# HANDOUT

**Materi :**

## Penginderaan Kompresif (Compressive Sensing)


OLEH

**TEAM PENYUSUN**

**Prof. Andriyan Bayu Suksmono, Ph.D**
**Dr. Koredianto Usman, S.T., M.Sc.**

**Fakultas Teknik Elektro**




**Telkom University Press**
**2019**

# Kata Pengantar

Puji dan syukur para penulis sampaikan ke hadirat Tuhan Yang Maha Esa atas selesainya penyusunan *handout* Penginderaan Kompresif (Compressive Sensing) ini. Materi tentang penginderaan kompresif masih langka di pasaran, di samping keilmuannya yang relatif baru, materi ini juga spesifik pada bidang pengolahan sinyal, sehingga hanya kalangan ini yang bersentuhan dengan teknik ini. Dengan motivasi kelangkaan ini, para penulis berusaha untuk menyajikan materi dasar sehingga konsep-konsep penginderaan kompresif dapat dipelajari dengan mudah sebelum para pembaca terjun ke teks-teks yang lebih serius.

Handout ini telah diujicobakan pada acara Summer School Juli 2019 pada lingkungan Fakultas Teknik Elektro, Universitas Telkom. Masukan dan saran dari semua pihak untuk perbaikan handout ini kami terima dengan rasa senang dan terima kasih.

Bandung, 19 Agustus 2019
Team penulis

# Part III

# Practical Aspects

# SUMMER SCHOOL 2019

## PART I : THEORY OF COMPRESSIVE SENSING

By : Prof. Andriyan B. Suksmono, Ph.D
Dr. Koredianto Usman

July 15, 2010

Institut Teknologi Bandung and Telkom University

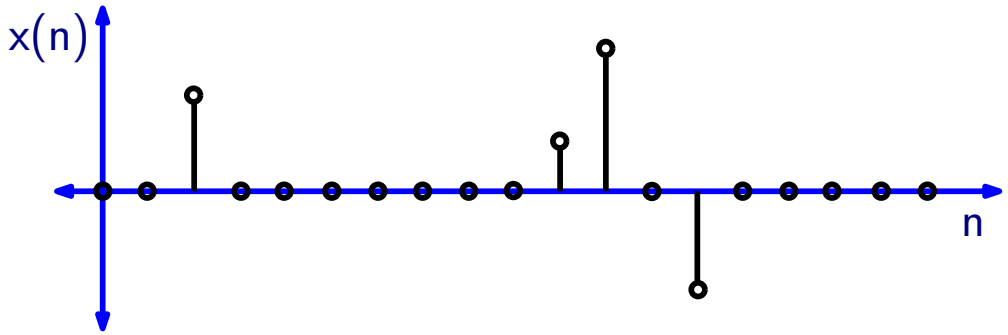
# Objectives

1. to introduce some fundamental concept in Compressive Sensing (CS)

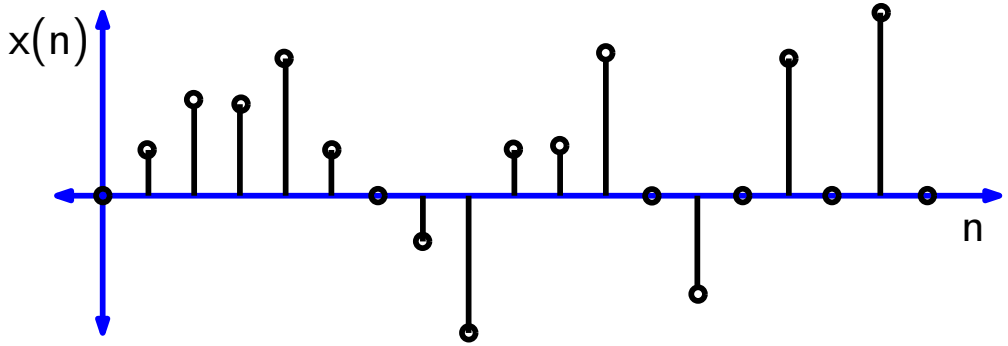
# Table of content

# Sparse Signals

1. Sparse signal is the key assumption in compressive sensing
2. it is defined as: discrete signal with a lot of zeros and a small amount of non-zeros values
3. An example of discrete, time domain, sparse signal (*sparsity*: $k = 4$):

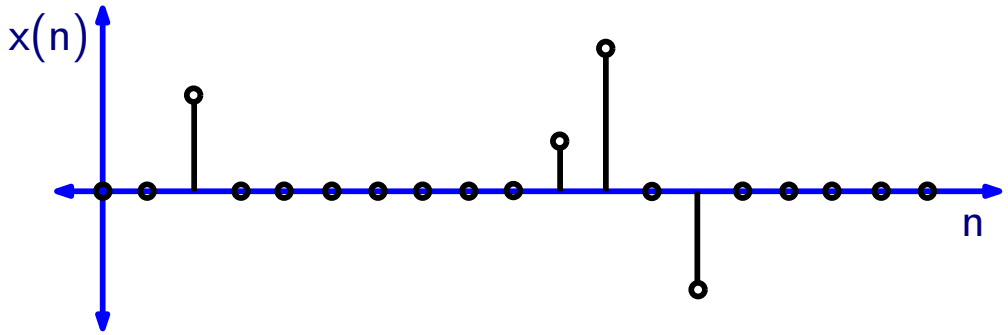



4. An example of non-sparse signal

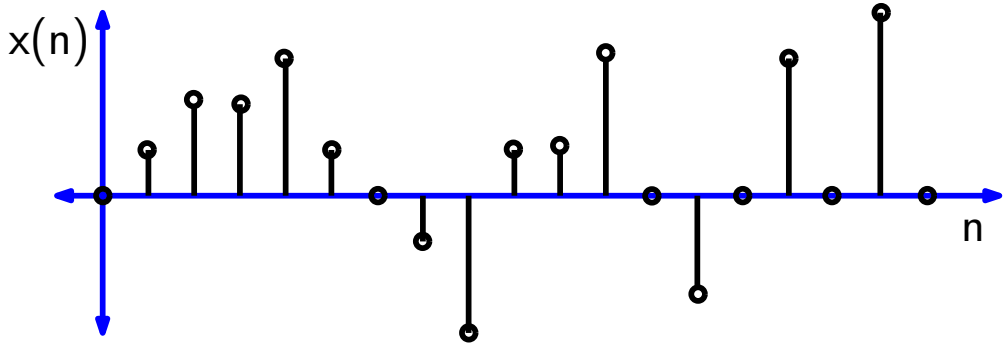


# Sparse Signals

1. Sparse signal is the key assumption in compressive sensing
2. it is defined as: discrete signal with a lot of zeros and a small amount of non-zeros values
3. An example of discrete, time domain, sparse signal (*sparsity*: $k = 4$):

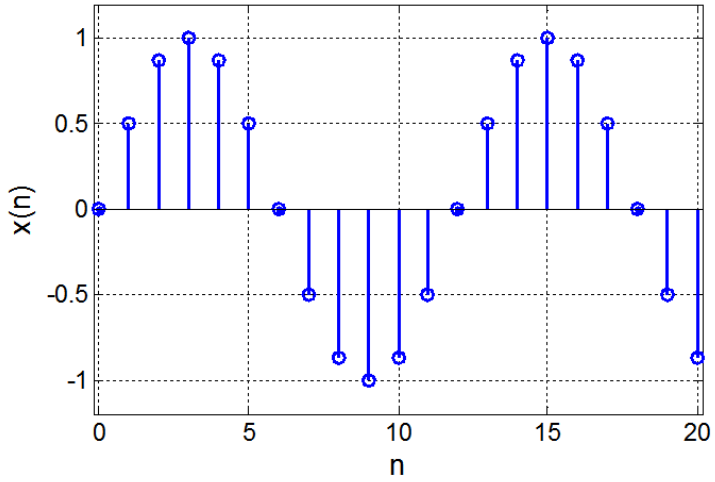



4. An example of non-sparse signal

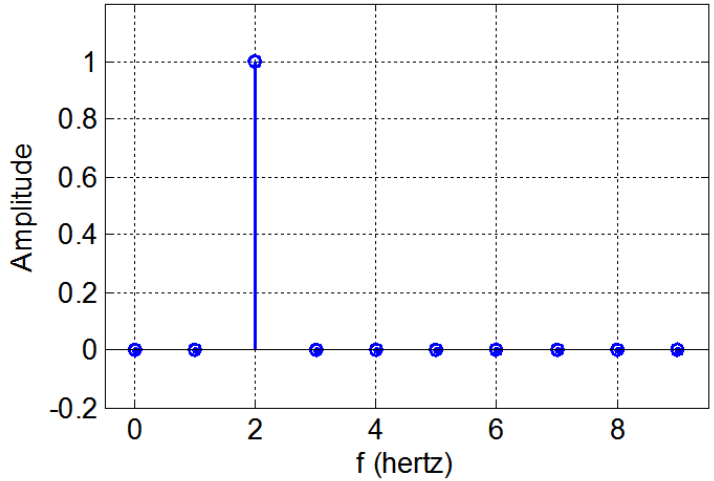


# Sparse signal : Some concepts

1. *sparsity level* ($k$) = number of non-zeros element
2. minimum sparsity = *1*
3. Sparsity can be extended in any bases
4. For example, **sinusoidal signal**: *not sparse in time domain*, but *sparse in frequency domain*.




Time domain sinusoid signal




Sparse frequency domain signal

# Sparse signal : Some concepts

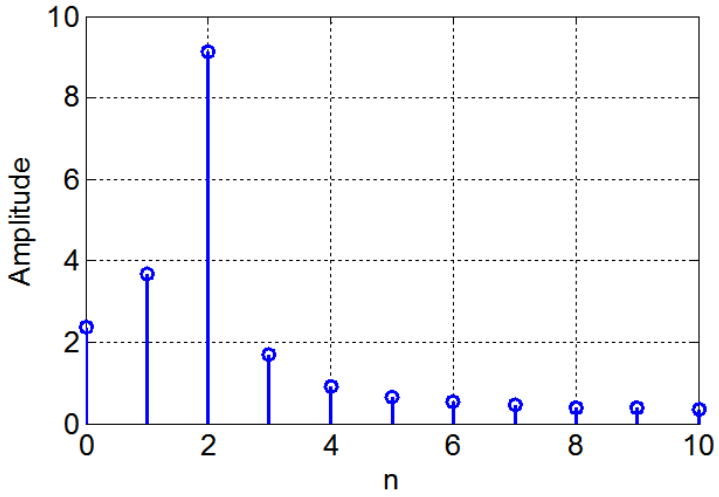
1. In **most cases** it is always possible to transform a non-sparse signal into a sparse signal using a certain transform (base)
2. Finding this base transform is the real challenge
3. A signal **x** is called sparse (in a base $\Phi$), if **Dx** is sparse, where

$$\mathbf{D_x} = \Phi\mathbf{x}$$

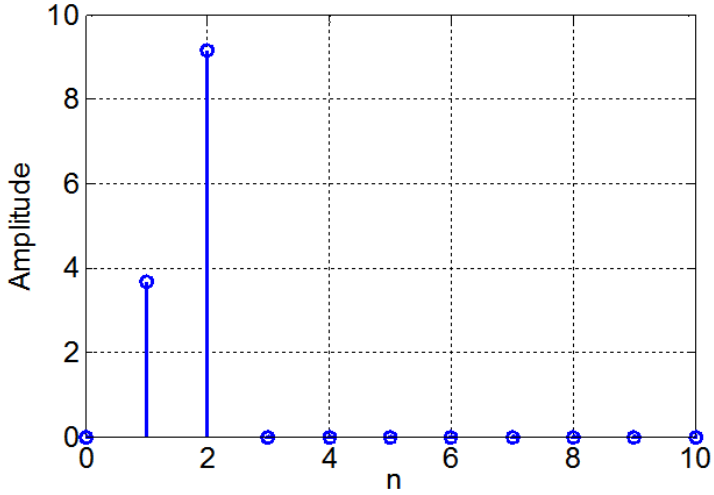
4. Some popular transforms: **Fourier**, **DCT**, etc.

# Sparse signal : Some concepts

1. Also, most of real life signal falls into **compressible** rather than **sparse signal**
2. Compressible signal is amplitude decaying signal with view spikes
3. By thresholding, compressible signal can be converted to sparse signal

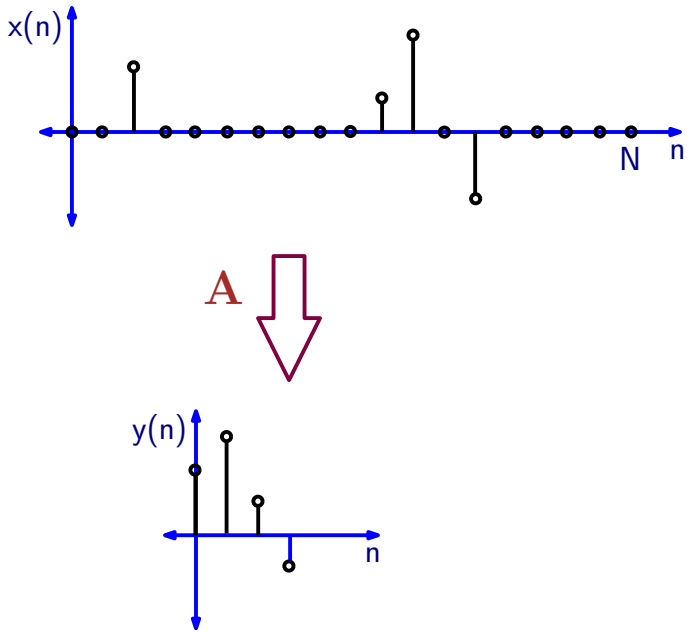



Compressible signal




After thresholding ($v_{thresh} = 3$)

# CS Compression




1. CS compression is performed by linear (matrix) multiplication

$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

2. $x$ is sparse signal or vector of length $N$
3. $y$ is compressed signal of length $M$ ($M << N$)
4. **A** a compression matrix **or** measurement matrix **or** sensing matrix $m \times n$

# CS Compression : Example

1. let $\mathbf{x} = [3 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 0]^T$
2. let $\mathbf{A} =$
$$\begin{bmatrix} -0.0604 & 0.7160 & 0.3883 & 0.7957 & 0.2403 & -0.6501 \\ 0.7252 & 0.3393 & 0.8827 & 0.5590 & -0.6438 & -0.6058 \\ 0.6859 & -0.6101 & 0.2647 & -0.2333 & 0.7265 & -0.4588 \end{bmatrix}$$
3. then: $\mathbf{y} = \mathbf{A}\mathbf{x} = \begin{bmatrix} 0.207 & 3.058 & 2.322 \end{bmatrix}^T$
4. In this example, the length of $\mathbf{x}$ is 6 with sparsity of 2
5. Compression matrix $\mathbf{A}$ is normalized Gaussian random distributed with dimension $3 \times 6$
6. $\mathbf{y}$ has length 3, while $\mathbf{x}$ has 6 (compression ratio 1: 2).

