

ΨΗΦΙΑΚΗ ΕΠΕΞΕΡΓΑΣΙΑ ΚΑΙ ΑΝΑΛΥΣΗ ΕΙΚΟΝΑΣ

ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ, ΜΕΡΟΣ Α

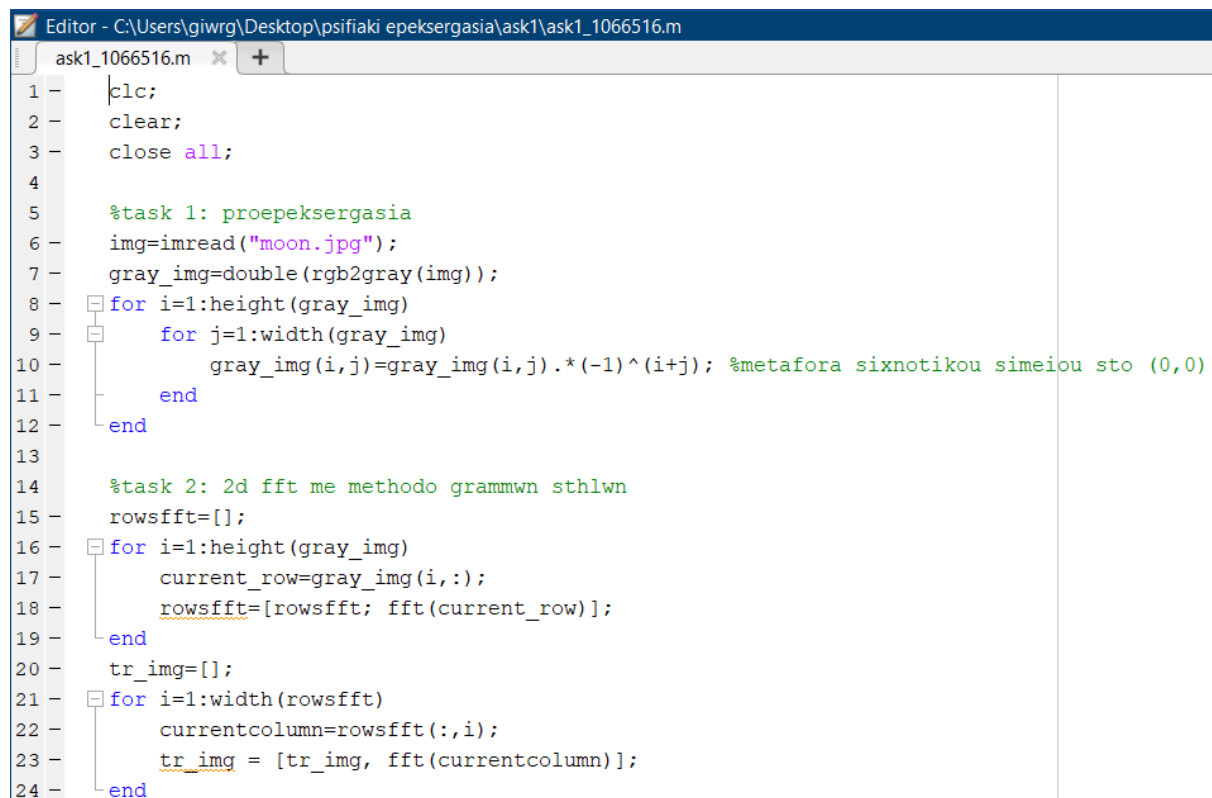
ΦΟΙΤΗΤΗΣ: ΓΕΩΡΓΙΟΣ ΠΕΤΡΑΚΗΣ

ΤΜΗΜΑ: ΗΜΤΥ

ΑΜ: 1066516

ΕΡΓΑΣΙΑ 1^Η: Φιλτράρισμα στο πεδίο συχνοτήτων

Στα παρακάτω στιγμιότυπα εμφανίζεται ο κώδικας της 1^{ης} επεξεργασίας ανά ερώτημα μαζί με τις απαραίτητες διευκρινίσεις



```
Editor - C:\Users\giwrg\Desktop\psifiaki epeksergasia\ask1\ask1_1066516.m
ask1_1066516.m  x  +
1 -  clc;
2 -  clear;
3 -  close all;
4
5 -  %task 1: proepeksergasia
6 -  img=imread("moon.jpg");
7 -  gray_img=double(rgb2gray(img));
8 -  for i=1:height(gray_img)
9 -      for j=1:width(gray_img)
10 -          gray_img(i,j)=gray_img(i,j).*(-1)^(i+j); %metafora sixnotikou simeiou sto (0,0)
11 -      end
12 -  end
13
14 -  %task 2: 2d fft me methodo grammwn sthlwn
15 -  rowsfft=[];
16 -  for i=1:height(gray_img)
17 -      current_row=gray_img(i,:);
18 -      rowsfft=[rowsfft; fft(current_row)];
19 -  end
20 -  tr_img=[];
21 -  for i=1:width(rowsfft)
22 -      currentcolumn=rowsfft(:,i);
23 -      tr_img = [tr_img, fft(currentcolumn)];
24 -  end
```

Αρχικά, μετατρέπουμε την εικόνα σε grayscale format και μεταβάλλουμε τους «συντελεστές» του πίνακα ώστε να μεταφέρουμε το συχνοτικό σημείο στο κέντρο της εικόνας. Έπειτα, για το δεύτερο ερώτημα χρησιμοποιούμε 2 for loops για να εφαρμόσουμε τον μονοδιάστατο fft με την μέθοδο γραμμών-στηλών στην εικόνα. Λαμβάνουμε το δισδιάστατο φάσμα της εικόνας και το αποθηκεύουμε στον πίνακα “tr_img”.

```
25 %task3: apeikonisi platous fasmatos se grammiki kai logarithmiki klimaka
26 - lin_mag=abs(tr_img);
27 - lin_mag=mat2gray(lin_mag);
28 - imshow(lin_mag);
29 - title("Magnitude (Linear)");
30 - figure;
31 - log_mag=abs(log(tr_img+1));
32 - log_mag=mat2gray(log_mag);
33 - imshow(log_mag);
34 - title("Magnitude (Log)");
```

Έπειτα απλώς απεικονίζουμε το πλάτος του φάσματος σε γραμμική και λογαριθμική κλίμακα.

```
35 %task4: dhmiourgia 2d lpf
36 - [f1,f2]=freqspace(256,'meshgrid'); %xrhsimopoioume meshgrid gia na dhmiourghsoume to filtro
37 - z=zeros(256,256); %gia na kathorisoume to cutoff freq
38 - for i=1:256
39 -     for j=1:256
40 -         z(i,j)=sqrt(f1(i,j).^2+f2(i,j).^2);
41 -     end
42 - end
43 - h=zeros(256,256); %to filtro
44 - for u=1:256
45 -     for v=1:256
46 -         if z(u,v)<=0.2 %normalised cutoff frequency 0.2
47 -             h(u,v)=1;
48 -         else
49 -             h(u,v)=0;
50 -         end
51 -     end
52 - end
```

Για να δημιουργήσουμε ένα ιδανικό χαμηλοπερατό φίλτρο στη συχνότητα χρησιμοποιούμε freqspace meshgrid. Εφαρμόζουμε την εξίσωση κύκλου στις δύο διαστάσεις και έπειτα την χρησιμοποιούμε για να δημιουργήσουμε το φίλτρο, το οποίο έχει τιμές (στη συχνότητα) ίσες με 1 εντός του κύκλου και 0 εκτός αυτού (τρισδιάστατη συνάρτηση). Η συνάρτηση μεταφοράς του φίλτρου απεικονίζεται στις 3 διαστάσεις και στα αποτελέσματα.

