Sparse Approximation and Applications

Chapter Intended Learning Outcomes:

(i) Realize that many real-world signals can be sparse when represented in another domain

(ii) Understand the importance of sparse modeling

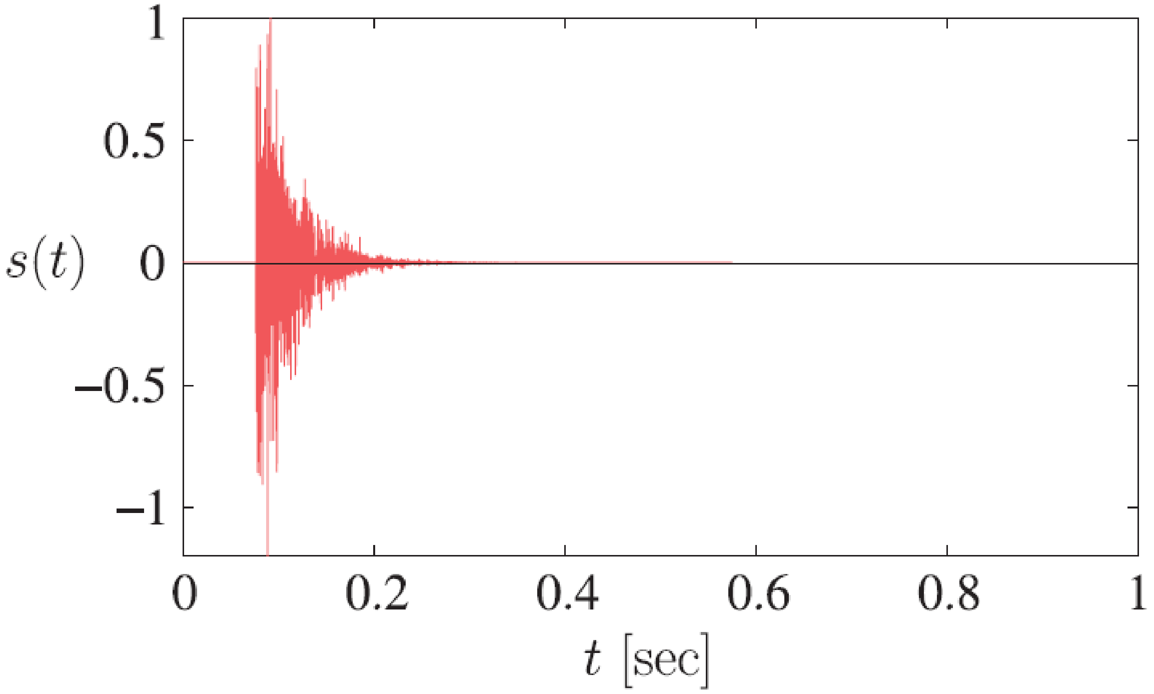
(iii) Able to solve the sparse approximation problem

(iv) Able to apply sparse approximation in real-world applications

Real-World Sparse Signal Examples

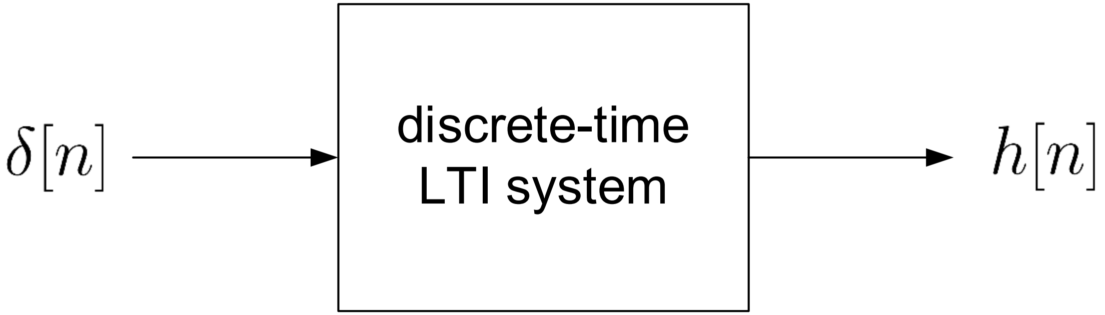
A signal is sparse if most of its coefficients are (approximately) zero.

The impulse response of an echo-path in a telephone network, which is a time-domain function, is sparse, because it only appears in an unknown short duration.



Recall the impulse response is the output $h[n]$ of a discrete-time linear time-invariant (LTI) system with input being an impulse function $\delta[n]$:

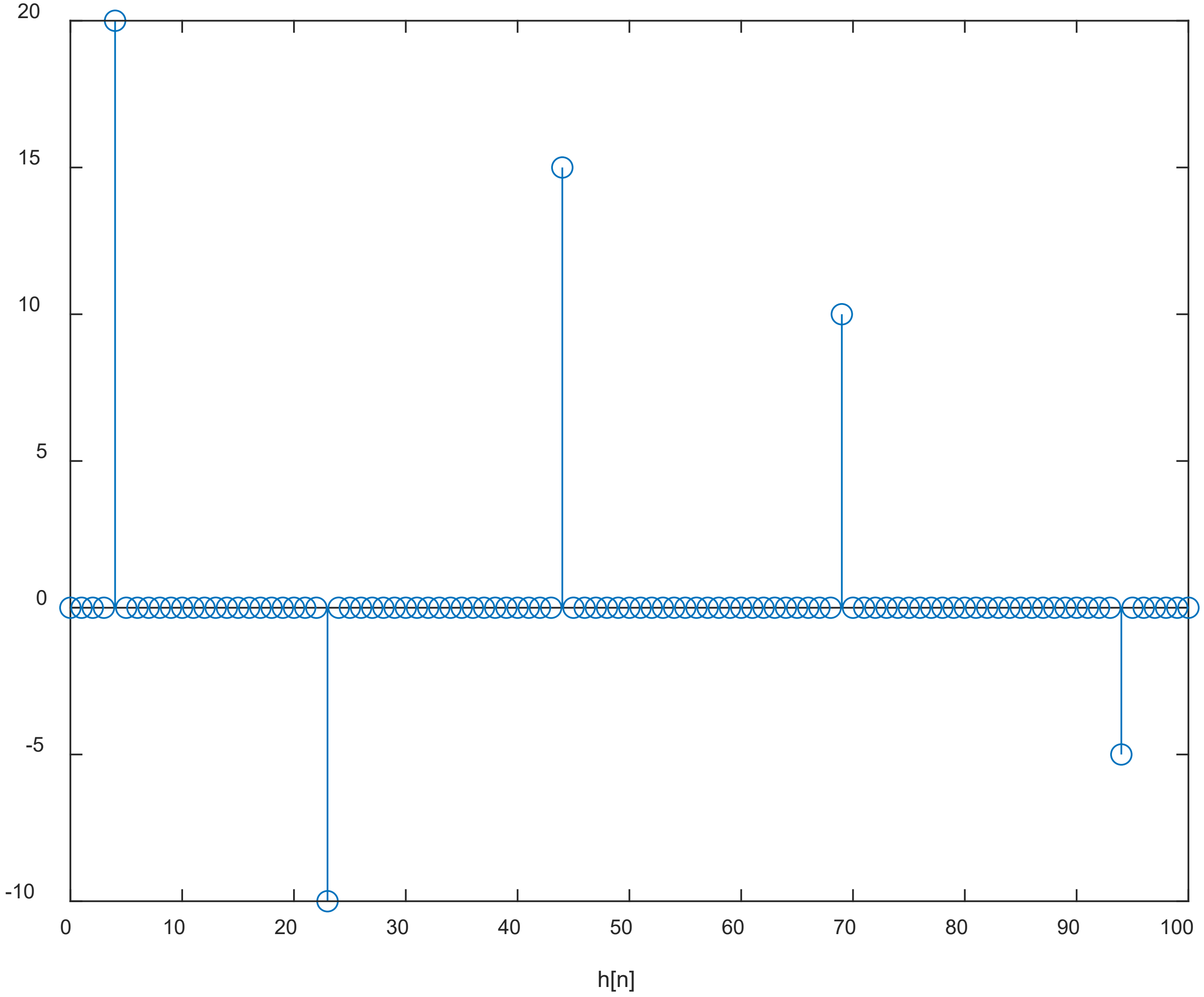
$$\delta[n] = \begin{cases} 1, & n = 0 \\ 0, & n \neq 0 \end{cases}$$



Suppose $h[n]$ is of length N, i.e., nonzero for $h[0], \cdots, h[N-1]$, and output $y[n]$ with input $x[n]$ is represented via convolution:

$$y[n] = h[n] \otimes x[n] = \sum_{m=0}^{N-1} h[m]x[n-m]$$

Another similar example regarding sparsity is multipath channel in wireless communications:



Apparently, this impulse response $h[n]$ is also sparse because it is nonzero only at the time indices 5, 24, 45, 70, and 95.

If the source signal is $x[n]$, then the resultant signal due to the channel is:

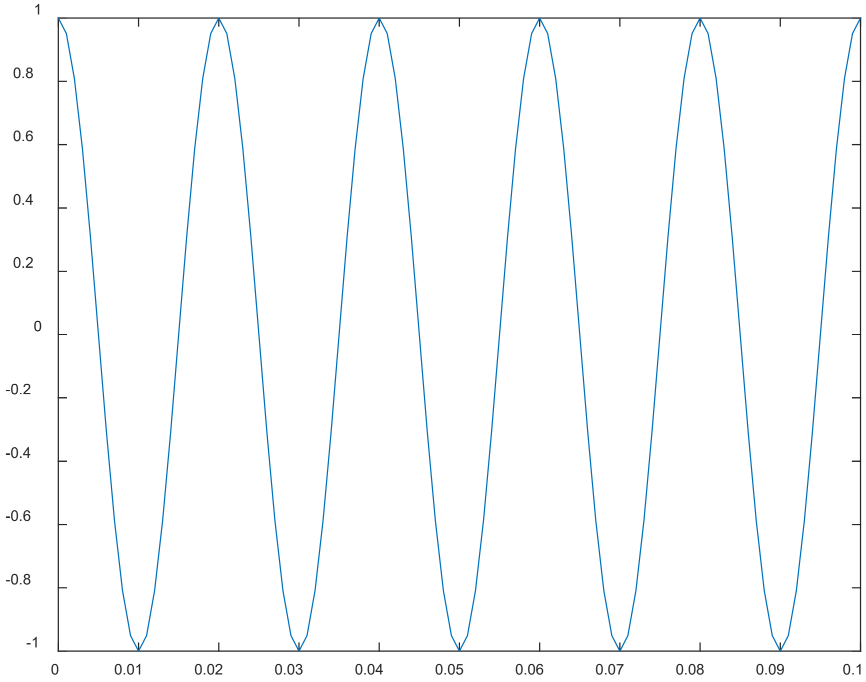
$$y[n] = h_5 x[n-5] + h_{24} x[n-24] + h_{45} x[n-45] + h_{70} x[n-70] + h_{95} x[n-95]$$

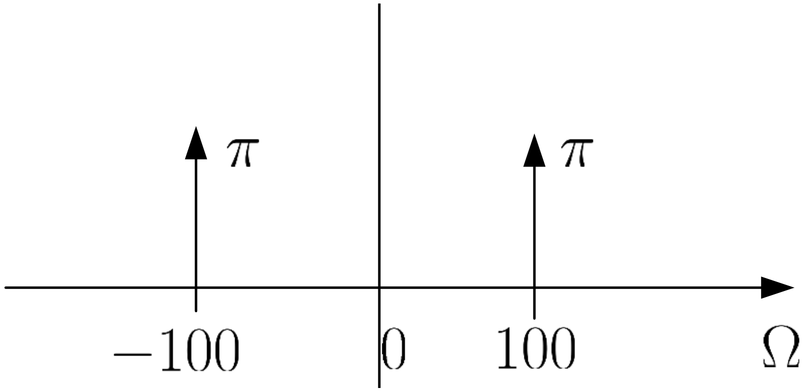
where we clearly see that there are 5 multipaths. If we have the input $x[n]$ and output $y[n]$, then the impulse response $h[n]$ can be computed.

Nevertheless, sparsity may not be directly observed from the original data but via a transformation.

