Compressed Sensing from Several Sets of Downsampled Fourier Values using Only FFTs

Andrew E. Yagle
Department of EECS, The University of Michigan, Ann Arbor, MI 48109-2122

Abstract— **Reconstruction of signals or images from a few Discrete Fourier Transform (DFT) values has applications in MRI and SAR. Compressed sensing is the reconstruction from a reduced set of observations of a signal or image that can be sparsified. Many real-world signals and images may be sparsified by convolution with a differencing operator, such as a wavelet; this multiplies the given DFT values by its known frequency response. We present a simple procedure for reconstructing the signal or image from a few sets of downsampled DFT values, using only the Fast Fourier Transform (FFT) algorithm and some multiplications. Three examples and Matlab programs are provided.**

Keywords— **Sparse reconstruction**
Phone: 734-763-9810. Fax: 734-763-1503.
Email: aey@eecs.umich.edu. EDICS: 2-REST.

I. INTRODUCTION

A. Problem Statement

The N-point DFT X_k of the length=N signal x_n is

$$X_k = \sum_{n=0}^{N-1} x_n e^{-j2\pi nk/N}, k = 0 \dots N-1. \quad (1)$$

We assume that x_n is sparsifiable, meaning that:
- There exists a sparsifying function ψ_n such that:
- z_n=$\sum_{i=0}^{N-1} x_i \psi_{(n-i) \bmod (N)}$ is K-sparse, meaning:
- z_n=0 unless $n \in \{n_1 \dots n_K\}$ (the n_i are unknown)

DFT Z_k=$X_k \Psi_k$ is known for three sets of values of k:

- $k \in \{L_1, 2L_1, 3L_1 \dots N-L_1\}$
- $k \in \{L_2, 2L_2, 3L_2 \dots N-L_2\}$
- $k \in \{L_3, 2L_3, 3L_3 \dots N-L_3\}$
- L_1, L_2, L_3 are all factors of N.
- L_1, L_2, L_3 are pairwise relatively prime.
- The DC value X_0=$\sum_{n=0}^{N-1} x_n$ is also known.

Otherwise, knowledge of X_k and of Z_k is equivalent.

The goal is to compute the K-sparse signal z_n, and then x_n, from the known DFT values X_k Note half of the given DFT values are complex conjugates of the other half by conjugate symmetry, if x_n is real.

B. Problem Significance

Many signals and images of practical interest have a sparse representation in a wavelet basis. Computation of the wavelet transform can be viewed as convolutions with scaled wavelet and scaling basis functions, which becomes multiplication in the DFT domain. Hence the results of this paper apply to both sparse signals and wavelet-sparsifiable signals.

A common approach to sparse reconstruction is to compute the minimum ℓ_1 norm solution, perhaps by linear programming. If the DFT frequencies are randomly chosen, and if enough of them are known, then it has been shown that the minimum ℓ_1 norm solution is in fact z_n. In many practical situations we do not have the luxury of choosing the k_i at random-they are pre-specified. And the number M of Z_{k_i} required is $O(K \log N)$ (the exact number is unknown).

Other approaches include thresholded Landweber iteration and orthogonal matching pursuit. These are much faster computationally, but require more of the problem in order to compute z_n. Since our approach is completely different from all of these, we refer the reader to the extensive literature on these methods.

The algorithm presented in this paper requires $O(KN^{1/3})$ observations of DFT values. This is more than the minimum ℓ_1 norm solution requires, but the DFT frequencies need not be random, and only the FFT algorithm is required. The locations $\{n_i\}$ of unknown values of z_n (along with some false alarm locations) can be recovered in closed form using three inverse DFTs, computed using the FFT. Reconstruction of z_n from the given Z_k and computed $\{n_i\}$ can be computed using an alternating-projections POCS algorithm, which again requires only FFTs. Finally, x_n is reconstructed from z_n using FFTs in X_k=$\frac{Z_k}{\Psi_k}$

II. Presentation of Algorithm

A. Replacing Unknown Z_k with Zeros

Given data $Z_k, k \in \{0, L, 2L, 3L \dots N–L\}$, define

$$Y_k = \begin{cases} Z_k & \text{for } k = 0, L, 2L \dots N-L \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The inverse N-point DFT of Y_k is then

$$y_n = \frac{1}{L} \sum_{i=0}^{L-1} z_{n+iN/L}. \quad (3)$$

So inserting zeros for the missing values of Z_k and computing the inverse N-point DFT gives L copies

of the nonzero values of z_n, each shifted in n by an integer multiple of N/L.

The easiest way to derive this result is to note

$$\begin{aligned} Y_k &= \sum_{n=0}^{N-1} y_n e^{-j2\pi nk/N} \\ &= \frac{1}{L}\sum_{n=0}^{N-1}\sum_{i=0}^{L-1} z_{n+iN/L} e^{-j2\pi nk/N} \\ &= \frac{1}{L} Z_k \sum_{i=0}^{L-1} e^{j2\pi(iN/L)k/N} \\ &= \begin{cases} Z_k & \text{for } k = 0, L, 2L \ldots N-L \\ 0 & \text{otherwise} \end{cases} \end{aligned} \quad (4)$$

This uses the time delay property of the DFT and the sum of L equally-spaced points on the unit circle is zero. But if L divides k, then $e^{j2\pi(iN/L)(k/N)}$=1.

