TU WIEN

MASTER THESIS

Optimized Sampling of Graph Signals

Author:
Valerija MIKLAUSIC

Main advisor:
Univ.Prof. Dipl.-Ing.
Dr.-Ing. Norbert
GÖRTZ

Co-advisor:
Univ.Ass. Dipl.-Ing.
Gabor HANNAK

January 11, 2017

TU WIEN

Abstract

Faculty of Electrical Engineering and Information Technology
Institute of Telecommunications

Master of Science

Optimized Sampling of Graph Signals

by Valerija MIKLAUSIC

A graph signal is a vector, for which complicated dependencies between the vector components can be expressed by a weight matrix that "connects" components that, e.g., tend to take similar values. A very common example is an image signal (=graph signal), which has a very regular weight matrix, in which neighbouring pixels (=graph signal components) tend to take similar values (textures, such as blue skies, white walls); edges are few, and only here the components take values that are significantly different from one pixel to another. Graph signals can also be defined in various other applications such as recommender systems in online-shopping and in social networks.

A common strand of research in the field is how to reconstruct a graph signal from few sampled components of the signal, when the weight matrix is known. The weight matrix is used to regularize and, thereby, solve the reconstruction problem (which as such is underdetermined). For so-called Tikhonov regularization the recovery problem can be solved analytically, and, moreover, the well-known Gauss-Seidel iterative algorithm can be used to solve the resulting system of linear equations efficiently (without LU-decomposition, Gaussian elimination or a matrix inversion). Hence, graph signal recovery with Tikhonov regularization is feasible even for very large signal dimension.

A particularly interesting problem is how to sample a graph signal with a small number of samples such that the reconstruction with Tikhonov regularization works well (measured by the mean squared error between the original graph signal and its reconstruction). Of course the choice of samples from the graph signal depends on the weight matrix, and it is a main goal of the thesis to analyze and numerically investigate specific sampling schemes. The most simple one is random sampling, which shall be used as a reference scheme. A more sophisticated approach is to exploit the weight matrix to optimize the choice of sampled components.

Acknowledgements

I would like to take a moment and express my gratitude to all the people supporting me, endorsing me and being with me throughout my studies.

I would like to thank my supervisor, Univ.Prof.Dipl.-Ing.Dr.Ing. Norbert GÖRTZ for his continuous support, patience, given knowledge and guidance during the whole time.

Thanks to my friends for being there to share all the happy and sad moments, bad and good times. Without you, these studies would be unimaginable to go through alone. Special thanks to Željko and Ayan for their patience, love and support.

Thanks to Lukas, for support, kindness and all the nice times spent together.

Last but not the least, the biggest thanks goes to my family.
To my sisters, Božana and Tea, for always being part of everything, supporting me, advising me and always having time for me.
And to my parents, Josip and Jelka, without your love and support it would all be impossible, so special thanks to both of you for making it all possible. Regardless on the bad times, this is just one little drop in the ocean of good things which are about to happen. Live strong.

List of Abbreviations

G	Graph
V	Set of Vertices
E	Set of Edges
A	Adjacency matrix
WF	Weighting Function
LS	Least Squares
TR, L_2	Tikhonov Regularization
LR	Linear Regression

List of Symbols

ψ	Graph incidence function
L_*	Lower triangular matrix
U_*	Upper triangular matrix
$\aleph$	Set of natural numbers
$\Re$	Set of real numbers
$\mathbb{C}$	Set of complex numbers
N	Graph signal dimension
M	Number of samples
λ	Scaling factor
σ	Standard deviation of variance of noise
fac	Inter-connectivity parameter
D	Degree matrix
α	Laplacian graph

Chapter 1

Introduction

Many real world situations can be described with a help of two or more points and lines connecting them due to some dependency. The set of those points and lines is called a graph.

Graphs are mathematical structures used to model relations and processes among different entities like biological, social or information systems. With a help of graph theory, one can mathematically describe the various kinds of systems. One example is if we imagine a graph where the vertices represent people. Furthermore, vertices which represent people who are friends will be connected with edges. Having that in mind, it is possible to describe the relation between each two (or more) vertices in the graph.

Other example can be a link structure of a website. If we take into account that each website is represented by one vertex, and a link from one website to another is represented by an edge, we can implement a graph which can describe relations among several websites.

As we can see, depicting everyday life phenomenons with a graph can be very useful in various situations. Even very complicated mathematical formulas sometimes can be explained much easier by using a graph.

In the scope of this master thesis, a reader will be introduced to the basic concepts of graph theory which will give a good introduction to the actual problem statement. The later chapters will describe into more detail the problem of graph signal recovery and graph signal sampling. It will give an explanation of Tikhonov regularization usage with Gauss-Seidel iterative approach for the recovery of a graph signal and an explanation of two sampling methods along with their performance comparisons.

In the end, the final conclusion will be found in which the short comment on the whole thesis will be stated.

Chapter 2

Graph

2.1 Graph

A *graph* G is an ordered triple (V(G), E(G), ψ(G)) consisting of a nonempty set V(G) of *vertices*, a set E(G), disjoint from V(G), of *edges*, and an *incidence function* ψ_G that associates with each edge of G an unordered pair of (not necessarily distinct) vertices of G[1].
If *e* would denote an edge of a graph, and *m* and *n* would denote the vertices of a graph such that $\psi(e) = mn$, then *e* joins the vertices *m* and *n*, and the vertices *m* and *n* are said to be *ends* of *e*.

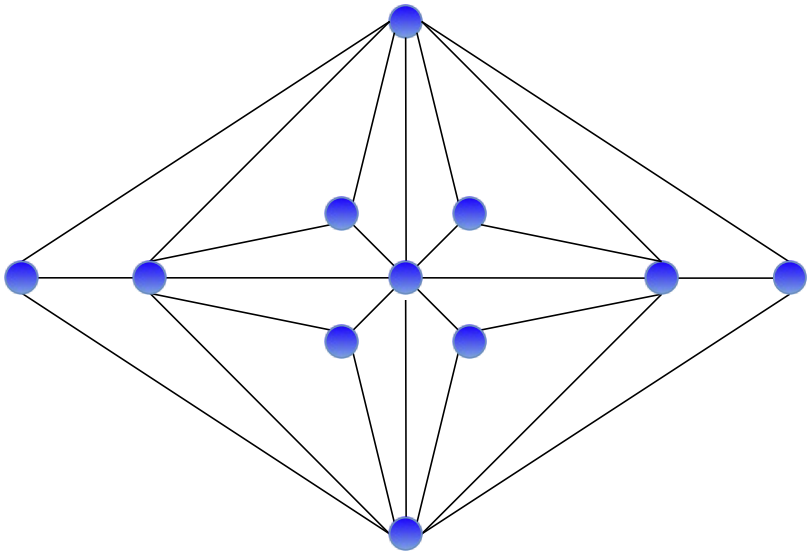


FIGURE 2.1: An example of a graph

Each vertex is indicated by a point, each edge by a line which joins the vertices.
The actual positioning of vertices, or the length of lines which represent the edges, do not play a role in the mathematical description of a graph. Therefore, there is no sole or more correct way of drawing graphs.
If we take a deeper look, we can see that the two graphs, G and X, are identical if and only if they have the same vertices, same edges,

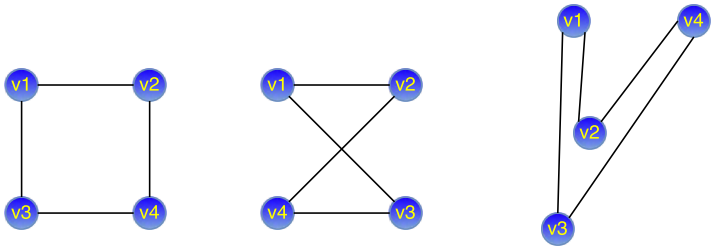


FIGURE 2.2: Three isomorphic graphs

and their incidence functions are the same:

$$V(G) = V(X) \qquad and \qquad E(G) = E(X) \qquad and \qquad \psi_G = \psi_X \tag{2.1}$$

Since they are identical, they can always be described in the same way, i.e. with the same diagram.
On the other hand, there are other graphs, which in the same time obtain the same diagram but are not identical. Those graphs are said to be *isomorphic* graphs.
In general, two graphs G and X are said to be *isomorphic* (written G≅X) if there are bijections $\theta = V(G) \to V(X)$ and $\phi : E(G) \to E(X)$ such that $\psi_G(e) = mn$ if and only if $\psi_X(\phi(e)) = \theta(m)\theta(n)$; such a pair θ, ϕ of mappings is called an *isomorphism* between G and X [1].

Three isomorphic graphs are depicted in the Figure 2.2. The important thing to remember is that once there are two or more isomorphic graphs, they can all be "redrawn", or in better words, translated to the forms of each other without any loss of mathematical or theoretical structure.

2.2 Adjacency matrix and weighted graph

One more way to describe a graph is with a help of it's *adjacency matrix*. To every graph G there is a corresponding v x v matrix called adjacency matrix. It is denoted by $\mathbf{A}(G) = a_{ij}$, where a_{ij} denotes the number of edges joining v_i and v_j. The graph and its corresponding adjacency matrix are shown in the Figure 2.3.

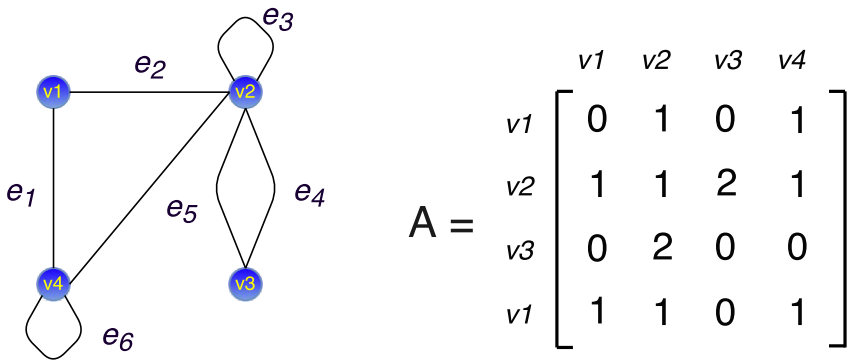


FIGURE 2.3: A graph and its adjacency matrix

In the scope of this thesis, the most used type of a graph will be the weighted graph.
A weighted graph G is an ordered pair (V, w) where V is a set of vertices of the graph defined like:

$$(V = v_1, v_2, ..., v_n \mid n \epsilon N) \tag{2.2}$$

where n is a number from the set of natural numbers, and w is a weighting function(**WF**). The weighting function is used to describe a connectivity degree among two vertices. It gives a real non-negative value $w(v_i, v_j)$ *to each pair of nodes* (v_i, v_j), where $v_i \epsilon V$, $v_j \epsilon V$ and $v_i \neq v_j$. If we assume that a graph has n vertices, the weighted graph will be called *undirected* when it's weighting function is symmetric, i.e., $w(v_i, v_j) \neq w(v_j, v_i)$, for all v_i, v_j, $v_i \neq v_j$. Usually, the weighting function can take any value which is non-negative and real. However, in this thesis we will restrict ourselves to the binary values, i.e., 0 and 1, since the weighting coefficient will be introduced as the measure of the "connection strength" in between the two vertices.
A simple weighted graph, along with some random weights, is depicted in the Figure 2.4.