Consider a simple sinusoidal signal:

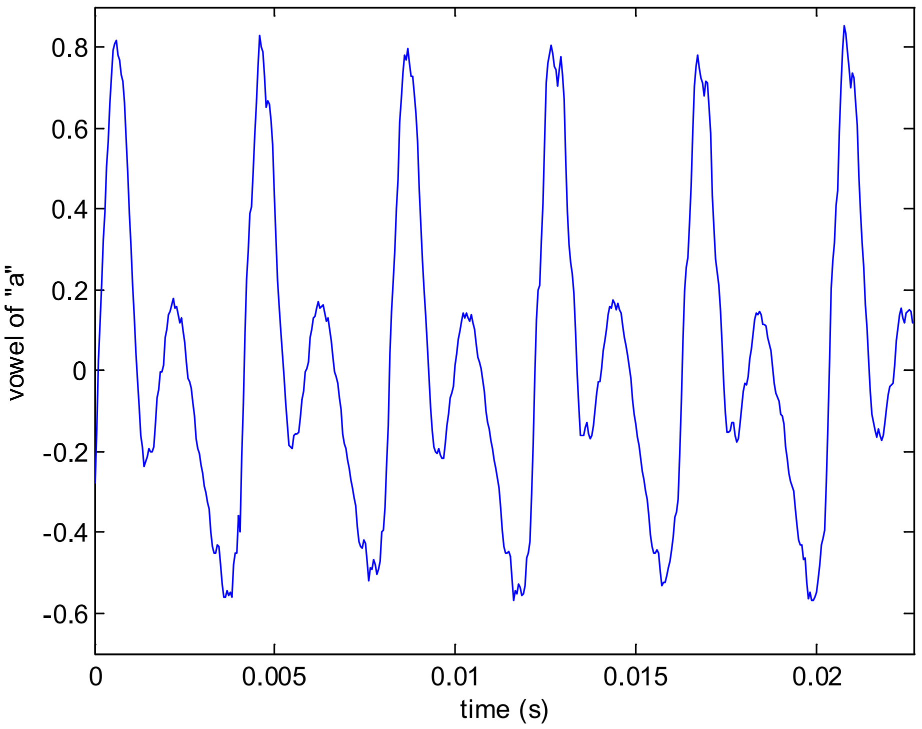
$$x(t) = \cos(100t) = \frac{e^{j100t} + e^{-j100t}}{2} \leftrightarrow X(j\Omega) = \pi\delta(\Omega + 100) + \pi\delta(\Omega - 100)$$

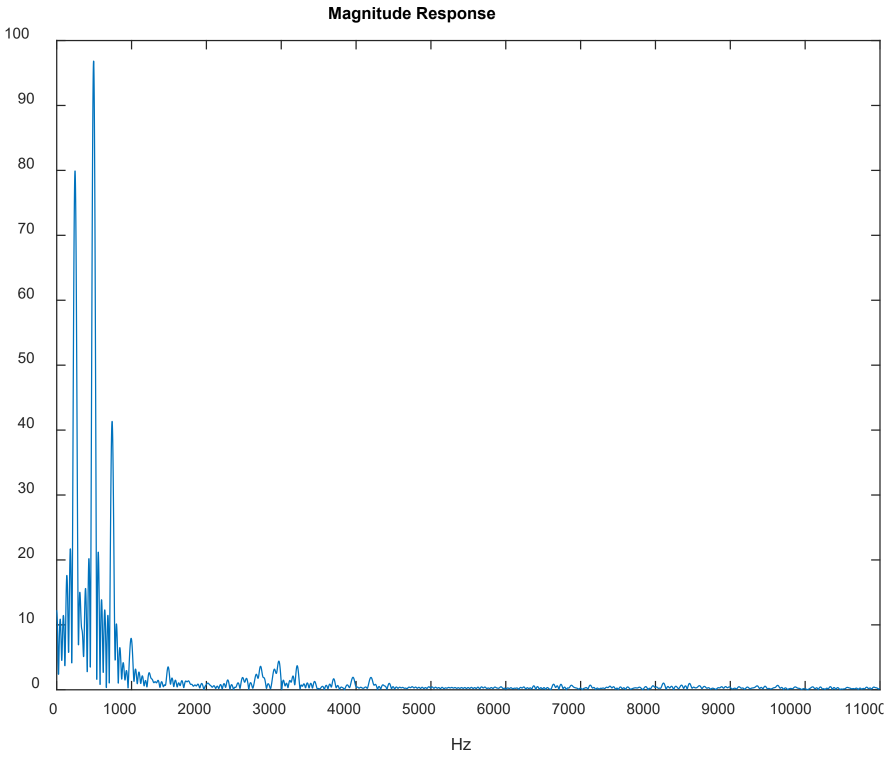




Clearly, it is not sparse in the time domain, but it is sparse if we view the signal in the frequency domain.

A speech segment in the time domain also becomes sparse when viewing in the frequency domain:





If we convert an image matrix to a vector, and then perform discrete cosine transform (DCT), we can find that the DCT coefficients are sparse, i.e., many coefficients can be approximated as 0. The DCT transform of $\boldsymbol{x} = [x_0, \cdots x_{N-1}]^T$ is:

$$X_k = c_k \sum_{n=0}^{N-1} x_n \cos\left(\frac{\pi}{2N}(2n+k)\right), \quad k = 0, \cdots, N-1$$

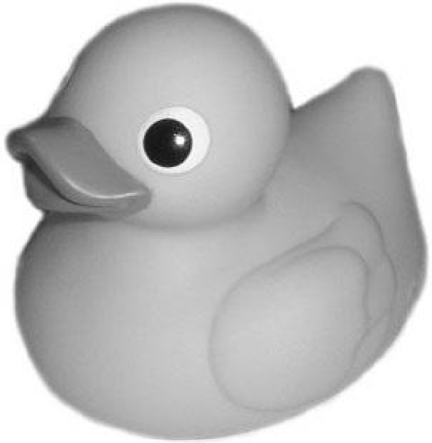
where

$$c_k = \begin{cases} \sqrt{1/N}, \ k = 0 \\ \sqrt{2/N}, \ k = 1, \cdots, N-1 \end{cases}$$

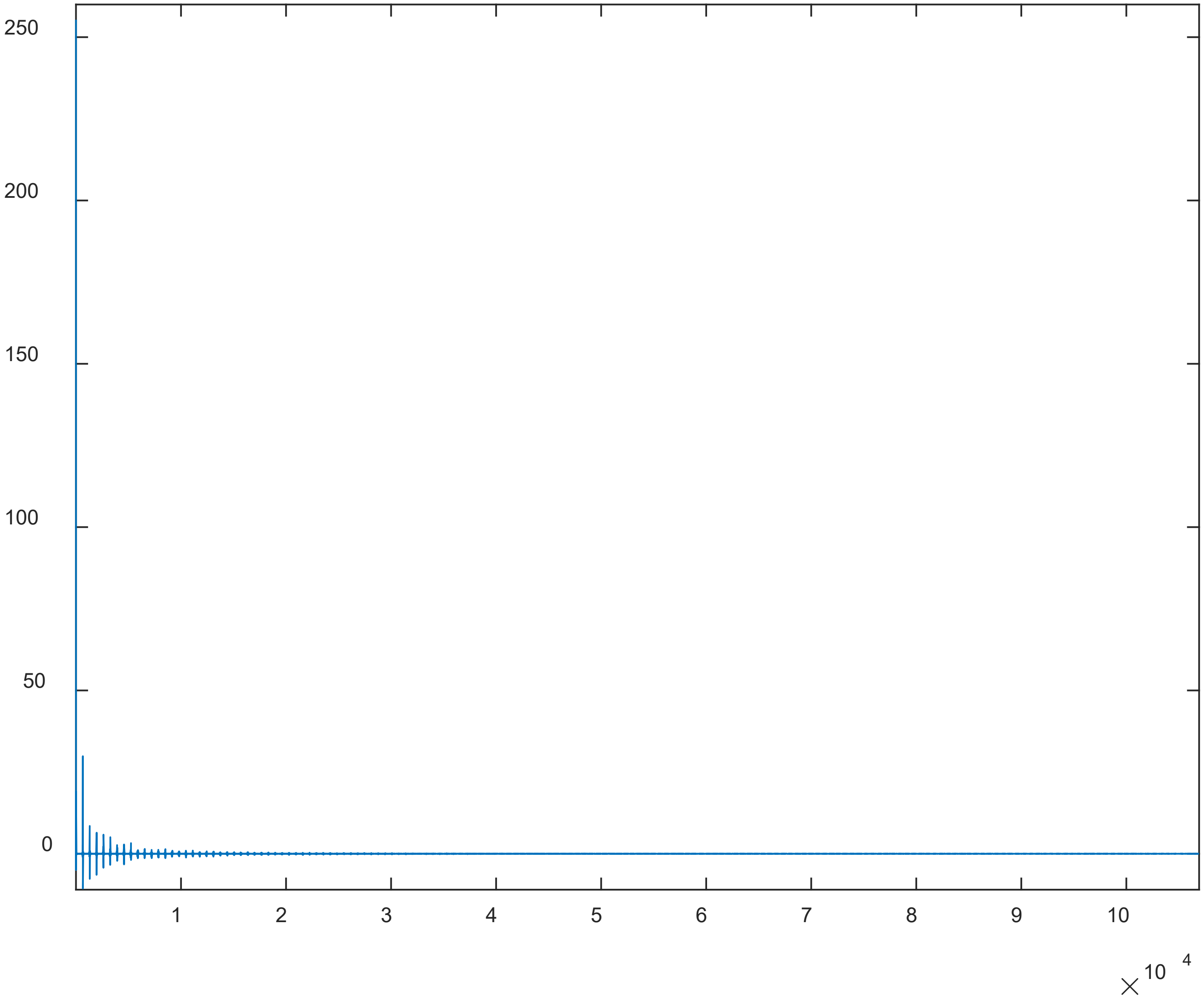
It looks like the discrete Fourier transform (DFT) with only keeping the cosine components and ignoring the sine components, and is a real-valued transform.

Note that direct transforming the matrix with 2D-DCT would be more appropriate but it is easier to use the 1D transform as illustration.

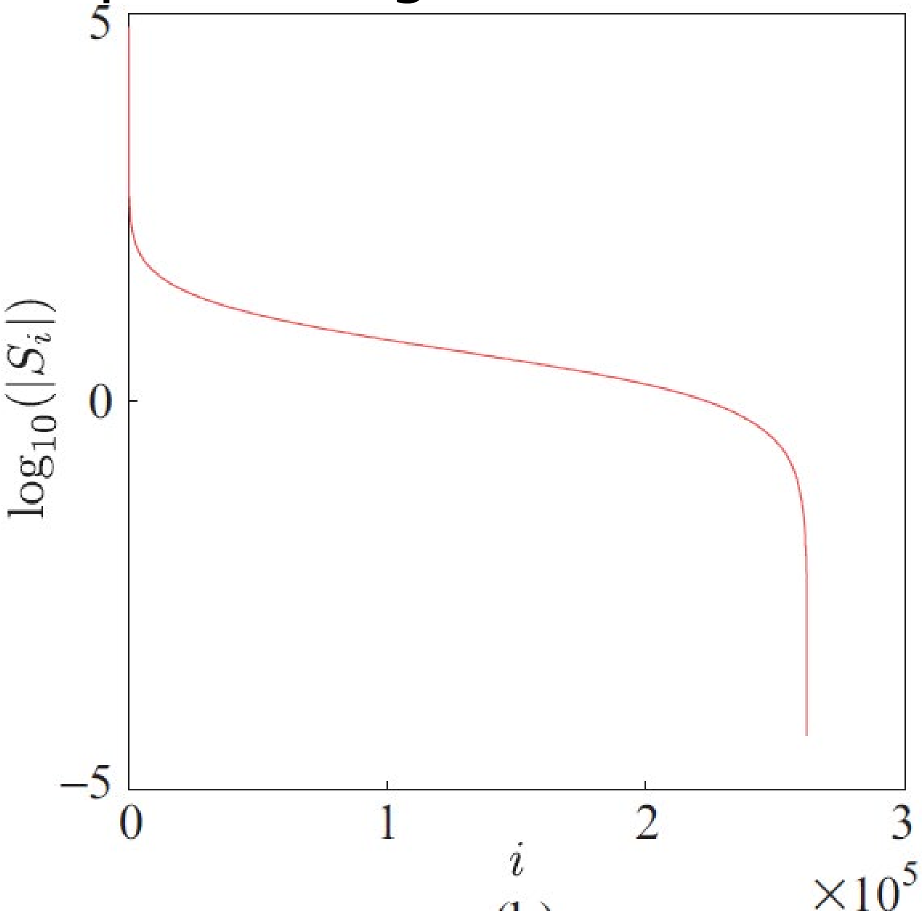
Consider vectorizing the gray-scale ducky image $\boldsymbol{X} \in \mathbb{R}^{327 \times 327}$ into a vector $\boldsymbol{x} \in \mathbb{R}^{106929}$



```
>> img = imread('Gray_Ducky.jpg');
gray = im2double(img);
duck_vec=gray(:);
duck_dct=dct(duck_vec);
plot(duck_dct)
```