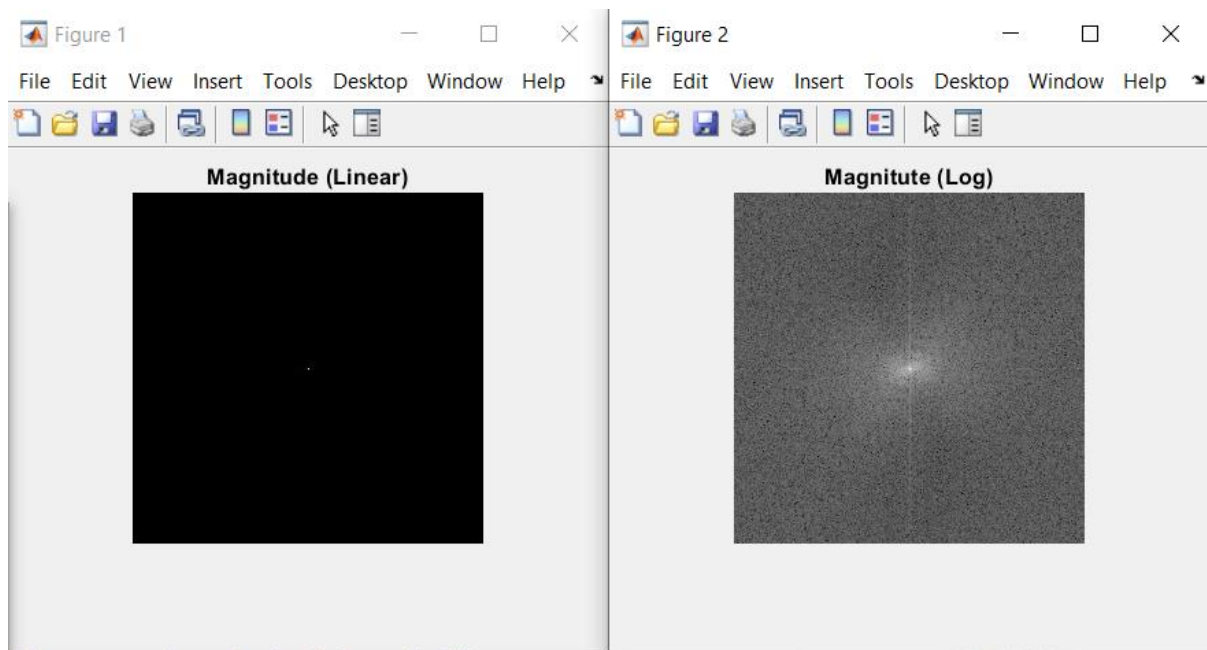
```

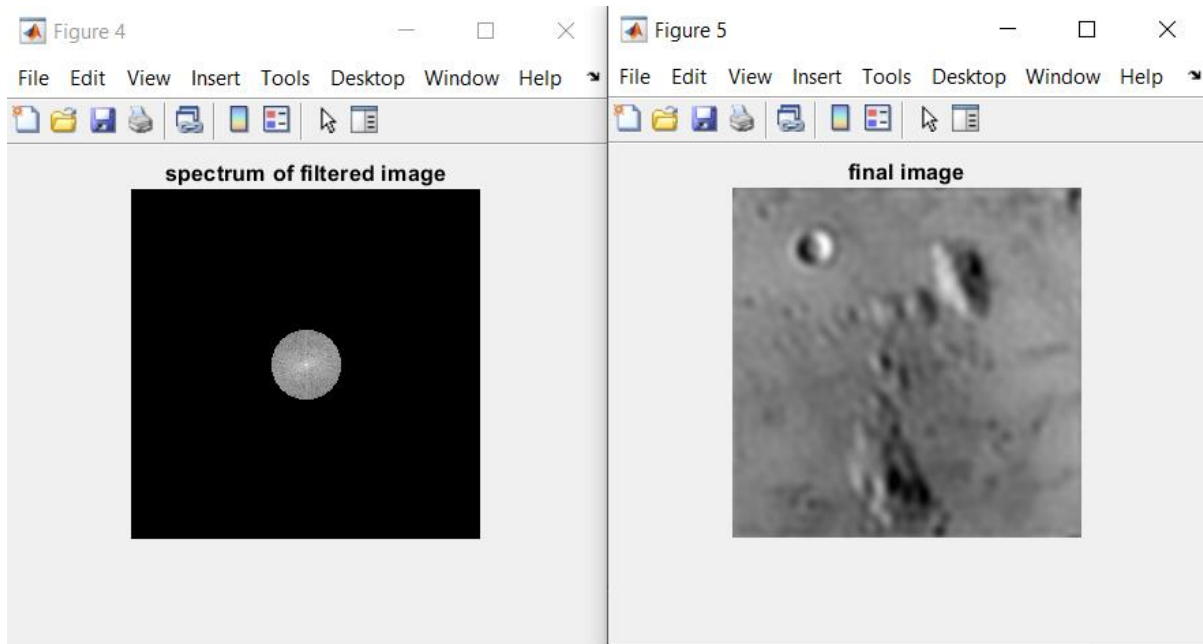
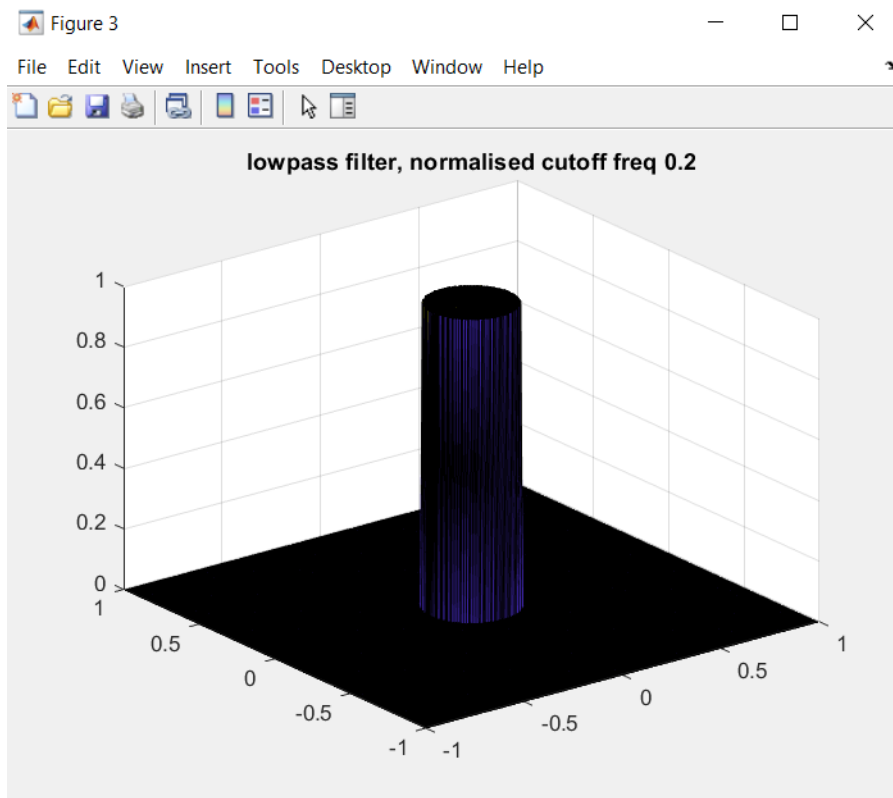
- end
figure;
surf(f1,f2,h);
title("lowpass filter, normalised cutoff freq 0.2");
filtered_img=tr_img.*h; %sineliksi ston xrono, pollaplasiasmos stin sixnotita
log_mag=abs(log(filtered_img+1));
log_mag=mat2gray(log_mag);
figure;
imshow(log_mag);
title("spectrum of filtered image");
%task 5: idft
filtered_img=ifftshift(filtered_img);
final_img=abs(ifft2(filtered_img));
figure;
imshow(final_img,[]);
title("final image");

```

Τελικά, απλώς εφαρμόζουμε το φίλτρο με πολλαπλασιασμό στη συχνότητα και απεικονίζουμε το νέο φάσμα και την φιλτραρισμένη εικόνα. Προτού εφαρμόσω τον αντίστροφο μετασχηματισμό επαναφέρω το συχνοτικό σημείο στην αρχή της εικόνας με την συνάρτηση `ifftshift()` της matlab.

Παρακάτω φαίνονται τα αποτελέσματα του κώδικα:





Όπως αναμενόταν, στην φιλτραρισμένη εικόνα τα περιγράμματα των σχημάτων εμφανίζονται αλλοιωμένα και η εικόνα θολή, εφόσον έχουμε απορρίψει τις υψηλές συχνότητες.

