1 Mathematical Techniques

Most natural signals are continuous, but storage and computational devices are digital. Thus, for any information processing, the signals need to be converted from continuous to discrete domain. The conversion from continuous/analog domain to the discrete/digital domain is called sampling. The sampling requirement is guided by the famous Shannon–Nyquist–Whittaker–Kotelnikov sampling theorem. Plainly speaking the theorem says: In order to reconstruct a signal from its sampled measurements, the sampling rate should be at least twice the maximum frequency content of the signal.

There are several problems with this approach. The first problem is that the signal is assumed to be low-pass, that is, limited by a maximum frequency. However, naturally occurring signals are not exactly low-pass. There are two approaches to address this discrepancy – the first one is to filter the signal through a low-pass filter; the second approach is to sample at a very high rate so that high frequency contents of the signal can be captured during sampling. When the signal is artificially made to be low-pass, it loses its natural sharpness. It is blurred at the onset; one can only sample and reconstruct the blurred signal. The second approach is even more problematic. In order to capture high frequency content of the signal, the sampling rate has to increase; requirements for such fast sampling challenge the physics of the sampling device. On the other hand, it produces a large amount of digital data that is difficult to store and manipulate (one must remember that computers are not always available for processing sampled signals). The third problem with this approach is that try, however, hard, there will always be some high frequency content that will not be captured during the reconstruction process. During reconstruction, this leads to Gibb's phenomenon along sharp signal discontinuities.

In the previous paragraph, we have discussed the problems pertaining to sampling (limitations on the physics of the scanner) and reconstruction (blurry images or Gibb's artifacts). The other problem with Shannon–Nyquist sampling is that it is wasteful. More often than not the signal is compressed for ease of storage and retrieval. The compression is generally effected via transform coding. Let us take the example of digital photography. To compress a digital image, its transform coefficients are computed – the discrete cosine transform for use in JPEG or wavelet transform for use in JPEG2000. In transform coding,

only the transform coefficients that have large values are preserved and quantized and the smaller coefficients are discarded. Since the number of large valued transform coefficients is much smaller than the total number of pixels in the image (usually 10% or less), the storage requirements for the quantized transform coefficients are much smaller. For example, a 10 megapixel raw image (from any standard commercial camera manufacturer) is usually represented by around 12 to 13 megabytes, but a high quality JPEG image (virtually indistinguishable from the raw image) would only consume 2 to 2.5 megabytes – thus effectively compressing the image by 5 to 6 times. Thus, the current digital image acquisition and storage is highly wasteful. We acquire a large amount of data (raw image) just to discard them later on (using transform coding). This elicits the following question: is it possible to acquire much less number of samples (than the full resolution of an image) and still be able to reconstruct the image from these samples? Compressive sampling (CS) answers this in the affirmative.

The requirement for reducing the number of samples for an MRI arises from a slightly different perspective. In MRI, we are interested in the spatial domain signal but are sampling in the Fourier (K-space) domain. The spatial field-of-view (FoV) dictates the K-space sampling interval. Larger the FoV, denser should be the sampling (smaller sampling interval). On the other hand, the resolution in the spatial domain dictates the K-space FoV; in order to get high resolution image, the FoV should be larger, that is, more high frequency samples need to be sampled. Ideally we require an MRI image that has a large spatial FoV as well as of high resolution (large K-space FoV). Meeting these two requirements simultaneously is a challenging task. A large spatial FoV demands dense sampling (small sampling interval) whereas a high spatial resolution requires increasing K-space FoV. Either way, the number of samples required to be collected increases.

The problem with collecting a large number of samples during MRI scan is the increase in data acquisition time. Densely sampling the K-space on a regular Cartesian grid is time-consuming. This is the reason, MR imaging is slower compared to other popular medical imaging modalities. There are several reasons why one would like to accelerate the scan by reducing the acquisition time. First, it is a question of patient comfort. It is uncomfortable for a patient to remain inside a claustrophobic scanner; to make matters worse, MRI scanners generate noise (owing to the gradient being switched on and off in the RF coils). Even if we forget about patient comfort, we should be concerned about the image quality. Since, MRI acquisition takes time, the patient has to stay in the scanner for a considerable period of time. He/she might move in there. Such movement would result in motion artifacts in the final image. These are the serious problems that must be addressed. Such problems arise in static MRI. The situation worsens for a dynamic MRI. In the dynamic MRI, we want to obtain scans with high spatial and temporal resolutions. But there is always a trade-off between the two. If one is interested in real-time events in dynamic MRI, the temporal resolution needs to be high. In such a case the cost to be paid is in terms of spatial resolution.