250
200
150
100
50
0
1
2
3
4
5
6
7
8
9
10
×10 4

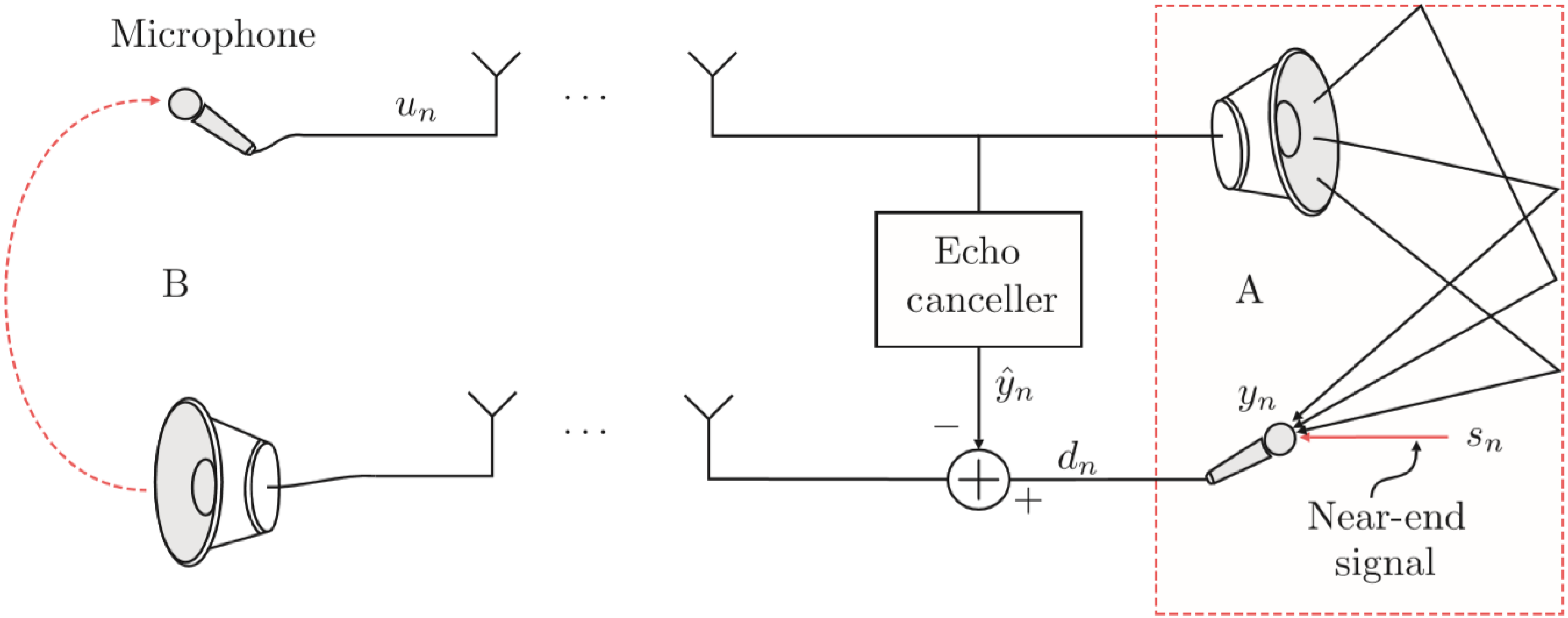
Another example of a 512×512 pixel image:



Arranging the magnitudes of the DCT coefficients in descending order and logarithmic scale, it is found that more than 95% of the total energy is contributed by only 5% of the largest components.

Advantages of Sparse Signal Representation

We can utilize the sparse information to achieve efficient channel or impulse response estimation, e.g., in echo cancellation in video conferencing where we want to remove far-end signal due to echo interfering with near-end signal.



The source signal at the far-end, denoted by u_n, is known.

Suppose we know that the maximum length of the echo impulse response h_n is N . We can use a vector $\boldsymbol{w} = [w_0 \ \cdots \ w_{N-1}]^T$ to model the unknown echo channel.

The echo canceller output and far-end response are then:

$$\hat{y}_n = \sum_{m=0}^{N-1} w_m u_{n-m} \qquad \text{and} \qquad y_n = \sum_{m=0}^{N-1} h_m u_{n-m}$$

Let v_n be the signal to be sent out. It is:

$$v_n = d_n - \hat{y}_n = s_n + (y_n - \hat{y}_n)$$

Clearly, if $w_n = h_n$, the echo can be cancelled with $v_n = s_n$.

Suppose we have $M \geq N$ measurements for both u_n and y_n. The weights can be obtained by minimizing the least squares (LS) cost function:

$$J(\boldsymbol{w}) = \sum_{m=1}^{M} v_m^2 = \sum_{m=1}^{M} \left(d_n - \sum_{m=0}^{N-1} w_m u_{n-m} \right)^2$$

Viewing $\hat{y}_n$ as a linear regression model:

$$\hat{y}_m = \boldsymbol{w}^T \boldsymbol{u}_m, \quad m = 1, \cdots, M, \quad \boldsymbol{u}_m = [u_m, \cdots, u_{m-N+1}]^T$$

We can construct a matrix $\boldsymbol{U} \in \mathbb{R}^{M \times N}$ with rows $\boldsymbol{u}_m^T$ yielding

$$J(\boldsymbol{w}) = (\boldsymbol{d} - \boldsymbol{U}\boldsymbol{w})^T (\boldsymbol{d} - \boldsymbol{U}\boldsymbol{w}) = ||\boldsymbol{d} - \boldsymbol{U}\boldsymbol{w}||_2^2 \Rightarrow \hat{\boldsymbol{w}} = (\boldsymbol{U}^T \boldsymbol{U})^{-1} \boldsymbol{U}^T \boldsymbol{d}$$

On the other hand, gradient descent can be used as well.

However, the complexity of LS solution will be high if N is very large.

While the convergence rate of the gradient descent can be very slow.

To increase the efficiency, a better formulation is to utilize the *a priori* sparsity information:

$$\text{minimize}: \quad J(\boldsymbol{w}) = (\boldsymbol{d} - \boldsymbol{U}\boldsymbol{w})^T(\boldsymbol{d} - \boldsymbol{U}\boldsymbol{w}), \quad \text{subject} \quad \text{to}: \quad ||\boldsymbol{w}||_0 \leq K$$

where we assume that the maximum number of nonzero components of h_n, namely, $K \ll N$, is known.

Basically, the task here is to find a sparse vector $\boldsymbol{w}$ such that $\boldsymbol{d} \approx \boldsymbol{U}\boldsymbol{w}$.

Similarly, the sparse linear regression problem for multipath channel estimation in wireless communications means finding a sparse vector $\boldsymbol{w}$ such that $\boldsymbol{y} \approx \boldsymbol{X}\boldsymbol{w}$ where $\boldsymbol{y}$ and $\boldsymbol{X}$ re constructed from the output $y[n]$ and input $x[n]$, respectively,

We expect there are only 5 nonzero coefficients for the above example.

In the above examples, we may be able to collect $M \geq N$ measurements where N is the length of vector to be estimated.

In fact, even when $M < N$, i.e., the number of equations is less than the number of unknowns, it is possible to solve it because of the known sparse information. This corresponds to applicability in many large-scale scenarios particularly real-time processing is required.

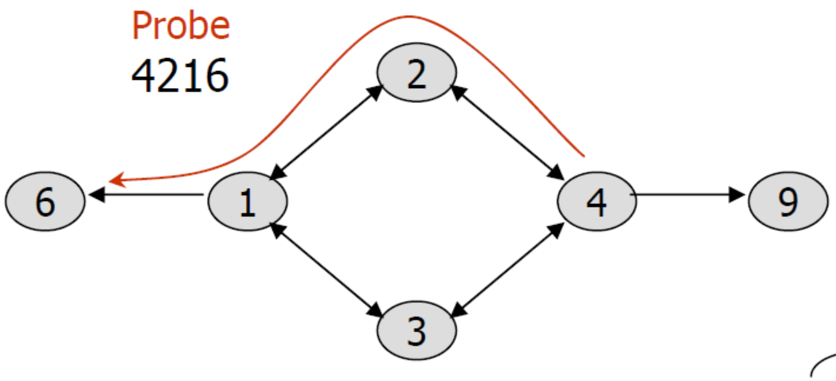
Consider the problem of identifying network performance bottlenecks, e.g., network links responsible for unusually high end-to-end delays.

In a large-scale system, monitoring every network link is costly or even infeasible, and real-time diagnosis will not be allowed.

Alternatively, we can collect a relatively small number of overall performance measures using end-to-end test measurements or probes, and then deduce the states of individual components.

In practice, there are only a few malfunctioning links responsible for transaction delays, while the remaining links function properly, i.e., the problematic links are considered sparse among all links.

(Beygelzimer, Kephart and Rish 2007)



N links (possible bottlenecks)

Y		X_1 ❶❷	X_2 ❶❸	X_3 ❶❻	X_4 ❷❹	X_5 ❸❹	X_6 ❹❾
Y_1	❹❷❶❻	1	0	1	1	0	0
Y_2	❹❸❶❻	0	1	1	0	1	0
Y_3	❸❶❷	1	1	0	0	0	0
Y_4	❶❷❹❾	1	0	0	1	0	1

M probes

vector Y of end-to-end probe delays

vector X of (unknown) link delays

Routing matrix A

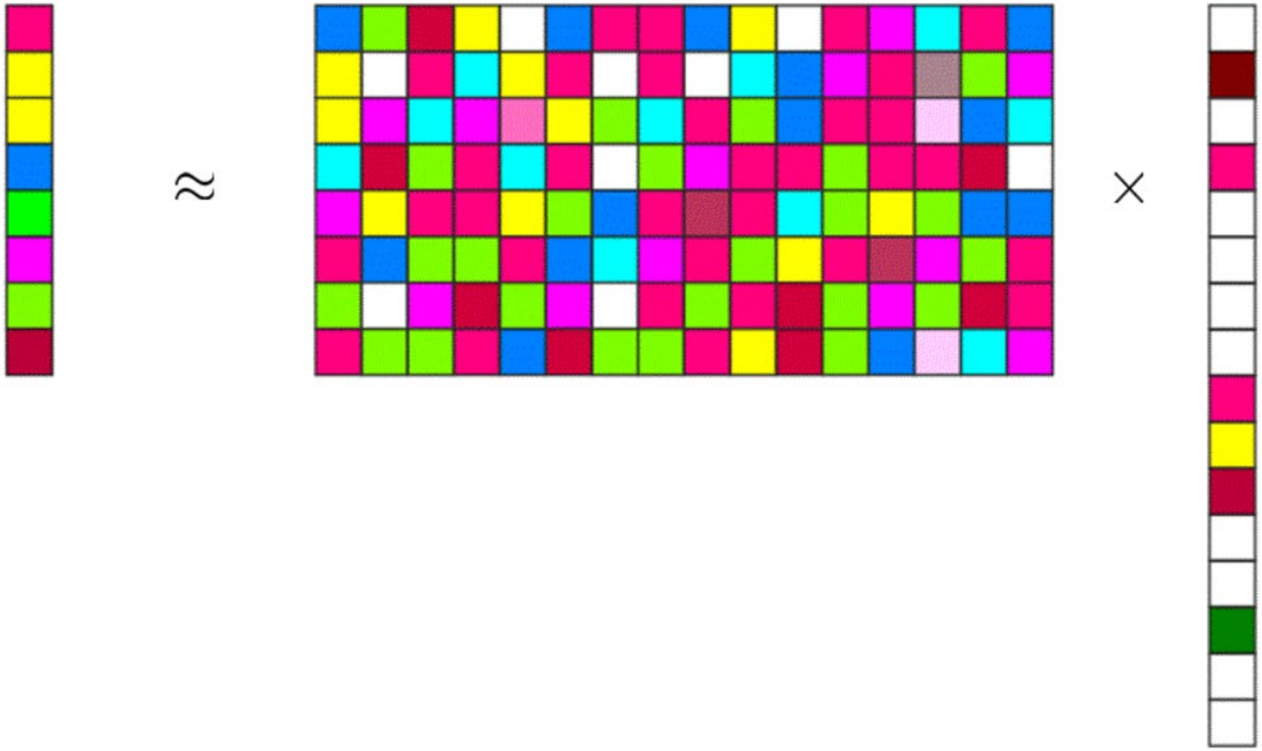
Routing matrix $\boldsymbol{A} \in \mathbb{R}^{M \times N}$ where $a_{i,j} = 1$ if the end-to-end test i goes through the link j, and 0 otherwise.

Unobserved vector of link delays $\boldsymbol{x} \in \mathbb{R}^N$ is well approximated by a sparse vector, where only a few entries have relatively large magnitudes, as compared to the rest.

Observed vector of end-to-end transaction delays $\boldsymbol{y} \in \mathbb{R}^M$

We can construct:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{q}$$



where $\boldsymbol{x} \in \mathbb{R}^N$ is assumed sparse and $\boldsymbol{q} \in \mathbb{R}^M$ is observation noise including the nonzero effect of normal links.