# CS Compression : Example

1. let $\mathbf{x} = [3 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 0]^T$
2. let $\mathbf{A} =$
$$\begin{bmatrix} -0.0604 & 0.7160 & 0.3883 & 0.7957 & 0.2403 & -0.6501 \\ 0.7252 & 0.3393 & 0.8827 & 0.5590 & -0.6438 & -0.6058 \\ 0.6859 & -0.6101 & 0.2647 & -0.2333 & 0.7265 & -0.4588 \end{bmatrix}$$
3. then: $\mathbf{y} = \mathbf{A}\mathbf{x} = \begin{bmatrix} 0.207 & 3.058 & 2.322 \end{bmatrix}^T$
4. In this example, the length of $\mathbf{x}$ is 6 with sparsity of 2
5. Compression matrix $\mathbf{A}$ is normalized Gaussian random distributed with dimension $3 \times 6$
6. $\mathbf{y}$ has length 3, while $\mathbf{x}$ has 6 (compression ratio 1: 2).

# CS Compression : Example

1. let $\mathbf{x} = [3 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 0]^T$
2. let $\mathbf{A} =$
$$\begin{bmatrix} -0.0604 & 0.7160 & 0.3883 & 0.7957 & 0.2403 & -0.6501 \\ 0.7252 & 0.3393 & 0.8827 & 0.5590 & -0.6438 & -0.6058 \\ 0.6859 & -0.6101 & 0.2647 & -0.2333 & 0.7265 & -0.4588 \end{bmatrix}$$
3. then: $\mathbf{y} = \mathbf{A}\mathbf{x} = \begin{bmatrix} 0.207 & 3.058 & 2.322 \end{bmatrix}^T$
4. In this example, the length of $\mathbf{x}$ is 6 with sparsity of 2
5. Compression matrix $\mathbf{A}$ is normalized Gaussian random distributed with dimension $3 \times 6$
6. $\mathbf{y}$ has length 3, while $\mathbf{x}$ has 6 (compression ratio 1: 2).

# Advantage of CS over classical compression

1. Classical compression has two steps: acquisition and then compression separately.

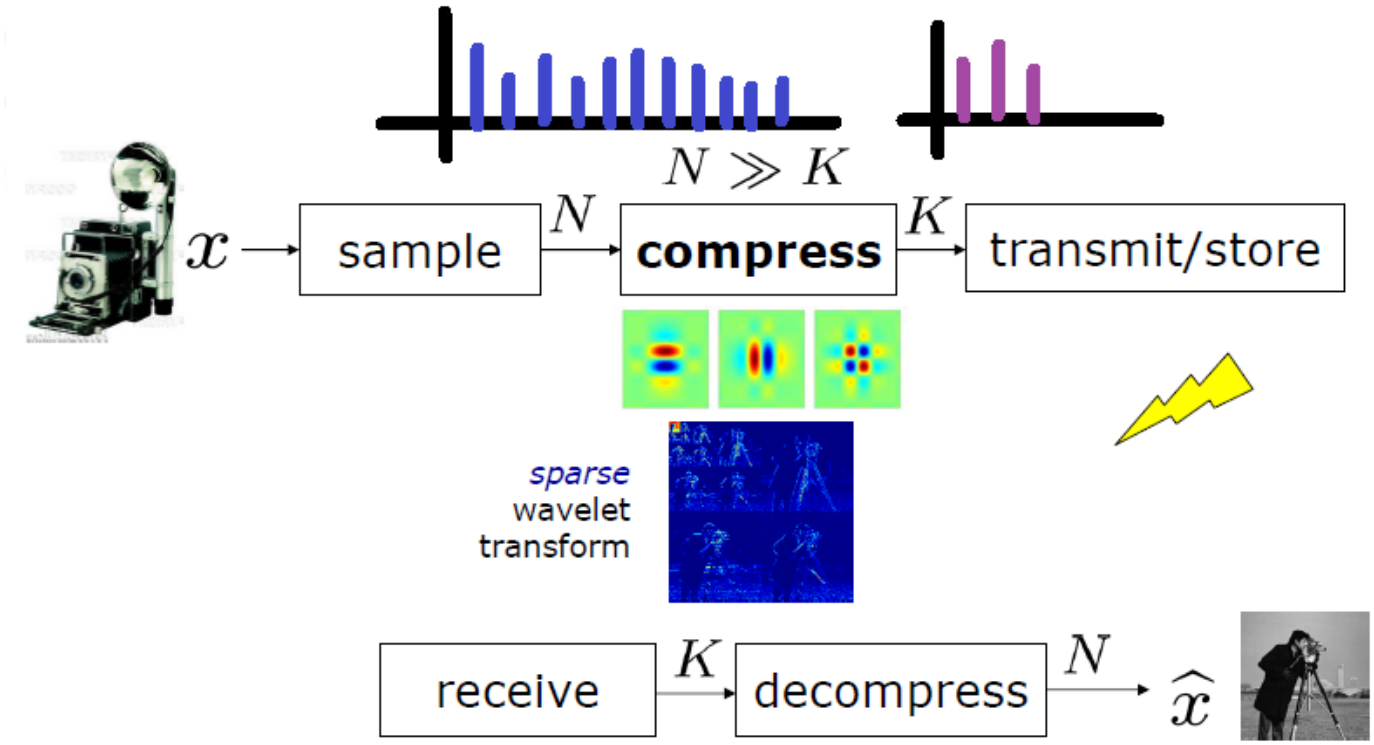



**Figure :** Classical sampling and compression [Source: Baraniuk]

# Advantage of CS over classical compression

1. CS combine acquisition and compression in single step.

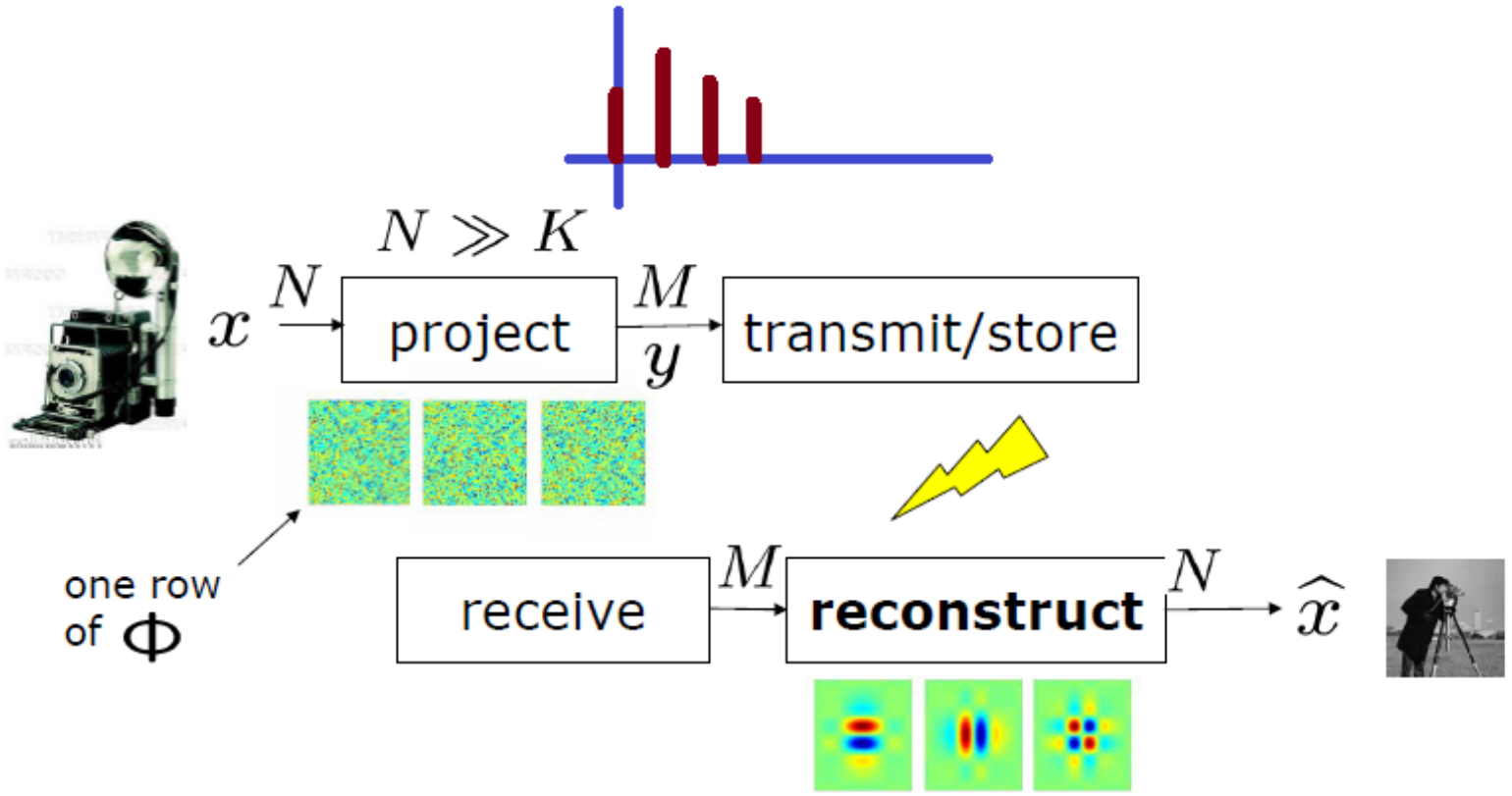



**Figure :** Single step CS [Source: Baraniuk]

# Practical application of CS

1. Bandwidth limited system (consider for example a large number of sensors in **Wireless Sensor Network** (WSN)).
2. Medical application (MRI example)

figure should be added here...

# CS Reconstruction

1. Reconstruction is to obtain original signal **x** from compressed signal **y** and compression matrix **A**

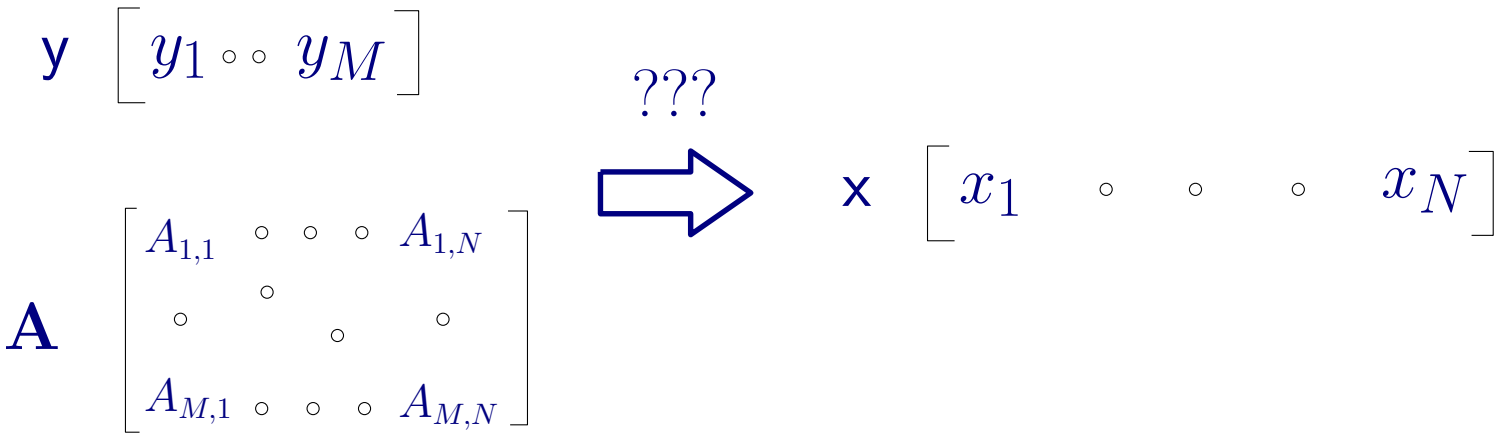



**Figure :** CS reconstruction problem

2. given **y** and **A**, there are infinite possible solution of **x**

# CS Reconstruction

## Ilustration

1. Given $\mathbf{x} = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$
2. $\mathbf{A} = \begin{bmatrix} 0,4 & 0,6 \end{bmatrix}$
3. $\mathbf{y} = \mathbf{Ax} = \mathbf{1,2}$

For reconstruction,

1. given **y** and **A**
2. **x** should be found
3. let $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
4. **Ax** = **y** gives: $\mathbf{x} = \begin{bmatrix} 0,4 & 0,6 \end{bmatrix} \times \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 1,2$
5. $0,4x_1 + 0,6x_2 = 1,2$ or $\mathbf{x} = \begin{bmatrix} t \\ 2 - \frac{2}{3}t \end{bmatrix} = \begin{bmatrix} 0 \\ 2 \end{bmatrix} + \begin{bmatrix} 1 \\ -\frac{2}{3} \end{bmatrix} t$

# CS Reconstruction

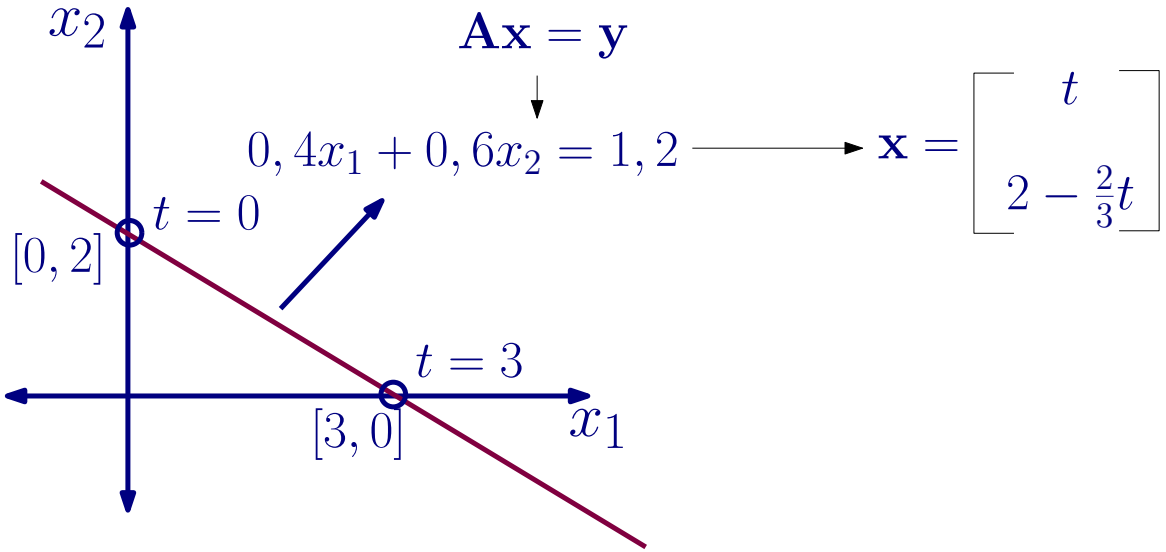



**Figure :** Reconstruction: infinite solution

1. in this example, two sparse solutions: [0 , 2] and [3 , 0].
2. different **A** will produce different line profile, but there is only one correct solution.
3. The compression matrix should 'guide' to correct solution, which is [0 , 2].

