Ζητούμενα:

1)α)Σύμφωνα με την θεωρία για να επιτύχουμε λογαριθμική απεικόνιση του πλάτους των δοθέντων μετασχηματισμών θα χρησιμοποιήσουμε την σχέση

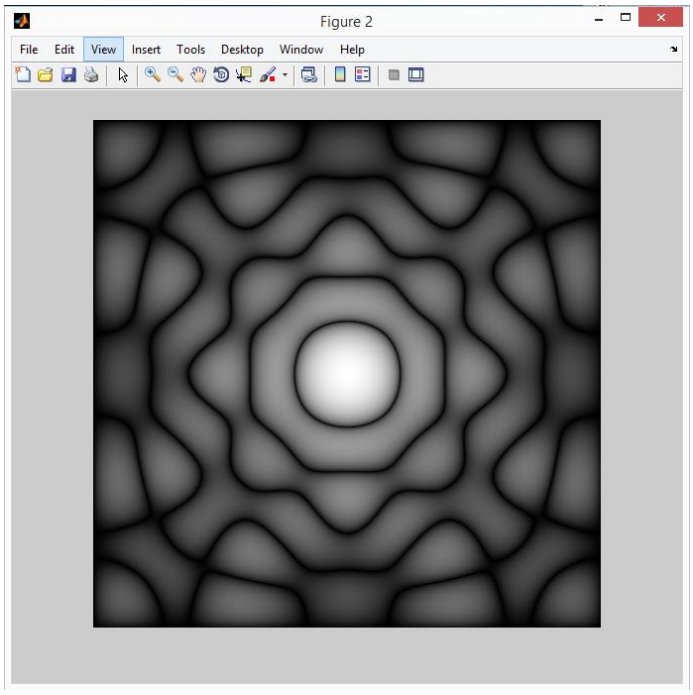
log|F(u,v)|+1, όπου F(u,v) κάθε pixel του παραπάνω μετασχηματισμού. Μετά την εφαρμογή της παραπάνω σχέσης καλούμε την συνάρτηση **erwthma1.m** από την πρώτη άσκηση για να εφαρμόσουμε το γραμμικό σχηματισμό και να θέσουμε εύρος τιμών [0,255]. Τέλος για να βρίσκεται το χωρικό σημείο (0,0) στο κέντρο καλούμε την συνάρτηση **`fftshift.`** Για να δούμε τι απεικονίζει η αρχική εικόνα εφαρμόζουμε τον αντίστροφο μετασχηματισμό και σε ότι προκύψει χρησιμοποιούμε τον γραμμικό μετασχηματισμό. Προκειμένου να υλοποιήσουμε τα παραπάνω δημιουργούμε την συνάρτηση **ask2_erwthma1.m** με τις εξής εντολές:

```
function [] = ask2_erwthma1(im)
   new=log(abs(im)+1);
   new_im=erwthma1(new,0,255);
   figure(1);
   imshow(im);
   figure(2);
   new_im=fftshift(new_im);
   imshow(uint8(new_im));
   start_image=erwthma1(ifft2(im),0,255);
   figure(3);
   imshow(uint8(start_image));
end
```

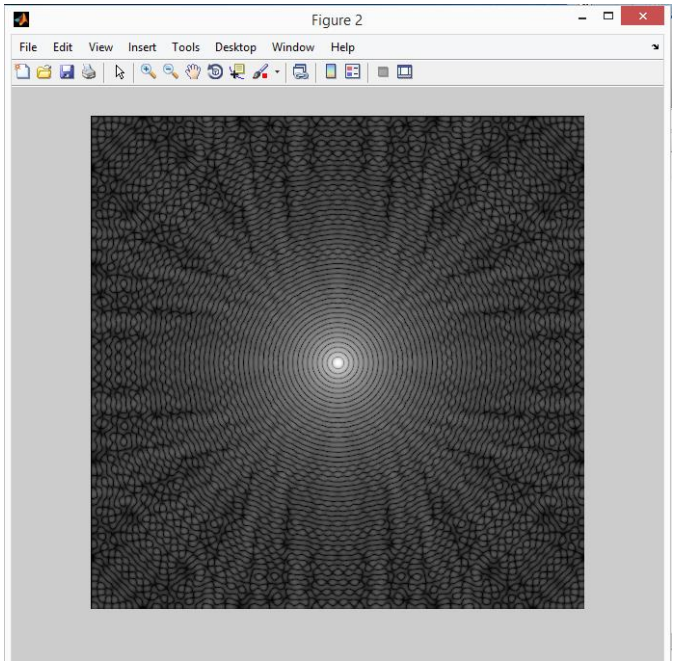
η οποία παίρνει σαν όρισμα τον μετασχηματισμό Fourier. Καλούμε την παραπάνω συνάρτηση για τους μετασχηματισμούς Fourier f1,f2,f3,f4,f5,f6 και παρακάτω έχουμε της λογαριθμική απεικόνιση για κάθε μετασχηματισμό. Για να καλέσουμε την συνάρτηση πληκτρολογούμε **ask2_erwthma1(f1);**

Λογαριθμικές απεικονίσεις για τους μετασχηματισμούς:

f1



f2



f3

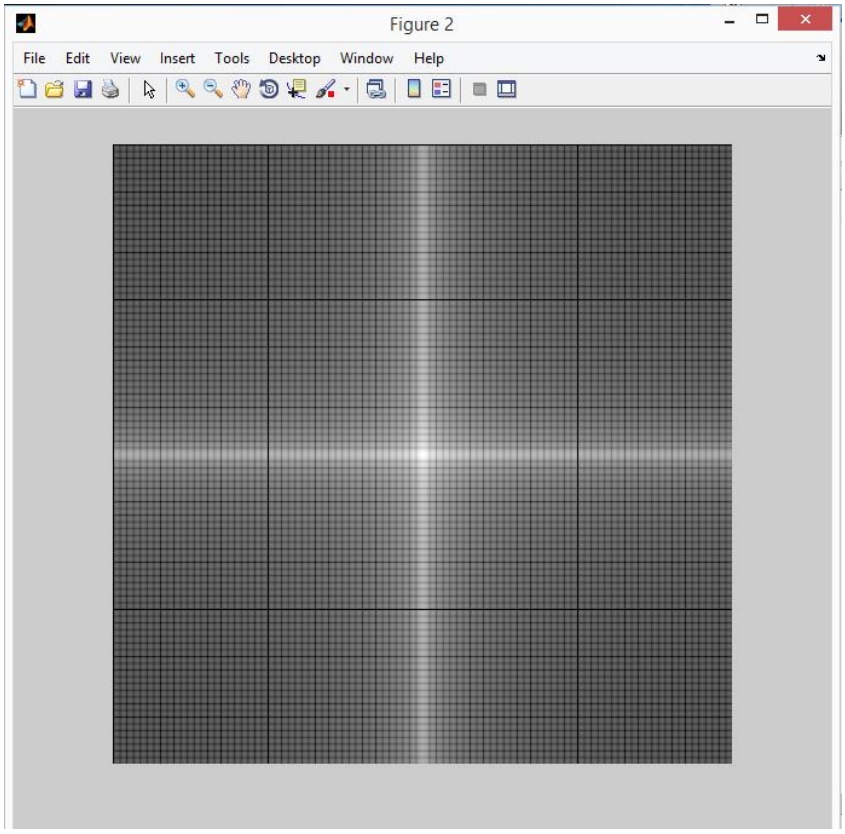


Figure 2
File Edit View Insert Tools Desktop Window Help

f4

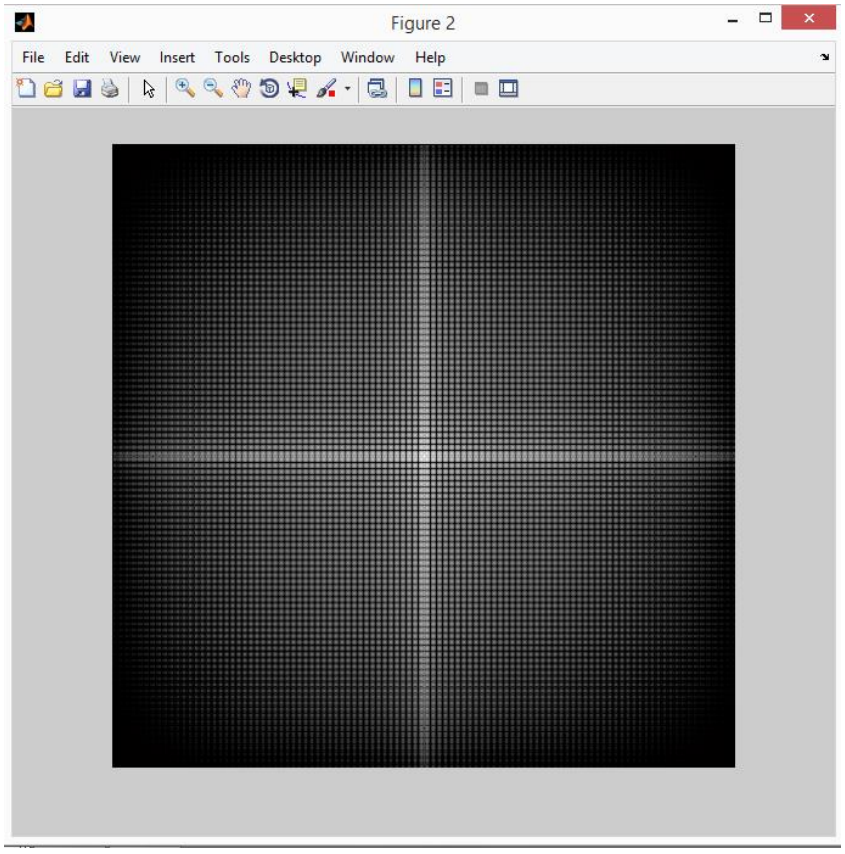
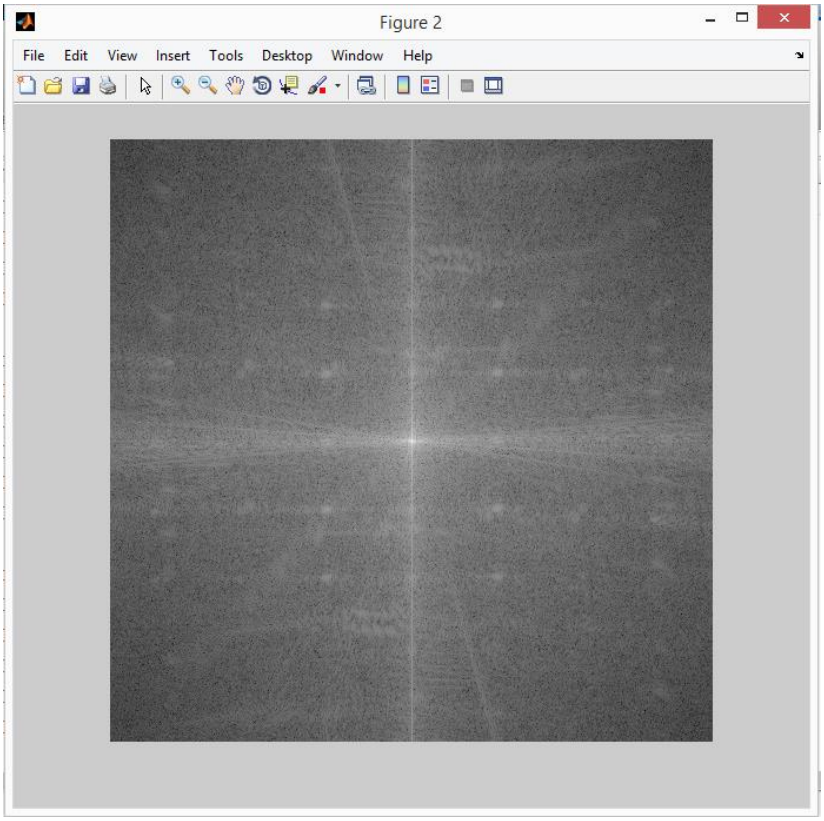
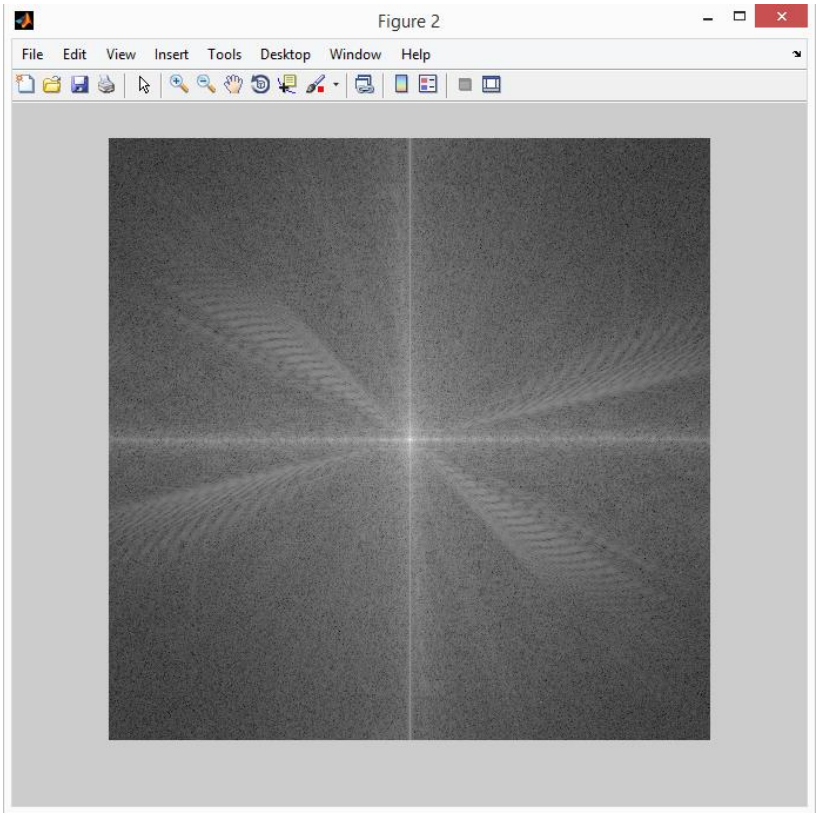