B. Indicator Function

Repeat the above procedure for each of the three sets of values of Z_k. This gives

$$\begin{aligned} y_{1n} &= \sum_{i_1=0}^{L_1-1} z_{n+i_1 N/L_1}. \\ y_{2n} &= \sum_{i_2=0}^{L_2-1} z_{n+i_2 N/L_2}. \\ y_{3n} &= \sum_{i_3=0}^{L_3-1} z_{n+i_3 N/L_3}. \end{aligned} \quad (5)$$

Then define the indicator function

$$y_n = y_{1n} y_{2n} y_{3n}. \quad (6)$$

The i_j=0 terms in (5) show that $y_{n_i} \neq 0$. But y_n for $n \neq n_i$ will be zero unless all three of $y_{in} \neq 0$. If N is large and z_n is sparse enough, this is unlikely, so y_n is an indicator for locations $\{n_i\}$ of nonzero z_n.

This concept has some similarity to harmonic product spectrum, as used for musical pitch detection, except that the time and frequency domains are exchanged, and shifted copies of z_n are used instead of harmonics of musical pitch frequencies.

If z_n is very sparse, then there will be no overlaps among the $\{y_{in}\}$ for $n \notin \{n_i\}$. Then

$$y_n = \begin{cases} z_n^3 & \text{for } n \in \{n_i\} \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

So not only is y_n an indicator function for locations of nonzero z_n, but the values of z_n can be recovered from the signed cube root of y_n (e.g., $\sqrt[3]{-8}$=–2). The following example shows closed-form reconstruction of x_n directly from X_k is possible using five FFTs.

C. Example #1: Closed-Form Reconstruction

The image shown in Figs. 1,2 consists of 4 cylinders of various heights on a 315×315 plane. The data are the 315×315 2-D DFT, downsampled by factors of 3,5,7, leaving the locations shown in Fig. 3. The problem has 8141 observations (excluding complex conjugates) in 315^2=99225 unknowns, so the problem is underdetermined by a factor of more than 12.

The sparsifying function used is $\psi_{n_1,n_2}=\begin{bmatrix}1\\-1\end{bmatrix}$. This is the Haar wavelet in the vertical direction. The *unknown* sparsified image is shown in Fig. 4. This image is 152-sparse, so K=152 (also unknown).

The indicator function y_n was computed by multiplying 3 inverse DFTs, computed using 3 FFTs, and computing the signed cube root of each y_n to get z_n. The result is shown in Fig. 5, and matches Fig. 4.

The formula $X_k=\frac{Z_k}{\Psi_k}$ was then used to compute the reconstructed original image. This required 2 more FFTs, for a total of 5, to compute Z_k from z_n and then to compute x_n from $X_k=\frac{Z_k}{\Psi_k}$. But since the 2-D DFT of $\psi_{n_1,n_2}=\begin{bmatrix}1\\-1\end{bmatrix}$ is $\Psi_{k_1,k_2}=e^{j2\pi k_2/N}$–1, which is zero for all k_1 when k_2=0, the 8141 given 2-D DFT values were augmented with additional $X_{k_1,0}$.

The final reconstructed image is shown in Fig. 6.

Matlab Program for Example #1

```
clear;N1=3;N2=5;N3=7;N=N1*N2*N3*3;
K1=[1:N1:N];K2=[1:N2:N];K3=[1:N3:N];
[x y]=meshgrid(1:N,1:N);%4 cylinders:
X1=((x-130).^2+(y-140).^2)<100;
X2=((x-150).^2+(y-170).^2)<100;
X3=((x-60).^2+(y-40).^2)<100;
X4=((x-90).^2+(y-60).^2)<100;
X=X1+2*X2+1.3*X3+1.8*X4;FI(N,N)=0;
figure,imagesc(X),colormap(gray)
figure,mesh(X),axis tight
FI(K1,K1)=1;FI(K2,K2)=1;FI(K3,K3)=1;
figure,imagesc(FI),colormap(gray)
FX=fft2(X);%Y=X(2:N,:)-X(1:N-1,:);
E=exp(-2j*pi*[0:N-1]/N);O=ones(N,1);
FY=FX.*(O*E).'-FX;%sparse image DFT.
figure,imagesc(real(ifft2(FY))),colormap(gray)
F1=FY(K1,K1);F2=FY(K2,K2);F3=FY(K3,K3);
%SOLUTION STARTS HERE:
G1(N,N)=0;G2(N,N)=0;G3(N,N)=0;
G1(1:N1:N,1:N1:N)=F1;Z1=real(ifft2(G1));
G2(1:N2:N,1:N2:N)=F2;Z2=real(ifft2(G2));
G3(1:N3:N,1:N3:N)=F3;Z3=real(ifft2(G3));
Z=Z1.*Z2.*Z3*N*N/9;Z=(abs(Z)).^(1/3).*sign(Z);
figure,imagesc(Z),colormap(gray)
H=(O*E).'-1;FXHAT=fft2(Z)./H;%Deconvolve H.
FXHAT(N,N)=0;FXHAT(1,:)=FX(1,:);%Known X(0,k).
figure,imagesc(real(ifft2(FXHAT))),colormap(gray)
```