The challenge today is to reduce MRI acquisition time and accelerate the scan. The data acquisition time is reduced if the number of K-space samples collected is less. The question is "Is it possible to collect less K-space samples and still be able to reconstruct the image fairly accurately?" The answer again is in the affirmative. One has to resort to compressed sensing (CS) to understand and implement such reconstructions.

When the K-space is not fully sampled, the reconstruction problem becomes under-determined. Consequently, there is an infinite number of solutions. To get a physically

plausible solution, one needs to have some prior information regarding the solution. MR images are known to be locally correlated/redundant. This local correlation is exploited by CS techniques to recover the solution. However, CS is not the only solution, there are other approaches to solve the under-determined problem. In this chapter, we will cover the mathematical topics that will be useful for various MRI reconstruction scenarios.

One way to study the different MRI reconstruction problems is to understand them topic wise – static MRI, quantitative MRI, multi-channel MRI, dynamic MRI, and so on. However, we will see that the similar techniques are being applied in various scenarios. Therefore, we decided to dedicate the first chapter in understanding the various mathematical techniques that will feature often in the later chapters. This chapter will broadly cover the topics of sparse recovery a.k.a compressed sensing, low-rank matrix completion, and dynamical modeling.

1.1 Compressed Sensing

1.1.1 Sparse Recovery

Compressed sensing [1], compressive sampling [2], or compressive sensing [3] all pertain to the same field of study; in short this area is referred to as CS in signal processing. CS studies the problem of solving the following inverse problem,

$$y_{m\times 1} = A_{m\times n} x_{n\times 1},\ m < n \tag{1}$$

This is an under-determined inverse problem and in general has infinite number of solutions. But what if the solution is known to be sparse? By sparse, we mean that the n dimensional vector has only s non-zeroes and rest n-s zeroes; we will call such vectors to be s-sparse. Is it possible to solve for an s-sparse vector by solving the under-determined inverse problem (1)?

Let us try to understand the problem intuitively. In the most favorable scenario, we would know the s positions corresponding to the s non-zeroes in x. Let Ω be the set of non-zero indices. In such a case, solving (1) is the same as solving the following,

$$y = A_\Omega x_\Omega \tag{2}$$

If, $s < m$, the inverse problem (2) is over-determined and therefore has a unique left inverse. Thus, the non-zero positions in x can be obtained by,

$$x_\Omega = \left(A_\Omega^T A_\Omega\right)^{-1} A_\Omega^T y \tag{3}$$

Unfortunately, the positions of the non-zero elements in x will not be known in general. In such a situation, a brute force technique can be applied to solve (1). We know that the solution is s-sparse. Since we do not know Ω, it is possible to choose s elements from n in ${}^nC_s$ ways; thus, there will be ${}^nC_s$ Ω. What we do is to solve (2) for every such Ω. One can see that this is a combinatorial problem. There is no polynomial time algorithm solve (1) in such a situation. Hence, it is a non-deterministic polynomial (NP) hard problem. There is no technique that is smarter than the brute force method just discussed and all have a combinatorial complexity. It is trivial to understand that the number of equation required for solving the inverse problem via this technique is $m > s$.