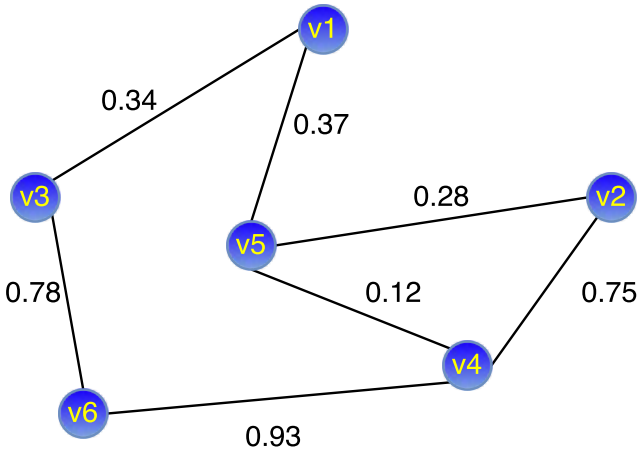
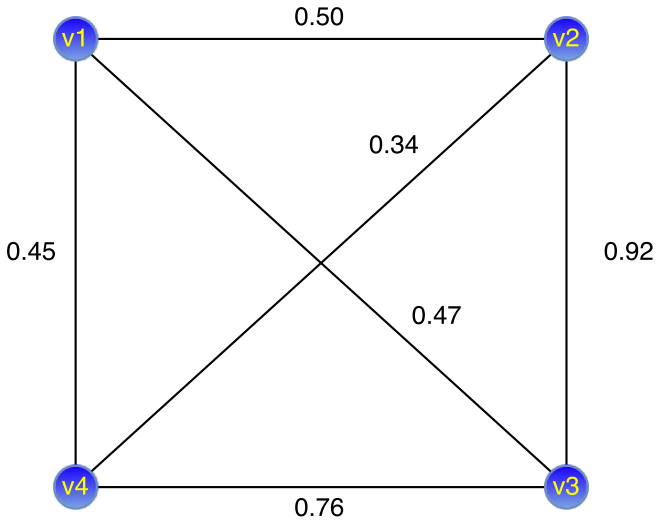


FIGURE 2.4: Weighted graph

Weighted graphs can be described with the adjacency matrices. The adjacency matrix of a weighted graph G = (*V*, *w*) where $V = v_1, v_2, ..., v_n$, is an *n* x *n* matrix *A* defined as[15]:

$$A_G = [a_{ij}] \begin{cases} a_{ij} = w(v_i, v_j), & \text{if } i \neq j. \\ a_{ii} = 0 & \text{otherwise.} \end{cases} \tag{2.3}$$

The matrix becomes a symmetric matrix if a graph *G* is an undirected graph. Undirected graph G with its adjacency matrix A is depicted in the Figure 2.5.



$$\begin{bmatrix} 0 & 0.50 & 0.47 & 0.45 \\ 0.50 & 0 & 0.92 & 0.34 \\ 0.47 & 0.92 & 0 & 0.76 \\ 0.45 & 0.34 & 0.76 & 0 \end{bmatrix}$$

FIGURE 2.5: Weighted graph and its adjacency matrix

Chapter 3

Graph signal

3.1 Graph Signal and applications

The most important applications of graphs can be seen in the fields of time, sensor, computer, transportation networks and more of other scientific fields out of which, digital imaging rises as the most relevant one. In these fields, the data is usually located on the vertices of weighted graphs which describe the wanted network. In order to process that data, the field of signal processing is introduced whose main purpose is to connect the spectral analysis of weighted graphs (which is a theoretical concept) and the algebraic concepts with a help of which wanted data is received and processed. On the further figures we can observe some applications of graph signals.
On the Figure 3.1 a finite periodic discrete time series are depicted. All edges are directed and have the same weight - the weight of 1.

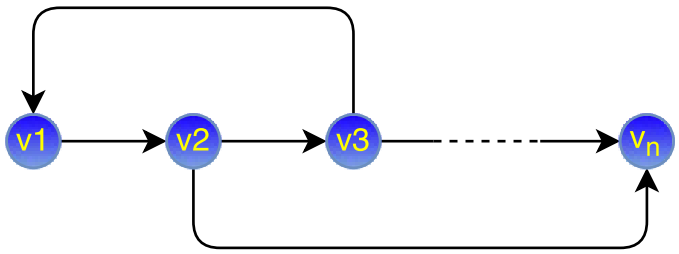


FIGURE 3.1: Periodic discrete time series

Furthermore, as we mentioned before, graph signal techniques are convenient for the usage of digital imaging methods. With a quick glance at Figure 3.2, we can conclude that vertices of a graph are correspondent to pixels. Each pixel depicted on a graph depends on the other four of its "neighbors", i.e., the value of intensity of a pixel is directly connected to the values of intensities of four adjacent pixels. On the Figure 3.2, all the edges are assumed to be undirected with equal weights.

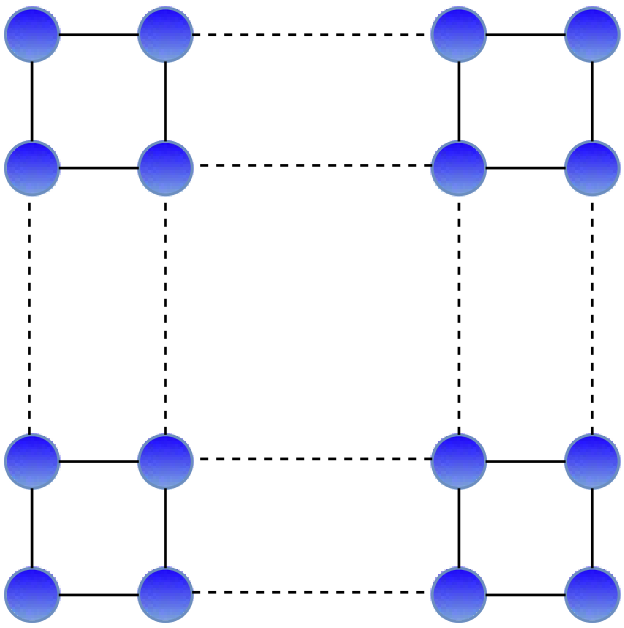


FIGURE 3.2: Digital image

The next really important application is the sensor network data processing. Mostly, this kind of application is used for temperature measurements. On the Figure 3.3., the temperature measurements form 150 weather stations,i.e., sensors, are shown. Each vertex of a graph is connected to it's closest neighbor, and the relation between temperature measurements are shown as distances between the sensors[4].

FIGURE 3.3: Sensor network[4]

The most familiar application of graph signals can be shown in the field of computer networks. Only one of many examples is the hyperlink reference usage. On the Figure 3.4 hyperlinked documents are shown. To be more precise, a set of 50 blogs in the World Wide Web connected by the hyperlink references. As in digital image example, the edges of this graph are equally weighted and unidirected[4].

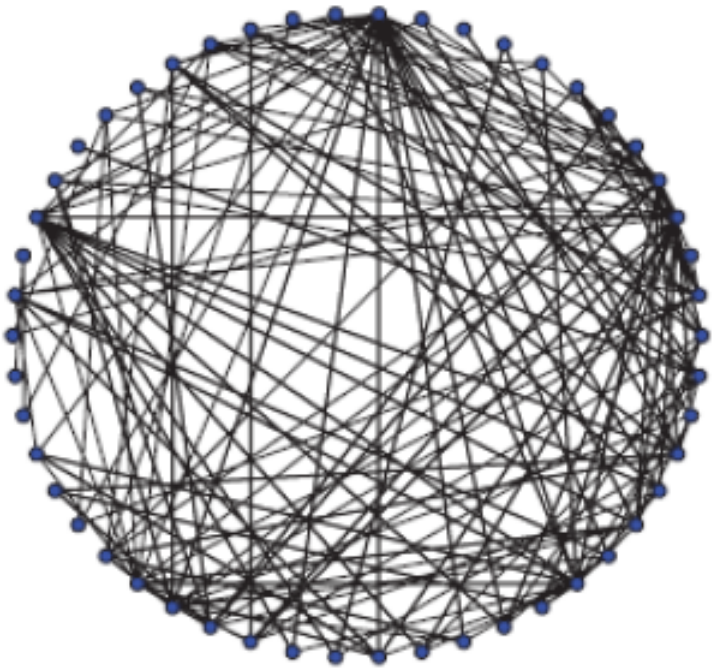


FIGURE 3.4: Hyperlinked documents[4]

3.2 Discrete signal processing on graphs

Graphs are used for the data representation. The vertices of a graph represent finite data samples and each sample resides at one vertex of a graph.
When considering a weighted graph, the weight associated with an edge in a graph usually represents the level of similarity between two vertices that edge is connecting.
The logic behind connectivities can be of a various kind, e.g. physical distance, raw data similarity etc.
In order to process the data which are represented by the vertices of a graph, we need to introduce some basic concepts of discrete signal processing on graphs. Next sections, which are describing the basic concepts of discrete signal processing on graphs, are based on [2],[8] and [6].

3.2.1 Graph shift

Discrete signal processing on graphs studies signals with complex, irregular structure represented by a graph G=(V,A), where $V = v_1, v_2, ..., v_n$ is the set of vertices and A is the *graph shift*, or a weighted adjacency matrix[2]. Adjacency matrix, still, represents the similarities between two nodes whose connection or relation it is describing.

3.2.2 Graph signal

After we have successfully described a graph alone as a set of vertices and edges connecting it, we defined vertices as the data representatives and edges as weight carriers. Weight represents the similarity or relation between two connecting vertices. To map an incoming signal and a graph, i.e. to get a graph, we need to introduce the term *graph signal*. Graph signal can be seen as an *one to one* mapper between the signal coefficients $x_n \epsilon C$ and the vertex v. Once we finish the mapping, we can write a graph signal as a vector:

$$x = [x_1, x_2, ..., x_N]^T \epsilon C^N \tag{3.1}$$

where the n-th signal coefficient will correspond to the node v_n. The example of a graph signal is shown on the Figure 3.5.

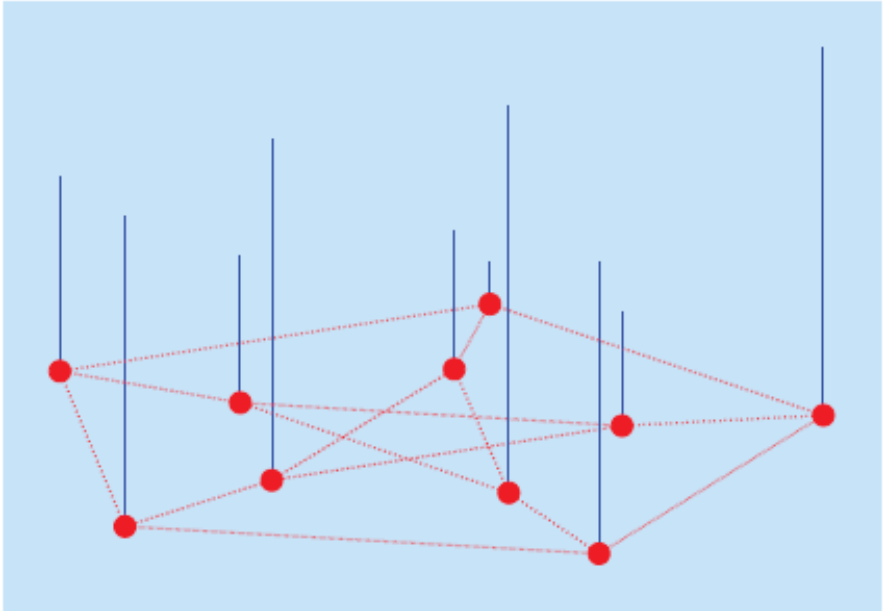


FIGURE 3.5: A random positive graph signal on the vertices of the Petersen graph. The height of each blue bar represents the signal value at the vertex. [6]

3.2.3 Graph Fourier transform

A Fourier transform is used for a signal expansion using basis elements that are non-variant to filtering. In this case, the basis elements which are non-variant to filtering are eigenbases of the adjacency matrix/graph shift A[1].
If the adjacency matrix A has a complete eigenbasis and we consider a spectral decomposition of A :

$$A = V\lambda V^{-1} \tag{3.2}$$

where columns of the matrix V are formed with the eigenvectors of A and $\lambda \epsilon C^{NxN}$ is the diagonal matrix made of corresponding eigenvalues of A: $\lambda_1, \lambda_2, ..., \lambda_N$. These eigenvalues are frequencies on the graph.

