Write a program to examine the effect of quantizing the DCT coefficients. For

each 8 x 8 block in the image, your program should first calculate the DCT of

the block, quantize the coefficients, and then take the inverse DCT. For quantization, use a scaled version of the JPEG quantization matrix as the

stepsizes. You only need to do this for a gray scale image or the luminance

component of a color image. Furthermore, the program should compute the

PSNR of the reconstructed image from quantized DCT coefficients and also

count the total number of non-zero coefficients after quantization and derive

the average number of non-zero coefficients per block. Your program should

show the original and reconstructed image. Examine the resulting image

quality with the following values for the scaling factor: 0.5, 1, 2, 4, 8. What is the largest value of the scaling factor at which the reconstructed image quality is very close to the original image? Please also plot PSNR vs. the

quantization scaling factor, the number of non-zero coefficients/block vs. the

quantization scaling factors, and finally the PSNR vs. the number of non-zero

coefficients. Interpret these plots, to comment on the effect of the quantization

factor on the image quality and bit rate. Note that you may roughly consider

the number of non-zero coefficients to be proportional to the required bits to

represent the quantized image. Hint: you may want to make use of the

”blockproc” function in MATLAB to speed up your program. You can use the

dct2( ) and idct2( ) functions in MATLAB.

close all

clear all

clc

I=imread('lena\_gray.bmp');

Qmatrix=[16 11 10 16 24 40 51 61;

12 12 14 19 26 58 60 55;

14 13 16 24 40 57 69 56;

14 17 22 29 51 87 80 62;

18 22 37 56 68 109 103 77;

24 35 55 64 81 104 113 92;

49 64 78 87 103 121 120 101;

72 92 95 98 112 100 103 99];

A=mycompress1(I,Qmatrix,0.5);

B=mycompress1(I,Qmatrix,1);

C=mycompress1(I,Qmatrix,2);

D=mycompress1(I,Qmatrix,4);

E=mycompress1(I,Qmatrix,8);

F=mycompress1(I,Qmatrix,16);

fun = @(block\_struct) dct2(block\_struct.data);

dctImg = blockproc(modeErr, [8 8], fun);

PSNRa = zeros(1,100);

ka = zeros(1,100);

for q = 1:100

dctImgq = round(dctImg/q)\*q + q/2;

ka(q) = sum(dctImgq(:) ~= q/2);

fun = @(block\_struct) idct2(block\_struct.data);

modeErrq = blockproc(dctImgq, [8 8], fun);

img2q = modePre + modeErrq;

PSNRa(q) = 10\*log10((255\*255)/mean((img2(:) - img2q(:)).^2));

end

figure(3);

plot(ka, PSNRa);

xlabel('K');

ylabel('PSNR');

figure

imshow(I,[])

title('original')

figure

subplot(1,2,1)

imshow(A,[])

title('QP=0.5')

subplot(1,2,2)

imshow(B,[])

title('QP=1')

figure

subplot(1,2,1)

imshow(C,[])

title('QP=2')

subplot(1,2,2)

imshow(D,[])

title('QP=4')

figure

subplot(1,2,1)

imshow(E,[])

title('QP=8')

subplot(1,2,2)

imshow(F,[])

title('QP=16')

function [ Y ] = mycompress1( I,Qmatrix,QP )

QM=Qmatrix\*QP;

fun=@dct2;

A=blkproc(I,[8 8],fun);

fun=@(x)(floor((x+QM/2)./(QM)));

B1=blkproc(A,[8 8],fun);

B2=blkproc(B1,[8 8],'x.\*P1',QM);

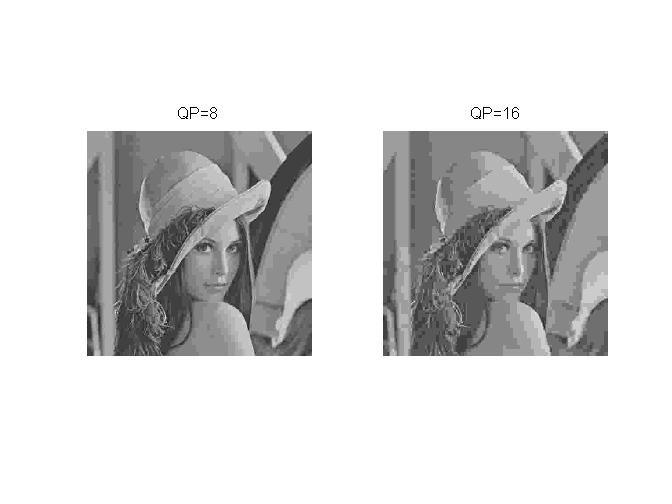
fun=@(x)(round(idct2(x)));

