Universidade Nova de Lisboa
Faculdade de Ciências e Tecnologia
Departamento de Informática

A Study on Parallel versus Sequential Relational Fuzzy Clustering Methods

Rui Miguel Meireles Felizardo

Orientadora:	Prof. Doutora Susana Nascimento
Arguente:	Prof. Doutor João Paulo Carvalho
Presidente do Júri:	Prof. Doutora Margarida Mamede

Dissertação para obtenção do Grau de Mestre em Engenharia Informática.

Fevereiro 2011

Acknowledgements

Firstly I would like to thank Professor Boris Mirkin for all of his suggestions and teachings throughout this work. All the clarifications were most valuable and I learned a lot.

This dissertation was proposed by Professor Susana Nascimento, for whom I have been working for more than two years in a project. I have to thank for all that I have learned during this time. Specifically with respect to the Dissertation, I have to thank for all the guidance and support during all the tasks, for which high availability was always provided.

This marks the end of the Master's Degree in Computer Sciences and a cycle in my academic route. There are also some people who I would like to thank for all these past years in this University.

I want to thank Luísa Lourenço, João Araújo, Rui Rosendo, Mónica Valverde, João Martins, Mauro Delgado, João Mateus, Cátia Ferreira, Frederico Malheiro and Carlos Fernandes, who have been my colleagues and who I have worked with.

I also thank Ricardo Salvador, with whom I have grouped with in projects the last years, for putting up with me for many working sleepless nights at the University. For giving support during most of the time I have to thank Ana Luísa Gestosa, Jorge Pestana, André Silva, Cátia Silvestre, Elói Barros and Márcio Corrêa.

I want to also thank Nuno Veiga for the support and advice all these years, with who I also worked with and learned a lot.

To Professor João Paulo Pimentão, I thank for the availability and support. Also for helping me practice my oral presentation of the thesis, for which I was nervous.

In the last months of the elaboration of the dissertation, I want to thank my close friends Cátia Tavares, Jorge Santos, Filipe Fernandes and Rúben Marques, who helped me calm down, kept me company and took me out sometimes.

I must thank Miguel Mendes also for helping me in the last stages of this work, for giving me strength, advice and lending me his designing skills in the presentation.

Last but not least, I thank my parents, sisters and grandparents for always being there. Essentially my parents, who invested time and money in our education and always believed in us. Without your help and support we wouldn't be where we are now.

A great thank you to all.

Resumo

O Agrupamento Relacional Difuso é uma área de estudo recente que se encontra em crescimento. Novos algoritmos têm sido desenvolvidos, como o *FastMap Fuzzy c-Means* (FMFCM) e o *Fuzzy Additive Spectral Method* (FADDIS), para os quais se têm resultados interessantes nos testes experimentais dos artigos de origem. Como estes algoritmos são bastante recentes na comunidade de agrupamento relacional difuso, não existem muitos estudos experimentais acerca dos mesmos.

Esta dissertação aparece como resposta à necessidade de se realizarem mais testes com esses algoritmos, nomeadamente um estudo comparativo de resultados de duas famílias de algoritmos: as versões paralelas e sequenciais. Estas duas famílias de algoritmos diferem no que diz respeito à forma como agrupam os dados. As versões paralelas extraem os grupos em simultâneo, necessitando do número de grupos como parâmetro inicial, enquanto que as versões sequenciais extraem os grupos um a um até se verificar uma condição de paragem, sendo o número de grupos um dos resultados do algoritmo.

Os algoritmos são estudados relativamente à sua eficácia em devolver boas estruturas de grupos, analisando tanto a qualidade das partições como a determinação do número de grupos, aplicando várias medidas de validação.

Um estudo extensivo de simulação tem sido conduzido através de dois geradores de dados especificamente construídos para os algoritmos em estudo, em particular para analisar a sua robustez em dados com ruído. Resultados com conjuntos de dados padrão reais também são

discutidos.

Presta-se particular atenção ao pré-processamento mais adequado para dados relacionais, em particular para transformações Laplacianas pseudo-inversas.

Palavras-chave: Dados Relacionais; Agrupamento Relacional Difuso; Agrupamento Aditivo Espectral Difuso; Número de Grupos; Índices de Validação.

Abstract

Relational Fuzzy Clustering is a recent growing area of study. New algorithms have been developed, as *FastMap Fuzzy c-Means* (FMFCM) and the *Fuzzy Additive Spectral Clustering Method* (FADDIS), for which it had been obtained interesting experimental results in the corresponding founding works. Since these algorithms are new in the context of the Fuzzy Relational clustering community, not many experimental studies are available.

This thesis comes in response to the need of further investigation on these algorithms, concerning a comparative experimental study from the two families of algorithms: the parallel and the sequential versions. These two families of algorithms differ in the way they cluster data. Parallel versions extract clusters simultaneously from data and need the number of clusters as an input parameter of the algorithms, while the sequential versions extract clusters one-by-one until a stop condition is verified, being the number of clusters a natural output of the algorithm.

The algorithms are studied in their effectiveness on retrieving good cluster structures by analysing the quality of the partitions as well as the determination of the number of clusters by applying several validation measures.

An extensive simulation study has been conducted over two data generators specifically constructed for the algorithms under study, in particular to study their robustness for data with noise. Results with benchmark real data are also discussed.

Particular attention is made on the most adequate pre-processing on relational data, in particular on the pseudo-inverse Laplacian transformation.

Keywords: Relational Data; Relational Fuzzy Clustering; Fuzzy Additive Spectral Clustering; Number of Clusters; Validation Indices.

Contents

List of Figures

List of Tables

List of Symbols

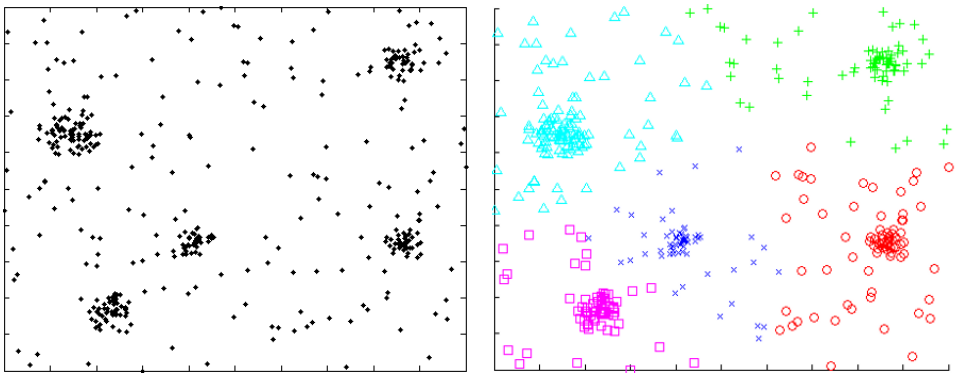
K	Number of clusters	5
d	Distance function	12
x_i	i^{th} object attribute vector	12
X	Object data matrix	12
N	Number of entities	12
S	Similarity matrix	13
D	Dissimilarity matrix	13
R	Relational matrix	14
u_{ik}	Membership of object i to cluster k	19
U	Membership matrix	19
V	Prototype vector	20
m	Fuzzyfication factor	27
μ_k	Intensity of cluster k	41
Λ	Eigenvalues matrix	41
ε	Minimum cluster contribution threshold	45
τ	Minimum residual scatter threshold	46

1 . Introduction

1.1 What is Clustering?

Clustering is considered to be a method of unsupervised learning. It is a technique used in statistic data analysis, such as in machine learning [31], data mining [45], bioinformatics [51] and image recognition [5]. It tries to find a structure in raw data, by creating subsets of similar instances. These subsets are the clusters, which are groups of objects that are similar between them given certain criteria and dissimilar to the ones from other clusters.

As an example of a clustering process, two figures are shown, the first is an unlabeled data set, and the second is a result from a clustering algorithm.



(a) Unlabeled Data (b) Result of Clustering

Figure 1.1 Clustering Example

The Clustering main objective is to find the best grouping of the data. In order to achieve this, some input data may be useful. There is a great range of clustering algorithms, each one may give different solutions for the given set. The problem that emerges is to decide whether a clustering result is acceptable or not, or even better than others.

When analyzing raw data, i.e data for which there is no original classification to compare results, many solutions must be taken under consideration and different algorithms may return different groupings. For example, the following figure obtained from [34]:

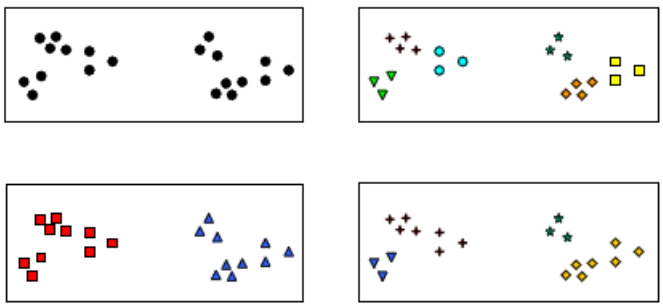


Figure 1.2 Different Clustering Results

Given the unclassified data, it could be concluded at first sight that there are 2 clusters, but there are other possibilities that can be better. Also, the structure of the grouping may be an issue as the next figure, taken from [33], shows:

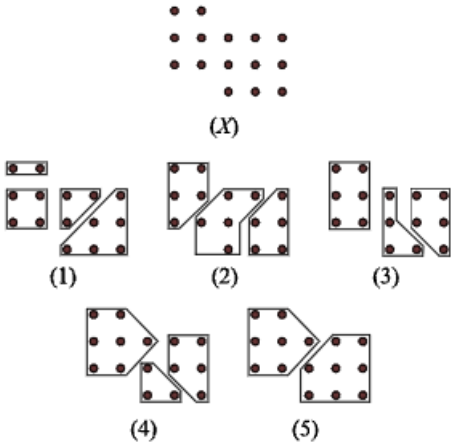


Figure 1.3 Different Clustering Results

As can be seen, many possible groupings do exist along with the number of clusters for the dataset X. Due to the subjectivity and inherent complexity of cluster analysis, many features

must be taken into consideration when analyzing and clustering data, such as the data properties, specialists point of view, etc.

Clustering can be proven to be useful in many fields, and some examples taken from [33] are:

1. **Marketing**: grouping customer data
2. **Biology**: grouping animals and plants
3. **Medicine**: identifying diseases and their severity
4. **Web Mining**: grouping access patterns to determine user's interests
5. **Face Recognition**: grouping similar patterns to identify faces in images
6. **Crime Analysis**: grouping similar patterns of crime scenes to identify certain areas with a higher criminality risk

This work will focus on relational fuzzy clustering algorithms, as will be explained in the next sections.

1.2 Problem Description and Context

There is a distinction between object data and relational data. Whereas object data is described by an attribute-value representation (e.g. height and weight on an object), relational data quantify the relation between each pair of objects, like the similarity or dissimilarity between two

objects. Relational data emerges in applications like Web user profiling [44] and DNA microarray experiments to characterize the expression of groups of genes in the presence of treatments [6].

In the framework of the project "Computational Approach to Ontology Profiling of Scientific Research Organizations" (PTDC/EIA/69988/2006), a more recent real world application concerns the definition of fuzzy membership profiles of the research activities conducted in a University department following the ACM-CCS taxonomy [29]. From an electronic survey tool (ESSA- https://copsro.di.fct.unl.pt/), the researchers of an organization are asked to select up to six topics among the leaf nodes of the ACM-CCS taxonomy and assign each with a percentage which expresses the proportion of the topic in the total of the researcher's activity. This describes the researcher's activity fuzzy membership profile. From that, it is derived a similarity matrix between research topics covered by the fuzzy profiles. Then, a research organization is represented by clusters of ACM-CCS topics to reflect thematic communalities between activities of researchers or teams working on these topics. The Fuzzy ADDItive Spectral clustering (FADDIS) method has been introduced by Mirkin and Nascimento [27][28], for this propose, and will be the object of main investigation in this dissertation.

Relational fuzzy clustering and additive fuzzy clustering are two types of methodological approaches for learning the structure of similarity between objects as groups that share common properties, standing on different model and algorithmic strategies.

The proposed work will be applied to benchmark relational datasets, affinity data and simulated data according to a new Data Generator developed for the Fuzzy Additive Spectral Clustering Algorithm (FADDIS) model, proposed by Mirkin and Nascimento in [27].

The experiments that will take place in this dissertation will take into consideration the used procedures in the mentioned work by Mirkin and Nascimento [28].

1.3 Main Contributions

The main goal of this thesis is to study, experimentally compare and analyse two families of fuzzy relational clustering algorithms:

1. *Parallel versions of Fuzzy Relational Clustering*, where the clusters are simultaneously constructed in each iteration of the algorithm and the number of clusters K is an input parameter of the algorithms. In order to analyse the best number of clusters, validation indices recently proposed in the literature [8][50] will be applied.

2. *Sequential Additive Fuzzy Relational Clustering*, where the clusters are iteratively defined one-by one until a composed stop condition holds, which allows to determine the number of clusters.

The selected parallel versions of Fuzzy Relational Clustering algorithms are the Non-Euclidean Relational Fuzzy C-Means (NERFCM) by Hathaway et al [21] and FastMap Fuzzy C-means (FMFCM) by Brouwer [9]. The chosen sequential algorithms are two recent versions of the Fuzzy Additive Spectral Clustering (FADDIS) algorithm due to Mirkin and Nascimento [28].

Two data generators have been constructed in order to study the effectiveness of former clustering algorithms in recovering a cluster structure of generated data in the presence of noise. More precisely, it had been developed a "Bivariate Normal Data Generator" with noise, and the "Fuzzy Core Clusters Data Generator" having as input a similarity data matrix generated according to the underlying FADDIS model with added Gaussian noise.

An extensive simulation study with data sets generated under distinct levels of noise from distinct number of clusters and with different cardinalities, has been conducted. Particular attention is given on the study of FADDIS own properties, and specific evaluation and validation measures have been implemented to analyse the properties of the algorithms. In particular,

two new error measures have been introduced to evaluate specific properties of the FADDIS algorithms, the Characteristic of the Error of Membership Recovery (CEMR) and the Relative Error of Intensities (REI). In the case of FADDIS, the pre-processing of the data with or without Laplacian Pseudo-Inverse (LAPIN) transformation is an object of analysis, as well as the ability of the algorithm on finding the "natural number of clusters" by tuning its stop conditions. On the other hand, the aspect of determination of the number of clusters in case of NERFCM and FMFCM is evaluated by using recent validation indices proposed in the literature [49]. All the algorithms have also been tested with real world benchmark data sets. In addition, the VAT algorithm was implemented in order to visualise cluster tendency from relational data.

To the present case study, the data that will be subject of analysis will be:

- Datasets generated from the Bivariate Normal Data Generator with noise, as an extension of the datasets generated in Brouwer's previous work [9].

- Datasets generated from the Fuzzy Core Clusters Data Generator;

- Several benchmark datasets taken from the literature [12][55].

In all cases, a previous classification of the data is known. All benchmark datasets have been clustered before using different algorithms and the generated data also has its labels generated.

The obtained results will give a better understanding of the algorithms behaviour under different datasets and it will be verified that some datasets can be clustered with better results with different clustering algorithms.

1.4 Organization

This thesis is organized in the following way: Chapter 1 was a general introduction to clustering, where the main goals and contributions of this dissertation were referred. Chapter 2 is an introduction to Relational Fuzzy Clustering, where the data format and the parallel and sequential methods will be addressed. In Chapter 3, the studied Relational Fuzzzy Clustering algorithms will be described as well as validation indices. Following in Chapter 4, the Spectral Fuzzy clustering is introduced and the Fuzzy Additive Spectral Method presented. Finally in Chapter 5, each one of the experimental studies will be described. The final conclusions and future work are presented in Chapter 6.

Further results of the experimental studies can be found in the Appendices and will be referenced.

2 . Relational Fuzzy Clustering

2.1 Introduction

A common type of clustering algorithms, the hard clustering methods, cluster data such that each entity belongs only to one group. This way, results are crisp and no additional information is given. Fuzzy clustering introduced by Bezdek [3] which will be subject of study in the present work, clusters data in a way that each point has a membership value for each cluster. This expresses the degree of inclusion of the entities to each cluster, and further analysis of these can be conducted.

In a first stage, clustering general information gathering must be made. Analyzing the Clustering algorithms behaviour and their mathematical application will give a better insight on what is happening when the data is being processed.

To start this research, the initial data must be analyzed. There are numerous forms in which data can be represented initially, and the adopted methods to treat this information will be decisive in the outcome.

The initial observation data may appear in different formats, besides being either object or relational data, it can be numerical, consist of strings, percentage, binary, etc.. Each case must be treated in a different way, and even may have a specific method to be normalized and transformed into a more generic set that can be analyzed and used by clustering algorithms.

To understand the data manipulation and treatment that is needed so that the algorithms can be used, some pre-processing techniques will be studied and some used in implementations.

An essential base for the greatest part of the fuzzy algorithms is the Fuzzy C-Means (FCM) Algorithm by Bezdek [4], which will be extended or improved to be used with relational data.

One such important extension is the Non-Euclidean Relational Fuzzy C-Means (NERFCM) due to Hathaway and Bezdek [21], where the relational data given as input does not have to be Euclidean.

Another approach is using the FastMap method by Faloutsos [15] to create object data from relational data. This method will be used since its implementation is linear and this constitutes an efficient and fast algorithm that has great potential. The resulting object data will then be used with the FCM.

It is important to note that the referenced algorithms belong to the family of algorithms where the number of clusters is an input parameter, which can be an issue, since validation techniques must be applied and the algorithm has to be run numerous times with different values for that number. The determination of a good number of clusters is a well known and still unresolved problem in cluster analysis.

A distinct approach, which constitutes the main focus of study is the Fuzzy Additive Spectral method developed by Mirkin and Nascimento [27][28]. This method extracts clusters sequentially, one at a time, until a composed stop condition is achieved. This way, the number of clusters is not an input parameter of the algorithm but instead is provided, in a natural way, by the algorithm.

2.2 Entity-Attribute and (Dis)Similarity Data

In the various clustering algorithms, the input data to be clustered may be in different formats. A common data structure is the entity-attribute matrix, which is a result of describing each object by a group of attributes. These attributes may be numerical, such as percentage, size and weigh,

categorical as gender, country, color, week days or binary. In order to work on these, some techniques must be used based on the knowledge about the data context. Usually the functions to determine the similarity degree for each attribute are distinct since different criteria is needed for the data evaluation.

The main objective is to work on a numerical matrix, where each row represents an object, and each column stand for one attribute which has its own domain of possible values. Many algorithms such as the FCM, receives as input matrices in this format.

In contrast, the relational data expresses how (dis)similar the objects are. Each entity is not defined in terms of a range of attributes, rather all values express a result of comparing all objects using certain criteria. It can either be done by specialists of the area, or can be determined using specific similarity measures. Relational matrices can either express how similar or dissimilar objects are. In a similarity matrix, its maximum value represents high affinity between the objects while in dissimilarity matrices high values mean the opposite. The similarity matrix can also be known as an affinity matrix because it expresses the affinity degree of the objects. A dissimilarity matrix is usually represented as:

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & & \\ d(n,1) & d(n,2) & \dots & \dots & 0 \end{bmatrix}$$

where $d(i, j)$ is the dissimilarity between objects i and j. As an example, a dissimilarity measure is the Euclidean distance, where higher values represent higher distance between instances, therefore greater dissimilarity. The diagonal is composed by only zeros, since an object is not dissimilar from itself.

The choice of the similarity or dissimilarity measure to construct a relational matrix is decisive for the quality of the clustering result, which may be determined by validation indices, as some will be mentioned in the current work.

The main type of data used in this work will be relational, either similarity or dissimilarity matrices. When applying the algorithms, if the given data is in entity-attribute form, a (dis)similarity measure will be used to build a matrix of the desired format (see Section 2.2.1). In the FastMap method the opposite will happen, since FCM receives as input a entity-attribute matrix, an affinity matrix will be converted into that type of data.

2.2.1 Transforming Entity-Attribute Data to Relational Data

Object-Entity data as mentioned before, can be transformed into Relational data using different measures. A well known method is the conventional proximity, that relies on the Euclidean distances to generate a relational dissimilarity matrix:

$$D_{ij} = d^2(x_i, x_j), \tag{2.1}$$

where d is the Euclidean distance and x_i and x_j are respectively the i_{th} and j_{th} objects.

Another used method is the Gaussian Kernel [24], following previous work by Ng. et al [23] and Hathaway et al [17]. Given an object-entity $N \times p$ matrix X, where N is the number of entities and p the number of attributes, affinity similarity data can be obtained by:

$$S_{ij} = e^{\frac{-d(x_i,x_j)^2}{2\sigma^2}}, \tag{2.2}$$

where x_i and x_j represent, respectively, the attribute vector for the objects i and j, d is the Euclidean distance function and the scaling parameter σ^2 controls how the distance between

the objects influences the affinity matrix S. Following the founding papers from Ng et al [23] and Shi and Malik [41], the diagonal elements are made equal to zero. The $2\sigma^2$ value is claimed to be a user defined parameter. In the experiments conducted by Mirkin and Nascimento [29], consistent results have been obtained for $\sigma^2 = p/9$ where p is the number of attributes and after the features have been normalized by range.

There are many occasions when a relational matrix is given to analyse. Since the relational clustering algorithms may receive as input a similarity or dissimilarity matrix, it may be useful converting between each other depending on the given and the required data format. Two examples used by Davé and Sen [12] will be presented to achieve this, where D and S will represent, respectively, dissimilarity and similarity matrices.

- Converting by maximum value:

$$D = \max S - S \tag{2.3}$$

- Converting by minimum value:

$$D = 1/S - \min 1/S \tag{2.4}$$

In particular, the presented methods can be applied the same way to obtain a similarity from a dissimilarity matrix.

2.3 Visualizing Clustering Tendency in Relational Data

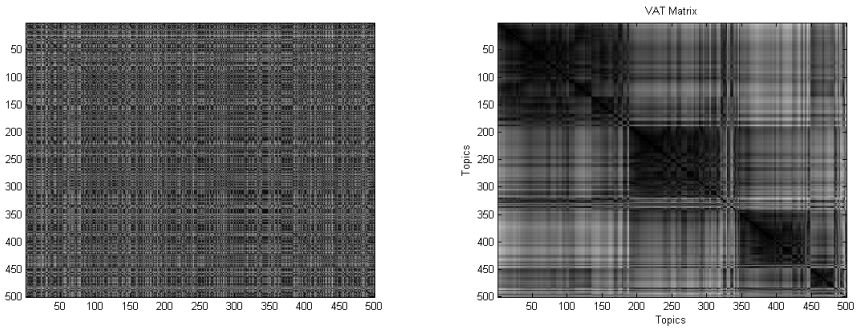
A tool for Visual Assessment of Cluster Tendency (VAT) was developed by Bezdek and Hathaway [7] in order to visualise the cluster tendency in relational dissimilarity matrices. This tool

gives the user a perception of the possible number of clusters and relative sizes. This method reorganizes the columns and rows of the relational matrix according to the dissimilarities between objects. As an example, two figures are displayed.

The expected output is a plot, where black areas mean strong similarities and white areas great dissimilarities between objects.

For a given relational matrix, if we directly plot it grayscale, the resulting image is for example 2.1(a). After applying the VAT tool to the dissimilarity matrix, the obtained figure is 2.1(b).

As can be analysed from the first figure, nothing can be concluded, but from the second one, four clusters can be distinguished located in the diagonal of the matrix with darker grey colors. This may indicate that the given dataset may have a structure of four clusters. The original classified dataset has in fact four clusters, and its plot is in figure 2.2.



(a) Normal plot output (b) VAT plot output

Figure 2.1 Example for applying VAT

The method algorithm is explained following [7], where R is the input relacional matrix, n is the number of entities.

1. *Initialize:*

   Set:

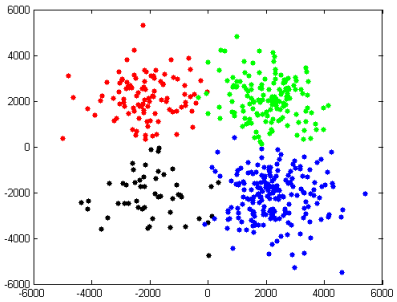


Figure 2.2 Original Classified Dataset example

$G = \{1, 2, ..., n\}$;

$J_1 = J_2 = []$;

$P = [0, ..., 0]$;

2. *Select* $(i, j) \in \underset{p \in G, q \in G}{argmax}\{R_{pq}\}$

   Set:

   $P(1) = i$;

   $J_1 = \{i\}$;

   $J_2 = G \backslash \{i\}$;

3. *For* $r = 2, ..., n$*:*

   Select $(i, j) \in \underset{p \in J_1, q \in J_2}{argmin}\{R_{pq}\}$

   Set:

   $P(r) = j$;

   $J_1 = J_1 \cup \{j\}$;

   $J_2 = J_2 \backslash \{j\}$;

   Iterate next r.

4. *Using the reordered vector P, determine the new relational matrix R' with:* $R'_{ij} = R_{P(i)P(j)}$, for $1 \leq i, j \leq n$.

5. *Output the matrix as a greyscale image, where low values are darker and higher values are brighter.*

2.4 Parallel versus Sequential Relational Fuzzy Clustering Algorithms

Each clustering method has an algorithm to group the given data. Two main approaches can be distinguished among the algorithms presented in the scientific literature. Parallel clustering algorithms require as input parameter the number of clusters to be given, therefore the data is seen as a K-cluster structure and populated taking this assumption in consideration. On the other hand, in case of sequential clustering, the algorithm extracts clusters one by one in each iteration of the algorithm, until a stop condition is reached whose threshold values may have to be tunned.

In the present work the described clustering algorithms are the following: the Fuzzy C-means (FCM), Non-Euclidean Relational Fuzzy C-means (NERFCM), FastMap FCM (FM-FCM) and the Any Relational Clustering Algorithm (ARCA). As for the studied sequential clustering algorithms, they will be the Relational Fuzzy Subtractive Clustering (RFSC) and the Fuzzy Additive Spectral Method (FADDIS).

There are several sequential clustering approaches, such as the one by Ott et al [36] or Urahama et al [14]. It usually starts with the selection of some similarity measure to build an affinity matrix. The next step is the clustering algorithm and no information about size, shape

or number of clusters is known. The main objective of a clustering algorithm is to extract the best clustering solution from the data.

Each of the clustering algorithms has its own advantages, but not all have these items in consideration. Most algorithms fail when the dataset has clusters with different shapes, like the Sequential extraction of fuzzy clusters using a spectral graph by Inoue and Urahama [22], where only Gaussian distributed data can be clustered correctly. Also, even when choosing from the best similarity measures and attributes to represent the data, some algorithms are unable to find acceptable results as mentioned before.

2.5 The Problem of the Number of Clusters

The determination of the number of clusters for a given dataset has been and still is a big issue in the clustering community. New and raw data is always a challenge to clustering since no additional information is known in most cases, leading to a number of clusters problem. In these cases, an expert intervention is crucial when available, as a range of number of clusters can be obtained. Having this, the algorithms can be applied as many times as the size of that range, and then the quality of clusters and clustering analyzed using validation indices [16][35][47][49][50].

As for the sequential clustering algorithms, the number of clusters is not needed as input parameter, but some information about the data is, since a stop criterion usually needs to be tuned. The same is applied in these cases, modifying the parameters and the stop criteria, various results may be obtained and therefore analysed and validated in order to select the best combination of values and cluster structure.

The best solutions are often based on validation indices, which may lead to different results

based on which index and algorithm are used. Having this, the number of clusters is again the main issue.

With this in mind, the practical results and analysis in this work will compare results from many datasets and using many different algorithms.

3. Relational Fuzzy Clustering Methods

3.1 Introduction

This chapter presents several algorithms of Relational Fuzzy Clustering. First, Fuzzy C-means (FCM) will be presented since some algorithms are either extensions or use it in their computation. Following, four relational fuzzy clustering algorithms will be introduced, two of which will be used in the practical experiments. Finally, some validation indices will be applied to evaluate the quality of fuzzy partitions, in particular validation indices to determine the best number of clusters from relational clustering will be described.

3.2 Fuzzy C-Means

The Fuzzy C-Means by Bezdek [4] is a reference algorithm for the clustering community. Based on object-feature data, the method determines subsets that will constitute the clusters. The number of generated clusters is an input variable to the algorithm. The FCM algorithm converges, at least along a subsequence, to either a local minimum or saddle point of the FCM objective function as presented by Hathaway et al in [19].

The produced membership matrix $U \in \mathbb{R}^{c \times n}$, where c is the number of clusters and n the number of objects. Each u_{ik} is the degree that the object O_k belongs to cluster i. Also, the matrix U will be under the following constraints:

1. $u_{ik} \in [0,1], \quad 1 \leq i \leq c, 1 \leq k \leq n$

2. $\sum_{i=1}^{c} u_{ik} = 1, \quad 1 \leq k \leq n$

3. $\sum_{k=1}^{n} u_{ik} > 0, \quad 1 \leq i \leq c$

The Fuzzy partitioning allows each object total membership to be distributed through all clusters. The algorithm assumes that the clusters all circular shaped, and will determine their centers v_i, prototypes.

The algorithm will be described then as:

1. *Initialize*: Select an initial partition U^0 that obeys the constraints mentioned above, startup a prototype matrix $V = [v_1, ..., v_c] \in \mathbb{R}^{c \times n}$, c is the input number of clusters, n is the number of objects, and select a fuzzyfication factor $m \geq 1$ and a $\|\cdot\|$, that is any inner product norm. A stop criterion ε must be defined as well as the number of maximum iterations for the algorithm, $r = 1...rmax$.

2. *Update distances*:

$$v_i^{(r)} = \frac{\sum_{k=1}^{n} (u_{ik}^{(r)})^m x_k}{\sum_{k=1}^{n} (u_{ik}^{(r)})^m} \quad 1 \leq i \leq c \tag{3.1}$$

$$(d_{ik}^{(r)})^2 = \left\| x_k - v_i^{(r)} \right\|^2 \quad 1 \leq i \leq c, 1 \leq k \leq n \tag{3.2}$$

3. *Update partition*:

   For each object $k = 1...n$,

   if $d_{ik}^{(r)} > 0$, *for* $i = 1...c$

$$u_{ik}^{(r+1)} = \left[\sum_{j=1}^{c} \frac{(d_{ik}^{(r)})^2}{(d_{jk}^{(r)})^2} \right]^{-\frac{1}{m-1}} \tag{3.3}$$

   otherwise,

$$u_{ik}^{(r+1)} = 0 \text{ if } d_{ik}^{(r+1)} > 0, \quad \text{where } u_{ik}^{(r+1)} \in [0,1] \text{ and } \sum_{i=1}^{c} u_{ik}^{(r+1)} = 1 \tag{3.4}$$

4. *Convergence check*: If $r = rmax$ or $\mid u^{(r+1)} - u^{(r)} \mid < \varepsilon$ stop the algorithm, otherwise go to step 2 of the algorithm with $r = r+1$.

3.3 Non-Euclidean Relational FCM (NERFCM)

The Relational Fuzzy C-means (RFCM) is a relational dual of fuzzy c-means, following Hathaway et al in [20] and [18]. The FCM as mentioned before is an algorithm that uses as input object-feature data and RFCM appears as an extension of FCM to relational data. the Non-Euclidean Relational FCM by Bezdek et al [21], is an extension for RFCM, works as a safeguard, since relational data may not be Euclidean, and applying a *spreading* transformation, the algorithm works as expected in these cases as well.

Some modifications to the original FCM will be made to achieve this. A relational matrix R is given as input, thats is defined as

$$R_{ij} = \left\| x_i - x_j \right\|^2 \quad 1 \leq i, j \leq n, \tag{3.5}$$

where n is the number of objects. The second step formulas in FCM will be changed for:

$$v_i^{(r)} = \frac{\left[(U_{i1}^{(r)})^m ... (U_{in}^{(r)})^m \right]^T}{\sum_{k=1}^{n} (U_{ik}^{(r)})^m} \quad 1 \leq i \leq c, \tag{3.6}$$

where c is the number of clusters, and

$$(d_{ik}^{(r)})^2 = (Rv_i^{(r)})_k - \frac{1}{2}(v_i^{(r)})^T R(v_i^{(r)}) \quad 1 \leq i \leq c, 1 \leq k \leq n. \tag{3.7}$$

Using these equations in the second step, RFCM is obtained. In order to build a non-Euclidean resistant algorithm, some modifications must be made. The transformation consists in the addition of a positive number β to all off-diagonal elements of R. The new matrix R_β can be defined as:

$$(R_\beta)_{jk} = \begin{cases} R_{jk} + \beta, & \text{if } j \neq k \\ 0, & \text{if } j = k \end{cases}. \tag{3.8}$$

If R is a relational matrix, but not Euclidean, then there exists a positive β_0 such that R_β is Euclidean for all $\beta \geq \beta_0$ and not Euclidean for $\beta < \beta_0$. Calculating the exact value for β_0 could be very expensive computationally, but an overestimation could lead into data loss as well, so the method will calculate values for β in each iteration.