Definition 1. *The graph Fourier transform of $x \epsilon C^N$ is:*

$$\hat{x} = V^{-1}x \tag{3.3}$$

The inverse graph Fourier transform is:

$$x = V\hat{x} \tag{3.4}$$

Where the $\hat{x}$ corresponds to the signal's expansion in the eigenvector basis and describes the frequency content of the graph signal x.
The inverse graph Fourier transform is used for the reconstruction of the graph signal from its frequency contect by combining the graph frequency components weighted by the coefficients of the signal's graph Fourier transform.

3.2.4 Laplacian graph

The (unnormalized) graph Laplacian is defined as $\alpha = D - W$, where the degree matrix D is a diagonal matrix whose ith diagonal element d_i is equal to the sum of the weights of all the edges incident to vertex i[6].
Laplacian graph can come to use as a difference operator because for any signal x, it can describe the difference between neighbouring components as[6]

$$\alpha(x)(i) = \sum_{j \epsilon \varphi} W_{ij}[x(i) - x(j)] \tag{3.5}$$

where φ denotes the neighborhood, i.e., the set of vertices which are directly connected to the vertex i.

[1] If there is no complete eigenbases, the Jordan eigenbases of A are used

The matrix of a graph Laplacian is symmetric matrix with real values. Therefore, it has a complete set of orthonormal eigenvectors.

Chapter 4

Graph signal recovery and sampling

4.1 Introduction

In the scope of this thesis, the aim was to describe a simple iterative approach to graph signal recovery out of sampled instances. Later on, one of the sampling processes that fit good to this iterative graph signal recovery problem is described. This chapter was based on a research paper published as an internal paper of TU Wien[3].

4.2 Problem setting

Assuming an incoming signal values are gathered and defined as x_j where *j* = 1, 2, ...,*N*. Again, all of the signal values will be represented by the vertices of the graph (Figure 4.1).

It is also assumed that the weights in between two graph vertices are non- negative real numbers. Since the graph is undirected, the weights in both directions are the same, i.e., $w_{ij} = w_{ji}$. The weight coefficient is written on top of an edge, and it describes the similarity between two vertices which that edge is connecting. For example, low values of weight coefficient, i.e., values close to zero, indicate that two signal components are completely uncorrelated numbers. On the other hand, large weight coefficients, i.e., coefficients closer to one, indicate that two signal components are similar, i.e., their numerical values are very close.

After the signal is received, it's signal components are mapped to vertices and the signal is described by a graph.
If we take into consideration a graph depicted on the Figure 4.1., we can see that only a few signal components are taken into observation (components in the squares), i.e. components x_2, x_3 and x_7. Those components which are observed contain an additive noise due to the channel imperfections. We wish to recover all the other signal components of a graph signal from them.

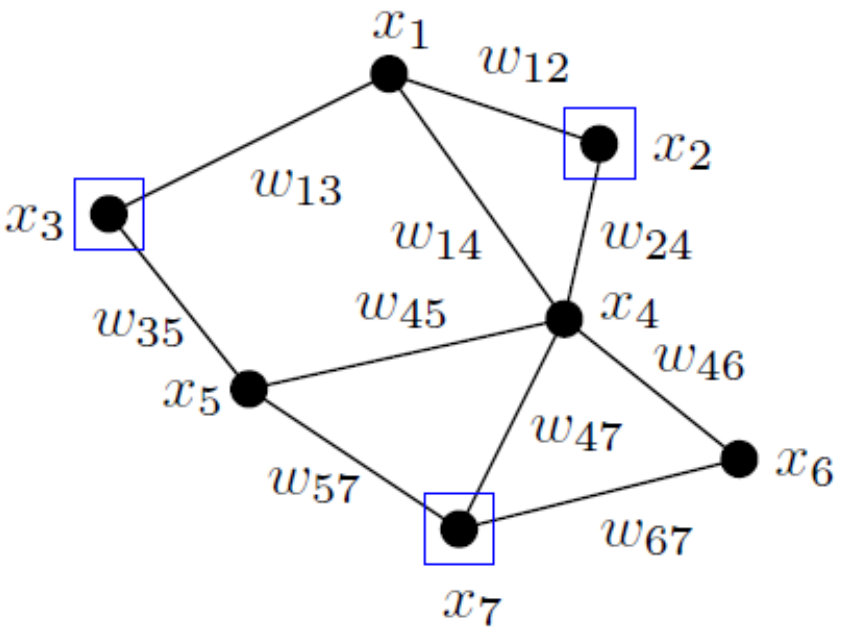


FIGURE 4.1: Undirected graph signal[3]

However, if we want to state the recovery problem, we need to introduce the sub-sampling matrix A. The sub-sampling matrix of the problem from the Figure 4.1. is defined as following:

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

The sub-sampling matrix A consists of ones on the places where the observed signal components are, and zeros on all the other places.
After constructing the sub-sampling matrix, the recovery problem can be formally defined:

$$y = Ax + n \tag{4.1}$$

where x represents the graph signal column vector, i.e. $x = x_j, j = 1, 2, ..., N$, n represents the noise column vector which is defined like an independent Gauss distributed variable, i.e. $n_k \sim \aleph(0, \sigma_n^2)$, and y represents a column vector o the measurements, i.e. $y = y_k, k = 1, 2, ..., M$.
The graph signal recovery problem is an inverse problem. Inverse problem is the mathematical process of predicting data based on some physical or mathematical model with a set of model parameters (and perhaps some other appropriate information, such as geometry, etc.)[5].

The ill-posed (inverse) problem can only be solved by exploiting the weight matrix, e.g. defining a regularization in an optimization problem[6]. In other words, we need to find a signal vector x which will reproduce the observations. After finding the signal, the weighting matrix will come to use in order to reproduce the rest of the signal components from the graph.
Since the weight in the graph is directly connected with similarities between two signal components, and since the weight is larger as the signal components connected are more similar, we can describe a problem with a help of difference operation between two signal components:

$$x_i - x_j \tag{4.2}$$

In other words, after we apply the operation of subtraction between two signal components, and square the result (since the operation can yield a negative number), the smaller number we get, the two signal components are more similar. Then we need to weight up the the difference with the belonging weights w_{ij} and add up the result for all the signal components.
To proceed further in defining a recovery problem, we need to introduce the measure of *smoothness* of a graph. The result which we get after weighting the squared differences is the measure of smoothness, we can formally write that measure as:

$$S_p(\hat{x}) = \frac{1}{p} \sum_{i=1}^{N} \left[\sum_{j=1, j \neq i}^{N} w_{ij} \mid \hat{x}_i - \hat{x}_j \mid^2 \right]^{\frac{p}{2}} \tag{4.3}$$

The equation (4.3) defines the measure of smoothness of a graph, and in the recovery problem, it can be introduced with the scaling factor λ'. The scaling factor λ' is defined in such a way that it can be seen as some kind of a *trade-off* between the smoothness of a graph and the accuracy of the graph reconstruction.
Now we can define the recovery problem as:

$$\hat{x} = argmin_{\hat{x}} \{ \mid\mid y - Ax \mid\mid_2^2 + \lambda' S_p(\hat{x}) \} \tag{4.4}$$

In the scope of this thesis, we will use the Tikhonov regularization method, which can be implemented for a factor p = 2 in the smoothness measure equation.

Tikhonov regularization Tikhonov regularization is a common approach while solving the ill-posed problems. Let us assume that there are two vectors, **x** and **b**. Let us also assume that the matrix A is given and we want to find out the values of the vector **x** from a given statement:

$$Ax = b \tag{4.5}$$

Way to do so is to apply the Least Squares (LS) linear regression which will lead us to a non unique solution. To avoid that happening we can set the problem in a bit different way, where the matrix A maps the vector **x** to the vector **b**. The least squares solution minimizes the difference of the squared residuals

$$\| Ax - b \|^2 \tag{4.6}$$

with $\| \bullet \|_2$ being the Euclidean norm.
In order to weight some solution more then others, the regularization coefficient can be introducet as the part of the equation.

$$\| Ax - b \|^2 + \| \Gamma x \|^2 \tag{4.7}$$

The regularization coefficient is in many cases the identity matrix which is giving more weight to the solution with a smaller norm. Also known as L_2 regularization[7].

Gauss-Seidel approach Gauss-Seidel approach is well known iterative method to solve a system of *n* linear equations with *x* being an unknown. Writing the problem statement formally:

$$Ax = b \tag{4.8}$$

The Gauss-Seidel approach is defined by the iteration:

$$L_* x^{k+1} = b - U x^k \tag{4.9}$$

Where $A = L_* + U$, i.e., A is decomposed into lower triangular[1] L_* and upper triangular[2] U component. And x^k is the *k*th iteration of *x* and x^{k+1} is the *k + 1* th iteration of *x*. Gauss-Seidel approach will be explained step by step in the section 4.3.

[1] square matrix in which all the entries above the main diagonal are zero
[2] square matrix in which all the entries below the main diagonal are zero

4.3 Graph signal recovery with Tikhonov regularization and Gauss-Seidel approach

Considering the problem setting from the equation (4.4) and applying the Tikhonov regularization setup, i.e., setting p = 2, the term transforms to:

$$\begin{aligned} L(\hat{x}) &= || \, y - A\hat{x} \, ||_2^2 + \lambda' S_2(\hat{x}) \\ &= \sum_{k=1}^{M} (y_k - (Ax)_k)^2 + \lambda \sum_{i=1}^{N} \sum_{j=1, j \neq i}^{N} w_{ij} (\hat{x}_i - \hat{x}_j)^2 \end{aligned} \tag{4.10}$$

Nex step is to take partial derivatives of the graph signal components.
Since the graph is undirected, i.e., weight coefficients are symmetric, after derivation, we find:

$$\frac{\partial L(\hat{x})}{\partial \hat{x}_l} = \begin{cases} -2(y_k - \hat{x}_l) + 4\lambda \sum_{j=1, j \neq l}^{N} w_{lj} (\hat{x}_l - \hat{x}_j), & \text{if } A_{kl} = 1 \text{ for some k.} \\ 4\lambda \sum_{j=1, j \neq l}^{N} w_{lj} (\hat{x}_l - \hat{x}_j), & \text{if } A_{kl} = 0 \text{ for all k.} \end{cases} \tag{4.11}$$

setting the equation to zero (in order to find an extremum) for each l = 1, 2, ..., N, we get:

$$y_k = \hat{x}_l + 2\lambda \hat{x}_l \sum_{j=1, j \neq l}^{N} w_{lj} - 2\lambda \sum_{j=1, j \neq l}^{N} w_{lj} \hat{x}_j \quad \text{if } A_{kl} = 1 \quad \text{for some } k \tag{4.12}$$

$$0 = \hat{x}_l \sum_{j=1, j \neq l}^{N} w_{lj} - \sum_{j=1, j \neq l}^{N} w_{lj} \hat{x}_j \quad \text{assuming } \lambda > 0 \quad \text{if } A_{kl} = 0 \quad \text{for all } k \tag{4.13}$$