# Selection of A

The theory of *A* has been discussed by Donoho, Baraniuk, and Candes[1].
Constraints for measurement matrix **A** can be summarized:

1. Has dimension of $M \times N$ with

$$M \geq c \times k \times \log N$$

here: $c$ is a constant, $k$ is sparsity of the signal, $N$ is the length of the signal

2. Fulfilled restricted isometric property (RIP)

$$(1 - \delta) \|\mathbf{x}\|_2 \leq \|\mathbf{Ax}\|_2 \leq (1 + \delta) \|\mathbf{x}\|_2$$

$\delta$ is a small positive number.

[1] a b c

# Selection of A

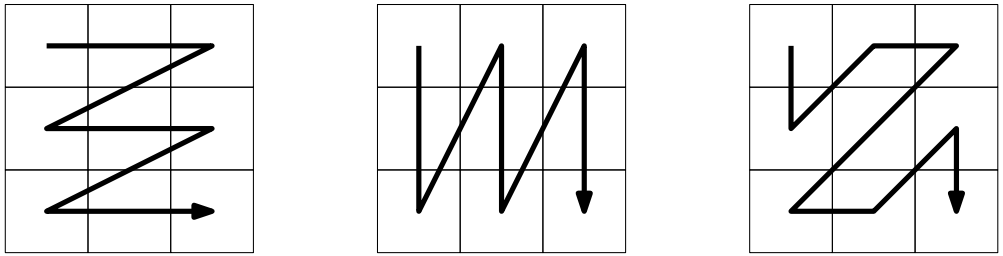
Example:

1. Let $\mathbf{x} = \begin{bmatrix} 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
2. Here: $k = 2$; $N = 10$
3. So: $M = c \times 2 \times \log 10 = 2c$
4. if we take $c = 2$, then $M = 4$
5. Hence, $\mathbf{A}$ is a $4 \times 10$ matrixs
6. There no exact method to determine $c$
7. in practical, $c$ can be taken in [1-2].
8. To fulfill RIP, a random matrix $\mathbf{A}$ can be generate, then its column can be normalized to unity.[2]

[2] A detailed example of designing sensing matrix can be found in for example in: *Designing sparse sensing matrix for compressive sensing to reconstruct high resolution medical images*, Tiwari et al. 2015

# Application of CS for image

1. It is often necessary to convert 2D image to 1D signal
2. Scanning technique is necessary for this conversion



(a) (b) (c)

**Figure :** Some scanning techniques

3. Matlab function such as `reshape` can be used:

$$\mathbf{x} = \begin{bmatrix} 2 & 3 & 4 \\ 1 & 1 & 2 \\ 0 & 2 & 5 \end{bmatrix} \rightarrowtail \mathbf{x_{1D}} = \begin{bmatrix} 2 & 1 & 0 & 3 & 1 & 2 & 4 & 2 & 5 \end{bmatrix}^T$$

4. Compression can be applied to new 1D signal $\mathbf{x_{1D}}$

# Compression in a certain basis

1. Let $\mathbf{x}$ of length $N$ is sparse in $\Psi$, that is

$$\mathbf{b} = \Psi\mathbf{x}$$

or

$$\mathbf{x} = \Psi^{-1}\mathbf{b}$$

where $\mathbf{b}$ is a sparse signal

2. We compress $\mathbf{x}$, using $\mathbf{A}$, that is

$$\mathbf{y} = \mathbf{A}\mathbf{x} = \left(\mathbf{A}\boldsymbol{\Psi}^{-1}\right)\mathbf{b}$$

3. Now, instead of viewing $\mathbf{y}$ as a compression of $\mathbf{x}$ using $\mathbf{A}$,
4. we view $\mathbf{y}$ as a compression of $\mathbf{b}$ using $\mathbf{A}\Psi^{-1}$
5. Therefore, we reconstruct $\mathbf{b}$ from $\mathbf{y}$ and $\mathbf{A}\boldsymbol{\Psi}^{-1}$
6. Then calculate $\mathbf{x} = \Psi^{-1}\mathbf{b}$

# Some examples of CS applications

The following are examples of CS applications on various field: [3]

1. Audio compression
2. Image compression
3. Missing data reconstruction
4. Direction of Arrival Estimation
5. Watermarking
6. .....

[3] This section illustrates some of CS applications in various fields. Detailed discussion is given in Part III of this course.

Continue to Part II : Reconstruction Theory

# SUMMER SCHOOL 2019

## PART II : RECONSTRUCTION THEORY

Prof. Dr. Andriyan B. Suksmono
Dr. Koredianto Usman

**August 13, 2019**

Institut Teknologi Bandung and Telkom University

# Objectives

1. Challenges in CS reconstruction
2. Introduce some methods for CS reconstruction
   - Linear Programming
   - Convex Optimization
   - Greedy Algorithms
   - minimum Total Variant

Notes:

- There are a lot of reconstruction algorithms
- Algorithms given here are just some examples
- Tutorial approach is used in this section
- Matlab and CVX are necessary to try examples and exercises.

# Norm of a vector

Before we start with reconstruction problem, let us review the norm of a vector.

1. Let $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_N \end{bmatrix}^T$
2. The $p$-th order norm of $\mathbf{x}$ ($p \geq 0$) is

$$\|\mathbf{x}\|_p = \sqrt[p]{x_1^p + p_2^p + \cdots + x_N^p}$$

3. 0-th order norm ($L_0$): $\|\mathbf{x}\|_0 = k =$ number of non-zeros elements in $\mathbf{x}$.
4. 1-st order norm ($L_1$): $\|\mathbf{x}\|_1 = |x_1| + |x_2| + \cdots + |x_N|$
5. 2-st order norm ($L_2$): $\|\mathbf{x}\|_1 = \sqrt{x_1^2 + x_2^2 + \cdots + x_N^2}$
6. etc...
7. $p$ is any non-negative real number $(0, \frac{1}{2}, \ldots)$

# Norm of a vector

Example

1. Let

$$\begin{cases} \mathbf{x_1} = \begin{bmatrix} 1 & 2 & 0 & 3 & -1 \end{bmatrix} \\ \mathbf{x_2} = \begin{bmatrix} 0 & 2 & 0 & 0 & -3 \end{bmatrix} \end{cases} \tag{1}$$

2. $\|\mathbf{x_1}\|_0 = 4$
3. $\|\mathbf{x_2}\|_0 = 2$
4. $\|\mathbf{x_1}\|_1 = 7$
5. $\|\mathbf{x_2}\|_1 = \cdots$
6. $\|\mathbf{x_2}\|_2 = \cdots$
7. $\|\mathbf{x_2}\|_2 = \cdots$

# Norm of a vector

Two dimension of **x**:

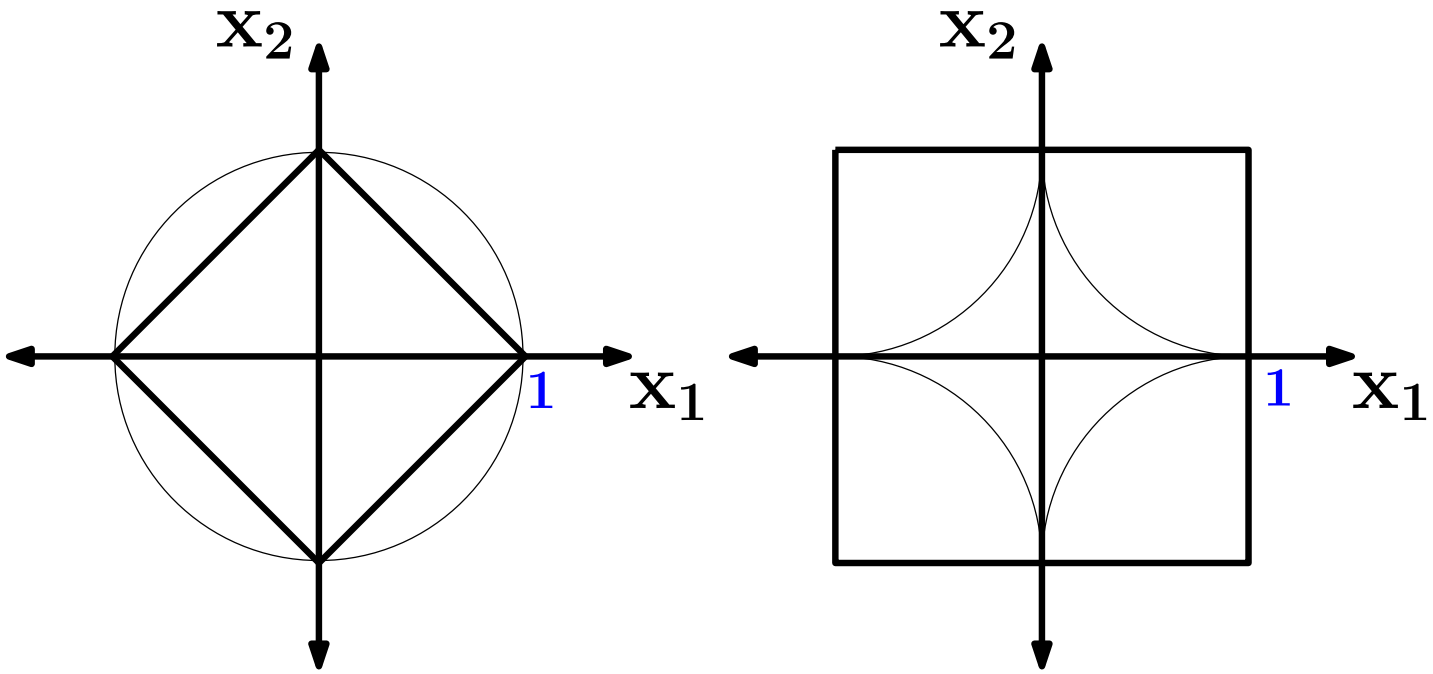



**Figure :** Illustration of norm for vector of length 2

# Reconstruction Problem

1. In CS compression:
$$\mathbf{y} = \mathbf{A}\mathbf{x}$$
2. as $\mathbf{A}$ is $M \times N$, where $M << N$, then solving $\mathbf{x}$ from $\mathbf{y} = \mathbf{A}\mathbf{x}$ is under-determined problem with infinite possible solution
3. A correct solution $\mathbf{x_{sol}}$ can be obtained from the fact that $\mathbf{x}$ is a sparse signal.
4. Thus, the actual solution is taken from the infinite possible solution that has the sparsest property
5. Sparsest solution can be described by minimum $L_0$-norm

# Reconstruction Problem

Example:

1. Let

$$\begin{cases} \mathbf{x_1} = \begin{bmatrix} 1 & 2 & 0 & 3 & -1 \end{bmatrix} \\ \mathbf{x_2} = \begin{bmatrix} 0 & 2 & 0 & 0 & -3 \end{bmatrix} \\ \mathbf{x_3} = \begin{bmatrix} 1 & 0 & 0 & 4 & -4 \end{bmatrix} \\ \mathbf{x_4} = \begin{bmatrix} 2 & 2 & 0 & 4 & -1 \end{bmatrix} \end{cases} \tag{2}$$

2. be the solutions of $\mathbf{Ax} = \mathbf{y}$
3. the sparsest solution for this particular example $\mathbf{x_2}$, $k = 2$

# Reconstruction Problem

Let examine a simple example:

1. Let
$$\mathbf{x} = \begin{bmatrix} 2 & 0 & 0 \end{bmatrix}^T$$
2. Let $\mathbf{A} = \begin{bmatrix} 0.8 & 0.7 & -0.5 \\ 0.6 & 0.7 & 0.87 \end{bmatrix}$
3. $\mathbf{y} = \mathbf{Ax} = \cdots$

# Reconstruction Problem

1. Now, given $\mathbf{y} = \begin{bmatrix} 1.6 & 1.2 \end{bmatrix}^T$ and $\mathbf{A} = \begin{bmatrix} 0.8 & 0.7 & -0.5 \\ 0.6 & 0.7 & 0.87 \end{bmatrix}$
2. Find the original $\mathbf{x}$ !

# Reconstruction Problem

1. As we know that $\mathbf{x}$ is a three dimensional vector, let $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T$
2. As $\mathbf{Ax} = \mathbf{y}$
3. then $\begin{bmatrix} 0.8 & 0.7 & -0.5 \\ 0.6 & 0.7 & 0.87 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1.6 \\ 1.2 \end{bmatrix}$, or
4. $\begin{bmatrix} 0.8x_1 + 0.7x_2 - 0.5x_3 \\ 0.6x_1 + 0.7x_2 + 0.87x_3 \end{bmatrix} = \begin{bmatrix} 1.6 \\ 1.2 \end{bmatrix}$ .... (*)
5. There are infinite solution of (*)
6. But, as $\mathbf{x}$ is sparse, then the sparsest solution of (*) can be selected as the estimated solution

# Reconstruction Problem

CS reconstruction problem can be formulated as:

$$\min \|\mathbf{x}\|_0 \quad \text{subject to} \quad \mathbf{Ax} = \mathbf{y}$$

A more practical formulation is to use L1-Norm (also called: Basis Pursuit (BP)):

$$\min \|\mathbf{x}\|_1 \quad \text{subject to} \quad \mathbf{Ax} = \mathbf{y}$$

We can solve BP using LP, Convex Optimization, etc.

# Reconstruction Problem

CS reconstruction problem can be formulated as:

$$\min \|\mathbf{x}\|_0 \quad \text{subject to} \quad \mathbf{Ax} = \mathbf{y}$$

A more practical formulation is to use L1-Norm (also called: Basis Pursuit (BP)):

$$\min \|\mathbf{x}\|_1 \quad \text{subject to} \quad \mathbf{Ax} = \mathbf{y}$$

We can solve BP using LP, Convex Optimization, ....