Figure 2
File Edit View Insert Tools Desktop Window Help

f5

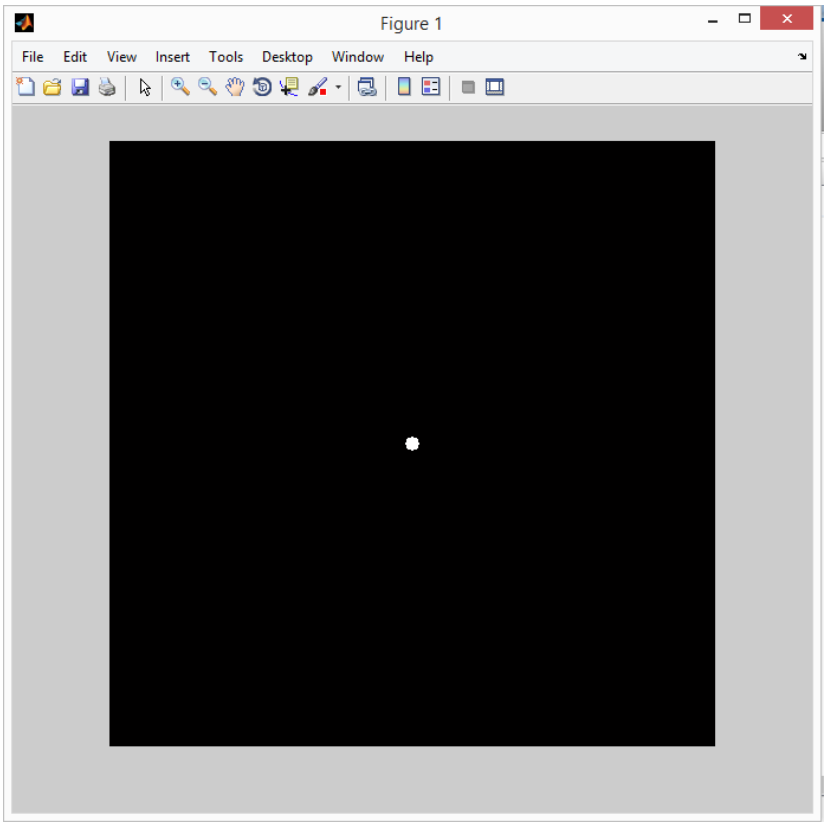


f6

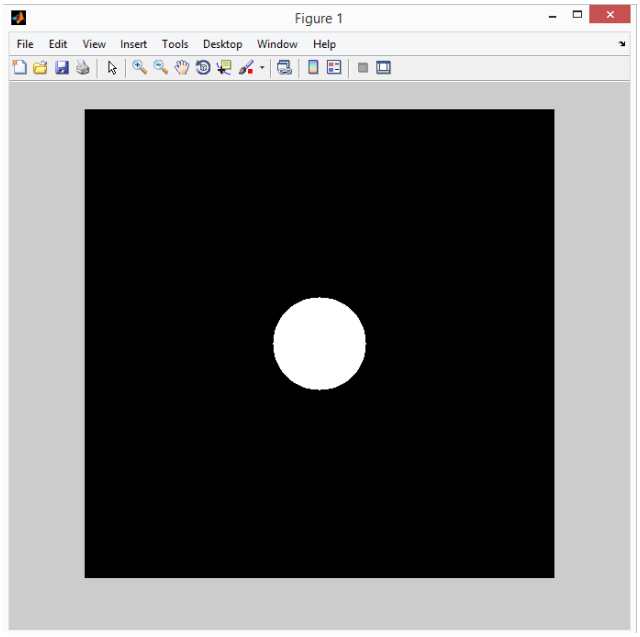


b)a)Χοντρικά χαρακτηριστικά για τις παραπάνω εικόνες και επιβεβαίωση:

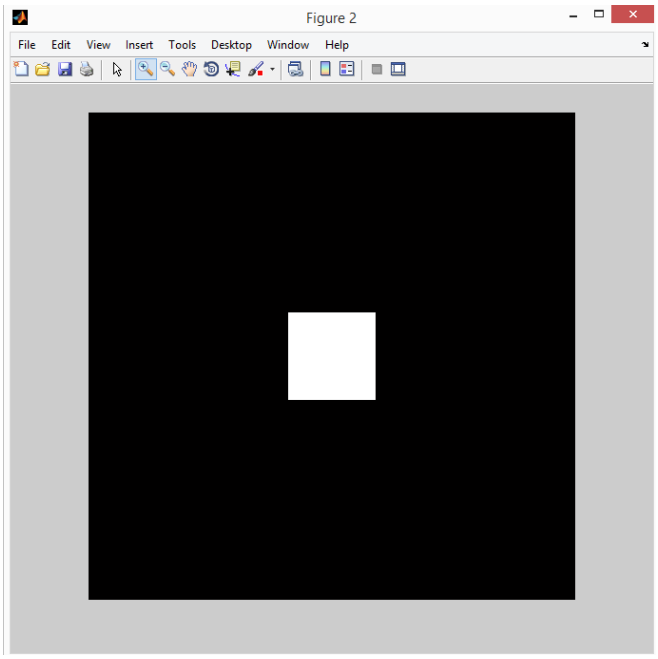
- **f1**: περιμένουμε στην αρχική εικόνα να έχουμε έναν κύκλο, ανομοιόμορφο και μικρών διαστάσεων. Αυτό, γιατί στην αντίστοιχη περίπτωση που έχουμε έναν παλμό μεγάλου εύρους η αντίστοιχη sinc συνάρτηση είναι περισσότερο «στενή» και το αντίστροφο. Επίσης ανομοιόμορφος, γιατί η λογαριθμική απεικόνιση δεν έχει κάποια συμμετρία. Για να έχουμε την αρχική εικόνα από την οποία προήλθε ο μετασχηματισμός Fourier χρησιμοποιούμε την εντολή ifft2 για να πάρουμε τον αντίστροφο μετασχηματισμό. Έτσι για να πάρουμε την αρχική εικόνα του μετασχηματισμού f1 πληκτρολογούμε **imshow(uint8(ifft2(f1)));**



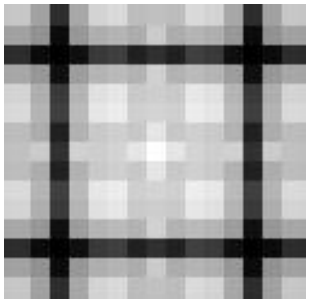
- **f2**: βγάζουμε συμπέρασμα πως έχουμε έναν ομοιόμορφο κύκλο μεγάλων όμως διαστάσεων σε σχέση με την λογαριθμική απεικόνιση. Ο κύκλος έχει μεγαλύτερες διαστάσεις για τον λόγο που περιγράψαμε στον f1 μετασχηματισμό.

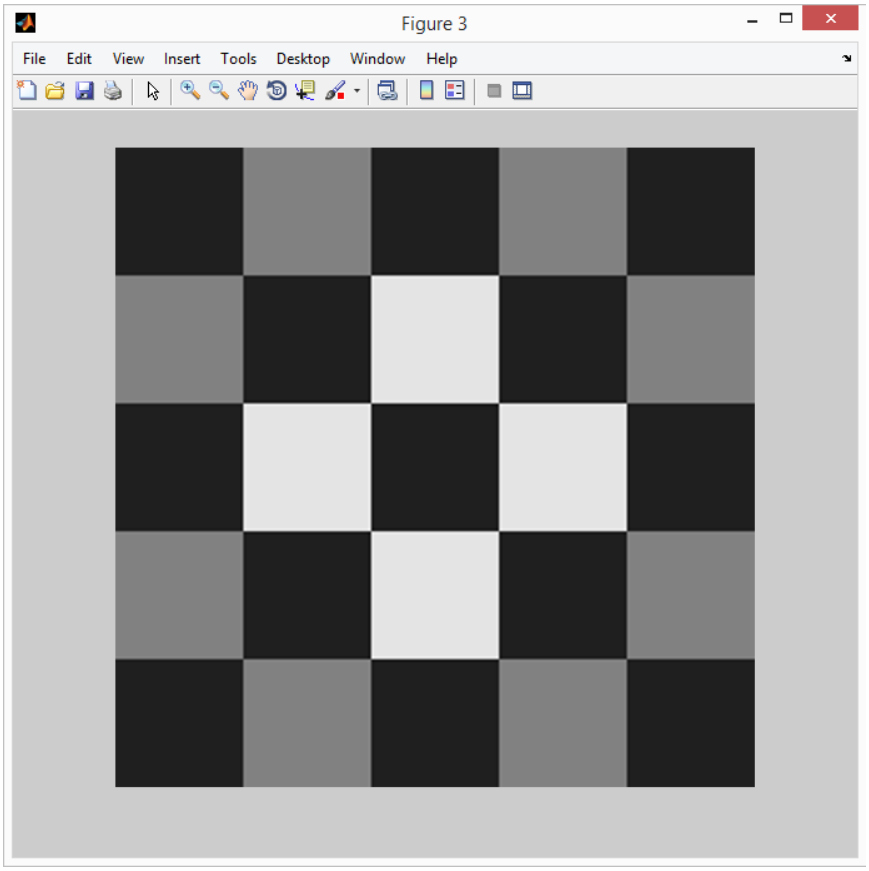


- **f3**: συμπεραίνουμε πως η αρχική μας εικόνα αναπαριστά έναν τετραγωνικό παλμό. Μεγέθους μεγαλύτερου από το τετραγωνάκι της λογαριθμικής απεικόνισης.



- **f4**: κάνοντας zoom στο κέντρο της λογαριθμικής απεικόνισης παρατηρούμε πως το κεντρικό «κελί» δεν παρουσιάζει ομοιομορφία ως προς το εσωτερικό του χρώμα που σημαίνει πως η αρχική εικόνα, δεν αποτελείται μόνο από ένα ομοιόμορφο σχήμα όπως παραπάνω. Παρατηρούμε πως σχηματίζεται κάτι σαν σταυρός στο κέντρο της λογαριθμικής απεικόνισης.
  Zoom λογαριθμικής απεικόνισης εικόνας:





- **f5**: παρατηρούμε πως έχουμε υψηλότερες συχνότητες, περιμένοντας στην αρχική εικόνα να έχουμε περισσότερες εναλλαγές και όχι τόσο πολύ ομοιόμορφες περιοχές, όπως πριν.