Grouping with graph signal component $\hat{x}_l$ in the (4.7) equation, we get the system of equations with graph signal component $\hat{x}_l$ being the unknown component. The system is defined as:

$$y_k = \hat{x}_l (1 + 2\lambda \sum_{j=1, j \neq l}^{N} w_{lj}) - 2\lambda \sum_{j=1, j \neq l}^{N} w_{lj} \hat{x}_j \quad \text{if } A_{kl} = 1 \quad \text{for some } k \tag{4.14}$$

$$0 = \hat{x}_l \sum_{j=1, j\neq l}^{N} w_{lj} - \sum_{j=1, j\neq l}^{N} w_{lj}\hat{x}_j \quad \text{assuming } \lambda > 0 \quad \text{if } A_{kl} = 0 \quad \text{for all } k \tag{4.15}$$

General measurement matrix When a problem consists of a general measurement matrix, i.e., a matrix which can have negative entries and non zero components, the system of linear equations will take more of a general form:

$$\sum_{k=1}^{M} A_{kl} y_k = \sum_{j=1, j\neq l}^{N} \hat{x}_j \left(\sum_{k=1}^{M} A_{kj} A_{kl} - 2\lambda w_{lj} \right) + \hat{x}_l \left(\sum_{k'=1}^{M} A_{k'l}^2 + \sum_{j'=1, j'\neq l}^{N} 2\lambda w_{lj'} \right) \tag{4.16}$$

For the purpose of easier calculation and description, we can reformulate the (4.16) equation as:

$$b_l = \sum_{j=1, j\neq l}^{N} \hat{x}_j c_{lj} + \hat{x}_l c_{ll} \tag{4.17}$$

This statement can easily be turned back into (4.16) statement by using the sub-sampling matrix A with a single 1 in each row and column and putting 0 to all other places. In that case, $A_{kj}A_{kl} = 0 if j \neq l$ which leads us to the (4.16) equation.

Gauss-Seidel approach The best way of solving the problems stated by (4.16) is by the iterative approach to it. Technically speaking, it could be solved by the standard linear algebra techniques, but the question of complexity poses a big problem. In this thesis we will use the Gauss-Seidel approach which was already mentioned in the section 4.2. Here, the technique's steps will be described more detailed.
The iterative method assumes solving a problem statement in iterations, i.e. first solve it for $\hat{x}$ and then use it to solve the statement for the next step of the iteration i.e.$\hat{x}_l^{i+1}$.

In other words, formally written, step by step[3]

1. Compute b_l and c_{lj} for $j, l = 1, 2, ..., N$.
2. Set iteration counter to $i = 0$ and initialize solution vector $\hat{x}_l = 0 \forall l = 1, 2, ..., N$.Moreover set small threshold $1 \gg \varepsilon > 0$
3. If the inequality

$$\sum_{l=1}^{N} \left(b_l - \sum_{j=1}^{N} c_{lj} \hat{x}_j^{(i)} \right)^2 < \epsilon \tag{4.18}$$

   is fulfilled, stop iterating and the solution will be given by $\hat{x}_l^{(i)}, l = 1, 2, ..., N$.
   Otherwise, go to step 4).
4. Compute:
   - for $l = 1$:

$$\hat{x}_l^{(i+1)} = \left(b_l - \sum_{j=l+1}^{N} c_{lj} \hat{x}_j^{(i)} \right) \Big/ c_{ll} \tag{4.19}$$

   - for $l = 2, 3, ..., N - 1$:

$$\hat{x}_l^{(i+1)} = \left(b_l - \sum_{j=1}^{l-1} c_{lj} \hat{x}_j^{(i+1)} - \sum_{j=l+1}^{N} c_{lj} \hat{x}_j^{(i)} \right) \Big/ c_{ll} \tag{4.20}$$

   - for $l = N$:

$$\hat{x}_l^{(i+1)} = \left(b_l - \sum_{j=1}^{l-l} c_{lj} \hat{x}_j^{(i+1)} \right) \Big/ c_{ll} \tag{4.21}$$

5. Set $i = i + 1$ and go to step 3).

4.4 Graph signal sampling

One of the key issues in the graph signal analysis is how to sample a graph signal with a given (weighting) sub-sampling matrix using the least number of signal components sampled. During this section and next chapter, Tikhonov regularization with Gauss-Seidel approach used for the recovery of a graph signal was assumed.
In order to describe two different sampling techniques used, the graph signal on the Figure 4.2 is used as an example.

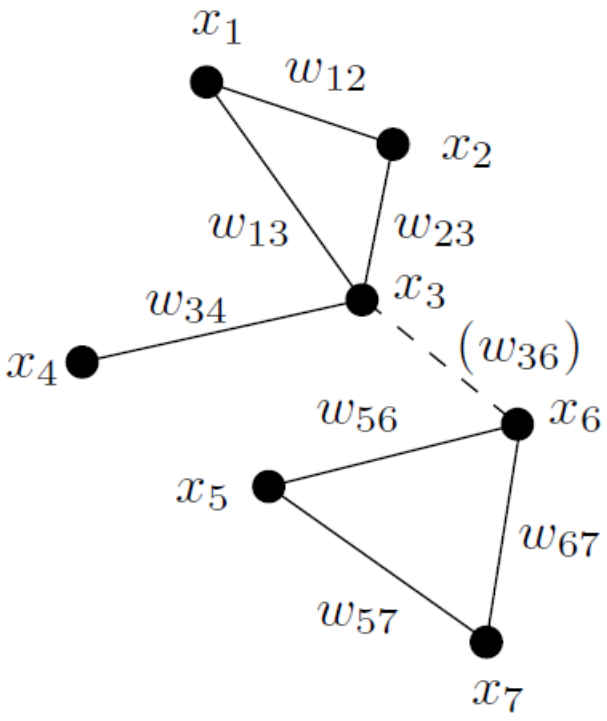


FIGURE 4.2: Graph signal[3]

Weighting matrix for this graph is stated as:

$$W = \begin{bmatrix} 0 & w_{12} & w_{13} & 0 & 0 & 0 & 0 \\ w_{12} & 0 & w_{23} & 0 & 0 & 0 & 0 \\ w_{13} & w_{23} & 0 & w_{34} & 0 & (w_{36}) & 0 \\ 0 & 0 & w_{34} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{56} & w_{57} \\ 0 & 0 & (w_{36}) & 0 & w_{56} & 0 & w_{67} \\ 0 & 0 & 0 & 0 & w_{57} & w_{67} & 0 \end{bmatrix}$$

At this point, for the sake of further usage, we can define a degree matrix D. The D matrix is a diagonal matrix whose d_jth diagonal element is a sum of the weights of signal components which are directly connected to the signal component j over non-zero entries in the weighting matrix W. For the Figure 4.2. the diagonal matrix is:

$$D = \begin{bmatrix} w_{12}+w_{13} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & w_{12}+w_{23} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & w_{13}+w_{23}+w_{34}(+w_{36}) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & w_{34} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{56}+w_{57} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & w_{56}+w_{67}(+w_{36}) & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & w_{57}+w_{67} \end{bmatrix}$$

Therefore, the Laplacian graph is defined as[6]:

$$\alpha = D - W \tag{4.22}$$

..so it reads as:

$$\alpha = \begin{bmatrix} w_{12}+w_{13} & -w_{12} & -w_{13} & 0 & 0 & 0 & 0 \\ -w_{12} & w_{12}+w_{23} & -w_{23} & 0 & 0 & 0 & 0 \\ -w_{13} & -w_{23} & w_{13}+w_{23}+w_{34}(+w_{36}) & -w_{34} & 0 & (-w_{36}) & 0 \\ 0 & 0 & -w_{34} & w_{34} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & w_{56}+w_{57} & -w_{56} & -w_{57} \\ 0 & 0 & (-w_{36}) & 0 & -w_{56} & w_{56}+w_{67}(+w_{36}) & -w_{67} \\ 0 & 0 & 0 & 0 & -w_{57} & -w_{67} & w_{57}+w_{67} \end{bmatrix}$$

4.4.1 Ad-hoc scheme

In the sampling process, we want to find out which signal components x_j can be sampled in order to make a full reconstruction of a graph. The weight matrix, which describes the similarity weights between two signal components, is one of the key features of solving the reconstruction problem. The idea of such leads us to the definition of the following matrix used for ad-hoc sampling:

$$C = \begin{bmatrix} 1 & w_{12} & w_{13} & 0 & 0 & 0 & 0 \\ w_{12} & 1 & w_{23} & 0 & 0 & 0 & 0 \\ w_{13} & w_{23} & 1 & w_{34} & 0 & (w_{36}) & 0 \\ 0 & 0 & w_{34} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & w_{56} & w_{57} \\ 0 & 0 & (w_{36}) & 0 & w_{56} & 1 & w_{67} \\ 0 & 0 & 0 & 0 & w_{57} & w_{67} & 1 \end{bmatrix}$$

From all the signal components given, we randomly choose one component and insert a single 1 on the desired place:

$$X = \begin{bmatrix} 0 \\ 0 \\ \dots \\ 1 \\ 0 \\ \dots \\ 0 \end{bmatrix}$$

Next step is to multiply the matrix *C* and vector *X* and look for the non-zero components in the given result Y_1:

$$Y^1 = CX \tag{4.23}$$

Before we proceed to the iterative multiplication algorithm, the scaled weighting matrix needs to be defined. The reason behind it is that if the multiplication like in the equation (4.23) is proceeded further and further, the output values are getting larger and larger which is in not convenient for the results which are expected. Therefore, the weighting matrix is scaled, and the *new* matrix is used further on. The scaled weighting matrix formally written reads:

$$W_tmp = \frac{W}{max(\sum_{i=1}^{max(rows)} \sum_{j=1}^{max(columns)} w_{ij} + 0.1)} \tag{4.24}$$

The process can then be proceeded for $k = 2, 3, ..., N$, with multiplying with the scaled weighting matrix W_{tmp} :

$$Y^{(k)} = W_{tmp}W_{tmp}...W_{tmp}Y^1 = W_{tmp}^{k-1}CX \tag{4.25}$$

The logic behind this procedure is that if there is some component observed, different from a sampling component, i.e., $j' \neq j$, and that component takes a non-zero value after *k* iterations, then a signal component $x_{j'}$ can be reconstructed since signal component x_j is sampled. For only one step, i.e., $k = 1$, and non-zero value, we conclude that the two components are directly connected.
The steps of an *ad-hoc greedy approach* are given in detail below[3]:

1. Set the number *M* of graph signal components to be sampled.
2. Initially pick the component *j* for which the diagonal entry in the degree matrix $\mathbf{D}^3$ takes the largest value. Add the component to the sampling set *S* = *j*.
3. Set the signal components x_S with indicies in the sampling set *S* to one and compute :

   $$Y^1 = CX \tag{4.26}$$

   and then:

   $$Y^{(k)} = W_{tmp}^{k-1}Y^1 \tag{4.27}$$

   for $k = 2, 3, 4, ...$ Stop to iterate, when the iterations steps of computing those components in $Y^{(k)}$ that are not in the set *S* exceeds a given threshold value .
4. Pick the index *j'* for which the component $y_{j'}^{(k)}$ takes the smallest value. If several components in $Y^{(k)}$ are zero, pick one of them randomly. Add the chosen component to the sampling set *S*, i.e., $S = S \cup j'$.
5. If $M =\mid S \mid$, stop. Otherwise, go to Step 3.

The matrix exponentiation could be simplified by using the eigenvalue decomposition, but only for a high degree exponents. In that case, it would reduce the complexity compared to the step by step multiplication with itself.

