Lab 2 - Compressive Sensing for biomedical signal

- **Compressive Sensing Tutorial** -***What & Why is CS?***

  How to use it? (on biomedical signals)
- Lab Task 2.1 (need 1 week)

  Learning Goal: Understand Compressive sensing concept; Content: Use a simple transform to see how sparse samples could achieve comparable performance to Nyquist theorem.

- Lab Task 2.2 (need 1 week)

  Learning goal: Use compressive sensing theory to acquire EEG signals. Content: design a random matrix matched with real application to perform compressive sampling.

- Lab Task 2.3 (need 1 week)

  Learning goal: Use L1 optimization to reconstruct ECG/EEG signals.

What & Why is CS?

1. Compressive Sensing Tutorial

Compressive sensing is a technique for finding sparse solutions to underdetermined linear systems. In engineering, it is the process of acquiring and reconstructing a signal utilizing the prior knowledge that the signal is sparse or compressible.

a) Background & Motivation

As the fast development of digital sensor system, we have much more data to store or transmit than before based on higher resolution, large numbers of sensors and increasing numbers of modalities.

In point of energy saving which is considered important in many real applications, how to reduce the data size and keep the good reconstruction results is under nowadays research.

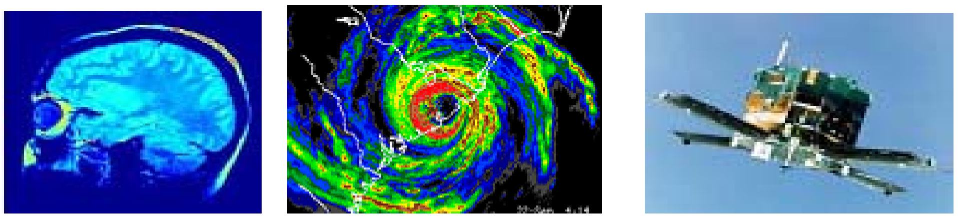


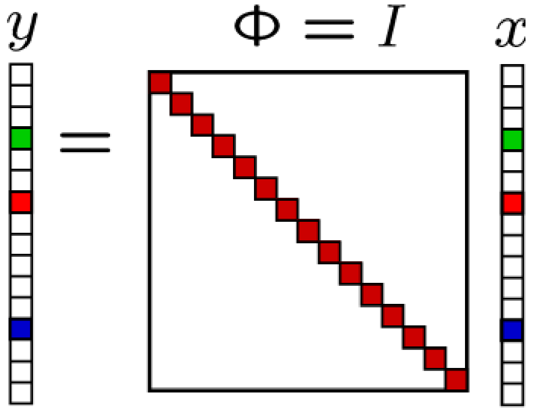
Figure 1 More data is generated by new tech & applications

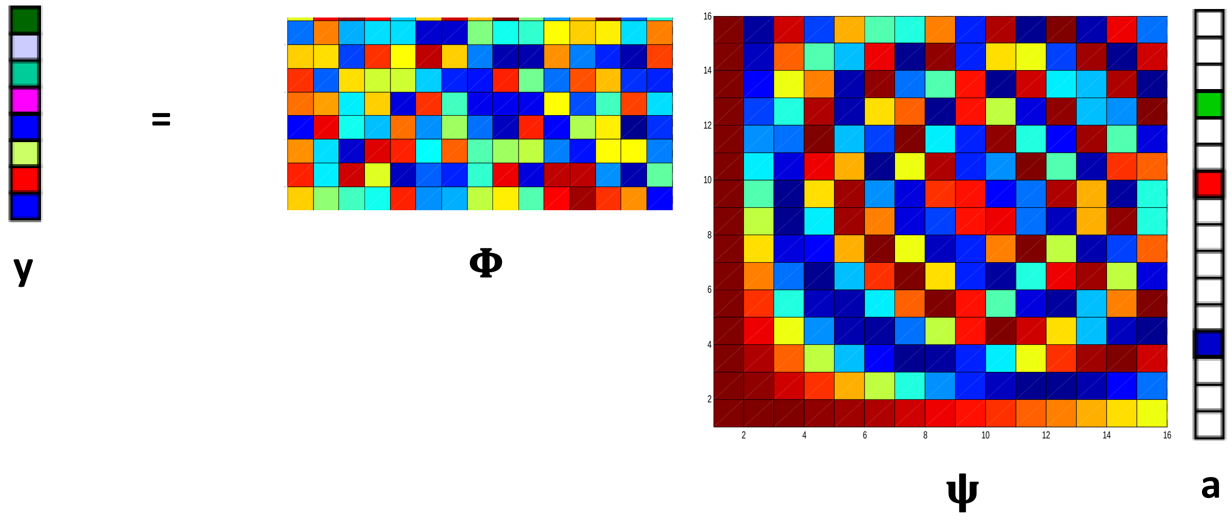
- **Shannon/Nyquist theorem**

– Shannon theorem is much over sampling
– 2x oversampling Nyquist rate is a worst-case bound
  for any bandlimited data
– sparsity/compressibility irrelevant
– Shannon sampling is a linear process while compression is a
  nonlinear process

