Problem 2:

2. (10%) What is compressive sensing? Give an example of a compressive sensing application and explain how it works.

Ans:

Compressive sensing in the classical sense is the ability to recover a signal with less samples than specified from the Nyquist Sampling Theorem. Central to compressive sensing is an incoherence between a sensing matrix and a representation matrix that allows for the above fore-mentioned recovery of the original signal. Incoherency between “matrix spaces” is best achieved when one of the spaces is sparse. Incoherency has been studied and shown to exist in a practical setting when the sensing matrix is random: Gaussian or Bernoulli. Remarkably, combining the representation matrix with the sensing matrix preserves the randomness of the original matrix. Finally, the computations required to solve the matrix inverse problem are usually of the form: (O) = k\*log(N), which is a very scalable solution for large data sets as N grows.

There are many applications, with MRI being one of the early raison d’etre for development. However, there is one, sparse Fourier that lends itself to a truly transformative technology that incorporates the spirit of compressive sensing.

Below are two uses of sparse Fourier transforms:

Chart

Description automatically generated with low confidence

While FFTs developed in the 1960s was a computational break thru for digital signal processing as it allowed discrete Fourier transforms to be computed in N\*Log N, this is still in polynomial time as opposed to N^2 for DFTs. With the advent of compressive sensing computations can now be reduced to k\*log K in best cases.

How do they work?

Diagram, timeline

Description automatically generated

First, does sparse sensing use the ideas of compressive sensing in its implementation? The two main ideas in compressive sensing, are sparsity and incoherency between representation and sensing spaces. The sparse Fourier meets our first criteria is that it is sparse or can be made sparse thru another basis function. The second incoherency is achieved by randomness of measurement.

The basic idea of the Sparse Fourier Transform is to perform a normal FFT at a lower sample rate, and in the process aliasing many of the signals of interest. However thru a co-prime sub-sampling scheme signals that are unique will collide and those that do not are really aliases that can be eliminated. With “random sampling” and applying the FFT many times the true signals will remain, with a large reduction in processing proportional to log K, where K is the number of signals.

Problem 3:

3. (30%) Based on the lecture demo on compressive imaging, use MATLAB to simulate the computations of a single pixel camera, which uses DMD to generate the random sensing matrix ( ∅𝑖,𝑗∈{0,1}).

Ans:

Here we used the code provided by Prof Liang and Romberg’s L1 optimization. We also use a coherence function that calculates the mutual incoherence of the matrix. While the RIP (Restricted Isometry Property) and the Null Space Property (NSP) provide guarantees for the recovery of a k-sparse signal, they are not easy to compute. The mutual coherence of a matrix, with a lower number meaning that the matrices, sensing, and representation matrices are more incoherent and gives a good recovery of k sparse vectors.

%---------------------------------------------------------------------

% file name : hmwk\_4\_prob\_3\_dmd\_simulation.m

% Student: Ray Duran

% Date: 10/16/21

% Class : EECS 590 Professor Liang, Fall Semester

% University of North Dakota

% Descr:

% Model a Digital Micro-Mirror Device(DMD) with a Bernouli Random Matrix

%

%

% Code Borrowed from :

% compressive\_sensing\_demo

% Use "L1 magic" by Justin Romberg.

% http://www.acm.caltech.edu/l1magic

% Download the L1 magic toolbox and add to the matlab dir

%---------------------------------------------------------------------

A = imread('cameraman.tif');

M = 50; %Downsample the image to 50x50

N = 50;

A = imresize(A,[M N]);

x = double(A(:));

n = length(x);

%\_\_\_MEASUREMENT MATRIX\_\_\_

m = 1000; % Number of samples

%Phi = randn(m,n); % Generate Gaussian distribute random numbers

p = 0.2;

Phi = binornd(1,p,m,n); % Generate Bernouli distribute random numbers

%\_\_\_COMPRESSION\_\_\_

y = Phi\*x;

%\_\_\_THETA\_\_\_

% NOTE: Avoid calculating Psi (nxn) directly to avoid memory issues.

Theta = zeros(m,n);

for ii = 1:n

ek = zeros(1,n);

ek(ii) = 1;

psi = idct(ek)';

Theta(:,ii) = Phi\*psi;

end

% Calculate coherence

u = mutual\_coherence(Theta);

lower\_bound\_on\_mutual\_coherence = ((N-M)/(M\*(N-1)))^0.5;

%\_\_\_l2 NORM SOLUTION\_\_\_ s2 = Theta\y; %s2 = pinv(Theta)\*y

s2 = pinv(Theta)\*y;

%\_\_\_BP SOLUTION\_\_\_

s1 = l1eq\_pd(s2,Theta,Theta',y,5e-3,20); % L1-magic toolbox

%\_\_\_IMAGE RECONSTRUCTIONS\_\_\_

x1 = zeros(n,1);

for ii = 1:n

ek = zeros(1,n);

ek(ii) = 1;

psi = idct(ek)';

x1 = x1+psi\*s1(ii);

end

%\_\_\_\_\_ Plots

coherence\_str = (['Coherence =' num2str(u)]);

figure('name','Compressive sensing image reconstructions')

subplot(1,2,1), imagesc(reshape(x,M,N)), xlabel('original'), axis image

subplot(1,2,2), imagesc(reshape(x1,M,N)), xlabel('basis pursuit'), axis image

str = sprintf( 'Digital Micro-Mirror Device Single Pixel Camera Simulation \n mututal coherence w/ Bernouli random matrix = %4.2f ',u)

sgtitle(str)

%subtitle(coherence\_str)

colormap gray



Problem 5.