In most cases, the number of non-zeroes (s) will not be known; only what is known is that the solution will be sparse. It was mathematically shown by David Donoho [4] that

for most large under-determined system, the sparsest solution is unique, that is, there will not be two (or more) sparse solutions satisfying (1). If that be the case, it is not necessary to seek an s-sparse solution (s is unknown anyway). What we can do is to seek the sparsest solution to (1); mathematically this is posed as

$$\min_x \|x\|_0 \text{ subject to } y = Ax \tag{4}$$

Strictly speaking, $\|\bullet\|_0$ is not a norm, it only counts the number of non-zeroes in the vector. CS theory says that if $m > 2s$, the solution of the combinatorial optimization problem (4) is the unique sparsest solution [5, 6]. Unfortunately, as discussed before, (4) is proven to be an NP hard problem [7]. The requirement of having $m > 2s$ can be intuitively understood. An s-sparse vector has $2s$ unknowns – its s positions and the corresponding s values. Thus, in order to solve $2s$ unknowns, at least $2s$ equations will be needed; thus the requirement $m > 2s$.

Since solving for the sparsest solution is an NP hard problem, it is not feasible for practical large scale systems. Is it possible to solve for the sparsest solution via l_2-norm minimization?

$$\min_x \|x\|_2 \text{ subject to } y = Ax \tag{5}$$

This is the minimum energy solution. The nicest property of (5) is that it has a closed-form solution (right inverse). Unfortunately, this will almost never yield the sparsest solution. The reason is "because of the minimum energy solution" – the vector has minimum energy when its total energy is distributed over all the coefficients of the vector. Thus (5) will seek a solution, which is dense as opposed to sparse, but will have small values for all the coefficients.

Between the two extremes of NP hard l_0-norm and the smooth l_2-norm, lies the convex but non-smooth l_1-norm. The l_1-norm is the tightest convex surrogate to the l_0-norm. As it turns out, minimizing the l_1-norm indeed yields the sparsest solution for a large number of problems [1–6, 8],

$$\min_x \|x\|_1 \text{ subject to } y = Ax, \ \|x\|_1 = \sum_{i=1}^{n} |x_i| \tag{6}$$

The l_1-norm prevents diffusion of signal energy into all the coefficients (like l_2-norm) thereby preserving the requirement of the original problem (4). However, since the l_1-norm is convex (albeit non-smooth), (6) can be solved via linear programming, which has a computational complexity of $O(n^3)$. But the price one has to pay in lieu of computational complexity is an increase in the number of equations required; whereas it was possible to solve (1) via (4) from just $m > 2s$ samples, solving (1) via (6) would require more samples,

$$m > Cs \log n \tag{7}$$

where C is a constant.

Instead of using an optimization-based approach, we can also use greedy approximate algorithms to seek the sparsest solution. Some of the popular greedy algorithms are explained in the appendix.

The reason for the equivalence of l_0-l_1 norms and the failure of l_1-norm can be understood geometrically from Figure 1.1. Geometrically the solution of the optimization problems, (4), (5), and (6) can be interpreted as points where the l_p-ball (p = 0, 1, or 2)

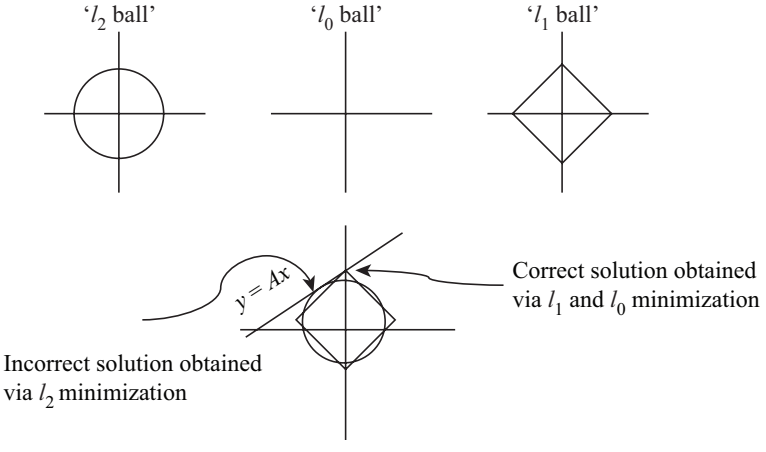

Figure 1.1 Geometry of l_1-norm minimization.

touches the hyperplane $y = Ax$. The l_0-ball will touch the hyperplane only at points where the solution will be sparse (one of the axis in Fig. 1.1); this is the solution we seek. But when we use l_2-minimization (5), the solution will be dense, that is, there will be non-zero values for all the coefficients in the solution vector. However, with l_1-minimization (6), the l_1-ball will touch the hyperplane only at one of the axis – the same positions as the l_0-ball. Theoretical insight into the equivalence of l_0-l_1 norms based on high dimensional geometry can be found in Donoho's work [9].