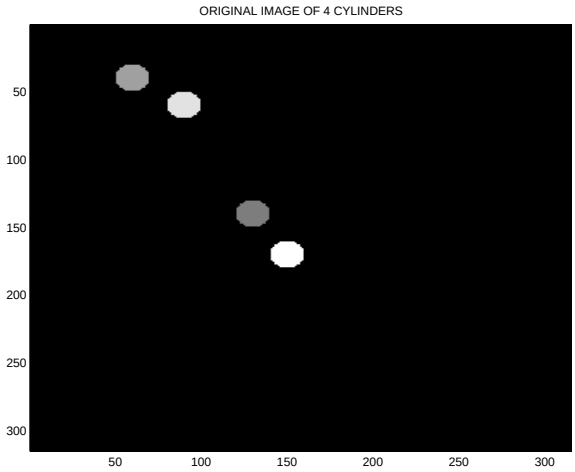

Fig. 1. Original 315×315 Image.

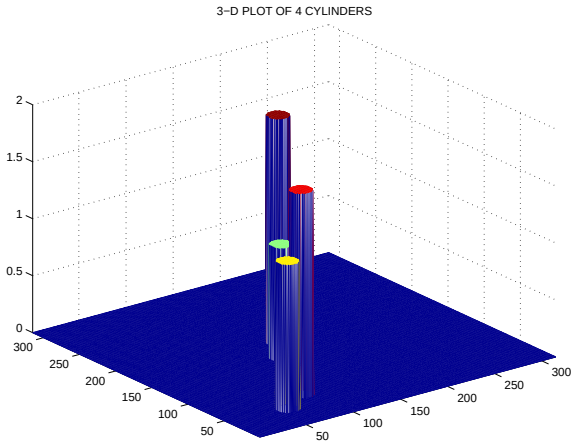

Fig. 2. Original 315×315 Image.

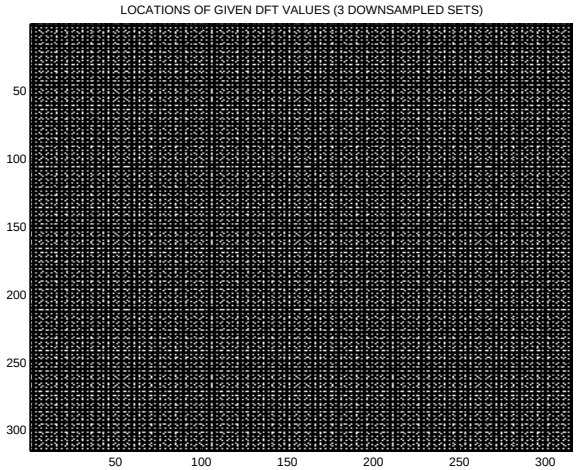

Fig. 3. Locations of Given 2-D DFT Data.

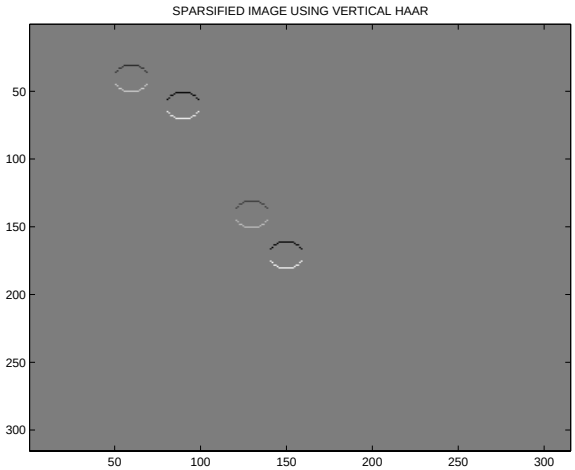

Fig. 4. (Unknown) Sparsified Image.

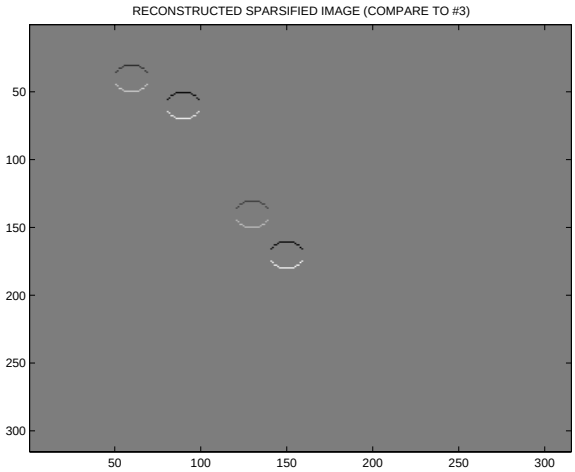

Fig. 5. Reconstructed Sparsified Image.