b) CS and System Setup

Most of the data are sparse or compressible in different domains. When data is sparse or compressible, we can directly acquire a condensed representation with no/little information loss. This one is obtained by a new sensing theory which is based on uncertainty principles.





For the linear system illustrated in the above Figure, we have $y = \Phi x = \Phi \Psi \alpha$, where y is the measurement, x is the signal, is the measurement matrix, is the signal bases, and is sparse representation of signal x using bases. □
□

We surprisingly find out that when **Φ** is random which satisfy with the

R.I.P.

(Restricted Isometry Property) the measurement matrix is far less than signal’s length, the original signal can be reconstructed very well.

$$K < M \ll N$$

and K satisfy

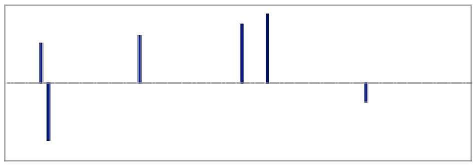
$$K < M/logN$$

□□□□□□□□□□To solve $y = \Phi x = \Phi \Psi \alpha$, we have the non-linear optimal question for

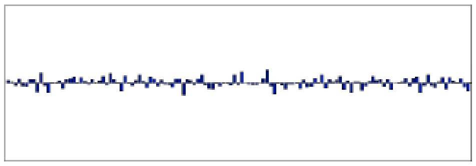
reconstruction.

• ℓ_2	fast, wrong	$\widehat{x} = \arg \min\limits_{y=\Phi x} \|x\|_2$
• ℓ_0	correct, slow	$\widehat{x} = \arg \min\limits_{y=\Phi x} \|x\|_0$
• ℓ_1	**correct, efficient mild oversampling** [Candes, Romberg, Tao; Donoho]	$\widehat{x} = \arg \min\limits_{y=\Phi x} \|x\|_1$ ***linear program***

An example of sparse coefficients of a signal is shown in the following Figure.



x



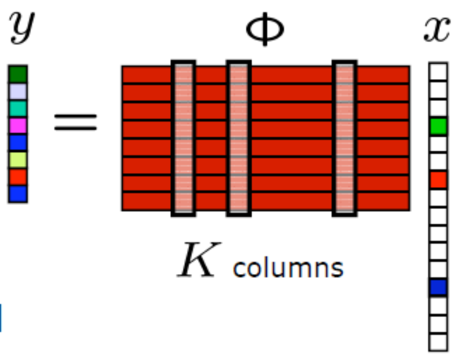
$$\widehat{x} = (\Phi^T \Phi)^{-1} \Phi^T y$$

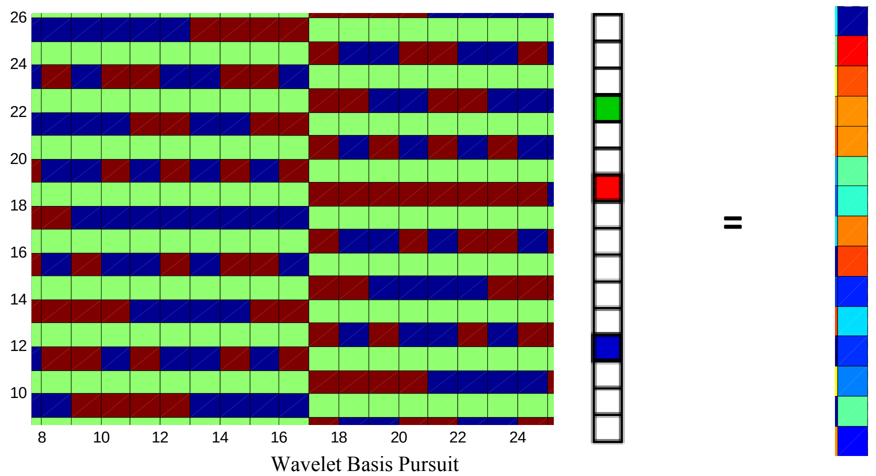
pseudoinverse

How to use CS?

To solve the NP-hardness problem $y = \Phi x = \Phi \Psi \alpha$. We choose different methods to compare.