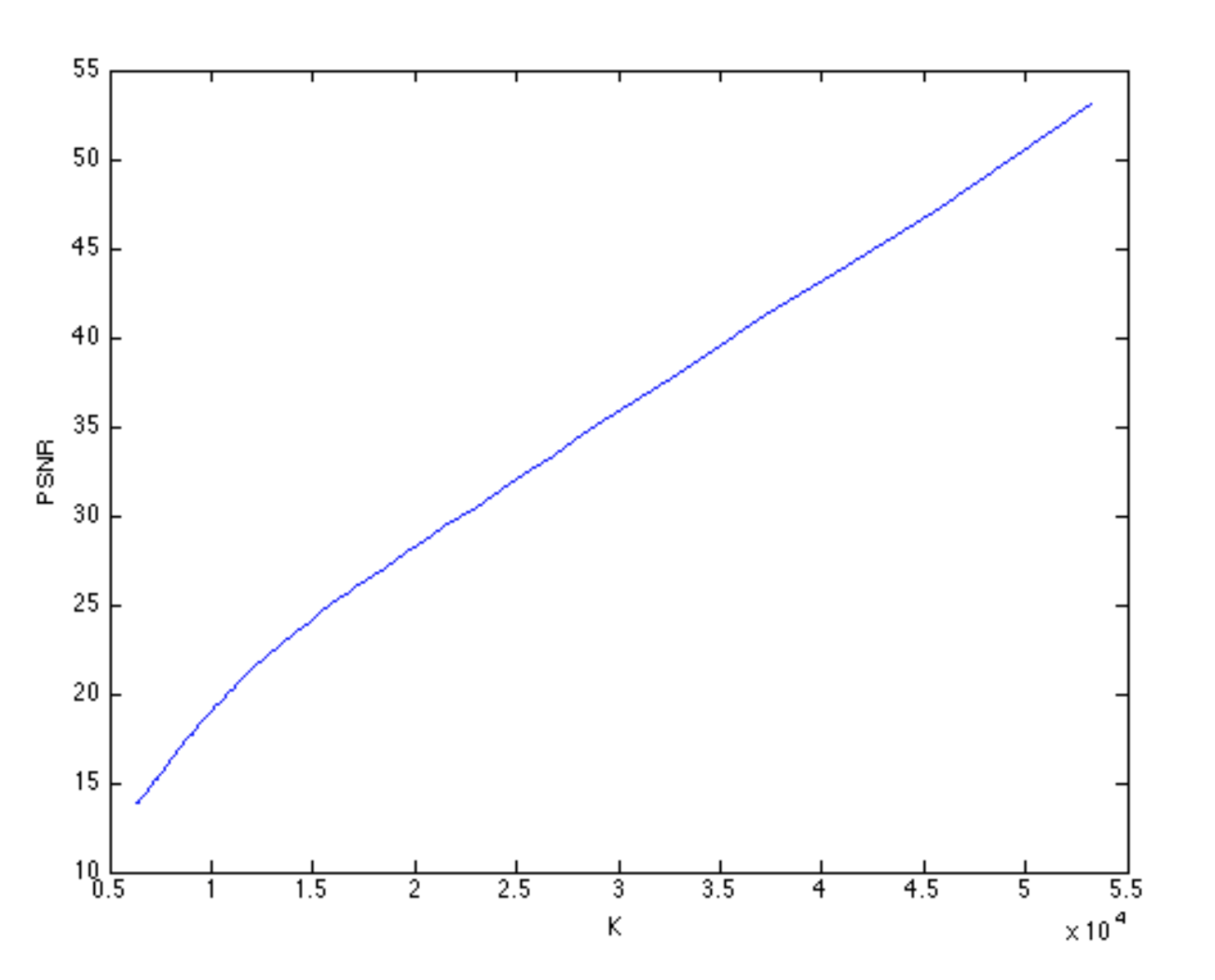
B3=blkproc(B2,[8 8],fun);

Y=B3;

end







**Therefore, the largest value of the scaling factor is 2 at which the reconstructed image quality is very close to the original image.**