The changes that need to be made to FCM will be adding the initialization parameter $\beta = 0$ to the first step, and replacing the second with:

$$v_i^{(r)} = \frac{\left[(u_{i1}^{(r)})^m \ldots (u_{in}^{(r)})^m\right]^T}{\sum_{k=1}^{n}(u_{ik}^{(r)})^m} \quad 1 \leq i \leq c, \tag{3.9}$$

$$(d_{ik}^{(r)})^2 = (R_\beta v_i^{(r)})_k - \frac{1}{2}(v_i^{(r)})^T R_\beta (v_i^{(r)}) \quad 1 \leq i \leq c, 1 \leq k \leq n. \tag{3.10}$$

If $(d_{ik}^{(r)})^2 < 0$ for some i and k then

$$\triangle\beta = max\left\{\frac{-2(d_{ik}^{(r)})^2}{\left\|v_i^{(r)} - e_k\right\|^2}\right\} \tag{3.11}$$

and modify

$$(d_{ik}^{(r)})^2 \leftarrow (d_{ik}^{(r)})^2 + \frac{\triangle\beta}{2}\left\|v_i^{(r)} - e_k\right\|^2 \quad 1 \leq i \leq c, 1 \leq k \leq n, \tag{3.12}$$

updating β value

$$\beta \leftarrow \beta + \triangle\beta. \tag{3.13}$$

In the mentioned formulas e_k is the kth unit vector in $\mathbb{R}^n$. The NERFCM will be a reference method in the present work.

This algorithm has been applied with successful results within many datasets. Examples of such data in bioinformatics, are the gene similarity matrices obtained from using Basic Local Alignment Search Tool (BLAST) by Altschul et al [1][54] and Gene Ontology [53] similarities in a dataset of 194 human genes, following Popescu et al [38].

3.4 FastMap FCM

This method, by Brouwer [9], starts by using the Fast Map algorithm [15] on given data to produce points in the Euclidean space so that it can be visually analysed and FCM [4] applied. Visual clustering allows a better understanding and analysis of the obtained results and a comparison to other methods, where similar elements are positioned closer and dissimilar elements

further. This visualization maps the objects preserving their proximity relationship. The produced data will be k-dimensional, where k is an input parameter, and the selected algorithm to use on the result is the FCM.

3.4.1 FastMap Method

The FastMap method maps data into points in a k-dimensional space, preserving their dissimilarities. This method is quite fast, since its implementation is linear. The only input is the dissimilarity matrix, that should be Euclidean, even though the method works if it is not. Given a matrix with n objects and their distance information, the method will find some other n objects but in a k-dimensional space such that the distances are maintained.

This is achieved by creating an axis from 2 well separated objects, O_a and O_b. With this, the projections of each object using this axis, can be given in terms of the Dissimilarity matrix D:

$$x_i = \frac{D_{a,i}^2 + D_{a,b}^2 - D_{b,i}^2}{2 * D_{a,b}}, \tag{3.14}$$

and the distance between the object projections in the reduced space:

$$D'^2_{i,j} = D^2_{i,j} - (x_i - x_j)^2 \qquad i, j = 1...n, \tag{3.15}$$

where n is the number of objects, and i,j are 2 object indexes. The idea is to project the n objects on k mutually orthogonal directions. The main difficulty is to achieve this with only one input, that is the distances matrix.

3.4.2 Describing the method

First, 2 pivot objects must be chosen in order to generate an axis on the k-dimensional space.

choose-distant-objects(O,D

1. Choose a random object as second pivot O_b
2. Set O_a as the farthest object from O_b
3. Set O_b as the farthest object from O_a
4. Return O_a,O_b

Having this function, the algorithm for FastMap can be presented as follows:

FastMap:

begin Global variables:

N x k matrix X	In the end, the i-th row is the projected i-th object
2 x k pivot array PA	Array of pivot objects for each iteration
int col = 0;	Current X column being updated

FastMap(k,D,O)

1)if $k \leq 0$ then *end*;

else $col++$; fi

2)*choose pivot objects*

$[O_a, O_b] = choose\_distant\_objects(O,D)$;

3)*put pivot object ids in PA*

Pa[1,col] = a;

Pa[2,col] = b;

4)if $D(O_a, O_b) = 0$

then for $i := 1$ to n

do $X[i, col] = 0$; *end*; od fi

5)Project all objects on the axis formed by O_a and O_b

for $i := 1$ to n do $X[i, col] =$ compute (3.14); od

6)Compute D' from (3.15)

call $FastMap(k-1, D', O)$

After constructing the k-dimensional object data, the FCM algorithm is then applied as usual.

This algorithm is quite recent and not yet many studies have been made, but some experiments were made with randomly generated matrices by Brouwer in [9]. All the matrices were clustered using different algorithms and FMFCM returned better results in many cases. Being an interesting method to analyse, this algorithm will be implemented and used in experiments in the current work.

3.5 Any Relation Clustering Algorithm (ARCA)

The Any Relation Clustering Algorithm (ARCA) by Corsini et al [11], is based on the FCM algorithm and is claimed by the authors to outperform the NERFCM method [21]. In ARCA, each datum is represented by a vector of its relation strengths to the other objects of the data set and the prototype is an object that may not exist in the data set, and its relationship with all the objects is representative of the mutual relationships of a group of datum with high similarity. The algorithm starts by partitioning the dataset minimizing the euclidean distance between objects and the prototype of the cluster.

This partitioning is obtained by minimizing:

$$J_m(U,V) = \sum_{i=1}^{C}\sum_{k=1}^{N} u_{i,k}^{m}\delta^2(x_k, v_i), \tag{3.16}$$

where C is the number of clusters, N is the number of objects, x_i is an object, v_i is a prototype, $u_{i,k}$ is the membership degree of object x_k to the cluster with prototype v_i, m is the fuzzyfication factor and $\delta(x_k, v_i)$ is the deviation between the relation of x_k and the other objects and the relation of v_i and the other objects. δ is defined as:

$$\delta(x_k, v_i) = \sqrt{\sum_{s=1}^{N}(r_{k,s} - v_{i,s})^2}, \tag{3.17}$$

where $r_{k,s}$ is the relation between x_k and x_s, and $v_{i,s}$ is the relation between the prototype v_i and x_s. Using the Lagrange multipliers [46] minimization method, the ARCA algorithm can be defined as:

1. Define C as $2 \leq C < N$, $1 < m < \infty$, and choose an initial partition U(0).

2. Start the iteration process, with iteration l:

(a) calculate $V(l) = [v_{i,k}^{(l)}]$:

$$v_{i,k}^{(l)} = \frac{\sum_{s=1}^{N} u_{i,k}^{m^{(l)}} r_{k,s}}{\sum_{s=1}^{N} u_{i,k}^{m^{(l)}}} \tag{3.18}$$

(b) update $U(l)$ to $U(l+1)$:

$$u_{i,k}^{(l+1)} = \left[\sum_{j=1}^{C} \left(\frac{\delta(x_k, v_i)^{(l)}}{\delta(x_k, v_j)^{(l)}} \right)^{\frac{2}{m-1}} \right]^{-1} \tag{3.19}$$

(c) if $U(l+1) - U(l) < \varepsilon$ then stop, else continue iteration and restart step 2 with $l = l+1$

This algorithm has been applied to 4 benchmark datasets (Fat Oil Data, cat cortex, proteins and Tamura dataset) and 4 synthetic relational generated datasets in work by Corsini et al [11]. The algorithm returned good results in most cases according to the validity indices used by the authors.

3.6 Relational Fuzzy Subtractive Clustering (RFSC)

The algorithm, as described by Suryavanshi et al [43], receives as input a relational matrix R, where R_{ij} is the similarity or dissimilarity between datum x_i and x_j. $R_{ij} \geq 0$, $R_{ij} = R_{ji}$ and $R_{ii} = 0$. The algorithm starts by obtaining the potential of being a center cluster of each object, using:

$$P_i = \sum_{j=1}^{N_U} e^{-\alpha R_{ij}^2}, \tag{3.20}$$

where

$$\alpha = 4/\gamma^2, \tag{3.21}$$

where N_U is the total number of points to be clustered and γ is calculated as follows. Each object i has a neighbourhood-dissimilarity value γ_i, which is the median of dissimilarity to all other objects. The neighbourhood-dissimilarity for the dataset γ is the mean of the various γ_i, for each i. The object with the highest potential P_1^* is then extracted as the first cluster center, and the potential of other objects is reduced proportionally to the similarity to the extracted center. It is trivial to note that the potential of objects that are nearer to the extracted center will be reduced more than the ones that are farther away. After this, the next cluster center will be obtained from the next highest potential object, x_t, with P_t^*, as it will be a potential cluster center. At this point 2 thresholds have to be set, the accept ratio A and the reject ratio F, having $0 < A$, $F < 1$ and $F < A$. With this we can decide if the extracted object will be a cluster center:

1. If $P_t > AP_1^*$, then x_t is selected as the next cluster center and the subtractive method repeats.

2. If $P_t < FP_1^*$, then x_t is rejected and the algorithm stops.

3. If $FP_1^* < P_t < AP_1^*$, then the potential is said to have fallen in the gray region, and if the object offers a good trade-off between its potential and distance from the existing clusters then it is accepted as the new cluster center.

This process continues until the item 2 is verified, which is the termination condition. After this, C cluster centers were found, and the memberships of each object to each cluster is determined with:

$$u_{ij} = e^{-\alpha R_{c_i j}^2}, \tag{3.22}$$

where $i = [1..C], j = [1..N]$ and $R_{c_i j}$ is the dissimilarity of the i^{th} cluster center with object x_j. If $x_j = x_{c_i j}$, then $R_{c_i j} = 0$ and the membership $u_{ij} = 1$. Unlike the fuzzy c-means, RFSC relaxes the constraint $\sum_{i=1}^{C} u_{ij} = 1$, which makes it less sensitive to noise.

A cluster validity index can then be used to measure the quality of the clustering. In general good clustering provides high inter-cluster distance values and low intra-cluster distance values. The defined validity measure for the RFSC algorithm uses a ratio between the compactness to the separation of clusters. The compactness is obtained as follows:

$$Compactness = \sum_{i=1}^{C} \left(\frac{\sum_{j=1}^{N} u_{ij}^2 R_{c_i j}^2}{\sum_{j=1}^{N} u_{ij}} \right) / C \tag{3.23}$$

As for separation, it is calculated as:

$$Separation = min_{i \neq k} R_{c_i c_k}^2, \tag{3.24}$$

where $R_{c_i c_k}$ is the dissimilarity between cluster centers c_i and c_k.

The index is obtained as:

$$Index\,of\,goodness = \frac{Compactness}{Separation} \tag{3.25}$$

The best clustering is found using the accept ratio A and the reject ratio F such that the index of goodness finds its minimum value.

Some experiments were conducted with user access logs from the web server of Computer Science and Software Engineering Department (CSE) at Concordia University by Suryavanshi et al [44]. The objective was to build some user profiles based on their web navigation.

3.7 Validation Indices

Clustering validation does not only entail the analysis of the best number of clusters, but also the quality of the clustering partition and of each cluster itself. These indices express how ”good” are clusters in a dataset, based on geometric or statistical properties according to a predefined meaning of good clusters.

Validity indices can be used having different perspectives:

- When analyzing only the resulting membership matrix U, the validity index measures the quality of U in terms of how the partitioning reveals a substructure.
- Having initial information, an index can be used to measure how a clustering algorithm can recover the original cluster structure.

Applying validity measures usually constitutes a problem in the clustering community, since no matter how good an index may be, there is always a dataset for which it will not give acceptable results. There is no best measure to be used with all algorithms and data structures. In order to bypass this problem, it is suggested according to Bezdek et al [5] to apply different validation measures and select the best value among the results. Even though this approach can be more accurate, it may still fail due to the various reasons already referred. The best strategy is to use many different parameters when applying the clustering algorithms and use various indices. If the procedure leads to consistent results, then it can be assumed that an acceptable structure was found, otherwise more simulation has to be done before taking under consideration any results.

Some validation indices were studied and will be implemented to compare results of clustering and cluster quality using different relational clustering algorithms. When using parallel clustering algorithms, a range of values for the number of clusters is usually applied to analyse results and take conclusions regarding the best parameters according to indices. These will be

used as a criterion in the various data sets, as some algorithms return better results than others depending on the used data set.

In the next sections some indices that were used by Brouwer [9] will be presented. Some approaches for cluster validity can be found in the list by Bezdek [**?**].

3.7.1 Roubens

The measure proposed by Roubens [39] (and also Bezdek [2]) only requires the resulting fuzzy membership matrix obtained from the clustering algorithm, and the value represents how close it is to being crisp. The method is

$$V_R = \frac{\sum_{p=1}^{np} \sum_{k=1}^{nk} (M_{n,p})^2}{np}, \tag{3.26}$$

where M is the fuzzy membership matrix, np is the number of objects and nk the number of clusters. The closer this value is to 1 the more crisp are the results and if the value is closer to 0 it means the resulting matrix is fuzzier.

3.7.2 Rand Index and Adjusted Rand Index

In order to compare clustering results with some already existent ones, a measure of agreement of partitions memberships is needed. Given two partitions: U being the original classification and V the clustering result, both composed by the objects from the set $S = o_1, o_2, ..., o_n$, the contingency table can be built as the one in Figure 3.1. It will have r rows for r predefined classes and c columns for c obtained clusters. Each n_{ij} stands for the number of objects that are in class U_i and cluster V_j; a_i is the number of objects in class U_i; b_j is the number of objects in cluster V_j.

U/V	V_1	V_2	$\cdots$	V_C	Sums
U_1	n_{11}	n_{12}	$\cdots$	n_{1C}	a_1
U_2	n_{21}	n_{22}	$\cdots$	n_{2C}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
U_R	n_{R1}	n_{R2}	$\cdots$	n_{RC}	a_R
Sums	b_1	b_2	$\cdots$	b_C	

Figure 3.1 Contingency table example

For o_x,$o_y \in S$, the Rand Index (RI) according to Yeung and Ruzzo [50] can be calculated by:

$$RI = \frac{a+d}{a+b+c+d}, \tag{3.27}$$

where

1. a is the number of pair of objects (o_x, o_y) such that both belong to the same cluster in both sets of results.

2. b is the number of pair of objects (o_x, o_y) such that both belong to different clusters in both sets of results.

3. c is the number of pair of objects (o_x, o_y) such that both belong to the same cluster in the first set of results and belong to different clusters in the second set.

4. c is the number of pair of objects (o_x, o_y) such that both belong to the same cluster in the second set of results and belong to different clusters in the first set.

The result obtained from this index will be in the interval $[0,1]$ and is best when closer to 1. The Adjusted Rand Index (ARI), following the same authors [50], can be calculated from

$$ARI = \frac{\sum_{ij}\binom{n_{ij}}{2} - \frac{\sum_i \binom{a_i}{2}\sum_j \binom{b_j}{2}}{\binom{N}{2}}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \frac{\sum_i \binom{a_i}{2}+\sum_j \binom{b_j}{2}}{\binom{N}{2}}}, \tag{3.28}$$

where $N = \sum_{i=1}^{c} b_i + \sum_{j=1}^{r} a_j$.

As the RI, ARI takes its values from the interval $[0,1]$ and the best result corresponds to the maximum value.

3.7.3 Extended Xie-Beni Fuzzy Validity Index

This index appears as an extension to the well known Xie and Beni validity measure [49], which takes under consideration the separation and compactness of clusters in a fuzzy c-partition. This is expressed as a ratio of the intra-cluster scatter and minimum inter-cluster distance.

The presented index consists of a relational version of the Xie and Beni measure, introduced by Bezdek et al [8]. This technique determines the cluster prototypes, used to calculate the index, as

$$p_j = \underset{1\leq k\leq n}{\operatorname{argmax}} \left\{ \sum_{i=1}^{n} u_{j,i}^{m} \cdot r_{k,i} \right\},\ j \in \mathbb{N}_{1,c}, \tag{3.29}$$

where p_j is the index to the object that represents the j^{th} prototype, m the fuzzification factor, $u_{j,i}$ is the membership of the i^{th} entity for the j^{th} cluster from the membership matrix U and $r_{k,i}$ is the affinity value between the k^{th} and i^{th} entities from the relational data matrix R. The relational Xie-Beni validity index can then be computed as

$$V_{RXB} = \frac{\sum_{i=1}^{n}\sum_{j=1}^{c} u_{j,i}^{m} \cdot r_{p_j,i}}{n \cdot \min\limits_{j\neq k}\left\{r_{p_j,p_k}\right\}}. \tag{3.30}$$

As the resulting value decreases, the clustering quality increases according to this validity measure.

4. Spectral Fuzzy Clustering

4.1 Introduction

Spectral clustering, following Ng et al [23], is a family of graph-based algorithms, which usually start from local information structured in a weighted graph based on the eigenvectors of the corresponding similarity matrix. These algorithms when compared to traditional methods like K-means by Ding and He [13], return better results with higher quality, as it is an easy algorithm to implement that relies on linear algebra operations.

These algorithms make use of the eigenvalues and eigenvectors [10] as a crucial factor when determining the clusters. As pointed before, the number of clusters still is an unresolved issue in the Clustering community, as there are many algorithms that need this value as input, but there are also some that use some criteria to determine the number of clusters to extract. Several approaches are made in this subject.

Initially, a similarity graph must be obtained from the dataset. In the majority of spectral clustering algorithms, the Gaussian similarity function (**??**) is used when building the similarity matrix.

Note that different values for σ in equation (**??**) may affect the clustering results, as it represents the width of the neighbourhoods. A similarity graph can then be obtained from the similarity matrix. Some algorithms used the similarity matrix as the similarity graph. But there are different approaches as:

- ε-*neighbourhood graph*: Each node is connected to all the others nodes within a range of value ε.

- *K-NN graph*: Each node is connected to the K nearest nodes.

- *Mutual K-NN graph*: Node i is connected to node j, if and only if j is one of the K-NN of i and i is one of the K-NN of j.

In spectral clustering there are two ways to obtain clusters, using minimization methods or maximization methods.

As for minimization methods, some use the Laplacian matrix, explained in more detail by Weisstein [48] and by Luxburg [26], which can be defined in many ways:

- Unnormalized Laplacian,

$$L = D - W \tag{4.1}$$

- Normalized symmetric Laplacian,

$$L = I - D^{-1/2} W D^{-1/2} \tag{4.2}$$

- Normalized random walk Laplacian,

$$L = I - D^{-1} W \tag{4.3}$$

These Laplacians are used by algorithms as the unnormalized spectral clustering, Shi and Malik [41] algorithm and Ng et al. [23] algorithm. The last two are cases of normalized spectral clustering. The clustering algorithm then proceeds, where a method using the eigenvectors corresponding to the first k lowest eigenvalues determines the clusters. These values usually are 0 or closer. These methods are based on the minimization of the eigenvalues. Lower eigenvalues, correspond to eigenvectors that identify clusters.

As for spectral clustering which are based on maximization problems, the sequential fuzzy

clustering by Inoue and Urahama [22] is an example. These algorithms extract clusters sequentially until certain criteria are verified. It determines the extracted number of clusters.

The spectral clustering approaches have some limitations as shown by Nadler and Galun [32]. Using some similarity measures, the first eigenvectors of the corresponding adjacency matrices not always can cluster datasets that contain structures with different scales of size and density successfully.

4.2 Fuzzy Additive Spectral Method (FADDIS)

A new spectral method for fuzzy clustering was proposed by Mirkin and Nascimento [27][28]. This method was inspired in the clustering problem obtained when trying to group researcher topics of interest on their current research, based on the respective efforts. The obtained clusters would represent the research department's main areas of application and could be placed within a taxonomy, the ACM-CCS [52].

The algorithm serves as an extension of the spectral decomposition of a square matrix, extracting the clusters one at a time, stopping when a certain criteria is met.

The data is given in the form of entities (researchers) and topics, where each matrix cell has a value in the interval [0,1] corresponding to the proportion of the effort given by the researcher on that topic. Initially the similarity between topics must be defined using a similarity measure proposed by the same authors.

A square similarity matrix A is obtained where a_{mj} is the similarity between topics m and j. To accomplish this, the used measure will be presented and illustrated. Given the initial table from figure 4.1 taken from [27], one can easily calculate the individual weighs n_v that represent the number of topics with a positive score:

	i_1	i_2	i_3	i_4
A	0.6			0.2
B	0.4		0.2	0.2
C		0.2	0.4	0.2
D		0.3	0.4	0.2
E		0.5		0.2
F				
n_v	2	3	3	5

Figure 4.1 Membership values for six topics assigned by four individuals

In the table, the topics are represented by $\{A,B,C,D,E,F\}$, the individuals by i_n, $n = 1..4$ and as mentioned before the individual weighs by n_v. Note that each individual has to give an effort value for the chosen topics such that they sum up to one. Since the individuals that select less topics get higher scores than the ones that choose a larger number, a weighting factor is used to compensate:

$$a_{tt'} = \sum_{v=1}^{V} \frac{n_v}{n_{max}} f_{tv} f_{t'v}, \tag{4.4}$$

where n_{max} is the maximum value of n_v over all $v = 1,2,...,V$ and f_{tv} is the proportion of research efforts given by the individual v to the topic t.

This similarity measure has some relevant properties that should be mentioned:

- the resulting matrix is definite semipositive
- Higher membership values lead to greater similarity
- The similarity between two topics increases if more researchers choose both
- Two topics have a positive similarity value if at least one researcher selects both.

A fuzzy cluster of topics, can be expressed by $u = (u_t)$, where t is a topic and $0 < u_t < 1$ for

all t. Also an intensity value $\mu > 0$ is considered, such that it represents the significance of the cluster in the organization.

Given K fuzzy clusters, the additive clustering model can be expressed as:

$$a_{tt'} = \sum_{k=1}^{K} \mu_k^2 u_{kt} u_{kt'} + e_{tt'}, \tag{4.5}$$

where $u_k = (u_{kt})$ is the membership vector of cluster k, μ_k its intensity and $e_{tt'}$ the associated error. The contribution of cluster k to the similarity between t and t' is represented by the expression $\mu^2 u_{kt} u_{kt'}$, so it depends on the cluster intensity and the memberships.

A more generic similarity measure to use with this algorithm is the Gaussian Kernel (see Section 2.2.1), which will also be applied in most of the experiments described in Chapter 5.

Given a similarity matrix $A = (a_{tt'})$, the objective is to find K fuzzy clusters u_k and their respective intensities such that the sum of the squared errors is minimized, $\sum_{t,t'} e_{tt'}^2$. A similarity matrix W is defined at the start of the algorithm such that $W = A$.

In order to make the cluster structure expressed by the similarity matrix more clear, some techniques can be used. Such methods are the Laplacian transformations [26] (see Section 4.1) as presented previously. Since the presented Laplacian transformations rely on minimum eigenvalues and the model uses the maximum eigenvalue, a Laplacian pseudo-inverse transformation (Lapin) will be applied, as proposed by Mirkin and Nascimento [28][29], defined as follows:

$$L_n^+(W) = \tilde{Z}\tilde{\Lambda}^{-1}\tilde{Z}', \tag{4.6}$$

where $\tilde{\Lambda}$ and $\tilde{Z}$ are obtained from the spectral decomposition $L_n = Z\Lambda Z'$ of the matrix $L_n = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$, equation 4.2. First the eigenvalues Λ of the similarity matrix are calculated and a vector T' of indexes of the non-zero values is established. With this, one can obtain the

following:

- $\tilde{\Lambda} = \Lambda(T', T')$
- $\tilde{Z} = Z(:, T')$

Since only the non zero eigenvalues were taken in consideration, after the Lapin transformation they will remain unchanged. The maximum eigenvalues of L_n^+ are the inverse of the minimum eigenvalues of L_n which constitute the same eigenvector.

This transformation can improve the data structure in such way that given two entities t and t', according to W they are dissimilar, but in L_n^+ they are highly related if there are many series of linked entities connecting both in W. The ability of Lapin transformation in transforming elongated structures into convex clusters has been a subject of mathematical attention [25].

An example to make this clear, given by Mirkin and Nascimento [28], is to consider $\lambda_1 = 0.05$ and $\lambda_2 = 0.2$ so that thei complements to unity are 0.95 and 0.8 while the inverses are 20 and 5. The growth of the gap between the values, from 0.15 to 15, is impressive. The latter suits the FADDIS approach much better.

This way the pseudo-inverse Laplacian can be used with this algorithm. Next, the algorithm will be described and W is the used similarity matrix with or without the Lapin transformation.

The approach used on FADDIS extracts one cluster at a time by a principal component analysis[1] method. At each step, a minimization problem is considered:

$$E = \sum_{t,t'} (w_{tt'} - \xi u_t u_{t'})^2, \tag{4.7}$$

[1] for a detailed description of Principal Component Analysis consult [40]

where ξ is an unknown positive weight such that $\mu = \sqrt{\xi}$, ξ is the intensity, $u = (u_k)$ is the membership vector and $W = (w_{tt'})$ is the given similarity matrix.

At each step, the matrix W will change such that the part that corresponds to the similarities for the found cluster are subtracted:

$$W - \mu^2 uu^T, \tag{4.8}$$

where μ and $\mathbf{u}$ are respectively the intensity and membership vector of the found cluster.

The optimal value of ξ is given by

$$\xi = \frac{\mathbf{u}'W\mathbf{u}}{(\mathbf{u}'\mathbf{u})^2}. \tag{4.9}$$

By replacing the value of ξ in (4.7), the following expression is defined:

$$E = S(W) - \xi^2 \left(\mathbf{u}'\mathbf{u}\right), \tag{4.10}$$

where $S(W) = \sum_{t,t'} w_{tt'}^2$ is the similarity data scatter. It can be expressed as:

$$S(W) = G(\mathbf{u}) + E, \tag{4.11}$$

and

$$G(\mathbf{u}) = \xi^2 \left(\mathbf{u}'\mathbf{u}\right)^2 = \left(\frac{\mathbf{u}'W\mathbf{u}}{\mathbf{u}'\mathbf{u}}\right)^2. \tag{4.12}$$

Since the objective is to find an optimal cluster and E constitutes a part that is undefined, the main problem is to maximize $G(\mathbf{u})$, or rather its square root:

$$g(u) = \xi \mathbf{u}'\mathbf{u} = \frac{\mathbf{u}'W\mathbf{u}}{\mathbf{u}'\mathbf{u}}, \tag{4.13}$$

and u can be normalized such that $\mathbf{u}'\mathbf{u} = 1$. Equation (4.12) becomes $G(\mathbf{u}) = \xi^2$ and it can be concluded that the weight of an additive fuzzy cluster is in fact equal to its contribution to the data scatter. Note that (4.13) represents the Rayleigh quotient, therefore its maximum value will be the maximum eigenvalue λ of W obtained from the respective eigenvector u.

The objective is to maximize $g(\mathbf{u})$ and take its projection to the set of nonnegative fuzzy membership vectors. The projection operator $P(\mathbf{z})$ is difined as follows:

$$P(\mathbf{z}) = \mathbf{u} / \|\mathbf{u}\|, \tag{4.14}$$

where $\mathbf{u} = (u_t)$ is defined as:

$$u_t = \begin{cases} 0, & \text{if } z_t \leq 0; \\ z_t, & \text{if } 0 \leq z_t \leq 1; \cdot \\ 1, & \text{if } z_t \geq 1 \end{cases} \tag{4.15}$$

The algorithm extracts clusters one by one, and since all the memberships for each topic t over all clusters must sum up to one, a value must be subtracted in each iteration, $1 - \alpha_{kt}$ where $\alpha_{kt} = \sum_{l=1}^{k} u_{lt}$. This way $P(z)$ suffers a slight change, replacing 1 for $1 - \alpha_{kt}$ so that $\sum_{k=1}^{K} u_{kt} \leq 1$.

As explained in [28], if z is an eigenvector of W with eigenvalue λ, so is $-z$, therefore $u = P(z)$ and $u^- = P(-z)$ should be taken into consideration. $P(-z)$ gets the negative components of z, and a problem arises, whether to select u or u^-, and the respective intensity $\mu = u'Wu$ or $\mu^- = u^{-\prime}Wu^-$. To solve this problem, a maximization of the contribution $G = \xi^2$ to (4.11) is considered. The case that maximizes its intensity, either μ or μ^-, according to E in (4.7), is chosen.

Having this in mind, the maximization of $G = \xi^2$ can be extended such that instead of considering only the eigenvectors that correspond to the maximum eigenvalue, it should consider

all the eigenvectors. This is justified by the fact that for some matrices, the obtained μ and μ^- from non maximum eigenvalues can be greater than the intensities resulting from the maximum eigenvalue.

Two versions of the FADDIS algorithm are constituted according to the way a cluster is chosen:

1. Version "m": the projections of the eigenvectors that correspond to maximum eigenvalues are considered;

2. Version "a": the projections of all the eigenvectors that correspond to all positive eigenvalues are considered.

Over the iterations of the algorithm, some properties are pointed out in [28]:

- It is equivalent to replace the original matrix A for its symmetric version $(A+A')/2$.
- The cluster contributions are additive, so that each can be represented as a proportion of the initial similarity data scatter $S(A)$.
- The cluster contributions tend to decrease in each iteration of the algorithm.
- The residual matrix W can eventually get negative eigenvalues which will stop the algorithm because $\xi < 0$ when applying (4.9).
- The scatter of the residual data decreases in each iteration.

The stop criteria for the algorithm are the following:

SC_1 : As mentioned before, the value of ξ gets a negative value;

SC_2 : The cluster contribution is lower than a prespecified threshold $\varepsilon > 0$;

SC_3 : The residual scatter E is lower than a prespecified $1-\tau$ value (a percentage of the original similarity scatter, for example);

SC_4 : The number of extracted clusters has reached the maximum number of clusters allowed.

The one-by-one fuzzy cluster extraction spectral algorithm is described by Mirkin and Nascimento [28], as follows:

FADDIS Algorithm

- Input: Symmetric similarity matrix A, threshold of the contribution of an individual cluster $\varepsilon > 0$, threshold of the total clusters contribution $\tau > 0$.
- Output: The number of fuzzy clusters K, cluster membership vectors u_1,u_2,...,u_K, their intensity values μ_1, μ_2,...,μ_K and contributions $G(u_1)$,$G(u_2)$,...,$G(u_K)$.

0. *Initialization*: Set $k = 1$ and $W = A$; compute the data scatter $S = \sum_{i,j} w_{ij}^2$.
1. *Spectral*: Find the set of all positive eigenvalues $\Lambda = \lambda$ and corresponding normed eigenvectors $Z = z_\lambda$ for matrix W.
2. *Stop-condition*: If Λ is empty, computation stops and outputs clusters, along with their intensities and contributions, have been found so far.
3. Fuzzy cluster projection in either "m" or "a" version:
    - "m": Take eigenvectors z and $-z$ corresponding to maximum $\lambda \in \Lambda$, use operator P to compute their projections u and u^- to the set of fuzzy cluster membership vectors, and take that of them maximizing the contribution, $G(u)$ or $G(u^-)$, as u_k along with corresponding $\mu_k = u_k' W u_k$ and $G(u_k)$.

- "a": Take eigenvectors z and $-z$ corresponding to all $\lambda \in \Lambda$, use operator P to compute their projections to the set of fuzzy cluster membership vectors, and take that of them maximizing the contribution, $G(u)$, as u_k along with corresponding $\mu_k = u_k' W u_k$ and $G(u_k)$.

4. *Stop-condition*: Check whether $G(u_k)/S < \varepsilon$ or $\sum_{l=1}^{k} G(u_l)/S > \tau$. If this is verified, the computation stops, k is taken as K, and all found clusters are output. Otherwise, add 1 to k, set W equal to $W - \mu^2 u_k u_k'$ and go to step 1.

5 . Experimental Studies

5.1 Introduction

In this chapter, the experimental studies using 4 relational fuzzy clustering algorithms: two parallel, namely FMFCM and NERFCM, and two sequential, the two versions of FADDIS, will be described and results analysed. It should be stressed that the FMFCM has been selected in this study as a very recent Relational Fuzzy Clustering algorithm proposed in the literature that outperforms both Windham's [56] and Rouben's [39] methods. The NERFCM has been selected as a well established algorithm in the Relational Fuzzy Clustering community.

The studies with Bivariate Normal Data Generator will be presented firstly, followed by the Fuzzy Core Cluster Data Generator and finally Benchmark datasets.

During the presentation of the results, the appendices will be referenced in order to point out more results.