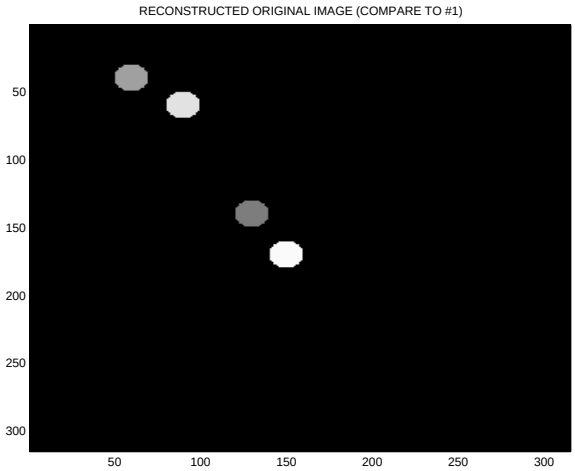

Fig. 6. Reconstructed Original Image.

D. Reconstruction Using POCS

Usually there will be overlaps among the $\{y_{in}\}$ for $n \notin \{n_i\}$. Then $y_n \neq z_n^3$, and it is necessary to reconstruct the sparsified signal z_n from the indicator function y_n. Worse, there will be some "false alarms," i.e., nonzero values of y_n, even though the corresponding z_n=0, because copies of z_n in all three of the y_{in} are all nonzero at some values of n.

There are two ways around this problem. One is to set up and solve a linear system of equations to compute z_n from y_n. The computed solution will have z_n=0 at the "false alarm" locations where $y_n \neq 0$. Matlab code for setting up and solving the system of equations is provided for Example #2 below.

However, it is easier (but more time-consuming) to use a Projection-Onto-Convex-Sets (POCS) algorithm to compute z_n from y_n. The constraints are: (1) the given DFT values Z_k; and (2) the support constraint z_n=0 except at locations where $y_n \neq 0$. Both are convex sets. The two projections for the estimates z_n^K and its DFT Z_k^K at the K^{th} iteration:

- $\mathcal{P}_{\mathcal{A}}\{Z_k^K\} = \begin{cases} Z_k & \text{where known} \\ Z_k^K & \text{otherwise} \end{cases}$
- $\mathcal{P}_{\mathcal{B}}\{z_n^K\} = \begin{cases} 0 & \text{where } y_n = 0 \\ z_n^K & \text{otherwise} \end{cases}$

The POCS algorithm (which is often also called an Alternating Projections (AP) algorithm) is then:

$$z_n^{K+1} = \text{DFT}^{-1}\{\mathcal{P}_{\mathcal{A}}\{\text{DFT}\{\mathcal{P}_{\mathcal{B}}\{z_n^K\}\}\}\}. \quad (8)$$

The algorithm requires no computation but FFTs, and it is guaranteed to converge to the solution.

E. Example #2: Reconstruction Using POCS

The image shown in Fig. 7 consists of five cylinders of various heights on a 210×210 plane. The data are the 210×210 2-D DFT, again downsampled by factors of 3,5,7. The problem has 3618 observations (excluding complex conjugates) in 210^2=44100 unknowns, and is underdetermined by more than 12.

The vertical 2-D Haar wavelet ψ_{n_1,n_2} from Example #1 was again used to sparsify the image. The 190-sparse sparsifed image z_{n_1,n_2} is shown in Fig. 8.

The indicator function y_{n_1,n_2} is shown in Fig. 9. Comparing Fig. 8 and Fig. 9 shows that there are 62 "false alarms," since y_{n_1,n_2} is 252-sparse. To clarify Fig. 9, all of the nonzero y_{n_1,n_2} were set to unity.

A POCS algorithm was used to compute z_{n_1,n_2} from y_{n_1,n_2}. The Frobenius norm (square root of sum of squares) of the difference between successive iterations was used to track convergence, and is plotted in Fig. 10. Each iteration required two FFTs. The convergence was soon linear (each error proportional to the error in the previous iteration), so the norm decreased exponentially after an initial, faster decrease. At this point, the usual acceleration procedures could have been used to greatly speed up POCS.

The reconstructed sparsified image z_{n_1,n_2} is shown in Fig. 11, and the reconstructed original image is shown in Fig. 12. They match the actuals closely.

Alternatively, a 3618×252 linear system of equations could have been solved, instead of using POCS. Matlab code for this is commented out in the Matlab program listed below. Complex conjugates and some duplicate rows brought the size up to 7564×252.