- **Use the matching pursing method,**

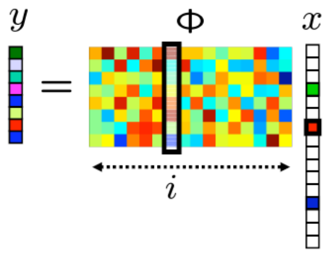




Wavelet Basis Pursuit

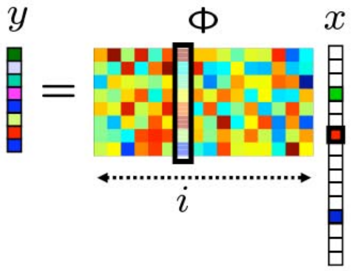
Matching Pursuit

- For each column ϕ_i compute
  $\widehat{x}_i = \langle y, \phi_i \rangle$



- Choose largest $|\widehat{x}_i|$ (greedy)
- Update estimate $\widehat{x}$ by adding in $\widehat{x}_i$
- Form residual measurement and iterate until convergence $y' = y - x_i \phi_i$

● Orthogonal Maching Pursiut



- Same procedure as Matching Pursuit
- Except at each iteration:
  - remove selected column ϕ_i
  - re-orthogonalize the remaining columns of Φ
- Converges in K iterations

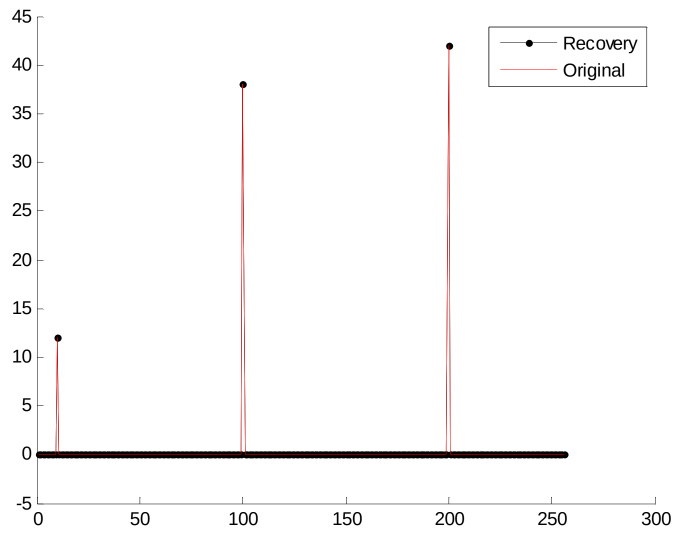


Figure simple DCT Basis pursuit

OMP:

- **Suppose Φ is orthogonal, $\Phi^{-1} = \Phi^{T}$**
- **Solution to Exact problem is unique**
  $c = \Phi^{-1}x = \Phi^{T}x$ i.e., $c_l = <x, \psi_l>$
- **Solution to Sparse problem similar**
  Let l1 be s.t. $|< x, \psi_{l1}>|$ maximized. Set $c_l \longleftarrow < x, \psi_{l1}>$.
  Let l2 be s.t. $|< x, \psi_{l2}>|$ maximized. Set $c_2 \longleftarrow < x, \psi_{l2}>$.

Repeat k times.

Set $c_l \longleftarrow 0$ for l != l1, l2, . . . , lk

Approximate $x \approx \sum_{i=1}^{k} < x, \psi > \psi$

Lab Task 2.2 (need 1 week) - Learning goal: Use compressive sensing theory to acquire ECG/EEG signals. Content: design a random matrix to perform compressive sampling.

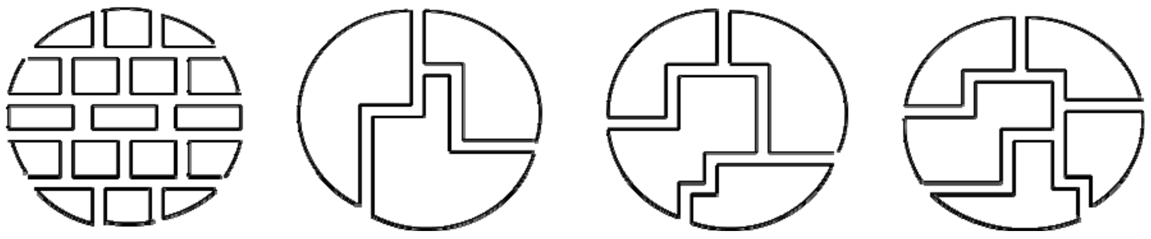


Figure 1. *Sampling structures: (a) uniform sampling with 17 elements (b) non-uniform sampling with 3 elements (c) non-uniform 4 elements and (d) non-uniform sampling with 5 elements.*

