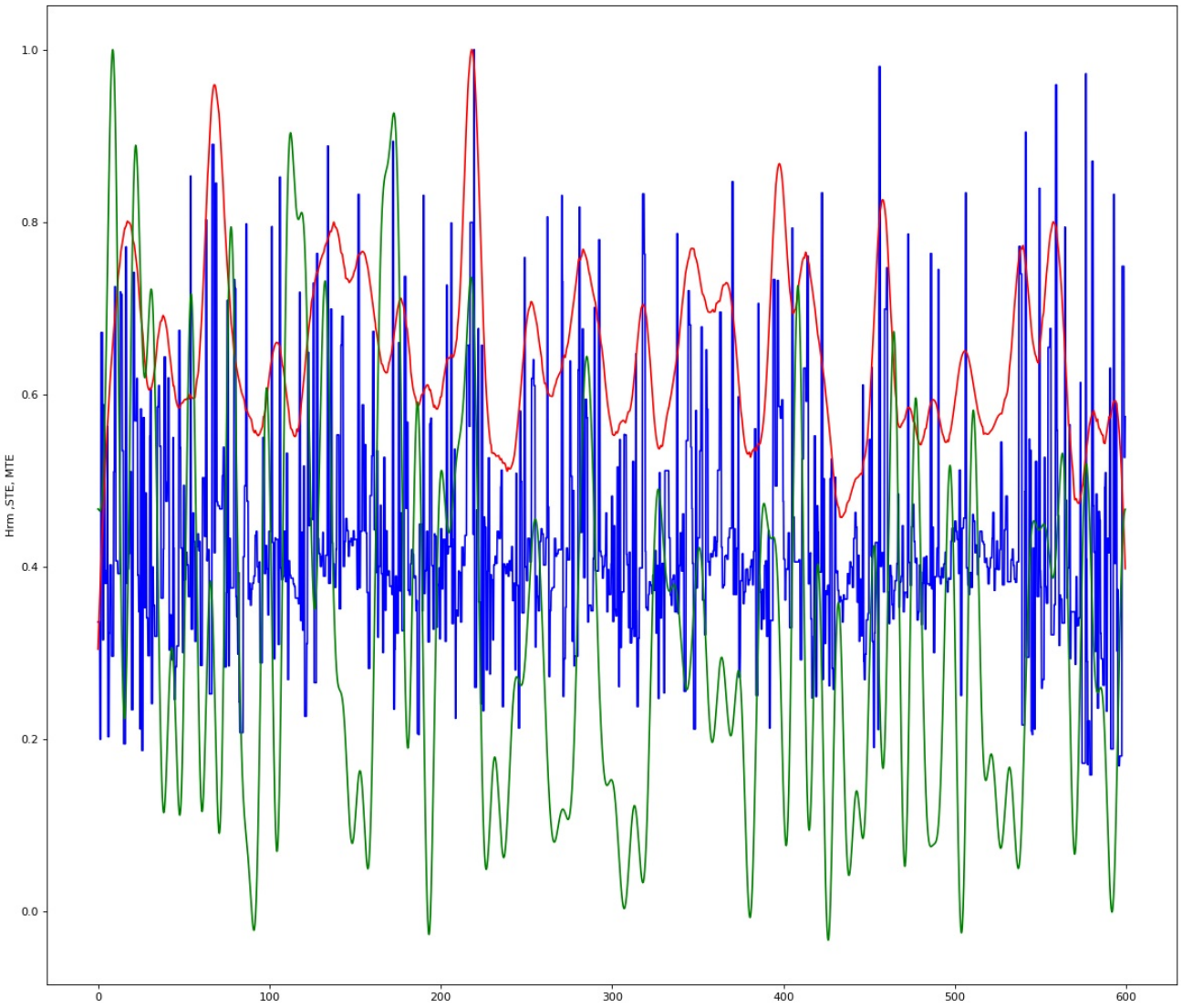
In [137]:

```
#plot for gyrx
fig=plt.figure(figsize=(18, 16), dpi= 80, facecolor='w', edgecolor='k')
plt.plot(t20, gyrx/max(gyrx), color = 'blue')
plt.plot(t20, ste_gyrx/max(ste_gyrx), color = 'red')
plt.plot(t20, GYR_X/max(GYR_X), color = 'green')
plt.ylabel('Gyr_X ,STE, MTE')
plt.show()
```



In [138]:

```
#plot for hrm
fig=plt.figure(figsize=(18, 16), dpi= 80, facecolor='w', edgecolor='k')
plt.plot(t5, hrm/max(hrm), color = 'blue')
plt.plot(t5, ste_hrm/max(ste_hrm), color = 'red')
plt.plot(t5, HRM/max(HRM), color = 'green')
plt.ylabel('Hrm ,STE, MTE')
plt.show()
```



Τέλος 2ης εργαστηριακής

In [ ]:

```

```