Matlab Program for Example #2

```
clear;N1=3;N2=5;N3=7;N=N1*N2*N3*2;
K1=[1:N1:N];K2=[1:N2:N];K3=[1:N3:N];
[x y]=meshgrid(1:N,1:N);%5 cylinders:
X1=((x-150).^2+(y-150).^2)<100;
X2=((x-160).^2+(y-160).^2)<100;
X3=((x-70).^2+(y-30).^2)<100;
X4=((x-80).^2+(y-20).^2)<100;
X5=((x-80).^2+(y-70).^2)<100;
X=X1+2*X2+1.5*X3+1.3*X4+1.8*X5;
figure,imagesc(X),colormap(gray)
FX=fft2(X);%Y=X(2:N,:)-X(1:N-1,:);
E=exp(-2j*pi*[0:N-1]/N);O=ones(N,1);
FY=FX.*(O*E).'-FX;%Sparsified image DFT.
figure,imagesc(real(ifft2(FY))),colormap(gray)
F1=FY(K1,K1);F2=FY(K2,K2);F3=FY(K3,K3);
%SOLUTION STARTS HERE:
G1(N,N)=0;G2(N,N)=0;G3(N,N)=0;
G1(1:N1:N,1:N1:N)=F1;Z1=real(ifft2(G1));
G2(1:N2:N,1:N2:N)=F2;Z2=real(ifft2(G2));
G3(1:N3:N,1:N3:N)=F3;Z3=real(ifft2(G3));
Z=Z1.*Z2.*Z3*N*N;Z(abs(Z)>0.001)=1;
%Z=indicator function for nonzero Y.
figure,imagesc(Z),colormap(gray)
%AP POCS ALGORITHM: Initialize YHAT=Z:
YH=Z;kmax=200;for k=1:kmax;OY=YH;FH=fft2(YH);
FH(K1,K1)=F1;FH(K2,K2)=F2;FH(K3,K3)=F3;
YH=real(ifft2(FH)).*Z;D(k)=norm(YH-OY,'fro');
end;figure,plot(D) %LINEAR SYSTEM OF EQNS:

%[I J]=find(abs(Z)>0.00001);
%FF=[F1(:);F2(:);F3(:)];%Given data.
%KK1=[kron(ones(1,N/N1),K1-1);kron(K1-1,ones(1,N/N1))];
%KK2=[kron(ones(1,N/N2),K2-1);kron(K2-1,ones(1,N/N2))];
%KK3=[kron(ones(1,N/N3),K3-1);kron(K3-1,ones(1,N/N3))];
%KK=[KK1 KK2 KK3];A=exp(2j*pi/N*([I-1 J-1]*KK));
%YHAT1=A'\FF;YHAT(N,N)=0;%Computed sparsified
%for II=1:length(I);YHAT(I(II),J(II))=real(YHAT1(II));end;
figure,imagesc(YH),colormap(gray)
H=(O*E).'-1;FXH=fft2(YH)./H;%Deconvolve H.
FXH(N,N)=0;FXH(1,:)=FX(1,:);%Known X(0,k).
figure,imagesc(real(ifft2(FXH))),colormap(gray)
```

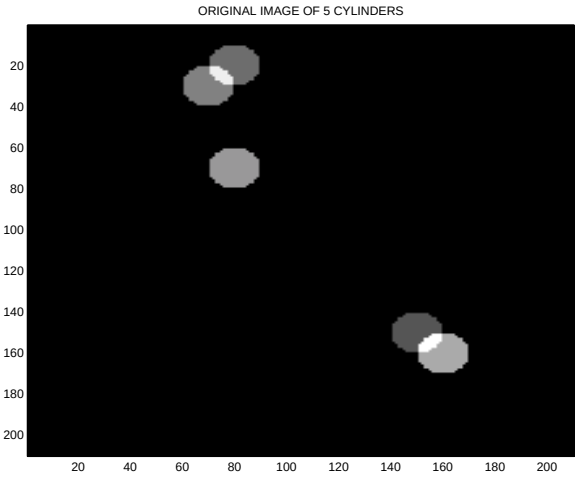

Fig. 7. Original 210×210 Image.

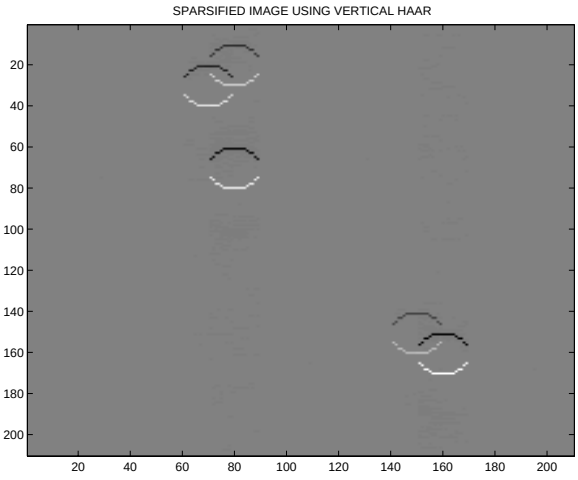

Fig. 8. (Unknown) Sparsified Image.