5.2 Study with Bivariate Normal Data Generator with Noise

5.2.1 Description of the Data Generator

This Data Generator is based on Brouwer's [9] generated datasets. It is an extension such that noise is introduced.

The datasets consist of 4 clusters of entities generated from a Bivariate Normal distribution, with variance of $\sigma^2 = 950$. The center of the each cluster is defined as: C_1, C_2, C_3 and C_4. This is done such that the center of each cluster is placed in a different quadrant and the center of the set of all clusters is $(0,0)$.

The centers of the clusters to be generated are the following:

1. $C_1 = (1500, 1500)$
2. $C_2 = (-1500, 1500)$
3. $C_3 = (-1500, -1500)$
4. $C_4 = (1500, -1500)$

Each cluster is generated around each center with different numbers of entities: 50, 100, 200 and 150.

The noise is achieved by introducing a scale parameter *sca* that either gets the clusters closer or more distant from each other. To do so, a value will be summed for all points in the clusters such that each cluster will move in its own direction (depending on which quadrant their center is located).

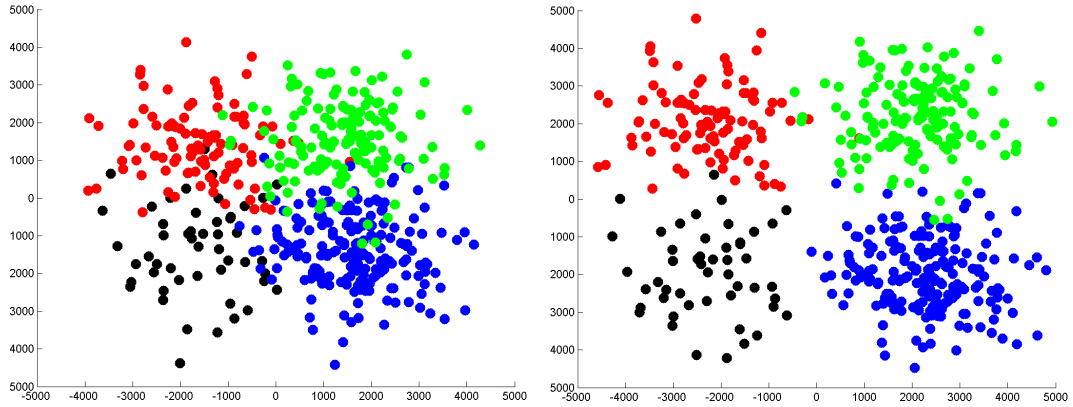
The suggested method calculates the distance *dist* from the center of the cluster in the 1^{st} quadrant and the 3^{rd} quadrant. Then the scaling parameter *sca* defines the number of times that this distance *dist* is summed to the positions of the entities of each cluster.

Each data point from each cluster is relocated according to its respective quadrant: The cluster from the 1^{st} quadrant moves in the direction of vector $(1,1)$; the 2^{nd} quadrant cluster moves according to vector $(-1,1)$, the 3^{rd} to the vector $(-1,-1)$ and the 4^{th} according to $(1,-1)$. This way, each Data point P_{C_i} (generated from cluster with center C_i) is transformed in point P'_{C_i} as follows:

1. $P'_{C_1} = P_{C_1} + (1,1) * sca * dist$
2. $P'_{C_2} = P_{C_2} + (-1,1) * sca * dist$
3. $P'_{C_3} = P_{C_3} + (-1,-1) * sca * dist$

4. $P'_{C_4} = P_{C_4} + (1,-1) * sca * dist$

An example, a dataset consisting of 500 entities has been generated with scale value $sca = 0$ (Figure 5.1(a)), while Figure 5.1(b) presents the dataset generated with scale parameter $sca = 10$. Each cluster is visualised in diferent colors: cluster C_1 has 150 entities colored in green, C_2 has 100 entities in color red, C_3 has 50 entities in black and finally C_4 has 200 entities and is colored blue.



(a) Generated Dataset with $sca = 0$ (b) Generated Dataset with $sca = 10$

Figure 5.1 Dataset with two different scales of noise

5.2.2 Goal of the Study

The main goal of this study is to analyse the ability of the relational fuzzy clustering algorithms being studied (FADDIS-m [28], FADDIS-a [28], FastMap FCM [9], NERFCM [21]) to recover the cluster structure of random generated data sets under a Bivariate Normal distribution, with different levels of noise. Due to the different characteristics of FADDIS-m/a, FCFCM, and NERFCM algorithms the following aspects will be analysed:

1. Adequate pre-processing measure to transform the generated object data into relational

data. Specifically, two similarity transformations are compared:

(a) Conventional proximity (2.1)

(b) Gaussian Kernel (**??**)

In the case of FADDIS algorithms, and when applying the Gaussian Kernel, it will be analysed the effect of pre-processing or not with the Lapin transformation [28].

2. Ability to recover the original number of clusters of generated data. In case of FADDIS, this will be analysed via its stop conditions and corresponding thresholds. In case of FMFCM and NERFCM the correct number of clusters will be evaluated through the analysis of the Extended Xie-Beni index (3.30).

3. For all the algorithms, the obtained fuzzy partitions are compared against the original ones via the Adjusted Rand Index (3.28).

5.2.3 Setting of the Experiments

The experiments had the following setting:

- Each of the four clusters is generated with cardinalities: $\{50, 100, 200, 150\}$;
- The scale values *sca* used are: $\{-5, 0, 5, 10, 20, 50\}$;
- Ten datasets had been generated and used for each scale value, which sum up to 60 different datasets, and the mean and standard deviation of the results for each group of 10 datasets for each scale value was analysed.
- The NERFCM and FMFCM are run for $K = \{3, 4, 5\}$;
- In case of FADDIS, the minimum contribution threshold ε was fine tuned for each case:

1. Gaussian Kernel: $\varepsilon = 0.04$
2. Gaussian Kernel with LAPIN: $\varepsilon = 0.018$

In order to analyse the clustering results, each fuzzy partition is defuzzyfied by maximum membership.

5.2.4 Discussion of the Results

In this section the results for each algorithm will be shown and finally a summary table with the best results will be presented in order to compare the algorithms clustering results.

5.2.4.1 FADDIS-a

For this algorithm, two methods for pre-processing were studied, Gaussian Kernel with and without the Lapin transformation.

The capacity of the algorithm to extract the correct number of clusters and the associated stop condition for each scale value. The tables for the Gaussian Kernel and Gaussian Kernel with Lapin are the tables 5.1 and 5.2, respectively. The description of each stop condition SC_i can be found in Section 4.2 (p. 45).

	FADDIS-a / no Lapin	
scale	Extracted Clusters (Mode)	Stop Condition
-5	5	SC_2
0	4	SC_2
5	4	SC_2
10	4	SC_2
20	4	SC_2
50	4	SC_2

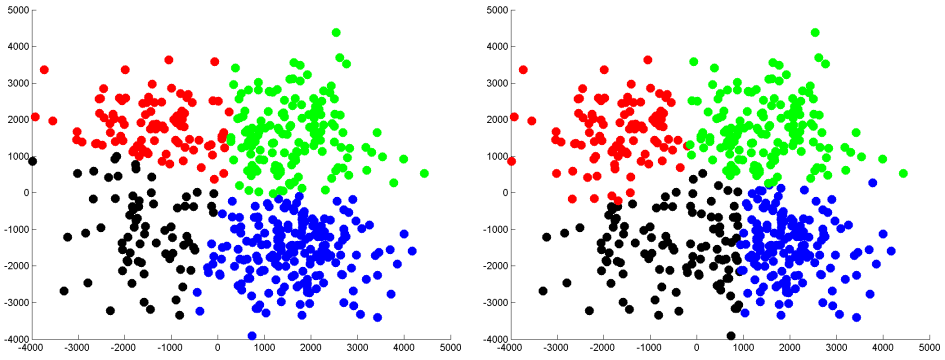
Table 5.1 FADDIS-a Most frequent Stop condition and Number of extracted clusters for Gaussian Kernel (in 10 runs)

	FADDIS-a / Lapin	
scale	Extracted Clusters (Mode)	Stop Condition
-5	4	SC_2
0	4	SC_2
5	4	SC_2
10	4	SC_2
20	4	SC_2
50	4	SC_2

Table 5.2 FADDIS-a Most frequent Stop condition and Number of extracted clusters for Gaussian Kernel with Lapin transformation (in 10 runs)

The results for both pre-processing methods have similar results with the exception of Gaussian Kernel for $scale = -5$ where more noise is introduced and 5 clusters were extracted. The stop condition for this algorithm is always the SC_2.

Two clustering results examples, one for each pre-processing (with or without Lapin), using $sca = 0$ are ilustrated in Figures 5.2(a) and 5.2(b).



(a) Clustering Result with Lapin (b) Clustering Result without Lapin

Figure 5.2 Two FADDIS-a clustering results for $sca = 0$ using Gaussian Kernel pre-processed with Lapin or without Lapin transformation

Figure 5.3 shows the plots of the mean and standard deviation of the ARI index for the FADDIS-a results with/without Lapin, varying the *sca* parameter. The corresponding mean/std values are summarized in Table 5.3 and the best (highest) values are marked in bold face.

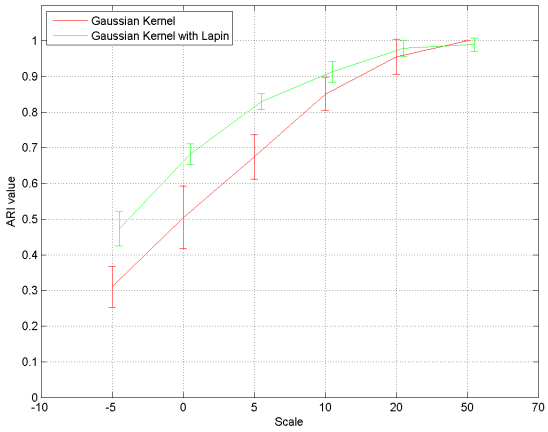


Figure 5.3 FADDIS-a AVG/STD plots for ARI

	FADDIS-a	
scale	Gaussian Kernel	Gaussian Kernel with Lapin
-5	0.31/0.06	**0.47/0.048**
0	0.5/0.09	**0.68/0.02**
5	0.67/0.063	**0.83/0.02**
10	0.85/0.047	**0.91/0.03**
20	0.96/0.05	**0.98/0.2**
50	**1/0**	0.99/0.018

Table 5.3 FADDIS-a ARI AVG/STD for Gaussian Kernel with and without the Lapin transformation

Due to the higher ARI values in case of Gaussian Kernel with Lapin transformation, one can conclude that it is the best pre-processing method for FADDIS-a for this generated data.

5.2.4.2 FADDIS-m

The same kind of experiment has been conducted with the FADDIS-m algorithm, like the one presented in the last subsection.

This algorithm has a different behaviour when compared to FADDIS-a, since it appears to be more unstable as will be verified. The number of extracted clusters and associated stop condition for the Gaussian Kernel without and with Lapin are presented respectively in Tables

5.4 and 5.5.

	FADDIS-m / no Lapin	
scale	Number of Clusters	Stop Condition
-5	5	SC_2
0	4	SC_2
5	4	SC_2
10	3	SC_2
20	3	SC_2
50	4	SC_2

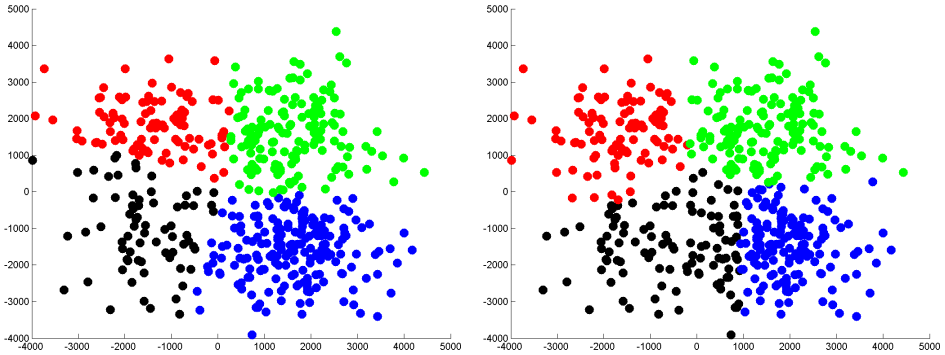
Table 5.4 FADDIS-m Most frequent Stop condition and Number of extracted clusters for Gaussian Kernel (in 10 runs)

	FADDIS-m / Lapin	
scale	Number of Clusters	Stop Condition
-5	4	SC_2
0	4	SC_1
5	4	SC_1
10	3	SC_2
20	3	SC_2
50	2	SC_1

Table 5.5 FADDIS-a Most frequent Stop condition and Number of extracted clusters for Gaussian Kernel with Lapin transformation (in 10 runs)

The FADDIS-m algorithm recovers four clusters for low values of the scale *sca*. As the noise is reduced, by increasing the value of the scale parameter, the algorithm retrieves less clusters. The stop condition for the Gaussian Kernel is always the minimum cluster contribution being smaller than the specified threshold ε. As for the Gaussian Kernel with the Lapin transformation, in some cases the algorithm stops since no positive eigenvalues are obtained from the residual matrix.

Two clustering results examples, one for each pre-processing (with or without Lapin), using $sca = 0$ are ilustrated in Figure 5.4.



(a) Clustering Result with Gaussian Kernel with Lapin

(b) Clustering Result with Gaussian Kernel without Lapin

Figure 5.4 Two FADDIS-m clustering results for $sca = 0$ with and without the Lapin transformation

Analysing the ARI for both pre-processing methods, the plot from Figure 5.5 and respective Table 5.6 are obtained.

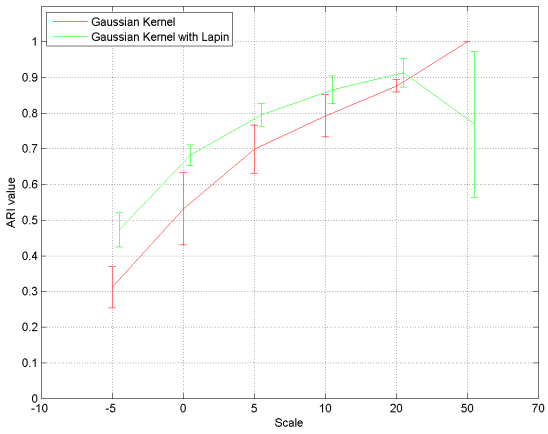


Figure 5.5 FADDIS-m avg/std plots for ARI

After analysing the results, it can be concluded that the Lapin transformation pre-processing improves the results of the algorithm as higher ARI values are obtained. From the scale value $sca = 50$ it is observed that the ARI index of found partitions with pre-processing by Gaussian Kernel with Lapin is lower, and so leads to worse partitions. These results confirm some conclusions stated in Mirkin and Nascimento [28], that the Lapin pre-processing destroys well

	FADDIS-m	
scale	Gaussian Kernel	Gaussian Kernel with Lapin
-5	0.31/0.06	**0.47/0.048**
0	0.53/0.1	**0.68/0.03**
5	0.7/0.068	**0.79/0.03**
10	0.79/0.06	**0.87/0.04**
20	0.88/0.02	**0.91/0.04**
50	**1/0**	0.77/0.2

Table 5.6 FADDIS-m ARI avg/std for Gaussian Kernel with and without the Lapin transformation

defined cluster structures, but improves the clustering process when more noise is introduced.

For the present study, lower scale values are of more interest to be analysed and the Gaussian Kernel with Lapin transformation is selected as the best pre-processing for FADDIS-m.

5.2.4.3 FastMap FCM

In order to analyse the ability of FMFCM in recovering the "natural" number of clusters for different levels of noise, the algorithm is run for the number of clusters $K = \{3,4,5\}$ and the found partitions evaluated according to the Extended Xie-Beni index (3.30) whose minimum indicates the best number of clusters and the ARI index (3.27), which compares the obtained partitions to the original, is to be maximized.

The analysis of the provided results from both indices are concordant. The FMFCM recovers 3 clusters with better ARI and EXB indices results than 4 clusters, for higher level of noise (i.e $sca = \{-5,0\}$). However, when the cluster structure is well defined (i.e $sca = \{5,10,20,50\}$), the FMFCM achieves the best results for $K = 4$ clusters.

For FastMap FCM, the conventional proximity is the used dissimilarity measure to build a relational matrix.

The obtained ARI values for each case are presented in the table 5.7.

According to ARI, the best resulting cluster partitions with more noise are with $K = 3$. From

	FastMap FCM		
scale	K = 3	K = 4	K = 5
-5	**0.47/0.047**	0.44/0.045	0.37/0.03
0	**0.66/0.035**	0.61/0.096	0.54/0.013
5	0.76/0.018	**0.84/0.016**	0.67/0.031
10	0.82/0.015	**0.93/0.021**	0.75/0.029
20	0.86/0.008	**0.99/0.009**	0.82/0.067
50	0.87/0.007	**1/0**	0.87/0.07

Table 5.7 Adjusted Rand Index avg/std for FastMap for each K

$scale = 5$ the highest ARI is always obtained for $K = 4$, which indicates that the algorithm should be run with an input of four clusters.

Analysing the results of the Extended Xie-Beni index from the table 5.8, the same can be concluded: with more noise, three clusters as input obtains lower EXB values which indicates a better result, and from $scale = 5$, $K = 4$ is the best input.

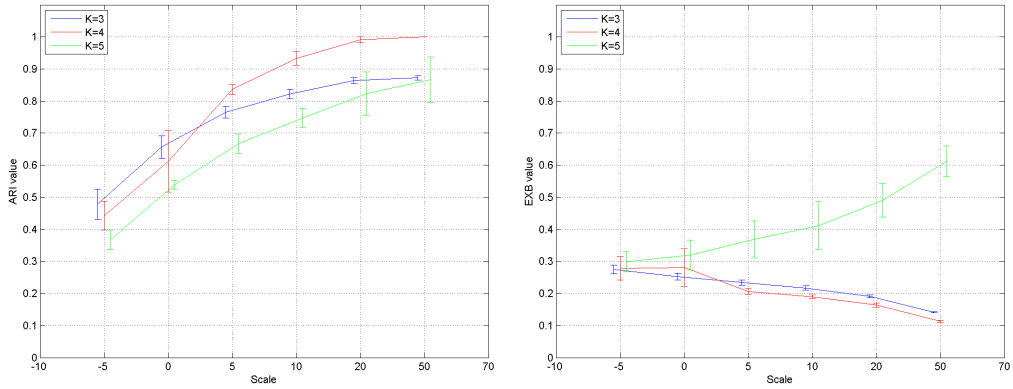
	FastMap FCM		
scale	K = 3	K = 4	K = 5
-5	**0.28/0.014**	0.28/0.037	0.3/0.03
0	**0.25/0.011**	0.28/0.059	0.32/0.047
5	0.23/0.008	**0.21/0.01**	0.37/0.058
10	0.22/0.006	**0.19/0.006**	0.41/0.075
20	0.19/0.0045	**0.16/0.007**	0.49/0.05
50	0.14/0.0018	**0.11/0.003**	0.61/0.048

Table 5.8 Extended Xie-Beni avg/std for FastMap for each K

The corresponding figures for ARI and EXB for FastMap are respectively in Figures 5.6(a) and 5.6(b).

Three clustering results examples, one for each value of the input parameter K, using $sca = 0$ are ilustrated in Figure 5.7.

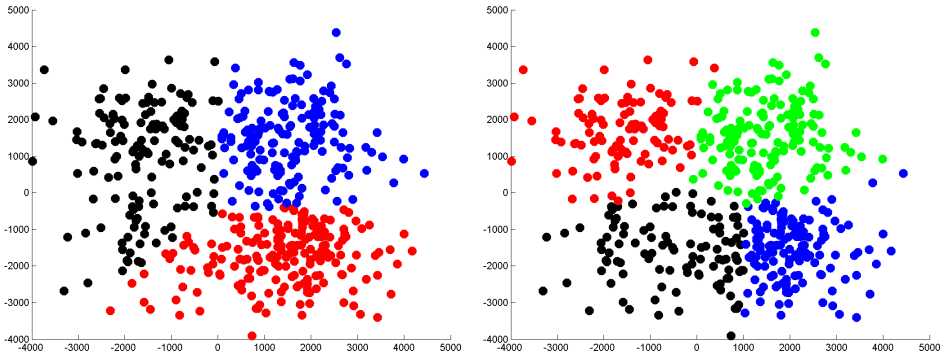
The best input number of clusters for FastMap FCM for this Data Generator is of $K = 4$, which is the original number of clusters. Analysing the results for ARI and EXB, it can be



(a) Adjusted Rand Index avg/std plot for FMFCM results

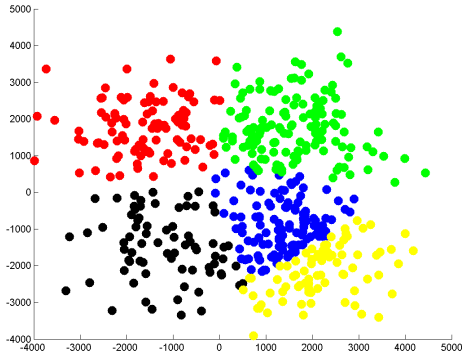
(b) Extended Xie-Beni avg/std plot for FMFCM results

Figure 5.6 FastMap FCM avg/std plots for EXB and ARI



(a) Clustering Result with $K = 3$

(b) Clustering Result with $K = 4$



(c) Clustering Result with $K = 5$

Figure 5.7 Three FMFCM clustering results for $sca = 0$ and $K = \{3,4,5\}$

	NERFCM		
scale	K = 3	K = 4	K = 5
-5	**0.47/0.05**	0.44/0.05	0.37/0.035
0	**0.66/0.034**	0.64/0.058	0.53/0.032
5	0.76/0.018	**0.84/0.016**	0.67/0.036
10	0.82/0.015	**0.93/0.021**	0.74/0.025
20	0.86/0.008	**0.99/0.009**	0.85/0.07
50	0.87/0.007	**1/0**	0.87/0.075

Table 5.9 Adjusted Rand Index avg/std for NERFCM for each K

concluded that FMFCM is not very robust to noise, as the best output is obtained with $K = 3$.

5.2.4.4 NERFCM

As in the case of FastMap FCM, the conventional proximity is the dissimilarity measure used to build a relational matrix as input for the NERFCM algorithm. The Extended Xie-Beni index and the Adjusted Rand Index will also be used to evaluate the best input value of the number of clusters.

The obtained ARI values for each K are presented in the table 5.9.

The results are similar as FastMap FCM. For lower scale values, the ARI index indicates that the best resulting partitions are with $K = 3$. From $scale = 5$, the best result is always attained with $K = 4$.

The Extended Xie-Beni results are shown in the table 5.10.

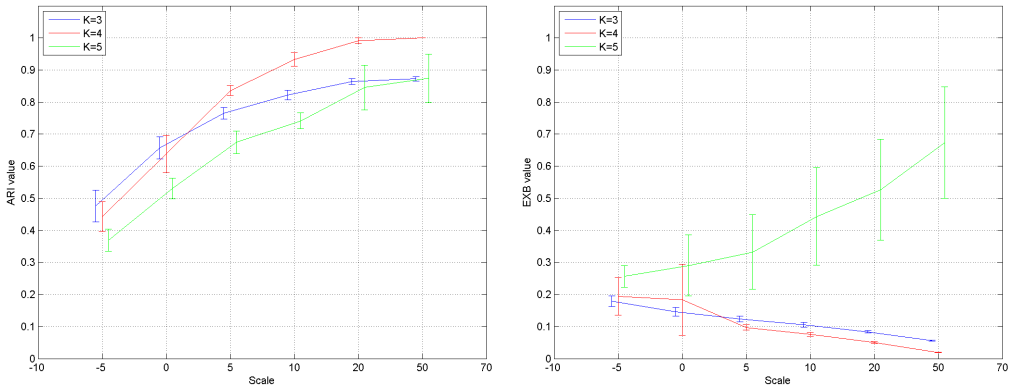
For EXB, the lowest results, that indicate better solutions, are also obtained with $K = 3$ for datasets with more noise and with $K = 4$ for $scale = \{5, 10, 20, 50\}$.

The plots corresponding to the tables 5.9 and 5.10 are respectively in figures 5.8(a) and 5.8(b).

The results from both indices are concordant and similar to those of FMFCM. The NERFCM recovers 3 clusters with better ARI and EXB for lower values of the scale parameter $sca =$

	NERFCM		
scale	K = 3	K = 4	K = 5
-5	**0.18/0.016**	0.19/0.058	0.26/0.034
0	**0.15/0.013**	0.18/0.11	0.29/0.09
5	0.12/0.01	**0.097/0.008**	0.33/0.12
10	0.11/0.007	**0.075/0.006**	0.44/0.15
20	0.08/0.004	**0.05/0.003**	0.53/0.16
50	0.06/0.002	**0.02/0.001**	0.67/0.17

Table 5.10 Extended Xie-Beni avg/std for NERFCM for each K



(a) Adjusted Rand Index avg/std plot for NERFCM results

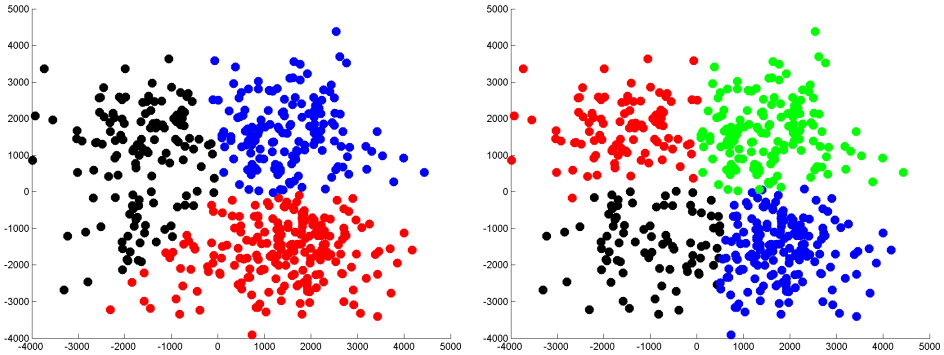
(b) Extended Xie-Beni avg/std plot for NERFCM results

Figure 5.8 FastMap FCM avg/std plots for EXB and ARI

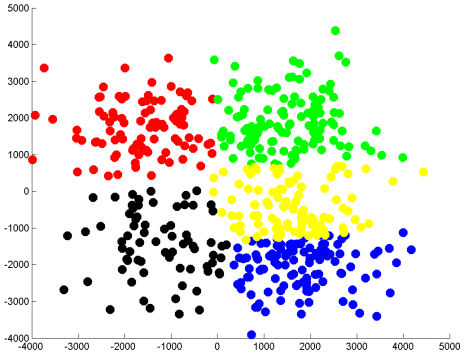
$\{-5,0\}$ (i.e with more noise). However, when the cluster structure becomes more well defined (i.e $sca = \{5,10,20,50\}$), the NERFCM achieves the best indices results for $K = 4$ clusters.

Three NERFCM clustering results examples, one for each value of the input parameter K, using $sca = 0$ are shown in Figure 5.9.

The best input number of clusters for NERFCM for this Data Generator is of $K = 4$, which is the original number of clusters. Having in consideration the results for the EXB index, it can be concluded that NERFCM is not very robust to noise, as the best output is obtained with $K = 3$. As of the ARI index, it is concordant with EXB in the previous statement, as it has lower values compared to situations with less noise (i.e $scale \geq 5$).



(a) Clustering Result with $K = 3$ (b) Clustering Result with $K = 4$



(c) Clustering Result with $K = 5$

Figure 5.9 Three NERFCM clustering results for $sca = 0$ and $K = \{3,4,5\}$

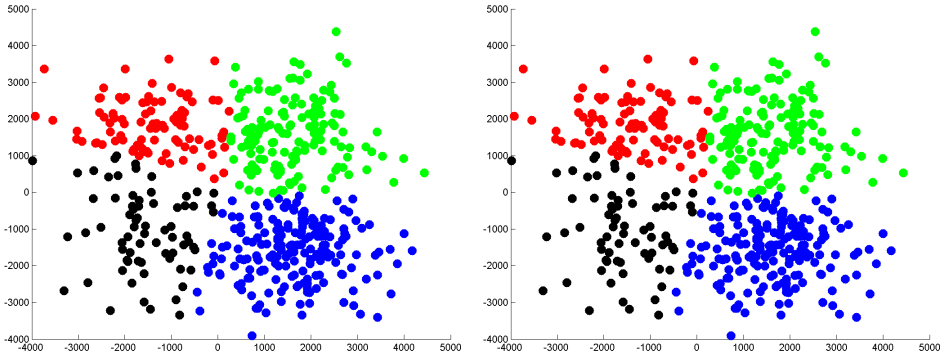
5.2.4.5 Summary of Results

Preliminary studies have been performed using the conventional proximity with both versions of FADDIS and using the Gaussian Kernel with FMFCM and NERFCM. The results were worse, therefore not taken into consideration.

Selecting the best pre-processing in case of FADDIS and input parameters in case of FMFCM and NERFCM, the results can be compared in the summary Table 5.11. Also, one example of a clustering result for each algorithm with scale parameter $sca = 0$ and in its best conditions is presented in Figure 5.10.

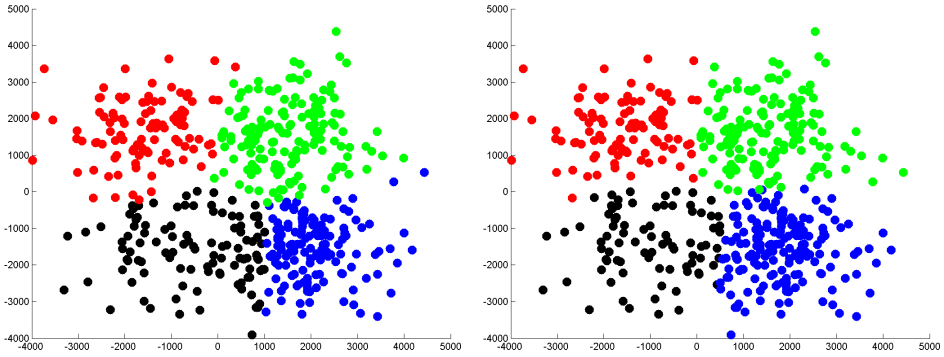
scale	Algorithm					
	FADDIS-a		FADDIS-m		FastMap FCM	NERFCM
	GK+Lapin	K	GK+Lapin	K	$K=4$	$K=4$
-5	**0.47/0.048**	4	**0.47/0.048**	4	0.44/0.045	0.44/0.047
0	**0.68/0.029**	4	**0.68/0.029**	4	0.61/0.096	0.64/0.058
5	0.83/0.022	4	0.79/0.031	4	**0.84/0.016**	**0.84/0.016**
10	0.91/0.029	4	0.87/0.039	3	**0.93/0.021**	**0.93/0.021**
20	0.98/0.022	4	0.91/0.041	3	**0.99/0.009**	**0.99/0.009**
50	0.99/0.019	4	0.77/0.20	2	**1/0**	**1/0**

Table 5.11 Adjusted Rand Index avg/std for all algorithms in best conditions and number of extracted clusters Mode for FADDIS



(a) FADDIS-a using GK/Lapin Clustering Result

(b) FADDIS-m using GK/Lapin Clustering Result



(c) FMFCM Clustering Result with $K=4$

(d) NERFCM Clustering Result with $K=4$

Figure 5.10 Three NERFCM clustering results for $sca=0$ and $K=\{3,4,5\}$

Analysing the Table 5.11, one can conclude that both versions of FADDIS are more robust

to noise than FMFCM or NERFCM, since the ARI index values for the former are better for $sca = \{-5, 0\}$. On the other hand, as the level of noise decreases (i.e $sca \geq 5$), FMFCM and NERFCM achieve better results.

Moreover, for each scenario of high level ($sca = \{-5, 0\}$) or low level ($sca = \{5, 10, 20, 50\}$) of noise, there is no distinction in the ARI index values between FADDIS-a and FADDIS-m or FMFCM and NERFCM. On the other hand, for higher values of $scale$, the FADDIS-m recovers less clusters than FADDIS-a, that always extracts the correct number of clusters.

However, the main conclusion that should be taken is that FADDIS-a with Gaussian Kernel and Lapin pre-processing is the winner, because even though the ARI index can be a little bit lower than the other cases, it is able to determine the correct number of clusters.

5.3 Study with Core Data Generator with Noise

5.3.1 The Fuzzy Cluster Core Data Generator

Given the dimension N of an entity set I, and the number of clusters K, the Fuzzy Cluster Core Data Generator (FCC DG) generates a $N \times N$ similarity data matrix G according to the underlying (FADDIS) model $A = UWU'$, as follows:

$$G = UWU' + \alpha E, \tag{5.1}$$

where:

- Fuzzy $N \times K$ membership matrix U is randomly generated using a fuzzy cluster generating procedure.
- Positive real valued $K \times K$ diagonal intensity matrix W, where each cluster intensity, w_k, is defined as the norm of the corresponding membership vector $\mathbf{u}_k$.
- Symmetric $N \times N$ real valued error matrix E, where each entry is independently generated from a Normal distribution $N(0,1)$.
- $\alpha \in [0,1]$ is the parameter that controls the level of Gaussian noise introduced into the model $A = UWU'$. Specifically, α is the standard deviation of the Gaussian noise introduced into the model.