# Linear Programming

1. LP is used to solve linear optimization problem [1]
2. min *objective function* *subject to* *constraint function*
3. example: minimize $x_1 - 2x_2$
   subject to
   $x_1 + x_2 \leq 10$
   $-3x_1 + 2x_2 \leq 15$
   $x_1 \geq 0$ and $x_2 \geq 0$
4. We write the objective function as :
   $f(\mathbf{x}) = x_1 - 2x_2 = \begin{bmatrix} 1 & -2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \mathbf{c}^T\mathbf{x}$, where $\mathbf{c} = \begin{bmatrix} 1 & -2 \end{bmatrix}^T$
   and $\mathbf{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$.

[1] Historically, Dantzig is considered to be the first one to introduce LP problem, and he proposed the *simplex algorithm* to solve LP in 1948, one of the most elegant algorithm in 20th century. A breakthrough method called *Interior Point Method* proposed by N. Karmarkar in 1984 to solve LP more efficiently

# Linear Programming

1. Constraint function $g(\mathbf{x}) : x_1 + x_2 \leq 10$ and $-3x_1 + 2x_2 \leq 15$, can be written as:
   $g(\mathbf{x}) = \begin{bmatrix} 1 & 1 \\ -3 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \begin{bmatrix} 10 \\ 15 \end{bmatrix} \leq \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ or $g(\mathbf{x}) = \mathbf{Ax} - \mathbf{b} \leq \mathbf{0}$
   where $\mathbf{A} = \begin{bmatrix} 1 & 1 \\ -3 & 2 \end{bmatrix}$, and $\mathbf{b} = \begin{bmatrix} 10 \\ 15 \end{bmatrix}$, and $\mathbf{x} = [x_1 \;\; x_2]^T$
2. Finally, $x_1 > 0; x_2 > 0$ is called as lower bound (LB) condition
   $\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} > \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \mathbf{lb}$, and $\mathbf{lb} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$
3. Using $\mathbf{c}$, $\mathbf{A}$, $\mathbf{b}$, and $\mathbf{lb}$ as known parameters, reconstructed $\mathbf{x}$ can be calculated using simplex algorithm in Matlab (`linprog` function)
4. `xEstimated=linprog(c,A,b,Aeq,Beq,lb,ub)`

# Linear Programming

1. `xEstimated=linprog(c,A,b,Aeq,beq,lb,ub)`
2. `Aeq`, and `beq` is matrix that fulfills equality: **Aeq x = beq**
3. Aeq = [] and beq = [] in case no equality in LP
4. `lb` and `ub` is lower and upper bound
5. If `ub` is not available, just ommit from the function
6. next slide show Matlab example

# Linear Programming

**Problem**: minimize $x_1 - 2x_2$
subject to:
$x_1 + x_2 \leq 10$
$-3x_1 + 2x_2 \leq 15$
$x_1 \geq 0$ and $x_2 \geq 0$

Matrices: $\mathbf{c} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$; $\mathbf{A} = \begin{bmatrix} 1 & 1 \\ -3 & 2 \end{bmatrix}$; $\mathbf{c} = \begin{bmatrix} 10 \\ 15 \end{bmatrix}$ and $\mathbf{lb} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

**Matlab code**:

```
c=[1;-2];          % objective function coefficients
A=[1 1; -3 2];     % the LHS constraint matrix
b=[10; 15];        % the RHS constraint matrix
lb=[0 0] ;         % lower bounds of x1 and x2
[xEst fVal]=lisnprog(c,A,b,[],[],lb)
disp(sprintf('x1= %f x2=%f\n',xEst(1),xEst(2)))
```

# Linear Programming

Exercise 1:

Identify **c**, **A**, **b**, and **lb** and using Matlab solve the following LP problem:

**Problem**: minimize $-x_1 - 2x_2 + 3x_3$

subject to:

$x_1 + x_2 + x_3 \leq 10$

$x_1 - 2x_2 + 3x_3 \leq 15$

$x_1 \geq 0$

$x_2 \geq 0$

$x_2 \geq 1$

# Linear Programming dan CS reconstruction

Now we discuss how CS rec. using BP is transformed to LP:

1. Problem statement is to solve

$$\min \|\mathbf{x}\|_1 \quad \text{subject to} \quad \mathbf{Ax} = \mathbf{y}$$

2. $f(\mathbf{x}) = \min \|\mathbf{x}\|_1$ is objective function.
3. $\mathbf{Ax} = \mathbf{y}$ is constraint function
4. Let $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_N \end{bmatrix}^T$
5. Then $f(\mathbf{x}) = \|\mathbf{x}\|_1 = |x_1| + |x_2| + \cdots + |x_N|$
6. Let $|x_1| \leq t_1$, $|x_2| \leq t_2$, $\cdots$, $|x_N| \leq t_N$, then

$$x_i \leq t_i \quad \textit{and} \quad -x_i \leq t_i$$

# Linear Programming

1. Using this relationship of $x_i$ to $t_i$, the optimization problem can be modified from

$$\min \|\mathbf{x}\|_1 = \min \left(|x_1| + |x_2| + \cdots + |x_N|\right) \quad \textit{subject to} \quad \mathbf{Ax} = \mathbf{y}$$

to

$$\min t_1 + t_2 + \cdots + t_N \quad \textit{subject to} \quad \mathbf{At} \geq \mathbf{y} \quad \textit{and} \quad -\mathbf{At} \geq \mathbf{y}$$

2. The constraint $\mathbf{At} \geq \mathbf{y}$ and $-\mathbf{At} \geq \mathbf{y}$ can be combined into

$$\mathbf{A_p t} \geq \mathbf{y_p}$$

where

$$\mathbf{A_p} = [\mathbf{A} \;\; ; \;\; -\mathbf{A}]$$

$$\mathbf{y_p} = [\mathbf{y} \;\; ; \mathbf{y}]$$

# Linear Programming

1. Now the BP is modified from

$$\min \|\mathbf{x}\|_1 = \min \left(|x_1| + |x_2| + \cdots + |x_N|\right) \quad \textit{subject to} \quad \mathbf{Ax} = \mathbf{y}$$

to

$$\min\ t_1 + t_2 + \cdots + t_N \quad \textit{subject to} \quad \mathbf{A_p t} \geq \mathbf{y_p}$$

with

(3)

2. This LP can be solved using Matlab with:

$$\mathbf{c} = \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^T \text{ N elements}$$

$$\mathbf{A_p} = \begin{bmatrix} A \\ -A \end{bmatrix}$$

$$\mathbf{y_p} = \begin{bmatrix} A \\ -A \end{bmatrix}$$

# Example

Let us try to convert the following BP to LP, and solve it using Matlab.

1. $\min \|\mathbf{x}\|_1$ subject to $\mathbf{Ax}=\mathbf{y}$
   where $\mathbf{A} = \begin{bmatrix} -0.707 & 0.8 & 0 \\ 0.707 & 0.6 & -1 \end{bmatrix}$ and $\mathbf{y} = \begin{bmatrix} 1.65 \\ -0.25 \end{bmatrix}$.
2. As $\mathbf{A}$ is $2 \times 3$, then $\mathbf{x}$ has length of 3 : $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T$
3. The equivalent LP formulation is:

$$\min t_1 + t_2 + t_3 \qquad \text{subject to} \qquad \mathbf{A_p t} = \mathbf{y_p}, \qquad \text{with:}$$

$$\mathbf{A} = \begin{bmatrix} -0.707 & 0.8 & 0 \\ 0.707 & 0.6 & -1 \\ 0.707 & -0.8 & 0 \\ -0.707 & -0.6 & 1 \end{bmatrix}, \mathbf{y_p} = \begin{bmatrix} 1.65 \\ -0.25 \\ 1.65 \\ -0.25 \end{bmatrix}, \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}.$$

# Matlab code

1. We identify that $\mathbf{c} = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^T$,

$$\mathbf{A_{LP}} = \mathbf{A_p} = \begin{bmatrix} -0.707 & 0.8 & 0 \\ 0.707 & 0.6 & -1 \\ 0.707 & -0.8 & 0 \\ -0.707 & -0.6 & 1 \end{bmatrix}, \text{ and } \mathbf{b} = \mathbf{y_p} = \begin{bmatrix} 1.65 \\ -0.25 \\ 1.65 \\ -0.25 \end{bmatrix}$$

2. There is no lower bound **lp** here.

Matlab Code:

```
c=[1;1;1];          % objective function coefficients
A=[-0.707 0.8 0;  0.707 0.6 -1; 0.707 -0.8 0;...
        -0.707 -0.6 1];  % the LHS constraint matrix
b=[1.65; -0.25; 1.65; -0.25];     % the RHS constr. mtx
Aeq = A; beq = b; lb = [0 0 0 0 0 0]
[xEst fVal]=linprog(c,A,b, Aeq, beq, lb)
```

# Exercise

Transform the following BP to LP, determine **c**, **A**, and **b**, and solve it using Matlab:

1. $\min \|\mathbf{x}\|_1$ subject to $\mathbf{Ax}=\mathbf{y}$

   where $\mathbf{A} = \begin{bmatrix} 0.8 & 0.5 & -0.3 \\ -0.6 & -0.87 & -0.95 \end{bmatrix}$ and $\mathbf{y} = \begin{bmatrix} 0.6 \\ 0.9 \end{bmatrix}$.

2. .....

# Convex Optimization

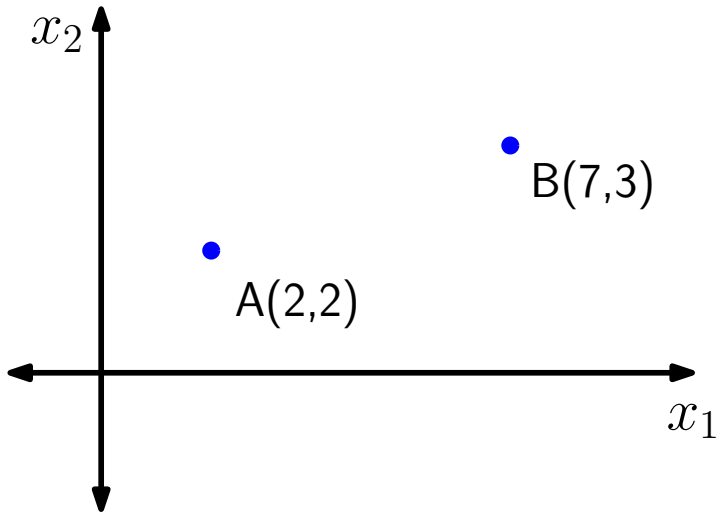
1. Convex optimization is a general case of LP
2. Optimization problem: min $f(\mathbf{x})$ subject to $g(\mathbf{x}) = 0$
3. If $f(\mathbf{x})$ is a convex function, and $g(\mathbf{x}) = 0$ form a convex perimeter/area
4. then this optimization problem can be solved using convex optimization method

# Linear, Affine, and Convex Combination

1. Consider any two points $A$ and $B$
2. a combination pf coordinate of $A$ and $B$:

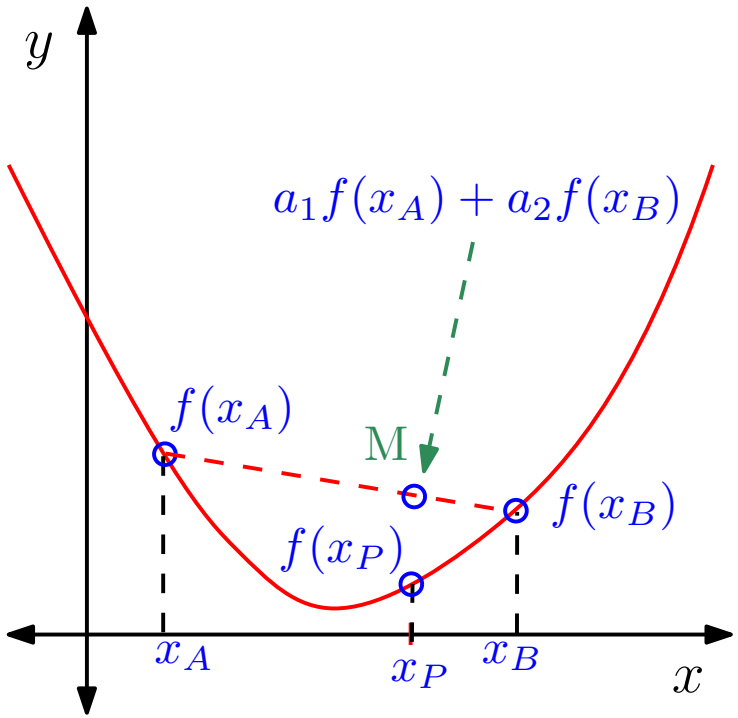
$$\alpha A + \beta B$$

3. is called Linear Combination if $\alpha$ and $\beta$ random real number
4. is called Affine Combination if $\alpha$ and $\beta$ random, but $\alpha + \beta = 1$
5. is called Convex Combination if $\alpha$ and $\beta$ non-negatif real number with $\alpha + \beta = 1$

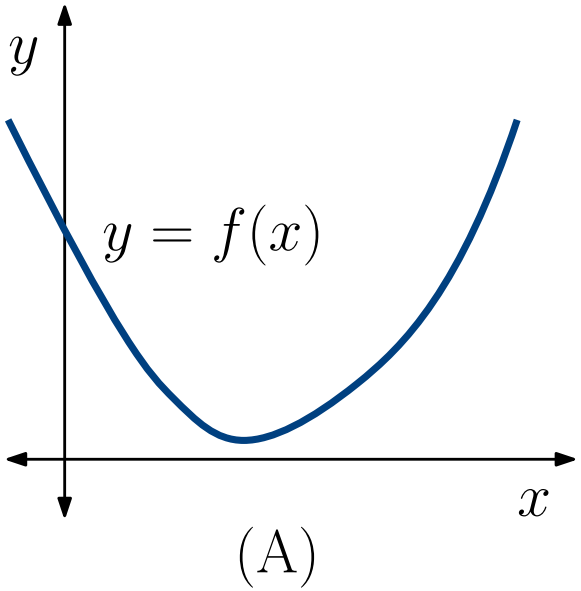


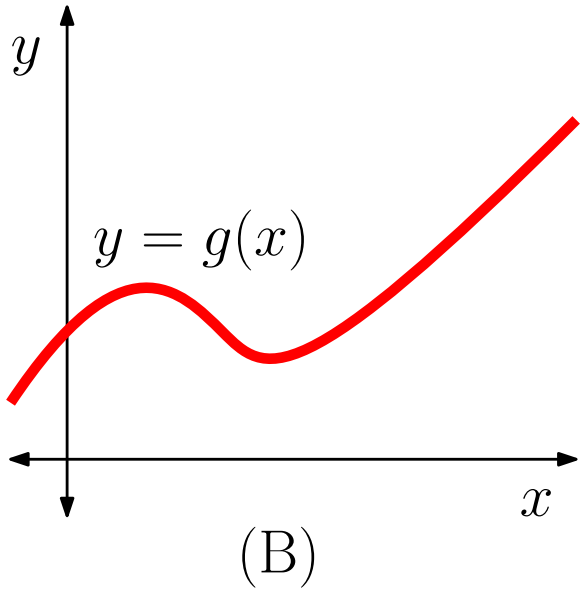
# Convex Function

1. In convex optimization, we often encounter the terminology of *convex function*.
2. Let $f(\mathbf{x})$ be a certain real function
3. Let also $a_1$ and $a_2$ be non-negative real number
4. if $a_1 f(x_A) + a_2 f(x_A) \geq f(a_1 x_A + a_2 x_B)$ is fulfilled for any real values $x_A$ and $x_B$, then $f(\mathbf{x})$ is called convex