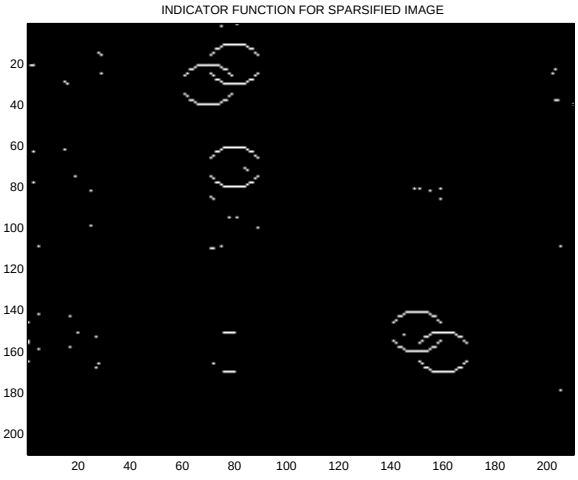

Fig. 9. Computed Indicator Function.

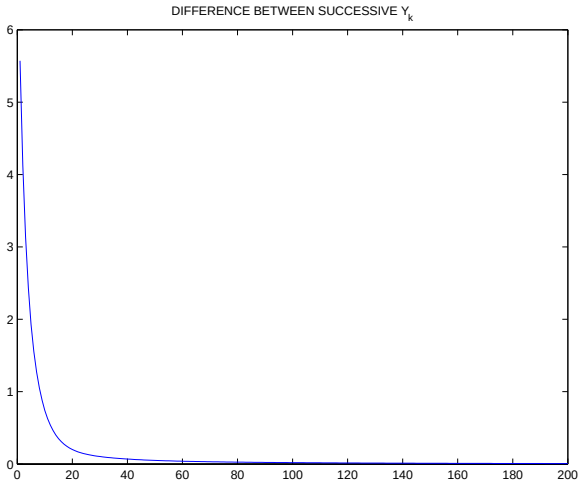

Fig. 10. Convergence of POCS: $||Y^{K+1} - Y^K||$.

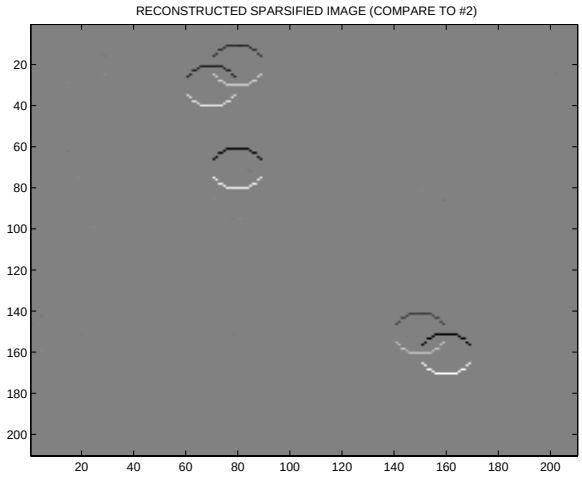

Fig. 11. Reconstructed Sparsified Image.

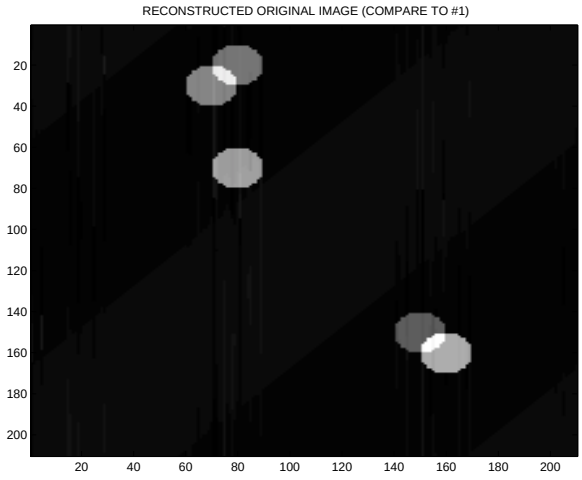

Fig. 12. Reconstructed Original Image.

F. Example #3: Slanted Shepp-Logan Phantom

The image shown in Fig. 13 is the 420×420 Shepp-Logan phantom, multiplied by a 2-D ramp function that varies from 1 at the upper left to 4 at the lower right. So each cylinder now has a slanted top.

The data are the 420×420 2-D DFT, downsampled by factors of 3,4,5,7. The problem has 19080 observations (excluding complex conjugates) in 420^2=176,400 unknowns, so the problem is underdetermined by a factor of more than 9.

The sparsifying function used is now

$$\begin{aligned} \psi_{n_1,n_2} &= [1,-2,1]^T \\ \Psi_{k_1,k_2} &= e^{j2\pi k_2/N} + e^{-j2\pi k_2/N} - 2 \\ &= 2\cos(2\pi k_2/N) - 2 \\ &\approx -(2\pi k_2/N)^2 \end{aligned} \quad (9)$$

for small k_2. This sparsifies linear functions to zero except at endpoints. Note the resemblance to a partial second derivative in the vertical direction. This shows that $\Psi_{k_1,k_2} \approx 0$ for all k_1 when $|k_2|$ is small, not just zero. So the 19080 given 2-D DFT values were augmented with additional X_{k_1,k_2} for $|k_2| \leq 10$.