ΕΡΓΑΣΙΑ 2: Συμπίεση εικόνας με χρήση μετασχηματισμού DCT

Ομοίως, παρουσιάζεται ο κώδικας τμηματικά μαζί με επεξηγήσεις

```
Editor - C:\Users\giwrg\Desktop\psifiaki epeksergia\ask2\ask2_1066516.m
ask2_1066516.m x +
1 -   clc;
2 -   clear;
3 -   close all;
4
5 -   file=load("barbara.mat");
6 -   img=file.barbara;
7
8 -   img=double(rgb2gray(img));
9 -   dct_m=dctmtx(32); %dhmiourgia pinaka metasxhmatismou DCT megethous 32x32
10 -  dct_fun=@(block_struct) dct_m * block_struct.data * dct_m';
11 -  dct_im=blockproc(img,[32 32],dct_fun); %efarmogi DCT ana block
12 -  %methodos zwnhs: dhmiourgw thn zwnh kai meta kanw padding gia na ftiaksw
13 -  %tin maska
14 -  msevalues_zone=[];
15 -  msevalues_threshold=[];
16 -  r_zone=[];
17 -  for r=1:16
18 -      zone=ones(r,r);
19 -      r_zone=[r_zone r/32]; %pososto pixels pou kratame gia plotting
20 -      mask=padarray(zone,[32-r 32-r],'post');
21 -      %discard bits according to mask
22 -      compressed_img=blockproc(dct_im,[32 32], @(block_struct) mask.* block_struct.data);
23 -      inversedct= @(block_struct) dct_m'*block_struct.data*dct_m;
24 -      reconstructed_img=blockproc(compressed_img,[32 32],inversedct);
25 -      msevalues_zone=[msevalues_zone immse(reconstructed_img, img)];
26 -  end
```

Για την υλοποίηση του μετασχηματισμού κατά «μπλοκ» μεγέθους 32x32, χρησιμοποίησα την συνάρτηση “blockproc()” (line 11) του image processing toolbox της matlab. Η συνάρτηση αυτή δέχεται έναν πίνακα δεδομένων, τον χωρίζει σε μπλοκ ([32 32] στην προκειμένη περίπτωση) και εφαρμόζει σε κάθε ένα μπλοκ μία συνάρτηση που δέχεται ως όρισμα ένα “block_struct” (στην προκειμένη περίπτωση την dct_fun, στην οποία ορίζουμε ουσιαστικά την πράξη του μετασχηματισμού DCT).

```
17 -  for r=4:21
18 -      zone=ones(r,r);
19 -      r_zone=[r_zone r^2/32^2]; %pososto sintelestwn pou kratame gia plotting
20 -      mask=padarray(zone,[32-r 32-r],'post');
21 -      %discard bits according to mask
22 -      compressed_img=blockproc(dct_im,[32 32], @(block_struct) mask.* block_struct.data);
23 -      inversedct= @(block_struct) dct_m'*block_struct.data*dct_m;
24 -      reconstructed_img=blockproc(compressed_img,[32 32],inversedct);
25 -      msevalues_zone=[msevalues_zone immse(reconstructed_img, img)];
26 -  end
27 -  %endeiktika kanoume imshow to r=50%
28 -  figure;
29 -  subplot(221);
30 -  imshow(reconstructed_img, []);
31 -  title("zone mask, r=0.5");
```

Μετά την εφαρμογή του μετασχηματισμού, δημιουργούμε την μάσκα συντελεστών DCT με την μέθοδο ζώνης. Αρχικά ορίζουμε έναν πίνακα με μονάδες διαστάσεων (rxr) και κάνουμε padding με 32-r μηδενικά για να τον φέρουμε σε διαστάσεις 32x32, που είναι και το μέγεθος του μπλοκ. Αρχίζουμε από 4 στοιχεία και καταλήγουμε σε 22 στοιχεία διατηρώντας το ποσοστό r_zone μέσα στο διάστημα [5%, 50%]. Η εφαρμογή της μάσκας γίνεται ξανά με την συνάρτηση blockproc() σε κάθε μπλοκ της εικόνας ξεχωριστά. Τέλος, υπολογίζουμε το μέσο τετραγωνικό σφάλμα με την συνάρτηση "immse()" της matlab για κάθε τιμή του r.

```

33 - r_threshold=[]; %pososta
34 - for threshold=0.23:-0.001:0
35 -     threshold_dct_m=dct_m; %gia na dhmiourghsw ton neo dct_m me xrhsh tou threshold
36 -     for x=1:32
37 -         for y=1:32
38 -             if(dct_m(x,y)<threshold) threshold_dct_m(x,y)=0;
39 -             end
40 -         end
41 -     end
42 -     discarded_pixels=sum(threshold_dct_m(:)==0);
43 -     r_threshold=[r_threshold (32*32-discarded_pixels)/(32*32)];
44 -     inversedct= @(block_struct) threshold_dct_m'*block_struct.data*threshold_dct_m;
45 -     reconstructed_img=blockproc(compressed_img,[32 32],inversedct);
46 -     msevalues_threshold=[msevalues_threshold immse(reconstructed_img, img)];
47 - end
48
49 - subplot(222);
50 - imshow(reconstructed_img, []);
51 - ttl=sprintf("threshold mask, r=%1.3f",r_threshold(length(r_threshold)));
52 - title(ttl);
53

```

Με όμοιο τρόπο δημιουργούμε και την μάσκα με την μέθοδο κατωφλίου. Απλώς κρατάμε τις τιμές των συντελεστών πάνω από ένα ορισμένο κατώφλι. Αρχίζοντας με μεγάλη τιμή κατωφλίου έχουμε υψηλή συμπίεση (χαμηλό ποσοστό r) και μειώνοντας έχουμε χαμηλότερη συμπίεση (υψηλό ποσοστό r). Λαμβάνουμε την συμπιεσμένη εικόνα από τον αντίστροφο μετασχηματισμό DCT για κάθε τιμή κατωφλίου και υπολογίζουμε το μέσο τετραγωνικό σφάλμα.

Και στις δύο περιπτώσεις ο αντίστροφος μετασχηματισμός DCT εφαρμόζεται με την συνάρτηση blockproc με αντίστοιχο τρόπο όπως τον ευθύ μετασχηματισμό.

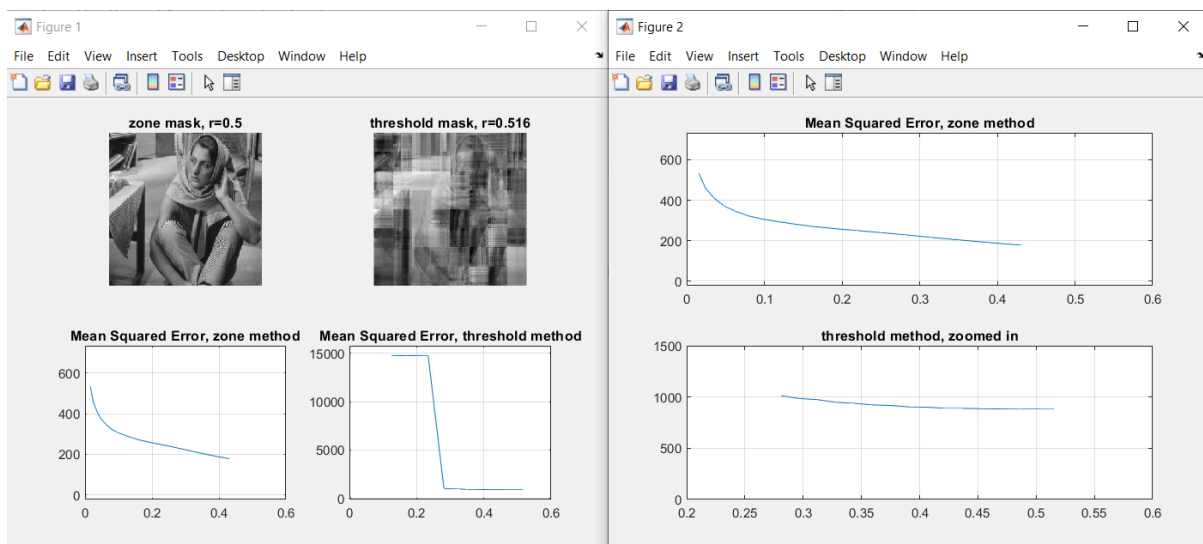
```

49 - subplot(222);
50 - imshow(reconstructed_img, []);
51 - ttl=sprintf("threshold mask, r=%1.3f", r_threshold(length(r_threshold)));
52 - title(ttl);
53
54 - subplot(223);
55 - plot(r_zone, msevalues_zone);
56 - axis([0.0 0.6 min(msevalues_zone)-200 max(msevalues_zone)+200]);
57 - grid on;
58 - title("Mean Squared Error, zone method");
59
60 - subplot(224);
61 - plot(r_threshold, msevalues_threshold);
62 - axis([0.0 0.6 min(msevalues_threshold)-1000 max(msevalues_threshold)+1000]);
63 - grid on;
64 - title("Mean Squared Error, threshold method");