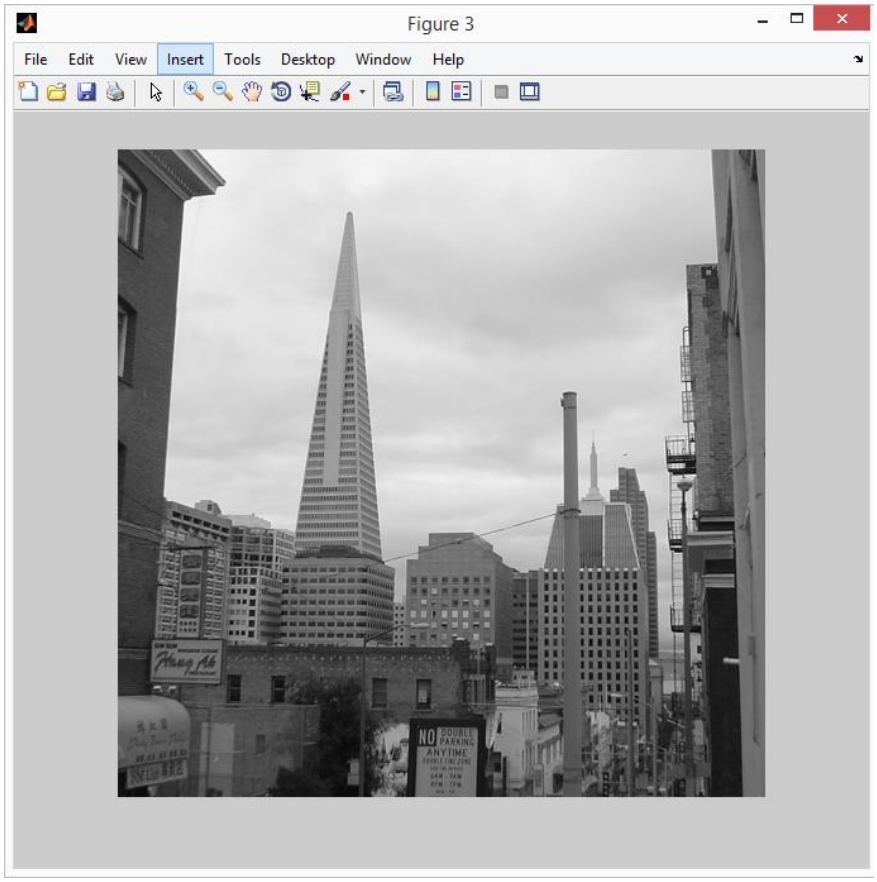
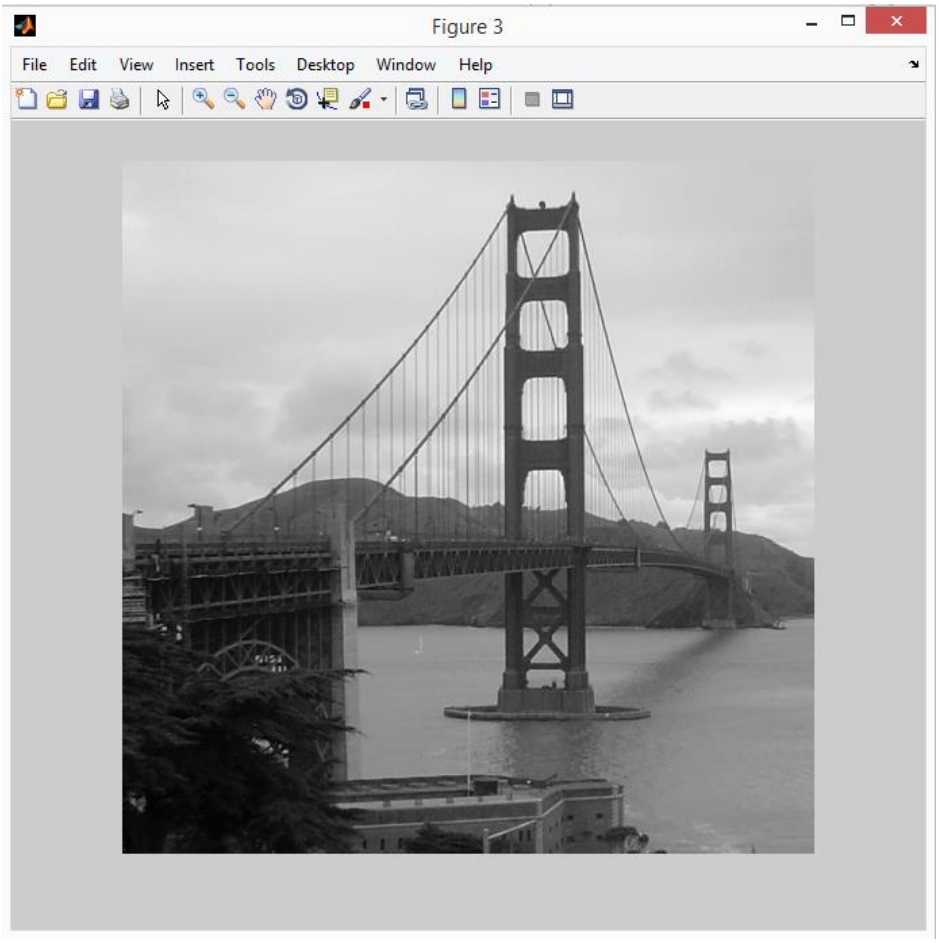


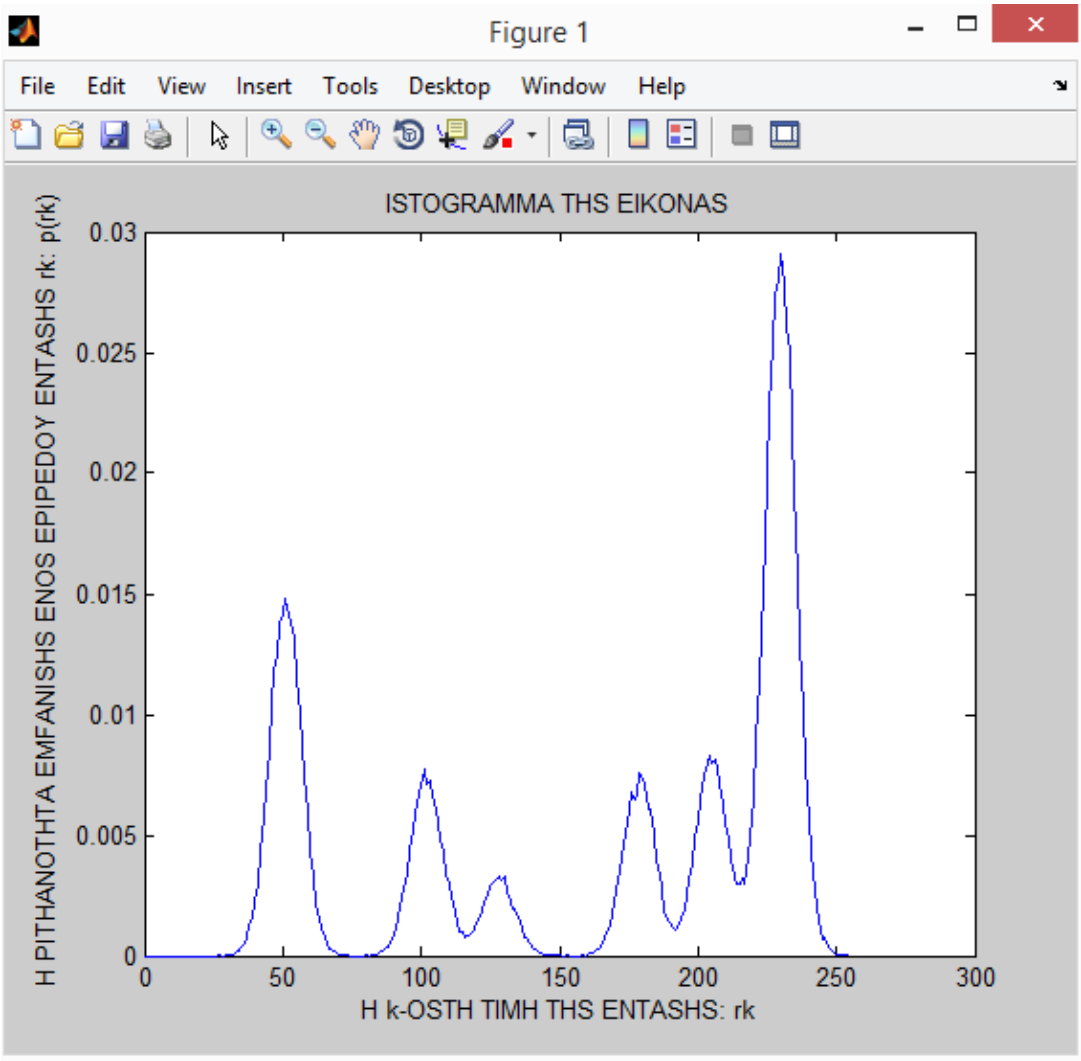
Figure 3
File Edit View Insert Tools Desktop Window Help
NO DOUBLE PARKING ANYTIME

- **f6**: παρατηρούμε πως εξίσου έχουμε υψηλότερες συχνότητες, δηλαδή λιγότερες ομαλές περιοχές, έντονες ακμές σε γωνίες συν-πλην 45^{o} και λευκές γραμμές.

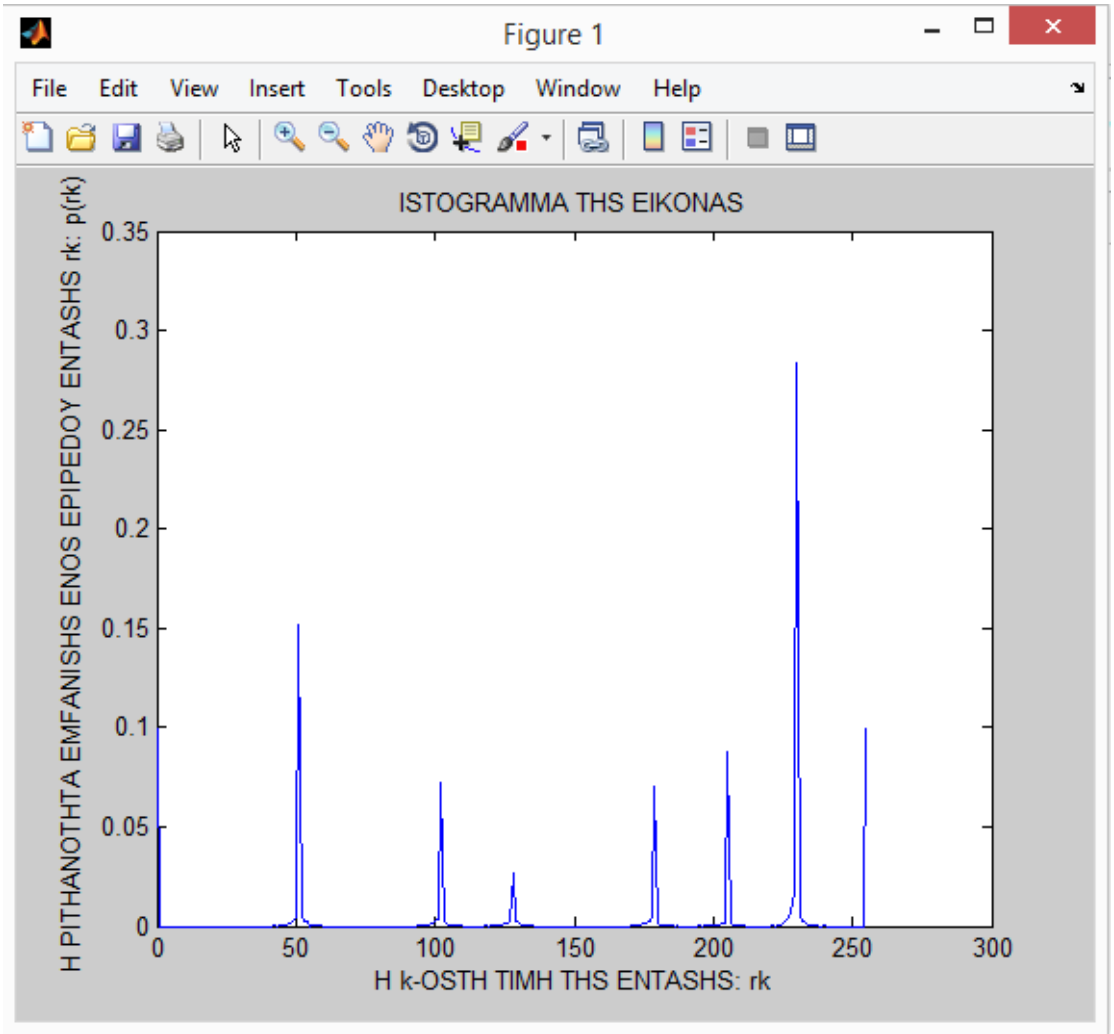


2)Για να βρούμε τον τύπο θορύβου των εικόνων noisy_image1 και noisy_image2, θα χρησιμοποιήσουμε την συνάρτηση υπολογισμού του ιστογράμματος μια εικόνας που είχαμε υλοποιήσει στην πρώτη άσκηση με όνομα erwthma2_1.m . Καλώντας για παράδειγμα την συνάρτηση για την εικόνα y1 γράφοντας erwthma2_1(y1); και για την εικόνα y2 γράφοντας erwthma2_1(y2); παίρνουμε τα παρακάτω ιστογράμματα.