$$\Phi=\begin{bmatrix} 1 & 1 & 1 & \cdots & 0 \\ 0 & 1 & 1 & 1 & \\ & & \cdots & & \\ 0 & \cdots & 1 & 1 & 1 \end{bmatrix}_{M\times(KN)} \begin{bmatrix} 1 & -1 & \cdots & 1 & 1 \\ -1 & 1 & \cdots & 1 & -1 \\ & & \cdots & & \\ 1 & 1 & \cdots & -1 & -1 \end{bmatrix}_{(KN)\times(PN)} \begin{bmatrix} 1 & 0 & 1 & \cdots & 0 \\ 0 & 1 & 0 & 1 & \\ & & \cdots & & \\ 0 & \cdots & 1 & 0 & 1 \end{bmatrix} \cdots \begin{bmatrix} w_1 I_{N\times N} & 0 & \cdots & 0 & 0 \\ 0 & w_2 I_{N\times N} & \cdots & 0 & 0 \\ & & \cdots & & \\ 0 & 0 & \cdots & 0 & w_P I_{N\times N} \end{bmatrix}_{(PN)\times N}$$

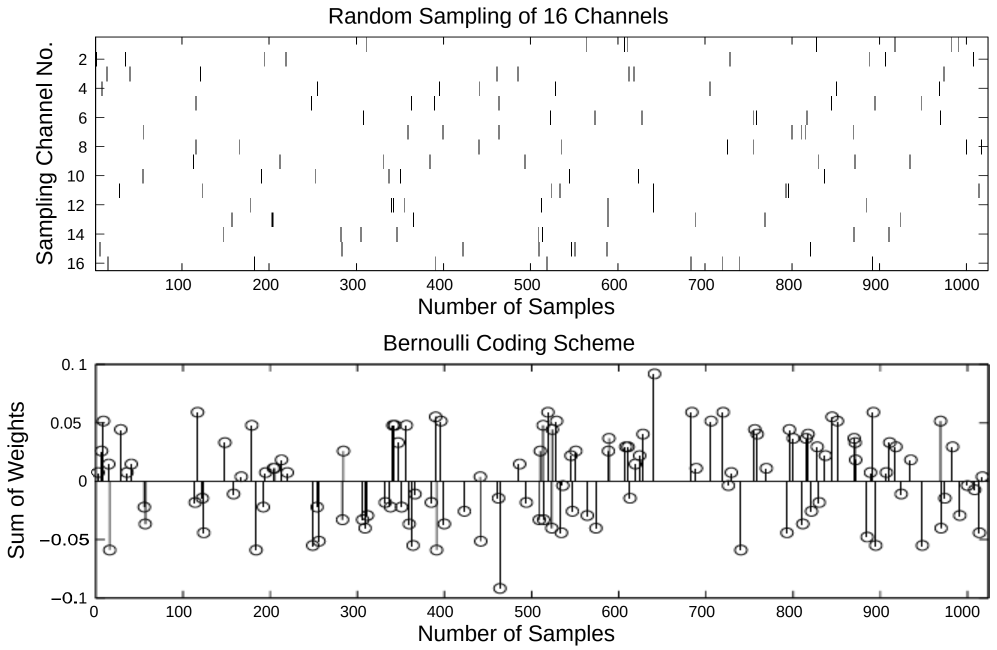


Figure 3. *Random compressive measurement of 1024 samples. (Upper) Random sampling scheme with 16 sensing elements, where the number of non-zero values is 128. (Lower) The sum of sampling weights under Bernoulli coding scheme.*

Lab Task 2.3 (need 1 week) - Learning goal: Use L1 optimization to reconstruct ECG/EEG signals.

We use PROC method to solve the L1 optimization problem :

$$\min \|\Psi\beta\|_{TV} + \lambda \cdot \|\beta\|_{\ell_1} \quad \text{such that} \quad \mathcal{M}'\beta = y$$

total variation (TV):