```

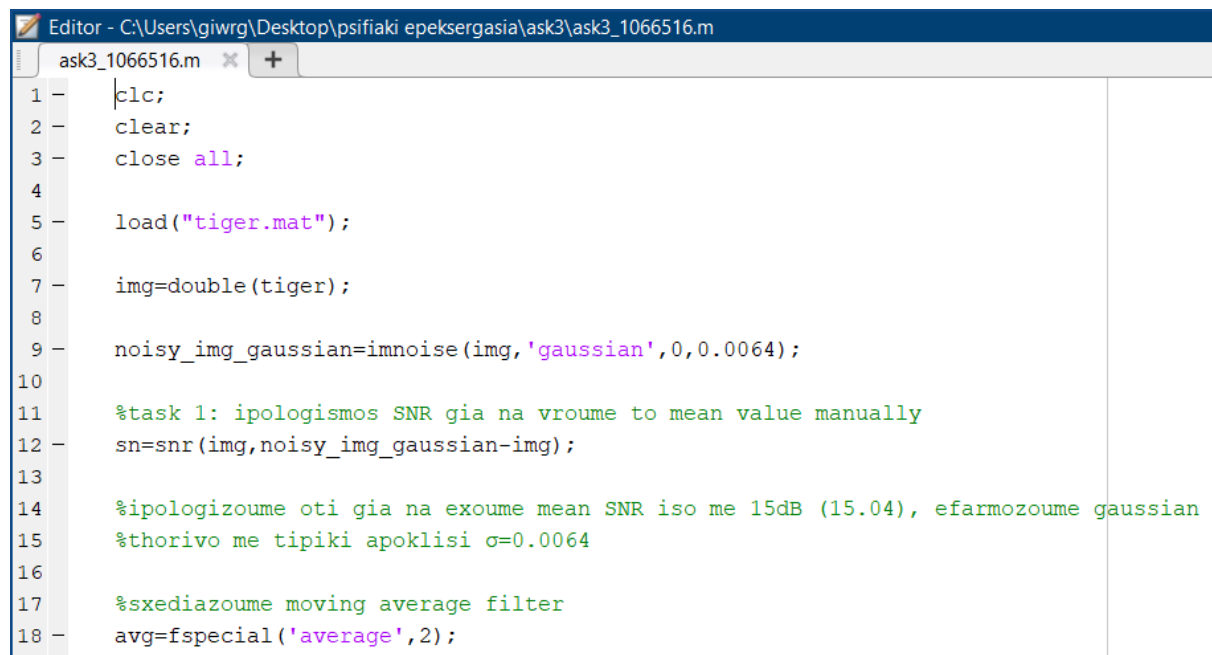
Ο υπόλοιπος κώδικας έχει να κάνει απλώς με την ευδιάκριτη επίδειξη των αποτελεσμάτων σε plots.

Τα αποτελέσματα φαίνονται παρακάτω



Από ότι φαίνεται η μέθοδος ζώνης είναι αρκετά πιο αποτελεσματική σχετικά με την μέθοδο κατωφλίου. Μάλιστα, ακόμα και για ποσοστό 50% διατηρούμενων συντελεστών, η μέθοδος κατωφλίου έχει αρκετά μεγαλύτερο μέσο τετραγωνικό σφάλμα από την μέθοδο ζώνης και η εικόνα φαίνεται αρκετά παραμορφωμένη. Τα αποτελέσματα αυτά φαίνονται μεγεθυμένα και στο δεύτερο διάγραμμα, όπου η μέθοδος ζώνης φαίνεται να έχει καλύτερες επιδόσεις ακόμα και για μικρότερα ποσοστά συντελεστών. Τέλος, όπως αναμενόταν, παρατηρούμε ότι η απόδοση της μεθόδου ζώνης φαίνεται να αυξάνεται με φθίνον ρυθμό με την αύξηση του ποσοστού διατηρούμενων συντελεστών.

ΕΡΓΑΣΙΑ 3^Η: Βελτίωση εικόνας – Φιλτράρισμα θορύβου



```
Editor - C:\Users\giwrg\Desktop\psifiaki epeksergia\ask3\ask3_1066516.m
ask3_1066516.m x +
1 - clc;
2 - clear;
3 - close all;
4
5 - load("tiger.mat");
6
7 - img=double(tiger);
8
9 - noisy_img_gaussian=imnoise(img,'gaussian',0,0.0064);
10
11 %task 1: ipologismos SNR gia na vroume to mean value manually
12 - sn=snr(img,noisy_img_gaussian-img);
13
14 %ipologizoume oti gia na exoume mean SNR iso me 15dB (15.04), efarmozoume gaussian
15 %thorivo me tipiki apoklisi sigma=0.0064
16
17 %sxediazoume moving average filter
18 - avg=fspecial('average',2);
```

Αρχικά υπολογίζουμε «χειροκίνητα» την τυπική απόκλιση του λευκού θορύβου μηδενικής μέσης τιμής, ώστε να έχουμε λόγο σήματος προς θόρυβο ίσο με 15dB. Η τιμή της τυπικής απόκλισης προσδιορίζεται ως 0.0064.

Έπειτα, δημιουργούμε το φίλτρο κινούμενου μέσου όρου με την συνάρτηση “fspecial()” του image processing toolbox της matlab. Κατά την εφαρμογή του παρατήρησα ότι η καλύτερη απόδοση επιτυγχάνεται για φίλτρο διαστάσεων 4x4.