Για την y1:

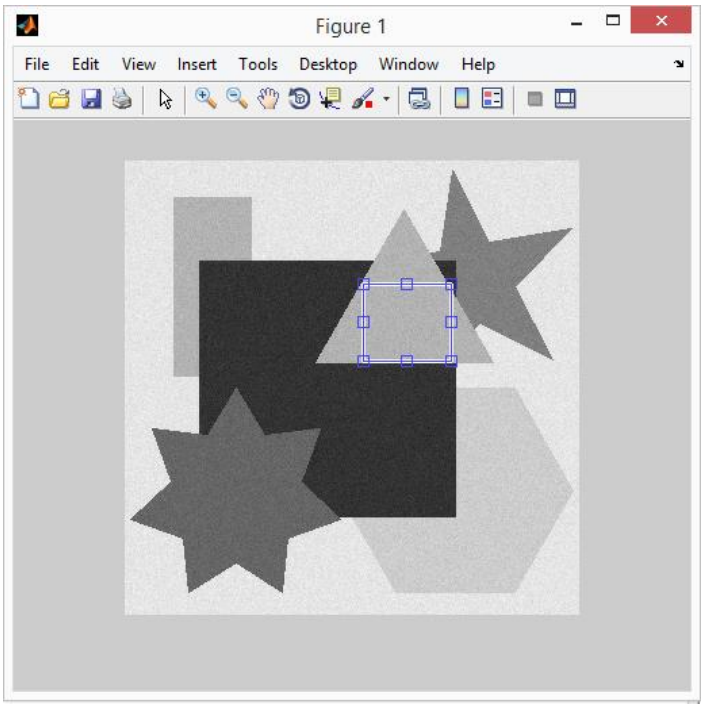


Για την y2:

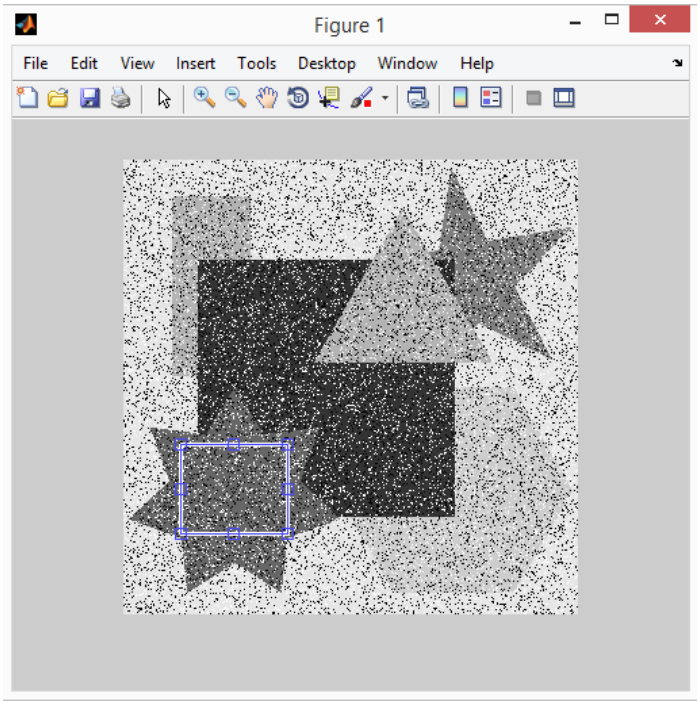


Από τα ιστογράμματα μπορούμε να συμπεράνουμε πως στην πρώτη εικόνα υπάρχει Gaussian θόρυβος και στην δεύτερη κρουστικός θόρυβος. Για την εκτίμηση όμως των παραμέτρων του θορύβου επειδή έχουμε πολλούς λοβούς είναι δύσκολο να υπολογίσουμε την μέση τιμή, την τυπική απόκλιση και την διακύμανση. Για αυτόν τον λόγο θα χρησιμοποιήσουμε την imcrop συνάρτηση με την οποία θα πάρουμε ένα αντιπροσωπευτικό δείγμα από την κάθε εικόνα, το οποίο θα έχει κάποια σχετική ισορροπία χρωμάτων σε σχέση με την συνολική εικόνα. Η παραπάνω διαδικασία γίνεται, γιατί θα μας βόλευε περισσότερο να απομονώσουμε μόνο ένα λοβό από το ιστόγραμμα. Επομένως παίρνουμε το σχετικά ισορροπημένο δείγμα μας για κάθε εικόνα και καλούμε την συνάρτηση υπολογισμού του ιστογράμματος αυτή την φορά με παράμετρο το κάθε μας δείγμα. Για να πάρουμε το δείγμα πληκτρολογούμε deigma1=imcrop(uint8(y1)); και deigma2=imcrop(uint8(y2)); , αντίστοιχα για κάθε εικόνα.

Λήψη δείγματος για την y1:

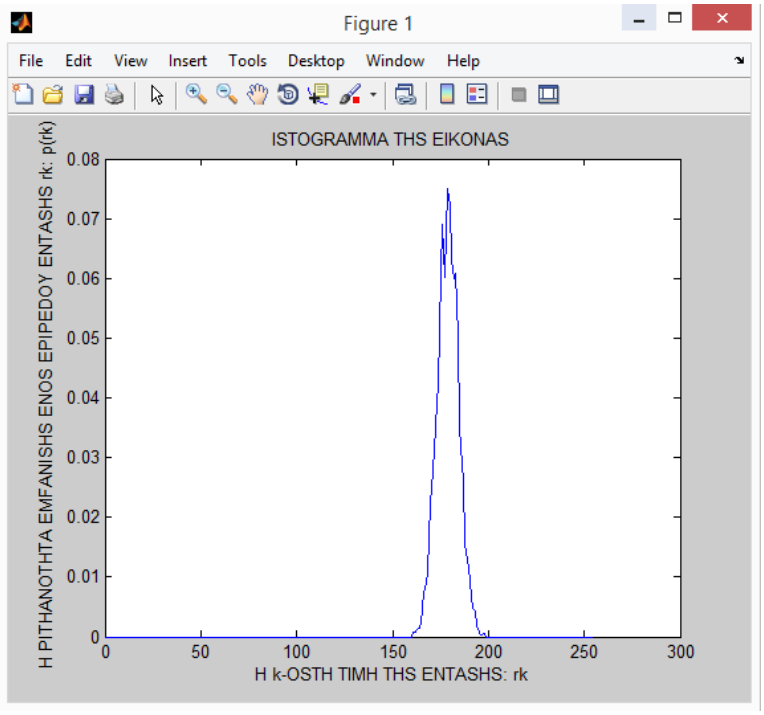


Λήψη δείγματος για την y2:

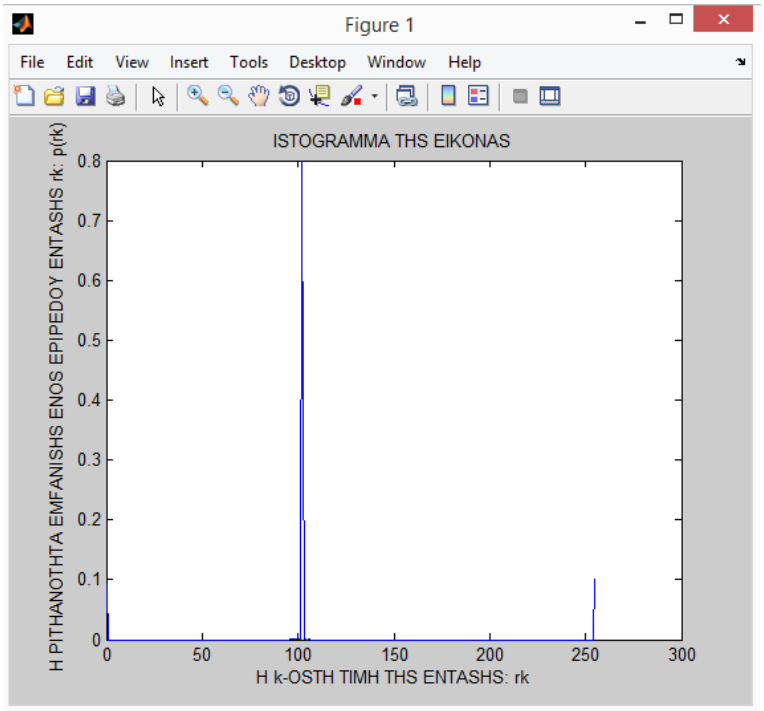


Ξανατρέχουμε την συνάρτηση υπολογισμού του ιστογράμματος για κάθε δείγμα μας πληκτρολογώντας erwthma2_1(deigma1); και erwthma2_1(deigma2); .

Ιστόγραμμα για το δείγμα 1:

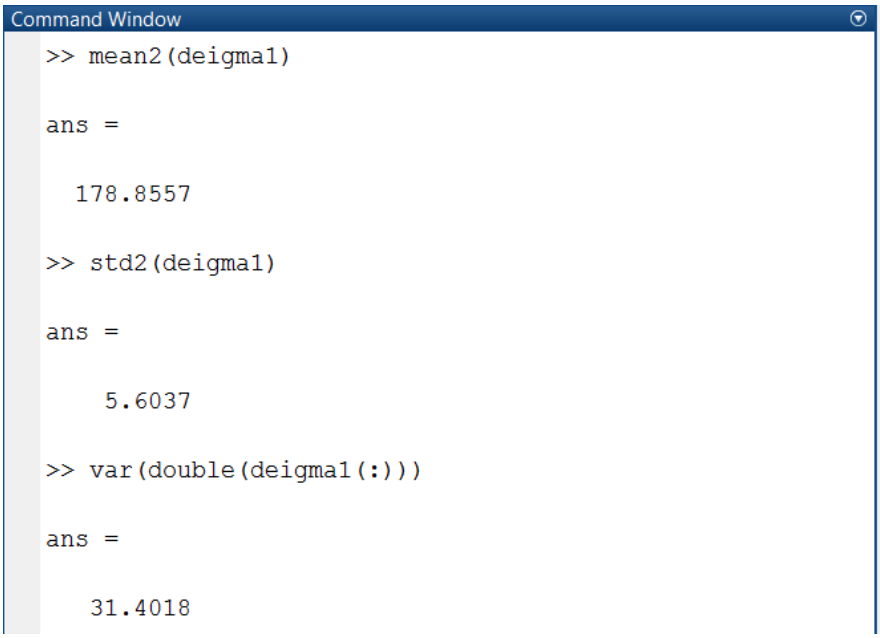


Ιστόγραμμα για το δείγμα 2:

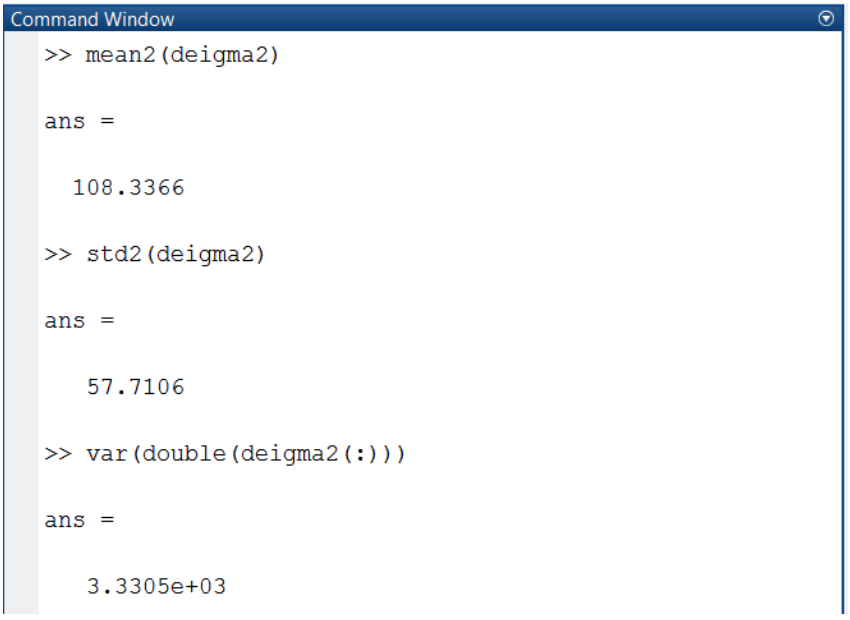


Στην συνέχεια για κάθε μας δείγμα θα υπολογίσουμε την μέση τιμή, την τυπική απόκλιση και την διακύμανση. Για να βρούμε τα παραπάνω χρησιμοποιούμε τις έτοιμες συναρτήσεις της matlab mean2,std2 και var. Παρακάτω φαίνονται οι τιμές για κάθε ένα από τα δείγματα μας.

Δείγμα 1



Δείγμα 2



3)i)Για την υλοποίηση του χαμηλοπερατού φίλτρου Butterworth υλοποιήθηκε η συνάρτηση ask2_erwthma3_1, η οποία παίρνει σαν παραμέτρους την εικόνα, την συχνότητα αποκοπής και την τάξη του φίλτρου. Εν συντομία, όσο μεγαλύτερη είναι η συχνότητα αποκοπής, τόση μεγαλύτερη είναι η ακτίνα του κύκλου στην αντίστοιχη απεικόνιση του φίλτρου, με αποτέλεσμα να κόβω λιγότερες συχνότητες και μικρή τάξη του φίλτρου μας παραπέμπει σε Gaussian φίλτρο, ενώ για μεγάλη τάξη του φίλτρου οδηγούμαστε σε ιδανικά φίλτρα. Με το Butterworth φίλτρο επιτυγχάνουμε ομαλές μεταβάσεις στο θάμπωμα και το ringing, δεν είναι ορατό λόγω των ομαλών μεταβάσεων από τις χαμηλές στις υψηλές συχνότητες. Ο κώδικας της συνάρτησης υλοποιήθηκε ακολουθώντας τα βήματα της εκφώνησης και φαίνεται παρακάτω μαζί με την επεξήγηση του.