[3] diagonal matrix whose *j*th diagonal element is equal to the sum of the weights of all the signal components connected to the signal component *j* via non-zero entries in the weight matrix **W**[3]

4.4.2 Random sampling

The other scheme in the scope of this thesis, which is compared to ad-hoc greedy scheme, is the random sampling scheme.
Here, the signal components which are observed are taken completely randomly.
Performance comparisons between two different schemes are stated throughout the Chapter 5.

Chapter 5

Graph signal sampling implementation

5.1 Introduction

In this chapter, the two implementations of graph signal sampling and their performances will be shown. Important to note is that a random *incoming* signal is assumed and Tikhonov regularization with Gauss-Seidel approach as a recovery scheme is used.
In the next sections, implementations of signal generation, signal recovery, ad-hoc sampling and random sampling will be shown. The Matlab code of the implementations can be found in the Appendices. Since we can not use the completely random signals to test the sampling and recovery, we used the used the manually generated signal with a random weight matrix.

5.2 Generation of a graph signal and Tikhonov regularization

In order to test the ad-hoc and random sampling implementations, the graph signal needed to be generated. As it could have been seen before, recovery method used in this thesis (Tikhonov regularization with Gauss-Seidel approach), does not yield good results if a completely random signal is used, since the signal is not smooth enough. Therefore, we imagine that some random signal has came and got sampled (by some of the algorithms explained later). Later on, with a help of a weight matrix, which is defined as a random, symmetric, weight matrix, we generate a graph signal, i.e., we map the signal components into the vertices of a graph.
In the implementation code of a graph signal generator (See Appendix B), a few components which are of a big importance and can be adjusted were used. Their adjustments yield different kinds of results which are all of significance regarding the sampling and recovery problem. Those parameters are:

- Graph signal dimension N - the number which describes the dimension of a signal. For the higher graph signal dimension,

more components of an *incoming* signal will be mapped into a graph signal. The parameter is of the high importance for the random weight matrix generation.

- Number of sampled components M - describes a number of samples taken from the signal. The number of samples are very significant for the Tikhonov regularization problem, i.e., for a higher number of samples, recovery should be better consistent graph signal.

- Standard deviation of noise σ - noise which is added to the graph signal to make it less consistent with the recovery process.

In the *GenerateGraphSignal.m* file, the algorithm of generation and the Tikhonov recovery algorithm can be found(Appendix B). Three main parameters described above were used for the generation and recovery.
In the process of generation, the consistent graph signal was generated for a given weight matrix.
The parameter M, the number of sampled components, set the number of sampled components from the graph, which were used to recover the graph signal. The recovered version is supposed to be consistent with the weight matrix in the sense that the signal model (which is not defined by the weight matrix) is implicitly posed by Tikhonov regularization used in the generation of the graph signal.
The last parameter, sigma σ, represents the standard deviation of the variance of the noise that is added to the recovered graph signal components after the process of recovery. The purpose of it is to make the graph signal less consistent with Tikhonov regularization.
On the Figure 5.1 generation process can be followed.

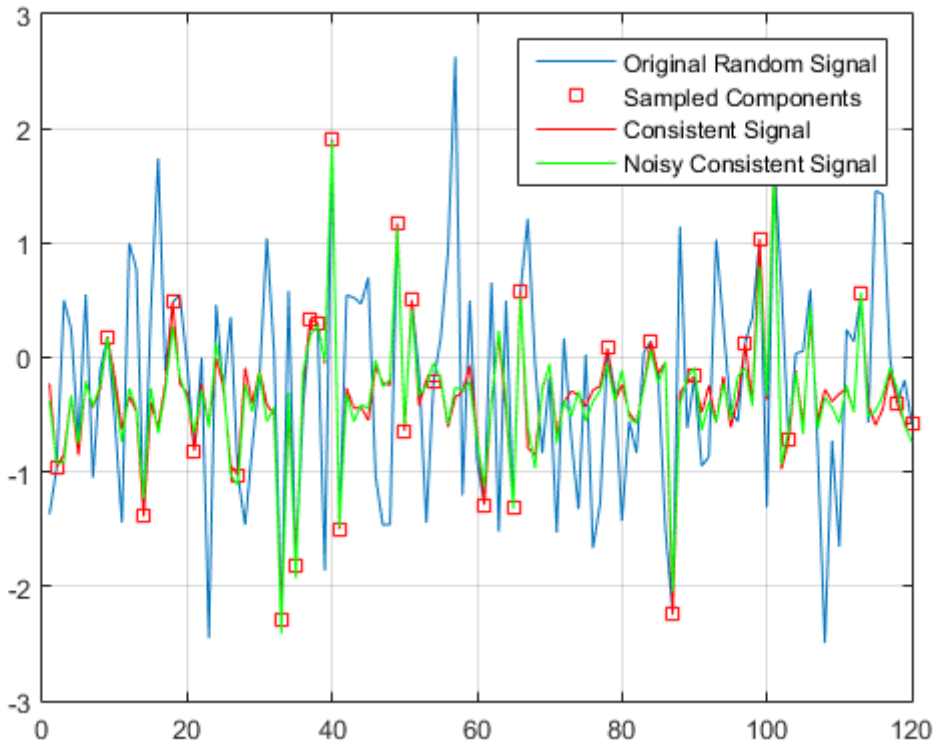


FIGURE 5.1: Graph signal generation and recovery

The blue colored signal represents the random incoming signal, the signal gets sampled by the points which are marked on the Figure with red points. Consistent signal recovered is the red colored signal, and since there is the noise added after the recovery, noisy signal is shown by the green color. The standard deviation of variance of noise, σ is in this example set up on 0.1.
Just to compare, on the Figure 5.2. the graph signal generation and recovery is shown with the σ set up on 1. As it is shown, the standard deviation of the variance of noise has a major influence on the final version of a signal, more will be shown in the later sections.

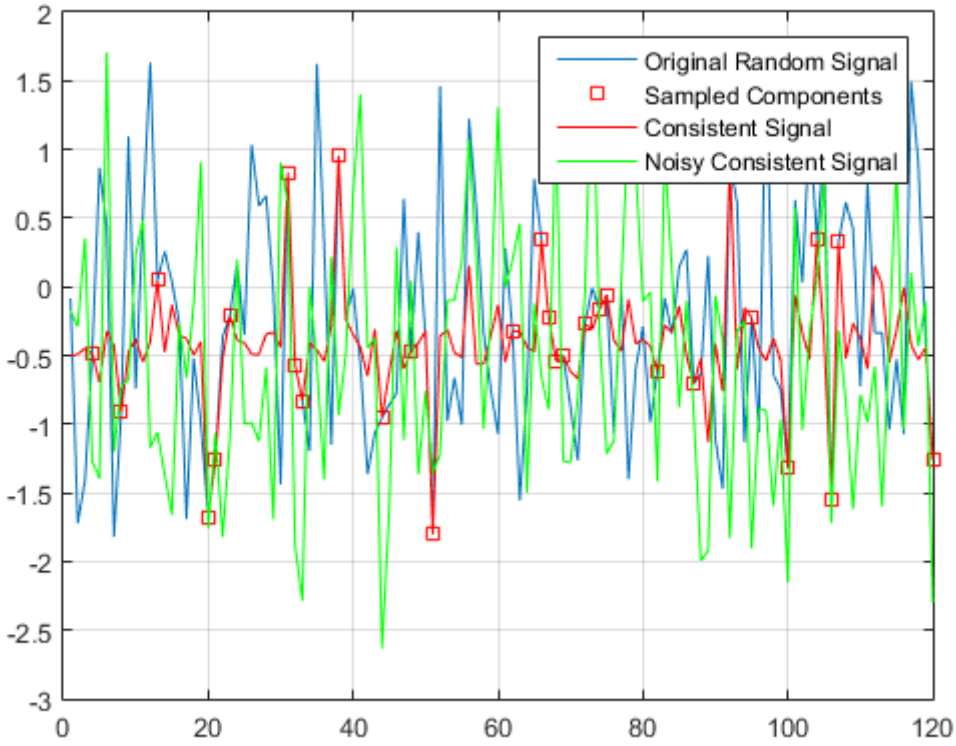


FIGURE 5.2: Graph signal generation and recovery with more noise

5.3 Implementation of sampling methods

As mentioned before in the Chapter 4, throughout this thesis two kinds of samplings were introduced and their performances compared.

5.3.1 Ad-hoc sampling and Random sampling

As described in the step by step process from the Chapter 4, ad-hoc sampling is implemented and the code is shown in the Appendix C. The function *adhocsampling.m* as parameters takes the weighting matrix, number of samples and an ϵ parameter used for the iteration termination.
As opposed to ad-hoc approach, which had some *method* of choosing the samples of a signal, the random sampling is implemented in a way that the sampled points are chosen randomly. The code of the random sampling method can be seen in the Appendix D. The function *randomsampling.m* receives two parameters, weighting matrix and the number of samples. Random sampling chooses randomly M number of samples and applies to the signal.
In the next section, the two methods will be compared regarding to

the change of various parameters.
As expected, in all the comparisons the ad-hoc sampling technique will, naturally yield better results. On the Figure 5.3. the signal flow of the random *incoming* signal, ad-hoc sampled and recovered signal and random sampled and recovered signal are shown.

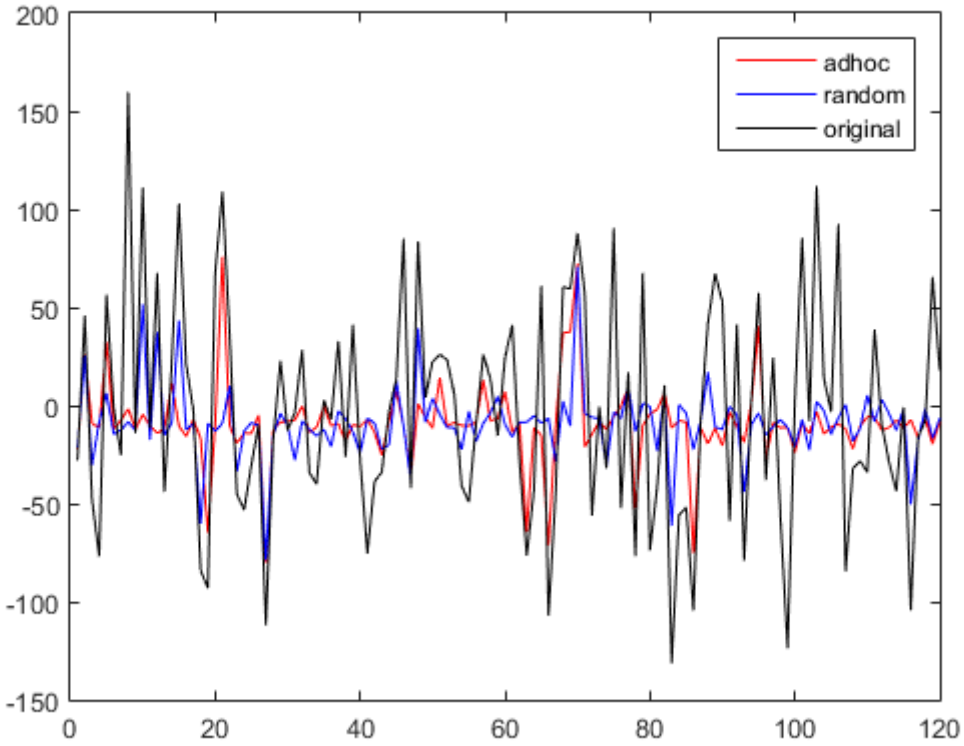


FIGURE 5.3: Random graph signal, ad-hoc sampled signal and random sampled signal

5.4 Ad-hoc sampling and random sampling performance comparison

In this section, performances of two different sampling methods will be compared. The base of comparisons will be the variation of different parameters. Here, we will vary the most important parameters, number of samples M, scaling factor λ, standard deviation of variance of the noise σ and the weighting matrix.