# Convex function



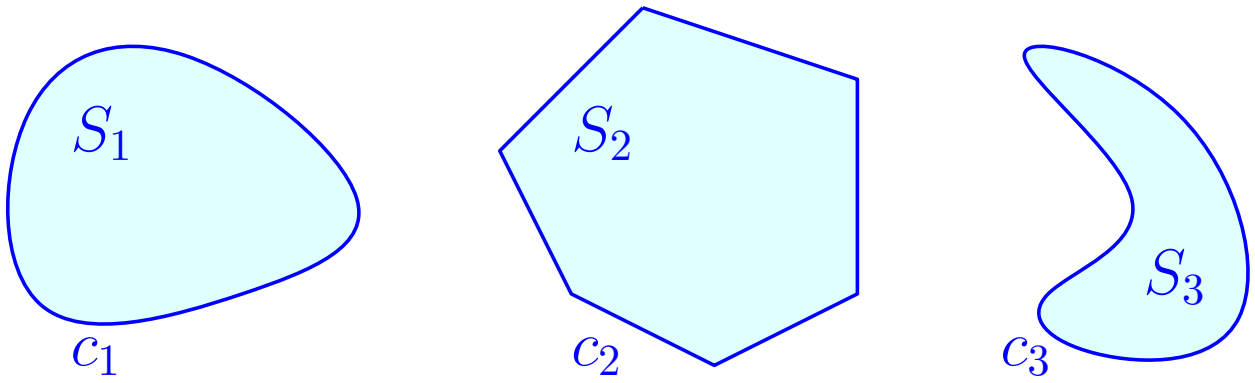




1. Fig. (A) is a convex function
2. Fig. (B) is not
3. For any convex function, a Global minimum in guaranteed
4. Iterative algorithm is possible to find minimum

# Convex Area

An area $S$ is called convex, if for any convex combination of two points in $S$ is also in $S$.

**Mathematically**: $S$ is convex if $A \in S$ and $B \in S$ then point P which is $P = a_1 A + a_2 B$ for $a_1 + a_2 = 1$ and $a_1 \geq 0$ and $a_2 \geq 0$ is also $\in S$.



# Convex Optimization

1. In CS reconst. problem, as objective function is convex and constraint functions forms convex area, then Convex Optimization can be utilized.
2. One of popular method to perform convex optimization is using Interior Point Method (with Newton-Raphson iteration as main step)
3. Stephen Boyd (Stanford) wrote a routine to solve convex optimization problem in Matlab
4. This routine is called CVX [2]
5. The syntax to use CVX is called CVX programming
6. It consist of three components: calling CVX, declaring the variables, declaring the objective function, and finally declaring the constraint function

[2] CVX is available for free at CVX research webpage: http://cvxr.com/cvx

# Convex Optimization

CVX syntax to solve BP:

```
cvx_begin

    variables x complex
    minimize norm(x,1)
    subject to
        A * x = y
cvx_end
```

# Convex Optimization

Steps to install and to run CVX in Matlab:

1. Download the CVX distribution either 32 bit or 64 bit from cvxr.com
2. Extract CVX in a certain folder
3. Run Matlab
4. In Matlab, go to extracted CVX folder
5. Install CVX by running the script `cvx_setup`
6. Test installation by issuing command `cvx_begin` followed by `cvx_end`

# Convex Optimization

```
cvx_begin

    variables x complex
    minimize norm(x,1)
    subject to
        A * x = y
cvx_end
```

# Convex Optimization

1. Solve Let us try to solve the following BP:
2. min $\|\mathbf{x}\|_1$ subject to $\mathbf{Ax}=\mathbf{y}$

   where $\mathbf{A} = \begin{bmatrix} -0.707 & 0.8 & 0 \\ 0.707 & 0.6 & -1 \end{bmatrix}$ and $\mathbf{y} = \begin{bmatrix} 1.65 \\ -0.25 \end{bmatrix}$.

Matlab Code:

```
A = [-0.707 0.8 0;  0.707 0.6 -1];   % mixing matrix
y = s[1.65; -0.25];    % compressed signal
N = size(A, 2)  % length of x
cvx_begin
    varibles xEst(N)
    minimize norm(xEst, 1)
    subject to
          A*xEst == y
cvx_end
```

# Convex Optimization

Exercise 3: solve using Matlab!

1. Let $\mathbf{x} = \begin{bmatrix} 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$ $(N = 10)$
2. generate $\mathbf{A}$ of size $5 \times 10$ with Gaussian Random distribution
3. Normalize each column in $\mathbf{A}$ to unity
4. Calculate $\mathbf{y} = \mathbf{A}\mathbf{y}$
5. From $\mathbf{A}$ and $\mathbf{y}$, using CVX, estimate the original $\mathbf{x}$
6. Try to reduce dimension of $\mathbf{A}$ to $3 \times 10$, can we estimate $\mathbf{x}$?
7. Reduce further dimension of $\mathbf{A}$ to $2 \times 10$, can we still estimate $\mathbf{x}$ ?

# Noisy system

1. There is a case when $\mathbf{x}$ is contaminated by noise (typically AWGN) before compression:

$$\mathbf{x_N} = \mathbf{x} + \mathbf{n}$$

2. The noisy $\mathbf{x_N}$ go through compression:

$$\mathbf{y} = \mathbf{A}\mathbf{x_N}$$

3. As actual noise $\mathbf{n}$ can not be known and can not be estimated, therefore CS reconstruction should be modified from BP : to finding a sparse $\mathbf{x_{est}}$ that $\mathbf{A}\mathbf{x_{est}}$ as closed as possible to the $\mathbf{y}$

$$\min \|\mathbf{x_{est}}\|_1 \quad \text{subject to} \quad \|\mathbf{A}\mathbf{x_{est}} - \mathbf{y}\|_2 \leq \epsilon$$

# Noisy system

1. Reconstruction problem:
   $\min \|\mathbf{x_{est}}\|_1$ subject to $\|\mathbf{A x_{est}} - \mathbf{y}\|_2 \leq \epsilon$
2. is called *second order cone problem - SOCP*
3. SOCP is a convex problem, therefore CVX capable to solve it

Matlab Code:

```
cvx_begin
    varibles xEst(N)
    minimize norm(xEst, 1)
    subject to
        norm( A*xEst-y,2) < epsilon
cvx_end
```

# Exercise 4

1. Let $\mathbf{x} = \begin{bmatrix} 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$ $(N = 10)$
2. Add $\mathbf{x}$ with AWGN with SNR = 20 dB
3. generate $\mathbf{A}$ of size $5 \times 10$ with Gaussian Random distribution
4. Normalize each column in $\mathbf{A}$ to unity
5. Calculate $\mathbf{y} = \mathbf{A}\mathbf{x_N}$
6. From $\mathbf{A}$ and $\mathbf{y}$, using CVX, estimate the original $\mathbf{x_N}$

Continues to next slide: Greedy Algorithm...

# Answer to Exercise 3

```
x = [2 0 2 0 0 0 0 0 0 0]';    % original x
M = 5; N = 10;                 % dimension of A
A=randn(M,N);                  % mixing matrix
for ii=1:N    % normalize each column
   A(:,ii) = A(:,ii)/norm(A(:,ii),2);
end
y=A*x;                         % compressed signal
%Reconstruction using CVX (BP case)
cvx_begin
    variables xEst(N)
    minimize norm(xEst, 1)
    subject to
       A*xEst == y
cvx_end
xEst % display the estimation result
```

# Answer to Exercise 4

```
x = [2 0 2 0 0 0 0 0 0 0]';    % original x
M = 5; N = 10;                 % dimension of A
SNR = 20                        % SNR in dB
xN = awgn(x,SNR);              % Add AWGN to  x
A=randn(M,N);                  % mixing matrix
for ii=1:N    % normalize each column
   A(:,ii) = A(:,ii)/norm(A(:,ii),2);
end
y=A*xN;                        % compressed signal
%Reconstruction using CVX (SOCP case)
epsilon = 10^(-SNR)   % set epsilon
```

Telkom University

# SUMMER SCHOOL 2019

## PART II : RECONSTRUCTION THEORY (Continued)

**Prof. Dr. Andriyan B. Suksmono**
**Dr. Koredianto Usman**

**August 13, 2019**

Institut Teknologi Bandung

# Objectives

1. Introduce some methods for CS reconstruction
   - Linear Programming
   - Convex Optimization
   - Greedy Algorithms
   - minimum Total Variant
2. Reconstruction Examples

Notes:

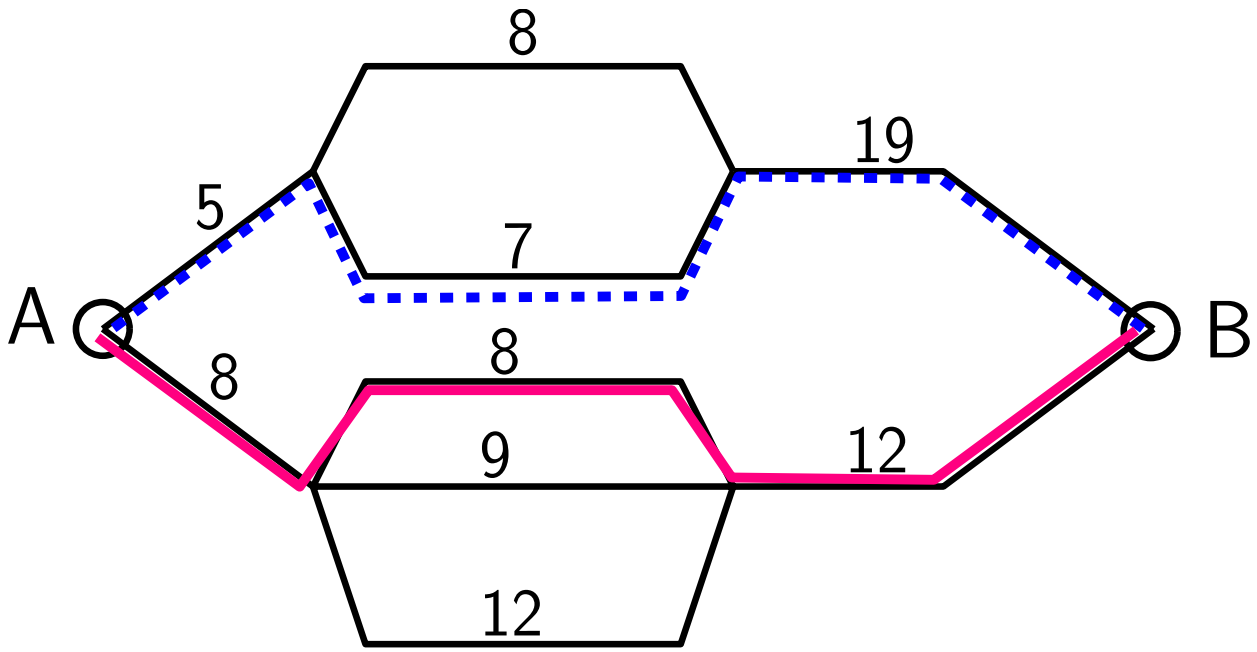
- Tutorial approach is used in this section
- `CVX`, `omp`, and `L1-magic` are necessary to try examples and exercises.

# Greedy Algorithm

1. Greedy algorithm is a heuristic algorithm
2. contains iterative steps
3. local optimum is selected in each step
4. global optimum is expected at the end of iteration
5. usually faster than analytic methods
6. from greedy point of view, selected solution is optimal
7. but, correct solution is not guaranteed

# Greedy Algorithm

Finding shortest path:



# Greedy Algorithm

1. There are many greedy method in CS reconstruction
2. The famous one is the Orthogonal Matching Pursuit (OMP)
3. OMP has many derivation: StOMP, ROMP, etc
4. Matching Pursuit (MP) is considered as basic mechanism of OMP
5. MP is introduced by Mallat and Zhang in 1993
6. OMP is introduced by Chen et al in 1989 (independently from MP)
7. There are a lot of CS reconstruction using OMP (because of speed)

# Matching Pursuit

1. Introduced by Mallat and Zhang, in their research on representing signal in time and frequency domain.
2. MP works as follows:
3. $\mathbf{y} = \mathbf{A}\mathbf{x}$: $\mathbf{y}$ is viewed as a linear combination of each column of $\mathbf{A}$ using coefficients of $\mathbf{x}$
4. Consider a $2 \times 3$ matrix $\mathbf{A}$
5. 