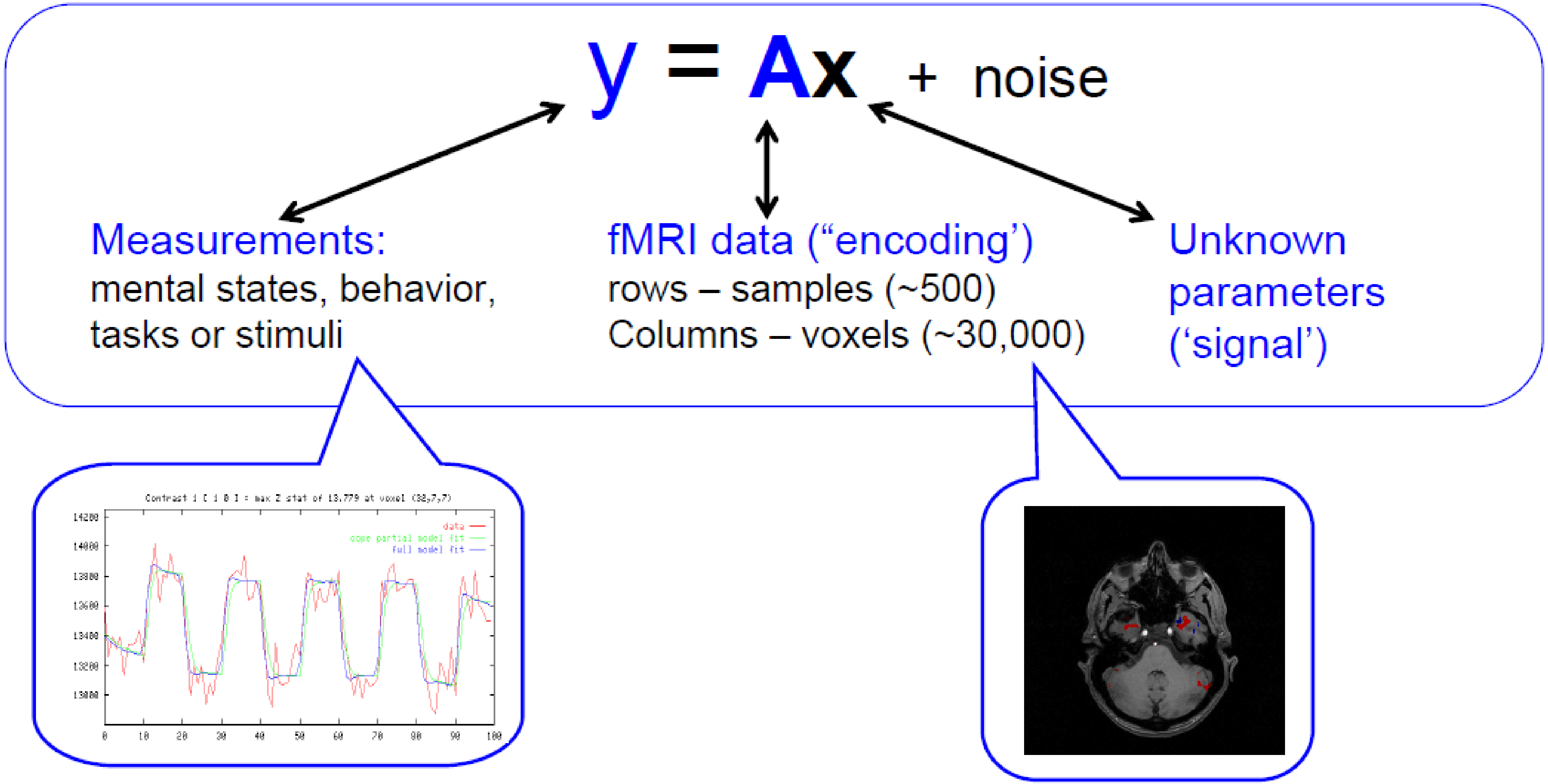
Note that without any constraints on $\boldsymbol{x}$, we cannot obtain a unique solution as it is an underdetermined system of linear equations because number of unknowns is more than the number of equations.

With sparsity constraint, we can get a unique solution for $\boldsymbol{x}$.

The nonzero entries correspond to the bottlenecks or extremely slow links and in practice $M \ll N$ is desired.

In the field of neuroimaging analysis, e.g., medical imaging, one key objective is to discover voxels (brain areas) that are most relevant to a given task, stimulus, or mental state, using functional magnetic resonance imaging (fMRI).

The signal model can also be cast as:



$$\mathbf{y} = \mathbf{A}\mathbf{x} + \text{noise}$$

Measurements:
mental states, behavior,
tasks or stimuli

fMRI data (“encoding’)
rows – samples (~500)
Columns – voxels (~30,000)

Unknown
parameters
(‘signal’)

Given $\boldsymbol{y}$ and $\boldsymbol{A}$, our task is to find a sparse $\boldsymbol{x}$.

The small number of the nonzero entries in $\boldsymbol{x}$ correspond to the most relevant voxels.

Sparse modeling can also be used to increase the efficiency of data compression with the use of transform.

Recall the basic principle of data compression: transform the original signal into another domain such that it looks sparse, then store only the large-magnitude components; when "playing back", perform inverse transform on the transformed data with setting non-dominant components 0.

Example 1

Perform data compression of a discrete-time finite-duration sinusoid expressed as follows:

$$x[n] = \cos(0.125\pi n + 1), \quad n = 0, 1, \cdots, N-1$$

Since it contains only two frequencies: $\pm 0.125\pi$, it is appropriate to transform $x[n]$ to frequency domain via discrete Fourier transform (DFT):

$$X[k]=\sum_{n=0}^{N-1} x[n]e^{-\frac{j2\pi kn}{N}}, \quad 0 \leq k \leq N-1$$

The signal transformation can be viewed in matrix form:

$$\boldsymbol{X}=\boldsymbol{T}\boldsymbol{x}$$

where

$$\boldsymbol{x}=[x[0],\cdots x[N-1]]^T$$

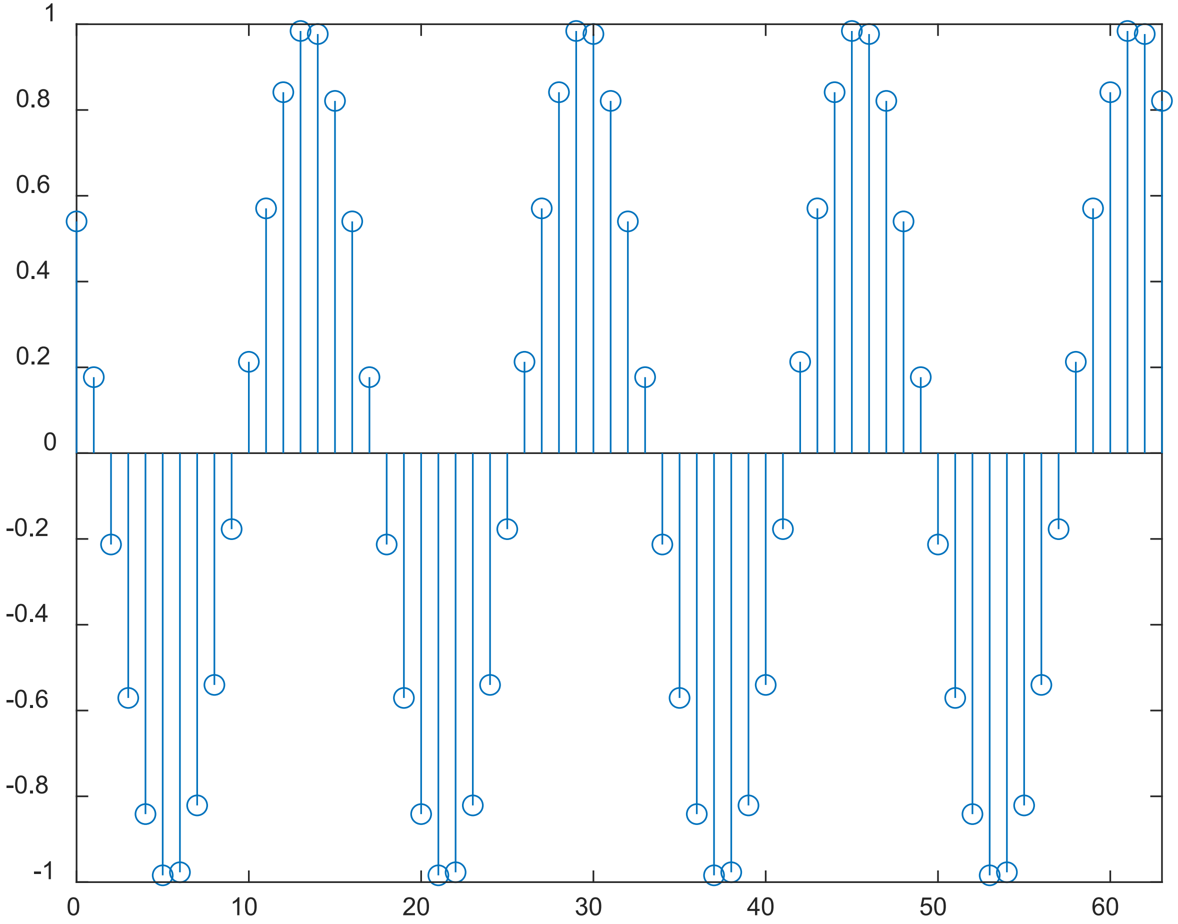
$$\boldsymbol{X}=[X[0],\cdots X[N-1]]^T$$

$$\boldsymbol{T}=\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{-\frac{j2\pi}{N}} & \cdots & e^{-\frac{j2\pi(N-1)}{N}} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & e^{-\frac{j2\pi(N-1)}{N}} & \cdots & e^{-\frac{j2\pi(N-1)^2}{N}} \end{bmatrix} \in \mathbb{C}^{N\times N}$$

is the DFT matrix.

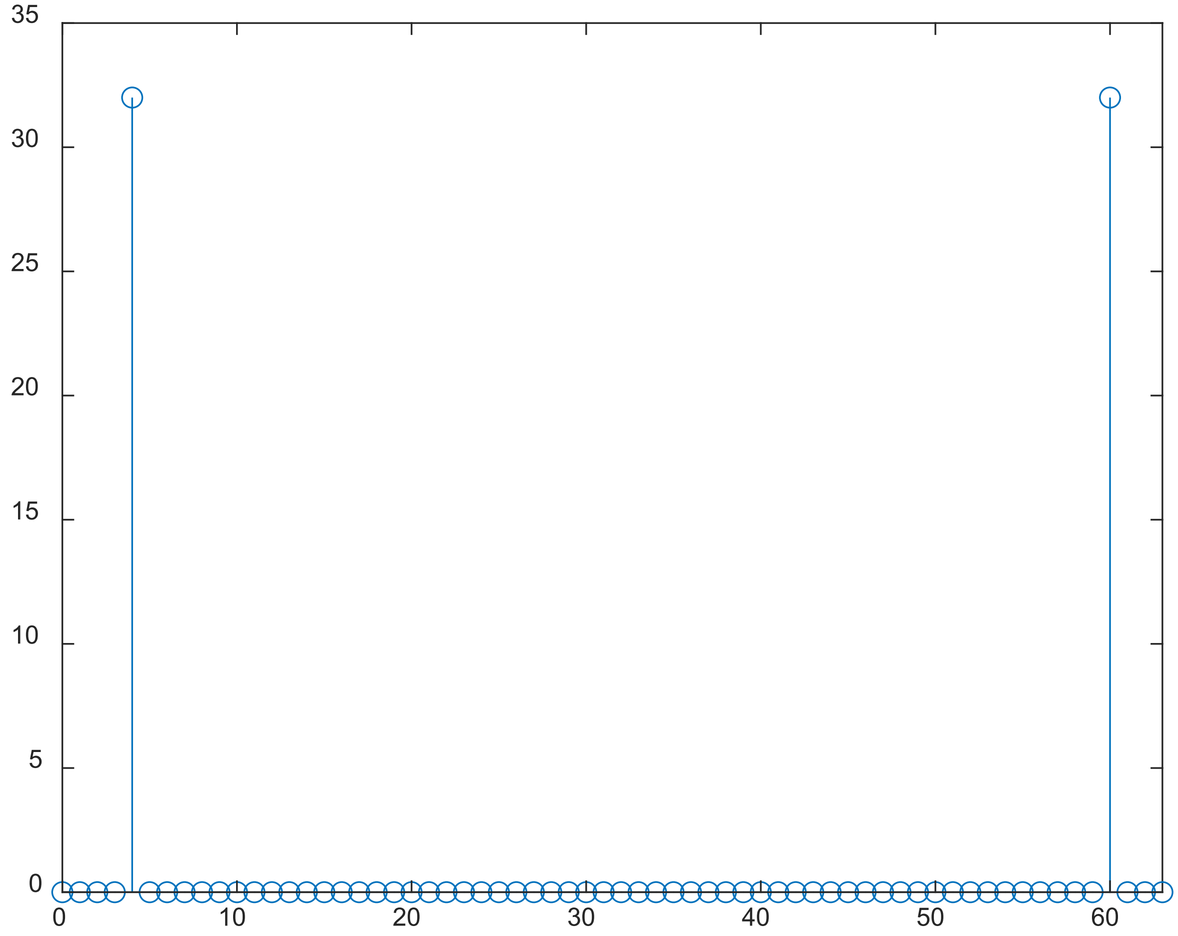
For illustration, we set $N = 64$:

```
>> x=cos(0.125.*pi.*[0:63]+1);
>> stem([0:63],x)
```



In frequency domain, the signal is complex, and we plot the magnitude plot:

```
>> stem([0:63],abs(fft(x)));
```



To perform compression for the sinusoid, we just store two DFT coefficients, $X[4]=17.29+j26.93$ and $X[60]=17.29-j26.93$, which correspond to the two frequency components.