Procedure to Construct the Clusters' Cores membership matrix U

Fixed the number K of cores covering the entire data set, I, the data generator builds each core $R_k, k = 1, \ldots, K$ by filling in each one with fuzzy memberships, such that: (a) the membership values of k-th fuzzy cluster $\mathbf{u}_k$ are very high at k-th core (e.g. $u_{ik} > 2/3$ for $i \in R_k$); and (b)

the fuzzy clusters form a probabilistic fuzzy partition at each entity $i \in I$ ($\sum_k u_{ik} = 1$). After all the membership vectors $\mathbf{u}_k$ are generated, that is the entire membership $N \times K$ matrix U is filled in, the norms of $\mathbf{u}_k$'s are computed and assigned as the clusters' intensities w_k, to "adjust" them to the additive fuzzy clustering model. So, the final membership matrix has its membership vectors $\mathbf{u}_k$ normalized.

1. Generating the Cores Sizes

To give some flavour to the FADDIS method of sequential extraction of clusters, the cores are generated of different random sizes, so that the size N_k of each next core R_k is at most half-size of the remaining part of the entity set, $k = 1, \ldots, K-2$. The first core is taken to be $N/2$ or less, and the last core's size is taken to complement the cumulative core size to N.

After the sizes N_k of all the K cores have been defined, K core membership $N_k \times K$ matrices U_k are defined independently of each other and then combined vertically into the final membership $N \times K$ matrix U.

2. Filling in each k-th core and corresponding matrix U_k

In order to define the $N_k \times K$ matrix U_k of k-th core, one starts filling in its k-th column, defined as a N_k-dimensional vector $\mathbf{a} = (a_1, \ldots, a_{N_k})$ of uniformly random values a_j such that, each a_j has a high membership value (e.g $a_j \in [2/3, 1]$). Then we need to fill in each entity $i (i = 1, \ldots, N_K)$ in U_k with random numbers $u_{ik'} (k' = 1, \ldots, K, k' \neq k)$, summing up to $1 - a_i$. The process proceeds as follows:

(i) if $K = 2$, there is only one column in U_k different from the k-th column already filled in. This other column is filled in with the complementary vector $\mathbf{1} - \mathbf{a}$.

(ii) if $K = 3$, another N_k uniform random vector $\mathbf{b}$ between 0 and 1 is generated and then

conditioned by $(1-\mathbf{a})$-values so that it becomes satisfying inequality $\mathbf{b} \leq (1-\mathbf{a})$, which guarantees that $\mathbf{a}+\mathbf{b} \leq 1$. Then this $\mathbf{b}$ is put into another column of the U_k matrix of k-th core, after which the third column is filled in with the supplement, $\mathbf{1}-\mathbf{a}-\mathbf{b}$.

(iii) In the case of $K > 3$, the data generation follows the random probabilistic generation, as follows. For each entity i from the k-th core, we need to generate $K-1$ probability values $p_1, \ldots, p_{K-1}$, totalling to $1-a_{ik}$.

To make them random, we first generate $K-2$ random values, each less than $1-a_{ik}$. Then we sort them in the ascending order $r_1 < \ldots < r_{K-2}$ to define the sought probabilities as the differences $p_{k+1} = r_{k+1} - r_k (k = 1, \ldots, K-3)$ with $p_1 = r_1$ and $p_{K-1} = 1 - a_i - r_{K-2}$.

5.3.2 Goal of the Study

The main goal of this study is to analyse the effectiveness of the selected relational fuzzy clustering algorithms (FADDIS-m [28], FADDIS-a [28], FastMap FCM [9], NERFCM [21]) to be able to recover the cluster structure of random generated data according to the FCC data generator. The FCC DG generates cluster structures, where subsets of entities belong much to one cluster (i.e core of the cluster) and quite nothing to others. In addition, different levels of generated Gaussian noise are added to the generated data. It is expected that one or both FADDIS versions can recover the cluster structure with more precision than the other algorithms, since this data generator was specifically designed following the FADDIS model.

The FCC DG provides as output a similarity matrix which will be given as input to the FADDIS algorithm after pre-processing with the Lapin transformation. In case of NERFCM and FMFCM, the input similarity data matrix is converted into dissimilarity data using the transformation defined by (2.3).

Particular attention will be given to the FADDIS algorithms in this experiments, whose

analysis will be made in terms of various aspects. The experiments will have in consideration the following items:

1. The number of found clusters in case of FADDIS, that will be analysed via the algorithm stop condition and threshold of relative contribution to the data scatter. On running the FMFCM and NERFCM algorithms, the input parameter number of clusters K, is set equal to the original number of clusters from which data had been generated.

2. The obtained fuzzy partitions will be compared against the original ones via the Adjusted Rand Index (3.27).

3. For each induced cluster $\hat{k}$ and corresponding generated cluster k, it will be analysed:

   (a) The percentage of unclustered entities.

   (b) The percentage of matching of the R_k cores ($K = 1, 2...k$).

4. For FADDIS in particular, for each induced cluster $\hat{k}$ and corresponding generated cluster k, it also will be analysed:

   (a) Characteristic of the error of membership recover (CEMR) between a generated cluster k with membership vector u_k and an induced membership vector $\hat{u}_k$. This measure is defined by:

$$CEMR(k) = \sum_{i=1}^{N} \left(\frac{u_i^2 + \hat{u}_i^2}{2} \right) \cdot \frac{2 \mid u_i - \hat{u}_i \mid}{\mid u_i + \hat{u}_i \mid}, \tag{5.2}$$

   subject to: $\sum_{i=1}^{N} u_k^2 = 1$, $\sum_{i=1}^{N} \hat{u}_k^2 = 1$, FADDIS algorithm constraints.

   This formula of the error was introduced based on the analysis of multiple experiments, in order to make the generated and induced membership values contribute

equally, that is to make the formula of the error symmetric in this regard. While $\frac{2|u_i-\hat{u}_i|}{|u_i+\hat{u}_i|}$ gives a symmetric relative error, the weighting factor $\frac{u_i^2+\hat{u}_i^2}{2}$ is introduced as a normalization factor in such way that the differences in memberships are less penalized in case of low membership values than in case of high membership values.

(b) Relative error between generated and induced intensities μ_k and $\hat{\mu}_k$:

$$REI(k) = \frac{\mid \mu_k - \hat{\mu}_k \mid}{\mid \mu_k + \hat{\mu}_k \mid}. \tag{5.3}$$

As the previous formula, REI was built from the analysis of multiple experiments. It is a relative error where both induced and generated cluster intensities contribute equally. This formula is also symmetric.

5.3.3 Setting of the Experiments

The datasets have been generated organized in three groups covering 3 distinct numbers of clusters: $K = \{3,5,7\}$. The experiments had been conducted with the following setting:

- The total number of entities $N = \{50,100,200,400,700\}$
- The α values that determine the standard deviation of Gaussian noise were $\alpha = \{0,0.02,0.05,0.1,0.15,0.25,0.5,1,5\}$.
- For each value of K, 10 distinct datasets had been generated for each pair (N,α) fixed, resulting in a total of 450 datasets for each K value, and finally a total of 1350 datasets.
- The threshold of stop condition 'SC_2' (the cluster contribution is too small) was fine tuned for each group of datasets (K,N). These values are presented in Table 5.12 for the case of FADDIS-a and Table 5.13 of the case of FADDIS-m.

	Dataset size N				
K	50	100	200	400	700
3	0.01	0.008	0.003	0.0015	0.0009
5	0.015	0.007	0.003	0.0015	0.0009
7	0.01	0.008	0.003	0.0015	0.0009

Table 5.12 Minimum contribution threshold ε fine tuning for each case for FADDIS-a

	Dataset size N				
K	50	100	200	400	700
3	0.01	0.008	0.002	0.0015	0.0006
5	0.01	0.005	0.0001	0.0005	0.0009
7	0.005	0.002	0.001	0.0003	0.0008

Table 5.13 Minimum contribution threshold ε fine tuning for each case for FADDIS-m

5.3.4 Summary and Discussion of the Results

In this section the results for $K = 5$ will be presented and discussed. Results for $K = 3$ and $K = 7$ can be found in Appendix B. A summary of all the results are presented in the end of this section.

Applying the Visual Assessment Tool (VAT) for cluster tendency [7] to a generated dataset with $K = 3$, $N = 200$ and $\alpha = 0$, the result in Figure 5.11 is obtained, where a cluster structure with 3 clusters can be identified.

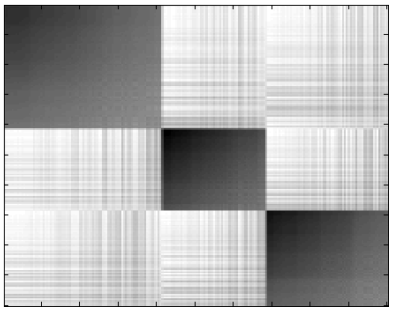
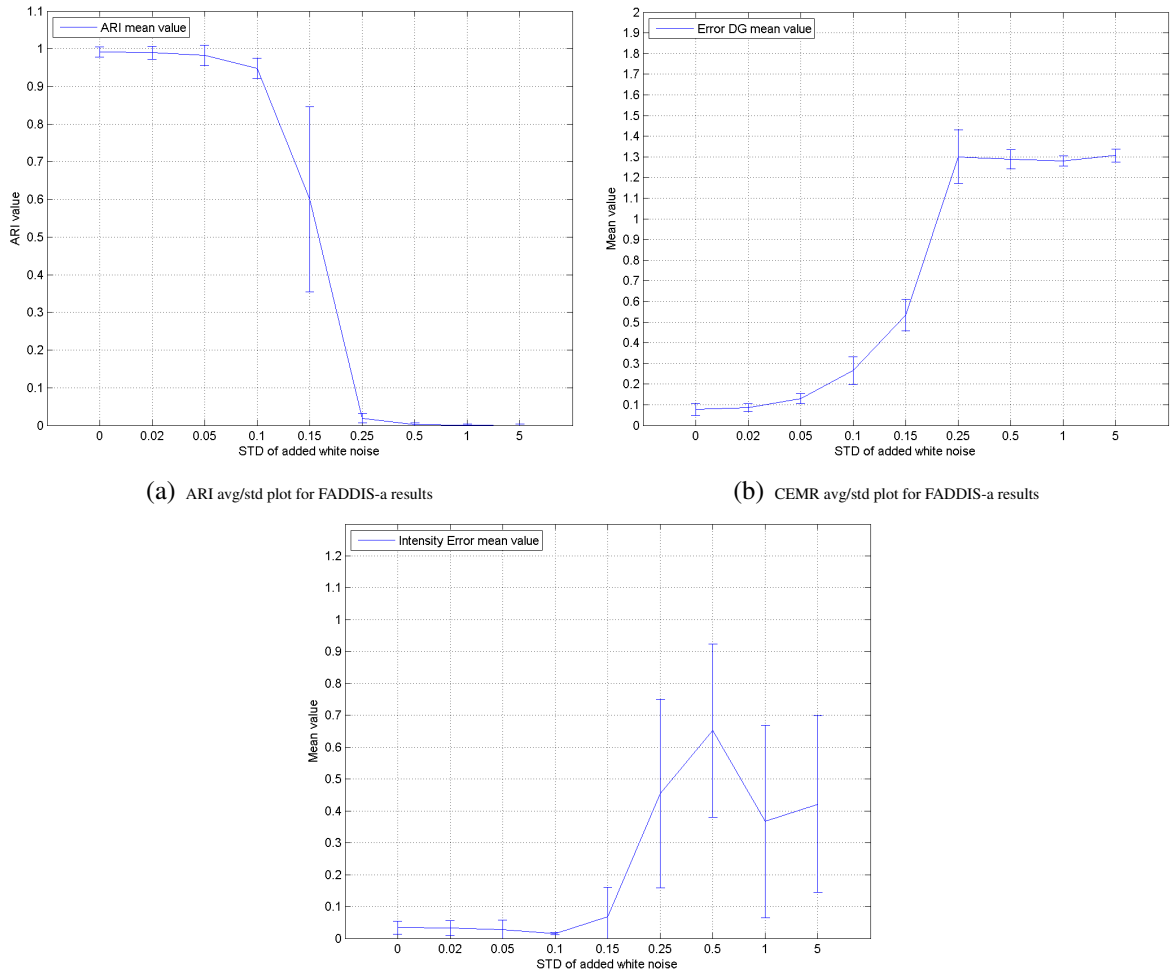


Figure 5.11 VAT applied to a FCC generated dataset with $K = 3$, $N = 200$ and $\alpha = 0$

In a first stage the results were analysed in terms of mean and standard deviation of the 10

datasets for each case.

As an example of the ARI, REI and CEMR results for all α values, the obtained plots for FADDIS-a for $N = 400$ and $K = 5$ are shown in Figure 5.12.



(a) ARI avg/std plot for FADDIS-a results

(b) CEMR avg/std plot for FADDIS-a results

(c) REI avg/std plot for FADDIS-a results

Figure 5.12 FADDIS-a plots for $K = 5$ and $N = 400$ for ARI, CEMR and REI

As can be seen from the Figure B.9(a), the obtained ARI index values for $\alpha \in [0, 0.1]$ are very close to the best value ($ARI = 1$). On the other hand, both CEMR and REI values have low values for the same α interval. Therefore, it can be concluded that FADDIS is robust to noise and very effective with acceptable results in the interval $\alpha = [0, 0.1]$. The same kind of results

were obtained for FADDIS-m as well as for the other values of the number of entities N and number of clusters K and can be found in Appendix B.

Knowing this, the mean and standard deviation values will have in account all experiment results for the datasets in the interval $\alpha \in [0, 0.1]$.

Firstly, some FADDIS outputs will be analysed, since there are some properties that are exclusive for these algorithms.

In the Table 5.14, the extracted number of clusters is analysed: the mode of the number of esctracted clusters and avg/std of the percentages of the experiments in which the correct number of clusters was recovered. As it can be seen, FADDIS-a can always recover more successfully the correct number of clusters than FADDIS-m.

	Algorithm			
$N, K = 5$	FADDIS-a		FADDIS-m	
	Mode	mean(%)/std	Mode	mean(%)/std
50	5	**57.5/25**	5	42.5/17.1
100	5	**75/31.1**	4	40/14.1
200	5	**80/21.6**	5	37.5/15
400	5	**92.5/15**	5	52.5/17.1
700	5	**75/17.3**	5	55/19.1

Table 5.14 Mode and Percentage avg/std of correct extracted clusters for std of Gaussian noise=[0, 0.1] for $K = 5$

For all dataset sizes N, FADDIS-a determines the correct number of clusters with a high success rate mean.

Calculating the Characterist error of membership recover (CEMR), the obtained avg/std are shown in Table 5.15. The minimum value is presented in boldface.

As for the Relative Error of the Intensities (REI), FADDIS-a obtains similar and almost always slightly lower error values compared to FADDIS-m, and this difference can be seen in Table 5.16.

In a second stage, the ARI and Crisp Core Matching will be analysed for all algorithms. The

	Algorithm	
$N, K = 5$	FADDIS-a	FADDIS-m
50	**0.16/0.09**	0.19/0.1
100	**0.14/0.08**	**0.14/0.07**
200	0.14/0.08	**0.13/0.09**
400	**0.14/0.09**	**0.14/0.09**
700	0.14/0.07	**0.13/0.07**

Table 5.15 CEMR avg/std for std of added Gaussian noise=[0, 0.1] for K=5

	Algorithm	
$N, K = 5$	FADDIS-a	FADDIS-m
50	**0.07/0.034**	0.1/0.1
100	**0.06/0.031**	**0.06/0.03**
200	**0.05/0.038**	0.07/0.07
400	**0.05/0.036**	0.07/0.05
700	0.1/0.07	**0.09/0.07**

Table 5.16 REI avg/std for std of added Gaussian noise=[0, 0.1] for K=5

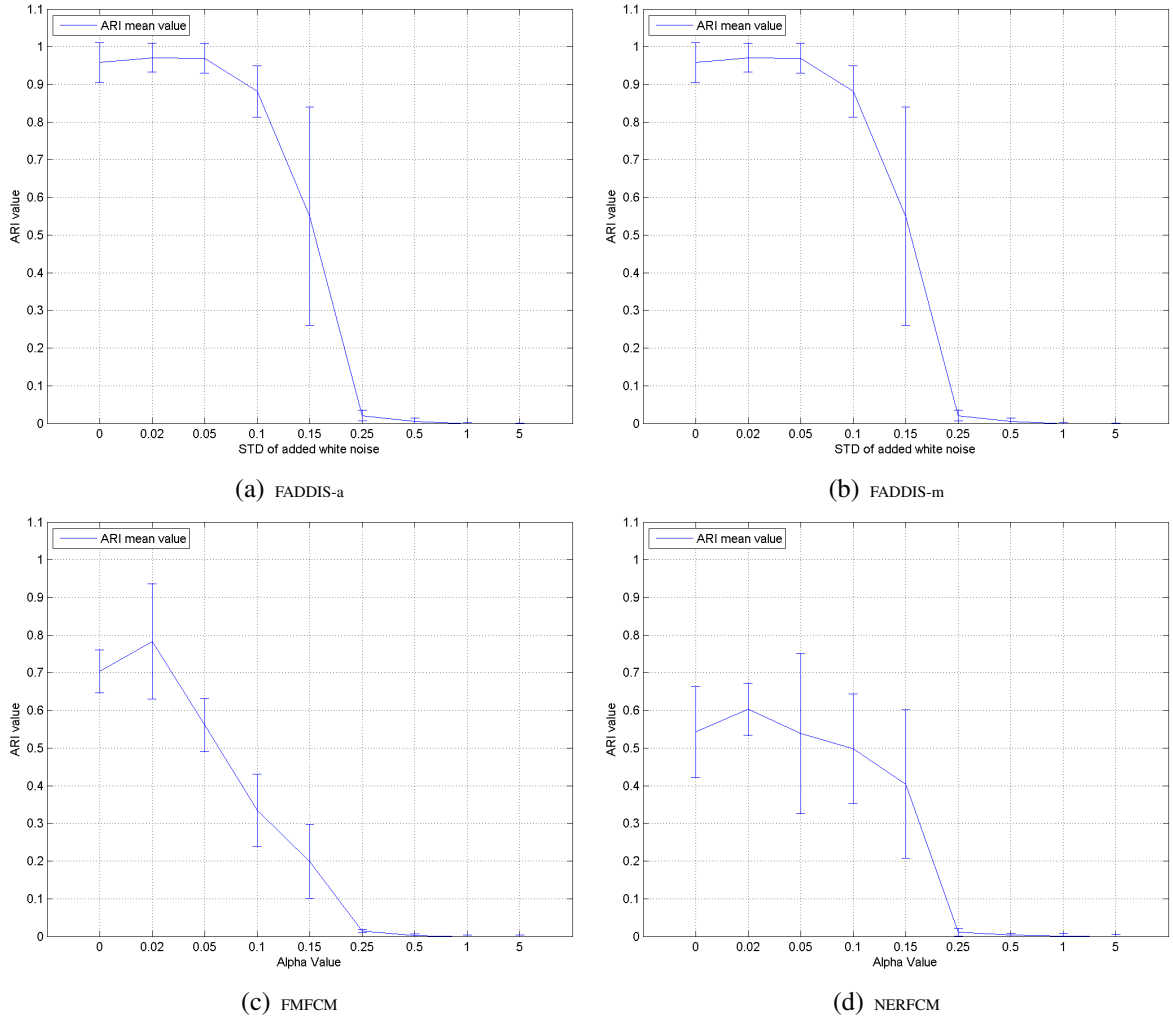
clustering result of FADDIS-a, FADDIS-m, FastMap FCM and NERFCM will be compared.

Four plots of the ARI std/mean for each algorithm are shown in Figure 5.13.

The four algorithms are relatively effective for $\alpha \in [0, 0.1]$, and so the summary analysis will be made for this interval of the std of Gaussian noise added to data. The Adjusted Rand Index compares the obtained partitions to the original. The mean and standard deviation of the ARI are shown in the Table 5.17.

	Algorithm			
Dataset Size	FADDIS-a	FADDIS-m	FastMap FCM	NERFCM
50	**0.9/0.16**	**0.9/0.15**	0.77/0.2	0.62/0.17
100	**0.96/0.05**	0.93/0.08	0.72/0.16	0.67/0.16
200	**0.96/0.05**	0.92/0.08	0.65/0.18	0.69/0.12
400	**0.98/0.03**	0.95/0.06	0.6/0.2	0.55/0.15
700	**0.95/0.07**	0.93/0.09	0.51/0.22	0.39/0.15

Table 5.17 Summary Table for ARI avg/std for std of added Gaussian noise=[0, 0.1] for all algorithms in best conditions for K=5



(a) FADDIS-a (b) FADDIS-m

(c) FMFCM (d) NERFCM

Figure 5.13 ARI avg/std plots for all algorithms for $K = 5$ and $N = 400$

Once again, FADDIS-a returns better ARI results than the other algorithms, with FADDIS-m getting slightly lower values. FMFCM and NERFCM have similar ARI values, but much lower than both versions of FADDIS. Analysing the Figure 5.13 and summary Table 5.17 it is important to point out the differences in the values of the standard deviation. It can be observed from the plots that the FMFCM and NERFCM tolerance to noise is much lower.

The Crisp Core Matching function, defuzzifies the original partitions and the obtained partitions, and returns the percentage of the original cluster's entities that were grouped into the

correct clusters, i.e, gives the percentage of the Cores that were recovered in the clustering result. A percentage is obtained for each core and a mean and standard deviation is calculated in Table 5.18.

	Algorithm			
Dataset Size	FADDIS-a	FADDIS-m	FastMap FCM	NERFCM
50	**92.9/7.1**	89.4/6.7	79.3/8.5	71/4.9
100	**97.7/1.4**	95.6/2.5	75.7/5	77.8/3.7
200	**97.9/1.8**	92.8/4.5	68.8/6.8	79.2/0.6
400	**98.8/1.2**	95.2/3.3	65.8/8.8	67.9/2.5
700	**97.4/2**	93.5/2.4	62/9.8	57.4/2.4

Table 5.18 Crisp Core Matching (%) avg/std for std of added Gaussian noise=[0, 0.1] for K=5

As it can be concluded, FADDIS-a succeeds in a better recovering of the original Cores, as its results have really high matching values. FADDIS-m also obtains high values even though lower than FADDIS-a. FMFCM and NERFCM return much lower values than both versions of FADDIS.

When comparing both versions of FADDIS, the most noticeable difference lies in the percentage of tests that extracted the correct number of clusters. Analysing the most common stop conditions (described in p. 45):

1. *FADDIS-a*: The stop condition for all analysed experiments (i.e $K = \{3,5,7\}$) with FADDIS-a was 'SC_2', the minimum contribution of the cluster to the data scatter.

2. *FADDIS-m*: The most common stop condition is 'SC_2', in 63% of the cases, and with 33% of the datasets was the lack of positive weights at spectral clusters, 'SC_1'. In addition, in 4% of the tests the stop condition was 'SC_4'.

One can conclude that FADDIS-a is more coherent when extracting clusters than FADDIS-m, since the latter had worse results for the correct number of extracted clusters. Comparing

each *i*-th obtained cluster contribution of FADDIS-m for each dataset, it could be concluded that the values are not similar between each other. This difficulted the fine tuning of the minimum contribution threshold ε and may have been the cause for the low success rate in this item by FADDIS-m.

The summary ARI results for the whole sets of data involved in this study (i.e for $K = \{3,5,7\}$) and algorithms is shown in Table 5.19 and for the Crisp Core Matching in Table 5.20. Detail of the results are presented in Appendix B.

		Algorithm			
	Dataset Size	FADDIS-a	FADDIS-m	FastMap FCM	NERFCM
$K = 3$	50	**0.9/0.25**	**0.9/0.19**	0.77/0.26	0.67/0.26
	100	**0.92/0.12**	**0.92/0.13**	0.73/0.21	0.62/0.25
	200	**0.83/0.18**	0.82/0.19	0.69/0.28	0.43/0.29
	400	**0.97/0.047**	0.95/0.07	0.71/0.3	0.21/0.15
	700	**0.8/0.23**	**0.8/0.22**	0.52/0.33	0.14/0.12
$K = 5$	50	**0.9/0.16**	**0.9/0.15**	0.77/0.2	0.62/0.17
	100	**0.96/0.05**	0.93/0.08	0.72/0.16	0.67/0.16
	200	**0.96/0.05**	0.92/0.08	0.65/0.18	0.69/0.12
	400	**0.98/0.03**	0.95/0.06	0.6/0.2	0.55/0.15
	700	**0.95/0.07**	0.93/0.09	0.51/0.22	0.39/0.15
$K = 7$	50	**0.94/0.12**	0.86/0.21	0.67/0.18	0.6/0.19
	100	**0.92/0.16**	0.87/0.2	0.65/0.19	0.69/0.19
	200	**0.97/0.06**	0.91/0.16	0.64/0.17	0.67/0.19
	400	**0.97/0.04**	0.91/0.09	0.54/0.17	0.6/0.14
	700	**0.97/0.04**	0.94/0.05	0.48/0.2	0.56/0.15

Table 5.19 Summary Table for ARI avg/std for std of added Gaussian noise=[0, 0.1] for all algorithms in best conditions for $K = \{3,5,7\}$

As it can be seen, FADDIS-a always obtains the highest ARI values in all cases. FADDIS-m also obtains high values. Both FMFCM and NERFCM always get much lower mean ARI index values and higher std than both versions of FADDIS. The same happens with the Crisp Core Matching, where both FADDIS algorithms always obtain the highest score, mostly FADDIS-a. FMFCM and NERFCM also obtain acceptable matching results, but still lower than those of

		Algorithm			
	Dataset Size	FADDIS-a	FADDIS-m	FastMap FCM	NERFCM
$K=3$	50	**92/2.7**	91/1.1	87.9/5.4	81.8/1.3
	100	**85.8/4.5**	84.7/5.9	84.8/7	78.5/2.7
	200	78.2/5.3	**82.8/2.3**	83.7/7.6	68.1/0.8
	400	**98.5/1.3**	95.8/2.8	86/15.4	56.4/2.9
	700	79.4/4.1	**84.9/1.7**	75.1/17.4	50.2/3.9
$K=5$	50	**92.9/7.1**	89.4/6.7	79.3/8.5	71/4.9
	100	**97.7/1.4**	95.6/2.5	75.7/5	77.8/3.7
	200	**97.9/1.8**	92.8/4.5	68.8/6.8	79.2/0.6
	400	**98.8/1.2**	95.2/3.3	65.8/8.8	67.9/2.5
	700	**97.4/2**	93.5/2.4	62/9.8	57.4/2.4
$K=7$	50	**95.2/7.9**	88.6/13	68.2/7.2	70/8.4
	100	**93.8/11.2**	89.7/12.3	64.4/8	73.8/9
	200	**97.8/3**	92.1/10.1	63.3/6	73.8/9.6
	400	**97.5/3**	91.3/3.3	56.8/4.7	67.8/3.6
	700	**97.5/2.3**	94.4/3.1	51/8.5	65.1/3.8

Table 5.20 Summary Table of Crisp Core Matching (%) avg/std for std of added Gaussian noise=[0, 0.1] for all algorithms in best conditions for $K = \{3,5,7\}$

FADDIS.

The minimum contribution threshold was fine tuned for each algorithm and each combination of K and N. In the case of FADDIS-m, the fine tuning was not enough to ensure a higher rate for extracting the correct number of clusters for all datasets as have been stated before. Better results might have been achieved by fine tuning this parameter for each dataset. In the case of FADDIS-a, the fine tuning was adequate to obtain the correct number of extracted clusters in more than 50% of the cases.

This Data Generator was designed for the FADDIS model, and as expected both versions of the algorithm return better and more precise clustering results than either FMFCM or NERFCM. Such was verified using the Crisp Core Matching function and the Adjusted Rand Index, where both versions of FADDIS returned very high values. Analysing the results for CEMR and REI in case of FADDIS, both algorithms obtain very low values for these error measures for

$\alpha \in [0, 0.1]$, which indicates good results in the recovery of the cluster structure. Values lower than 0.2 are considered to be good results.

5.4 Study with Benchmark Datasets

5.4.1 Goal of the study

The goal of this study is to analyse some properties of the two FADDIS versions (Section 4.2) with Benchmark datasets. All datasets were analysed and clustered in previous work [21] [55], therefore some results are provided in order to compare to the obtained with FADDIS. In addition, the data will be clustered using the *FastMap Fuzzy C-Means* (Section 3.4) and the *Non-Euclidean Relacional Fuzzy C-Means* (Section 3.3) so that the results can be compared.

In order to achieve this, some items will be taken under consideration:

1. The number of found clusters by FADDIS and the respective stop condition;
2. The percentage of unclustered data by FADDIS;
3. Misclassification error according to the confusion matrix;
4. Quality of found partitions according to the ARI index (3.28).

5.4.2 Setting of the Experiments

In these experiments, the object data has been transformed into relational data using the conventional proximity (2.1) or the Gaussian Kernel (**??**).

Each data set is subject to the following pre-processing:

1. Conventional proximity measure in case of application of the FMFCM and NERFCM;
2. Gaussian Kernel in case of application of two versions of FADDIS will be applied, along with the application or not of the Lapin transformation;

This way a total of 6 clustering results will be presented and analysed.

A stop criteria for the FADDIS algorithm will be fine tuned for each test: the minimum cluster's relative contribution to the data scatter ε (stop condition SC_2, Section 4.2). The threshold value was fine tuned for each dataset, due to their different nature.

The two versions of FADDIS, FADDIS-a and FADDIS-m, correspond to the versions "a" and "m" explained in Section 4.2.

Since the FMFCM and NERFCM algorithms have the number of clusters K as input parameter, the values $\{K-1, K, K+1\}$ were used for the cases that $K > 2$ and $\{K,K+1\}$ when $K = 2$. For these algorithms, all object data is transformed into dissimilarity data using the Euclidean distance measure. As for the Davé and Sen datasets [12], namely Fat Oil, Country and Microcomputer data, since these are already in the format of relational matrices, they can be used directly when in dissimilarity form or transformed into one when in format of similarity matrix as will be explained.

The benchmark datasets to be analysed are: Iris, Breast Cancer Wisconsin, Wine from the UCI Machine Learning Repository [55] and Fat Oil, Country, Microcomputer Data taken from Davé and Sen [12].

In the next section the Iris data set experiments are presented as well as the summary of the analysis of all the experimental results. A detailed description of the experiments for the other datasets is presented in Appendix C.

5.4.3 Summary and Discussion of the Results

The Iris, Cancer and Wine Data sets, taken from UCI [55], are classified, and the results validation will be based on the original labels.

The Country, Fat Oil and Microcomputer Data do not have a classification, and in this

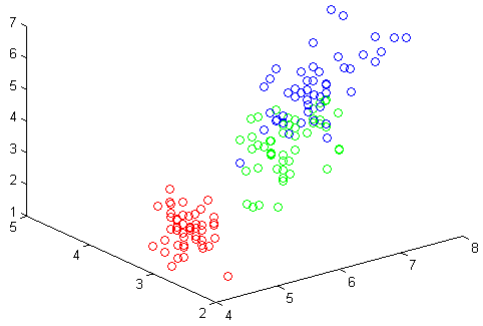


Figure 5.14 Original Iris Dataset Classification

work, some results were retrieved from previous work of Davé and Sen [12]. The given data is relational; therefore it will be in the format of a dissimilarity or similarity matrix. Depending on the used algorithm, the used conversion method is described in equation 2.3 in Section 2.2.1, to convert from similarity to dissimilarity or vice versa.

In the above cited article, all data was clustered with Fuzzy Relational Clustering (FRC), and the FADDIS, NERFCM and FMFCM results will be compared according to the results obtained by FRC. Also a second test is made using the Gaussian Kernel for each dataset in case of FADDIS.

The Iris dataset is organized in 3 classes with 50 objects each, having a total of 150 objects. The data comes in the form of object data; therefore some similarity measure must be used in order to obtain relational data.

An example plot of the dataset using the benchmark original labels is shown in figure 5.14.

The obtained results for FMFCM and NERFCM using $K = 2$, $K = 3$ and $K = 4$, as the input parameter for the number of clusters, are presented, respectively, in tables 5.21 and 5.22.