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$= \begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix} x_1 + \begin{bmatrix} a_{12} \\ a_{22} \end{bmatrix} x_2 + \begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix} x_3$$

# Matching Pursuit

1. Consider a more actual example:
2. MP works as follows:
3. Let $\mathbf{x} = \begin{bmatrix} -1.2 & 1 & 0 \end{bmatrix}^T$
4. $\mathbf{A} = \begin{bmatrix} -0.707 & 0.8 & 0 \\ 0.707 & 0.6 & -1 \end{bmatrix}$
5. $\mathbf{y} = \mathbf{A}\mathbf{x} = \begin{bmatrix} -0.707 \\ 0.707 \end{bmatrix}(-1.2) + \begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix}(1) + \begin{bmatrix} 0 \\ -1 \end{bmatrix}(0) = \begin{bmatrix} -1.65 \\ -0.25 \end{bmatrix}$
6. Here $\mathbf{y} = \begin{bmatrix} -1.65 \\ -0.25 \end{bmatrix}$ is contributed by each column of $\mathbf{A}$
7. Now, in reconstruction, given $\mathbf{y}$ and $\mathbf{A}$
8. MP works reversely by finding each column starting from largest contribution

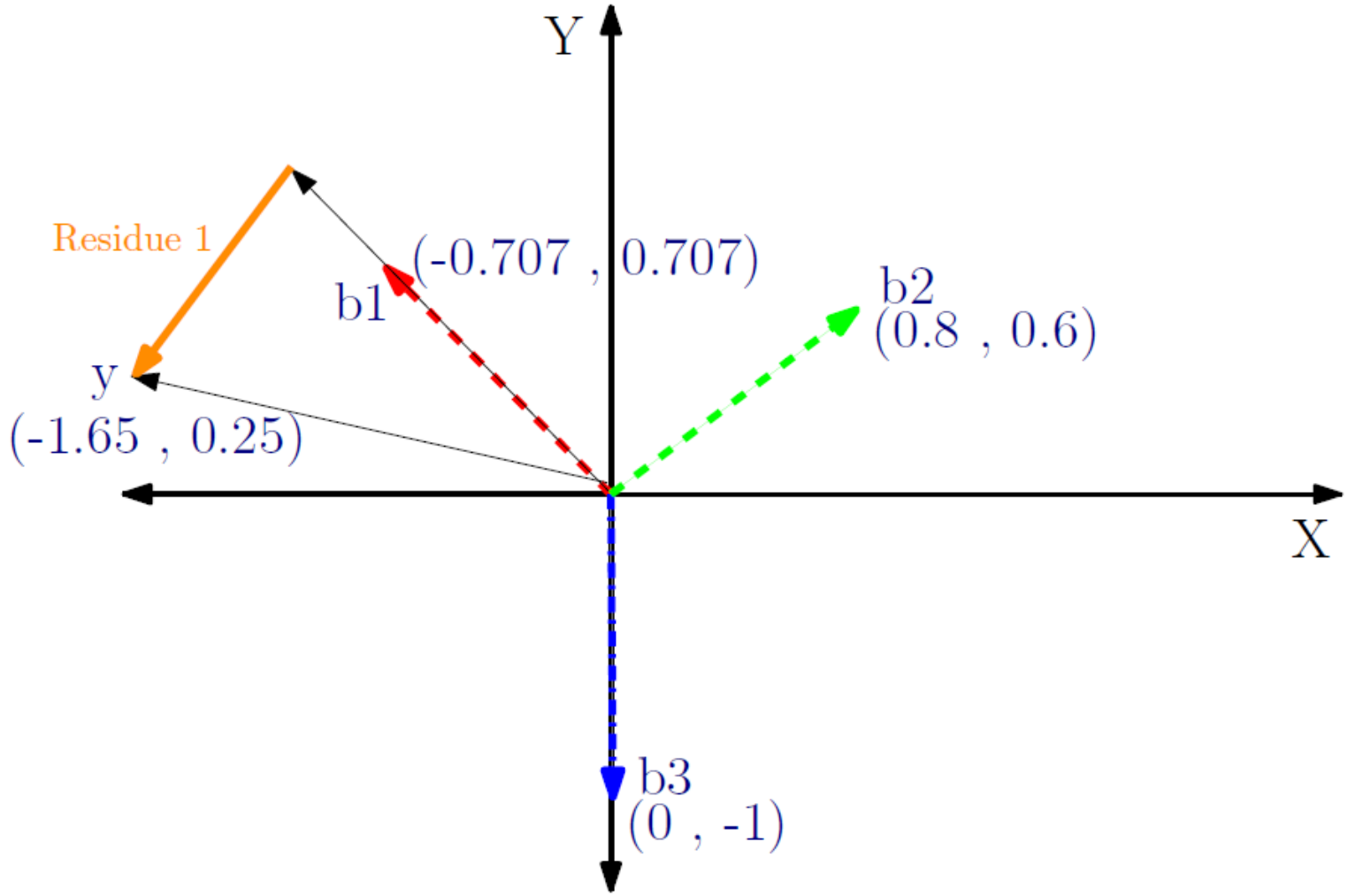
# Matching Pursuit

1. Contribution is calculated by absolute inner product of $\mathbf{y}$ and each column of $\mathbf{A}$
2. $\mathbf{g} = \mathbf{A}^T\mathbf{y} = \begin{bmatrix} -1.34 \\ 1.17 \\ 0.25 \end{bmatrix}$
3. maksimum component $\lambda_i = max(\mathbf{g})$
4. maksimum position : $idx = argmax\mathbf{g}$
5. As the first component is maximum, and taken as the highest contribution
6. After taking first component, the residue is calculated

$$\mathbf{r} = \mathbf{y} - \lambda_i \mathbf{A}(:, )$$

7. the process is repeated from $\mathbf{r}$

# Residue



# Matching Pursuit

1. Using MP, the reconstructed signal : $\mathbf{x_{est}} = \begin{bmatrix} -1.34 \\ 1 \\ 0 \end{bmatrix}$
2. The actual solution is actually $\mathbf{x} = \begin{bmatrix} -1.2 \\ 1 \\ 0 \end{bmatrix}$
3. To improve accuracy of MP, an orthogonalization step is necessary
4. The method of MP is perfected by OMP, by adding this orthogonalization step.
5. Interestingly, OMP is developed rather independently by Chen et al. in 1989.

# Orthogonal Matching Pursuit - OMP

1. OMP has similar step to MP
2. OMP adds orthogonalization step, to ensure that previous coefficients does not counted for the next iteration.
3. OMP is fast and quite accurate
4. OMP can be used for real or complex valued signals.
5. OMP is modified by many researchers for even a better algorithm such as: CoSAMP(Needell and Tropp, 2008), ROMP (Needell and Vershinin, 2010), StOMP (Donoho et al. 2012), etc.
6. Application of OMP for CS is pioneered by Tropp and Gilbert in 2007.

# Orthogonal Matching Pursuit - OMP

**OMP algorithm**

*Input :* **A** a $M \times N$ matrix, and **y** a vector of length $M$

*Output :* **x**, a vector of length **N**.

1. initiate residue vector $\mathbf{r_0} \longleftarrow \mathbf{y}$, normalize reconstruction vector $\mathbf{B} = \mathbf{A}$, collecting basis matrix $\mathbf{A_{new}} \longleftarrow \mathbf{0}$ counter $i \longleftarrow 1$, and temporary reconstruction matrix $\mathbf{x_{rec}} \longleftarrow \mathbf{0}$
2. normalize the length of each vector in **B**: $\mathbf{B}(:,j) \longleftarrow \frac{\mathbf{B}(:,j)}{\|\mathbf{B}(:,j)\|_2}$ for $j$ from 1 to $N$.
3. calculate the contribution vector $\mathbf{w} = \mathbf{B}^T \cdot \mathbf{r_i}$
4. determine the index of maximum value in **w**. I.e. the location of maximum $\gamma = arg(max(\mathbf{w}))$

# OMP Algorithm (continued)

5. assign $\mathbf{A_{new}}$ according to $\gamma$
   $\mathbf{A_{new}}(:, \gamma) \longleftarrow \mathbf{A}(:, \gamma)$
6. Solve the Least Square Problem
   $\min \|\mathbf{A_{new}} \cdot \mathbf{L_p} - \mathbf{y}\|_2$
   for $\mathbf{L_p}$
7. calculate Residue
   $\mathbf{r_i} = \mathbf{y} - \mathbf{A_{new}} \cdot \mathbf{L_p}$
8. assign $\mathbf{x_{rec}} = \mathbf{L_p}$
9. remove the selected basis from $\mathbf{B}$
   $\mathbf{B} \longleftarrow \mathbf{B} \backslash \mathbf{B}(:, \gamma)$
10. increase the counter $i \longleftarrow i + 1$
11. if $i \leq M + 1$ then repeat from step 3, else assign $\mathbf{x} \longleftarrow \mathbf{x_{rec}}$ and stop.

# OMP Algorithm

1. Many researchers wrote OMP algorithm in Matlab
2. For example: M Shaban (2018), S.K. Sahoo (2018), Becker (2016), M. Tummalacherla (2017), Seung Hwan Yoo (2016) etc.
3. The implementation is often shared in online community, such as Matlab Exchange, GitHub, etc.
4. In this lecture, we will use S.H. Yoo implementation (availabel in GitHub : https://github.com/seunghwanyoo/omp).
5. The syntax :

```
xEst = omp(A, y, k)
```

# OMP

1. Solve Let us try to solve the following problem:
2. Given $\mathbf{A} = \begin{bmatrix} -0.707 & 0.8 & 0 \\ 0.707 & 0.6 & -1 \end{bmatrix}$ and $\mathbf{y} = \begin{bmatrix} 1.65 \\ -0.25 \end{bmatrix}$.
3. Find $\mathbf{x}_{\mathbf{est}}$ using OMP!

**Ans: Matlab Code:**

```
A=[-0.707 0.8 0;  0.707 0.6 -1];   % mixing matrix
y=[1.65  -0.25];          % compressed signal
k = 2;                      % sparsity of x
tic                            %start measure recons time
xest = omp(A,y,k)          %estimate x using OMP
elapse = toc                % stop measuring time
```

# Complex valued using OMP

OMP works also for complex valued signal:

1. Let $\mathbf{x} = \begin{bmatrix} 1+i & 0 & i & 0 & 0 & 0 & 0 \end{bmatrix}$
2. Generate random matrix measurement matrix $\mathbf{A}$ of size $5 \times 8$ (real of complex)
3. calculate $\mathbf{y} = \mathbf{Ax}$
4. from $\mathbf{y}$ and $\mathbf{y}$, estimate $\mathbf{x}_{\mathbf{est}}$ using OMP!

**Ans: Matlab Code:**

```
x=[1+i 0 i 0 0 0 0].';   % mixing matrix
A=[-0.707 0.8 0;  0.707 0.6 -1];   % mixing matrix
y=A*x;                             % compressed signal
k = 2;                             % sparsity of x
tic                                % start measure recons time
xest = omp(A,y,k)                  % estimate x using OMP
elapse = toc                       % stop measuring time
```

# Exercise 5

Repeat again Exercise 3, but instead of CVX, solve it using OMP!

1. Let $\mathbf{x} = \begin{bmatrix} 2 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$ $(N = 10)$
2. generate $\mathbf{A}$ of size $5 \times 10$ with Gaussian Random distribution
3. Normalize each column in $\mathbf{A}$ to unity
4. Calculate $\mathbf{y} = \mathbf{A}\mathbf{x}$
5. From $\mathbf{A}$ and $\mathbf{y}$, using CVX, estimate the original $\mathbf{x}$
6. Try to reduce dimension of $\mathbf{A}$ to $3 \times 10$, can we estimate $\mathbf{x}$?
7. Reduce further dimension of $\mathbf{A}$ to $2 \times 10$, can we still estimate $\mathbf{x}$ ?

# Minimum Total Variance

1. Total Variance is a parameter in data analysis
2. Its the summation of data variance in the collected data
3. At first, it does not related to CS
4. But its technique, currently are used for CS reconstruction
5. Especially in the case of image compression, where image data is typically low frequency signal (*small variance*)
6. Let $\mathbf{x}$ is the original signal
7. $\mathbf{A}$ is the compression matrix
8. $\mathbf{y} = \mathbf{Ax}$
9. Reconstruction problem:

$$\min(TV(\mathbf{x})) \text{ subject to } \mathbf{Ax} = \mathbf{y}$$

# Minimum Total Variance

1. From reconstruction problem:

$$\min(TV(\mathbf{x})) \text{ subject to } \mathbf{Ax} = \mathbf{y}$$

2. The variation of $1D$ of $\mathbf{x}$ is:

$$var(\mathbf{x}) = \frac{1}{N}\left((x_1 - \bar{x})^2 + (x_2 - \bar{x})^2 + \cdots + (x_N - \bar{x})^2\right)$$

3. Since Variance is second order statistics, minimum TV problem is solved by casting it into a SOCP problem.
4. Some package to use minimum TV method for example is L1-magic
5. L1-magic is Matlab routine written by Romberg and Candes from Stanford University (USA) : https://statweb.stanford.edu/ candes/l1magic/
6. After extraction, the file for min TV is `tveq_logbarrier` located at folder `optimization`

# Minimum Total Variance

1. Syntax for `tve_logbarrier`:
   ```
   xEst = tveq_logbarrier(x0, A, At, y)
   ```
2. $x_0$ is initial value, can be chosen as $x_0 = \mathbf{A}^T\mathbf{y}$
3. **A** is sensing matrix
4. **At** is sensing matrix for large scale problem (ignored if A is inputed as $M \times N$ matrix)
5. $y$ is compressed signal
6. *lbtol* duality gap threshold, typically a small number (default = 0.001)
7. *mu* a barrier constant for each iteration (default = 10)
8. slqtol - Tolerance for estimate error; ignored if A is a matrix. Default = 1e-8.
9. slqmaxiter - Maximum number of iterations; ignored if A is a matrix. Default = 200.

# Exercise 5

Solve the following CS reconst using min TV!

1. Let $\mathbf{x} = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$ $(N = 9)$
2. generate $\mathbf{A}$ of size $5 \times 9$ with Gaussian Random distribution
3. Normalize each column in $\mathbf{A}$ to unity
4. Calculate $\mathbf{y} = \mathbf{A}\mathbf{y}$
5. From $\mathbf{A}$ and $\mathbf{y}$, using CVX, estimate the original $\mathbf{x}$

# Answer to Exercise 5

**Ans: Matlab Code:**

```
x=[2 0 0 0 0 0 0 0 0 0]';    % Original data
N=length(x); M = 4;          % dimension of A
A=randn(M,N);               % compressed matrix
y = A*x;                     % compressed signal
x0 = A'*y;                     % initial value
tic                           %start measure recons time
xest = tveq_logbarrier(x0,A,[], y, 1e-3, 5, 1e-8, 200)
elapse = toc                % stop measuring time
```

Telkom University

# SUMMER SCHOOL 2019

## PART III : Reconstruction Cases and CS Applications

Prof. Dr. Andriyan B. Suksmono
Dr. Koredianto Usman

17 Juli 2019

Institut Teknologi Bandung and Telkom University

# Objectives

1. Introduce some cases in CS reconstruction
2. Intoduce some CS applications

Notes:

- Tutorial approach is used in this section
- `CVX`, and `omp` are necessary in most examples.