Note that in this example, the other DFT coefficients are 0, indicating lossless compression.

Similarly, for an image, we can perform DCT, and keep only large-magnitude DCT coefficients and set remaining zero.

Based on the concept of sparse modeling, we can perform compression via finding a sparse $\boldsymbol{X}$ given $\boldsymbol{x}$ and $\boldsymbol{A}$:

$$\boldsymbol{x} \approx \boldsymbol{A}\boldsymbol{X} \Leftrightarrow \boldsymbol{T}\boldsymbol{x} \approx \boldsymbol{X}$$

where $\boldsymbol{A}$ is inverse transform matrix. We can skip performing whole transform which computes unused small coefficients, and saves complexity when the length of $\boldsymbol{x}$ is large.

Approaches for Sparse Approximation

In the noise-free scenario, the sparse approximation problem is formulated as:

$$\text{minimize}: \quad \|\boldsymbol{x}\|_0 \quad \text{subject} \quad \text{to}: \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \tag{1}$$

where $\boldsymbol{y} \in \mathbb{R}^M$ is measurement vector, $\boldsymbol{A} \in \mathbb{R}^{M\times N}$ is known matrix, and $\boldsymbol{x} \in \mathbb{R}^N$ is vector to be estimated with minimum number of nonzero entries.

However, minimizing the ℓ_0-norm under the linear constraints is a task of combinatorial nature.

That is, we need to try all valid combinations of zeros in $\boldsymbol{x}$ and identify the one with smallest number of nonzero elements, which is infeasible for large M as the search complexity is exponentially dependent of M.

Example 2

Given

$$\boldsymbol{y} = \begin{bmatrix} 40 \\ 8 \\ 24 \\ 36 \end{bmatrix} \quad \text{and} \quad \boldsymbol{A} = \begin{bmatrix} 8 & 8 & 8 & 8 & 8 & 8 \\ 8 & -8 & 8 & -8 & 8 & -8 \\ 3 & -3 & -6 & 6 & 10 & -10 \\ 3 & 3 & 6 & 6 & 10 & 10 \end{bmatrix}$$

Find the sparsest vector $\boldsymbol{x} \in \mathbb{R}^6$ such that $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$.

Writing $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ as the sum of 6 vectors:

$$\boldsymbol{y} = x_1\boldsymbol{a}_1 + x_2\boldsymbol{a}_2 + x_3\boldsymbol{a}_3 + x_4\boldsymbol{a}_4 + x_5\boldsymbol{a}_5 + x_6\boldsymbol{a}_6$$

where $\boldsymbol{A} = [\boldsymbol{a}_1 \ \cdots \ \boldsymbol{a}_6]$ and $\boldsymbol{x} = [x_1 \ \cdots \ x_6]^T$. Basically we want to find the solution which can remove the column vectors as many as possible.

Note that without any constraint, there will be infinite solutions for $\boldsymbol{x}$.

We may start from picking up only 1 column vector and there are 5 combinations to see if $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ can be satisfied.

If not, we pick 2 column vectors, there are $5!/(2!\,3!) = 10$ combinations to check.

If still not, we pick 3 column vectors, and so on, until $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ is satisfied.

In doing so, we find:

$$\boldsymbol{x} = [0\ 2\ 0\ 0\ 3\ 0]^T$$

It is clear that this approach is not feasible when the length of $\boldsymbol{y}$ is very long.

A feasible approach to solve (1) is to use greedy pursuit, which is built upon a series of locally optimal single-term updates.

Matching pursuit (MP) is one of the representative greedy algorithms and utilizes correlation to find the dominant column of $\boldsymbol{A}$ sequentially, That is, it is an iterative procedure and update the solution until a stopping criterion is reached.

Given $\boldsymbol{y} \in \mathbb{R}^M$ and $\boldsymbol{A} = [\boldsymbol{a}_1 \ \cdots \ \boldsymbol{a}_N] \in \mathbb{R}^{M \times N}$, $\boldsymbol{x} \in \mathbb{R}^N$ is determined using MP as follows.

We start with residual $\boldsymbol{r}^0 = \boldsymbol{y}$ and $\boldsymbol{x}^0 = [x_1^0 \ \cdots \ x_N^0]^T = \mathbf{0}$, and it is desired that after k iterations, $\boldsymbol{r}^k = \mathbf{0}$ and $\boldsymbol{x}^0 = \boldsymbol{x}$.

At the 1st iteration, we determine the column index i_1 of $\boldsymbol{A}$ that has maximal correlation with the residual:

$$i_1 = \arg \max_{n \in \{1, \cdots, N\}} \frac{\left| \boldsymbol{a}_n^T \boldsymbol{r}^0 \right|}{\|\boldsymbol{a}_n\|_2}$$

Then the coefficient of x_{i_1} can be estimated via LS:

$$J(x) = (\boldsymbol{r}^0 - \boldsymbol{a}_{i_i} x)^T (\boldsymbol{r}^0 - \boldsymbol{a}_{i_1} x) = ||\boldsymbol{r}^0 - \boldsymbol{a}_{i_1} x||_2^2 \Rightarrow x_{i_1}^1 = (\boldsymbol{a}_{i_1}^T \boldsymbol{a}_{i_1})^{-1} \boldsymbol{a}_{i_1}^T \boldsymbol{r}^0 = \frac{\boldsymbol{a}_{i_1}^T \boldsymbol{r}^0}{\|\boldsymbol{a}_{i_1}\|_2^2}$$

$$x_{i_1}^1 = x_{i_1}^1 + x_{i_1}^0$$

Then we update the residual:

$$\boldsymbol{r}^1 = \boldsymbol{r}^0 - \boldsymbol{a}_{i_1} x_{i_1}^1$$

and the solution:

$$\boldsymbol{x}^1 = [0 \ \cdots \ x_{i_1}^1 \ \cdots 0]^T$$

which contains only one nonzero entry $\hat{x}_{i_1}$ at the i_1th position.

Similarly, at the 2nd iteration, we perform:

$$i_2 = \arg \max_{n \in \{1, \cdots, N\}} \frac{\left| \boldsymbol{a}_n^T \boldsymbol{r}^1 \right|}{\|\boldsymbol{a}_n\|_2}$$

$$x_{i_2}^2 = x_{i_2}^1 + (\boldsymbol{a}_{i_2}^T \boldsymbol{a}_{i_2})^{-1} \boldsymbol{a}_{i_2}^T \boldsymbol{r}^1 = x_{i_2}^1 + \frac{\boldsymbol{a}_{i_2}^T \boldsymbol{r}^1}{\|\boldsymbol{a}_{i_2}\|_2^2}$$

and

$$\boldsymbol{r}^2 = \boldsymbol{r}^1 - \boldsymbol{a}_{i_2} x_{i_2}$$

with updated solution $\boldsymbol{x}^2$.

This procedure is then repeated until a stopping criterion is met.

Note that the MP allows selecting the same column index more than once.

MP Algorithm

```
**Input**: $\boldsymbol{A}$, $\boldsymbol{y}$
  **Initialization**: Set $k=0$, $\boldsymbol{r}^0=\boldsymbol{y}$, $\boldsymbol{x}^0=\boldsymbol{0}$, index set $\mathcal{I}^0=\emptyset$
  **Repeat**
  $k \leftarrow k+1$
  Select the index $i_k$ via $i_k = \arg \max_{n\in\{1,\cdots,N\}} \frac{\left|\boldsymbol{a}_n^T\boldsymbol{r}^{k-1}\right|}{\|\boldsymbol{a}_n\|_2}$
  Augment the index set $\mathcal{I}^k = \mathcal{I}^{k-1} \cup i_k$
  Update the solution:
      $\boldsymbol{x}^k \leftarrow \boldsymbol{x}^{k-1}$
      $x_{i_k}^k = x_{i_k}^{k-1} + (\boldsymbol{a}_{i_k}^T\boldsymbol{a}_{i_k})^{-1}\boldsymbol{a}_{i_k}^T\boldsymbol{r}^{k-1}$
  Update the residual:
      $\boldsymbol{r}^k = \boldsymbol{r}^{k-1} - \boldsymbol{a}_{i_k}(\boldsymbol{a}_{i_k}^T\boldsymbol{a}_{i_k})^{-1}\boldsymbol{a}_{i_k}^T\boldsymbol{r}^{k-1}$
  Until a stopping criterion is reached
**Output**: The solution is $\boldsymbol{x}^k$
```

Example 3

Find sparse vector $\boldsymbol{x}$ in Example 2 using MP.

```
>> A=[8 8 8 8 8 8;
      8 -8 8 -8 8 -8;
      3 -3 -6 6 10 -10;
    3 3 6 6 10 10];
>> y=[40 8 24 36].'

>>  y.'*A(:,1:6)./[norm(A(:,1))   norm(A(:,2))   norm(A(:,3))
norm(A(:,4)) norm(A(:,5)) norm(A(:,6))]
   46.6770   24.1661   32.2441   43.5578   54.3323   20.7611

>> y.'*A(:,5)/norm(A(:,5))^2
   3.0000

>> r=y- y.'*A(:,5)/norm(A(:,5))^2* A(:,5);
```

At 1st iteration, we have $i_1 = 5$, $\mathcal{I}^1 = \{5\}$ and $x_5^1 = 3$:

$$\boldsymbol{x}^1 = [0\ 0\ 0\ 0\ 3\ 0]^T$$

```
>>  r.'*A(:,1:6)./[norm(A(:,1))  norm(A(:,2))  norm(A(:,3))
norm(A(:,4)) norm(A(:,5)) norm(A(:,6))]
    0.0000   24.1661    5.0912   18.1019    0.0000   20.7611
```

```
>> r.'*A(:,2)/norm(A(:,2))^2
   2.0000
```

```
>> r=r- r.'*A(:,2)/norm(A(:,2))^2* A(:,2)
```

At 2nd iteration, we have $i_1 = 2$, $\mathcal{I}^2 = \{2, 5\}$ and $x_2^2 = 2$:

$$\boldsymbol{x}^2 = [0\ 2\ 0\ 0\ 3\ 0]^T$$

It can be checked that the residual $\boldsymbol{r}^2 \approx \mathbf{0}$, we may terminate the MP and take $\boldsymbol{x}^2$ as the solution.

Orthogonal MP (OMP) is an improvement to MP such that a column will be at most selected once.