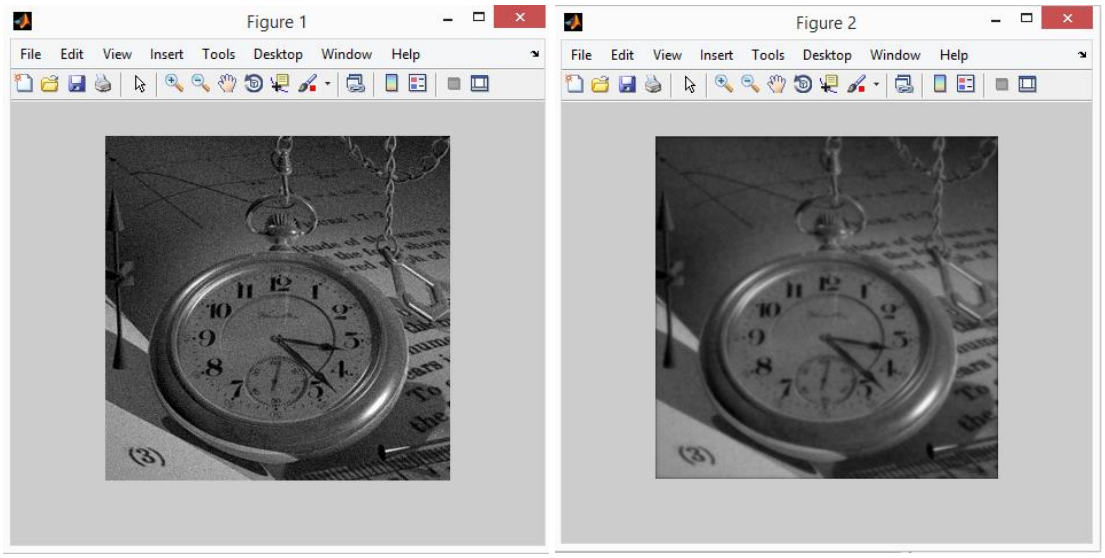
```
function [new] = ask2_erwthma3_1(im,f,n)
   fp=zeros(2*size(im,1),2*size(im,2));
   %eikona fp diastasewn 2m*2n
   for i=1:size(im,1)
      for j=1:size(im,2)
           fp(i,j)=im(i,j);
      end
   end
   %ipologismos tou estw g ginonenou ths fp me (-1)^(x+y)
   g=zeros(size(fp));
   for i=1:size(g,1)
      for j=1:size(g,2)
           g(i,j)=fp(i,j)*power(-1,(i+j));
      end
   end
   %ipologismos tou metasxhmatismou fourier ths g
   G=fft2(g);
   %dhmiourgia tou Butterworth Lowpass Filter
   %ipologismos tou D(u,v)
   D=zeros(size(fp));
   for i=1:size(D,1)
       for j=1:size(D,2)
          D(i,j)=power((power((i-
size(D,1)/2),2)+power((j-size(D,2)/2),2)),1/2);
       end
   end
   %dhmiourgia ths H
   H=zeros(size(fp));
   for i=1:size(H,1)
       for j=1:size(H,2)
           H(i,j)=1/(1+power((D(i,j)/f),2*n));
       end
   end
   %ipologismos tou S pou einai iso me to ginomeno ths G
me thn H logw tou
   %oti vriskomaste sto pedio twn sixnothtwn
   S=zeros(size(fp));
   for i=1:size(H,1)
       for j=1:size(H,2)
           S(i,j)=G(i,j)*H(i,j);
       end
   end
   %ipologismos tou s = ginomeno real(ifft2(S)) me to  (-
1)^(x+y)
   s=real(ifft2(S));
   for i=1:size(H,1)
       for j=1:size(H,2)
           s(i,j)=s(i,j)*power(-1,i+j);
       end
   end
```

```
    %kratame to kommati pou antistoixei sthn MxN eikona
dld twn arxikwn
    %diastasewn
    new=zeros(size(im));
    for i=1:size(im,1)
        for j=1:size(im,2)
            new(i,j)=s(i,j);
        end
    end
    figure(1);
    imshow(uint8(im));
    figure(2);
    imshow(uint8(new));
    figure(3);
    imshow(H);
end
```

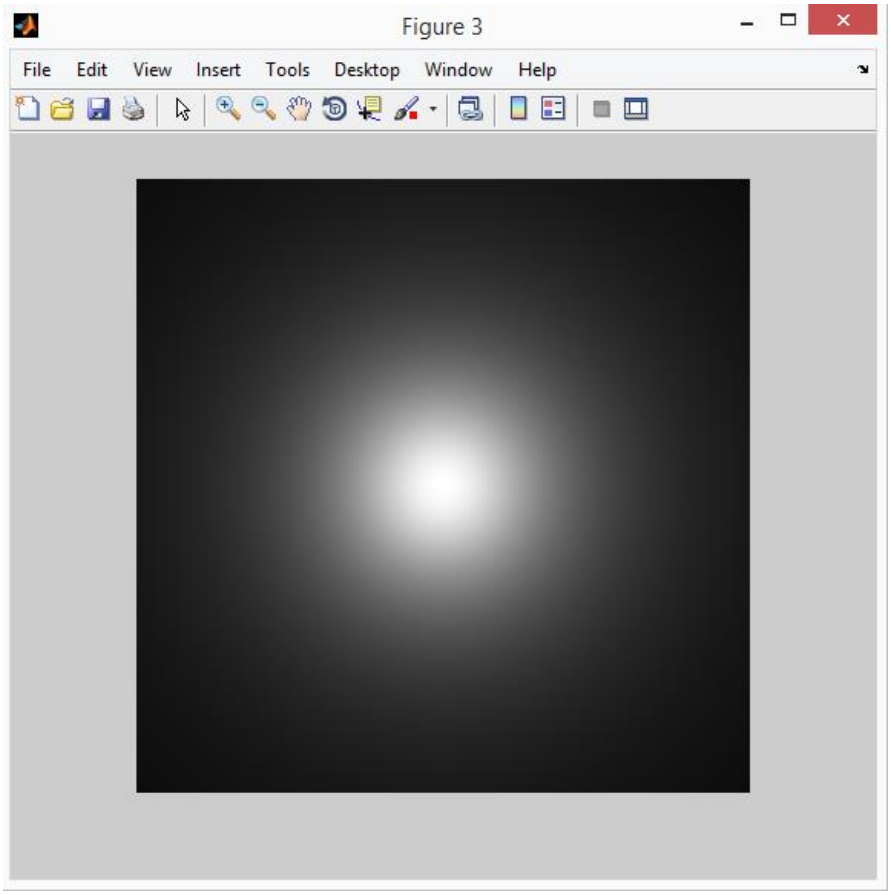
Από το τρέξιμο της συνάρτησης για τρεις διαφορετικές περιπτώσεις παίρνουμε τα εξής αποτελέσματα:

f=100, n=1 το φίλτρο μας πλησιάζει το Gaussian φίλτρο

Αφιλτράριστη Φιλτραρισμένη

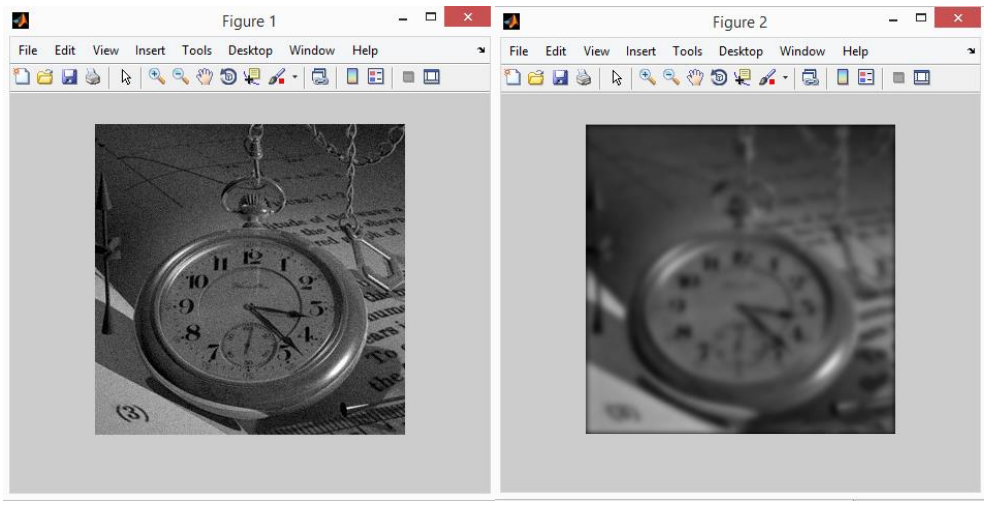


Φίλτρο

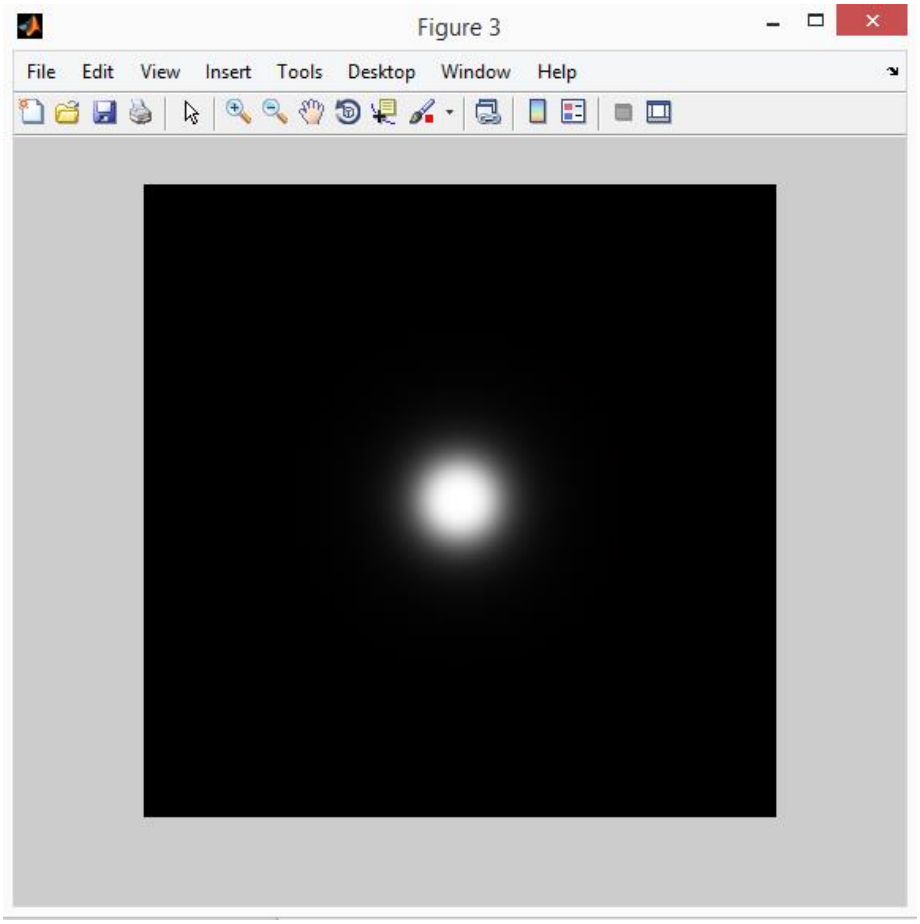


f=40, n=2 το φίλτρο είναι ανάμεσα στο ιδανικό και στο Gaussian φίλτρο

Αφιλτράριστη Φιλτραρισμένη

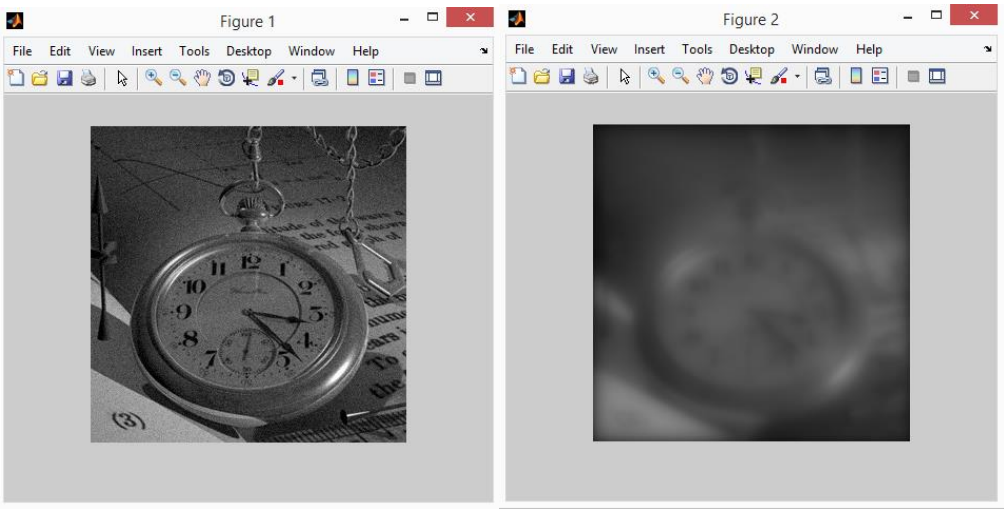


Φίλτρο

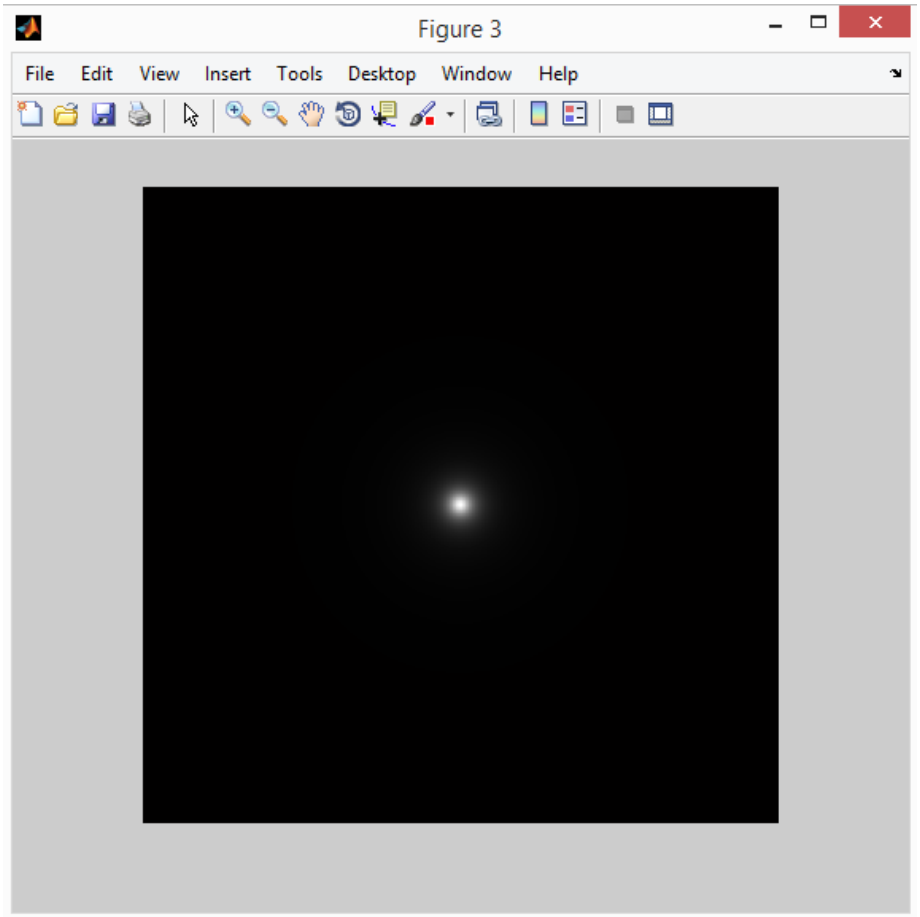


f=10, n=1 το φίλτρο είναι σχεδόν ιδανικό

Αφιλτράριστη Φιλτραρισμένη



Φίλτρο



Παρατηρούμε ότι όσο το φίλτρο μας πλησιάζει το ιδανικό η εικόνα μας τόσο περισσότερο θολώνει λόγω της απότομης κλίσης που έχει το ιδανικό φίλτρο. Μια απότομη μετάβαση ιδίως στις ακμές τις φωτογραφίας την θολώνει κατά πολύ.

ii)Καλύτερο αποτέλεσμα έχουμε για τάξη του φίλτρου n=1 και f=100, για την οποία έχουμε Gaussian φίλτρο. Για να κάνουμε την όξυνση με φίλτρο δεύτερης παραγώγου Laplace υλοποιήθηκε η συνάρτηση με όνομα ask2_erwthma3_2 η οποία παίρνει σαν είσοδο την αφιλτράριστη εικόνα και επιστρέφει την φιλτραρισμένη. Ο κώδικας της συνάρτησης φαίνεται παρακάτω.

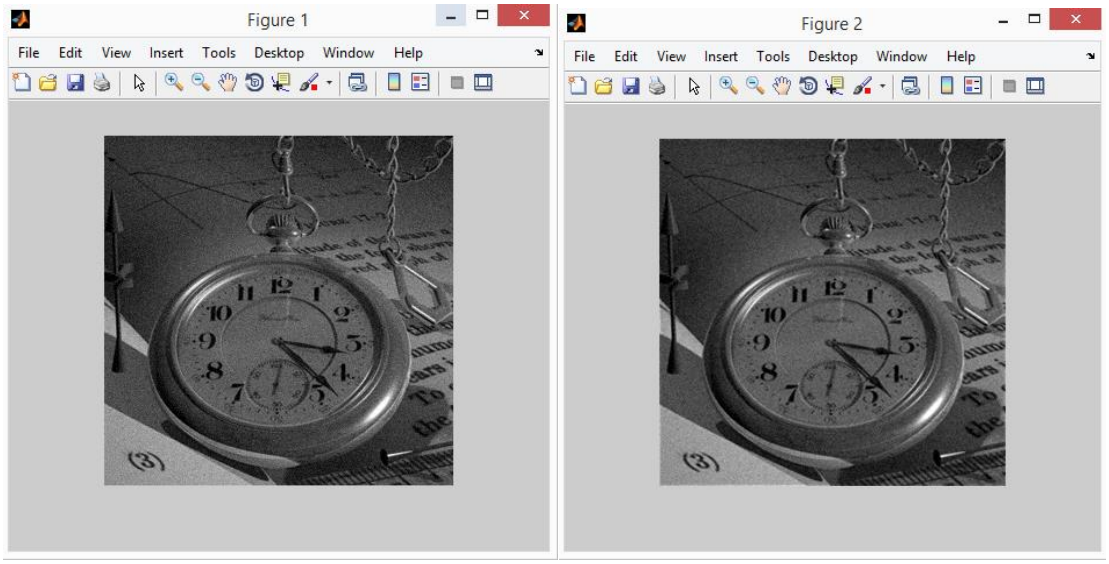
```
function [im] = ask2_erwthma3_2(image)
image=double(image);
figure(1);
imshow(uint8(image));
laplace=[0 1 0;1 -4 1;0 1 0];
%laplace=[1 1 1;1 -8 1;1 1 1];
image_new = conv2(image,laplace,'same');
im=zeros(size(image_new));
for i=1:size(image_new,1)
    for j=1:size(image_new,2)
```

```
            im(i,j)=image(i,j)-image_new(i,j);
        end
end
figure(2);
imshow(uint8(im));
end
```

Με το συγκεκριμένο φίλτρο που χρησιμοποιήσαμε πρέπει να αφαιρέσουμε τις τιμές της φιλτραρισμένης εικόνας από την αρχική.

Παρακάτω φαίνεται η εικόνα της πρώτης περίπτωσης με **f=100**,**n=1**, αφού έγινε όξυνση σε αυτή.

Αρχική εικόνα δεδομένων Φιλτραρισμένη

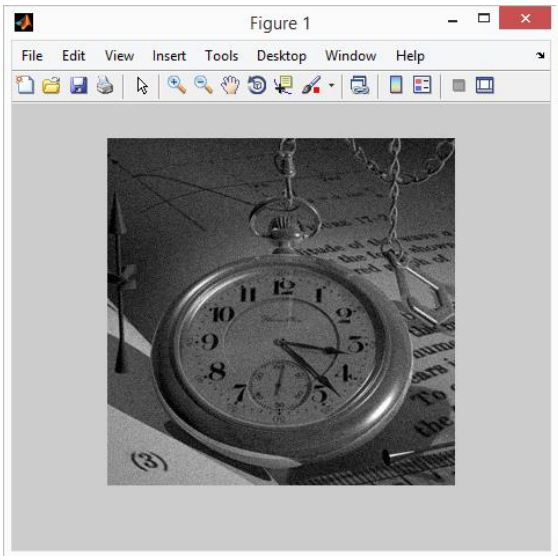


Δεξιά φαίνεται η φιλτραρισμένη εικόνα με το Butterworth φίλτρο για f=100,n=1 και το φίλτρο δεύτερης παραγώγου Laplace για την όξυνση.

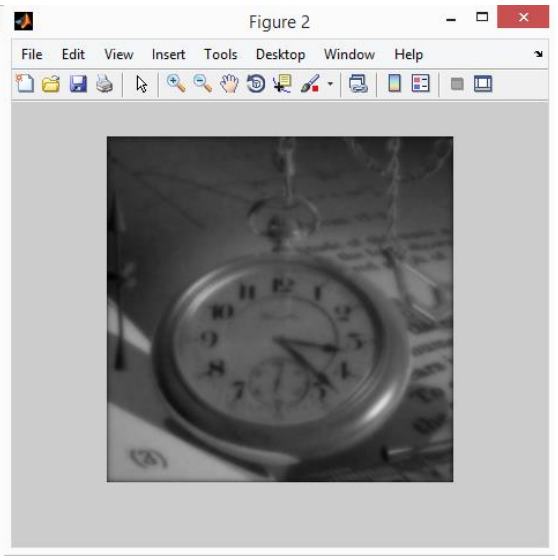
Δεν παρατηρούμε μεγάλες διαφορές ανάμεσα στις δύο εικόνες. Μεγαλύτερη διαφορά μπορούμε να διαπιστώσουμε με το μάτι στο κάτω αριστερά μέρος την εικόνας το οποίο είναι πιο ανοιχτόχρωμο παρατηρώντας λιγότερο θόρυβο σε σχέση με την πρώτη. Θα προσπαθήσουμε να εφαρμόσουμε το φίλτρο Butterworth, με f=50,n=1, όπου για f=50 θα καταφέρουμε να κόψουμε περισσότερες υψηλές συχνότητες με σκοπό να κόψουμε περισσότερο θόρυβο. Αναμένουμε όμως η αύξηση της συχνότητας να μας οδηγήσει σε πιο θολή εικόνα.

Παράμετροι για το φίλτρο Butterworth **f=50, n=1**

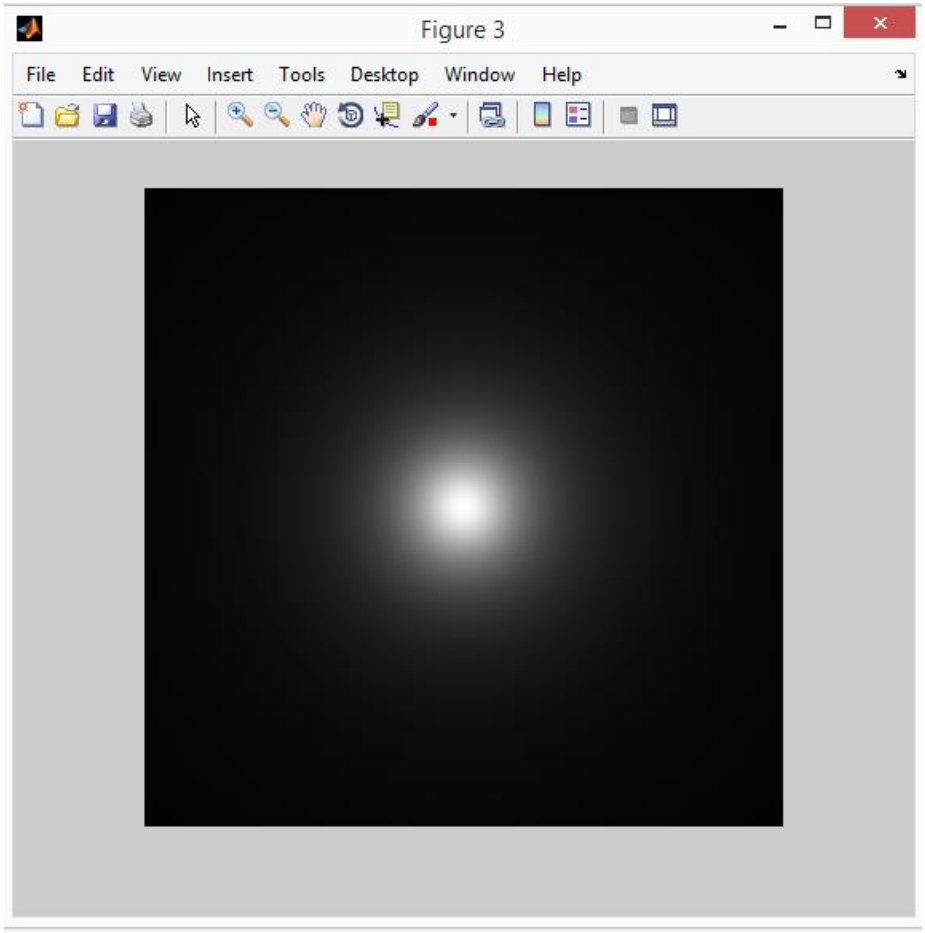
Αφιλτράριστη



Φιλτραρισμένη

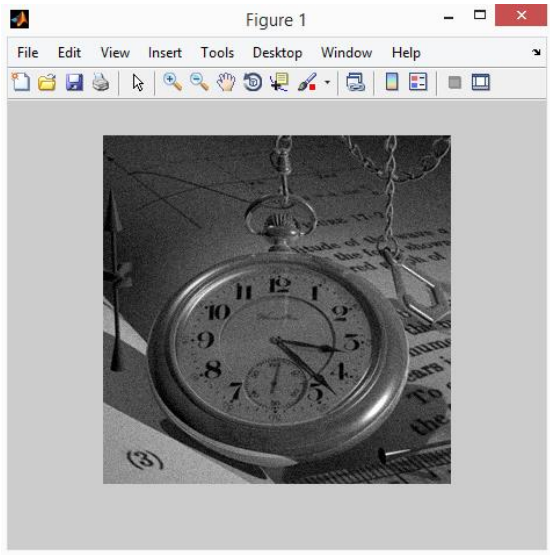


Φίλτρο

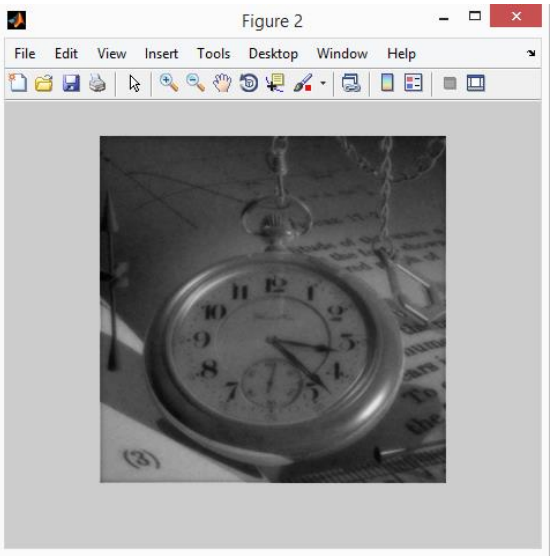


Παρακάτω φαίνεται η εικόνα που προέκυψε αφού έγινε όξυνσης με το φίλτρο δεύτερης παραγώγου Laplace:

Αρχική εικόνα δεδομένων



Φιλτραρισμένη



Παρατηρούμε πως μικραίνοντας την συχνότητα από f=100 σε f=50 για την ίδια τάξη του φίλτρου n=1, δηλαδή κόβοντας περισσότερες υψηλές συχνότητες από την μια έχω πιο καθαρή εικόνα σε σχέση με τον θόρυβο και από την άλλη έχω πιο θολή εικόνα, γιατί με το να κόψω υψηλότερες συχνότητες χαλάω τις ακμές τις εικόνας.