# Reconstruction cases

There are three most common CS reconstruction cases:

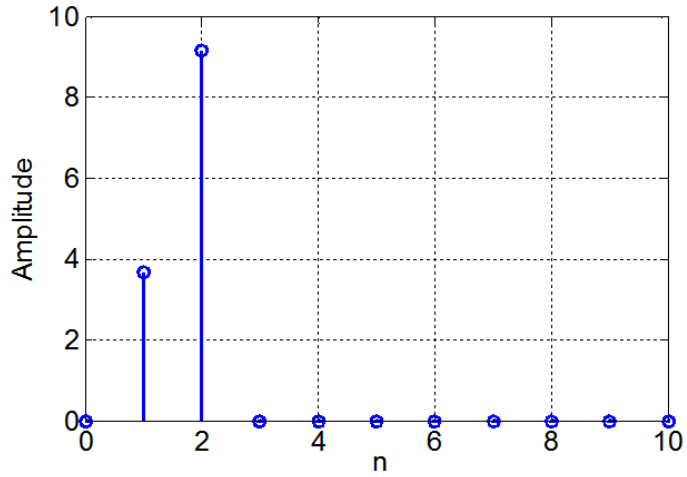
1. time domain sparsity
2. a certain domain sparsity
3. 2D CS reconstruction

# Time domain sparsity

1. In time domain sparsity, the signal **x** can be directly compress using **A**
2. It is important to carefully and slowly determine the compression matrix **A**
3. From **A** and **y**, the estimated **x** is obtained using CS reconstruction algorithm

# Time domain sparsity

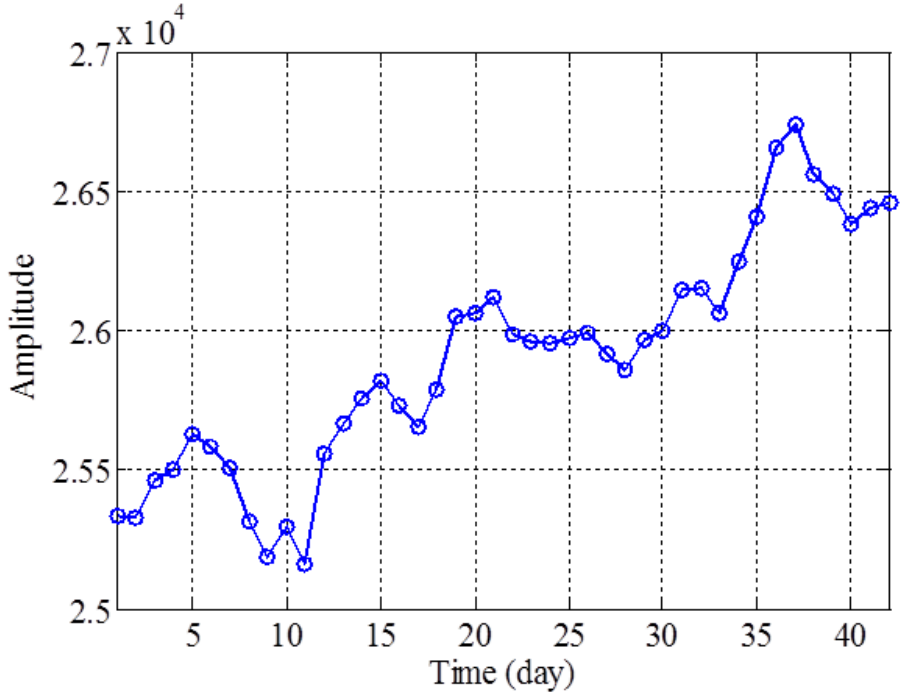
Consider a sparse signal of length 11 as follows:



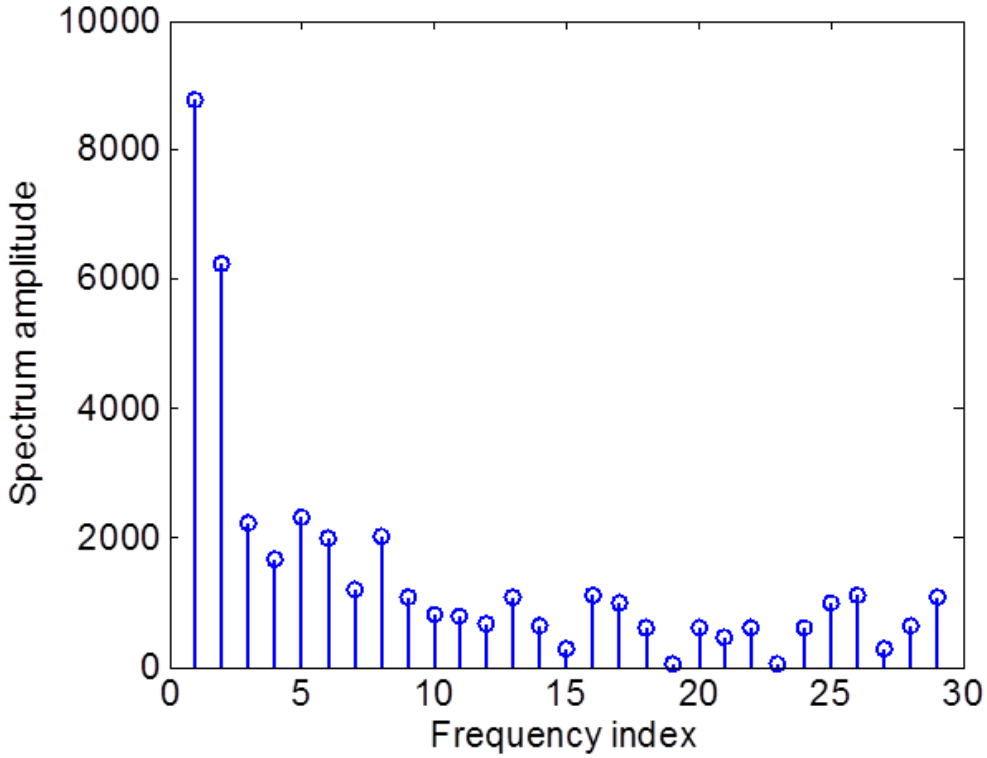

1. what the length of compressed signal do we prefer?
2. what the distribution of compression matrix do we prefer?
3. which reconstruction algorithm do we prefer?
4. Try to simulate our selection in Matlab!

# A certain domain sparsity

1. In fact, most of real-life signal is not in time domain sparsity
2. How do we know in which base our signal is sparse?
3. Some clues, such as homogenity, slow or regular pattern change, may suggest the signal is sparse in frequency domain
4. The following is an example of Dow Jones index values from 1st of August to 28th of September 2018



# A certain domain sparsity




1. Above is frequency spectrum of the signal (after FFT)
2. After thresholding (e.g vthresh = 4000), we obtain the sparse signal
3. Thresholding causes perfect reconstruction is not possible

# A certain domain sparsity (From Slide 1)

1. Let $\mathbf{x}$ of length $N$ is sparse in $\Psi$, that is

$$\mathbf{b} = \Psi\mathbf{x}$$

or

$$\mathbf{x} = \Psi^{-1}\mathbf{b}$$

where $\mathbf{b}$ is a sparse signal

2. We compress $\mathbf{x}$, using $\mathbf{A}$, that is

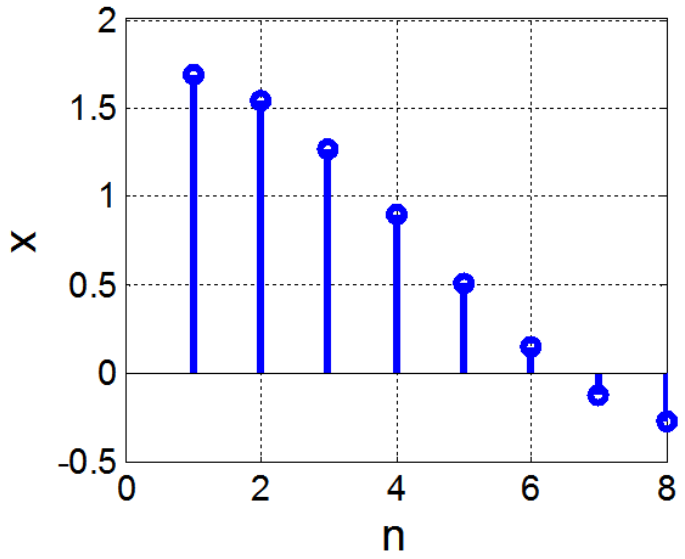
$$\mathbf{y} = \mathbf{A}\mathbf{x} = \left(\mathbf{A}\boldsymbol{\Psi}^{-1}\right)\mathbf{b}$$

3. Now, instead of viewing $\mathbf{y}$ as a compression of $\mathbf{x}$ using $\mathbf{A}$,
4. we view $\mathbf{y}$ as a compression of $\mathbf{b}$ using $\mathbf{A}\Psi^{-1}$
5. Therefore, we reconstruct $\mathbf{b}$ from $\mathbf{y}$ and $\mathbf{A}\boldsymbol{\Psi}^{-1}$
6. Then calculate $\mathbf{x} = \Psi^{-1}\mathbf{b}$

# Example

Let $\mathbf{x} =$

$$\begin{bmatrix} 1.688 & 1.539 & 1.263 & 0.902 & 0.512 & 0.152 & -0.124 & -0.274 \end{bmatrix}^T$$

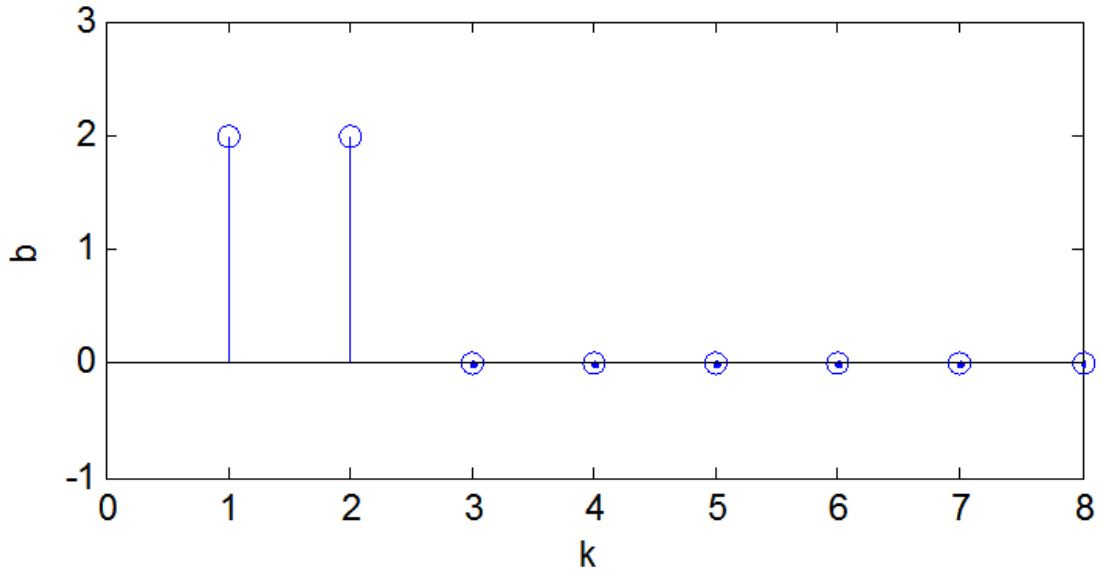



1. As the signal slowly change over time, we guess that this signal is sparse in frequency
2. Either we use FFT (complex valued) or DCT or DST (real-valued)

# Example

**Matlab Code**:

```
x = [ 1.688  1.539 1.263 0.902 0.512 0.152 -0.124 -0.274]';
PSI = dct(eye(length(x)));  % create N x N base matrix
b = PSI * x                 % get representation in base PSI
figure; stem(b)             % plot b
axis([0 8 -1 3])            % adjust axis
xlabel('k'); ylabel('b')     %put x and y label
```



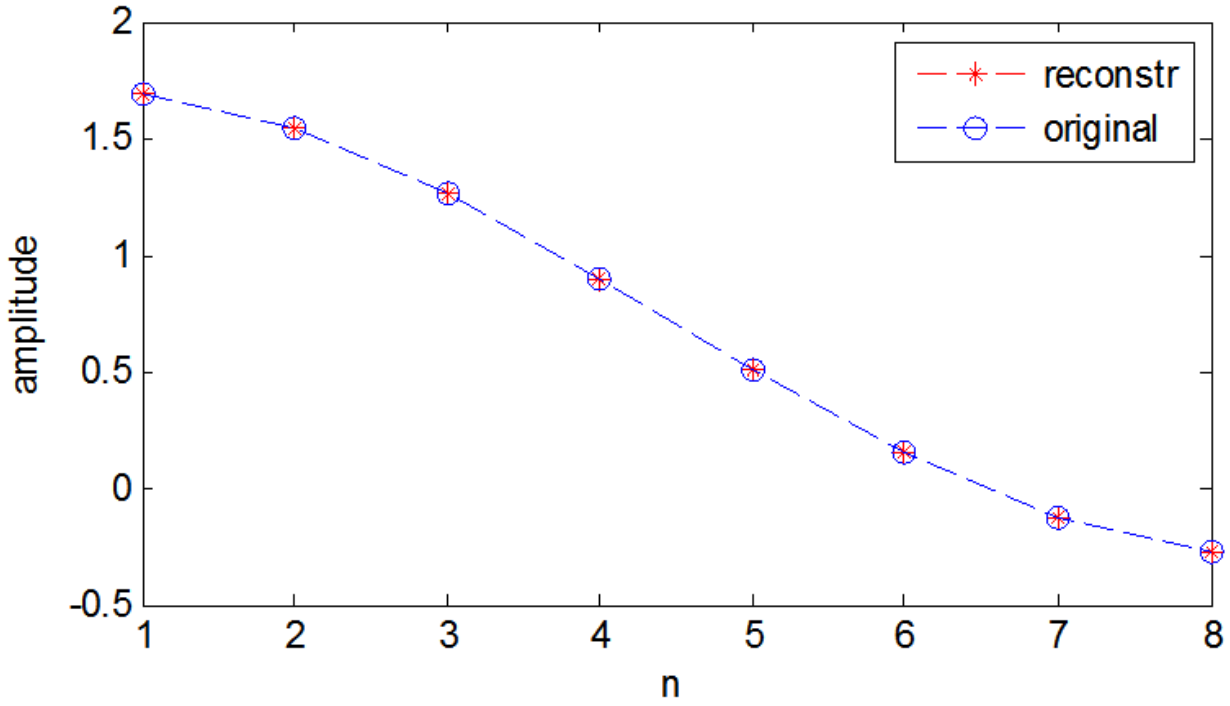
# Example

1. As we observe, **x** is sparse in DCT base
2. We can continue to compress **x**

**Matlab Code**

```
x = [ 1.688  1.539 1.263 0.902 0.512 0.152 -0.124 -0.274]';
iPSI = idct(eye(length(x)));  % prepare inverse base
M = 6;  N = length(x);   A = randn(M,N));  %compress  mtx
y = A*x                  % compress x using A
At = A*iPSI              % prepare for reconstruction
k = 2                   % declare sparsity for OMP recons.
bRec = omp(At, y, k)       % estimate b.
xrec = iPSI * bRec         % transform back to x
figure; plot(xrec, 'r--*'); hold on;  %plot reconstr
plot(x, 'b--o');                      % plot original
xlabel('n'); ylabel('amplitude')
legend({'reconstr' , 'original'})
```

Matlab Execution result (We may re-run the program several time until perfect reconstruction is obtained):




1. Re-run this program for $M = 5$ and $M = 4$
2. We can improve compression matrix by normalizing column length to unity.