* 1. 5. (30%) Lucy-Richardson deconvolution algorithm restores image I that was degraded by convolution with a point-spread function (PSF), and possibly by additive noise. The algorithm is based on maximizing the likelihood that the resulting image is an instance of the original image under Poisson statistics.

a) Use MATLAB to generate a blurred image of an image (attached ***cameraman.tif***) with a 5x5 Gaussian kernel (PSF), and add different levels of gaussian noises (𝜇=0; 𝜎=0,0.0001,𝑎𝑛𝑑 0.001) and test if the Lucy-Richardson deconvolution algorithm can restore the image.

* 1. b) Given the blurred images from a), if you don’t know the PSF. How you can restore the image?

Ans 5a:

%---------------------------------------------------------------------

% file name : hmwk\_4\_prob\_5\_lucy\_deconv.m

% Student: Ray Duran

% Date: 10/24/21

% Class : EECS 590 Professor Liang, Fall Semester

% University of North Dakota

% Descr:

% Blur and add noise to image and try to deconvolve to restore

%

%

% All code Borrowed from Prof Bo Liang in his MATLAB demo,

% from EECS 590

%---------------------------------------------------------------------%%

clear;

close all;

%% Deconvolution

I1 = im2double(imread('cameraman.tif'));

PSF = fspecial('gaussian',5,2);

I2 = imfilter(I1,PSF,'conv'); % blur

sigma\_1 = 0; % noise level

I3 = imnoise(I2,'gaussian',0,sigma\_1); % blur + noise

sigma\_2 = 0.0001;

I4 = imnoise(I2,'gaussian',0,sigma\_2);

sigma\_3 = 0.001;

I5 = imnoise(I2,'gaussian',0,sigma\_3);

% Note: Matlab recommends using the

% imgaussfilt function in place of fspecial

% but since we are going to apply a deconvolution

% algorithm we will need the PSF( impulse function)

% and that is easier to use with the fspecial.test imgauss needs a positive sigma

%Example use of imgauss:

% IT2 = imgaussfilt(IT1,0.01,'FilterSize',5);

figure(1)

subplot(1,2,1)

imshow(I1);

title('Original Image')

subplot(1,2,2)

imshow(I2);

title('Blurred Image')

figure(2)

subplot(1,3,1)

imshow(I3);

title('Blurred Image w/ Gaussian noise sigma = 0')

subplot(1,3,2)

imshow(I4);

title('Blurred Image w/ Gaussian noise sigma = 0.0001')

subplot(1,3,3)

imshow(I5);

title('Blurred Image w/ Gaussian noise sigma = 0.001')

%% 3. Lucy-Richardson method

I = I5;

I\_f5 = deconvlucy(I,PSF,5);

figure(3)

subplot(1,3,1)

imshow(I1);

title('Original Image')

subplot(1,3,2)

imshow(I);

title('Blurred Image w/ Gaussian noise sigma = 0.001')

subplot(1,3,3)

imshow(I\_f5);

title('Image REconstruct Using Lucy-Richardson iter = 5')

I\_f6 = deconvlucy(I,PSF,20);

figure(4)

subplot(1,3,1)

imshow(I1);

title('Original Image')

subplot(1,3,2)

imshow(I);

title('Blurred Image w/ Gaussian noise sigma = 0.001')

subplot(1,3,3)

imshow(I\_f6);

title('Image REconstruct Using Lucy-Richardson iter = 20')

%% 4. Blind deconvolution

I = I5;

WT = zeros(size(I));

WT(5:end-4,5:end-4) = 1;

INITPSF = ones(size(PSF));

sigma = 2;

% [J, P] = deconvblind(I,INITPSF,20);

[J, P] = deconvblind(I,INITPSF,20,10\*sqrt(sigma),WT);

figure(5)

subplot(221);imshow(I);

title('A = Blurred and Noisy');

subplot(222);imshow(PSF,[]);

title('True PSF');

subplot(223);imshow(J);

title('Deblurred Image');

subplot(224);imshow(P,[]);

title('Recovered PSF');



Original Image with convolution blurring.



Blurring + varying gaussian noise added.



Deconvolution reconstruction using Lucy-Richardson algorithm with “ 5” iterations. The picture is still noisy, and a bit blurred but there are no other artifacts.



The picture is less blurry here, and noisy but the cost of the deblurring is to introduce more artifacts manifested as square horizontal and vertical lines around image.

Answer 5b:

If we are not given the PSF or kernel of the mixing of the original image and we are still trying to solve the inversion problem this is considered a blind deconvolution. In signal processing this is like independent component analysis that tried to solve the “cocktail problem” of unknown mixing and trying to solve for the original signals at the same time. Here we employ a Maximum posterior-based approach that uses bayes rule with a kernel here that is initially guessed at and then iterated as an optimization problem.

