1. Implement a blurring filter using the equation (5.6-11，数字图像处理（第三版）)

in textbook, and blur the test image ‘book\_cover.jpg’ using parameters a=b=0.1 and

T=1. (20%)

**算法描述和文档：**

图像乘以(-1)^(x+y) 中心变换

图像傅里叶变换

根据5.6-11生成blurring filter H是运动模糊退化函数

图像退化/图像模糊

DFT反变换后取实部

图像乘以(-1)^(x+y) 反中心变换

**具体代码和解释：**

src=imread('book\_cover.jpg');

[m,n]=size(src);

src=double(src);

[N,M]=meshgrid(1:n,1:m);

P=m/2;

Q=n/2;

%频率从0开始而不是1 (-1)^(x+y)中心变换后，频率(0,0)位于傅里叶变换后中心

U=M-1-P; V=N-1-Q;

%图像乘以(-1)^(x+y) 中心变换

src\_trans=src.\*(-1).^(M+N);

% fft2 傅里叶变换

FF = fft2(src\_trans);

%根据5.6-11生成blurring filter H是运动模糊退化函数

T=1.0;

a=0.1;

b=0.1;

K=U.\*a+V.\*b;

S=sin( pi.\*K );

E=exp( (-1i\*pi) .\* K );

H= T ./ ( pi.\* K ) .\* S .\* E;

sumzeros=(find(K==0));

H(sumzeros)=T;

%图像退化/图像模糊

Gfourier=H.\*FF;

%DFT反变换后取实部

Gdegeneration=ifft2(Gfourier); G=real(Gdegeneration);

%图像乘以(-1)^(x+y) 反中心变换

G=G.\*(-1).^(M+N);

2. Add Gaussian noise of 0 mean and variance of 500 to the blurred image. (10%)

**算法描述和文档：**

生成正态分布随机数，当做高斯噪声，加到运动模糊图像。

**具体代码和解释：**

%生成正态分布随机数，当做高斯噪声，加到运动模糊图像

avg=0; derivation=sqrt(500);

gaussian\_noise=normrnd(avg,derivation,m,n);

%noise是运动模糊后图像的时域表达

noise=G+gaussian\_noise;

3. Restore the blurred image and the blurred noisy image using the inverse filter.

(30%)

**算法描述和文档：**

图像乘以(-1)^(x+y) 中心变换

图像傅里叶变换

生成inverse filter

对图像进行逆滤波

DFT反变换后取实部

图像乘以(-1)^(x+y) 反中心变换

**具体代码和解释：**

%对运动模糊图像进行逆滤波，此时运动模糊图像是时域而不是频域

Gbeforeinv=G.\*(-1).^(M+N);

GFtrans=fft2(Gbeforeinv);

Gafterinv=GFtrans./H;

Gafterinv( find(abs(H)<0.001) )=0;

Gafterinv=ifft2(Gafterinv); Gafterinv=real(Gafterinv);

Gafterinv=Gafterinv.\*(-1).^(M+N);

%加高斯噪声后的图像逆滤波

D0=20;

Butterworth=1./ ( 1 + ( Dis ./ D0 ) .^2 ) ;

Dis=sqrt((M-P).^2 + (N-Q).^2);

threshold=find(Dis>70);

Hinverse=1./H;

%乘以(-1)^(x+y) 中心变换

noise\_trans=noise.\*(-1).^(M+N);

% fft2 傅里叶变换

Gnoisefourier = fft2(noise\_trans);

%逆滤波

Gnoiseinverse=Gnoisefourier.\*Hinverse;

Gnoiseinverse( find(abs(H)<0.001) )=0;

Gnoiseinverse=Butterworth.\*Gnoiseinverse;

%DFT反变换后取实部

Gnoiseinverse=ifft2(Gnoiseinverse); Gnoiseinverse=real(Gnoiseinverse);

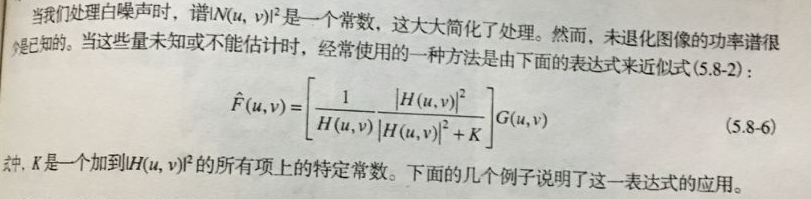
%图像乘以(-1)^(x+y) 反中心变换

Gnoiseinverse=Gnoiseinverse.\*(-1).^(M+N);

4. Restore the blurred noisy image using the parametric Wiener filter with at least 3

different parameters, and compare and analyse results with that of 3. (40%)

**算法描述和文档：**



分别求出H，H的共轭，调参选出最合适的K

**具体代码和解释：**

维纳滤波

Hconj=conj(H); %H共轭

Hsquare=Hconj.\*H; %H^2

Klist=[0.01,0.03,0.05];

for i = 1:3

K=Klist(i);

weiner= Hsquare ./ ( H .\* ( Hsquare + K ) );

weiner\_after = weiner .\* Gnoisefourier;

%DFT反变换后取实部

weiner\_after=ifft2(weiner\_after); weiner\_after=real(weiner\_after);

%图像乘以(-1)^(x+y) 反中心变换

weiner\_after=weiner\_after.\*(-1).^(M+N);

subplot(2,4,4+i);

imshow(weiner\_after,[]);title( sprintf('wiener filter K=%f ', K ));

end

3个参数分别是K=0.01,0.03,0.05

K=0.01时，对噪声的处理较差，对运动模糊的处理较好。

K=0.05时，对噪声的处理较好，对运动模糊的处理较差。

K=0.03时，对噪声和运动模糊的处理效果都不错，综上，K=0.03最好。



所有结果的图像：