<table>
<tr>
<td>K = 2
Confusion Matrix:
<table><tr><td></td><td></td><td colspan="3">Original Classes</td></tr><tr><td></td><td></td><td>1</td><td>2</td><td>3</td></tr><tr><td rowspan="2">Predicted Classes</td><td>1</td><td>50</td><td>3</td><td>0</td></tr><tr><td>2</td><td>0</td><td>47</td><td>50</td></tr></table>Mismatch Error: 0.35
Adjusted Rand Index: 0.54</td>
<td>K=3
Confusion Matrix:
<table><tr><td></td><td></td><td colspan="3">Original Classes</td></tr><tr><td></td><td></td><td>1</td><td>2</td><td>3</td></tr><tr><td rowspan="3">Predicted Classes</td><td>1</td><td>50</td><td>0</td><td>0</td></tr><tr><td>2</td><td>0</td><td>47</td><td>13</td></tr><tr><td>3</td><td>0</td><td>3</td><td>37</td></tr></table>Mismatch Error: 0.1
Adjusted Rand Index: 0.73</td>
<td>K=4
Confusion Matrix:
<table><tr><td></td><td></td><td colspan="3">Original Classes</td></tr><tr><td></td><td></td><td>1</td><td>2</td><td>3</td></tr><tr><td rowspan="4">Predicted Classes</td><td>1</td><td>50</td><td>3</td><td>0</td></tr><tr><td>2</td><td>0</td><td>29</td><td>1</td></tr><tr><td>3</td><td>0</td><td>0</td><td>27</td></tr><tr><td>4</td><td>0</td><td>21</td><td>22</td></tr></table>Mismatch Error: 0.29
Adjusted Rand Index: 0.62</td>
</tr>
</table>

Table 5.21 FMFCM results for Iris Data

<table>
<tr>
<td>K = 2
Confusion Matrix:
<table><tr><td></td><td></td><td colspan="3">Original Classes</td></tr><tr><td></td><td></td><td>1</td><td>2</td><td>3</td></tr><tr><td rowspan="2">Predicted Classes</td><td>1</td><td>50</td><td>3</td><td>0</td></tr><tr><td>2</td><td>0</td><td>47</td><td>50</td></tr></table>Mismatch Error: 0.35
Adjusted Rand Index: 0.54</td>
<td>K=3
Confusion Matrix:
<table><tr><td></td><td></td><td colspan="3">Original Classes</td></tr><tr><td></td><td></td><td>1</td><td>2</td><td>3</td></tr><tr><td rowspan="3">Predicted Classes</td><td>1</td><td>50</td><td>0</td><td>0</td></tr><tr><td>2</td><td>0</td><td>46</td><td>9</td></tr><tr><td>3</td><td>0</td><td>4</td><td>41</td></tr></table>Mismatch Error: 0.09
Adjusted Rand Index: 0.77</td>
<td>K=4
Confusion Matrix:
<table><tr><td></td><td></td><td colspan="3">Original Classes</td></tr><tr><td></td><td></td><td>1</td><td>2</td><td>3</td></tr><tr><td rowspan="4">Predicted Classes</td><td>1</td><td>50</td><td>0</td><td>0</td></tr><tr><td>2</td><td>0</td><td>29</td><td>1</td></tr><tr><td>3</td><td>0</td><td>0</td><td>35</td></tr><tr><td>4</td><td>0</td><td>4</td><td>14</td></tr></table>Mismatch Error: 0.13
Adjusted Rand Index: 0.68</td>
</tr>
</table>

Table 5.22 NERFCM results for Iris Data

Analysing the tables, it can be concluded that the best input number of clusters K for both FMFCM and NERFCM, is $K = 3$, as the obtained ARI index and mismatch error are best for that value. The original classification consists on 3 classes, and the results confirm that both algorithms cluster the data with better results using the original number of classes.

As for FADDIS versions, the Gaussian Kernel was used. It also was studied the effect of the Lapin transformation pre-processing. For each algorithm, a fine tuning of the minimum cluster contribution to the data scatter was made.

Stop Criteria Tuning:

- FADDIS-a: $\varepsilon = 0.001$
- FADDIS-m: $\varepsilon = 0.005$

For each FADDIS version and pre-processing (Gaussian Kernel with and without the Lapin transformation), the results are as follows:

After applying the algorithm, the following results are obtained:

FADDIS-a (Gaussian Kernel without Lapin)

· Number of clusters: 5

· Stop Condition: 'SC_4' - Maximum number of clusters reached

· Confusion Matrix in table 5.23 with Mismatch error: 0.37

· Unclustered data: 0%

· Adjusted Rand Index: 0.51

· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table 5.24

		Original Classes		
		1	2	3
Predicted Classes	1	41	0	0
	2	0	27	1
	3	0	0	26
	4	0	23	14
	5	9	0	9

Table 5.23 Confusion Matrix Obtained from original classification and clustering results of Iris Data with FADDIS-a using the Gaussian Kernel without Lapin

		Contribution	Intensity
Extracted Clusters	1	0.4713	42.391
	2	0.3628	37.192
	3	0.0384	12.11
	4	0.0351	11.565
	5	0.0023	2.9466

Table 5.24 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-a for the Iris Data using Gaussian Kernel without Lapin

After careful analysis of the results, it can be concluded that FADDIS-a can recover the correct number of clusters using the Lapin transformation pre-processing. It also returns the

FADDIS-m (Gaussian Kernel without Lapin)

· Number of clusters: 5

· Stop Condition: 'SC_2' - Cluster contribution is too small

· Confusion Matrix in table 5.25 with Mismatch error: 0.35

· Unclustered data: 0%

· Adjusted Rand Index: 0.6

· Contribution to the data scatter and Intensity of the extracted clusters in table 5.26

		Original Classes		
		1	2	3
Predicted Classes	1	50	0	0
	2	0	27	1
	3	0	0	20
	4	0	23	14
	5	0	0	15

Table 5.25 Confusion Matrix Obtained from original classification and clustering results of Iris Data with FADDIS-m using the Gaussian Kernel without Lapin

		Contribution	Intensity
Extracted Clusters	1	0.4713	42.391
	2	0.3628	37.192
	3	0.0385	12.11
	4	0.0351	11.565
	5	0.0002	0.8161

Table 5.26 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-m for the Iris Data using Gaussian Kernel without Lapin

best result for the ARI index as well as for the mismatch error. Interesting results are obtained for FADDIS-m algorithm using Lapin, where 2 clusters are recovered. This is noticeable, since in the clustering community [28] some argue that this dataset should have only 2 clusters, which are exactly the ones determined by FADDIS-m with Lapin transformation. If this is taken under consideration, then it constitutes the best result with ARI index of 1 and mismatch error of 0.

FADDIS-a (Gaussian Kernel with Lapin)

· Number of clusters: 3

· Stop Condition: 'SC_2' - Cluster contribution is too small

· Confusion Matrix in table 5.27 with Mismatch error: 0.16

· Unclustered data: 0%

· Adjusted Rand Index: 0.644

· Contribution to the data scatter and Intensity of the extracted clusters in table 5.28

		Original Classes		
		1	2	3
Predicted Classes	1	50	0	0
	2	0	50	24
	3	0	0	26

Table 5.27 Confusion Matrix Obtained from original classification and clustering results of Iris Data with FADDIS-a using the Gaussian Kernel with Lapin

		Contribution	Intensity
Extracted Clusters	1	0.4492	62.652
	2	0.1029	29.983
	3	0.0003	1.5593

Table 5.28 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-a for the Iris Data using Gaussian Kernel with Lapin

FADDIS-m (Gaussian Kernel with Lapin)

· Number of clusters: 2

· Stop Condition: 'SC_1' - No positive weights at spectral clusters

· Confusion Matrix in table 5.29 with Mismatch error: 0.33

· Unclustered data: 0%

· Adjusted Rand Index: 0.568

· Contribution to the data scatter and Intensity of the extracted clusters in table 5.30

		Original Classes		
		1	2	3
Predicted Classes	1	50	0	0
	2	0	50	50

Table 5.29 Confusion Matrix Obtained from original classification and clustering results of Iris Data with FADDIS-m using the Gaussian Kernel with Lapin

		Contribution	Intensity
Extracted Clusters	1	0.4492	62.652
	2	0.1029	29.983

Table 5.30 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-m for the Iris Data using Gaussian Kernel with Lapin

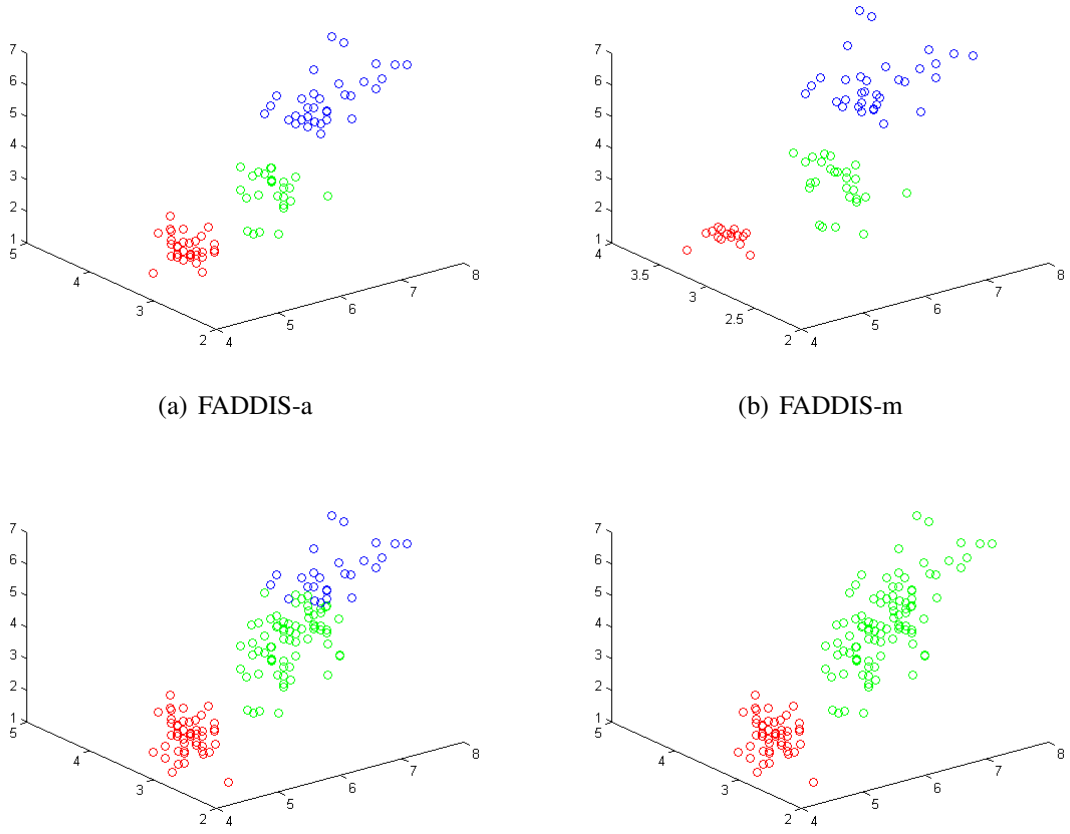
All the plots for the presented FADDIS experiments are shown in figure 5.15.

For this dataset, FADDIS-m appears to obtain the best results, followed by NERFCM and FMFCM. In the case of $K = 2$ for FADDIS-m using the Gaussian Kernel with the Lapin transformation, it achieves the best results for the ARI index of 1, which is a very interesting and good result. Also, when analysing FMFCM and NERFCM for $K = 2$, it can be seen that both have similar results as FADDIS-m in the mentioned case, but still lower ARI index values. Following, a summary table for all the studied datasets and algorithms in their best pre-processing or input parameters is presented in Table 5.31. The best results are marked with boldface.

Since in three datasets the considered original classifications were taken from the FRC results in Davé and Sen [12], the algorithm that was extended to obtain NERFCM, high results are expected for NERFCM. The exception is the Microcomputer Data, for which the considered original classification, also taken from the same source, was a result of FRC with different pre-processing, hence the low NERFCM results.

All algorithms appear to recover completely the cluster structure from the Fat Oil data, as the ARI index is always 1.

Analysing the behaviour of FADDIS algorithms when recovering the original number of



(a) FADDIS-a

(b) FADDIS-m

(c) FADDIS-a with Lapin

(d) FADDIS-m with Lapin

Figure 5.15 Iris Data Clustering Results for FADDIS with Gaussian Kernel

clusters, in some cases, for Iris and Fat Oil datasets, both versions extract the correct number of clusters. On the other hand, for the Cancer dataset, both extracted 4 clusters instead of 2, but obtained the best ARI index values. This is due to the 2 extra extracted clusters being populated with 1 entity each. As for NERFCM and FMFCM, both obtain the best results with the original number of clusters as input for most cases.

Notice that throughout the experiments results the mismatch error is concordant with the ARI index value, as the latter is calculated based on the confusion matrix as well.

No clear conclusions can be made from the results from these experiments. Depending

Dataset *Original K*	Algorithm			
	FADDIS-a	FADDIS-m	FastMap FCM	NERFCM
Iris $K = 2/3$	**output K = 3** ARI = 0.64 ME = 0.16	**output K = 2** **ARI = 1** **ME = 0**	**input K = 3** ARI = 0.73 ME = 0.1	**input K = 3** ARI = 0.77 ME = 0.09
Cancer $K = 2$	output K = 4 **ARI = 0.89** **ME = 0.03**	output K = 4 **ARI = 0.89** **ME = 0.03**	**input K = 2** ARI = 0.83 ME = 0.04	**input K = 2** ARI = 0.88 ME = 0.03
Wine $K = 3$	output K = 5 ARI = 0.34 ME = 0.43	output K = 6 ARI = 0.33 ME = 0.45	input K = 2 ARI = 0.36 ME = 0.34	input K = 2 **ARI = 0.4** **ME = 0.33**
FatOil $K = 2$	**output K = 2** **ARI = 1** **ME = 0**	**output K = 2** **ARI = 1** **ME = 0**	**input K = 2** **ARI = 1** **ME = 0**	**input K = 2** **ARI = 1** **ME = 0**
Country $K = 3$	output K = 5 ARI = 0.65 ME = 0.33	output K = 6 ARI = 0.48 ME = 0.42	**input K = 3** ARI = 0.73 ME = 0.08	**input K = 3** **ARI = 1** **ME = 0**
Microcomputer $K = 4$	output K = 5 ARI = 0.66 ME = 0.17	output K = 2 ARI = 0.5 ME = 0.42	**input K = 4** **ARI = 0.75** **ME = 0.08**	input K = 3 ARI = 0.68 ME = 0.25

Table 5.31 Summary Table of all benchmark datasets for all algorithms in best conditions

on the analysed dataset, different algorithms recover the original cluster structure with more precision.

5.5 Outlook

In this section, an overall comparison of the algorithms results will be presented. This will be based essentially on the Bivariate Normal Distribution data generator (BND DG) and Fuzzy Core Clusters data generator (FCC DG) results, since almost no general conclusions could be made from the study with Benchmark datasets.

Analysing results from the data generators using FastMap FCM and NERFCM, both algorithms return good clustering partitions as can be confirmed by the ARI index and EXB.

In particular, for the BND DG, both algorithms presented similar ARI index values for all scale values. However, in situations with more noise, i.e for $scale \in \{-5, 0\}$, the best results for both measures were obtained with $K = 3$ when the natural number of clusters is 4. Analysing with more detail the values for the EXB index, it can be seen that the obtained values for the NERFCM are lower than that of FMFCM, which may indicate that the clustering quality of the NERFCM results is better.

On the other hand, analysing the results for the FCC DG, the algorithms were run with the original number of clusters as input. The FastMap in theses experiments returned slightly higher ARI index values than NERFCM for $K = \{3, 5\}$. Notice that the ARI for FMFCM decreases slightly as the number of clusters K increases. With $K = 7$, NERFCM gets the highest ARI values for most cases. Both algorithms are not very robust to noise, as the ARI index start decreasing from $\alpha = 0.02$.

Comparing the results for all algorithms, it can be concluded that both FADDIS versions obtain better clustering partitions than the other algorithms in most of the cases where more noise is introduced. In the case of the BND DG, the FADDIS algorithms get better ARI index results for $scale \in \{-5, 0\}$ (more introduced noise), while FMFCM and NERFCM both get

higher results (ARI ≥ 0.84) for $scale \in \{5, 10, 20, 50\}$, where the cluster structure is more well defined. It should be noted that FADDIS-m recovers less clusters as the *scale* value increases, such did not happen with FADDIS-a. These results indicate that FADDIS-a performed better than FADDIS-m for those parameter values.

For the FCC DG, FADDIS-a almost always returns the highest ARI index values (≥ 0.9) and recovers the natural number of clusters. The FADDIS-m algorithm also obtains high ARI index values but slightly lower that those of FADDIS-a. It also should be stressed that FADDIS-m is more unstable than FADDIS-a, since the number of extracted clusters varies more frequently when comparing the output of each dataset for each pair (N, K). Analysing FMFCM and NERFCM, it can be concluded that both return worse results than the FADDIS versions, since the ARI index mean values for FMFCM and NERFCM are always lower. Also these values start decreasing with lower degree of noise $\alpha > 0.02$, when for FADDIS this only happens for $\alpha > 0.1$, which confirms that FADDIS is more robust to noise. In addition, the standard deviation of the ARI index for NERFCM and FMFCM is higher than those of both versions of FADDIS, therefore the results of the latter are more stable.

General conclusions and future work will be presented in the next chapter.

6 . Conclusion and Future work

An experimental comparative study has been performed between the Fuzzy Additive Spectral Clustering (FADDIS), a Sequential Relational Fuzzy clustering algorithm, and two versions of Parallel Relational Fuzzy clustering algorithms, the FastMap FCM and NERFCM.

Particular attention was made on the study of the 2 versions of the FADDIS algorithms. The following conclusions for the FADDIS algorithms can be made from essentially the results obtained with the two studied data generators, namely the the Bivariate Normal Distribution (BND) and the Fuzzy Core Clusters (FCC):

1. *Robustness to noise*: Both FADDIS versions are very robust in situations with noise. Such can be verified with the FCC DG, where for a interval of $\alpha = [0, 0.1]$ both algorithms returned good clustering results. The same conclusions can be made analysing the results for the BND DG with $scale = \{-5, 0\}$ (situations with more noise).

2. *Hability to recover the "natural number of clusters"*: The FADDIS-a algorithm shows a more stable performance than FADDIS-m in most of the experiments. Specifically in the BND DG, the former recovers less clusters as the noise is reduced and the cluster structure becomes more well defined. On the other hand, FADDIS-a almost always recovers the correct number of clusters in most of the experiments.

   In particular, for the Iris Dataset, FADDIS-m recovered 2 clusters and not 3. This is an interesting result for the clustering community, since it is claimed [28] that this dataset is considered to have 2 clusters that correspond exactly to the recovered by the algorithm. Also, the very low misclassification error is meaningful in such a reference dataset in cluster analysis.

3. *Quality of the clustering partitions*: This item is pretty much encouraging since the high

values obtained with the ARI index (i.e very close to 1) confirm the conclusions previously made. The FADDIS-a resulting partitions almost always get the highest ARI for both data generators. The FADDIS-m algorithm also obtains very high values for the ARI index, even though not always recovering the original number of clusters.

4. *LAPIN versus no-LAPIN*: The pre-processing of the data with the Lapin transformation in all experiments helps the FADDIS algorithms in retrieving better clustering structures. A careful analysis of the results, may confirm what was pointed out by Mirkin and Nascimento [28], that the Lapin transformation is a good pre-processing method for datasets where the cluster structure is not well defined, but on the contrary, destroys well defined cluster structures. In particular, this can be verified in the experiments for the BND DG, where for $scale = 50$ the ARI index for both algorithms is lower than the case where the Lapin transformation is not used. Preliminary experiments for higher values of the parameter *scale* had been performed, and the results confirm the previous statement.

5. *CEMR and REI error measures*: In case of the FCC DG, these two error measures were used to analyse FADDIS ability to recover the cluster structure of FCC generated data, namely the cluster contributions to the data scatter and intensities. It could be concluded that both FADDIS-a and FADDIS-m obtained good results for $\alpha \in [0, 0.1]$, which indicates that both algorithms are robust to noise and also succeed in recovering the original cluster structure for the mentioned error range.

Concerning the FastMap FCM and the NERFCM, both algorithms shown their ability to well recover the cluster structure, in case of data generated according to the BND DG. On the contrary, the worse results achieved for data generated with higher levels of noise show that these algorithms are not robust to noise.

Apart from the fact that these algorithms cannot determine the number of clusters, the EXB

index proved to be an effective index to be used by this type of Relational Fuzzy Clustering algorithms to determine the number of clusters.

The study with Benchmark datasets were not very conclusive, since all the algorithms returned good or bad results depending on the analysed dataset. As pointed before, an interesting result was obtained with FADDIS-m for the Iris dataset, using the pre-processing Gaussian Kernel with the Lapin transformation.

FADDIS is a recent algorithm and still under study, and so future work can still be conducted. FastMap FCM is also a recent algorithm and almost no experiments were made before. Suggested future work to continue what was accomplished in this thesis include, but are not limited to:

1. Study in more detail the CEMR error measure for each cluster individually, in order to make some conclusions about individual errors on extracted clusters;

2. Make more experiments with the FCC DG using other values of K.

3. Study some FADDIS "model-based" approach for the fine tuning of the threshold parameters of the algorithm stop condition.

4. Run experiments also for $K-1$ and $K+1$ with the parallel algorithms for the FCC DG datasets and apply the Extended Xie Beni index;

5. Apply Extended Xie-Beni on the parallel clustering algorithms results for the Benchmark datasets.

6. Use different methods to convert between similarity and dissimilarity data and compare results.

7. Implement more algorithms, such as ARCA or RFSC to use on the generated datasets, and make a comparative study with other relational clustering algorithms;

A . Results of Experiments with Bivariate Normal Data Generator with Noise

A.1 Original Data Examples

The Table A.1, presents an example dataset generated with all *scale* values and corresponding plots. Also, for the conventional similarity and Gaussian Kernel with and without the Lapin transformation pre-processing, the VAT tool [7] will be applied and the figures shown.

It is interesting to analyse the results of VAT, as a cluster structure can be identified more clearly as the *scale* parameter increases for all measures. The VAT for the Gaussian Kernel without Lapin, appears to identify the cluster structure more clearly.

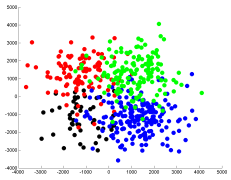
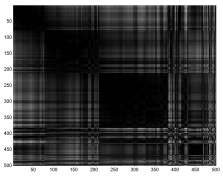
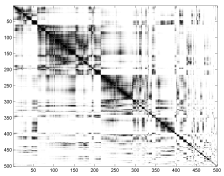
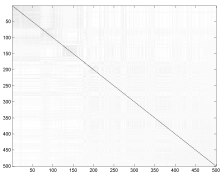
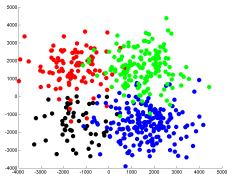
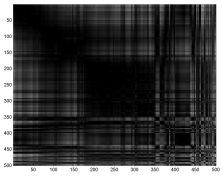
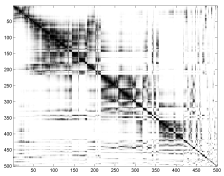
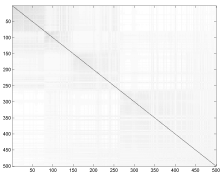
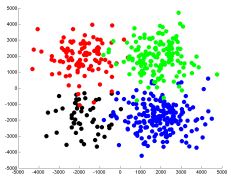
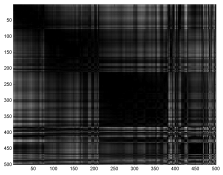
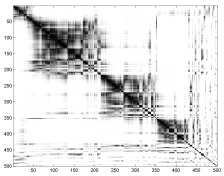
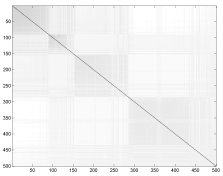
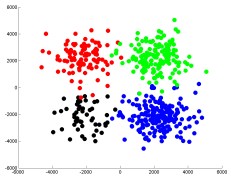
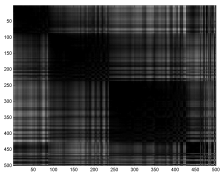
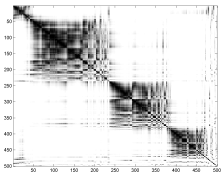
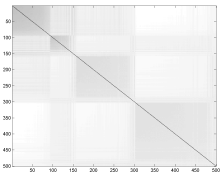
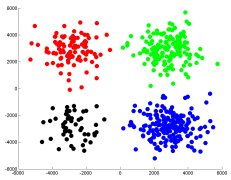
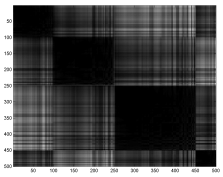
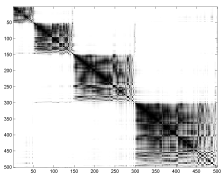
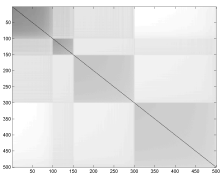
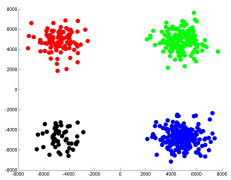
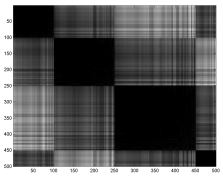
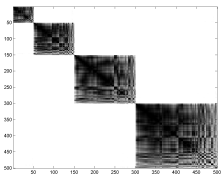
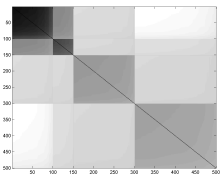
VAT tool results for each Measure				
scale	Original Data plot	Conventional Proximity	Gaussian Kernel	Gaussian Kernel with Lapin
-5	 (a) plot for scale=-5	 (b) CP VAT for scale=-5	 (c) GK VAT for scale=-5	 (d) GK/Lap VAT for scale=-5
0	 (e) plot for scale=0	 (f) CP VAT for scale=0	 (g) GK VAT for scale=0	 (h) GK/Lap VAT for scale=0
5	 (i) plot for scale=5	 (j) CP VAT for scale=5	 (k) GK VAT for scale=5	 (l) GK/Lap VAT for scale=5
10	 (m) plot for scale=10	 (n) CP VAT for scale=10	 (o) GK VAT for scale=10	 (p) GK/Lap VAT for scale=10
20	 (q) plot for scale=20	 (r) CP VAT for scale=20	 (s) GK VAT for scale=20	 (t) GK/Lap VAT for scale=20
50	 (u) plot for scale=50	 (v) CP VAT for scale=50	 (w) GK VAT for scale=50	 (x) GK/Lap VAT for scale=50

Figure A.1 VAT tool results for each dataset for Conventional Proximity

A.2 FADDIS-a

An example dataset was selected to show clustering results plots for FADDIS-a for each *scale* value and Gaussian Kernel with and without the Lapin transformation in Figures A.2 and A.3.

In addition, for each result, the number of extracted clusters K, the contributions and intensities of each extracted cluster, the ARI index and algorithm stop condition will be presented. As it was concluded in Section 5.3.4, the Gaussian Kernel with the Lapin transformation returned higher ARI index values in all *scales*.

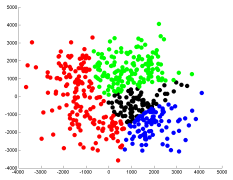
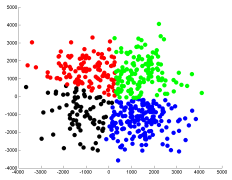
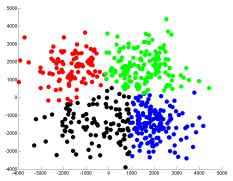
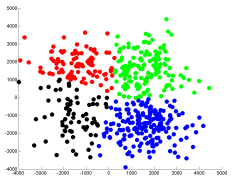
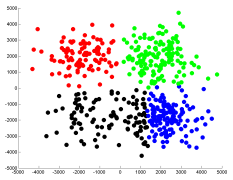
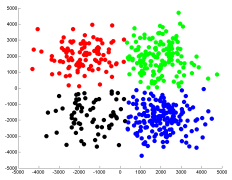
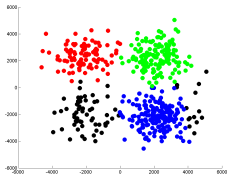
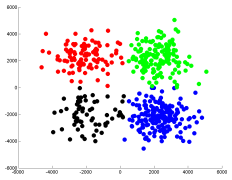
FADDIS-a results for one example Dataset		
scale	Gaussian Kernel	Gaussian Kernel with Lapin
-5	 (a) GK plot for scale=-5	 (b) GK/L plot for scale=-5
	K=4 Contrib = [0.39 0.12 0.04 0.022] Intensity = [81.8 46.2 26.3 19.4] ARI = 0.3 Stop = Small contribution	K=4 Contrib = [0.037 0.02 0.01 0.008] Intensity = [4.7 3.5 2.5 2.2] **ARI = 0.56** Stop = Small contribution
0	 (c) GK plot for scale=0	 (d) GK/L plot for scale=0
	K=4 Contrib = [0.39 0.17 0.08 0.03] Intensity = [83.3 54.4 37.7 21.9] ARI = 0.61 Stop = Small contribution	K=4 Contrib = [0.07 0.03 0.02 0.007] Intensity = [6.8 4.6 3.6 2.1] **ARI = 0.73** Stop = Small contribution
5	 (e) GK plot for scale=5	 (f) GK/L plot for scale=5
	K=4 Contrib = [0.41 0.2 0.1 0.03] Intensity = [89.1 62.3 44.3 22.6] ARI = 0.67 Stop = Small contribution	K=4 Contrib = [0.13 0.05 0.033 0.013] Intensity = [10.4 6.5 5.3 3.3] **ARI = 0.85** Stop = Small contribution
10	 (g) GK plot for scale=10	 (h) GK/L plot for scale=10
	K=4 Contrib = [0.44 0.23 0.11 0.02] Intensity = [96.6 69.3 48.7 22.9] ARI = 0.87 Stop = Small contribution	K=4 Contrib = [0.21 0.074 0.05 0.023] Intensity = [16.3 9.8 8.2 5.4] **ARI = 0.89** Stop = Small contribution

Figure A.2 FADDIS-a results for one example dataset for each scale value

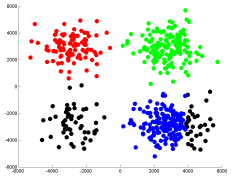
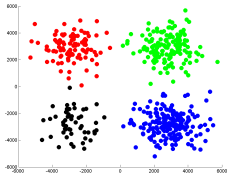
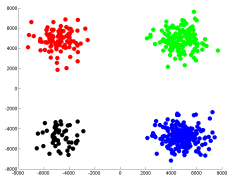
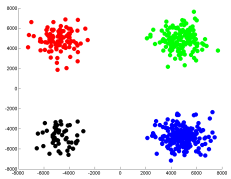
FADDIS-a results for one example Dataset		
scale	Gaussian Kernel	Gaussian Kernel with Lapin
20	 (a) GK plot for scale=20	 (b) GK/L plot for scale=20
	K=4 Contrib = [0.47 0.25 0.12 0.022] Intensity = [111.7 81.8 56.4 24.3] ARI = 0.86 Stop = Small contribution	K=4 Contrib = [0.35 0.1 0.08 0.04] Intensity = [44.6 24.2 21.4 14.4] **ARI = 0.99** Stop = Small contribution
50	 (c) GK plot for scale=50	 (d) GK/L plot for scale=50
	K=4 Contrib = [0.51 0.28 0.13 0.03] Intensity = [144.5 107.7 72.6 36.5] **ARI = 1** Stop = Small contribution	K=4 Contrib = [0.56 0.07 0.065 0.03] Intensity = [1068 388 364 236] **ARI = 1** Stop = Small contribution

Figure A.3 FADDIS-a results for one example dataset for each scale value

A.3 FADDIS-m

The same example dataset that was used with FADDIS-a previously will also be clustered using FADDIS-m. The clustering results plots for FADDIS-m for each *scale* value and Gaussian Kernel with and without the Lapin transformation will be shown in Figures A.4 and A.5.

As it was concluded in Section 5.3.4, the Gaussian Kernel with the Lapin transformation returned higher ARI index values for most of the *scale* values. For $scale \geq 50$ the Lapin transformation destroys the clustering structure and this algorithm is unable to recover it. It should also be noted that as the *scale* value increases, the FADDIS-m algorithm recovers less clusters.