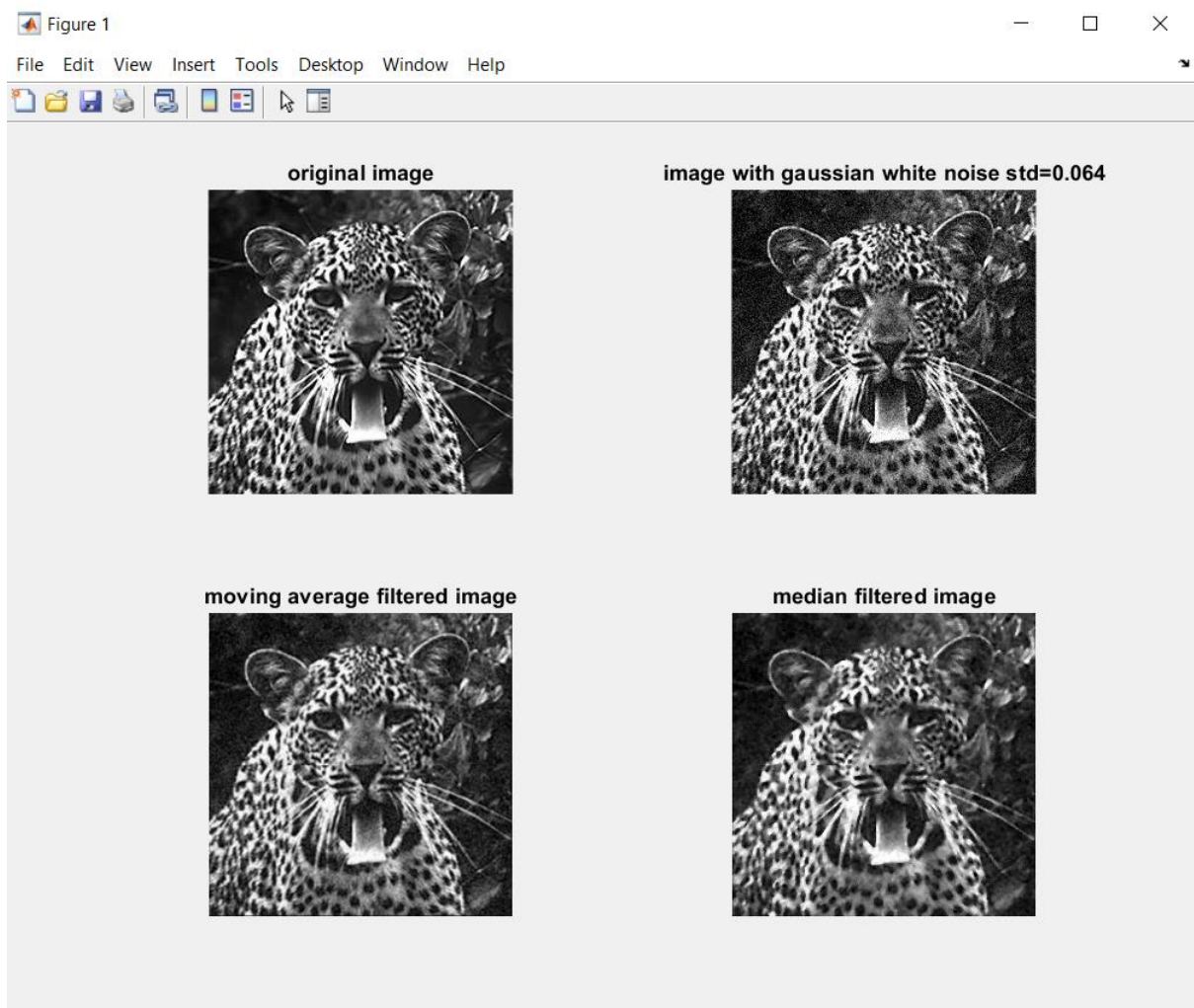

```

20 %apply to image
21 - filtered_avg=imfilter(noisy_img_gaussian,avg,'conv');
22 - filtered_median=medfilt2(noisy_img_gaussian);
23 - [peak,snratiofiltered_avg]=psnr(filtered_avg,img);
24 - [peak,snratiofiltered_median]=psnr(filtered_median,img);
25 %parathrw oti meta thn efarmogh tou filtrou o logos SNR den veltiwnetai
26 %parolo pou o thorivos ipsilis sixnotitas ehei apovlithei
27 %parathrw oti h auksisi twn diastasewn tou moving average filter
28 %xeirotereuei tin poiouthta ths eikonas
29 %parathrhsh: to moving average filter ine pio katallhlo apo to moving
30 %median giati to moving median prokalei tholwsi kai einai pio katallhlo gia
31 %kroustiko thorivo
32 - figure('Position',[200 100 800 600]);
33 - subplot(221);
34 - imshow(img);
35 - title("original image");
36 - subplot(222);
37 - imshow(noisy_img_gaussian);
38 - title("image with gaussian white noise std=0.064");
39 - subplot(223);
40 - imshow(filtered_avg);
41 - title("moving average filtered image");
42 - subplot(224);
43 - imshow(filtered_median);
44 - title("median filtered image");

```

Φιλτράρουμε την εικόνα με το φίλτρο κινούμενου μέσου όρου με την συνάρτηση `imfilter()` και με το median φίλτρο με την συνάρτηση `medfilt2` και υπολογίζουμε και σε αυτή τη περίπτωση τους λόγους σήματος προς θόρυβο ενδεικτικά. Έπειτα εμφανίζουμε την αρχική εικόνα, την εικόνα μετά την προσθήκη λευκού θορύβου και τα αποτελέσματα των 2 φιλτραρισμάτων.

Τα πρώτα αποτελέσματα φαίνονται παρακάτω

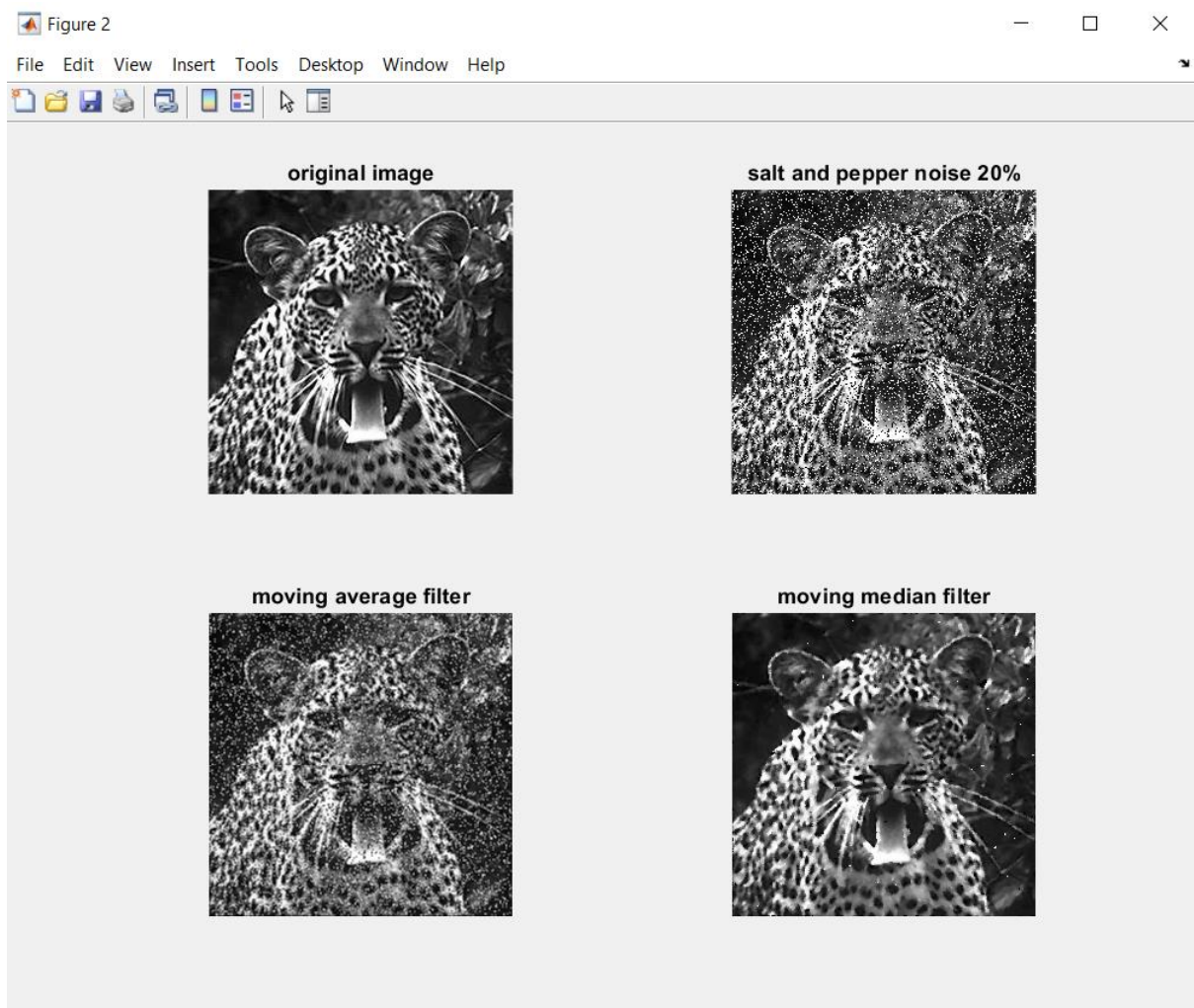


Ίσως δεν είναι ιδιαίτερα ευδιάκριτο, αλλά, όπως σημειώνω και με σχόλια στον κώδικα, το φίλτρο κινούμενου μέσου όρου φαίνεται να «εξομαλύνει» τον θόρυβο στην εικόνα, χωρίς να απορρίπτει «πληροφορία» από την αρχική εικόνα ή να «αλλοιώνει» τα σχήματα της (πληροφορία υψηλής συχνότητας). Αντίθετα, το median φίλτρο εκτός από το ότι δεν έχει ανάλογη αποτελεσματικότητα στην απόρριψη του θορύβου, προκαλεί θόλωση της εικόνας και αλλοίωση των περιγραμμάτων των σχημάτων (όπως για παράδειγμα στα περιγράμματα των μαύρων κηλίδων στο τρίχωμα του ζώου).