CS poses conditions on the matrix A so as to guarantee sparse recovery via l_1-norm minimization. One of the most commonly used conditions is the restricted isometric property (RIP) [10]. It says, that l_1-norm minimization will guarantee recovery of all s-sparse vectors if the following RIP condition holds for every s-sparse vector,

$$(1-\delta)\|v\|_2 \leq \|Av\|_2 \leq (1+\delta)\|v\|_2 , \tag{8}$$

where δ is a small constant.

The RIP condition demands that every group of s columns chosen from A must be approximately orthogonal, that is, will approximately preserve the l_2-norm. Checking if the RIP condition holds is a combinatorial problem in itself; since one need to check it for every ${}^nC_s$ groups of columns in A. Constructing such matrices deterministically is an even harder problem. Fortunately it has been proved that random matrices such as i.i.d Gaussian or Bernoulli ensembles or random Fourier ensembles satisfy the RIP with high probability. RIP-based recovery was initially used for proving guarantees with convex l_1-norm minimization. Later RIP-based guarantees have been extended for greedy approximate algorithms [10] and non-convex l_p-norm minimization [11] as well. The problem with greedy algorithms is that they cannot be guaranteed to solve all s-sparse solutions; they can recover most, but not all.

Let us briefly discuss about non-convex l_p-norm minimization ($0 < p \leq 1$).

$$\min_x \|x\|_p^p \text{ subject to } y = Ax, \ \|x\|_p^p = \sum_{i=1}^{n} |x_i|^p \tag{9}$$

We know that by l_0-norm minimization, we can solve the inverse problem (1) from just $m > 2s$ equations; but by l_1-norm minimization, we need about $m > Cs\log n$ equations. Non-convex l_p-norm minimization is between l_0-norm and l_1-norm minimization for $0 < p \leq 1$. Intuitively it is expected that the number of equations required by l_p-norm minimization will be intermediate between the two extremes. It was shown in [14], [15] that is number of equations required by l_p-norm minimization to succeed is

$$m > C_1 s + pC_2 s\log n \tag{10}$$

Here, C_1 and C_2 are constants.

As the value of p reduces ($p \to 0$), the requirement (number of equations) reaches that of l_0-norm minimization.

In real life, the solutions are generally not exactly sparse, they are only approximately so. Such signals are also called "compressible." By approximately sparse/compressible, we mean that the signal has a fast decay, that is, the difference between the signal and its best s-term approximation $\|x - x_s\|_2$ is small. Moreover, in real life, the system is corrupted by noise, therefore, instead of (1) one needs to solve a noisy system of the form,

$$y = Ax + \eta,\ \eta \to N(0,\sigma^2) \tag{11}$$

In practical situations, we need to solve for an approximately sparse solution from a noisy inverse problem (11). In such a situation, we need to relax the optimization constraint and solve the following instead,

$$\min_x \|x\|_p^p \text{ subject to } \|y - Ax\|_2^2 \leq \varepsilon,\ \varepsilon = n\sigma^2 \tag{12}$$

Let x_0 be the true solution, and let $\hat{x}$ be the solution obtained from solving (12). It has been proven (for both the convex [17] and the non-convex case [18]) that the error between the true solution and the obtained solution will be bounded by the s-term approximation error and the noise level,

$$\|x_0 - \hat{x}\|_2 \leq C_1 \|x_0 - x_s\|_2 + C_2\sigma / \sqrt{s} \tag{13}$$

Thus, l_p-norm minimization guarantees a stable and robust recovery in cases where the solution is approximately sparse and the system is noisy. There are standard procedures to solve (12) via quadratic programming.