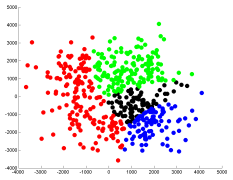
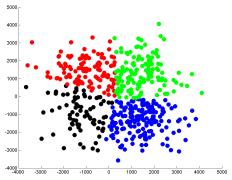
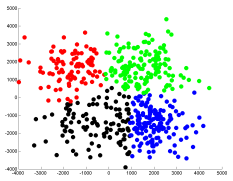
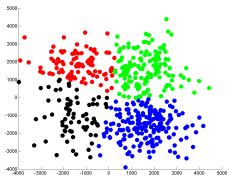
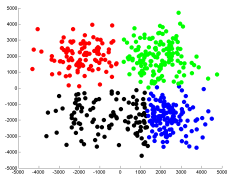
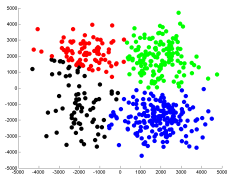
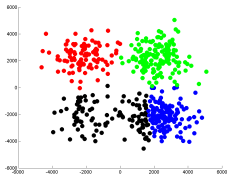
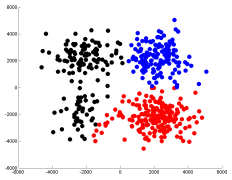
FADDIS-m results for one example Dataset		
scale	Gaussian Kernel	Gaussian Kernel with Lapin
-5	 (a) GK plot for scale=-5	 (b) GK/L plot for scale=-5
	K=4 Contrib = [0.39 0.12 0.04 0.02] Intensity = [81.8 46.2 26.3 19.4] ARI = 0.3 Stop = Small contribution	K=4 Contrib = [0.037 0.021 0.011 0.008] Intensity = [4.7 3.5 2.5 2.2] **ARI = 0.56** Stop = No positive weights
0	 (c) GK plot for scale=0	 (d) GK/L plot for scale=0
	K=4 Contrib = [0.39 0.17 0.08 0.03] Intensity = [83.3 54.4 37.7 22] ARI = 0.61 Stop = Small contribution	K=4 Contrib = [0.07 0.032 0.02 0.007] Intensity = [6.8 4.6 3.6 2.1] **ARI = 0.73** Stop = No positive weights
5	 (e) GK plot for scale=5	 (f) GK/L plot for scale=5
	K=4 Contrib = [0.4 0.2 0.1 0.027] Intensity = [89.1 62.3 44.3 22.6] ARI = 0.67 Stop = Small contribution	K=4 Contrib = [0.13 0.051 0.033 0.005] Intensity = [10.4 6.5 5.3 2.1] **ARI = 0.82** Stop = No positive weights
10	 (g) GK plot for scale=10	 (h) GK/L plot for scale=10
	K=4 Contrib = [0.44 0.23 0.11 0.02] Intensity = [96.6 69.3 48.7 20.8] ARI = 0.72 Stop = Small contribution	K=3 Contrib = [0.21 0.074 0.052] Intensity = [16.35 9.75 8.17] **ARI = 0.82** Stop = Small contribution

Figure A.4 FADDIS-m results for one example dataset for each scale value

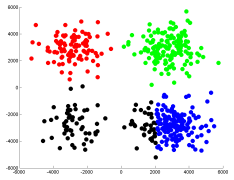
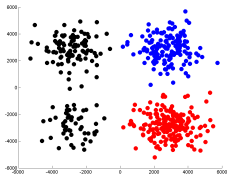
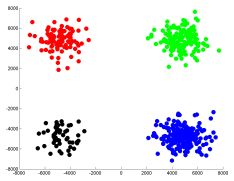
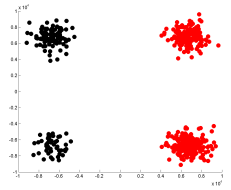
<table>
<tr><th colspan="3">FADDIS-m results for one example Dataset</th></tr>
<tr><th>scale</th><th>Gaussian Kernel</th><th>Gaussian Kernel with Lapin</th></tr>
<tr><td rowspan="2">20</td><td> 
(a) GK plot for scale=20</td><td> 
(b) GK/L plot for scale=20</td></tr>
<tr><td>K=4
Contrib = [0.47 0.25 0.12 0.022]
Intensity = [111.7 81.8 56.4 23.9]
ARI = 0.83
Stop = Small contribution</td><td>K=3
Contrib = [0.35 0.1 0.08]
Intensity = [44.6 24.2 21.4]
ARI = 0.91
Stop = Small contribution</td></tr>
<tr><td rowspan="2">50</td><td> 
(c) GK plot for scale=50</td><td> 
(d) GK/L plot for scale=50</td></tr>
<tr><td>K=4
Contrib = [0.51 0.28 0.13 0.032]
Intensity = [144.6 107.7 72.6 36.5]
ARI = 1
Stop = Small contribution</td><td>K=2
Contrib = [0.56 0.073]
Intensity = [1067.9 387.6]
ARI = 0.89
Stop = No positive weights</td></tr>
</table>

Figure A.5 FADDIS-m results for one example dataset for each scale value

A.4 FastMap FCM

To ilustrate the results for FMFCM, the same example dataset was used with all selected values of the *scale* parameter. In addition, for each *scale* value, the algorithm was run with input parameter K for the number of clusters of values $\{3,4,5\}$. These results are shown in Figures A.6 and A.7.

From analysing the results, it can be concluded that FMFCM obtains the best results for lower *scale* values (i.e $scale = \{-5,0\}$) with $K = 3$, therefore it is not very robust to noise. In the other hand, as the noise is reduced it obtains the best values with $K = 4$. This is concordant with the conclusions made previously in Section 5.3.4.

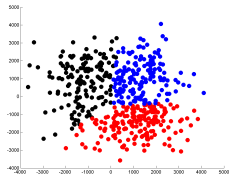
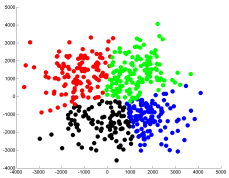
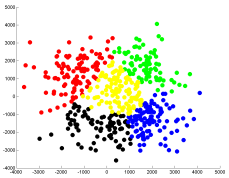
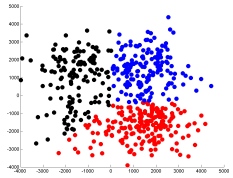
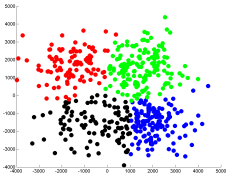
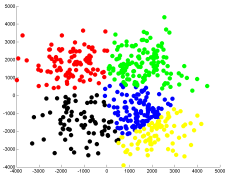
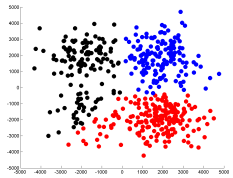
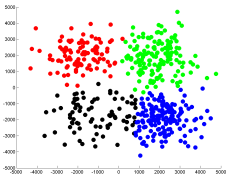
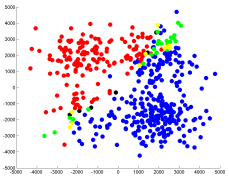
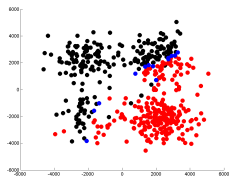
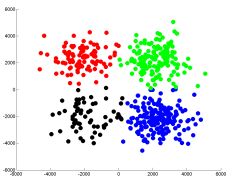
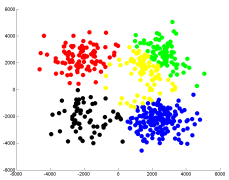
FastMap FCM results for one example Dataset			
scale	K = 3	K = 4	K = 5
-5	 (a) Plot for K=3 / scale=-5	 (b) Plot for K=4 / scale=-5	 (c) Plot for K=5 / scale=-5
	ARI = 0.46928 EXB = 0.22262	ARI = 0.43375 EXB = 0.20848	ARI = 0.3426 **EXB = 0.20381**
0	 (d) Plot for K=3 / scale=-5	 (e) Plot for K=4 / scale=-5	 (f) Plot for K=5 / scale=-5
	ARI = 0.6864 **EXB = 0.17316**	ARI = 0.61326 EXB = 0.21092	ARI = 0.52923 EXB = 0.53369
5	 (g) Plot for K=3 / scale=-5	 (h) Plot for K=4 / scale=-5	 (i) Plot for K=5 / scale=-5
	ARI = 0.75519 EXB = 0.1502	**ARI = 0.77893** **EXB = 0.13233**	ARI = 0.66476 EXB = 0.67081
10	 (j) Plot for K=3 / scale=-5	 (k) Plot for K=4 / scale=-5	 (l) Plot for K=5 / scale=-5
	ARI = 0.79885 EXB = 0.12393	**ARI = 0.86238** **EXB = 0.079383**	ARI = 0.72765 EXB = 0.60965

Figure A.6 FastMap FCM results for one example dataset for each scale value and K={3, 4, 5}

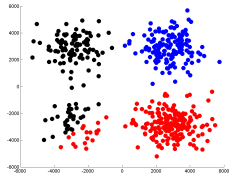
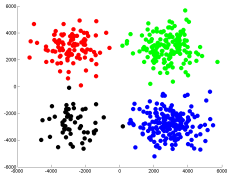
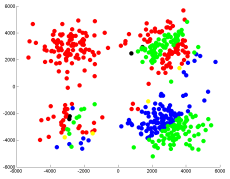
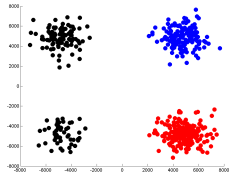
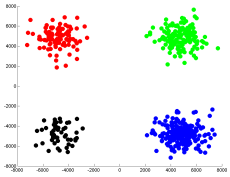
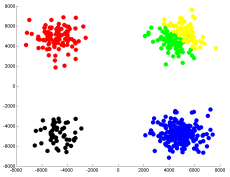
<table>
<tr><th colspan="4">FastMap FCM results for one example Dataset</th></tr>
<tr><td>scale</td><td>K = 3</td><td>K = 4</td><td>K = 5</td></tr>
<tr><td rowspan="2">20</td><td> 
(a) Plot for K=3 / scale=-5</td><td> 
(b) Plot for K=4 / scale=-5</td><td> 
(c) Plot for K=5 / scale=-5</td></tr>
<tr><td>ARI = 0.86531
EXB = 0.093261</td><td>ARI = 0.99433
EXB = 0.044892</td><td>ARI = 0.85704
EXB = 0.44454</td></tr>
<tr><td rowspan="2">50</td><td> 
(d) Plot for K=3 / scale=-5</td><td> 
(e) Plot for K=4 / scale=-5</td><td> 
(f) Plot for K=5 / scale=-5</td></tr>
<tr><td>ARI = 0.86938
EXB = 0.057896</td><td>ARI = 1
EXB = 0.019073</td><td>ARI = 0.988
EXB = 0.561</td></tr>
</table>

Figure A.7 FastMap FCM results for one example dataset for each scale value and K={3, 4, 5}

A.5 NERFCM

As the FMFCM the results for NERFCM will be of the same example dataset. The same setting was also used and the results can be seen in Figures **??** and **??**.

From analysing the results, it can be concluded that NERFCM obtains the best results for the original number of clusters $K = 4$ for $scale = \{5, 10, 20, 50\}$. It can be concluded that NERFCM is also not very robust to noise.

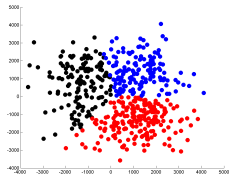
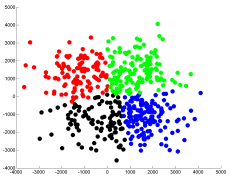
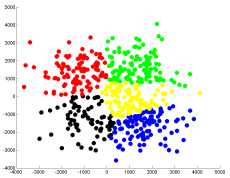
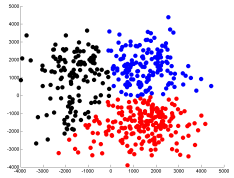
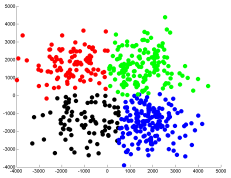
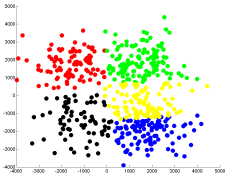
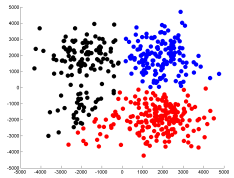
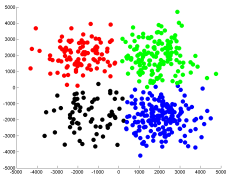
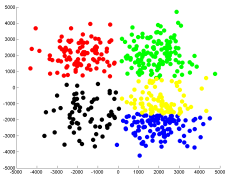
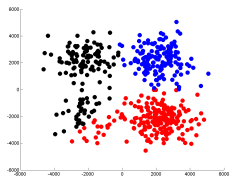
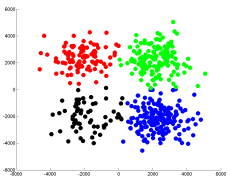
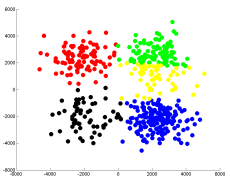
NERFCM results for one example Dataset			
scale	K = 3	K = 4	K = 5
-5	 (a) Plot for K=3 / scale=-5	 (b) Plot for K=4 / scale=-5	 (c) Plot for K=5 / scale=-5
	ARI = 0.54319 EXB = 0.18201	ARI = 0.47682 **EXB = 0.17209**	**ARI = 0.43784** EXB = 0.27015
0	 (d) Plot for K=3 / scale=-5	 (e) Plot for K=4 / scale=-5	 (f) Plot for K=5 / scale=-5
	ARI = 0.70535 **EXB = 0.14358**	ARI = 0.71886 EXB = 0.14439	ARI = 0.56143 EXB = 0.36568
5	 (g) Plot for K=3 / scale=-5	 (h) Plot for K=4 / scale=-5	 (i) Plot for K=5 / scale=-5
	ARI = 0.77774 EXB = 0.11868	**ARI = 0.8324** **EXB = 0.092466**	ARI = 0.66526 EXB = 0.6134
10	 (j) Plot for K=3 / scale=-5	 (k) Plot for K=4 / scale=-5	 (l) Plot for K=5 / scale=-5
	ARI = 0.8113 EXB = 0.10588	**ARI = 0.90144** **EXB = 0.066848**	ARI = 0.78636 EXB = 0.28763

Figure A.8 NERFCM results for one example dataset for each scale value and K={3, 4, 5}

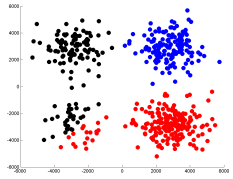
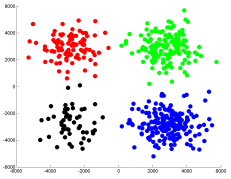
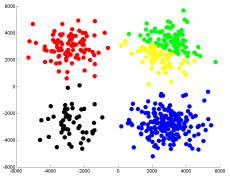
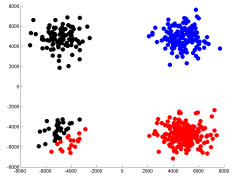
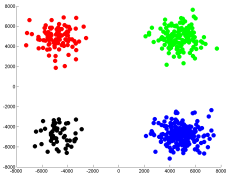
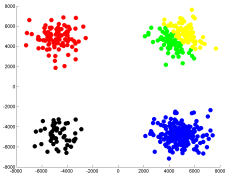
<table>
<tr><th colspan="4">NERFCM results for one example Dataset</th></tr>
<tr><td>scale</td><td>K = 3</td><td>K = 4</td><td>K = 5</td></tr>
<tr><td rowspan="2">20</td><td> 
(a) Plot for K=3 / scale=-5</td><td> 
(b) Plot for K=4 / scale=-5</td><td> 
(c) Plot for K=5 / scale=-5</td></tr>
<tr><td>ARI = 0.86531
EXB = 0.078367</td><td>ARI = 0.99433
EXB = 0.044892</td><td>ARI = 0.88356
EXB = 0.55234</td></tr>
<tr><td rowspan="2">50</td><td> 
(d) Plot for K=3 / scale=-5</td><td> 
(e) Plot for K=4 / scale=-5</td><td> 
(f) Plot for K=5 / scale=-5</td></tr>
<tr><td>ARI = 0.86531
EXB = 0.054167</td><td>ARI = 1
EXB = 0.018516</td><td>ARI = 0.9514
EXB = 0.93312</td></tr>
</table>

Figure A.9 NERFCM results for one example dataset for each scale value and K={3, 4, 5}

B . Results of Experiments with Fuzzy Core Cluster Generator Tests

B.1 K=3 Experiments Summary

In this section the Tables for the percentage of correct number of extracted clusters, CEMR error, REI error, Crisp Core Matching and ARI will be shown for the experiments made using the FCC DG with $K = 3$.

	Algorithm	
Dataset Size	FADDIS-a	FADDIS-m
50	90/14.1	75/5.8
100	80/11.5	80/11.5
200	70/14.1	82.5/5
400	95/10	82.5/5
700	67.5/17.1	70/8.2

Table B.1 Percentage of correct extracted clusters avg/std for std of added Gaussian noise=[0, 0.1] for K=3

	Algorithm	
Dataset Size	FADDIS-a	FADDIS-m
50	0.21/0.057	0.17/0.048
100	0.19/0.066	0.19/0.065
200	0.25/0.11	0.25/0.097
400	0.18/0.063	0.19/0.056
700	0.28/0.14	0.26/0.07

Table B.2 Characteristic of the membership Error avg/std for std of added Gaussian noise=[0, 0.1] for K=3

	Algorithm	
Dataset Size	FADDIS-a	FADDIS-m
50	0.091/0.076	0.097/0.075
100	0.082/0.018	0.081/0.014
200	0.16/0.032	0.166/0.03
400	0.36/0.034	0.38/0.036
700	0.33/0.087	0.34/0.095

Table B.3 Intensity Error avg/std for std of added Gaussian noise=[0, 0.1] for K=3

	Algorithm			
Dataset Size	FADDIS-a	FADDIS-m	FastMap FCM	NERFCM
50	91.9/2.7	91/1.1	87.9/5.4	81.8/1.3
100	85.8/4.5	84.7/5.9	84.8/7	78.5/2.7
200	78.2/5.3	82.8/2.3	83.7/7.6	68.1/0.8
400	98.5/1.3	95.8/2.8	86/15.4	56.4/2.9
700	79.4/4.1	84.9/1.7	75.1/17.4	50.2/3.9

Table B.4 Crisp Core Matching (%) avg/std for std of added Gaussian noise=[0, 0.1] for K=3

	Algorithm			
Dataset Size	FADDIS-a	FADDIS-m	FastMap FCM	NERFCM
50	0.89/0.25	0.9/0.19	0.77/0.26	0.67/0.26
100	0.91/0.12	0.92/0.13	0.73/0.21	0.62/0.25
200	0.83/0.18	0.82/0.19	0.69/0.28	0.43/0.29
400	0.97/0.047	0.95/0.07	0.71/0.3	0.21/0.15
700	0.79/0.23	0.8/0.22	0.52/0.33	0.14/0.12

Table B.5 Summary Table for ARI avg/std for std of added Gaussian noise=[0, 0.1] for all algorithms in best conditions for K=3

B.2 K=7 Experiments Summary

In this section the tables for the percentage of correct number of extracted clusters, CEMR error, REI error, Crisp Core Matching and ARI will be shown for the experiments made using the FCC DG with $K = 7$.

	Algorithm	
Dataset Size	FADDIS-a	FADDIS-m
50	62.5/41.1	20/18.3
100	57.5/40.3	22.5/12.6
200	67.5/33	27.5/20.6
400	60/27.1	37.5/5
700	57.5/33	12.5/5

Table B.6 Percentage of correct extracted clusters avg/std for std of added Gaussian noise=[0, 0.1] for K=7

	Algorithm	
Dataset Size	FADDIS-a	FADDIS-m
50	0.16/0.09	0.19/0.1
100	0.14/0.08	0.14/0.07
200	0.14/0.08	0.13/0.09
400	0.14/0.09	0.14/0.09
700	0.14/0.07	0.13/0.07

Table B.7 Characteristic of the membership Error avg/std for std of added Gaussian noise=[0, 0.1] for K=7

	Algorithm	
Dataset Size	FADDIS-a	FADDIS-m
50	0.068/0.034	0.1/0.1
100	0.062/0.031	0.058/0.03
200	0.053/0.038	0.065/0.07
400	0.051/0.036	0.066/0.05
700	0.1/0.072	0.092/0.07

Table B.8 Intensity Error avg/std for std of added Gaussian noise=[0, 0.1] for K=7

	Algorithm			
Dataset Size	FADDIS-a	FADDIS-m	FastMap FCM	NERFCM
50	95.2/7.9	88.6/13	68.2/7.2	70/8.4
100	93.8/11.2	89.7/12.3	64.4/8	73.8/9
200	97.8/3	92.1/10.1	63.3/6	73.8/9.6
400	97.5/3	91.3/3.3	56.8/4.7	67.8/3.6
700	97.5/2.3	94.4/3.1	51/8.5	65.1/3.8

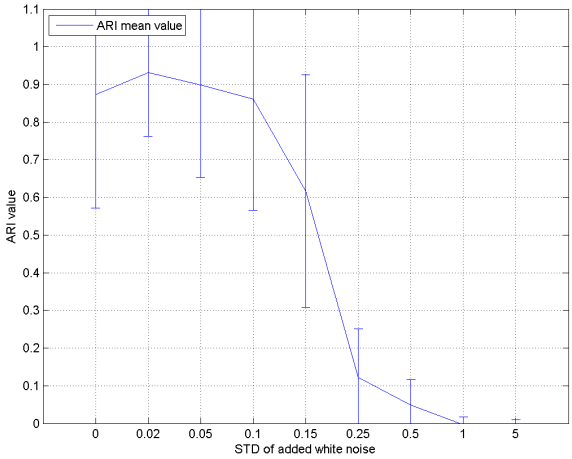
Table B.9 Crisp Core Matching (%) avg/std for std of added Gaussian noise=[0, 0.1] for K=7

	Algorithm			
Dataset Size	FADDIS-a	FADDIS-m	FastMap FCM	NERFCM
50	0.94/0.12	0.86/0.21	0.67/0.18	0.6/0.19
100	0.92/0.16	0.87/0.2	0.65/0.19	0.69/0.19
200	0.97/0.06	0.91/0.16	0.64/0.17	0.67/0.19
400	0.97/0.04	0.91/0.09	0.54/0.17	0.6/0.14
700	0.97/0.04	0.94/0.05	0.48/0.2	0.56/0.15

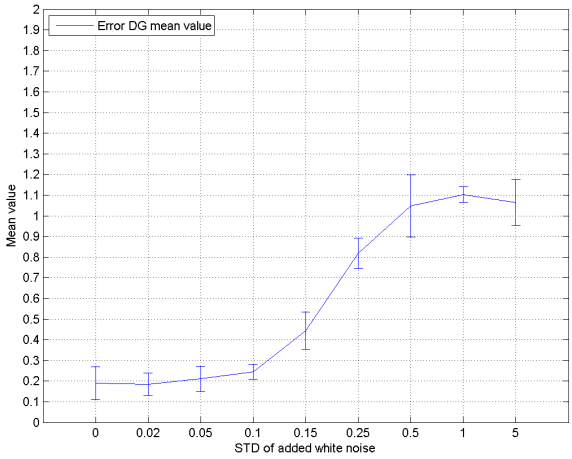
Table B.10 Summary Table for ARI avg/std for std of added Gaussian noise=[0, 0.1] for all algorithms in best conditions for K=7

B.3 FADDIS-a plot results

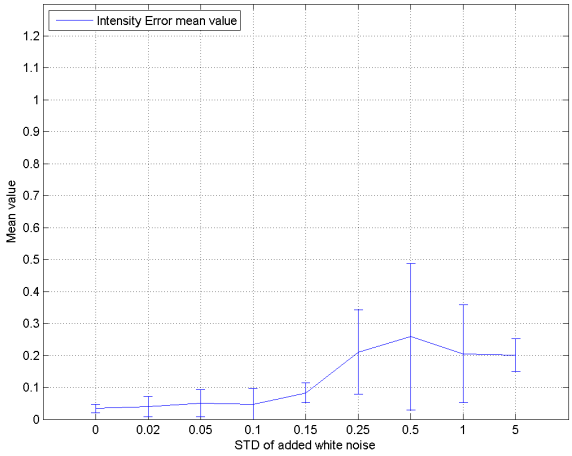
All the plots for the FADDIS-a ARI, CEMR and REI mean/avg in function of α and for $K = \{3,5,7\}$ and $N = \{50,100,200,400,700\}$ will be shown in this section.



(a) FADDIS-a ARI mean/avg for $K = 3$ and $N = 50$

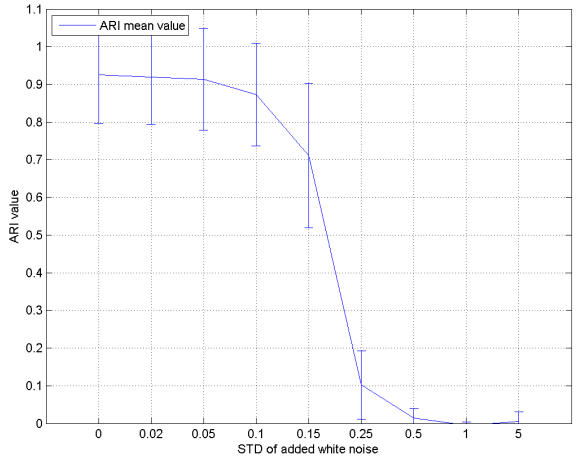


(b) FADDIS-a CEMR mean/avg for $K = 3$ and $N = 50$

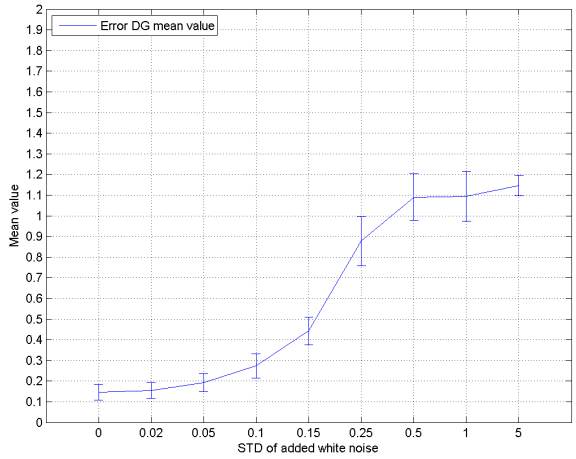


(c) FADDIS-a REI mean/avg for $K = 3$ and $N = 50$

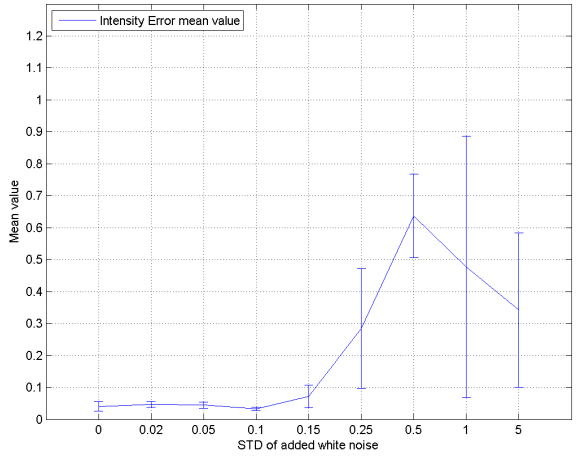
Figure B.1 FADDIS-a results with $K = 3$ and $N = 50$



(a) FADDIS-a ARI mean/avg for $K = 3$ and $N = 100$

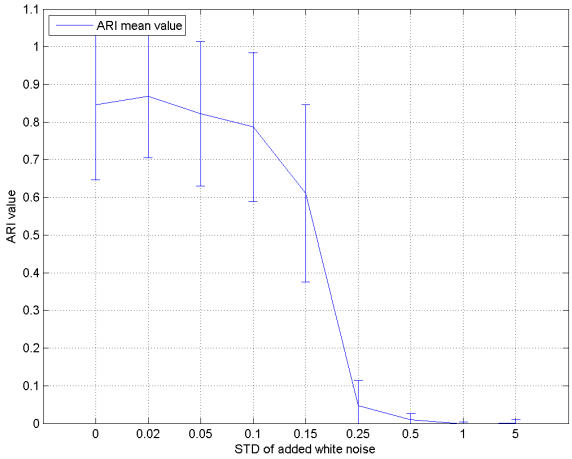


(b) FADDIS-a CEMR mean/avg for $K = 3$ and $N = 100$

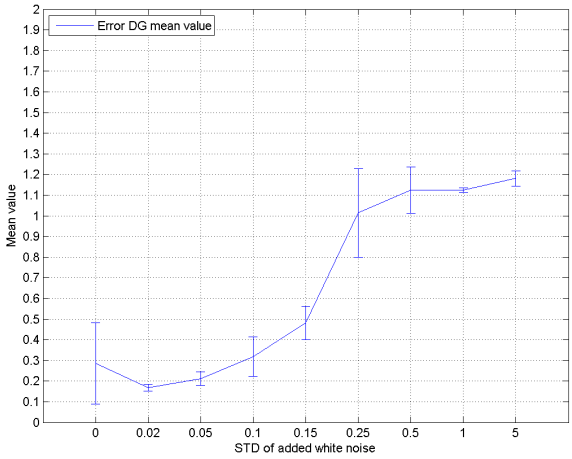


(c) FADDIS-a REI mean/avg for $K = 3$ and $N = 100$

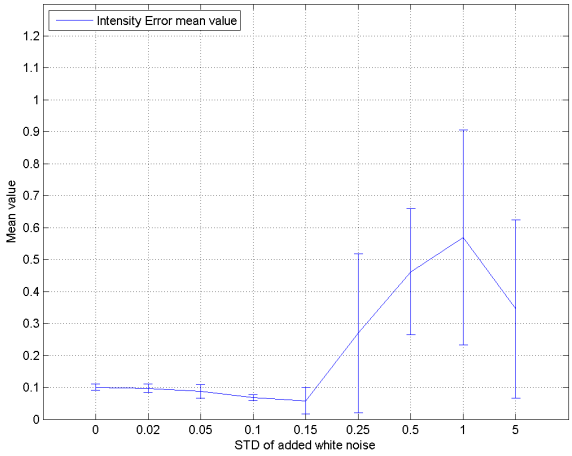
Figure B.2 FADDIS-a results with $K = 3$ and $N = 100$



(a) FADDIS-a ARI mean/avg for $K = 3$ and $N = 200$

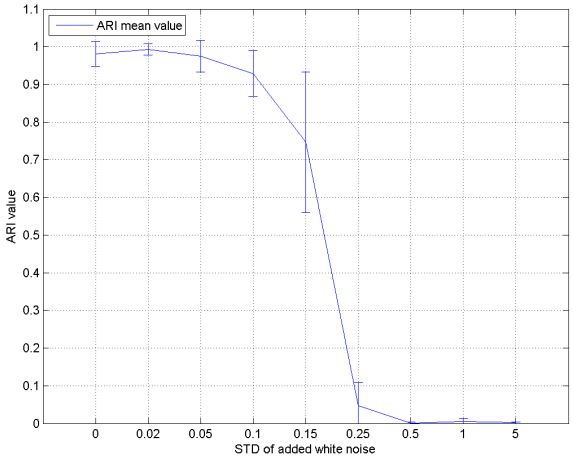


(b) FADDIS-a CEMR mean/avg for $K = 3$ and $N = 200$

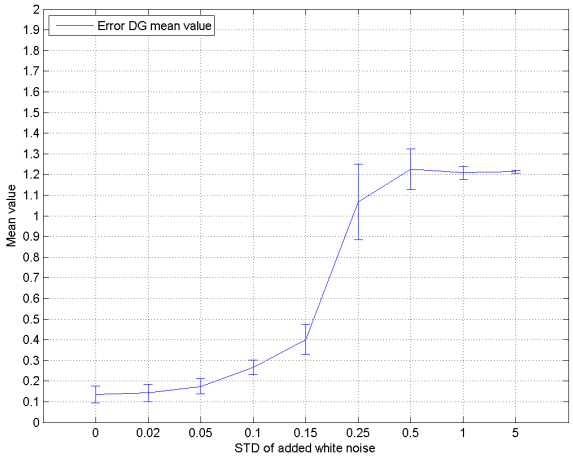


(c) FADDIS-a REI mean/avg for $K = 3$ and $N = 200$

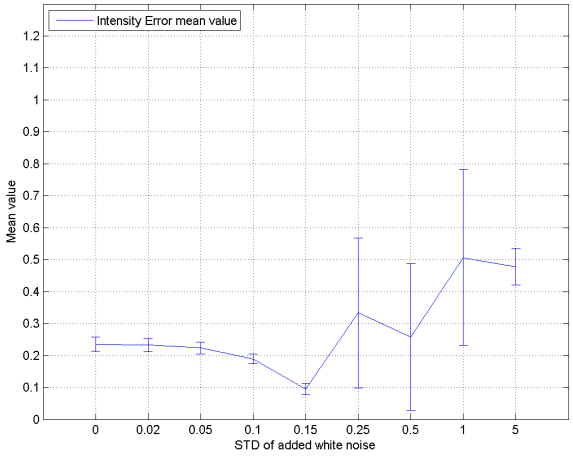
Figure B.3 FADDIS-a results with $K = 3$ and $N = 200$