# Exercise 6

1. Try to compress the Dow Jones Index [1]
2. Read the data from Excel File using `xlsread` command
3. Use DCT as sparsity basis
4. Use compression ratio of 1:2
5. Use OMP as reconstruction algorithm (Try $k = 5, 10$, etc)
6. Plot the reconstructed signal and original signal in the same figure.
7. Calculate the MSE value!
8. Try again using CVX!

[1] Get data from : quotes.wsj.com/index/DJIA/historicalprices **or** korediantousman.staff.telkomuniversity.ac.id/CS

# Audio Compression

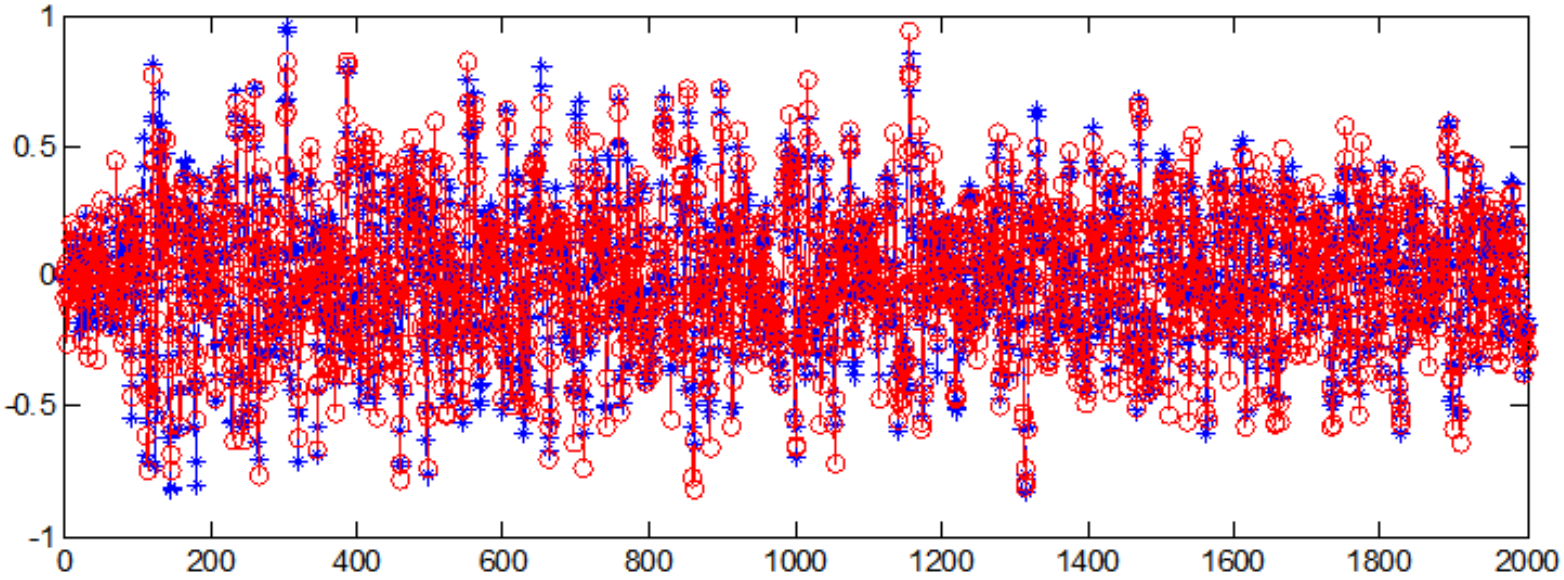
1. Most of audio signal is relatively sparse (or compressible) in frequency domain
2. A long audio file can be segmented into shorter interval or segments
3. compression is performed in each segment
4. DFT (complex valued signal) or DCT of DST (real valued signal) can be used as sparsity basis
5. In next example: piano file compression
6. Get wav file `piano.wav` [2]

[2] from : `korediantousman.staff.telkomuniversity.ac.id/CS`, available during this lecture

# Audio Compression

```
[name1 path1] = uigetfile('*.wav', 'Pick a file');
[x fs nbits] = wavread([path1 name1]);
temp = length(x); N = min([temp 2000]); % limit length
xshort = x(1:N);
M = round(N/2);
A = randn(M,N);
iPSI = dct(eye(N));
y = A*xshort;              % compression
k = M;                     % sparsity is unknown
At = A * iPSI;             % compress mtx in b
brec = omp(At, y, k); %reconstruct for brec
xrec = iPSI * brec;    % estimate x
figure; plot(xshort,'b--*'); hold on;
plot(xrec, 'r--o');
legend({'original', 'reconstructed'})
```

# Audio Compression




1. blue color is actual data and red is reconstructed one
2. In implementation, it is necessary to implement compression matrix and basis matrix so that **out of memory** problem can be solved.

# Data Missing Reconstruction

1. Missing data is special case of compression
2. Example:
3. original: `x = [2 0 1 3 0 9 0 5]`
4. missing data : `y = [2 3 9 0]`
5. elements at : 2nd, 3rd, 5th, and 8th are missing
6. missing data **y** can be obtained from **Ax**
7. this **A** is called random down-sampling matrix

# Data Missing Reconstruction

1. To produce missing data : `y = [2 3 9 0]`$^T$
2. From: `x = [2 0 1 3 0 9 0 5]`$^T$
3. Then

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

4. Random down-sampling **A** has only single '1' in each row
5. in each row, the element '1' appears at position that value is taken
6. maximum only single '1' at each column

# Data Missing Reconstruction

In Matlab, an $5 \times 10$ Random down-sampling can be created for example as follows:

```
N1 = 10;   M1 = 5;          % initiate the dimensio of A
permVector1 = randperm(N1);   % do random permutation
randPos1 = permVector1(1:M1); % take only first M values
randPos1 = sort(randPos1);    % short ascending
Now construct random down-sampling matrix
Atemp1 = zeros(M1, N1);
for ii = 1 : M1
   posii = randPos1(ii);
   Atemp1(ii,posii) = 1; %assign 1 to correct position
end % for ii = 1 : M1
A = Atemp1; % A is ready as random sampling matrix
```

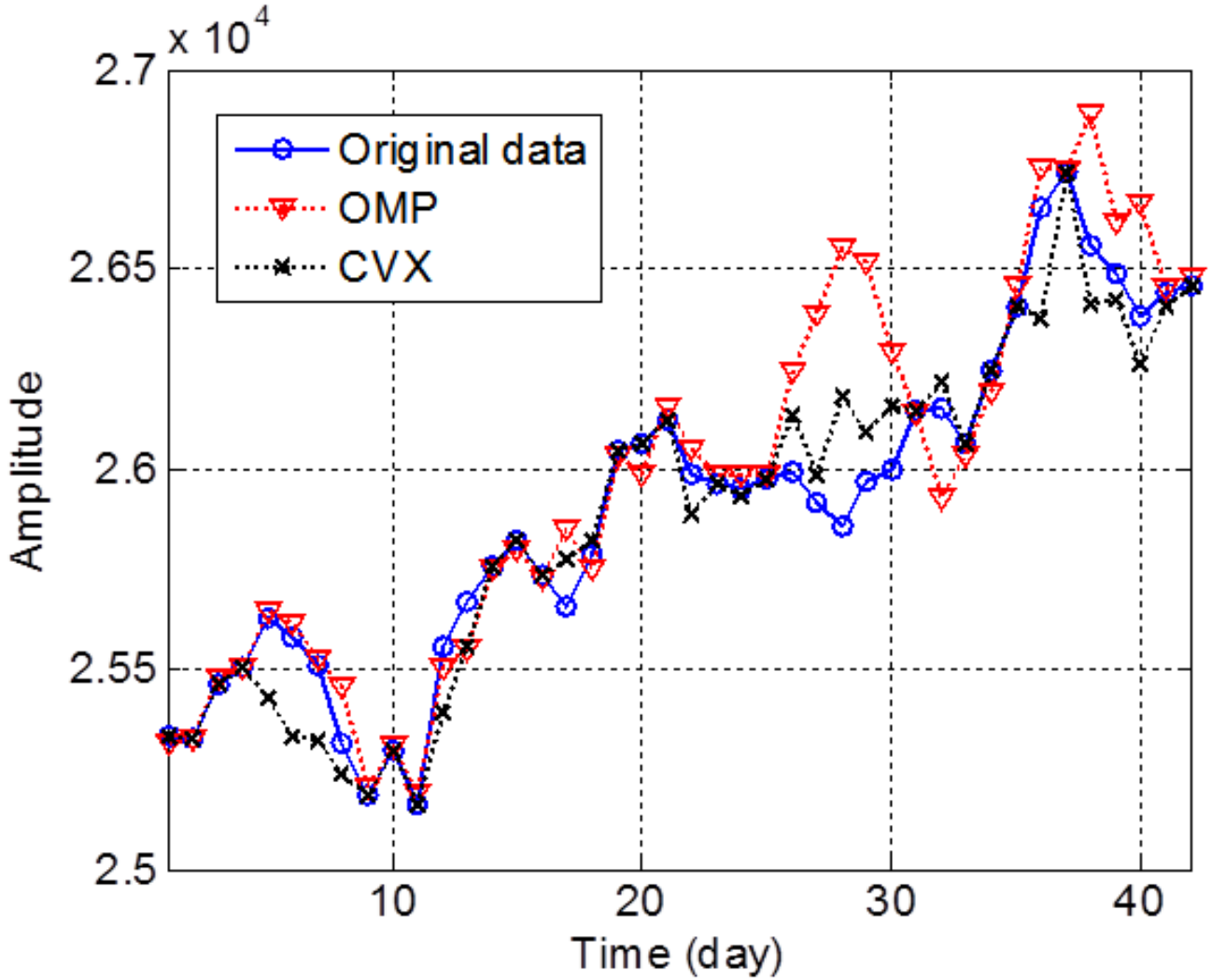
# Exercise 7

In this exercise, we will remove some data in Dow Jones index previously using Random Down-sampling matrix A at compression rate of 1:2.

1. Read the raw data **x** from EXCEL using `xlsread`
2. Create compression matrix **A** with M is about $N/2$ (use `round` function if N is odd)
3. Use DCT as sparsity base (`iPSI`)
4. Compress **x** using **A**
5. Reconstruct **b** using **A** and **iPSI**
6. Either OMP or CVX can be used.
7. Estimate **x** from **b** and **iPSI**
8. Plot both original **x** and its reconstructed value

# Missing Data Reconstruction

Example of the reconstruction result:



# Image Compression

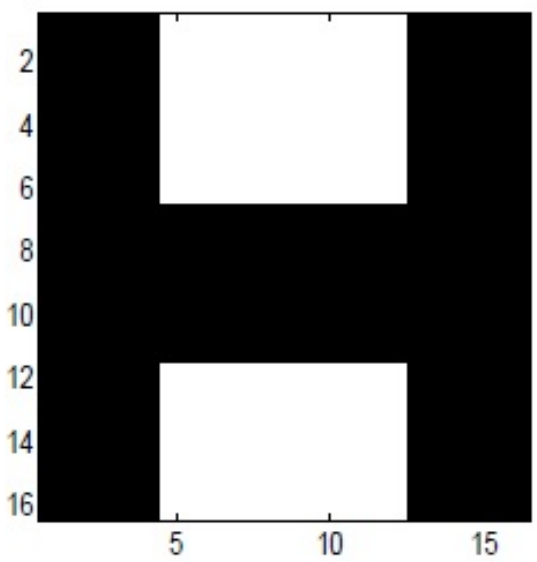
1. Image is 2D signal
2. 2D to 1D conversion is necessary before CS compression
3. after reconstruction, a 1D to 2D conversion is necessary.
4. either **2D to 1D** or **1D to 2D** conversion can be performed using `reshape` command Matlab.
5. Except the binary image, gray or color image is usually sparse or compressible in frequency domain
6. Binary image is usually sparse in time domain base
7. Method of minimum TV is usually more suitable for image CS reconstruction.

# Image Compression

1. Image is 2D signal
2. 2D to 1D conversion is necessary before CS compression
3. after reconstruction, a 1D to 2D conversion is necessary.
4. either **2D to 1D** or **1D to 2D** conversion can be performed using `reshape` command Matlab.
5. Except the binary image, gray or color image is usually sparse or compressible in frequency domain
6. Binary image is usually sparse in time domain base
7. Method of minimum TV is usually more suitable for image CS reconstruction.

# Image Compression

1. To give an illustration of Image Compression using CS, we discuss a compression of an 'H' image of dimension $16 \times 16$
2. Here the compression matrix **A** is generated using real and imaginary component of FFT coefficients.
3. Minimum TV from L1-Magic is used as the reconstruction algorithm



# Image Compression

```
% 1. Create original image : letter H, 16 x 16 pixel
  I = zeros(16,16);
  I([1:6],[5:12])=ones(6,8); I([11:16],[5:12])=ones(6,8);
%2. display original image;
  figure(1);
  imagesc(I); colormap(gray); title('original image');
%3. Normalize and remove DC component of signal
  I = I/norm(I(:)); %normalized image
  I = I - mean(I(:)); %zero mean image
%4. change image to 1-D array /vektor
  [n_row,n_col] = size(I); n = n_row;
  N = n_row*n_col; % total pixel in original signal
  x = reshape(I,N,1);
```

# Image Compression

```
%5. Prepare compression matrix
  f_comp=4;           %  compression factor
  K = N/f_comp;  %  observation sample
  P = randperm(N)';
  q = randperm(N/2-1)+1;
  OMEGA = q(1:K/2)';
  FT = 1/sqrt(N)*fft(eye(N));
  A = sqrt(2)*[real(FT(OMEGA,:)); imag(FT(OMEGA,:))];
  A = [1/sqrt(N)*ones(1,N); A];
  At = [];
%6. performs compression
  b = A*x;
```

# Image Compression

```
% 7. Prepare for reconstruction using min TV
% initial point
x0 = A'*b;
tic;
%reconstruct using total variance method
xp = tveq_logbarrier(x0, A, At, b, 1e-3, 5, 1e-8, 200);
time = toc
%display the reconstructed image
Ip = reshape(xp, n, n);
figure(2);imagesc(Ip);colormap(gray);
title('reconstructed image');
```