At the 1st iteration, we determine the column index i_1 of $\boldsymbol{A}$ that is most correlated with the residual:

$$i_1 = \arg \max_{n \in \{1,\cdots,N\}} \frac{\left|\boldsymbol{a}_n^T \boldsymbol{r}^0\right|}{\|\boldsymbol{a}_n\|_2}, \quad \mathcal{I}^1 = \mathcal{I}^0 \cup i_1$$

Then the coefficient of x_{i_1} can be estimated via LS:

$$J(x) = (\boldsymbol{y} - \boldsymbol{a}_{i_i} x)^T (\boldsymbol{y} - \boldsymbol{a}_{i_1} x) = ||\boldsymbol{y} - \boldsymbol{a}_{i_1} x||_2^2 \Rightarrow x_{i_1} = (\boldsymbol{a}_{i_1}^T \boldsymbol{a}_{i_1})^{-1} \boldsymbol{a}_{i_1}^T \boldsymbol{r}^0$$

Then we update the residual:

$$\boldsymbol{r}^1 = \boldsymbol{y} - \boldsymbol{a}_{i_1} x_{i_1}$$

At the 2nd iteration, we perform:

$$i_2 = \arg \max_{n \in \{1,\cdots,i_1-1,i_1+1,\cdots,N\}} \frac{\left|\boldsymbol{a}_n^T \boldsymbol{r}^1\right|}{\|\boldsymbol{a}_n\|_2}, \quad \mathcal{I}^2 = \mathcal{I}^1 \cup i_2$$

Let $\boldsymbol{A}_{\mathcal{I}^2} = [\boldsymbol{a}_{i_1}, \boldsymbol{a}_{i_2}]$. We update $\boldsymbol{x}_{\mathcal{I}^2} = [x_{i_1}, x_{i_2}]^T$ together via LS:

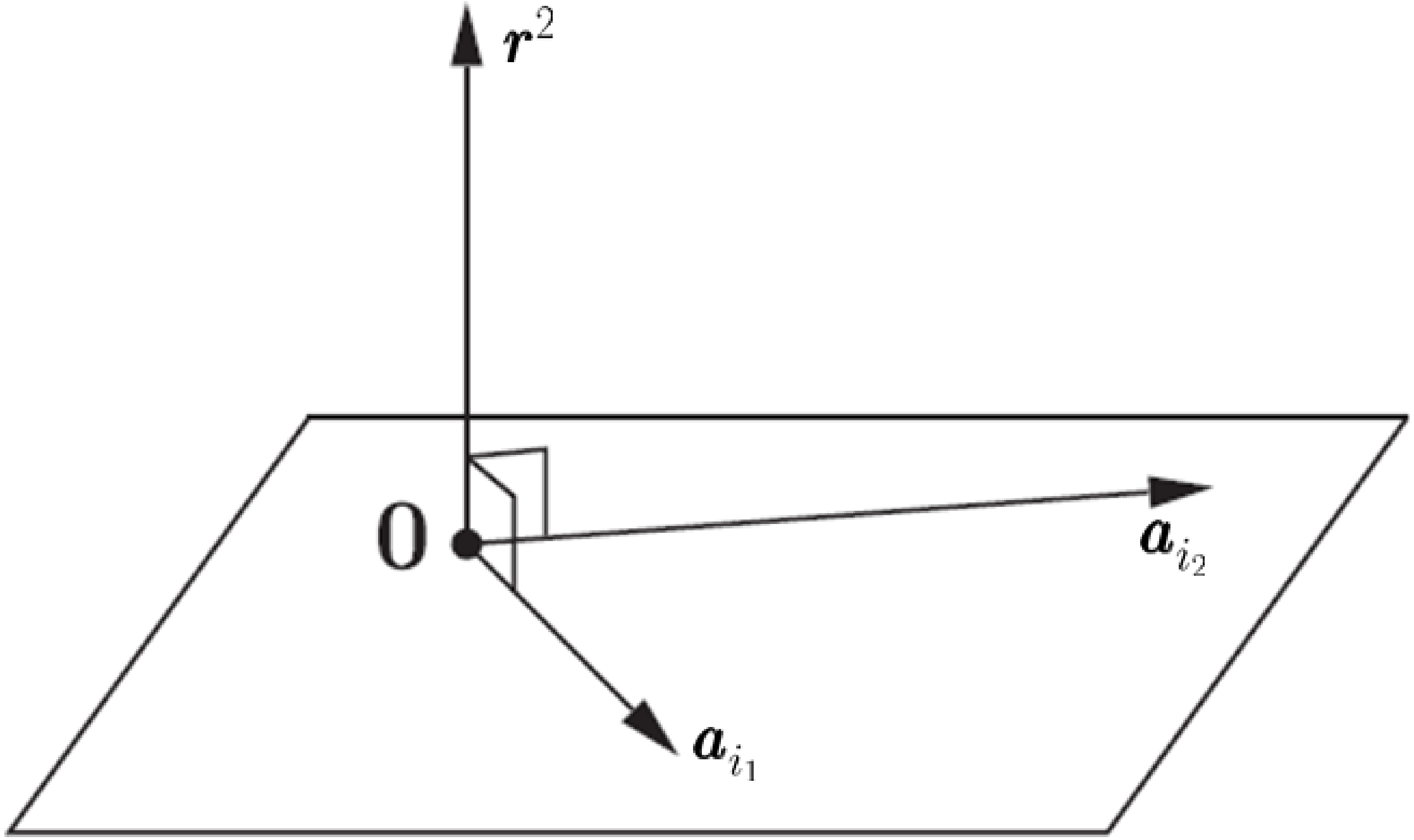
$$\text{minimize} \quad \|\boldsymbol{y} - \boldsymbol{A}_{\mathcal{I}^2}\boldsymbol{x}_{\mathcal{I}^2}\|_2^2 \Rightarrow \boldsymbol{x}_{\mathcal{I}^2} = \left(\boldsymbol{A}_{\mathcal{I}^2}^T \boldsymbol{A}_{\mathcal{I}^2}\right)^{-1} \boldsymbol{A}_{\mathcal{I}^2}^T \boldsymbol{y}$$

Then we update the residual:

$$\boldsymbol{r}^2 = \boldsymbol{y} - \boldsymbol{A}_{\mathcal{I}^2}\boldsymbol{x}_{\mathcal{I}^2}$$

Note that i_1 is excluded for selection and x_{i_1} is recomputed.

This procedure is then repeated until a stopping criterion is met.



As an illustration in 3D case, the residual vector $\boldsymbol{r}^2$ is orthogonal to the subspace spanned by the currently available set of active columns, $\boldsymbol{a}_{i_1}$ and $\boldsymbol{a}_{i_2}$.

OMP Algorithm

```
Input: $\boldsymbol{A}$, $\boldsymbol{y}$
  Initialization: Set $k=0$, $\boldsymbol{r}^0=\boldsymbol{y}$, $\boldsymbol{x}^0=\boldsymbol{0}$, index set $\mathcal{I}^0=\emptyset$
  Repeat
    $k \leftarrow k+1$
    Select the index $i_k$ via $i_k = \arg\max_{n \notin \mathcal{I}^{k-1}} \frac{\left|\boldsymbol{a}_n^T \boldsymbol{r}^{k-1}\right|}{\|\boldsymbol{a}_n\|_2}$
    Augment the index set $\mathcal{I}^k = \mathcal{I}^{k-1} \cup i_k$
    Assign $\boldsymbol{A}_{\mathcal{I}^k} = [\boldsymbol{A}_{\mathcal{I}^{k-1}}, \boldsymbol{a}_{i_k}]$ with $\boldsymbol{A}_{\mathcal{I}^0} = \emptyset$
    Update the solution:
        $\boldsymbol{x}_{\mathcal{I}^k} = \left(\boldsymbol{A}_{\mathcal{I}^k}^T \boldsymbol{A}_{\mathcal{I}^k}\right)^{-1} \boldsymbol{A}_{\mathcal{I}^k}^T \boldsymbol{y}$
    Update the residual:
        $\boldsymbol{r}^k = \boldsymbol{y} - \boldsymbol{A}_{\mathcal{I}^k} \boldsymbol{x}_{\mathcal{I}^k}$
  Until a stopping criterion is reached
Output: $\boldsymbol{x}_{\mathcal{I}^k}$, $\mathcal{I}^k$
```

Example 4

Find sparse vector $\boldsymbol{x}$ in Example 2 using OMP.

Similar to MP, at 1st iteration, we have $i_1 = 5$, $\mathcal{I}^1 = \{5\}$ and $\boldsymbol{x}_{\mathcal{I}^1} = 3$:

```
>> r.'*A(:,[1:4,6])./[norm(A(:,1)) norm(A(:,2)) norm(A(:,3))
norm(A(:,4)) norm(A(:,6))]
    0.0000   24.1661    5.0912   18.1019   20.7611

>> Ai=[(A(:,5)) (A(:,2))];
>> x=inv(Ai.'*Ai)*Ai.'*y
   3
   2

>> r=y- Ai*x
```

It can be checked that the residual $\boldsymbol{r}^2 \approx \mathbf{0}$, we may terminate the OMP and use $\boldsymbol{x}_{\mathcal{I}^2}$ and $\mathcal{I}^2$ to produce the solution:

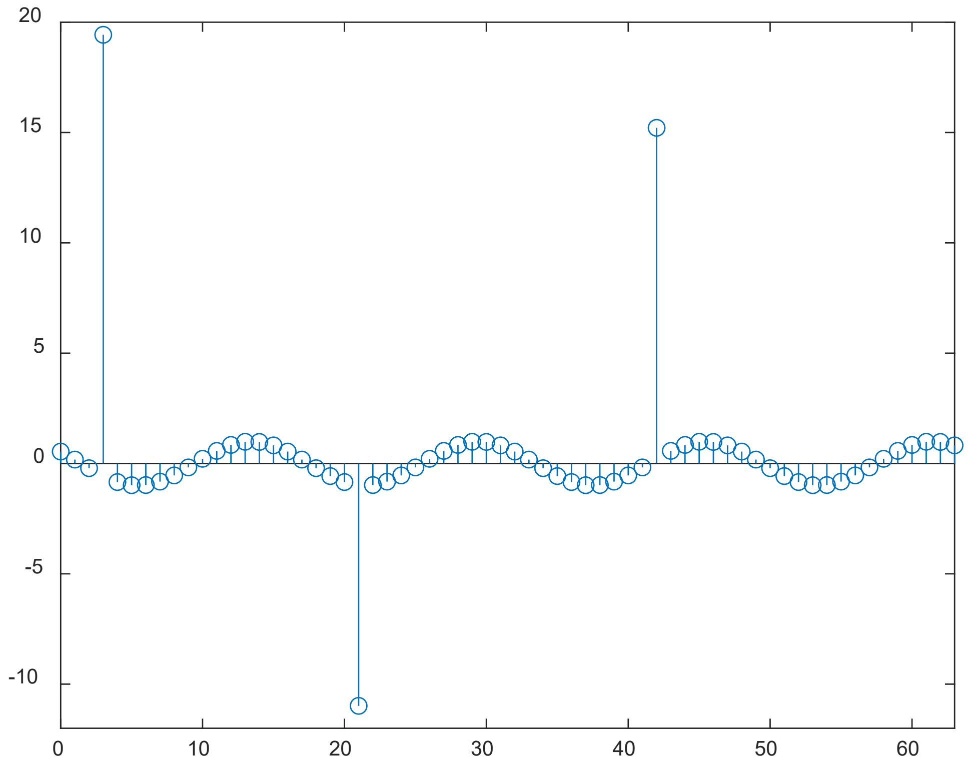
$$\boldsymbol{x} = [0\ 2\ 0\ 0\ 3\ 0]^T$$

Example 5

Suppose we have a signal:

$$x[n] = \cos(0.125\pi n + 1) + 20\delta[n-4] - 10\delta[n-22] + 15\delta[n-43]$$

for $n = 0, \cdots, N-1$ with $N = 64$. Discuss how to perform compression using MP and OMP.



It comprises the sum of a sinusoid and spiky-like pulses.

To compress the sinusoid, we choose inverse DFT (IDFT) and its matrix is given by:

$$\boldsymbol{T}^{-1} = \frac{1}{N} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{\frac{j2\pi}{N}} & \cdots & e^{\frac{j2\pi(N-1)}{N}} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & e^{\frac{j2\pi(N-1)}{N}} & \cdots & e^{\frac{j2\pi(N-1)^2}{N}} \end{bmatrix}$$

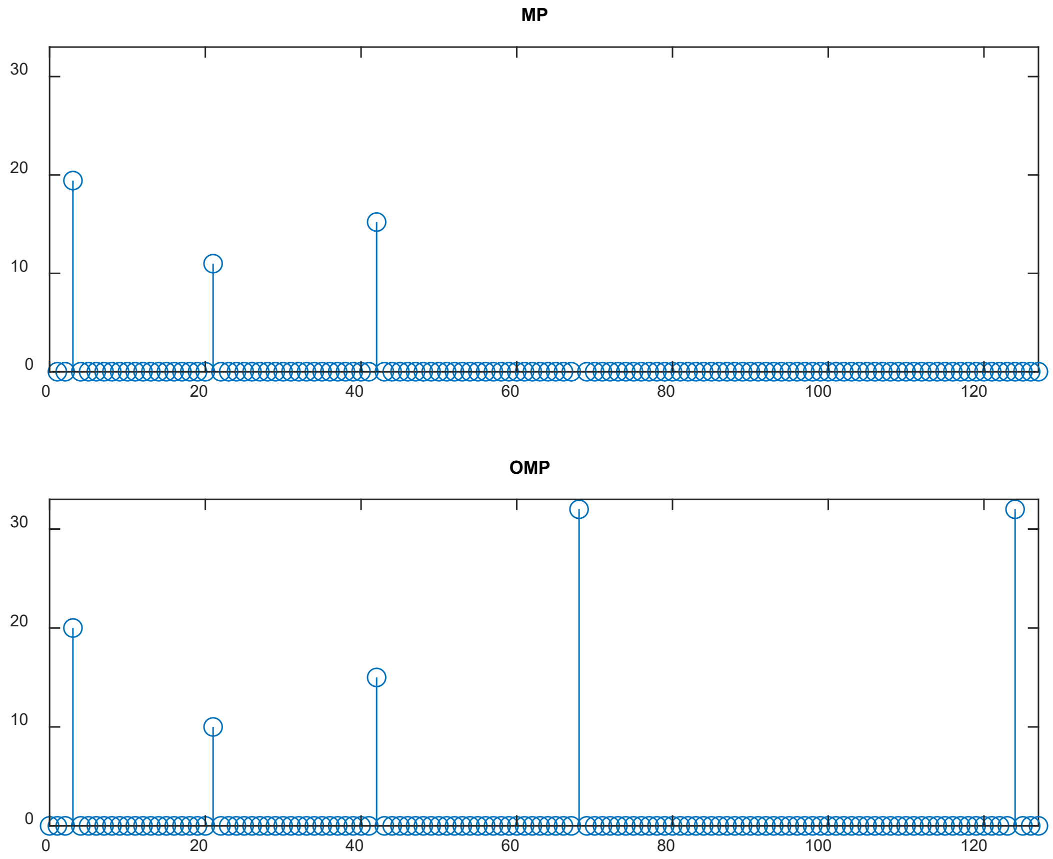
As the spikes are already sparse in the time domain, we choose a $N \times N$ identity matrix $\boldsymbol{I}$ for its sparse representation.