(a) FADDIS-a ARI mean/avg for $K = 3$ and $N = 400$

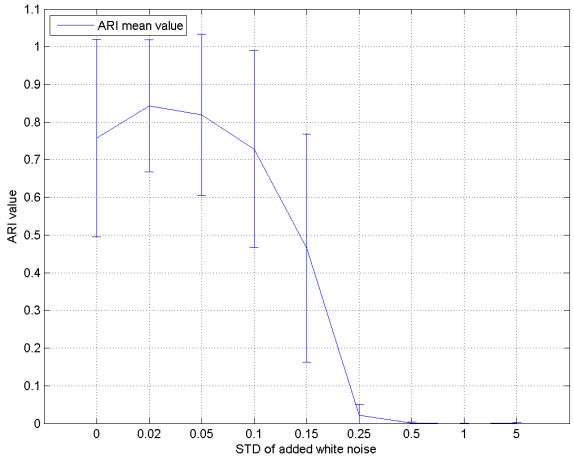


(b) FADDIS-a CEMR mean/avg for $K = 3$ and $N = 400$

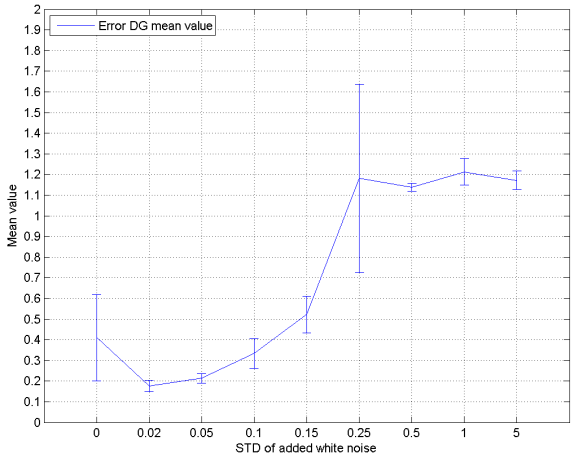


(c) FADDIS-a REI mean/avg for $K = 3$ and $N = 400$

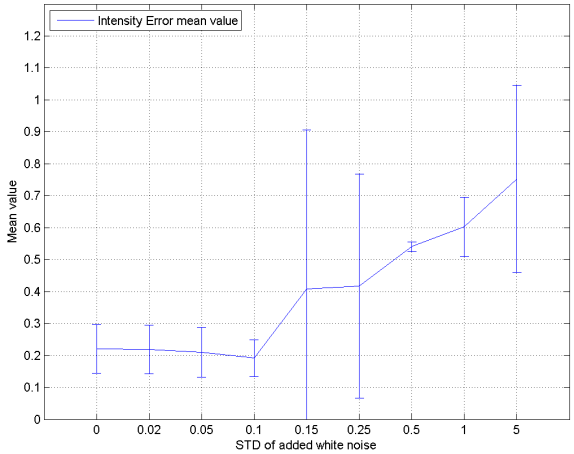
Figure B.4 FADDIS-a results with $K = 3$ and $N = 400$



(a) FADDIS-a ARI mean/avg for $K = 3$ and $N = 700$

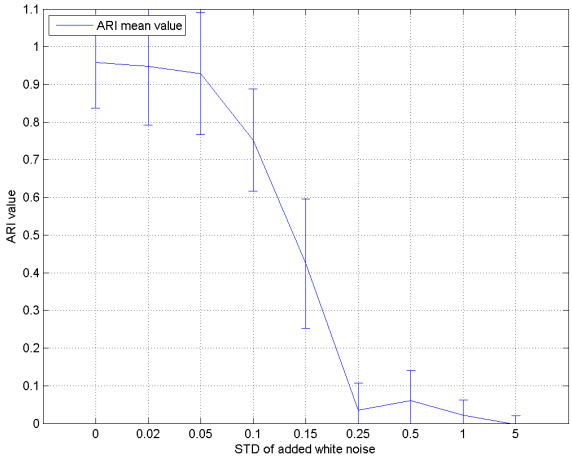


(b) FADDIS-a CEMR mean/avg for $K = 3$ and $N = 700$

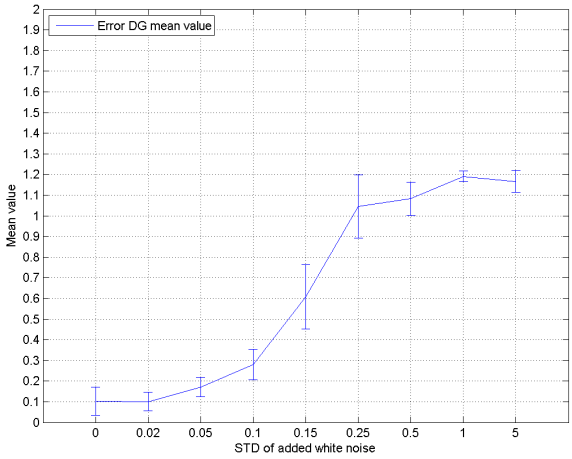


(c) FADDIS-a REI mean/avg for $K = 3$ and $N = 700$

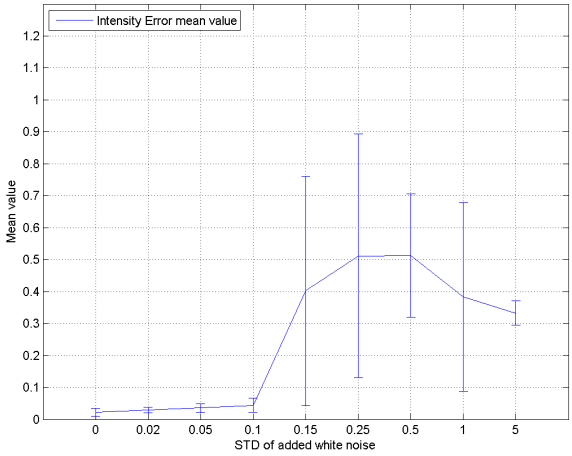
Figure B.5 FADDIS-a results with $K = 3$ and $N = 700$



(a) FADDIS-a ARI mean/avg for $K = 5$ and $N = 50$

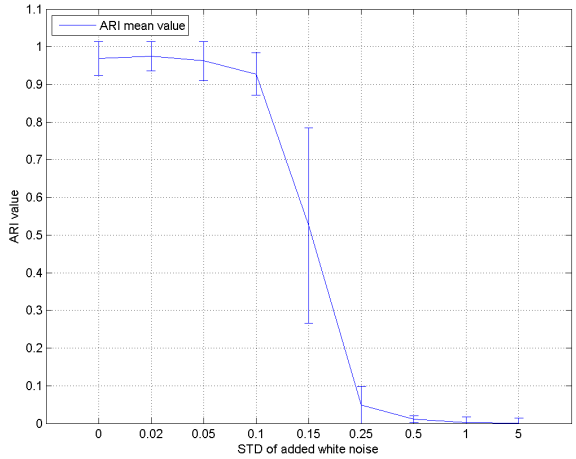


(b) FADDIS-a CEMR mean/avg for $K = 5$ and $N = 50$

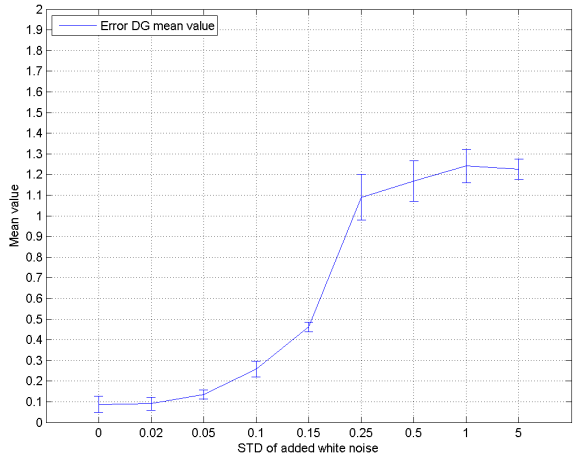


(c) FADDIS-a REI mean/avg for $K = 5$ and $N = 50$

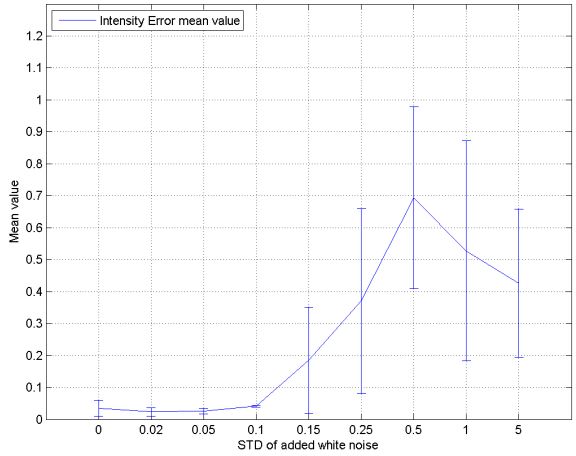
Figure B.6 FADDIS-a results with $K = 5$ and $N = 50$



(a) FADDIS-a ARI mean/avg for $K = 5$ and $N = 100$

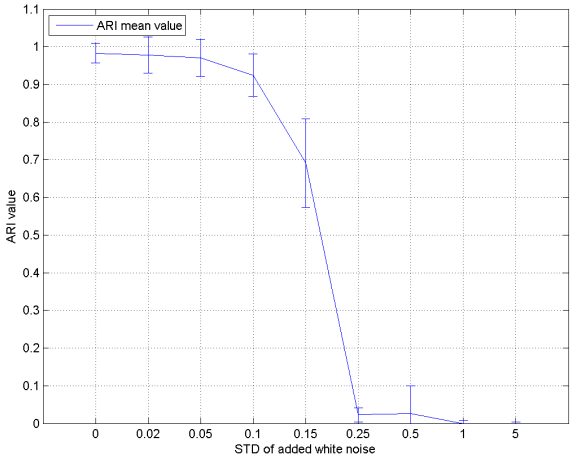


(b) FADDIS-a CEMR mean/avg for $K = 5$ and $N = 100$

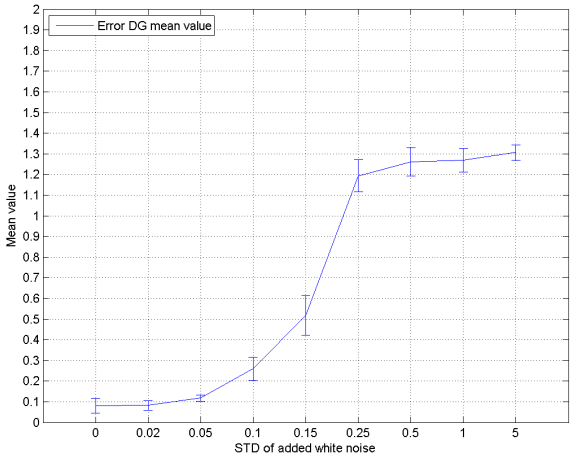


(c) FADDIS-a REI mean/avg for $K = 5$ and $N = 100$

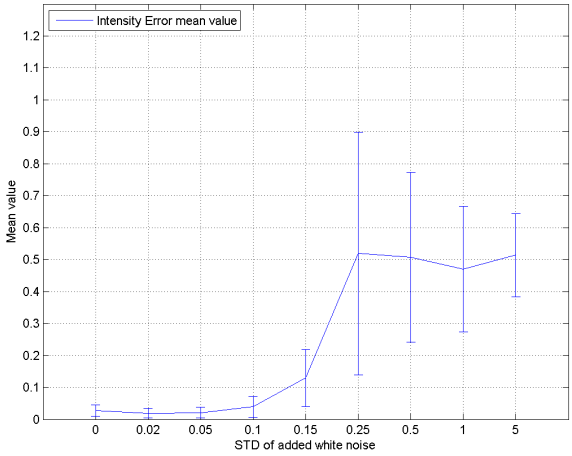
Figure B.7 FADDIS-a results with $K = 5$ and $N = 100$



(a) FADDIS-a ARI mean/avg for $K = 5$ and $N = 200$

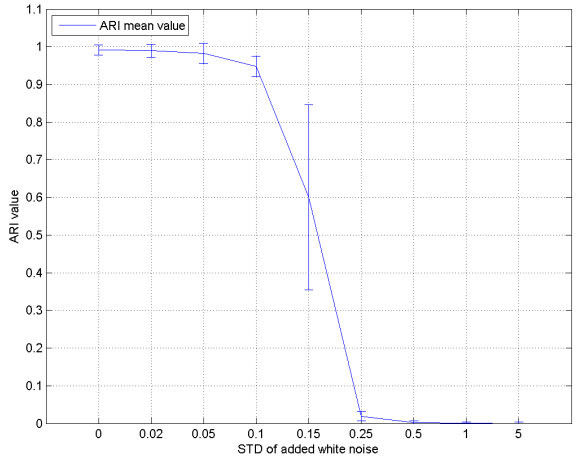


(b) FADDIS-a CEMR mean/avg for $K = 5$ and $N = 200$

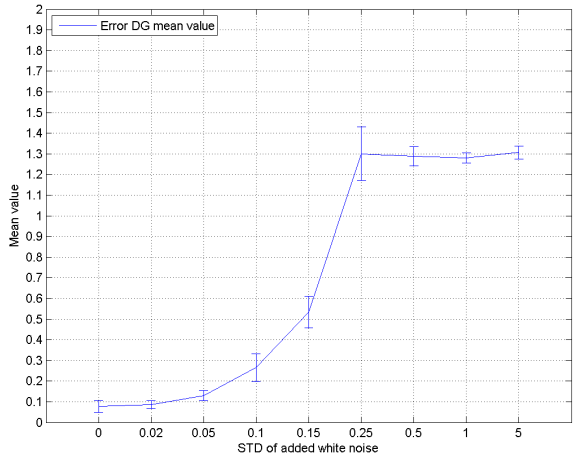


(c) FADDIS-a REI mean/avg for $K = 5$ and $N = 200$

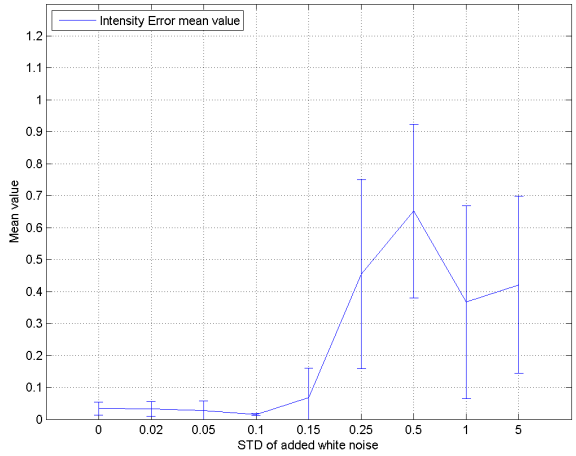
Figure B.8 FADDIS-a results with $K = 5$ and $N = 200$



(a) FADDIS-a ARI mean/avg for $K = 5$ and $N = 400$

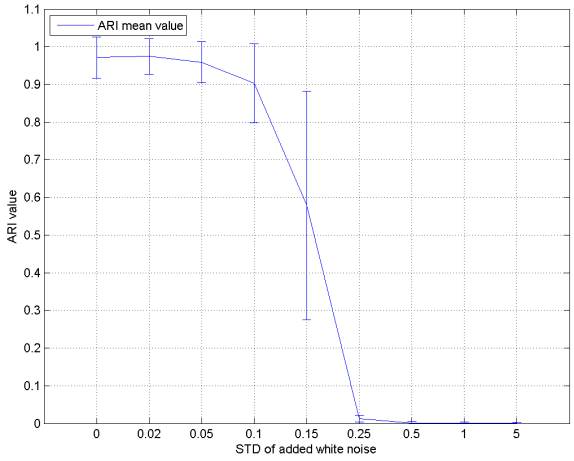


(b) FADDIS-a CEMR mean/avg for $K = 5$ and $N = 400$

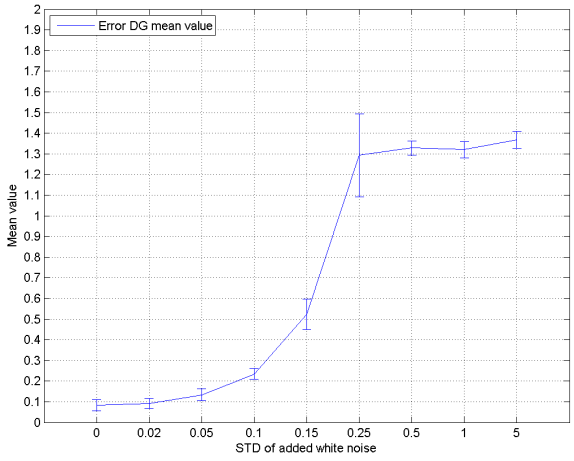


(c) FADDIS-a REI mean/avg for $K = 5$ and $N = 400$

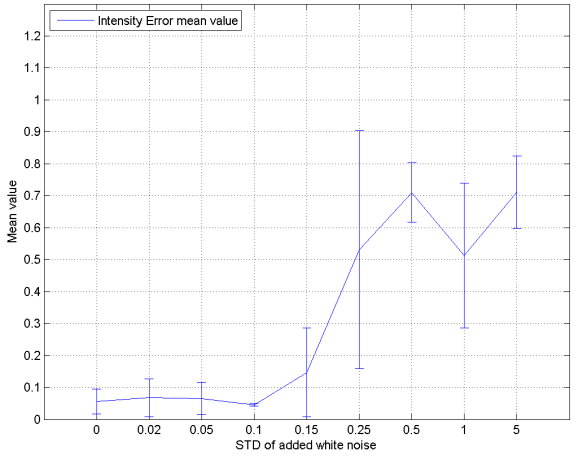
Figure B.9 FADDIS-a results with $K = 5$ and $N = 400$



(a) FADDIS-a ARI mean/avg for $K = 5$ and $N = 700$

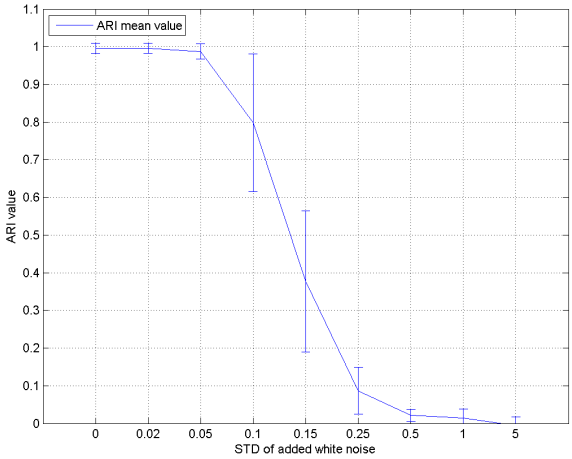


(b) FADDIS-a CEMR mean/avg for $K = 5$ and $N = 700$

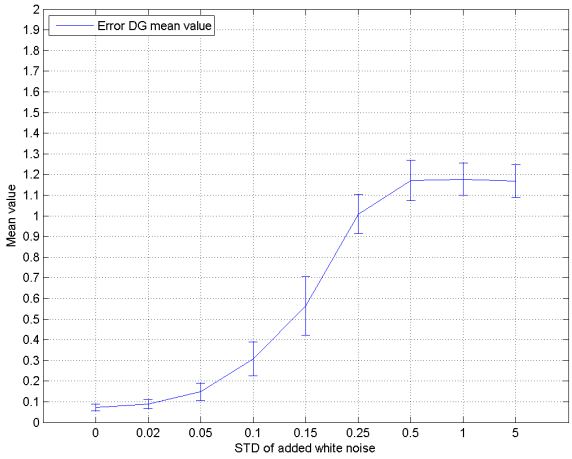


(c) FADDIS-a REI mean/avg for $K = 5$ and $N = 700$

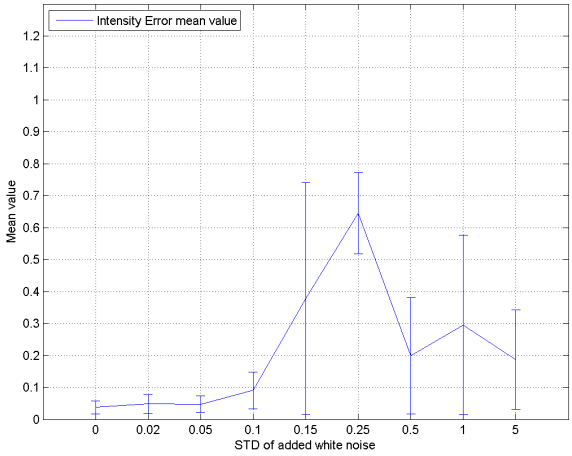
Figure B.10 FADDIS-a results with $K = 5$ and $N = 700$



(a) FADDIS-a ARI mean/avg for $K = 7$ and $N = 50$

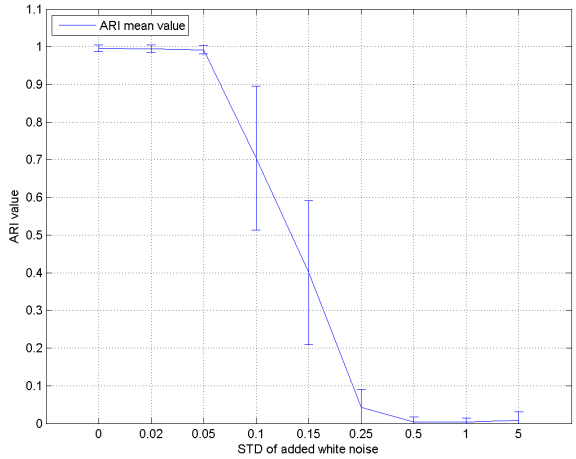


(b) FADDIS-a CEMR mean/avg for $K = 7$ and $N = 50$

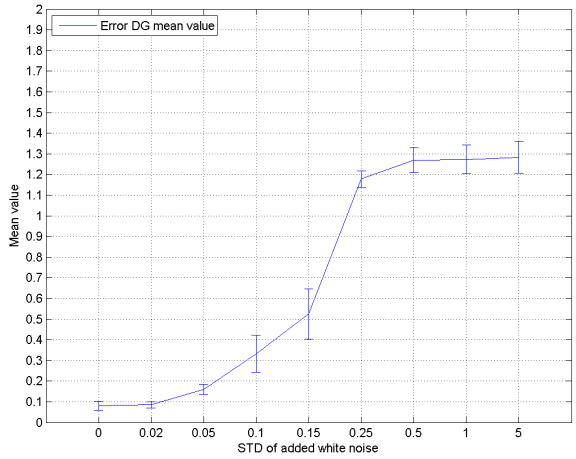


(c) FADDIS-a REI mean/avg for $K = 7$ and $N = 50$

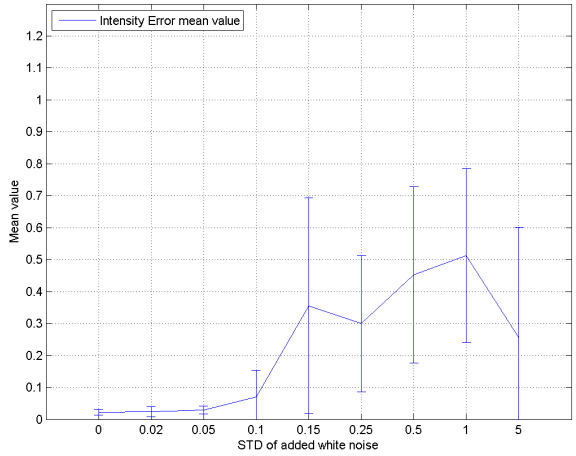
Figure B.11 FADDIS-a results with $K = 7$ and $N = 50$



(a) FADDIS-a ARI mean/avg for $K = 7$ and $N = 100$

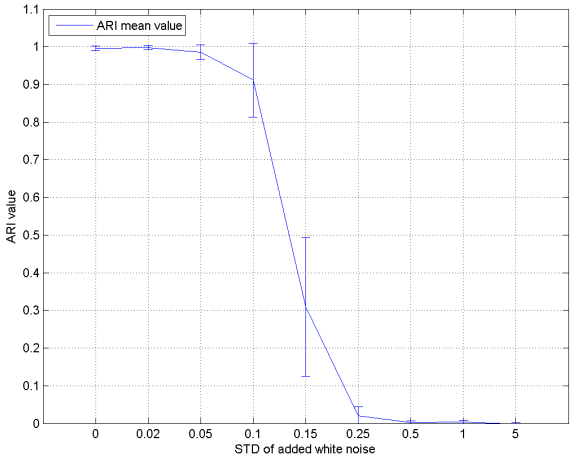


(b) FADDIS-a CEMR mean/avg for $K = 7$ and $N = 100$

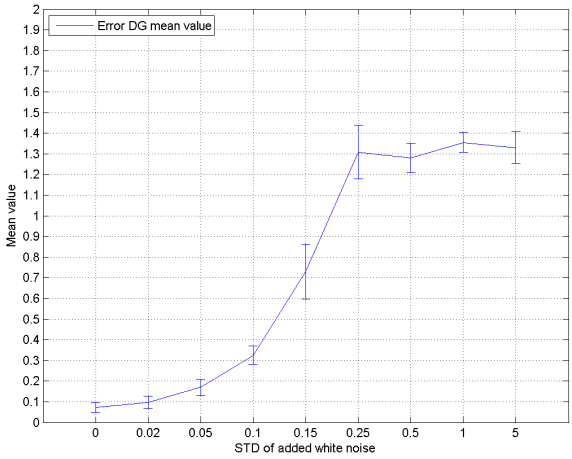


(c) FADDIS-a REI mean/avg for $K = 7$ and $N = 100$

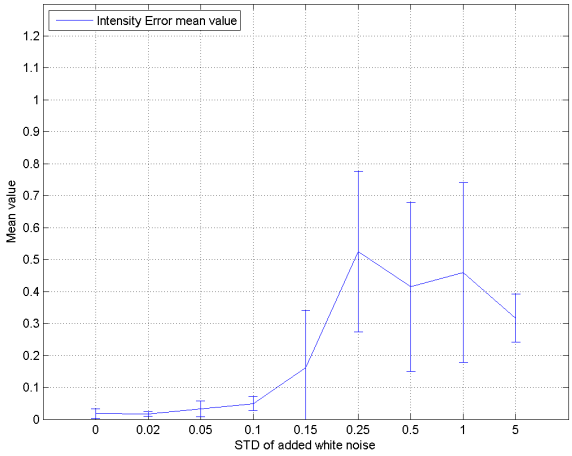
Figure B.12 FADDIS-a results with $K = 7$ and $N = 100$



(a) FADDIS-a ARI mean/avg for $K = 7$ and $N = 200$

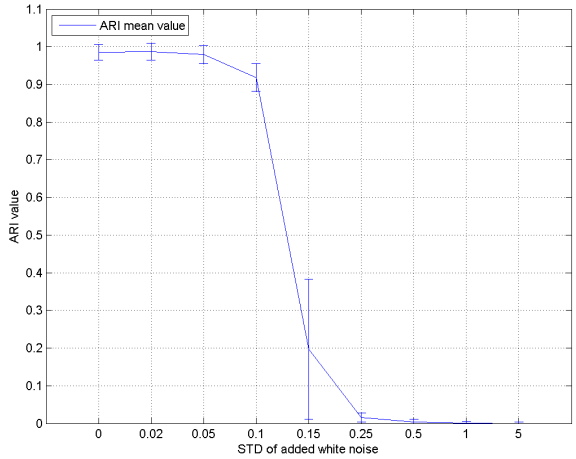


(b) FADDIS-a CEMR mean/avg for $K = 7$ and $N = 200$

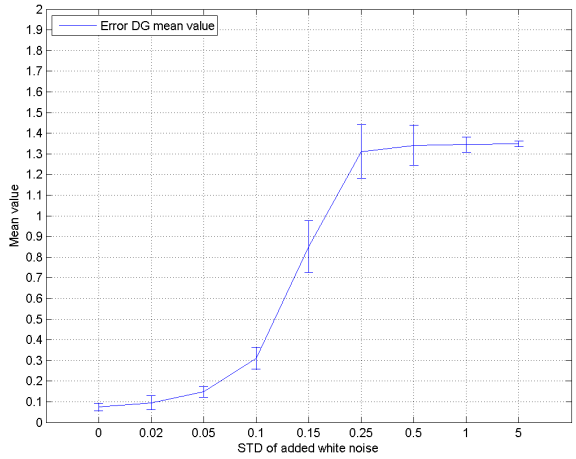


(c) FADDIS-a REI mean/avg for $K = 7$ and $N = 200$

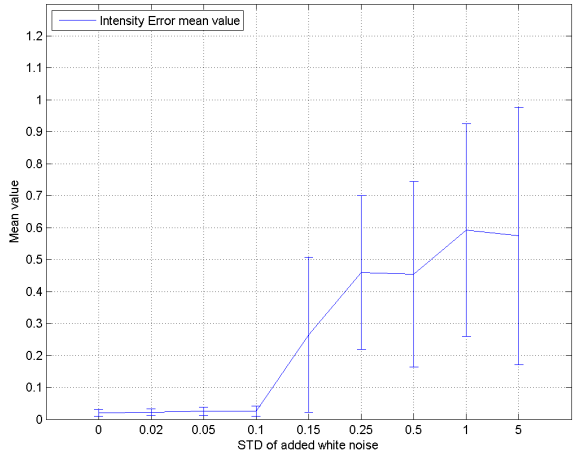
Figure B.13 FADDIS-a results with $K = 7$ and $N = 200$



(a) FADDIS-a ARI mean/avg for $K = 7$ and $N = 400$

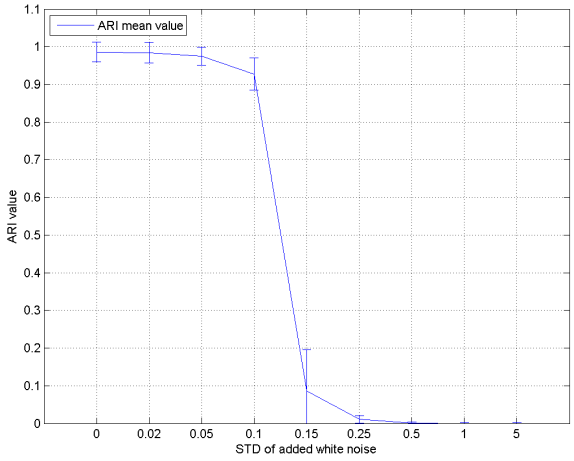


(b) FADDIS-a CEMR mean/avg for $K = 7$ and $N = 400$

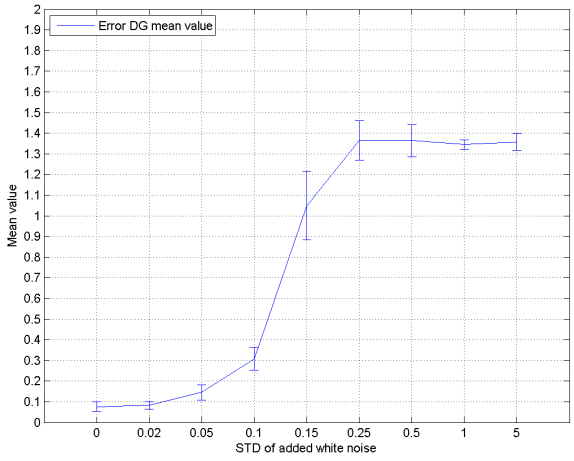


(c) FADDIS-a REI mean/avg for $K = 7$ and $N = 400$

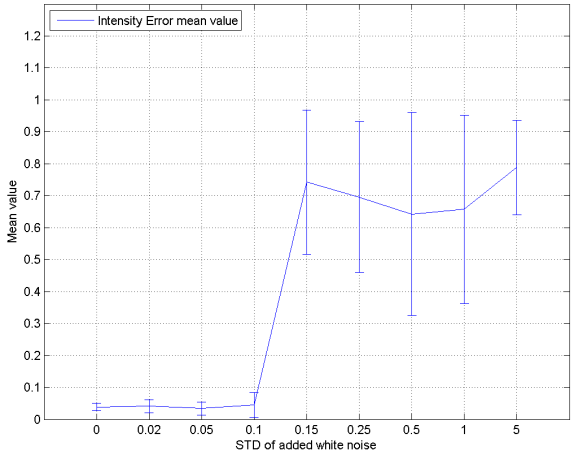
Figure B.14 FADDIS-a results with $K = 7$ and $N = 400$



(a) FADDIS-a ARI mean/avg for $K = 7$ and $N = 700$



(b) FADDIS-a CEMR mean/avg for $K = 7$ and $N = 700$

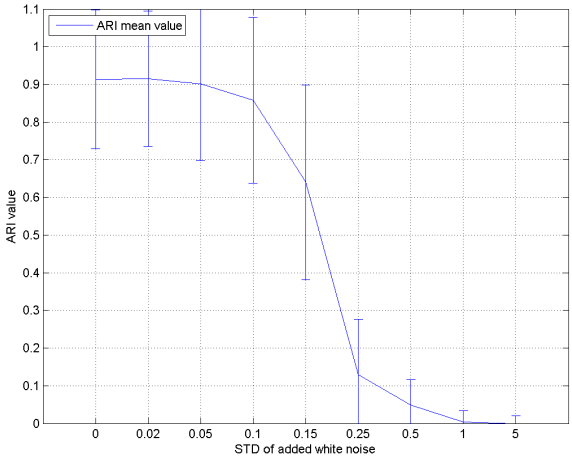


(c) FADDIS-a REI mean/avg for $K = 7$ and $N = 700$

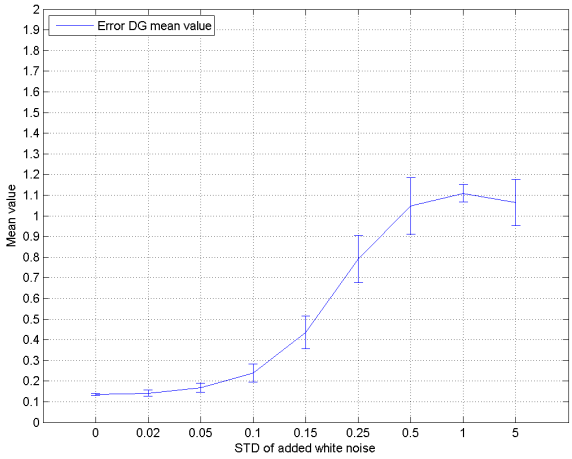
Figure B.15 FADDIS-a results with $K = 7$ and $N = 700$

B.4 FADDIS-m plot results

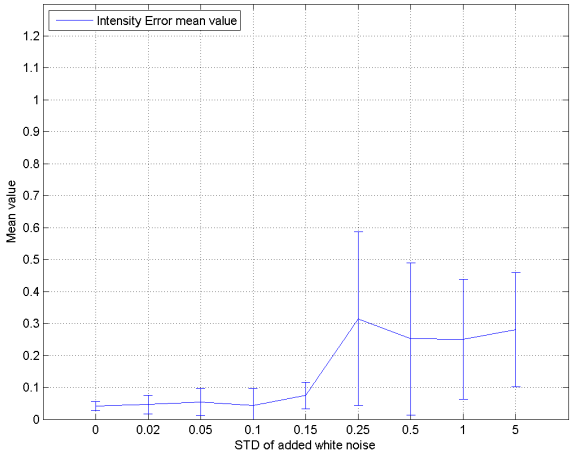
The plots for the FADDIS-m results for ARI, CEMR and REI avg/std in function of α for $K = \{3,5,7\}$ and $N = \{50,100,200,400,700\}$ will be shown in this section.