Με ανάλογο τρόπο εφαρμόζουμε και κρουστικό θόρυβο ("Salt and pepper noise"). Παρακάτω φαίνεται ο κώδικας και τα αποτελέσματα.

```
Editor - C:\Users\giwrg\Desktop\psifiaki epeksergasia\ask3\ask3_1066516.m
ask3_1066516.m  x  +
46 %task2: kroustikos thorivos
47 - noisy_img_imp=imnoise(img,'salt & pepper', 0.2);
48 - filtered_avg=imfilter(noisy_img_imp,avg,'conv');
49 - filtered_median=medfilt2(noisy_img_imp);
50
51 - figure('Position',[200 100 800 600]);
52 - subplot(221);
53 - imshow(img);
54 - title("original image");
55 - subplot(222);
56 - imshow(noisy_img_imp,[]);
57 - title("salt and pepper noise 20%");
58 - subplot(223);
59 - imshow(filtered_avg,[]);
60 - title("moving average filter");
61 - subplot(224);
62 - imshow(filtered_median,[]);
63 - title("moving median filter");
64 %parathrw oti to moving average filter ine teleiws akatalalhlo gia na
65 %afairesei ton kroustiko thorivo se antithesi me to moving median filter
```

Και τα αποτελέσματα



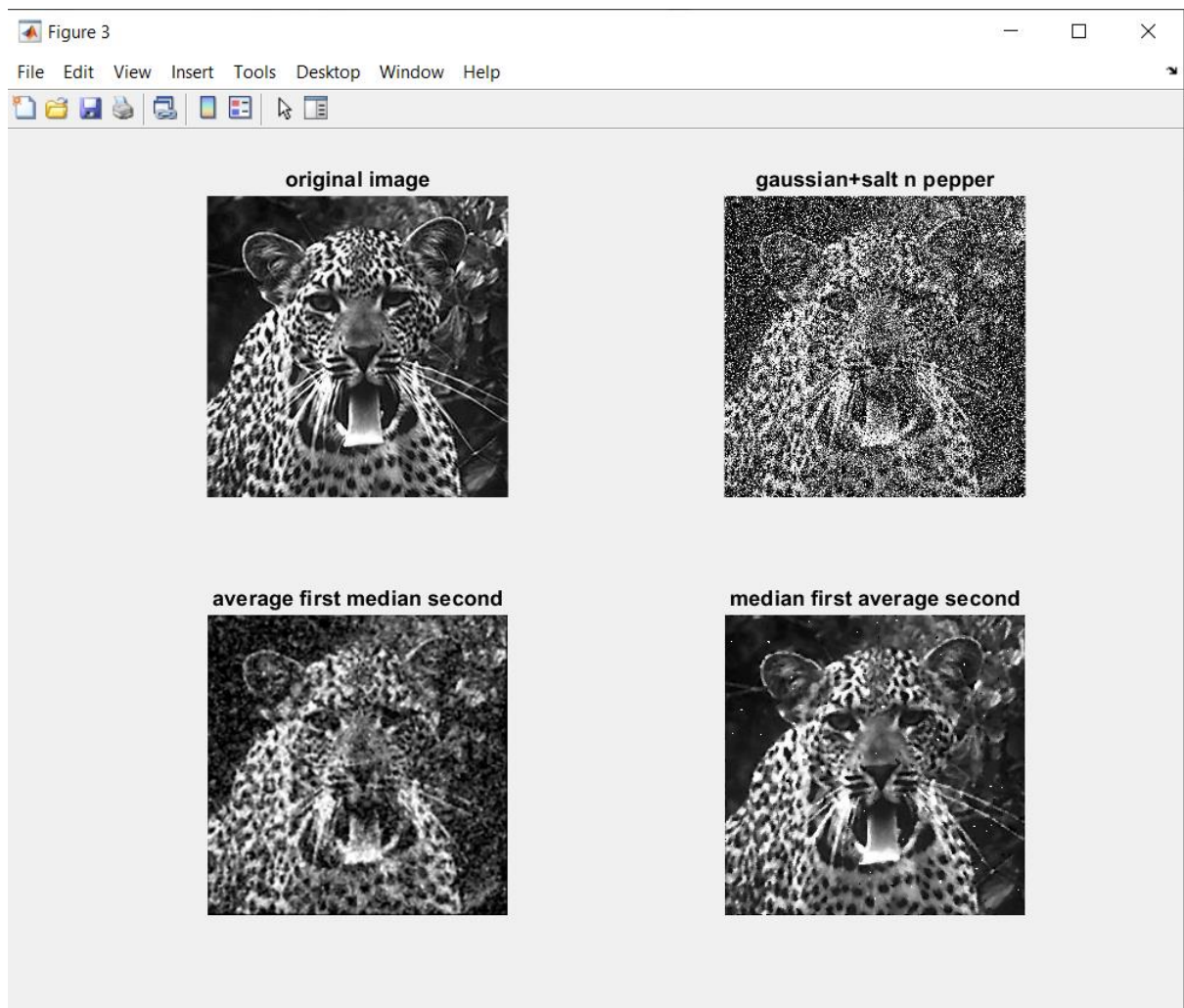
Παρατηρώ ότι το φίλτρο κινούμενου μέσου όρου δεν έχει καμία επίδραση στο φιλτράρισμα του θορύβου. Μάλιστα, θα μπορούσε κανείς να πει ότι το αποτέλεσμα του φιλτραρίσματος είναι ακόμα χειρότερο από την «θορυβώδης» εικόνα εφόσον φαίνεται σαν να έχουμε χάσει «πληροφορία» από την αρχική εικόνα και απλώς να έχουμε «λειάνει» τον θόρυβο υψηλής συχνότητας, ο οποίος ακόμα υποβαθμίζει αρκετά το τελικό αποτέλεσμα και κάνει την εικόνα να φαίνεται χειρότερη από ότι πριν το φιλτράρισμα. Αντίθετα, το φίλτρο median είναι πολύ αποτελεσματικό στην εξουδετέρωση των «άσπρων» κηλίδων, έχοντας όμως βέβαια επιδράσει και στην αρχική εικόνα με τρόπο παρόμοιο όπως και με το παράδειγμα του λευκού θορύβου.

Από τα αποτελέσματα αυτά, αναμένουμε ότι για να εξουδετερώσουμε τον συνδυαστικό θόρυβο θα έχουμε καλύτερα αποτελέσματα εφαρμόζοντας αρχικά το φίλτρο median και έπειτα το φίλτρο moving average (λόγω της αλλοίωσης που προκαλεί το δεύτερο φίλτρο στην περίπτωση ύπαρξης θορύβου υψηλής συχνότητας). Παρόλα αυτά παρουσιάζεται και το

παράδειγμα εφαρμογής των φίλτρων με αντίθετη σειρά για επιβεβαίωση της υπόθεσης αυτής.

```
67 %task3: sindiasmos thoriwn
68 - noisy_img=imnoise(img,'gaussian',0,0.064);
69 - noisy_img=imnoise(noisy_img,'salt & pepper',0.2);
70
71 - avg_median_filtered=imfilter(medfilt2(noisy_img),avg,'conv'); %prwta median meta average
72 - median_avg_filtered=medfilt2(imfilter(noisy_img,avg,'conv'));
73
74 - figure("Position",[200,100,800,600]);
75 - subplot(221);
76 - imshow(img);
77 - title("original image");
78 - subplot(222);
79 - imshow(noisy_img,[]);
80 - title("gaussian+salt n pepper");
81 - subplot(223);
82 - imshow(avg_median_filtered,[]);
83 - title("average first median second");
84 - subplot(224);
85 - imshow(median_avg_filtered,[]);
86 - title("median first average second");
87
88 %opws itan anamenomeno apo tin efarmogi tou average filter se
89 %eikona me kroustiko thoriwo (to apotelesma einai ousiastika apwleia
90 %plhroforias), to apotelesma efarmogis prwta tou average filter einai
91 %arketa xeirotero
```

Τα αποτελέσματα:



Τα αποτελέσματα επιβεβαιώνουν τις αρχικές υποθέσεις.