$$\|g\|_{TV} = \sum_{s,t} \sqrt{(g(s+1,t) - g(s,t))^2 + (g(s,t+1) - g(s,t))^2}$$

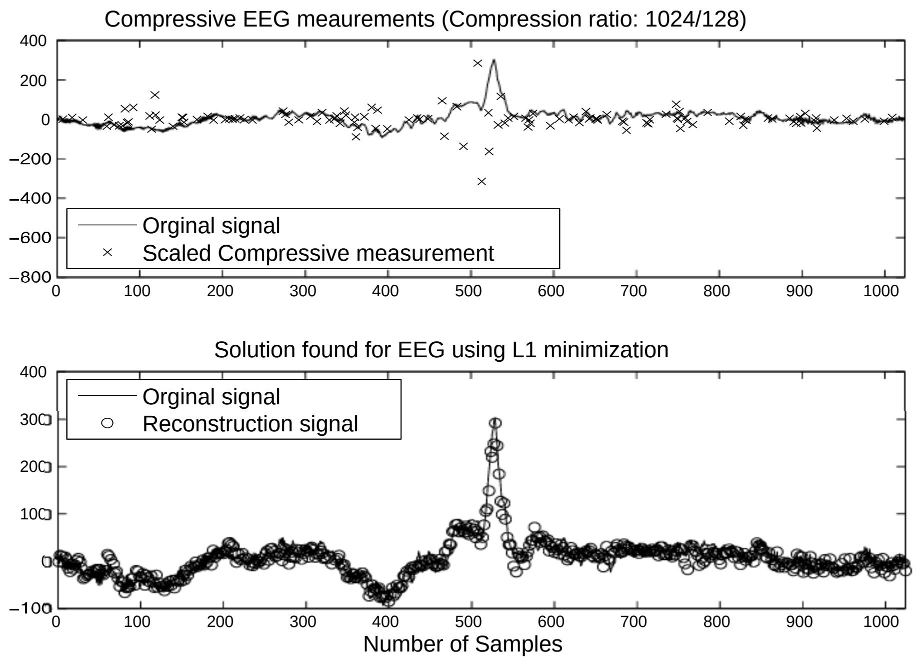


Figure 4. *Results of the compressive EEG sensing. (Upper) Compressive measurements of one channel EEG signals with a compression ratio of 1024:128. (Lower) Reconstructed EEG signals using ℓ1 minimization.*

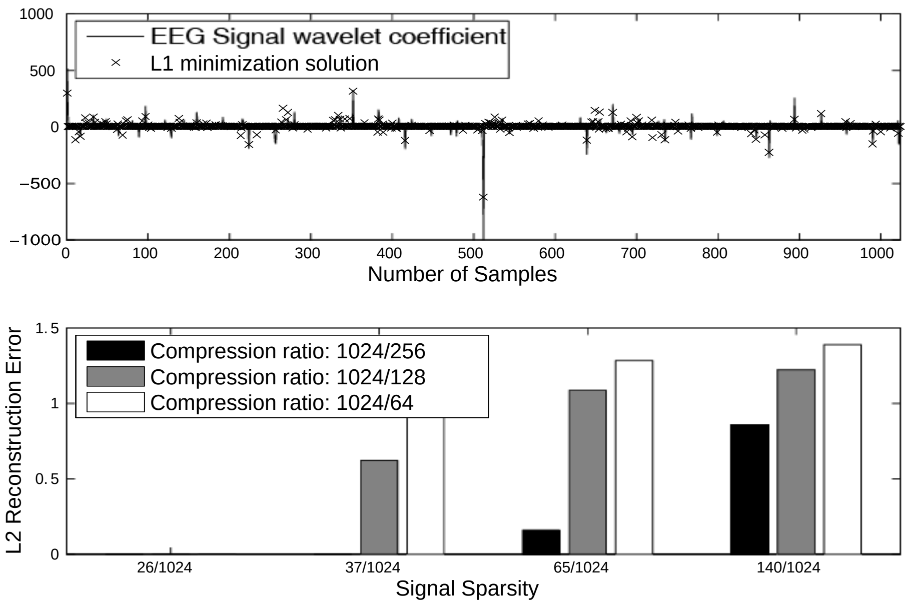


Figure 5. *(Upper) The sparse wavelet (db6) coefficients of EEG signals and the ℓ1 minimization solution. (Lower) Reconstruction errors with respect to different values of signal sparsity.*

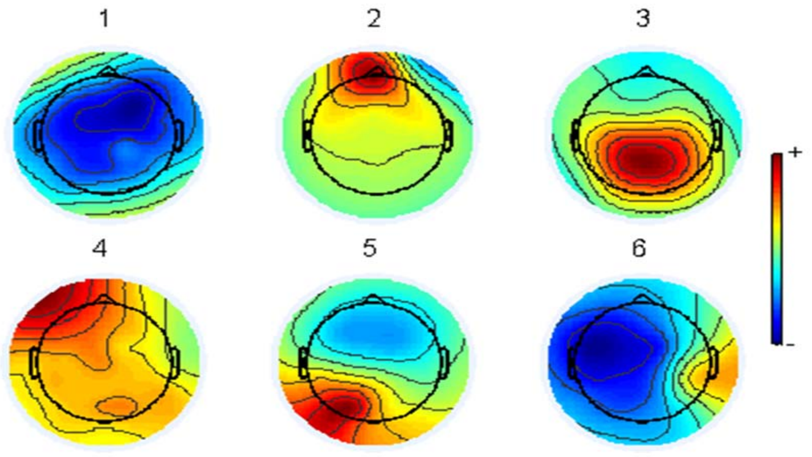


Figure 6. *2-D scalp map projection of the first 6 independent components of the compressively sampled data.*