Έτσι βγάζουμε συμπέρασμα πως θέλουμε το φίλτρο μας να έχει τέτοιες παραμέτρους, ώστε να πλησιάζει στο Gaussian φίλτρο με μια σχετικά μικρή συχνότητα, η οποία να αφαιρεί ικανοποιητικά τον θόρυβο χωρίς να μας χαλάει κατά πολύ την ποιότητα της εικόνας.

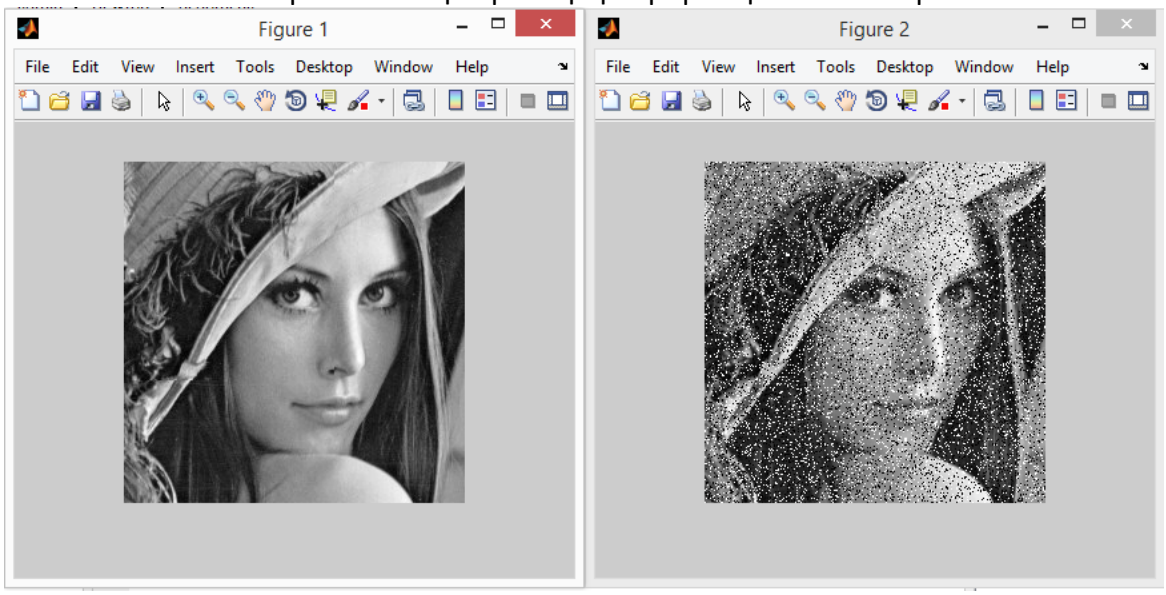
4)Για την προσθήκη θορύβου salt and pepper δημιουργήθηκε η συνάρτηση **ask2_erwthma4.m**, ο κώδικας της οποίας φαίνεται παρακάτω.

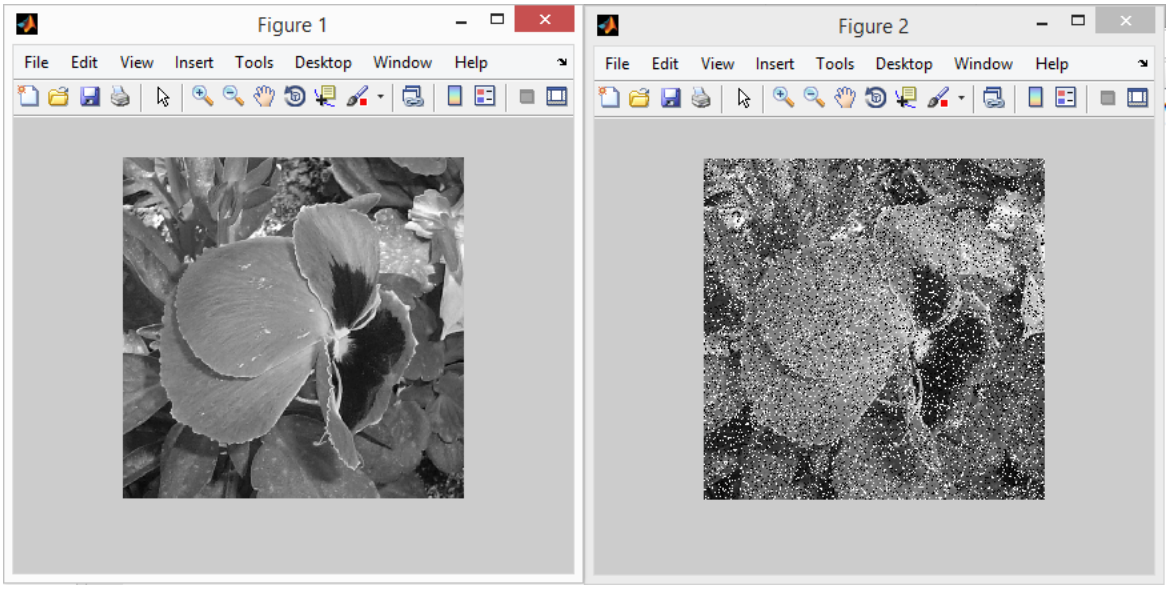
```
function [image] = ask2_erwthma4(image)
y=zeros(size(image));
for i=1:size(y,1)
    for j=1:size(y,2)
       y(i,j)=rand();
    end
end
figure(1);
imshow(uint8(image));
for i=1:size(y,1)
    for j=1:size(y,2)
       if(y(i,j)<0.1)
           image(i,j)=0;
       elseif(y(i,j)>0.9)
```

```
            image(i,j)=255;
        end
    end
end
figure(2);
imshow(uint8(image));
end
```

Αρχικά γίνεται αρχικοποίηση του πίνακα y ίδιων διαστάσεων με την εικόνα, στον οποίο εκχωρώ τυχαίες τιμές μέσω της rand() η οποία ακολουθεί την ομοιόμορφη κατανομή δίνοντας double τιμές από 0 έως 1. Οι δύο πίνακες y και image είναι παράλληλοι, κελί προς κελί και άρα pixel προς pixel. Κάνοντας προσπέλαση τον πίνακα y με τις τυχαίες τιμές αν το στοιχείο έχει τιμή μικρότερη του 0.1 τότε το αντίστοιχο pixel της εικόνας το κάνουμε 0, αλλιώς αν έχει τιμή μεγαλύτερη του 0.9 το κάνουμε 255. Λόγω της ομοιόμορφης κατανομής που ακολουθεί η rand() με τα διαστήματα να έχουν εύρος 0.1 το καθένα, θα έχουμε καταφέρει να προσθέσουμε θόρυβο στο 20% των pixels της εικόνας. Το 10% του θορύβου θα είναι μαύρα και το άλλο 10% θα είναι άσπρα pixels. Τρέχουμε την παραπάνω συνάρτηση για τις εικόνες Lenna και panses, πληκτρολογώντας noisy_lenna=ask2_erwthma4(x_le); και noisy_panses=ask2_erwthma4(pan); ,αντίστοιχα.

Τα αποτελέσματα από την προσθήκη θορύβου φαίνονται παρακάτω:





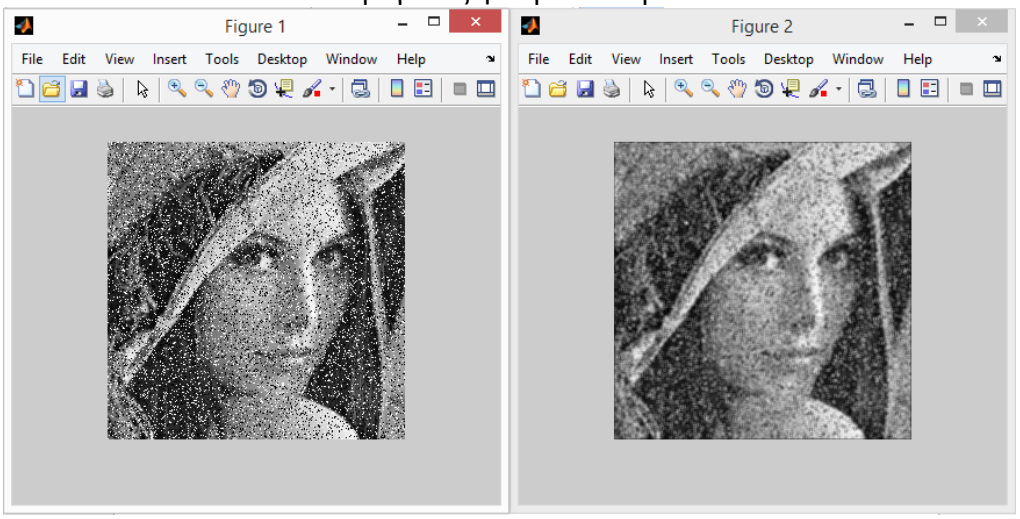
Αποκατάσταση των εικόνων

α)Για την υλοποίηση του φίλτρου μέσου όρου υλοποιήθηκε η συνάρτηση **ask2_erwthma4_average.m** η οποία εκτελεί τον κώδικα που φαίνεται παρακάτω.

```
function [] = ask2_erwthma4_average(image,filter_size)
    filter=ones(filter_size);
    filter=filter/(filter_size*filter_size);
    figure(1);
    imshow(uint8(image));
    image_new = conv2(image,filter,'same');
    figure(2);
    imshow(uint8(image_new));
end
```

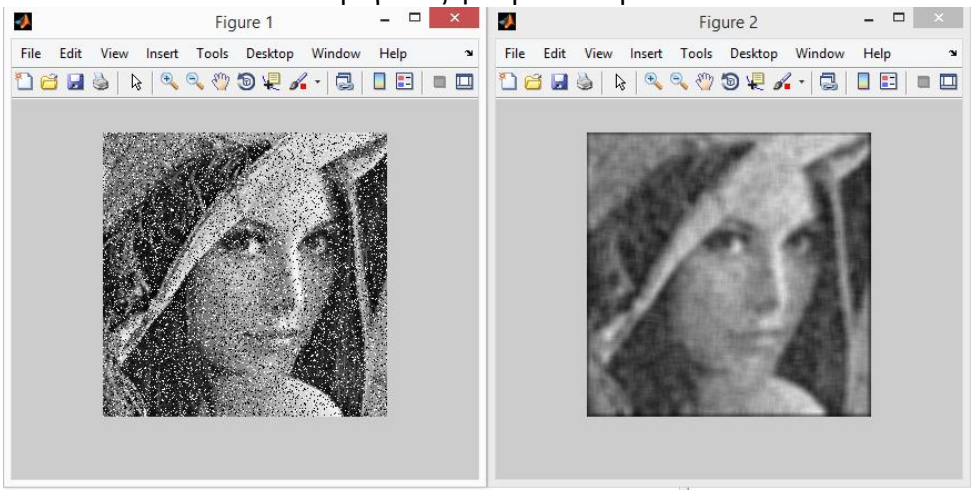
Η συνάρτηση παίρνει σαν είσοδο την αφιλτράριστη εικόνα και το μέγεθος του φίλτρου μέσου όρου που θα χρησιμοποιήσουμε σε αυτή. Δημιουργούμε το φίλτρο μας διαστάσεων filter_size*filter_size αρχικοποιούμε με 1 και διαιρούμε ως προς το πλήθος των pixels του φίλτρου. Με αυτό τον τρόπο κάθε pixel του φίλτρου έχει το ίδιο βάρος. Τέλος παίρνουμε την συνέλιξη του φίλτρου με την αρχική αφιλτράριστη εικόνα μας και εμφανίζουμε το αποτέλεσμα στην οθόνη. Καλούμε την συνάρτηση πληκτρολογώντας **ask2_erwthma4_average(noisy_lenna,3); ask2_erwthma4_average(noisy_panses,3);**, όπου το φίλτρο μας θα έχει μέγεθος 3*3.
Για φίλτρο μεγέθους 8*8 γράφουμε **ask2_erwthma4_average(noisy_lenna,8); ask2_erwthma4_average(noisy_panses,8);**, αντίστοιχα.