The *unknown*, 3500-sparse, sparsified image is shown in Fig. 14. Due to the dynamic range of nonzero pixel values, this figure shows the base-10 logarithm of absolute values of the sparsified image.

The computed indicator function y_{n_1,n_2} is shown in Fig. 15. Comparing Fig. 14 and Fig. 15 show that there are 1510 "false alarms," since y_{n_1,n_2} is 5010-sparse. All of the nonzero y_{n_1,n_2} were set to unity.

The system of equations is 19080×5010, so a POCS algorithm was used to compute z_{n_1,n_2} from y_{n_1,n_2}. The Frobenius norm (square root of sum of squares) of the difference between successive iterations was used to track convergence, and is plotted in Fig. 16. Each iteration required two FFTs. The convergence was soon linear (each error proportional to the error in the previous iteration), so the norm decreased exponentially after an initial, faster decrease. At this point, the usual acceleration procedures could have been used to greatly speed up POCS convergence.

The reconstructed sparsified image z_{n_1,n_2} is shown in Fig. 17, and the reconstructed original image is shown in Fig. 18. They match the actuals closely, although some streak artifacts appear in the reconstructed image. The streaks come from the "false alarms" that have not completely decayed to zero, and are amplified by division by Ψ_{k_1,k_2}. Running the POCS algorithm longer eliminates these artifacts.

The POCS algorithm could be accelerated. Another (simpler) approach is to threshold small values of z_{n_1,n_2} to zero. This also eliminates the artifacts.

Matlab Program for Example #3

```
clear;N1=3;N2=4;N3=5;N4=7;N=N1*N2*N3*N4;
K1=[1:N1:N];K2=[1:N2:N];K3=[1:N3:N];K4=[1:N4:N];
X=phantom(N);L=linspace(1,2,N);X=X.*(L'*L);
figure,imagesc(X),colormap(gray)
FX=fft2(X);%Y=X(3:N,:)+X(1:N-2,:)-2*X(2:N-1,:);
E=exp(-2j*pi*[0:N-1]/N);O=ones(N,1);
H=(O*E).'+(O*conj(E)).'-2*(O*O');
FY=FX.*H;Y=real(ifft2(FY));Y(N,N)=0;
figure,imagesc(log10(abs(Y))),colormap(gray)
F1=FY(K1,K1);F2=FY(K2,K2);F3=FY(K3,K3);F4=FY(K4,K4);
%GIVEN: F1,F2,F3,F4.  SOLUTION STARTS HERE:
G1(N,N)=0;G2(N,N)=0;G3(N,N)=0;G4(N,N)=0;
G1(1:N1:N,1:N1:N)=F1;Z1=real(ifft2(G1));
G2(1:N2:N,1:N2:N)=F2;Z2=real(ifft2(G2));
G3(1:N3:N,1:N3:N)=F3;Z3=real(ifft2(G3));
G4(1:N4:N,1:N4:N)=F4;Z4=real(ifft2(G4));
Z=Z1.*Z2.*Z3.*Z4*N*N;Z(abs(Z)>0.00001)=1;
figure,imagesc(Z),colormap(gray)
%AP POCS ALGORITHM: Initialize with YHAT=Z:
kmax=1000;YH=Z;for k=1:kmax;OY=YH;FH=fft2(YH);
FH(K1,K1)=F1;FH(K2,K2)=F2;FH(K3,K3)=F3;
YH=real(ifft2(FH)).*Z;D(k)=norm(YH-OY,'fro');end;
figure,plot(log10(D))
figure,imagesc(log10(abs(YH))),colormap(gray)
ky=10;FXH=fft2(YH)./H;K=[1:ky N+2-ky:N];
FXH(K,:)=FX(K,:);%Additional X(k1,k2)
figure,imagesc(real(ifft2(FXH))),colormap(gray)
```

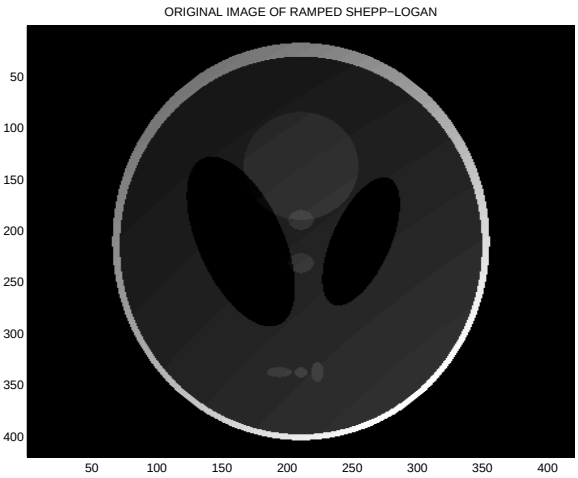

Fig. 13. Original 420×420 Slanted Shepp-Logan.