5.4.1 Performance comparison when number of samples M are varied

If the number of samples are varied, we can compare and plot the overall performances of ad-hoc sampling and random sampling algorithms.
As expected, for a smaller number of samples, ad-hoc sampling algorithm performs better. Reason why is that if there is a small number of samples which is supposed to be chosen, the ad-hoc sampling algorithm chooses them more carefully. On the other hand, for a really high number of samples to be chosen, both sampling methods perform similar. The reason behind is that if the number of samples *allowed* to be taken is somewhat close to the number which describes a dimension of a graph, then even randomly chosen points will yield pretty good results.
In the table 5.1 the performances for different values of numbers of samples *M* are shown.
Two measurements were made. The setup for the first one was set on a number of samples varying from 20 to 60 with an intermediate step of 5. The second setup was set on a number f samples varying from 20 to 110 with an intermediate step of 5. In both setups, the measurements were made with 300 iterations.

Number of samples M	Ad-hoc sampling MSE	Random Sampling MSE	|adhoc_sampling MSE-random_sampling MSE|
30	80.8	84.24	3.49
50	66.86	69.49	2.63
60	59.85	62.84	2.99
80	47.4	48.54	1.14
100	34.1	34.91	0.81

TABLE 5.1: Ad-hoc sampling vs. Random sampling performance comparison, number of samples varied from 20 to 100

On the Figure 5.4 the results from the setup 1 can be found. On the Figure 5.5 the results from the setup 2 can be found. As we can observe, in both cases, the ad-hoc sampling performs better. One more trend can be observable, with the fewer number of samples used, the difference between the MSE of the two sampling techniques gets bigger and bigger, naturally to the Ad-hoc sampling advantage.

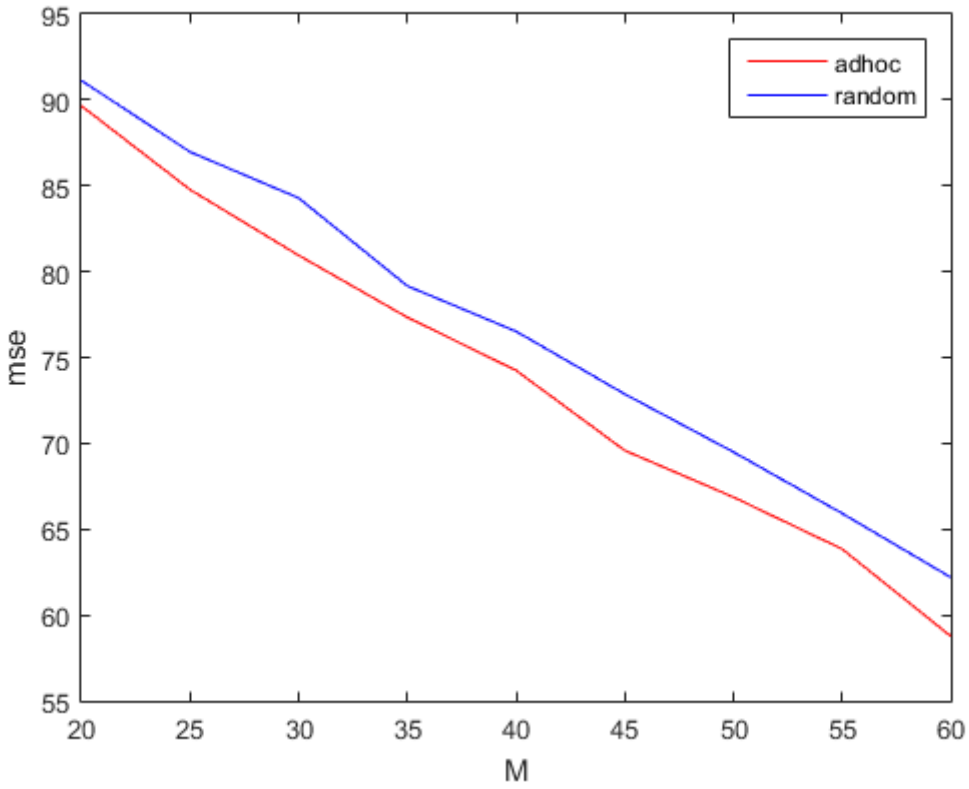


FIGURE 5.4: Ad-hoc sampling vs. Random sampling performance comparison while number of samples M= 20..60

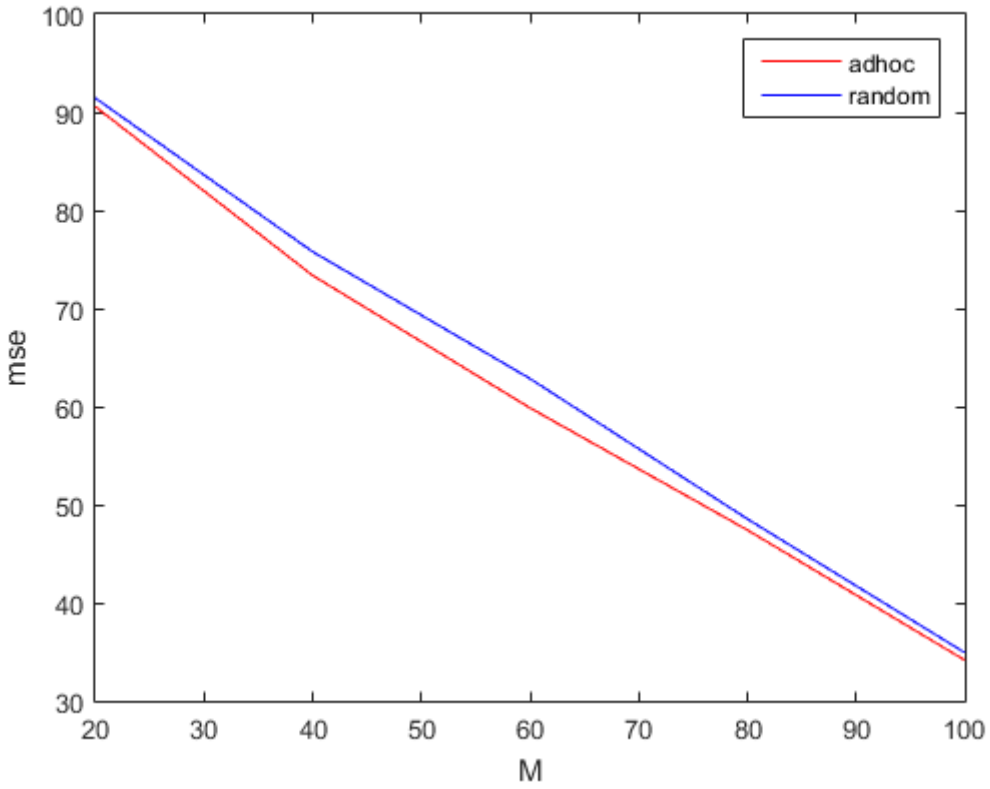


FIGURE 5.5: Ad-hoc sampling vs. Random sampling performance comparison while number of samples M = 20...110

5.4.2 Performance comparison when a scaling factor λ is varied

As mentioned before in the Chapter 4, scaling factor can be seen as a *trade-off* between the smoothness of a graph and the accuracy of the graph reconstruction. As it can be seen in the equation (4.4):

$$\hat{x} = argmin_{\hat{x}} \{ \mid\mid y - Ax \mid\mid_2^2 + \lambda' S_p(\hat{x}) \} \tag{5.1}$$

The larger the scaling factor λ' is, the smoother the reconstructed graph is.
On the other hand, scaling factor as well influences the performances of ad-hoc sampling scheme and random sampling scheme.
In the table 5.2 the performances of Ad-hoc sampling scheme and Random sampling scheme are shown when the parameter λ is varied. The measurements are made with the λ parameter varying from 0.1 to 3 in 20 steps through 300 iterations.

Scaling parameter λ	Ad-hoc sampling MSE	Random Sampling MSE	| adhoc_sampling MSE-random_sampling MSE |
0.1	81.32	82.97	1.65
0.55	90.89	94.12	3.23
1.02	93.72	96.02	2.3
1.5	96.11	98.68	2.57
2.23	98.46	100.4	1.94
3	98.52	9.81	1.29

TABLE 5.2: Ad-hoc sampling vs. Random sampling performance comparison, λ varied from 0.1 to 3

From the measurements it is observable that Ad-hoc sampling method performs better than Random sampling method. The scaling factor is a *trade-off* between smoothness and accuracy of the reconstruction. If the scaling factor becomes larger, the signal becomes smoother which is necessary for the reconstruction. On the other hand, for a smaller scaling factor, the reconstruction should become more accurate. In theory, it can not be concluded whether the recovery would work better if the scaling factor is really large or really small, but the measurements have shown that a smaller scaling factor suggests a better MSE.
In the same time, it is possible to conclude that Ad-hoc sampling method performs again better compared to Random sampling method. The performance comparison can be seen on the Figure 5.6.

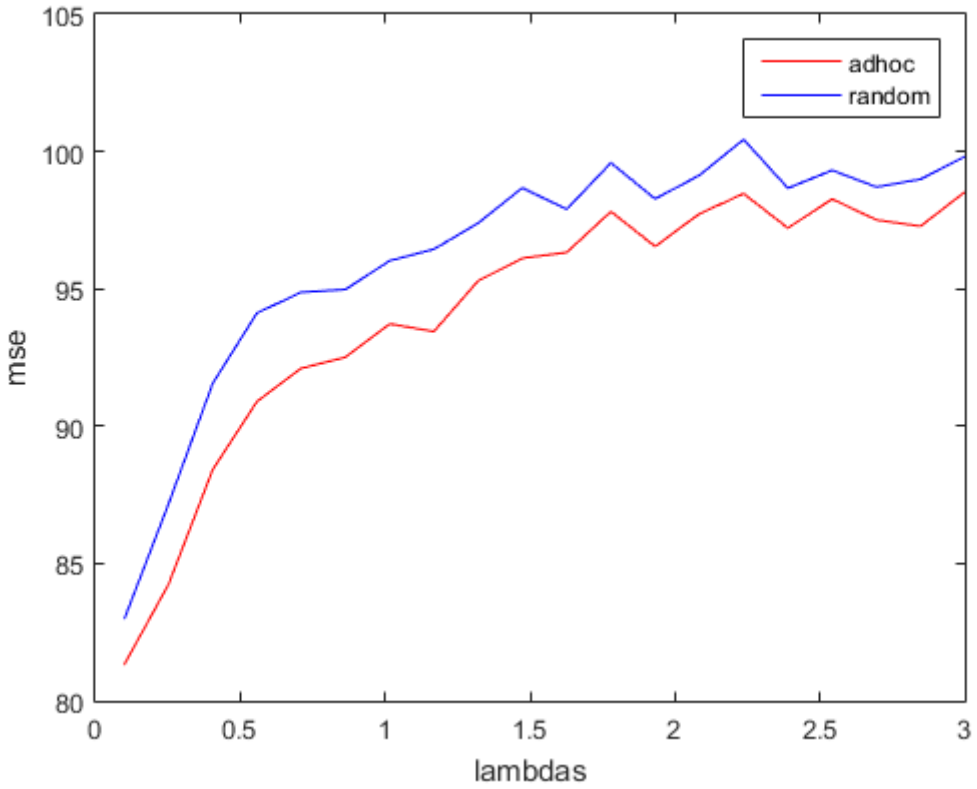


FIGURE 5.6: Ad-hoc sampling vs. Random sampling performance comparison while the scaling factor λ = 0.1...3

5.4.3 Performance comparison when standard deviation of variance of the noise σ is varied

Standard deviation of noise σ is added to the graph signal after the recovery with Tikhonov regularization in order to make the graph signal generated, less consistent with the Tikhonov regularization in a controlled way.
In the table 5.3 results from the measurement are shown. The measurement took 10 samples of σ in the range from 1 to 50 with overall 300 iterations.