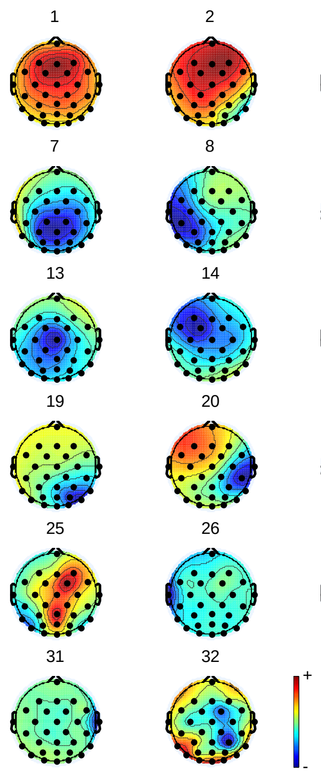


Figure 2D plot of 32 ICA components from EEG reconstruction

Matlab Code Samples:

1. Matching Pursuit

2. Orthogonal Matching Pursuit code sample

3. Measurement matrix building

4. L1 optimal reconstruction algorithm

1. Matching Pursuit

```
function [S,R,e,indx] = matchPurs(x,W)
```

```
% This function computes the projection of a given input vector or matrix
% onto a "dictionary" of other vectors or matrices using a matching pursuit
% algorithm.
%
% USAGES
% [S,R,e,indx] = matchPurs(x,W)
%
% INPUT
% x:    An Mx1 or MxN array. This array synthesized using dictionary
%       elements from matrix W.
% W:    An MxN or MxNxP array of dictionary elements used to synthesize
%       input x. If x is an Mx1 vector, W must be an MxN matrix. If x is
%       an MxN array, W must be an MxNxP matrix containing the dictionary
%       elements.
%
% OUTPUT
% S:    The projection of each residual onto each dictionary element.
% R:    The residual x - sum(W,dim), where dim is 2 if x is a vector and
%       is 3 if x is a matrix.
% e:    The projection coefficients.
% indx  The index vector for the dictionary elements used in projections
%
% ------------------------------------------------------------------------
% TEST
% To ensure that the signal energy is preserved, the following relations
% should hold:
%
% Matrix Case:
%
% (1) x = sum(S,3) + R;
% (2) norm(x,'fro') = sum(e.^2) + norm(R,'fro').^2
%
% Vector Case:
%
% (1) x = sum(S,2) + R;
% (2) norm(x).^2 = sum(e.^2) + norm(R).^2;


%% For Hilbert Spaces Whose Elements are Vectors
% This space has Euclidean inner product.  A vector is synthesized as:
%
% x = ( x'*W(:,:,1) )*W(:,:,1) + (R1'*W(:,:,2) )*W(:,:,2) + ...
%     ( Rk'*W(:,:,k) )*W(:,:,k);
%
% Where Rk is the k_th residual.
% ------------------------------------------------------------------------

if( ~isvector(x) )

    [M,N,P] = size(W);
    S       = zeros(size(W));
    %intialize vectors
    R       = x; %the residual
    e       = zeros(P,1); %the energy vector
    ipvec   = zeros(P,1); %the dictionary indices
    indx    = ipvec; %the index for the best match

    for n = 1:P

        for k = 1:P

            ipvec(k)    = trace(R'*W(:,:,k))./norm( W(:,:,k), 'fro');

        end;

        [m, i]      = max(abs(ipvec));
        px          = ipvec(i)*W(:,:,i)./norm( W(:,:,i), 'fro');
        S(:,:,n)    = px; %keep each S
        R           = R - px;
        indx(n)     = i;
        W(:,:,i)    = nan(M,N);
        e(n)        = ipvec(i);

    end;

end;

% -----------------------------------------------------------------------
%% For Hilbert Spaces Whose Elements are Vectors
% This space has Euclidean inner product.  A vector is synthesized as:
%
% x = ( x'*W(:,1) )*W(:,1) + (R1'*W(:,2) )*W(:,2) + ...
%     ( Rk'*W(:,k) )*W(:,k);
%
% Where Rk is the k_th residual.
% -----------------------------------------------------------------------

if( isvector(x) )

    [M,P]   = size(W);
    W       = normColumns(W);
    %intialize vectors
    R       = x; %the residual
    e       = zeros(P,1); %the energy vector
    ipvec   = zeros(P,1); %the dictionary indices
    indx    = ipvec; %the index for the best match

    for n = 1:P

        for k = 1:P

            ipvec(k)    = R'*W(:,k);

        end;

        [m, i]      = max(abs(ipvec));
        px          = ipvec(i)*W(:,i);
        S(:,n)      = px;
        R           = R - px;
        indx(n)     = i;
        W(:,i)      = nan(M,1);
        e(n)        = ipvec(i);

    end;

end;
```