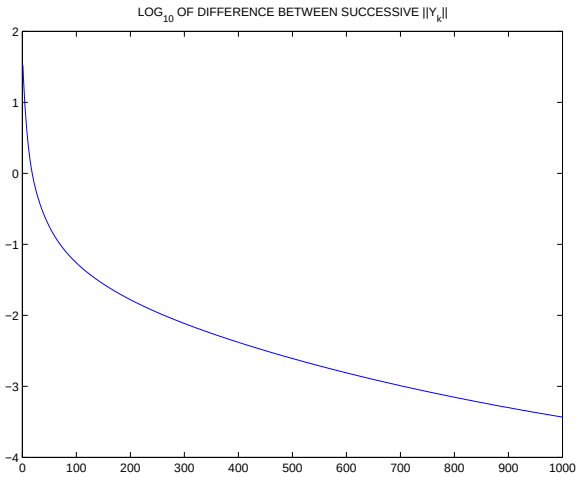

Fig. 16. Convergence of POCS: $\log_{10} ||Y^{K+1} - Y^K||$.

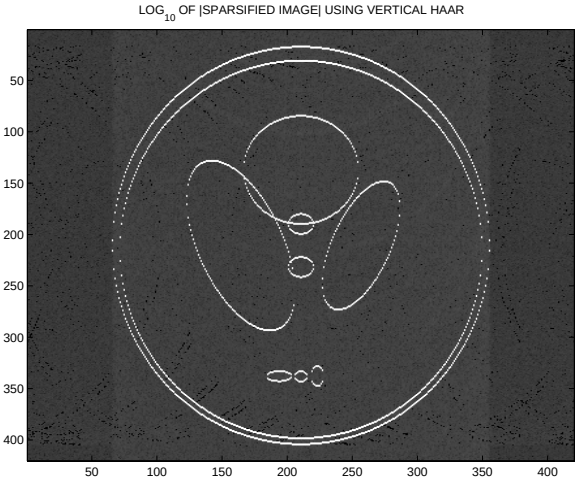

Fig. 14. (Unknown) $\log_{10} ||$Sparsified Image$||$.

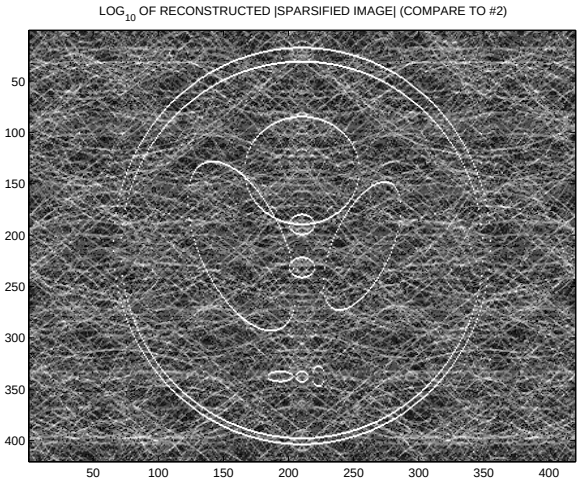

Fig. 17. Reconstructed $\log_{10} ||$Sparsified Image$||$.

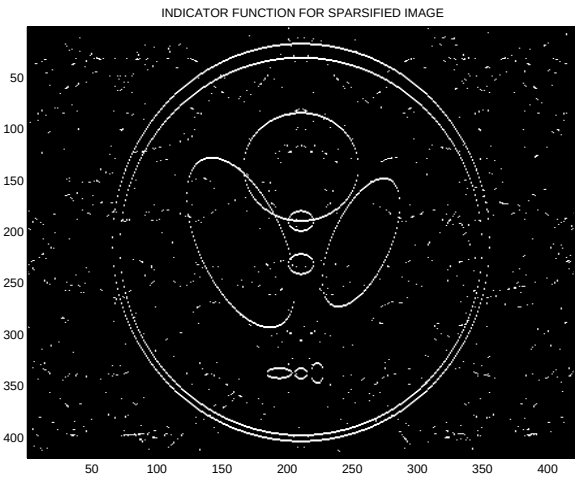

Fig. 15. Computed Indicator Function.

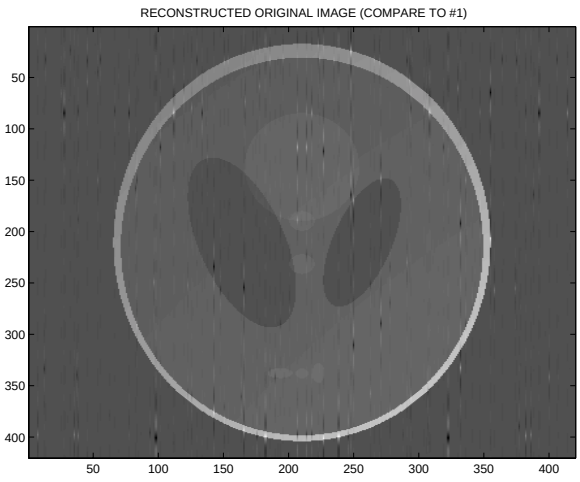

Fig. 18. Reconstructed Original Image.