σ	Ad-hoc sampling MSE	Random Sampling MSE	|adhoc_sampling MSE-random_sampling MSE|
1	1.027	1.052	0.025
17.3	242.3	247.7	5.4
28.22	675.4	653.9	21.5
39.11	1305	1265	40
50	2012	2080	68

TABLE 5.3: Ad-hoc sampling vs. Random sampling performance comparison, σ varied from 1 to 50

From the results in the table, it can be seen that for a smaller σ, i.e. less noise, the two methods are much closer regarding the performance. With more and more noise added, it becomes clear that ad-hoc sampling method yields better results. However, even though the ad-hoc sampling method yields better results, the minimum squared error becomes big enough so the recovery process is not completely reliable.
The performances of ad-hoc sampling method and random sampling method can be seen on the Figure 5.7.

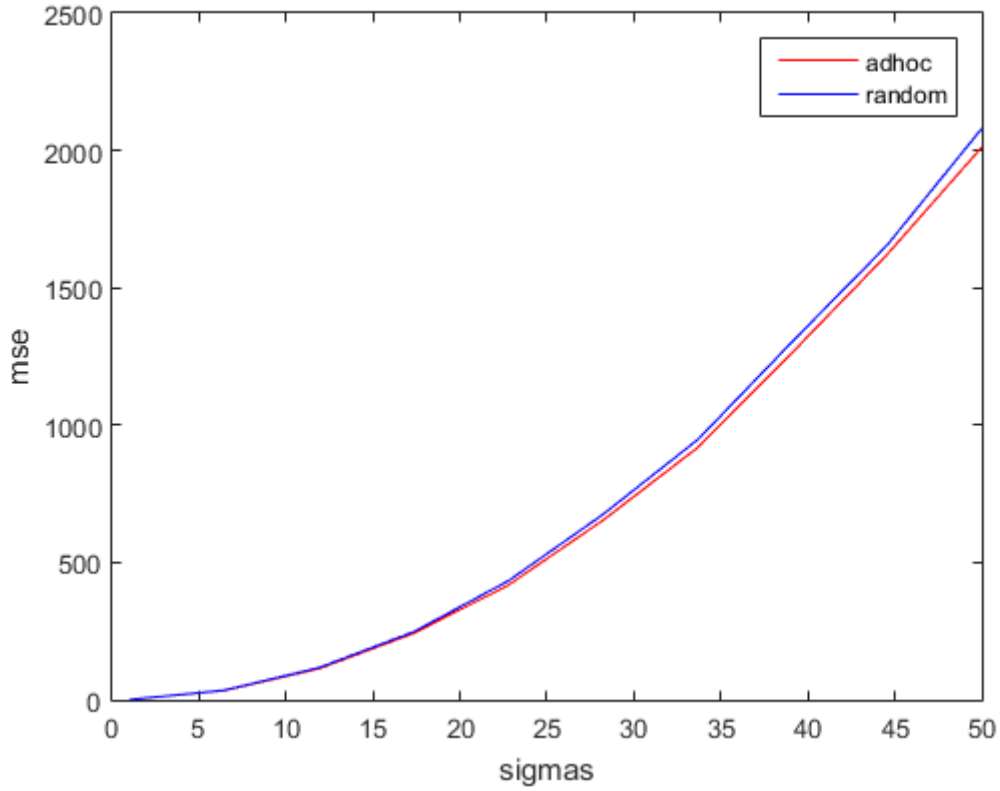


FIGURE 5.7: Ad-hoc sampling vs. Random sampling performance comparison while the σ factor varied from 1 to 50

5.4.4 Performance comparison when weighting matrix is varied

As mentioned before, weighting matrix is chosen randomly. That fact gives us a freedom to choose and modify the weighting matrix in a way that we can see what type of the weighting matrix, i.e. generated graph signal, suits the best for the sampling schemes.
The random symmetric weighting matrix is designed on a way that we choose a random number of graph signal components connected. In other words, each vertex of a graph can be connected with maximum of $(N-1)$ of other vertices. With a short look on the code

inter-connectivity parameter	Ad-hoc sampling MSE	Random Sampling MSE	| Ad-hoc sampling MSE- Random sampling MSE |
0.05	82.72	84.59	1.87
0.2	90.4	88.35	2.05
0.3	92.31	93.73	1.42
0.45	92.42	94.19	1.77
0.5	95.22	96.6	1.38

TABLE 5.4: Ad-hoc sampling vs. Random sampling performance comparison when an inter-connectivity factor is varied

in the Appendix A., it can be seen that with the usage of an inter-connectivity parameter (var. *fac*) we can dictate how many components of a graph signal will be actually connected together. Therefore, a measurement over varying the fractional variable was made. In the test, standard value of $N = 120$ as a graph signal dimension was used, 30 samples, scaling factor $\lambda = 0.1$ and all of that ran over 100 iterations. The inter-connectivity parameter was varied over a range of values from 0.06 to 0.5.
The results of a measurement can be seen in the table 5.4 and on the Figure 5.8.

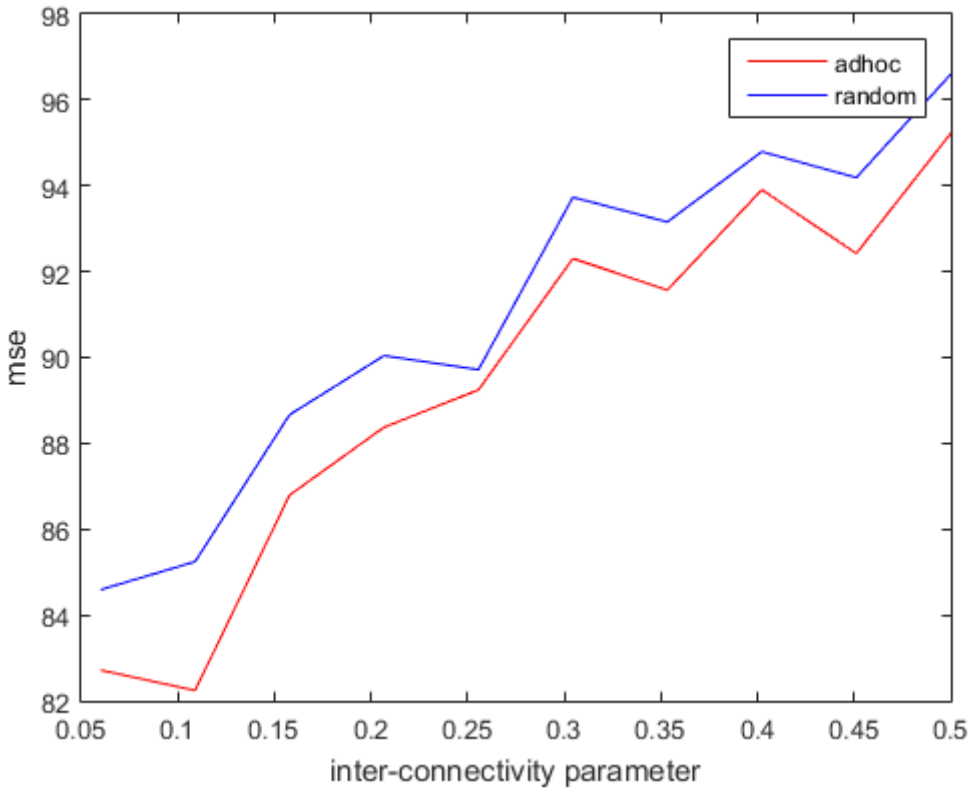


FIGURE 5.8: Ad-hoc sampling vs. Random sampling performance comparison while the inter-connectivity parameter varied

As expected, ad-hoc sampling technique performs better than random sampling technique. On the Figure 5.8 and in the Table 5.4, the growing MSE trend can be spotted. The reason behind is that, if the inter-connectivity parameter is a large value, the graph signal has more signal components connected, that is, for an inter-connectivity parameter of 1, all the signal components are connected and the complete graph is established. Furthermore, more the connections, i.e., edges, a graph signal has, the *harder* it is to sample it correctly. Therefore, the growing MSE trend can be seen.
As we can see, the weighting matrix, which was chosen randomly in this thesis, is also of a great importance in the overall sampling performance.

Chapter 6

Conclusion

The technology industry is a fast growing industry with more and more users each day who ask for more and more services. From the perspective of a service dealer, each dealer wants to be the best, to have the highest number of users, and to have a strong research team who make the new approaches of solving the already existing problems possible.

One of those new approaches is the approach of solving problems with a help of graph signals. Graph signal recovery and sampling problem is shown to be a very good technique of dealing with today's signal processing problems. It is a simple, usable and very elegant choice.

As the reader was able to mark during this thesis is that the biggest *obstacle* during the whole process is the signal recovery technique choice and the sampling technique choice.

In this thesis the signal recovery technique choice was set on the Tikhonov regularization technique with Gauss-Seidel iterative approach. But the sampling techniques were freely implemented and compared.

In all of the tests which were done, the ad-hoc sampling implementation performed better compared to random sampling technique. All the results were expected as they are.

The future developments on the field of graph signal sampling should be done to make the ad-hoc approach even better, because with a good sampling technique, the recovery process is more accurate, which is crucial for the usage of a whole.
In the end, graph signals techniques will be the one of the main future techniques of signal processing.

Bibliography

[1] John Adrian Bondy and Uppaluri Siva Ramachandra Murty. *Graph theory with applications*. Vol. 290. Citeseer, 1976.

[2] Siheng Chen et al. "Discrete signal processing on graphs: Sampling theory". In: *IEEE Transactions on Signal Processing* 63.24 (2015), pp. 6510–6523.

[3] Norbert Görtz. *Iterative Graph-Signal Recovery*. Tech. rep. TU Wien, 2016.

[4] Rui Vilela Mendes. "Signal processing on graphs". URL: http://label2.ist.utl.pt/vilela/Cursos/Signal_Proc_Net_SL.pdf. 2014.

[5] Randall M. Richardson and George Zandt. *Inverse Problems in Geophysics GEOS 567*. 2003.

[6] David I Shuman et al. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains". In: *IEEE Signal Processing Magazine* 30.3 (2013), pp. 83–98.

[7] Andrew Y Ng. "Feature selection, L 1 vs. L 2 regularization, and rotational invariance". In: *Proceedings of the twenty-first international conference on Machine learning*. ACM. 2004, p. 78.

[8] Aliaksei Sandryhaila and José MF Moura. "Discrete signal processing on graphs". In: *IEEE transactions on signal processing* 61.7 (2013), pp. 1644–1656.

[9] Mario Osvin Pavčević. "Uvod u teoriju grafova". In: *Element, Zagreb* (2006).

[10] C Vasudev. *Graph theory with applications*. New Age International, 2006.

[11] Dragan Stevanovic, Marko Miloševic, and Vladimir Baltic. "Diskretna matematika". In: *Zbirka rešenih zadataka, DMS, Beograd* (2004).

[12] Reinhard Diestel. "Graph theory. 1997". In: *Grad. Texts in Math* (1997).

[13] Gita Babazadeh Eslamlou et al. "Graph signal recovery from incomplete and noisy information using approximate message passing". In: *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2016, pp. 6170–6174.

[14] Gene H Golub, Per Christian Hansen, and Dianne P O'Leary. "Tikhonov regularization and total least squares". In: *SIAM Journal on Matrix Analysis and Applications* 21.1 (1999), pp. 185–194.