2. Orthogonal Matching Pursuit code sample

```
%  1-Dsignal compressive sensing implementation (Orthogonal Matching Pursuit)
%  measure number M>=K*log(N/K),K is the sparsity,N is the length of the
%  signal,reconstruction well


clc;clear

%%  1. time domain signal generation
K=8;      %  sparsity
N=256;    %  signal length
M=64;     %  Measurement number(M>=K*log(N/K),more than 40,have possibility for error)
f1=50;    %  signal frequence 1
f2=100;   %  signal frequence 2
f3=200;   %  signal frequence 3
```

```
f4=400;   %  signal frequence 4
fs=800;   %  sampling frequence
ts=1/fs;  %  sampling interval
Ts=1:N;   %  sampling sequence
x=0.3*sin(2*pi*f1*Ts*ts)+0.6*sin(2*pi*f2*Ts*ts)+0.1*sin(2*pi*f3*Ts*ts)+0.9*sin(2*pi*f4*Ts*ts);  %
whole signal

%%  2.  time domain signal compressive sensing
Phi=randn(M,N);                               %  measurement matrix(Gaussain while noise)
s=Phi*x.';                                %  measurement result y

%%  3.  orthogonal matching pursuit reconstruction(same as L_1 norm optimazition problem)
m=2*K;                                    %  iterative time(m>=K)
Psi=fft(eye(N,N))/sqrt(N);                        %  Fourier basis matrix
T=Phi*Psi';                               %  reconstruction matrix(measurement matrix*orthogonal transposed
matrix)

hat_y=zeros(1,N);                             %  reconstruction domain(transfer domain)vector
Aug_t=[];                                 %  augument matrix(initial is empty matrix)
r_n=s;                                    %  resual value

for times=1:m;                               %  iterative time
    for col=1:N;                             %  coloum number
        product(col)=abs(T(:,col)'*r_n);          %  inner product
    end
    [val,pos]=max(product);                      %  position
    Aug_t=[Aug_t,T(:,pos)];                      %  collected basis
    T(:,pos)=zeros(M,1);                       %  delete picked bais(set to zero)
    aug_y=(Aug_t'*Aug_t)^(-1)*Aug_t'*s;           %  LSE
    r_n=s-Aug_t*aug_y;                         %  reduil
    pos_array(times)=pos;                      %  record position
end
hat_y(pos_array)=aug_y;                        %  reconstructed new domain vector
hat_x=real(Psi'*hat_y.');                     %  time domain reconstructed signal

%%  4.  comparison between original and reconstrcted signal
figure;

hold on;
plot(hat_x,'k.-')                           %  reconstucted signal
plot(x,'r')                               %  original signal
legend('Recovery','Original')
norm(hat_x.'-x)/norm(x)                       %  error
```

3. Measurement matrix building

```
function [C, index_C, CC] = get_MeasurementMatrix(m, K, n, i, NON_UNIFORM, BERNOULLI)

CC = zeros(n, m);
for ii=1:i          %increase the measurement sparsity
    CC = CC + SparseMeasurementMatrix(n, m, K);
end
CC(find(CC~=0))=1;

if NON_UNIFORM
    non_uniform = repmat(abs(randn(1, m)),[n, 1]);
    CC = CC.*non_uniform;
end

if BERNOULLI
   index=find(CC~=0);
   aa=randn(length(index),1);
   aa(aa>0)= 1;
   aa(aa<0)=-1;
   CC(find(CC~=0))=CC(index).*aa;
end

C  = zeros(n-m+1, n);
for ii = 1: n-m+1
    C(ii,:) = [zeros(1, ii-1) CC(ii,:) zeros(1, n-m+1-ii)];
end
index_C     = find(any(C,2));
C           = C(index_C,:);
```