The problem (12) is called the basis pursuit denoising (BPDN) [18]. It has another equivalent form called the LASSO (Least Angle Selection and Shrinkage Operator). The LASSO formulation solves the least squares problem, the constraint being on the sparsity penalty (l_p-norm),

$$\min_x \|y - Ax\|_2^2 \text{ subject to } \|x\|_p^p \leq \tau \tag{14}$$

The LASSO formulation was proposed for sparse regression problems [19]. Although the two formulations (12) and (14) are equivalent for correct choice of ε and τ, this formulation is not suitable for signal processing problems. This is because, in most cases, we will know the noise in the system (know σ) but not the sparsity of the solution. In regression (machine learning), usually one has some prior idea regarding the sparsity of the solution; therefore, for them, solving the LASSO is preferred.

There is yet another unconstrained Lagrangian formulation for (12) and (14) given by the following quadratic programming problem,

$$\min_x \|y - Ax\|_2^2 + \lambda \|x\|_p^p \tag{15}$$

Here, λ is the Lagrangian multiplier. The three forms (12), (14), and (15) are equivalent for correct choice of λ, τ, and ε. Unfortunately, the relationship between the three parameters is not analytical. Hence, it is not possible to derive one from the other. However, the Pareto curve among the three parameters is smooth. If ε and τ are the two axes for plotting the Pareto curve, the λ is the tangent to the curve at a given value of x. For a detailed treatise on this topic, the reader is referred to [20].

1.1.2 *Group-sparse Recovery*

The problem is to solve an under-determined system of linear equations. So far we have only assumed that the solution is sparse. Nothing else regarding the inter-dependency of values and locations of the variables is assumed. If nothing else apart from sparsity of the signal is assumed, the number of equations required to solve the problem is $m \geq Cs \log n$ via l_1-norm minimization. The question is "Is it possible to reduce the required number of equations even further if dependencies between values and locations of the variables is known?"

The system is under-determined. Intuitively more prior information we have regarding the solution, the lesser will be the search space and better will be the solution; in other words, lesser will be the number of required equations. The answer to the question is in the affirmative and is the subject of model-based CS [21]. This paper addresses different kind of structural dependencies. All of them are not relevant to us. We are interested only in the topic of group-sparse recovery.

Consider a vector such as $\left[\underbrace{x_{1,1} \quad \dots \quad x_{1,n_1}}_{x_1} \quad \underbrace{x_{2,1} \quad \dots \quad x_{2,n_2}}_{x_1} \quad \dots \quad \underbrace{x_{g,1} \quad \dots \quad x_{g,n_1}}_{x_g}\right]^T$. Here the whole vector is divided into g groups, each of the groups has n_i, ($i = 1 \dots g$) variables. Each of the groups is represented by x_i ($i = 1 \dots g$). By group we mean that either all the values within a group are zeroes or all of them are non-zeroes; it is not that a few are zeroes while the rest are non-zeroes. A group-sparse vector is such that only a few groups (x_i's) will be non-zeroes and the rest of them will be zeroes.

A group-sparse vector can be assumed to be only sparse (ignoring the group structure); this is because the groups are zeroes, the vector will obviously be sparse. But the knowledge about the group structure is an extra piece of information. If we know that the solution to the under-determined problem (1) is group-sparse, the extra information regarding the structure of the sparsity is going to help us (in reducing the number of equations required to find the solution). There are only a few theoretical studies in this area [22, 23]. In [22], it is proven that the number of equations required for recovering a group-sparse solution is

$$m \geq C_1 \max(n_j) + C_2 s_g \log g \tag{16}$$

where $\max(n_j)$ denotes the number of elements in the largest group, s_g is the number of non-zero groups, and g is the total number of groups.

It has been analyzed in [22] that group-sparsity is useful when the number of elements in the largest group is small compared to the total number of non-zero elements in the solution. This condition is named as "strong group-sparsity" in [22]. As we have seen before, the number of equations required to solve the problem is intimately related with the algorithm used to solve it. For group-sparsity, the aforesaid estimate (16) holds only when the solution is recovered via the following optimization problem,

$$\min_x \|x\|_{2,1} \text{ subject to } \|y - Ax\|_2^2 \leq \varepsilon, \ \|x\|_{2,1} = \sum_{i=1}^{g} \|x_i\|_2 \tag{17}$$

Here $\|\bullet\|_{2,1}$ is a mixed norm. It is the sum of the l_2-norms of the groups. The theoretical analysis behind using this mixed norm can be found in [23]. Here we explain it intuitively. We know that minimizing the l_2-norm yields a dense solution; hence, the inner l_2-norm over the groups ensures that if a group is selected, all its elements should be non-zeroes. The sum over l_2-norms is similar to group l_1-norm – it enforces selection of only a few groups. It is easy to verify that if the group information is lost, that is, when each of the coefficients constitute a group, the $l_{2,1}$-norm minimization problem boils down to regular l_1-norm minimization.