Hence the matrix $\boldsymbol{A} \in \mathbb{R}^{N \times 2N}$ is:

$$\boldsymbol{A} = [\boldsymbol{I} \;\; \boldsymbol{T}^{-1}]$$

From Example 1, we know that the sinusoid can be represented as 2 DFT coefficients in the frequency domain.

As a result, we expect 5 nonzero entries in this domain at the 4th, 22nd, 43rd, 68th (4+64), 124th (60+64) positions.



The MP and OMP are terminated when $\|\boldsymbol{r}^k\|_2 < 10^{-6}$ is reached.

If the number of nonzero entries is known, we can also set $k=5$ as the stopping criterion.

MP and OMP may not necessarily provide the global optimum solution because they perform optimization in a local manner. In fact, MP can only get a local solution.

It is only guaranteed that $\|\boldsymbol{r}^{k+1}\|_2 < \|\boldsymbol{r}^k\|_2$, i.e., the residual vector decreases at every iteration step.

OMP and MP can also be used for the noisy model $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{q}$:

$$\text{minimize}: \quad \|\boldsymbol{x}\|_0 \quad \text{subject} \quad \text{to}: \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_2^2 \leq \epsilon \tag{2}$$

and $\|\boldsymbol{r}^k\|_2$ cannot reach 0 because of the noise component $\boldsymbol{q}$

OMP generally performs better than MP.

Although OMP and MP can deal with (1) and (2), the global solution is not guaranteed.

Another approach which can ensure a global solution is to approximate the ℓ_0-norm using the nearest convex norm ℓ_1-norm:

$$\text{minimize}: \quad ||\boldsymbol{x}||_1 \quad \text{subject} \quad \text{to}: ||\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}||_2^2 \leq \epsilon \tag{3}$$

which is called basis pursuit de-noising (BPDN). Alternatively, we can express (3) as ℓ_1 regularization:

$$\text{minimize}: \quad J(\boldsymbol{x}) = ||\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}||_2^2 + \lambda||\boldsymbol{x}||_1 \tag{4}$$

or least absolute shrinkage and selection operator (LASSO):

$$\text{minimize}: \quad ||\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}||_2^2, \quad \text{subject} \quad \text{to}: \quad ||\boldsymbol{x}||_1 \leq \rho \tag{5}$$

That is, (3), (4) and (5) are equivalent for specific choices of ϵ, λ and ρ.

We deal with the unconstrained formulation (4).

When $\lambda = 0$, it becomes the standard LS:

$$\left.\frac{dJ(\boldsymbol{x})}{d\boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}_{\mathrm{LS}}} = -2\boldsymbol{A}^T\boldsymbol{y} + 2\boldsymbol{A}^T\boldsymbol{A}\hat{\boldsymbol{x}}_{\mathrm{LS}} = \mathbf{0} \Rightarrow \hat{\boldsymbol{x}}_{\mathrm{LS}} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{y}$$

When $\lambda \neq 0$, the solution is found in

$$-2\boldsymbol{A}^T\boldsymbol{y} + 2\boldsymbol{A}^T\boldsymbol{A}\hat{\boldsymbol{x}} + \lambda \mathrm{sign}(\hat{\boldsymbol{x}}) = \mathbf{0}$$

To get better insight, we consider $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}$ so that closed-form solution can be obtained.

Now we have:

$$\hat{\boldsymbol{x}}_{\mathrm{LS}} = \boldsymbol{A}^T\boldsymbol{y} \qquad \text{and} \qquad -2\boldsymbol{A}^T\boldsymbol{y} + 2\hat{\boldsymbol{x}} + \lambda \mathrm{sign}(\hat{\boldsymbol{x}}) = \mathbf{0}$$

$$\Rightarrow -\hat{x}_{\mathrm{LS},i} + \hat{x}_i + \frac{\lambda}{2}\mathrm{sign}(\hat{x}_i) = 0$$

Recall:

$$\frac{d|x|}{dx} = \mathrm{sign}(x) = \begin{cases} 1, & x > 0 \\ -1, & x < 0 \\ [-1, 1], & x = 0 \end{cases}$$

When $\hat{x}_i = 0$, we get:

$$\hat{x}_i = 0 \Rightarrow -\hat{x}_{\mathrm{LS},i} + \frac{\lambda}{2}\mathrm{sign}(0) = 0 \Rightarrow |\hat{x}_{\mathrm{LS},i}| \leq \frac{\lambda}{2}$$

For $\hat{x}_i \neq 0$, the solution is:

$$\hat{x}_i = \begin{cases} \hat{x}_{\mathrm{LS},i} - \frac{\lambda}{2}, & x > 0 \\ \hat{x}_{\mathrm{LS},i} + \frac{\lambda}{2}, & x < 0 \end{cases}$$

Combining the results yields

$$\hat{x}_i = \text{sign}(\hat{x}_{\text{LS},i}) \left(|\hat{x}_{\text{LS},i}| - \frac{\lambda}{2} \right)_+ \tag{6}$$

where

$$u_+ = \begin{cases} u, & u \geq 0 \\ 0, & u < 0 \end{cases}$$

Hence the ℓ_1 regularization can force some elements of $\hat{\boldsymbol{x}}$ to zero if $|\hat{x}_{\text{LS},i}| \leq \lambda/2$, indicating sparsity can be promoted.

From (6), it is observed how λ controls the sparsity: if it is large, then there are more zero entries, while if it is small, then there are less zero entries.

Example 6

Suppose the LS solution of $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{q}$ with $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}$ is:

$$\hat{\boldsymbol{x}} = [0.2 \ -0.7 \ 0.8 \ -0.1 \ 1.0]^T$$

Find the ℓ_1 regularization solution with $\lambda = 1$. Compare the result with the ℓ_2 regularization.

Using (6), the ℓ_1 regularization solution with $\lambda = 1$ is:

$$\hat{\boldsymbol{x}} = [0.0 \ -0.2 \ 0.3 \ 0 \ 0.5]^T$$

with 2 zero entries. When $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}$, the ℓ_2 regularization solution with $\lambda = 1$ is:

$$\left(\boldsymbol{A}^T\boldsymbol{A} + \lambda\boldsymbol{I}\right)\hat{\boldsymbol{x}} = \boldsymbol{A}^T\boldsymbol{y} \Rightarrow (1+\lambda)\hat{\boldsymbol{x}} = \boldsymbol{A}^T\boldsymbol{y} \Rightarrow \hat{\boldsymbol{x}} = \frac{\boldsymbol{A}^T\boldsymbol{y}}{1+\lambda} = \frac{\hat{\boldsymbol{x}}_{\mathrm{LS}}}{1+\lambda} \qquad (7)$$

$$\hat{\boldsymbol{x}} = [0.1 \ -0.35 \ 0.4 \ -0.05 \ 0.5]^T$$

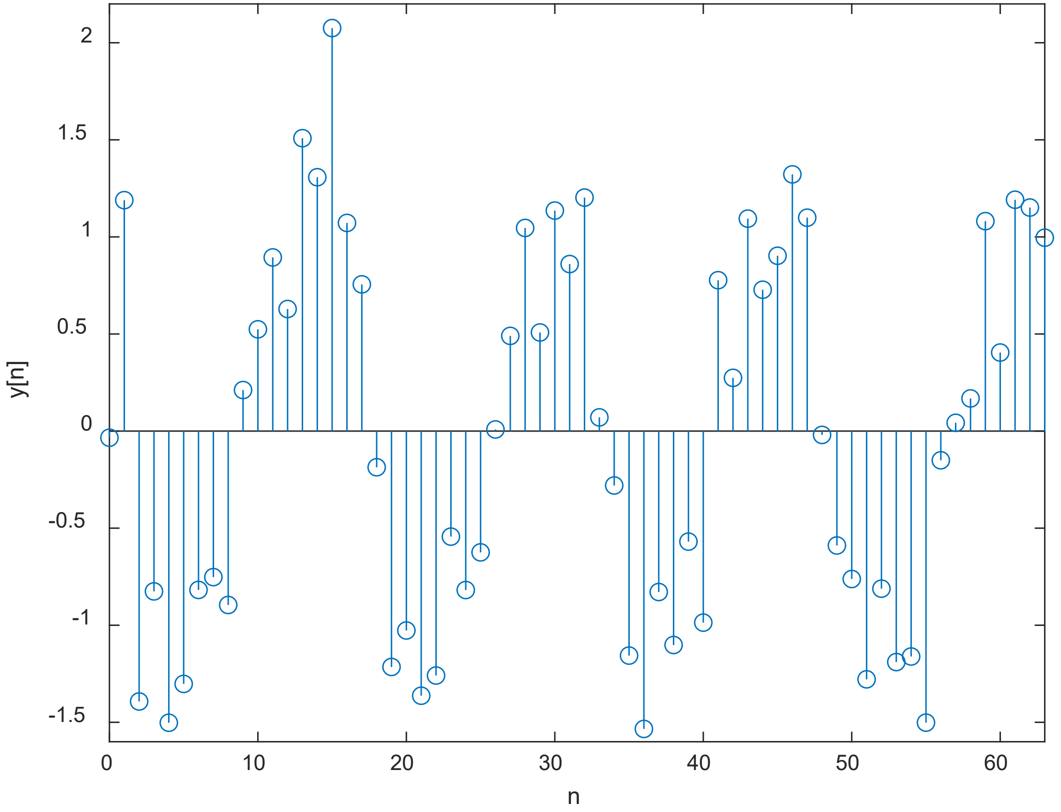
where the LS solution is scaled by $(1+\lambda)$ which is not sparse.

Example 7

Suppose we have a signal:

$$y[n] = \cos(0.125\pi n + 1) + q[n], \quad n = 0, \cdots, N-1, \quad N = 64$$

where $q[n]$ is an i.i.d. Gaussian noise with power of 0.25.



Discuss how to find its sparse approximation using ℓ_1 regularization.

Based on $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$, we consider using the scaled IDFT matrix:

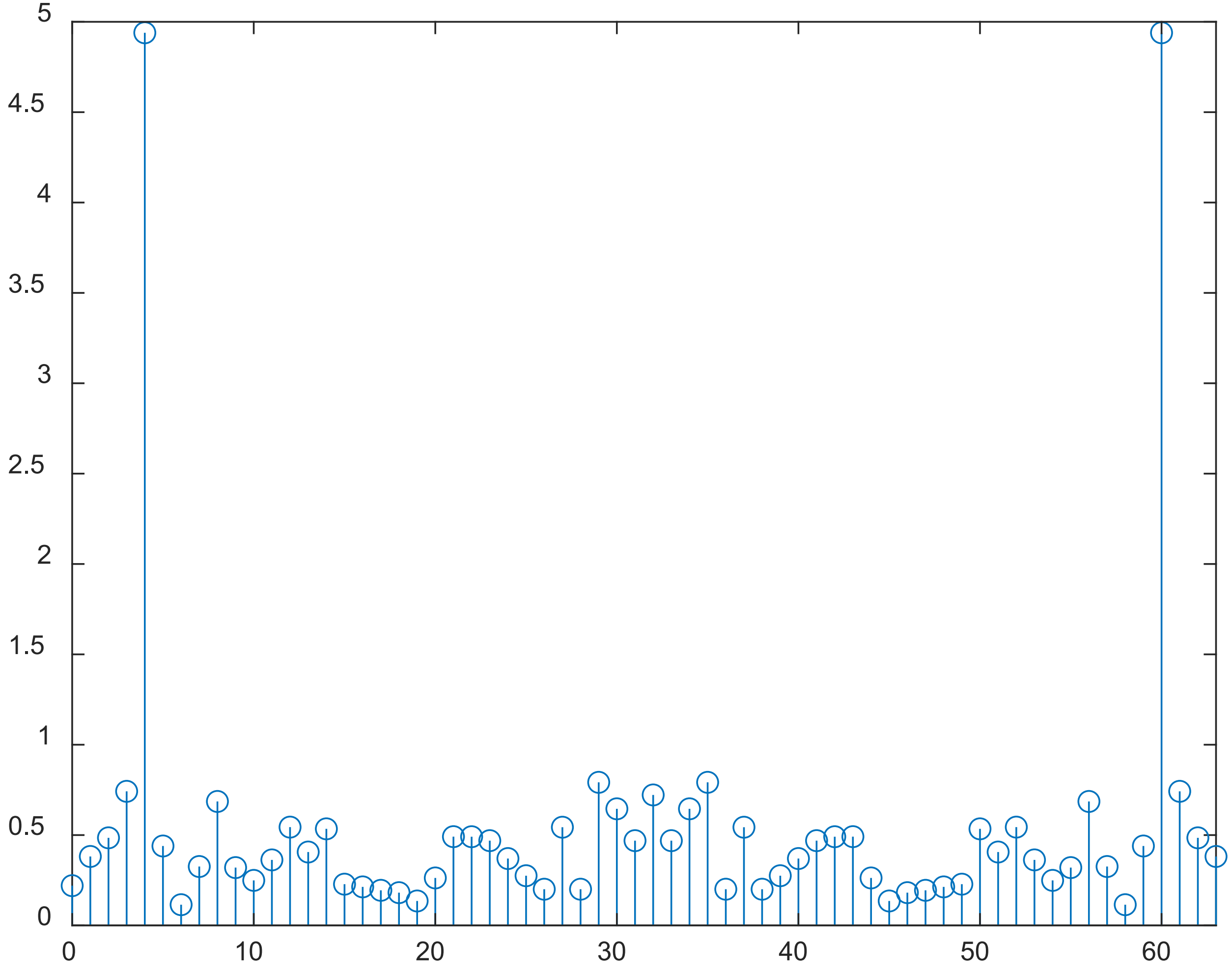
$$\boldsymbol{A} = \frac{1}{\sqrt{N}} \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & e^{\frac{j2\pi}{N}} & \cdots & e^{\frac{j2\pi(N-1)}{N}} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & e^{\frac{j2\pi(N-1)}{N}} & \cdots & e^{\frac{j2\pi(N-1)^2}{N}} \end{bmatrix}, \quad \boldsymbol{A}^H\boldsymbol{A} = \boldsymbol{I}$$