[15] Shinji Umeyama. "An eigendecomposition approach to weighted graph matching problems". In: *IEEE transactions on pattern analysis and machine intelligence* 10.5 (1988), pp. 695–703.

[16] F.R.K. Chung. *Spectral Graph Theory*. ams, 1997.

[17] Aliaksei Sandryhaila, Jelena Kovacevic, and Markus Puschel. "Algebraic Signal Processing Theory: 1-D Nearest Neighbor Models". In: *IEEE Transactions on Signal Processing* 60.5 (2012), pp. 2247–2259.

[18] "Big Data Analysis with Signal Processing on Graphs: Representation and processing of massive data sets with irregular structure". In: *IEEE Signal Process. Mag.* 31.5 (2014), pp. 80–90.

[19] Akshay Gadde and Antonio Ortega. "A Probabilistic Interpretation of Sampling Theory of Graph Signals". In: *CoRR* abs/1503.06629 (2015). URL: http://arxiv.org/abs/1503.06629.

[20] A. Singer. "From graph to manifold Laplacian: The convergence rate". In: *Applied and Computational Harmonic Analysis* 21.1 (2006), pp. 128–134. ISSN: 1063-5203. DOI: http://dx.doi.org/10.1016/j.acha.2006.03.004. URL: http://www.sciencedirect.com/science/article/pii/S1063520306000510.

[21] Andrej Nikolaevich Tikhonov and Vasiliy Yakovlevich Arsenin. *Solutions of ill-posed problems*. Winston, 1977.

[22] Michael I Jordan. "Graphical models". In: *Statistical Science* (2004), pp. 140–155.

[23] Michael I Jordan et al. "Major advances and emerging developments of graphical models [from the guest editors]". In: *IEEE Signal Processing Magazine* 27.6 (2010), pp. 17–138.

[24] Benjamin A Miller, Nadya T Bliss, and Patrick J Wolfe. "Toward signal processing theory for graphs and non-Euclidean data". In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE. 2010, pp. 5414–5417.

[25] Norbert Goertz et al. "Iterative recovery of dense signals from incomplete measurements". In: *IEEE Signal Processing Letters* 21.9 (2014), pp. 1059–1063.

[26] Narsingh Deo. *Graph theory with applications to engineering and computer science*. Courier Dover Publications, 2016.

[27] Douglas Brent West et al. *Introduction to graph theory*. Vol. 2. Prentice hall Upper Saddle River, 2001.

[28] Ingrid Daubechies, Michel Defrise, and Christine De Mol. "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint". In: *Communications on pure and applied mathematics* 57.11 (2004), pp. 1413–1457.

[29] Aamir Anis, Akshay Gadde, and Antonio Ortega. "Towards a sampling theorem for signals on arbitrary graphs". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2014, pp. 3864–3868.

[30] Michael Unser. "Sampling-50 years after Shannon". In: *Proceedings of the IEEE* 88.4 (2000), pp. 569–587.

[31] Gita Babazadeh Eslamlou, Alexander Jung, and Norbert Goertz. "Smooth graph signal recovery via efficient Laplacian solvers". In: *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2017.

[32] J Honerkamp and J Weese. "Tikhonovs regularization method for ill-posed problems". In: *Continuum Mechanics and Thermodynamics* 2.1 (1990), pp. 17–30.

[33] Xiaowen Dong et al. "Learning laplacian matrix in smooth graph signal representations". In: *arXiv preprint arXiv:1406.7842* (2014).

Summary

The main goal of the thesis was to research graph signal methods used to describe and solve the problems posed in today's technological aspects. The main problem was to determine and compare the performances of the two sampling techniques (ad-hoc sampling and random sampling). It was assumed that an incoming signal is a random signal. The signal would get sampled with one of the two techniques mentioned above. The random symmetric weighting matrix was generated and with a help of a weighting matrix and the samples, the graph signal was reconstructed using the Tikhonov recovery method with Gauss-Seidel approach. The better the performance of one of the sampling techniques is, the better the recovered signal will be. In all of the tests, ad-hoc algorithm gave out better results.

Appendix A

Appendix

```
%%
clear; clc;

% matlab file used for varying number of samples M
tic;

iterations = 50;

% intead of ems, any parameter which is to be
   varied can be inserted
for ijk = 1:length(ems)

mse_adhoc = [];
mse_random = [];

for iii = 1:iterations

% Standard deviation of noise added to the graph
   signal
sigma = 10;

% Set Weight-Matrix parameters
N = 120; % Graph Signal Dimension

M = 30;  % Number of Sampled Components

% Randomly choose  symmetric  Weight matrix

% fraction of connected components. Set  0<fac<1
fac = 0.075;


NmaxEdges = nchoosek(N,2);
NW = round(fac * NmaxEdges);


W = zeros(N,N);

jj_vec = mod(randperm(NmaxEdges),N)+1;
```

```
38
39 for ii=1:NW,
40  jj = jj_vec(ii);
41  kk = randi(N);
42
43  while( (kk == jj) || (W(jj,kk)>0) )
44      kk = randi(N);
45  end
46
47  % Set weight matrix component to a numer 0...1
48  W(jj,kk) = rand(1);
49  % Produce a symmetric weight matrix
50  W(kk,jj) = W(jj,kk);
51
52 end
53
54 %create a graph signal
55 [x, x_noise] = GenerateGraphSignal(W,M,sigma);
56
57 M_samples = ems(ijk);
58
59 sampling_vec_adhoc = sampling_adhoc(W, M_samples,
      1e-3);
60 sampling_vec_random = sampling_random(W,
      M_samples);
61
62
63 y_adhoc = x_noise;
64 y_random = x_noise;
65
66 y_adhoc(sampling_vec_adhoc==0) = 0;
67 y_random(sampling_vec_random==0) = 0;
68
69
70 lambda = 0.1;
71
72 x_adhoc = recoverSignal( y_adhoc, W, lambda,
      sampling_vec_adhoc );
73 x_random = recoverSignal( y_random, W, lambda,
      sampling_vec_random );
74
75 if(sum(isnan(x))>0)
76   disp('### NaN components in the solution! Too
      few edges in weight matrix')
77 end
78
79
80 tmp_mse_adhoc = norm(x_noise-x_adhoc,2)^2/N;
81 tmp_mse_random = norm(x_noise-x_random,2)^2/N;
82
```

```
83  if ~isnan(tmp_mse_adhoc) &&
       ~isnan(tmp_mse_random) &&
       ~isinf(tmp_mse_adhoc) && ~isinf(tmp_mse_random)
84      mse_adhoc = [mse_adhoc, tmp_mse_adhoc];
85      mse_random = [mse_random, tmp_mse_random];
86  end
87
88  end % end iterations
89
90  toc;
91
92  if true
93
94      figure(1);
95      N_x = length(x);
96      plot(1:N_x, x_adhoc, 'r-');
97      hold on;
98      plot(1:N_x, x_random, 'b-');
99      plot(1:N_x, x_noise, 'k-');
100     hold off;
101     legend('adhoc','random', 'original');
102 end
103
104 mean_adhoc(ijk) = mean(mse_adhoc)
105 mean_random(ijk) = mean(mse_random)
106
107 fprintf('progress %.1f %%\n',
       ijk/length(ems)*100);
108 end
109
110
111 figure(1)
112
113 N_x = length(x);
114 plot(ems, mean_adhoc, 'r-');
115 hold on;
116 plot(ems, mean_random, 'b-');
117 hold off;
118 legend('adhoc','random', 'original');
119 ylabel('mse');
120 xlabel('M');
```

LISTING 1: mk.m

Appendix B

```
function [ x, x_noise ] =
    GenerateGraphSignal(W,M,sigma)
% The function generates a graph signal for a
    given weight matrix W
% The parameter M is the number of samples
% The parameter sigma is the standard deviation
    of noise added to a signal to make the
% recovery less consistent in a controlled way

szW = size(W);
N = szW(1);

x = zeros(N,1);

% Set Lagrange multiplier for the Tikhonov
    regularization used below
% lambda = 0;

% Pick randomly M signal components to be sampled
Sall = randperm(N);

% Set of measurement indices
S = Sall(1:M);

% complement set of S
SC = Sall(M+1:N);

% randomly choose the graph signal's component
    values:
xr = randn(N,1);

% Get M randomly selected samples
y = xr(S);


% Do Gauss-Seidel iterations until change of
    solution falls below the
% threshold tol
Dold = 1E20;
Dnew =  0;
tol  = 1E-3;
xold = 1E10 .* ones(N,1);

```

```
cnt = 0;

while( abs(Dnew-Dold) > tol*Dold)

    cnt = cnt + 1;

    x(S) = y;
    Wcolsum = sum(W);


    for ll = M+1:N,
        tmp = Wcolsum(SC(ll-M)) - W(SC(ll-M));
        tmpX = 0;
        for jj=1:N,
            if(jj ~= SC(ll-M))
                tmpX = tmpX +
   W(SC(ll-M),jj)*x(jj);
            end
        end
        x(SC(ll-M)) = tmpX/tmp;
    end

    Dold = Dnew;
    err = x - xold;
    Dnew = sqrt(sum(err.^2)/N);
    xold = x;

end


% Produce noisy version of solution
x_noise = x + sigma.* randn(size(x));


 figure(1)
     plot(xr)
     hold on
     plot(S,xr(S),'rs')
     plot(x,'r')
     plot(x_noise,'g')
     hold off
     legend('Original Random Signal','Sampled
   Components','Consistent Signal','Noisy
   Consistent Signal')
     grid on
```

```
83 end
```

LISTING 2: Graph signal generation and Tikhnov regularization with Gauss-Seidel approach

Appendix C

```
function [ sampling_vector ] = sampling_adhoc( W, M, eps_break )
%adhoc1 returns a sampling vector of the signal where thinks should be sampled
% W ... weight matrix
% M ... number of samples
% eps_break ... some small number


N_nodes = size(W,2); % number of nodes, can also be calculated from the weight matrix


 % create this C matrix
 C = W + eye(size(W));

 % generate the degree matrix D
 D = diag(sum(W,2));

 % entries of D as a vector
 d = diag(D);

 % find the maximum and its index
 [~, index_max] = max(d);

 % set this index as the first sample
 S = [index_max];

 % create X
 X = zeros(N_nodes, 1);
 X(index_max) = 1;

 %algorithm
 Y = C*X;

 a_big_number = 30;


 alpha = 1/max(sum(W,2)+0.1);

  W_tmp = W*alpha;

for ss = 1:M-1
```

```
41      X = zeros(N_nodes, 1);
42      for si = S
43          X(si) = 1;
44      end
45
46      for ii = 1:a_big_number
47           Y = W_tmp*Y;
48           min(Y);
49           [min_v, index_min] = min(Y);
50      end
51
52      [val, ind] = sort(Y,'ascend');
53
54      for sw = 1:N_nodes
55          if ~any(abs(ind(sw)-S)<1e-3),
56              S = [S ind(sw)];
57              break
58          end
59      end
60
61
62  end
63  indices = S;
64  sampling_vector = zeros(N_nodes, 1);
65
66  for i=indices % could be implemented faster
        without a loop
67      sampling_vector(i) = 1;
68  end
69
70  end
```

LISTING 3: Ad-hoc sampling

Appendix D

```
function [ sampling_vector ] = sampling_random(
   W, M )
% returns a indicies vector with M 1s

N = size(W,1);
sampling_vector = zeros(N,1);
perm = randperm(N);

indices = perm(1:M);

for i=indices % could be implemented faster
   without a loop
    sampling_vector(i) = 1;
end

end
```

LISTING 4: Random sampling