4. L1 optimal reconstruction algorithm

```
function xest = cspocs_l1(Phi, aPhi, l1val, y, niter)
% Syntax :
%  xest = cspocs_l1(Phi, aPhi, l1val, y, niter)
%
% Description : Testing Candes and Romberg POCS (alternate Projection Onto
% Convex Sets) for CS recovery, aka recovering of x from y = Phi*x when
% size(Phi,1) < size(Phi,2), using l1 criterion.
%
% In :
% * Phi, aPhi : Measurement matrix and its reconstruction
% * l1val : the l1 norm of the intial signal x, i.e. norm(x,1)
% * niter : maximun number of alternate projections.
%
% Out :
% * xest : the recovered (or estimated) signal x
%
% Author of this mfile : L. Jacques, LTS2/EPFL, 2008.
%
% Reference :
%    Algo described in "Practical Signal Recovery from Random Projections",
%    Emmanuel Candès and Justin Romberg
% Example :
% >> N=128; K=20;
% >> x=[rand(1,K) zeros(1,N-K)]'; x=x(randperm(128));
% >> m=floor(3.5*K);Phi=randn(m,N)/sqrt(m);aPhi=Phi';
% >> y=Phi*x;
% >> figure; plot(x);
% >> nx=cspocs_l1(Phi,aPhi,norm(x,1),y,10000);
% Stop at n=2157, score=9.987234e-11 ...
% Final score = 9.987234e-11 ...
% >> hold on; plot(nx,'ro');
%
% Remark: This algo seems highly sensitive to the a priori l1 norm of x. To
% convince yourself of that, repeat the same experiment as above with
% norm(x,1)*0.99 and norm(x,1)*1.01 and observe the
% results. Recovery/NoRecovery transition points seems located around 3.2K
%
% This script/program is released under the Commons Creative Licence
% with Attribution Non-commercial Share Alike (by-nc-sa)
% http://creativecommons.org/licenses/by-nc-sa/3.0/
% Short Disclaimer: this script is for educational purpose only.
% Longer Disclaimer see  http://igorcarron.googlepages.com/disclaimer

  [m,N] = size(Phi);

  pPhi = pinv(Phi);

  %% First projector (on the hyperplane Phi*x=y)
  P = @(u) u + pPhi*(y - Phi*u);
```

```
    %% Second projector on the l1 ball of radius l1val
    H = @(u, l1val) sft_th(u, l1toth(u,l1val));

    %% Intializations
    xest = pPhi*y;
    Tol1 = 1e-7;
    score = 0;

    %for i=1:10;

    for n = 1:niter;
        xest = P(xest);
        xest = H(xest, l1val);

        oldscore = score;
        score = norm(Phi*xest - y) / norm(xest);

        if (score < Tol1)
            fprintf('Tolerance reached. Stop at n=%i\n',n);
            break
        end
    end

    %x=W'*xest;

    %xsmooth=x-norm(diff(W')).*xest;

    %xest=W*xsmooth;



    %end

    fprintf('Final score: ||y - Phi*xest|| = %e\n', score);


function out = l1toth(x,l1val)
% (Internal function)
% Obtain the threshold level corresponding to the one such that the l1
% norm of the corresponding soft thresholded signal is lesser than l1val
    N = length(x);
    k = (1:N)';
    ax = abs(x);

    xs = sort(ax, 'descend');
    cxs = cumsum(xs);
    pos = (cxs - k.*xs) < l1val;
    out = xs(pos);
    out = out(end);

function out = sft_th(x, gamma)
% (Internal function)
% Soft Thresholding
    out = sign(x) .* (abs(x) - gamma) .* (abs(x) >= gamma);
```

Reference

[1] Stephane Mallat and Sifen Zhong, “Characterization of Signals from Multiscale Edges”, IEEE TRANSACTIONS ON PATIERN ANALYSIS AND MACHINE INTELLIGENCE

[2] Stephane Mallat, A Wavelet Tour of Signal Processing

[3] Ingrid Daubechies, “10 couses on wavelet”

[4] Mallat, S. G.，A Theory for Multiresolution Signal Decomposition: The Wavelet Representation, IEEE Trans. PAMI, vol. 11, no. 7, July 1989, pp. 674-693.

[5] http://en.wikipedia.org/wiki/Wavelet

[6] http://en.wikipedia.org/wiki/Fourier_transform

[7] Qi Hao, Fei Hu, “ A Compressive Eletroencephalography (EEG) Sensor Design”, 2010 IEEE Sensors

[8] Tutorial on Compressive Sensing, Richard Baraniuk, Rice University, Justin Romberg, Georgia Tech, Michael Wakin University of Michigan.

[9] Anna C. Gilbert, Sparse approximations in image processing, Department of Mathemtics University of Michigan

[10] Emmanuel Cand` es and Justin Romberg, “Practical Signal Recovery from Random Projections”, Department of Applied and Computational Mathematics, Caltech