1.1.3 *Row-sparse Multiple Measurement Vector Recovery*

In sparse recovery, the problem is to recover a sparse vector from its noisy lower dimensional projections (11).

$$y = Ax + \eta$$

This is the single measurement vector (SMV) problem. In a multiple measurement vector (MMV) problem, multiple vectors are projected onto a lower dimensional subspace by the same projection matrix, that is,

$$y_{(j)} = Ax_{(j)} + \eta, \ j = 1...N \tag{18}$$

In a compact matrix–vector notation, this is represented as,

$$Y = AX + \eta, \ Y = [y_{(1)} | ... | y_{(N)}] \text{ and } X = [x_{(1)} | ... | x_{(N)}] \tag{19}$$

The problem is to recover the matrix X given Y and A. Compressed Sensing asks the question: "If we know that all the $x_{(j)}$'s have a common sparse support, how do we recover X?". In other words, we have to recover X knowing that the matrix is row-sparse, that is, there are only a few rows that are non-zeroes and the rest are zeroes. This is the row-sparse MMV recovery problem.

The recovery is formulated as the following mixed-norm optimization problem:

$$\min_X \|X\|_{2,1} \text{ subject to } \|Y - AX\|_F^2 \leq \varepsilon, \ \|X\|_{2,1} = \sum_{i=1}^{n} \left\|X^{i\rightarrow}\right\|_2 \tag{20}$$

where $\left\|\bullet^{i\rightarrow}\right\|_{2,1}$ denotes the ith row.

There are several papers [24, 25] that studied this problem. For theoretical guarantees, the reader is referred to the aforesaid works. However, the intuition behind using the

mixed norm is the same as before (group-sparsity). The l_2-norm over the rows enforces the selected rows to have non-zero values for all the elements; the sum over the l_2-norms promotes selection of only a few rows.

The problems of group-sparse recovery and row-sparse MMV recovery are somewhat related. More specifically, the latter can be recast as the former. This follows from the simple Kronecker product; the MMV problem can be recast in the following Kronecker product notation,

$$\text{vec}(Y) = I \otimes A\text{vec}(X) + \text{vec}(\eta) \tag{21}$$

Here, $I \otimes A$ is a block diagonal matrix with A in each block and vec(X) is formed by column concatenation of the matrix X, that is, $\text{vec}(X) = \left[x_{(1)}^T \; x_{(2)}^T, \ldots, x_{(N)}^T \right]^T$. Without loss of generality, we can assume vec(X) to be constituted of n groups, each group corresponding to a row in X, that is, $\text{vec}(X) = \left[X^{1\rightarrow} \; X^{2\rightarrow}, \ldots, X^{n\rightarrow} \right]^T$. As the matrix X is row-sparse, the vector vec(X) will be group-sparse. There lies the similarity between group-sparse recover and row-sparse MMV recovery. This also explains the similarity between the mixed norm minimization problems (17) and (20).

1.1.4 *Synthesis and Analysis Priors*

Till now we have been living in the "Sparseland." The solution itself was assumed to be sparse. However, for most natural systems, the solution is not sparse in itself; but most natural signals are sparse in a transform domain, for example, seismic images are sparse in curvelets, medical images are sparse in wavelets, EEG is sparse in Gabor transform, and speech is sparse in short time Fourier transform. Practical applications of CS exploit the transform domain sparsity of such signals.