Αποτελέσματα φίλτρου μέσου όρου:
Για μέγεθος φίλτρου ίσο με 3:

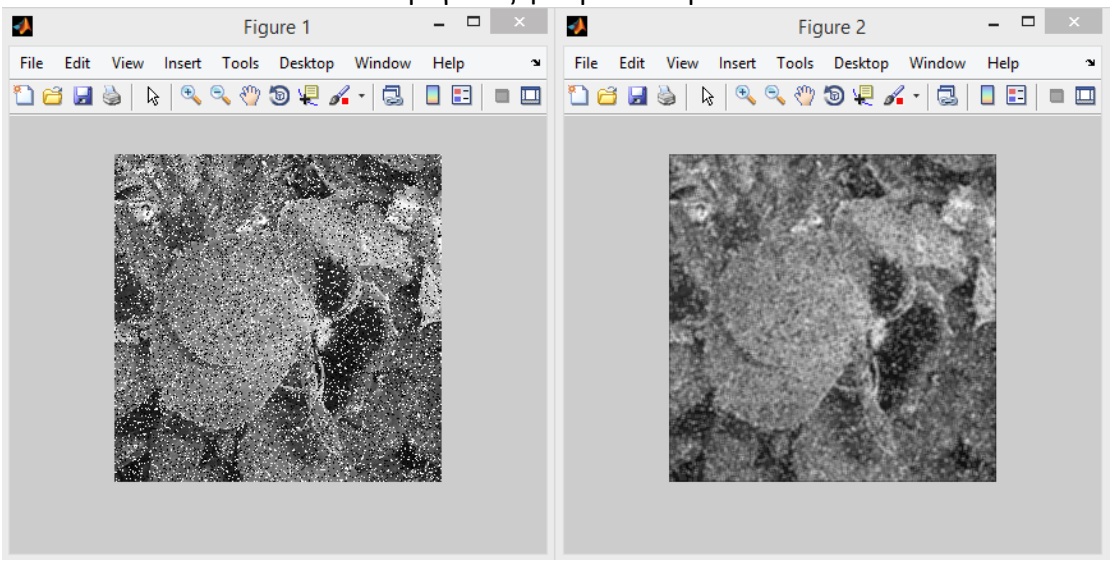


Παρατηρούμε πως η φιλτραρισμένη εικόνα δεν παρουσιάζει και ιδιαίτερη βελτίωση σε σχέση με την αφιλτράριστη λόγω του ότι για τον υπολογισμό κάθε νέου pixel λαμβάνεται ο μέσος όρος από 9 pixel της παλιάς εικόνας. Αν και πολλά pixels θορύβου έχουν φτιάξει σε σχέση με την αρχική εικόνα λόγω του ότι πειράζουμε και όλα τα pixels των ακμών σύμφωνα με τον παραπάνω τρόπο, αυτό μας θολώνει την εικόνα.

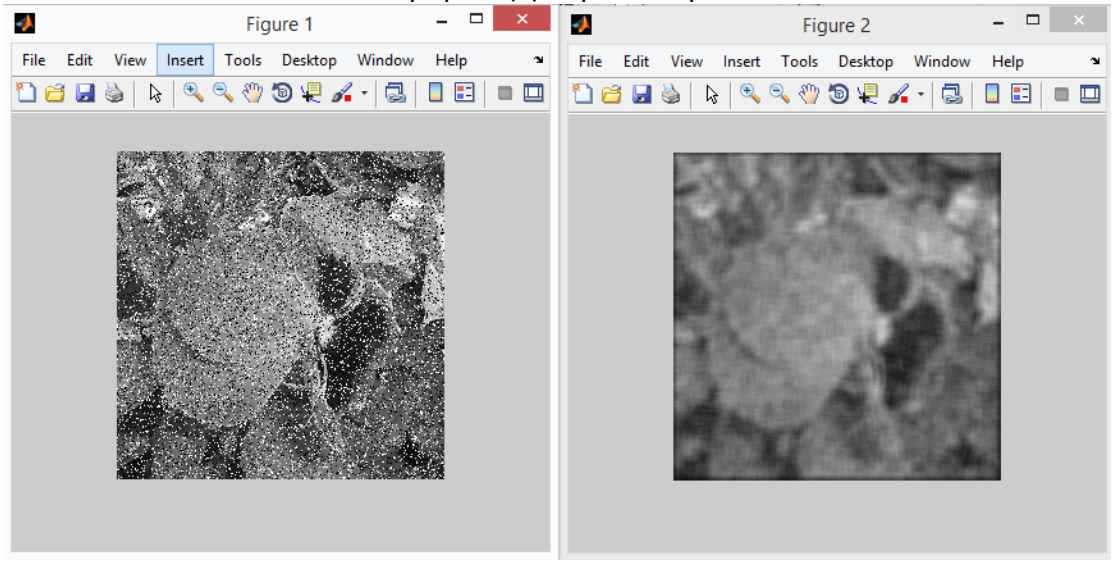
Για μέγεθος φίλτρου ίσο με 8:



Αντίστοιχα αποτελέσματα για την εικόνα panses:
Για μέγεθος φίλτρου ίσο με 3:



Για μέγεθος φίλτρου ίσο με 8:



Διαπιστώνουμε πως για μέγεθος φίλτρου 8*8 η εικόνα γίνεται περισσότερο θολή, εφόσον συντελούν περισσότερα pixels στην εύρεση νέας τιμής για την καινούργια εικόνα με αποτέλεσμα να έχουμε χειρότερο αποτέλεσμα.

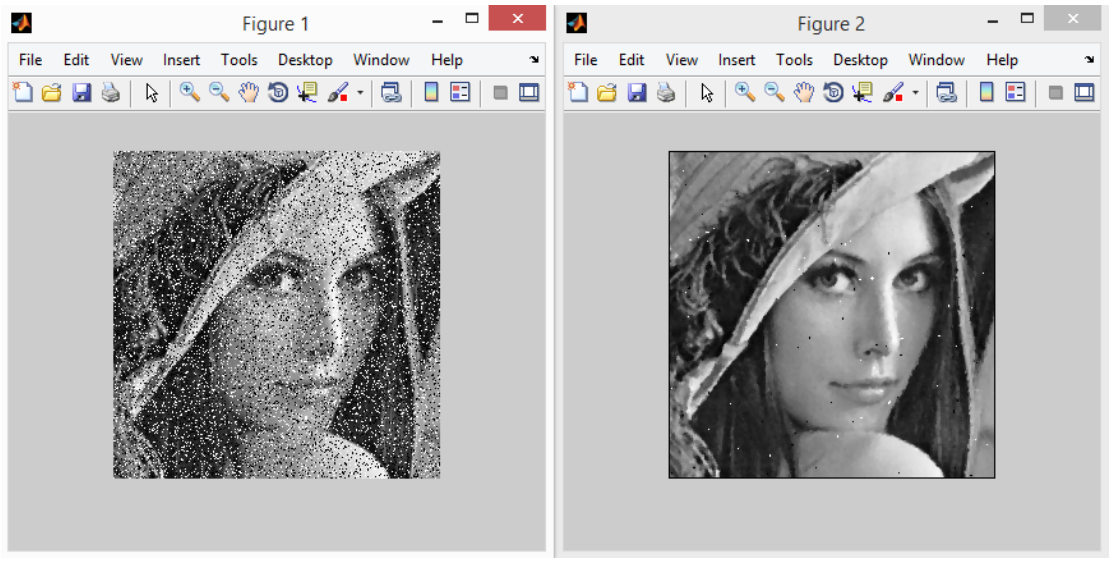
β)Για την υλοποίηση του φίλτρου median υλοποιήθηκε η συνάρτηση **ask2_erwthma4_median.m** ο κώδικας της οποίας φαίνεται παρακάτω.

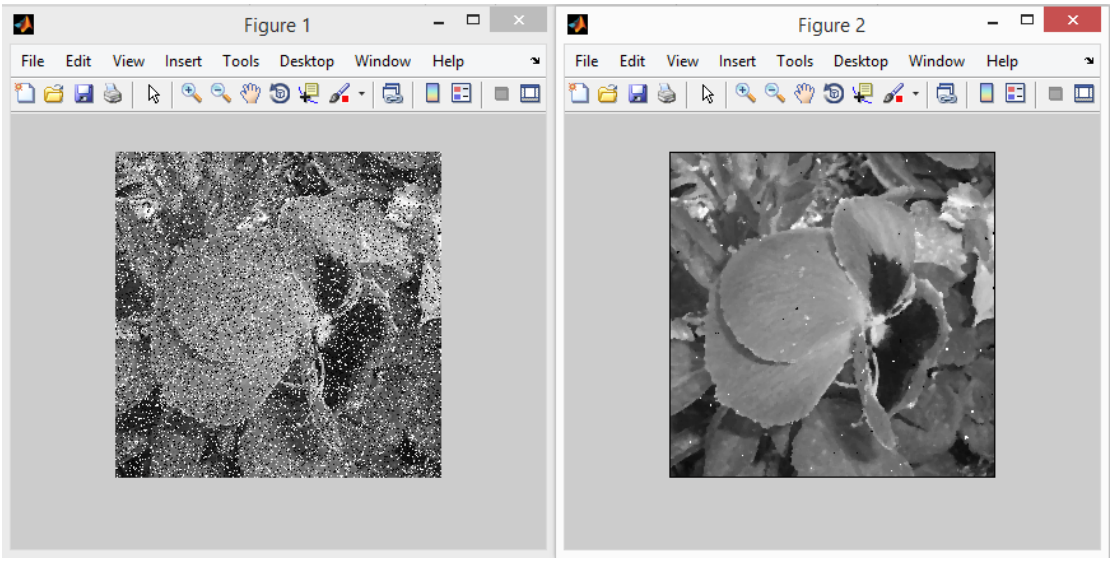
```
function [] = ask2_erwthma4_median(im)
    new_im=zeros(size(im));
    for i=2:(size(im,1)-1)
        for j=2:(size(im,2)-1)
```

```
            temp=[im(i-1,j-1) im(i-1,j) im(i-
1,j+1);im(i,j-1) im(i,j) im(i,j+1);im(i+1,j-1) im(i+1,j)
im(i+1,j+1)];
            temp=sort(temp(:));
            new_im(i,j)=temp(5);
        end
    end
    figure(1);
    imshow(uint8(im));
    figure(2);
    imshow(uint8(new_im));
end
```

Η συνάρτηση δέχεται σαν είσοδο την αφιλτράριστη εικόνα. Αρχικοποιούμε ένα πίνακα ίδιων διαστάσεων με την αφιλτράριστη εικόνα μας, ο οποίος θα περιέχει την καινούργια μας εικόνα. Περνάμε μία νοητή μάσκα 3*3 πάνω από την αρχική μας εικόνα, δημιουργούμε ένα πίνακα temp με όλα τα στοιχεία αυτής της μάσκας, σορτάρουμε τον πίνακα temp και θέτουμε στο αντίστοιχο pixel της νέας μας εικόνας το μεσαίο στοιχείο του σορταρισμένου πλέον temp πίνακα μας. Η νοητή μάσκα 3*3 με κάθε for μετακινείται, πάνω στην αρχική μας εικόνα. Κάθε δείκτης από κάθε for δείχνει στο κεντρικό σημείο της μάσκας. Παίρνοντας το κεντρικό σημείο του ταξινομημένου πίνακα temp, έχουμε καταφέρει να περιθωριοποιήσουμε τις ακραίες τιμές, δηλαδή τις τιμές θορύβου και να πάρουμε μια πολύ καλύτερη τιμή πιο κοντά στην εικόνα, από ότι στον θόρυβο. Έτσι αναμένουμε αυτός ο τρόπος να λειτουργεί πολύ καλύτερα από τον προηγούμενο. Τρέχουμε την συνάρτηση για κάθε εικόνα πληκτρολογώντας **ask2_erwthma4_median(noisy_lenna);** και **ask2_erwthma4_median(noisy_panses);**, αντίστοιχα.

Αποτελέσματα φίλτρου median:





Επιβεβαιώνουμε την μεγάλη διαφορά μεταξύ των δυο φίλτρων λόγω της περιθωριοποίησης του θορύβου, το οποίο επιτυγχάνεται μέσα από το σορτάρισμα της μάσκας.