The LS solution is shown to be

$$\hat{\boldsymbol{x}}_{\mathrm{LS}} = \boldsymbol{A}^H\boldsymbol{y}$$

Following the idea of (6), the sparse approximation is obtained via removing the entries with small magnitudes.

That is, we may just keep the 4th and 64th entries, which corresponds to the sinusoid as shown in Example 1.



However, when $\boldsymbol{A}$ is not orthogonal, we may not be able to obtain a closed-form solution.

A simple solution is to use the gradient descent to find the solution in an iterative manner:

$$\hat{\boldsymbol{x}}^{k+1} = \hat{\boldsymbol{x}}^{k} - \mu \boldsymbol{\nabla}(J(\hat{\boldsymbol{x}}^{k}))$$

Applying the chain rule, we have:

$$\begin{aligned}\frac{d(\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x})^T(\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x})}{d\boldsymbol{x}} &= \frac{d(\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x})}{d\boldsymbol{x}} \cdot \frac{d(\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x})^T(\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x})}{d(\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x})} \\ &= -\boldsymbol{A}^T \cdot 2(\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}) \cdot = -2\boldsymbol{A}^T(\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x})\end{aligned}$$

Combining the results yields:

$$\boldsymbol{\nabla}(J(\hat{\boldsymbol{x}}^{k})) = -2\boldsymbol{A}^T(\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}^k) + \lambda \text{sign}(\boldsymbol{x}^k)$$

or

$$\hat{\boldsymbol{x}}^{k+1} = \hat{\boldsymbol{x}}^{k} + \mu\left[2\boldsymbol{A}^T(\boldsymbol{y}-\boldsymbol{A}\boldsymbol{x}^k) - \lambda \text{sign}(\boldsymbol{x}^k)\right] \quad (7)$$

Example 8

Consider a system of linear equations:

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{q}$$

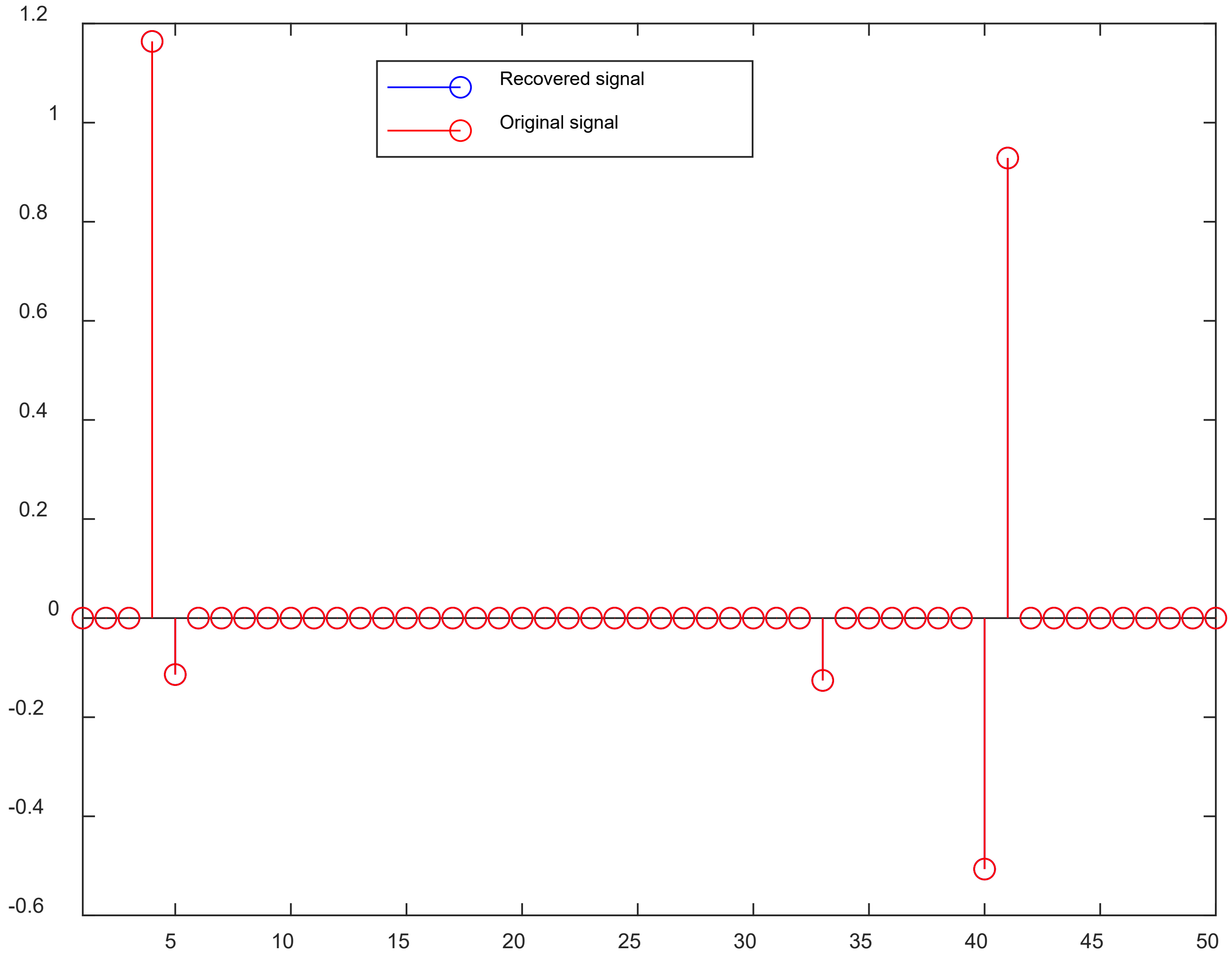
where $\boldsymbol{y} \in \mathbb{R}^{20}$, $\boldsymbol{A} \in \mathbb{R}^{20\times 50}$, $\boldsymbol{x} \in \mathbb{R}^{50}$ and $\boldsymbol{q} \in \mathbb{R}^{20}$. Given $\boldsymbol{y} \in \mathbb{R}^{20}$ and $\boldsymbol{A} \in \mathbb{R}^{20\times 50}$, the task is to find $\boldsymbol{x}$ which is a sparse vector.

To validate (7), we first construct a sparse $\boldsymbol{x}$ with only 5 nonzero entries:

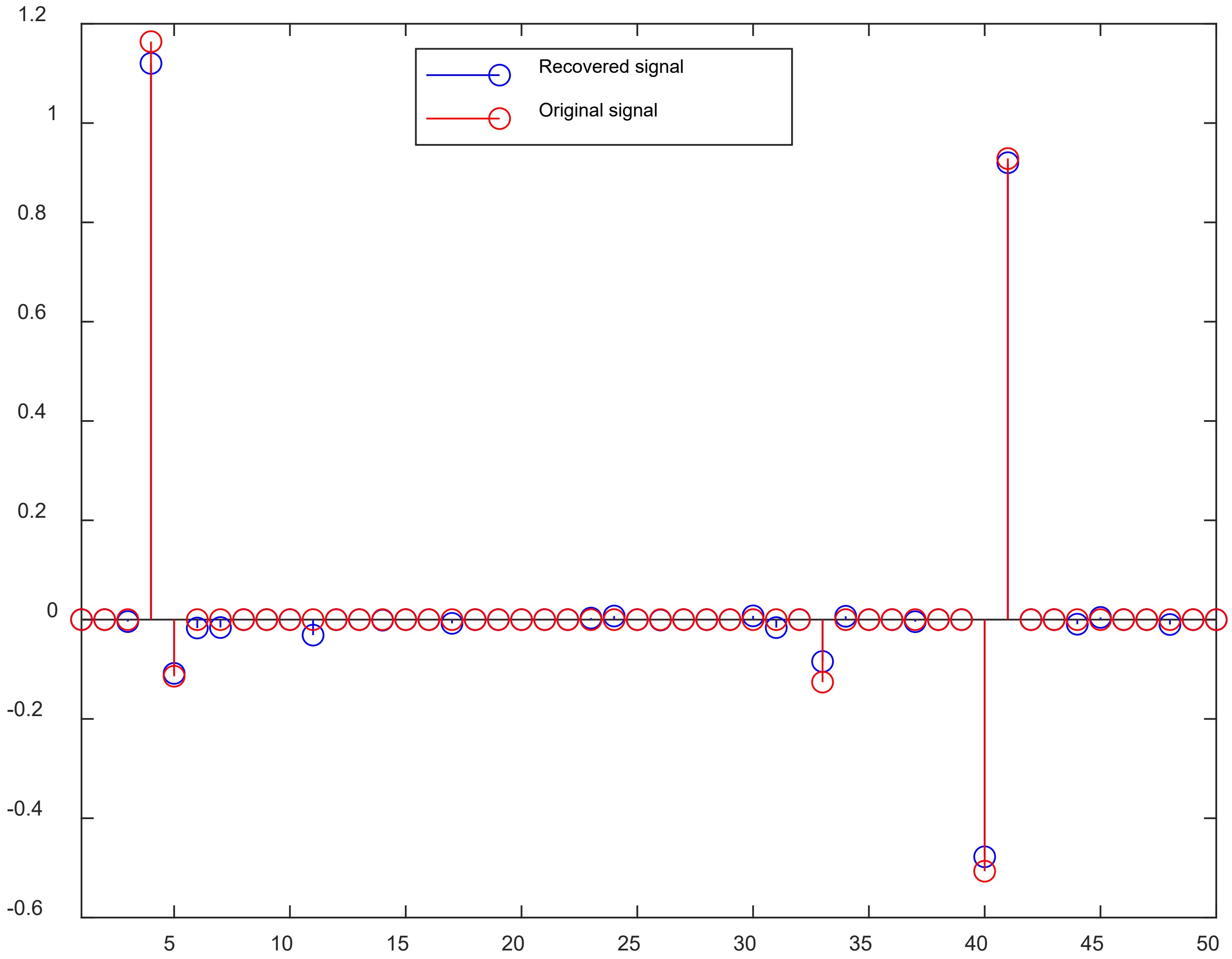
4th	1.1640	5th	-0.1140	33rd	-0.1258
40th	-0.5066	41st	0.9286		

Then we construct $\boldsymbol{A}$ where each element is an i.i.d. Gaussian variable with unit variance, and the noise-free $\boldsymbol{y}$ is then generated by $\boldsymbol{A}\boldsymbol{x}$.

Result of $q=0$:


Recovered signal
Original signal
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
5
10
15
20
25
30
35
40
45
50

Result of noisy case when power of $\boldsymbol{q} = 0.0025$:



References:

1. S. Theodoridis, *Machine Learning: A Bayesian and Optimization Perspective*, Academic Press, 2015
2. I. Rish and G. Grabarnik, *Sparse Modeling: Theory, Algorithms, and Applications*, CRC Press, 2014
3. Z. Zhang, Y. Xu, J. Yang, X. Li and D. Zhang, “A survey of sparse representation: Algorithms and applications,” *IEEE Access*, vol. 3, pp. 490-530, 2015
4. S. Boyd and L. Vandenberghe, *Convex Optimization*, Cambridge University Press, 2004