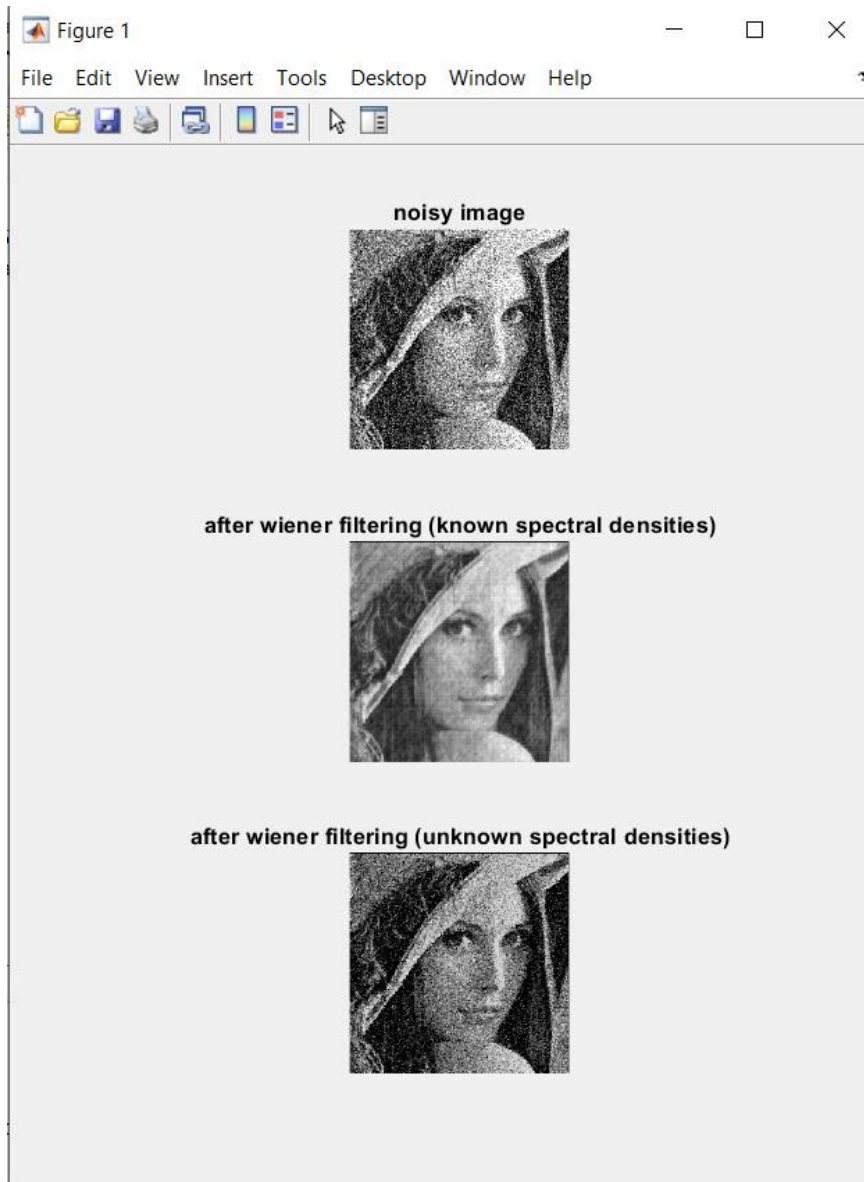
ΕΡΓΑΣΙΑ 5^Η: Αποκατάσταση εικόνας – Αποσυνέλιξη

Μέρος Α:

```
Editor - C:\Users\giwrg\Desktop\psifiaki epeksergasia\ask5\ask5_1066516.m
ask5_1066516.m x +
1 -   clc;
2 -   clear;
3 -   close all;
4
5 -   import psf.*;
6
7 -   img=imread("lenna.jpg");
8 -   img=double(mat2gray(rgb2gray(img))); %image scaling sto diasthma [0 1]
9
10 -  noisy_img=imnoise(img,'gaussian',0,0.03); %standard deviation iso me 0.03
11 -  noise=noisy_img-img;
12 -  sn=snr(img,noise); %gia SNR=10dB
13 -  figure('Position',[100 100 500 600]);
14 -  subplot(311);
15 -  imshow(noisy_img);
16 -  title("noisy image");
17
18 -  %ipologizw tous fft tou thorivou kai tou simatos kai apo auto ta fasmata
19 -  %isxios
20 -  %efarmozw ta vimata tou filtrarismatos wiener
21 -  noisy_img=padarray(noisy_img,[round(257/2),round(255/2)-1],'both'); %2Nx2N padding
22 -  noise=padarray(noise,[round(257/2),round(255/2)-1],'both');
23 -  meang=mean(noisy_img(:));
24 -  noisy_img=noisy_img-meang; %afairw thn mesh timi tou noisy shmatos
25 -  noisylimg_sp=fft2(noisy_img); %ipologizw tous metasxhmatismous fft2d
26 -  noise_sp=fft2(noise);
27
28 -  %fasmatikes piknotites isxios
29 -  Gg=abs(noisylimg_sp).^2/(256*256);
30 -  Gn=abs(noise_sp).^2/(256*256);
31
32 -  Gf=Gg-Gn;
33 -  Hw=Gf./(Gg+Gn); %dhmiourgw tin sinartisi metaforas tou filtrou 2Nx2N
34 -  filtered_fft=Hw.*noisylimg_sp; %Hadamard product elementwise pollaplasiasmos
35 -  filtered_img=ifft2(filtered_fft); %2D-IFFT
36
37 -  filtered_img=filtered_img+meang; %prosthetw ksana thn mesh timh stin eikona
38 -  filtered_img=filtered_img(round(257/2):round(257/2)+257,round(255/2):round(255/2)+255); %afairw to padding
39 -  subplot(312);
40 -  imshow(filtered_img,[]);
41 -  title("after wiener filtering (known spectral densities)");
42
43 -  %xwris gnwsi tw n fasmatikwn piknotitwn (?)
44 -  filtered_img=wiener2(noisy_img);
45 -  subplot(313);
46 -  imshow(filtered_img(round(257/2):round(257/2)+257,round(255/2):round(255/2)+255));
47 -  title("after wiener filtering (unknown spectral densities)");
```

Αρχικά, φορτώνουμε την εικόνα στο πρόγραμμα και την κλιμακώνουμε στο διάστημα [0 1] ώστε να αντιστοιχεί στον λευκό θόρυβο που παράγει η συνάρτηση `imnoise()`. Έπειτα, όπως στην προηγούμενη άσκηση, καθορίζουμε πρακτικά την τυπική απόκλιση του λευκού θορύβου ώστε να έχουμε λόγω σήματος προς θόρυβο ίσο με 10dB. Η τυπική απόκλιση προκύπτει ίση με 0.03.

Έπειτα, εφαρμόζουμε τα βήματα του φιλτραρίσματος που αναφέρονται στις διαφάνειες του μαθήματος. Κάθε ξεχωριστό βήμα αναφέρεται υπό μορφή σχολίου μαζί με τον κώδικα. Για το φιλτράρισμα χωρίς την γνώση των φασματικών πυκνοτήτων, χρησιμοποιούμε την συνάρτηση `wiener2` της Matlab. Τα αποτελέσματα φαίνονται παρακάτω



Μέρος Β:


```

Editor - C:\Users\giwrg\Desktop\psifiaki epeksergia\ask5\ask5_1066516.m
ask5_1066516.m x +
49 %merosb
50 %gia na ipologisoume tin kroustiki apokrisi tou sistimatos katagrafis
51 %aplws xrhsimopoioume gia eisodo thn 2D kroustiki sinartisi
52 - d=zeros(257,255);
53 - d(round(257/2),round(255/2))=1;
54 - hpsf=psf(d);
55 - sp_hpsf=fft2(hpsf);
56 - figure("Position",[100 100 400 600]);
57 - subplot(311);
58 - imshow(abs(sp_hpsf),[]);
59 - title("psf 2D impulse response");
60
61 - subplot(312);
62 - psfnoisy=psf(img);
63 - imshow(psfnoisy);
64 - title("after psf");
65 - sp_psfnoisy=fft2(psfnoisy);
66 - H=1./sp_hpsf; %antistrofo filtro
67 - sp_filtered_img=zeros(257,255);
68 - msevalues=[];

```

Για να καθορίσουμε την κρουστική απόκριση του συστήματος psf απλώς εισάγουμε κρουστική είσοδο στο σύστημα. Έπειτα υπολογίζουμε την συνάρτηση μεταφοράς του από τον fft μετασχηματισμό της κρουστικής απόκρισης. Από την συνάρτηση μεταφοράς δημιουργούμε και το αντίστροφο φίλτρο.

