Chapter 1

Filter in the Frequency Domain

It is really important to develop the first function because it will be used in every single function in the following assignment. The objective is to develop a function that can take as input an image and a filter and has as output the result filtered image.
There are several steps that need to be followed in order to create this function.

- First of all multiply the input image with $(-1)^{n_1+n_2}$, this will ensure that the matrix will be in the middle of the image.
- Fourier Transform of the result image.
- Multiplication pixel by pixel of the filter with the result of the Fourier Transformation.
- Back Fourier transform and of course again multiplication of the pixels with $(-1)^{n_1+n_2}$.

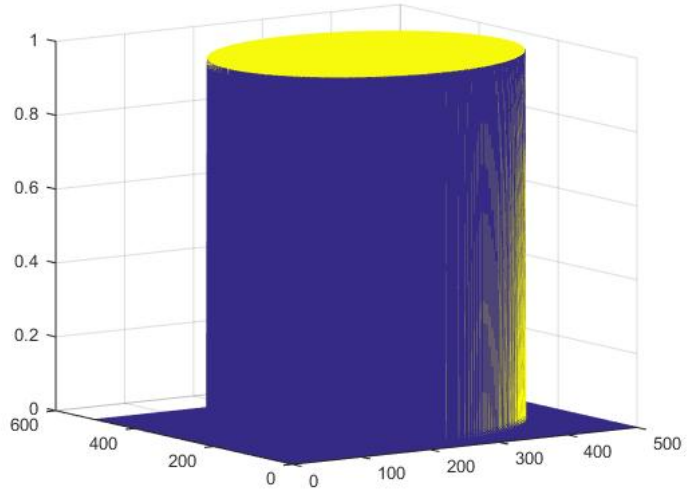
```
function [ imOut ] = myFiltFreq( imIn, someFilt )

  N=size(imIn,1);

  for n1=1:N
      for n2=1:N
```

```
            imIn(n1,n2)=imIn(n1,n2)*( (-1)^(n1+n2) );

        end
    end

    %% Metasximatismo fourier tis ketrarismenis

    Fnew = fft2(imIn);

    %% Ginomeno  fourier me filtro

    G=Fnew.*someFilt;

    %%  Use of ifft2 in order to take f(n1,n2)

    g=ifft2(G);

    %% Multiply f(n1,n2) with (-1)^(n1+n2)
    % In order to transform the matrix in the midle

    for n1=1:N
        for n2=1:N

            g(n1,n2)=g(n1,n2)*( (-1)^(n1+n2) );

        end
    end

    imOut=g;

end
```

Chapter 2

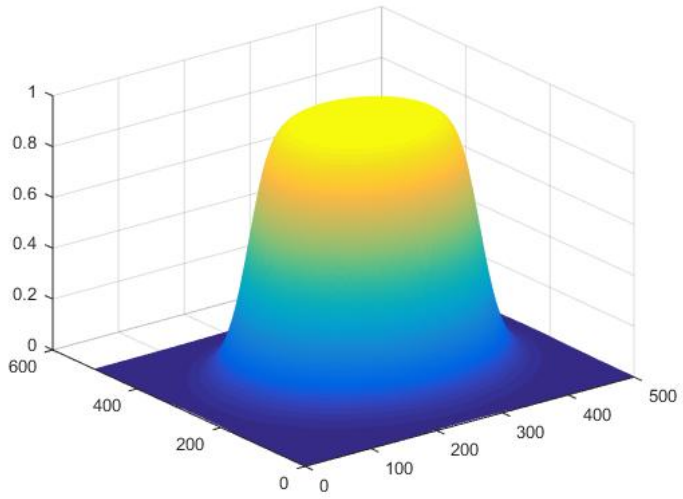
Designing of Filters

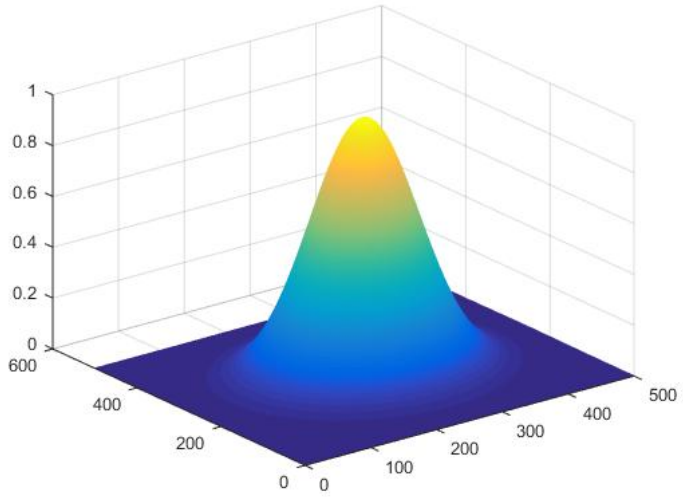
In this part of the assignment there are developed several filters in the frequency domain, with the purpose of comparing them in the *Demo2*. Different categories of filters that are being developed:



- Low Pass Filters
- High Pass Filters
- Band Pass Filters
- Ideal Filters
- Butter-worth Filters

- Gauss Filters





At this point the code of the ButterWorth filter will be shown just for reference.

```
function filtOut = myLowPassButterworth(cutOff , n, M)
%  cutOff -> I sixnotita apokopis
% n -> I taxi tou filtrou
% M-> I diastasi tou filtrou
% filtOut -> I perigrafi tou idanikou vathiperatou filtrou sto pedio tis
```

```
    filtOut=zeros(M,M);

    for u1=1:M
        for u2=1:M

            D=sqrt((u1-M/2)^2+(u2-M/2)^2);

            D=D/cutOff;

            filtOut(u1,u2)=1/( 1+D^(2*n) );


        end
    end

     surf(filtOut)

end
```

Chapter 3

Design of Directional Filters

The first step for the creation of a direction filter is to create a mask that follows the correct direction. For that reason a function was created that creates this mask for input the size of the mask, θ and ϕ.

```
function [ filtOut ] = directional_funciton( M,theta,phi )


    filtOut=zeros(M,M);

    l1=tand(theta-phi/2);
    l2=tand(theta+phi/2);

    M2=round(M/2);

    for x=-M2-1:M2+1

        y1=round(l1*x);
        y2=round(l2*x);

        if y1<y2
            ymin=y1;
            ymax=y2;
        else
```

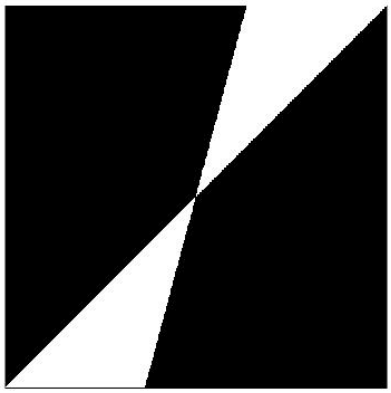
```
            ymin=y2;
            ymax=y1;
        end

        for yy=ymin:ymax
            if M2-yy>0 && M2+x>0 && M2-yy<M && M2+x<M
                filtOut(M2-yy,M2+x)=1;
            end
        end

    end

    imshow(filtOut);

end
```

It is important to note that a transformation of the system is mandatory in order to get the correct results and the correct shape,

$$x' = \frac{M}{2} - y$$

$$y' = \frac{M}{2} + x$$

Then the process is straight forward, the corresponding filter is multiplied pixel by pixel with the mask in order to create a filter only in the correct positions that will recreate this exact shape.



This mask is been used in every possible filter from pass to high pass and band pass filters.