Linear transforms that are orthogonal or are tight-framed are the best for us. For readers those who are not aware of the difference between the two, the simple distinction is

$$\text{Orthogonal: } \Psi^T\Psi = I = \Psi\Psi^T \tag{21a}$$

$$\text{Tight frame: } \Psi^T\Psi = I \neq \Psi\Psi^T \tag{21b}$$

Both the orthogonal and tight-frame transforms have a left-inverse, but only orthogonal transforms have a right-inverse. In other words, for both the orthogonal and tight-frame transforms, the forward transform followed by the backward transform is identity, but the reverse (backward followed by forward resulting in identity) only holds true for orthogonal transforms but not tight-frames.

The property that makes orthogonal and tight-frame transforms useful to us are the analysis (forward) and synthesis (inverse) equations (holds true for both):

$$\text{Analysis: } \alpha = \Psi x \tag{22a}$$

$$\text{Synthesis: } x = \Psi^T\alpha \tag{22b}$$

Inverse problems that are not sparse by themselves, but are known to be sparse in an orthogonal/tight-frame domain can be framed as follows:

$$y = Ax + \eta = A\Psi^T\alpha + \eta \tag{23}$$

Here, x is not sparse but α is. Thus, the solution is obtained via the following equation,

$$\min_{\alpha} \|\alpha\|_1 \text{ subject to } \left\|y - A\Psi^T\alpha\right\|_2^2 \leq \varepsilon \tag{24}$$

Once the sparse transform coefficient vector is solved, the required solution x is obtained via applying the synthesis equation (22b).

This is the synthesis prior model. More than 90% of CS applications are based on the synthesis prior model. This problem is well understood theoretically and there are well known algorithms to solve this problem. However, this model is restrictive owing to the stringent requirements of orthogonality/tight-frame property. Sometimes signals are modeled to be piecewise smooth (e.g., magnetic resonance images); such signals can have a sparse gradient. Since finite differencing is used for computing the gradient, the signal is also said to be sparse in finite differencing domain (D). Unfortunately, finite differencing do not follow the analysis and synthesis equations ($\alpha = Dx$ is sparse but $x \neq D^T\alpha$), thus it is not possible to frame the inverse problem as (23) and hence, it is not possible to solve for x using (24). There are learned dictionaries that are empirical learned basis for sparsely representing certain signals. In such cases, the analysis equation leads to a sparse transform coefficient vector but the synthesis equation is not an inverse for the learned dictionary. Such cases necessitate the following analysis prior model:

$$\min_{x} \|Dx\|_1 \text{ subject to } \|y - Ax\|_2^2 \leq \varepsilon \tag{25}$$

At the onset we make clear that for orthogonal transforms, the synthesis and the analysis priors are the same, but for tight-frames, they are different. This follows trivially. For orthogonal transforms, the synthesis prior model (24) can be expressed as $\min_{\alpha} \left\|\Psi\Psi^T\alpha\right\|_1$ subject to $\left\|y - A\Psi^T\alpha\right\|_2^2 \leq \varepsilon$; this is nothing but the analysis prior model $\min_{x} \|\Psi x\|_1$ subject to $\|y - Ax\|_2^2 \leq \varepsilon$. However, for tight-frame transforms, this formulation is not possible because $\Psi\Psi^T \neq I$.

The analysis prior model is more generalized. The synthesis prior model can be cast as an analysis prior but not the other way round. Unfortunately, the analysis prior model has not been studied extensively. It is only in the last few years that people have started theoretically analyzing this model. The simple difference between the synthesis and the analysis prior models is that the former solves for the transform coefficients of the solution while the latter solves for the solution itself. Digging a little deeper, the difference between the two models is in the emphasis – the synthesis prior model seeks a solution that has very few non-zeroes while the analysis prior seeks a solution that has many zeroes; the synthesis prior seeks a sparse signal while the analysis prior seeks a co-sparse signal [27].

For understanding the rest of the book, it is not required to dig any further into the sparse synthesis prior and the co-sparse analysis prior models. Remembering the reasons for synthesis and analysis prior modeling is good enough for this book. For a better theoretical understanding of the topic, we ask the reader to refer [27], [28].

1.2 Low-rank Matrix Recovery

In the previous section, we learned the basics of CS theory. The idea of CS germinated from transform coding. Transform coding is the *de facto* standard for all kinds of lossy