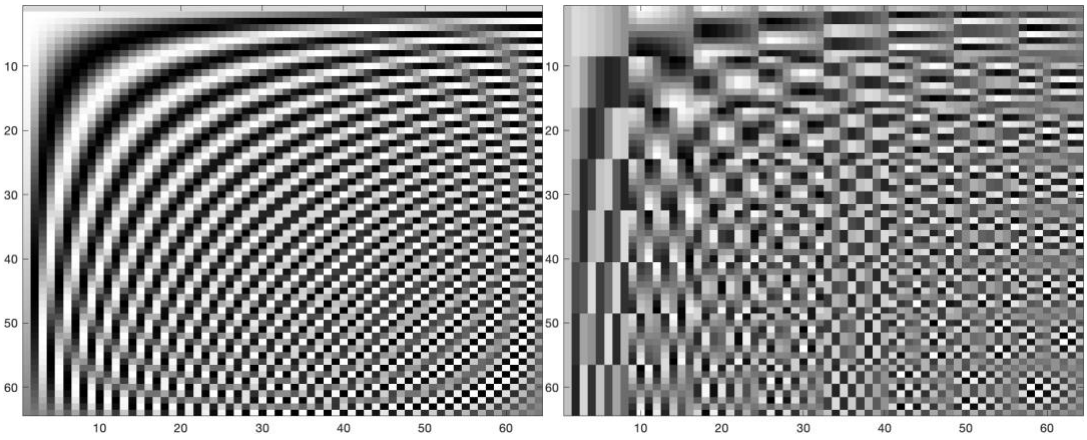
ECE 580 Mini Project Peer Review
Sanika Gupte

Questions moving forward:
1.) Does my basis function look correct?
I implemented the basis function using the formula given but I also found the matlab function dctmtx and the output image looked different (dctmtx on left and my image on right)



2.) How many times should we run 4-fold cross validation? Is it 20 for this project? How does changing that parameter affect the MSE? Why 4 folds?

3.) Would just looking for the minimal MSE be effective? In the homework, the MSE for each alpha didn't look very smooth. Would it be good to smooth out the signal and then look for the minimum?

4.) What range of lambdas should we try/ what range is effective?

Code:

```
%% Mini Project 1 - Compressed Sensing with LASSO

%% Read the images and display one
nature = imread("nature.bmp");
fishing_boat = imread("fishing_boat.bmp");
image = fishing_boat;
imshow(image);

%% Get the user to input the block size
validBlockSize = 0;
while(~validBlockSize)
    k = input("Input block size: ");
    [xSize, ySize] = size(image);
    xBlocks = xSize/k;
    yBlocks = ySize/k;
    if mod(xSize,k) == 0 && mod(ySize,k) == 0
        validBlockSize = 1;
    else
        disp("Invalid Block Size.")
    end
end

%% Put each block into an array
blockArray = zeros(k,k,xBlocks*yBlocks);
blockIncrement = 1;
for xblockNum = 1:xBlocks
    for yblockNum = 1:yBlocks
        blockArray(:,:,blockIncrement) = image(k*(xblockNum-
1)+1:k*xblockNum,k*(yblockNum-1)+1:k*yblockNum);
        blockIncrement = blockIncrement + 1;
    end
end

% %% Reconstruct Image from blocks to make sure you did this correctly
% newImage = zeros(xSize,ySize);
% blockIncrement = 1;
% for xblockNum = 1:xBlocks
%     for yblockNum = 1:yBlocks
%         newImage(k*(xblockNum-1)+1:k*xblockNum,k*(yblockNum-
1)+1:k*yblockNum) = blockArray(:,:,blockIncrement);
%         blockIncrement = blockIncrement + 1;
%     end
% end
% figure;imshow(newImage/max(max(newImage)))
%% Sample a random S pixels

s = input("How many pixels do you want to sample (max is " + k^2 +")? ");
% Don't go over k^2 pixels
if s > k^2
    s = k^2;
end

% Estimating -255
```

```
randPixels = -255*ones(k,k,xBlocks*yBlocks);
for blockNum = 1:xBlocks*yBlocks
    randPixelInd = randperm(k^2, s);
    [randPixelRow, randPixelCol] = ind2sub([k,k],randPixelInd);
    for i = 1:s
        randPixels(randPixelRow(i),randPixelCol(i),blockNum) =
blockArray(randPixelRow(i), randPixelCol(i), blockNum);
    end
    randPixels(:,:,blockNum) = randPixels(:,:,blockNum)/255;
end
imagesc(randPixels(:,:,1)); colormap gray

%% Display sampled image
newImage = zeros(xSize,ySize);
blockIncrement = 1;
for xblockNum = 1:xBlocks
    for yblockNum = 1:yBlocks
        newImage(k*(xblockNum-1)+1:k*xblockNum,k*(yblockNum-1)+1:k*yblockNum)
= randPixels(:,:,blockIncrement);
        blockIncrement = blockIncrement + 1;
    end
end
figure; imshow(newImage/max(max(newImage)));

%% Create the T matrix
% Sample the T matrix by rows --> A matrix
%   If a pixel is -1, then delete that row

P = k;
Q = k;

T = zeros(k*k);
colCounter = 0;

for u = 1:k
    for v = 1:k
        % Updates on every change in u and v
        colCounter = colCounter + 1;
        T_col = zeros(P,Q); % Actually a matrix but it will be reshaped
        if u == 1
            a = sqrt(1/P);
        else
            a = sqrt(2/P);
        end
        if v == 1
            b = sqrt(1/Q);
        else
            b = sqrt(2/Q);
        end

        % Calculate the values of each column and then concatenate them all
        % together into one big T matrix
        for x = 1:k
            for y = 1:k
                T_col(x,y) = a*b*cos(pi*(2*x-1)*(u-1)/(2*P))*cos(pi*(2*y-
1)*(v-1)/(2*Q));
```

```
            end
        end
        T_col = reshape(T_col,[P*Q,1]);

        T(:,colCounter) = T_col;
    end
end

%% Sample T and C for a block to get A and B
A_mats = zeros(s, k*k, blockNum);
B_mats = zeros(s, 1, blockNum);
for blockIndex = 1:blockNum
    block = randPixels(:,:,i);
    pixCol = reshape(block, [k*k,1]);
    missingPicIndices = find(pixCol == -1);

    % Delete the rows you don't want from T
    T_temp = T;
    T_temp(missingPicIndices,:) = [];
    A_mats(:,:,blockIndex) = T_temp;

    % Delete the rows you don't want from C
    pixCol(missingPicIndices) = [];
    B_mats(:,1,blockIndex) = pixCol;

end

%% Run Lasso Regression for a block
gamma = lasso(A_mats(:,:,1),B_mats(:,:,1));


% %% DCT Coefficients
% blockDCT = zeros(k,k,xBlocks*yBlocks);
% for blockNum = 1:xBlocks*yBlocks
%     blockDCT(:,:,blockNum) = dct(randPixels(:,:,blockNum),[],2);
% end
%
% %% Reshape into a 1D column array
% gammaArray = zeros(k*k,1,xBlocks*yBlocks);
% for blockNum = 1:xBlocks*yBlocks
%     gammaArray(:,:,blockNum) = reshape(blockDCT(:,:,blockNum),[k*k,1]);
% end
```