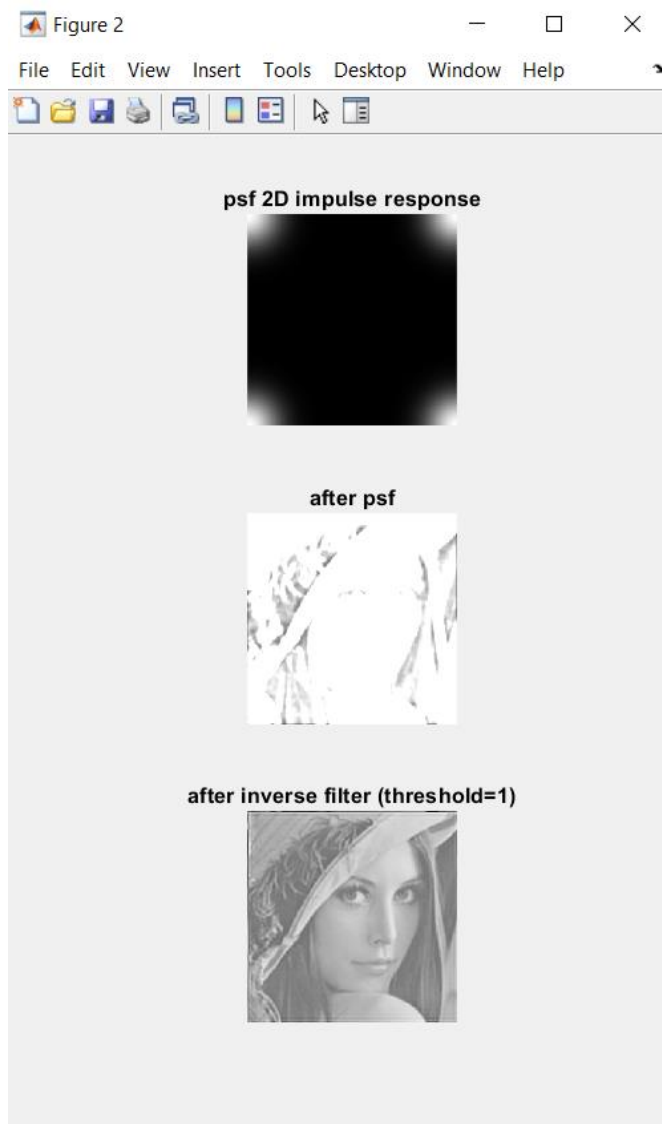
```

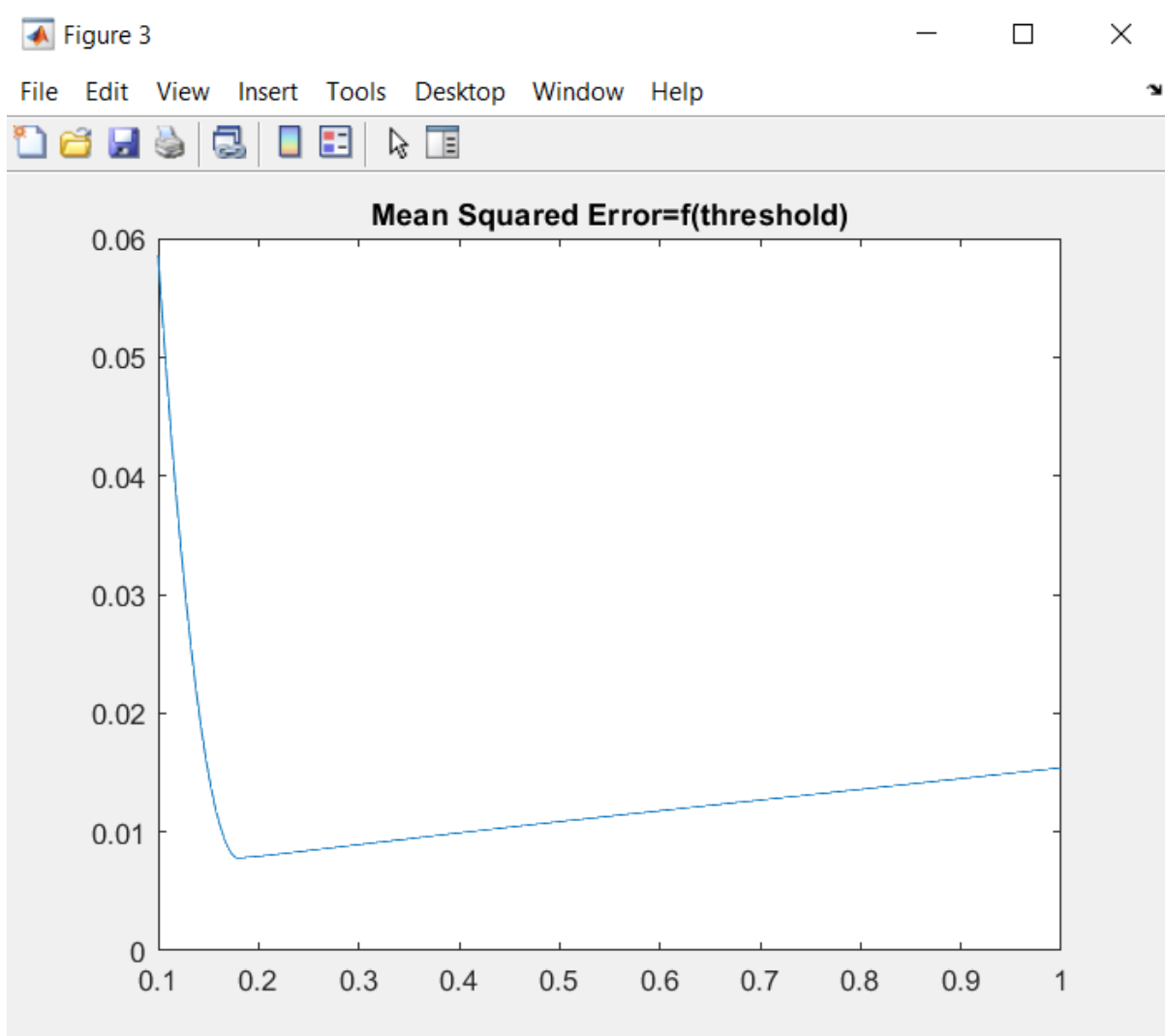
69 - for threshold=0.1:0.001:1
70 -     for x=1:height(H)
71 -         for y=1:width(H)
72 -             if(abs(H(x,y))<threshold)
73 -                 sp_filtered_img(x,y)=sp_psfnoisy(x,y)*H(x,y);
74 -             else
75 -                 sp_filtered_img(x,y)=sp_psfnoisy(x,y)*H(x,y)*threshold*abs(1./H(x,y));
76 -             end
77 -         end
78 -     end
79 -     filtered_img=ifft2(sp_filtered_img);
80 -     filtered_img=ifftshift(filtered_img);
81 -     msevalues=[msevalues immse(img,filtered_img)];
82 - end
83
84 - subplot(313);
85 - imshow(filtered_img,[]);
86 - title("after inverse filter (threshold=1)");
87 - figure;
88 - plot(0.1:0.001:1,msevalues);
89 - title('Mean Squared Error=f(threshold)');

```

Τέλος εφαρμόζουμε το φίλτρο για διάφορες τιμές κατωφλίου και λαμβάνουμε

κάθε φορά το μέσο τετραγωνικό σφάλμα. Τα αποτελέσματα φαίνονται παρακάτω





Στο διάγραμμα φαίνεται ότι το μέσο τετραγωνικό σφάλμα μειώνεται αρχικά απότομα για τιμές στο διάστημα (0.1, 0.2) αλλά αρχίζει σταδιακά να αυξάνεται μαζί με το κατώφλι για μεγαλύτερες τιμές. Από το διάγραμμα αυτό μπορούμε να προβλέψουμε και ότι το μέσο τετραγωνικό σφάλμα θα λάβει μεγάλες τιμές σε περίπτωση που δεν χρησιμοποιήσουμε κατώφλι (threshold > άπειρο).