(a) FADDIS-m ARI mean/avg for $K = 3$ and $N = 50$

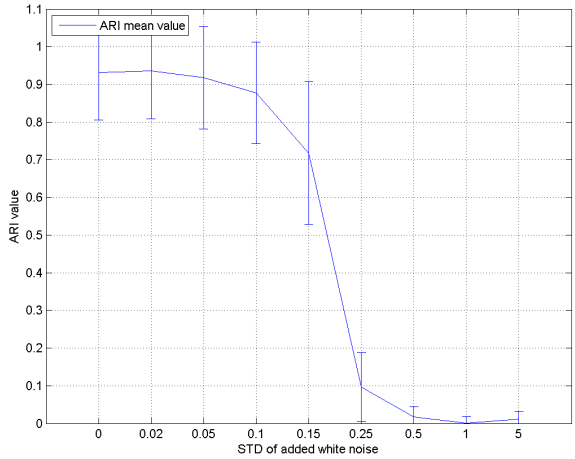


(b) FADDIS-m CEMR mean/avg for $K = 3$ and $N = 50$

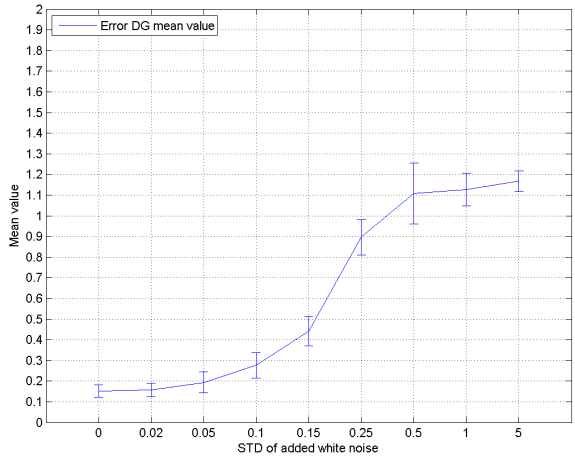


(c) FADDIS-m REI mean/avg for $K = 3$ and $N = 50$

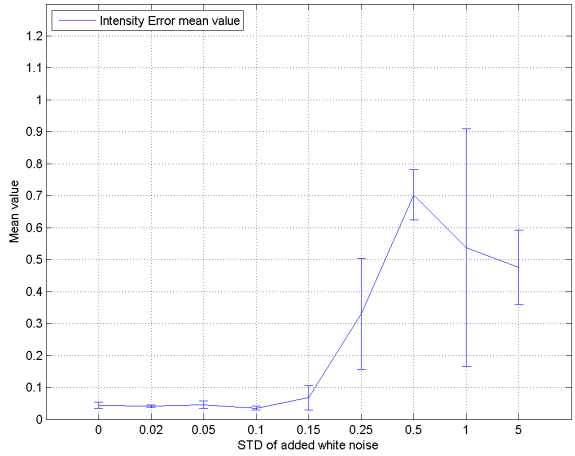
Figure B.16 FADDIS-m results with $K = 3$ and $N = 50$



(a) FADDIS-m ARI mean/avg for $K = 3$ and $N = 100$

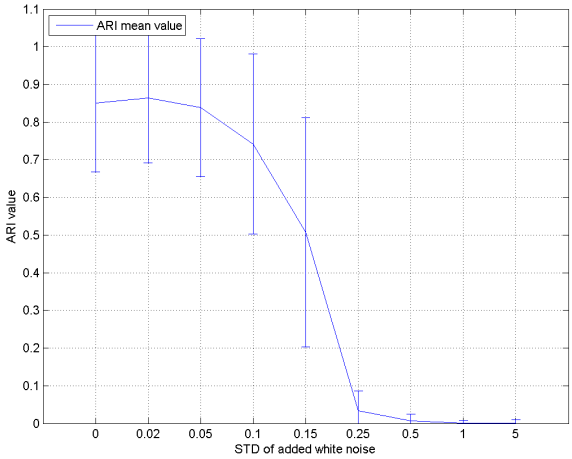


(b) FADDIS-m CEMR mean/avg for $K = 3$ and $N = 100$

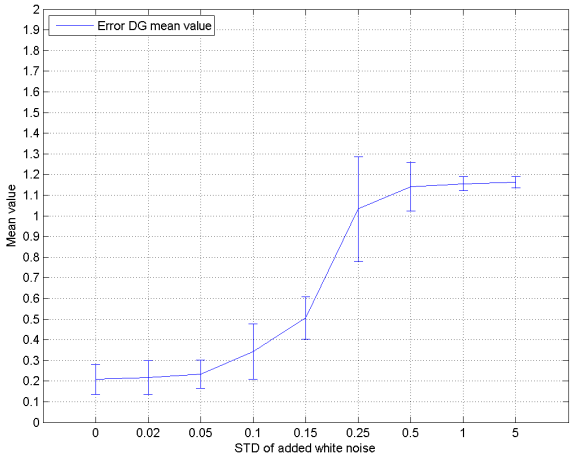


(c) FADDIS-m REI mean/avg for $K = 3$ and $N = 100$

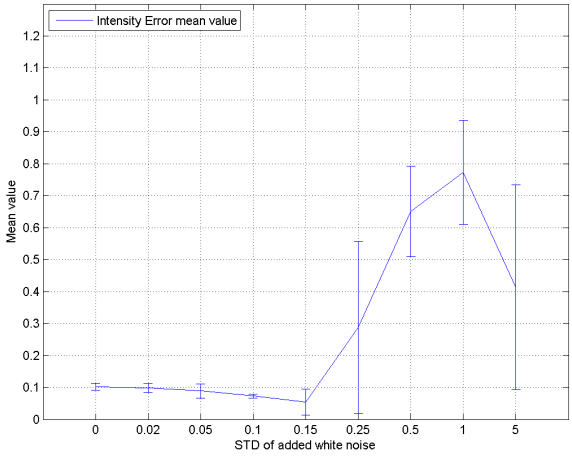
Figure B.17 FADDIS-m results with $K = 3$ and $N = 100$



(a) FADDIS-m ARI mean/avg for $K = 3$ and $N = 200$

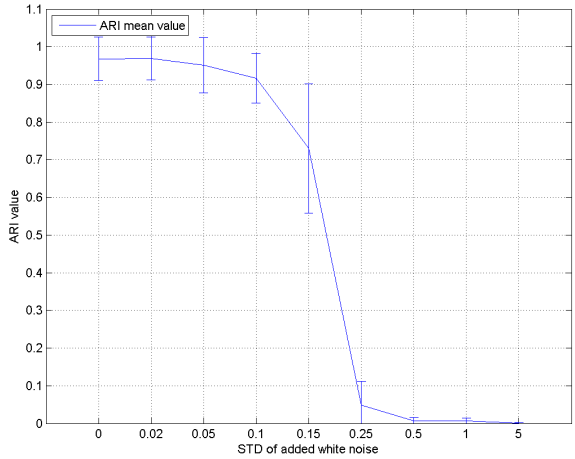


(b) FADDIS-m CEMR mean/avg for $K = 3$ and $N = 200$

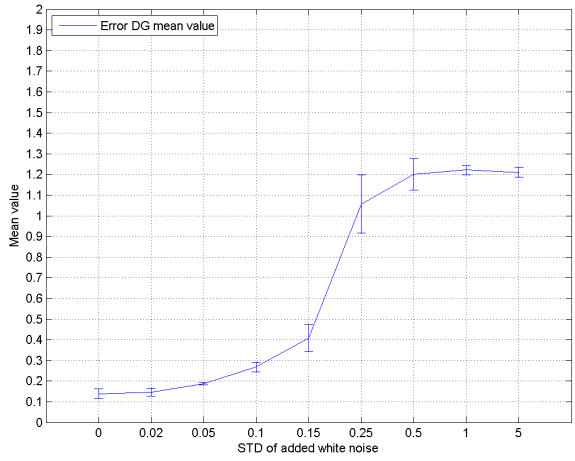


(c) FADDIS-m REI mean/avg for $K = 3$ and $N = 200$

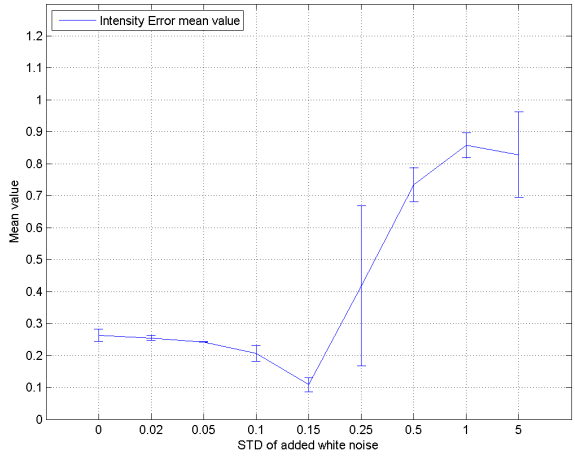
Figure B.18 FADDIS-m results with $K = 3$ and $N = 200$



(a) FADDIS-m ARI mean/avg for $K = 3$ and $N = 400$

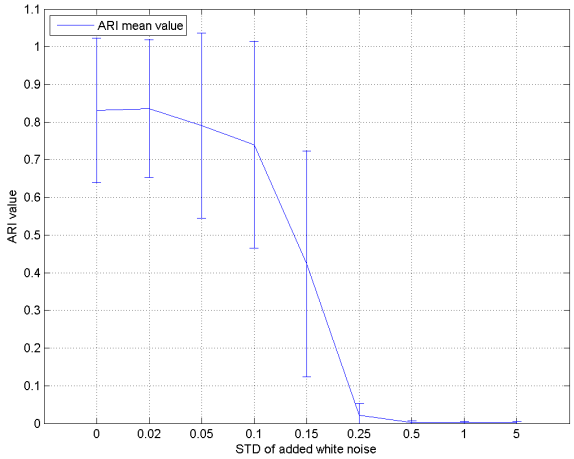


(b) FADDIS-m CEMR mean/avg for $K = 3$ and $N = 400$

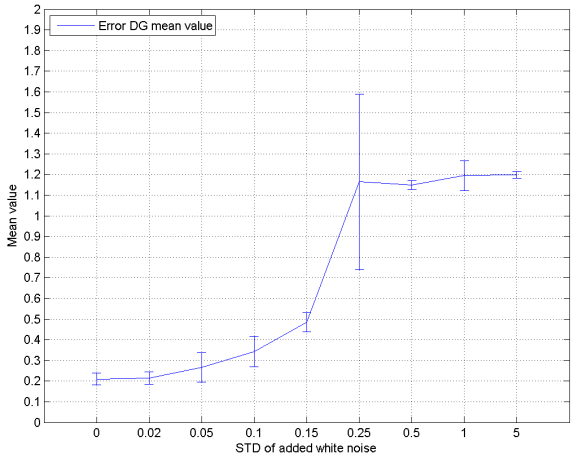


(c) FADDIS-m REI mean/avg for $K = 3$ and $N = 400$

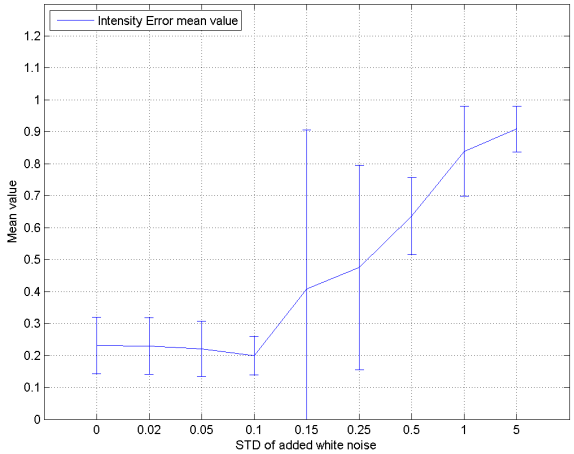
Figure B.19 FADDIS-m results with $K = 3$ and $N = 400$



(a) FADDIS-m ARI mean/avg for $K = 3$ and $N = 700$

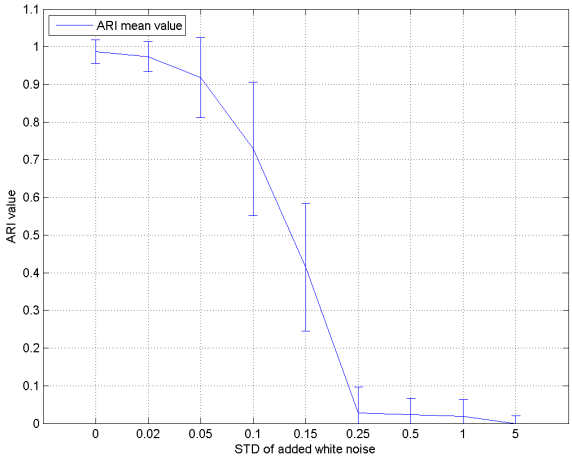


(b) FADDIS-m CEMR mean/avg for $K = 3$ and $N = 700$

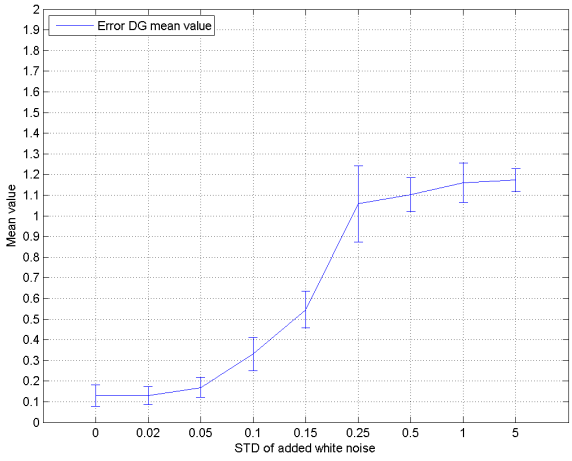


(c) FADDIS-m REI mean/avg for $K = 3$ and $N = 700$

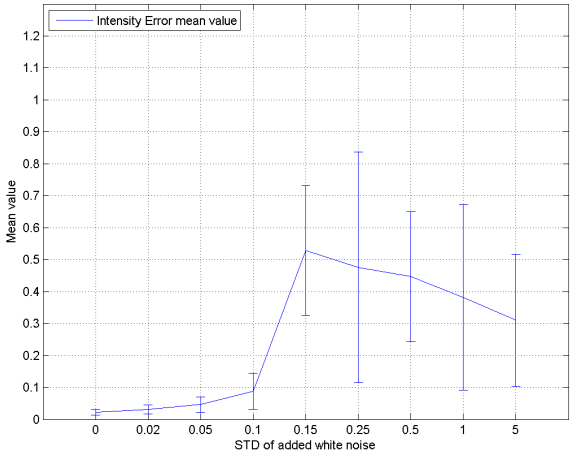
Figure B.20 FADDIS-m results with $K = 3$ and $N = 700$



(a) FADDIS-m ARI mean/avg for $K = 5$ and $N = 50$

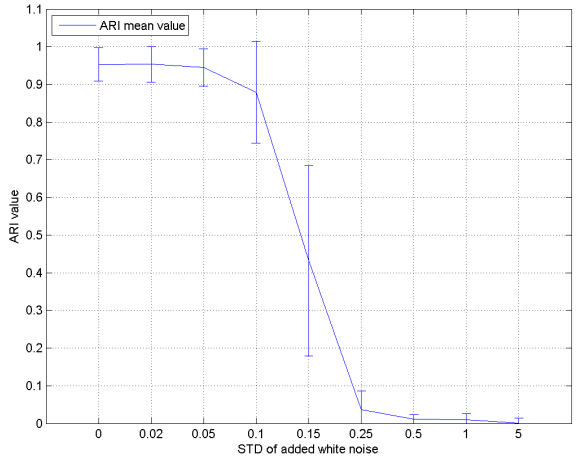


(b) FADDIS-m CEMR mean/avg for $K = 5$ and $N = 50$

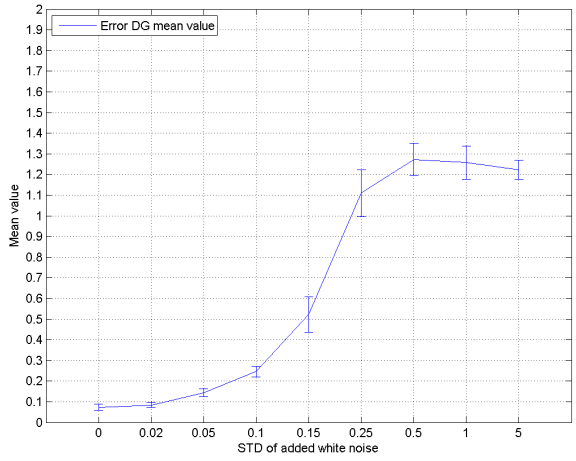


(c) FADDIS-m REI mean/avg for $K = 5$ and $N = 50$

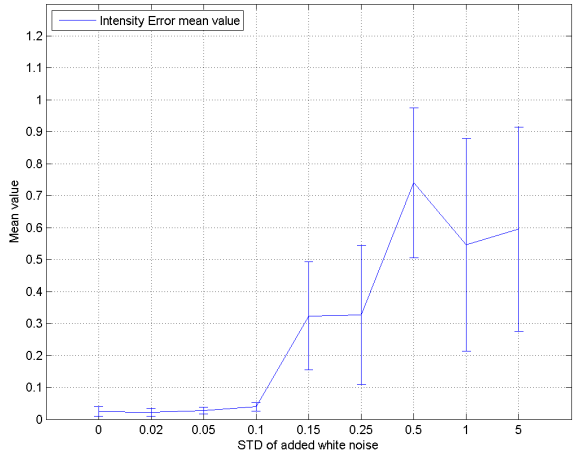
Figure B.21 FADDIS-m results with $K = 5$ and $N = 50$



(a) FADDIS-m ARI mean/avg for $K = 5$ and $N = 100$

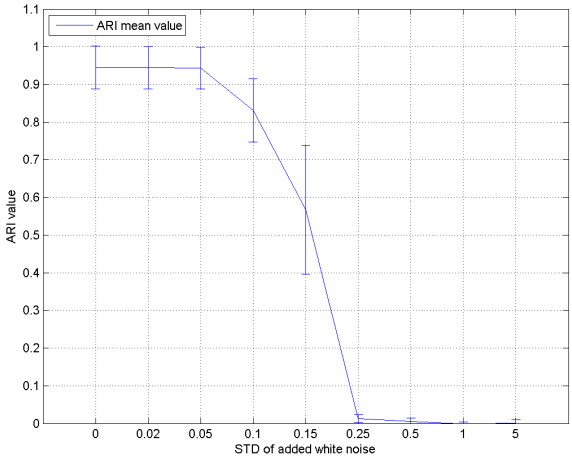


(b) FADDIS-m CEMR mean/avg for $K = 5$ and $N = 100$

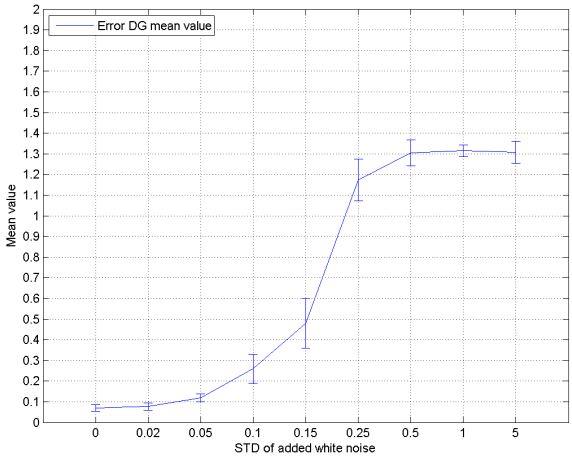


(c) FADDIS-m REI mean/avg for $K = 5$ and $N = 100$

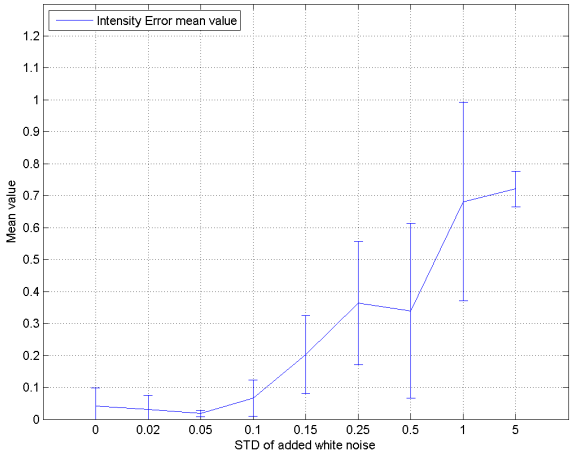
Figure B.22 FADDIS-m results with $K = 5$ and $N = 100$



(a) FADDIS-m ARI mean/avg for $K = 5$ and $N = 200$

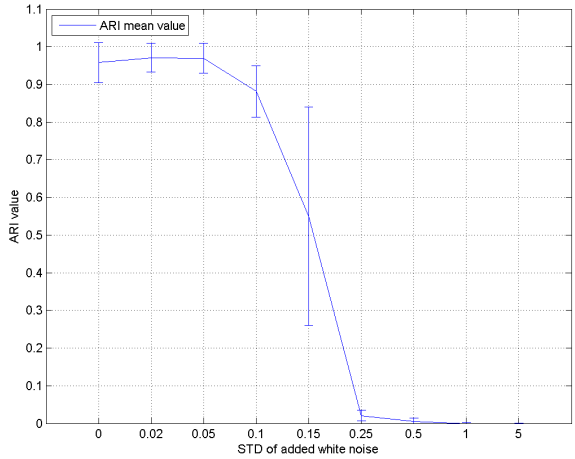


(b) FADDIS-m CEMR mean/avg for $K = 5$ and $N = 200$

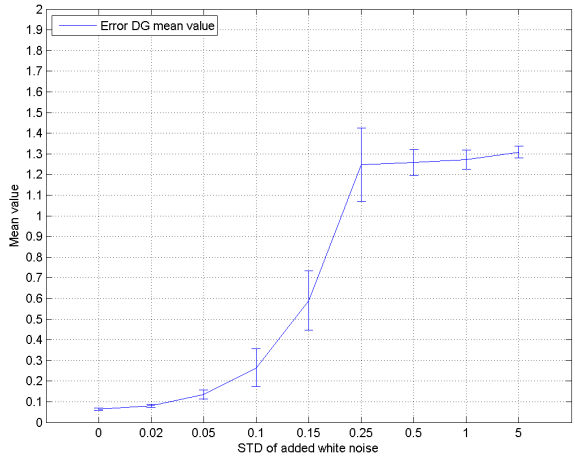


(c) FADDIS-m REI mean/avg for $K = 5$ and $N = 200$

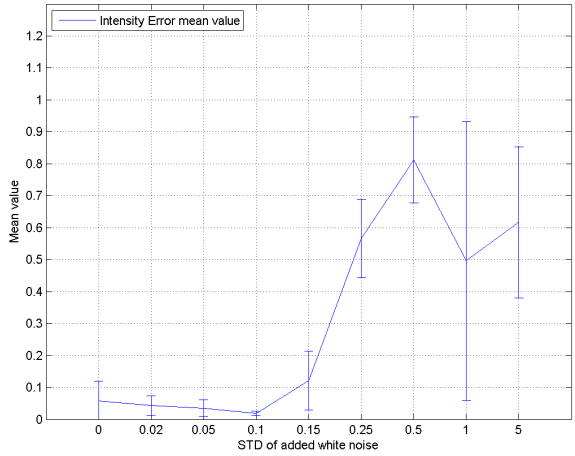
Figure B.23 FADDIS-m results with $K = 5$ and $N = 200$



(a) FADDIS-m ARI mean/avg for $K = 5$ and $N = 400$

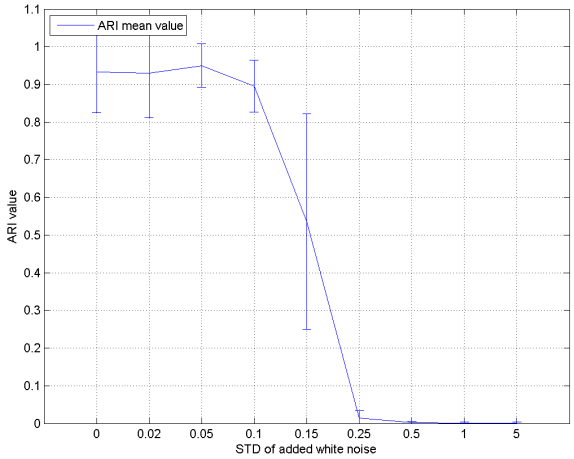


(b) FADDIS-m CEMR mean/avg for $K = 5$ and $N = 400$

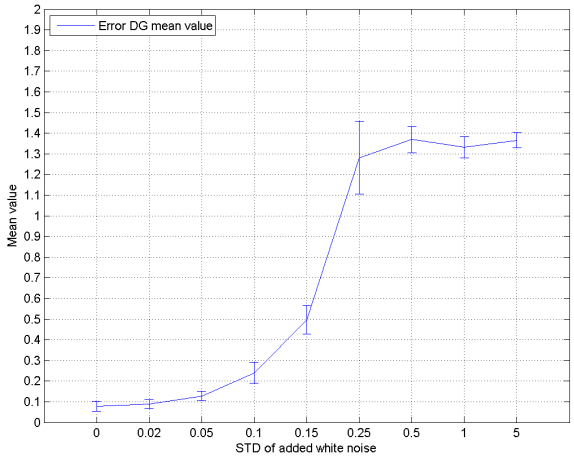


(c) FADDIS-m REI mean/avg for $K = 5$ and $N = 400$

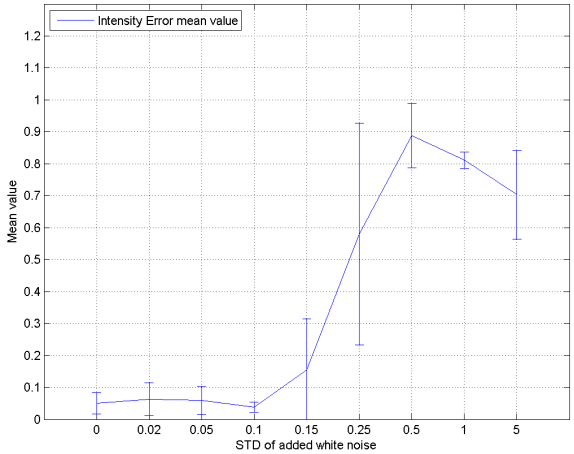
Figure B.24 FADDIS-m results with $K = 5$ and $N = 400$



(a) FADDIS-m ARI mean/avg for $K = 5$ and $N = 700$

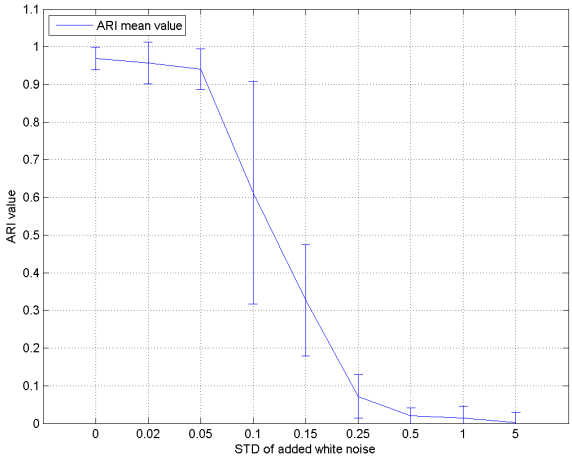


(b) FADDIS-m CEMR mean/avg for $K = 5$ and $N = 700$

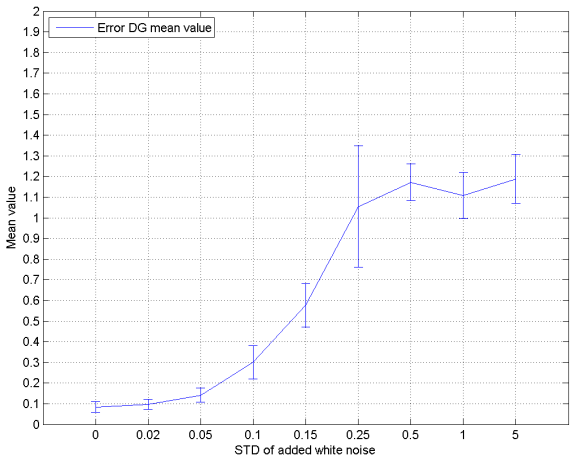


(c) FADDIS-m REI mean/avg for $K = 5$ and $N = 700$

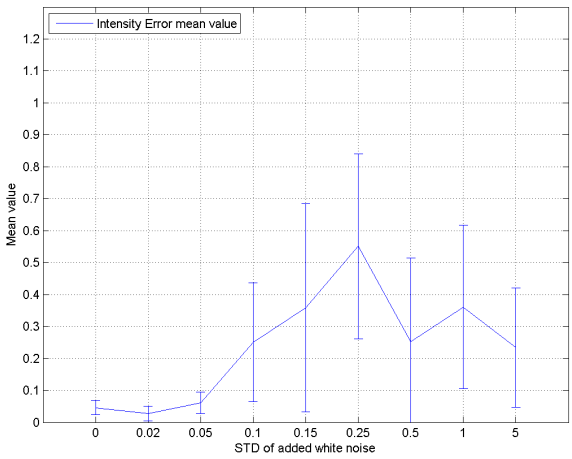
Figure B.25 FADDIS-m results with $K = 5$ and $N = 700$



(a) FADDIS-m ARI mean/avg for $K = 7$ and $N = 50$

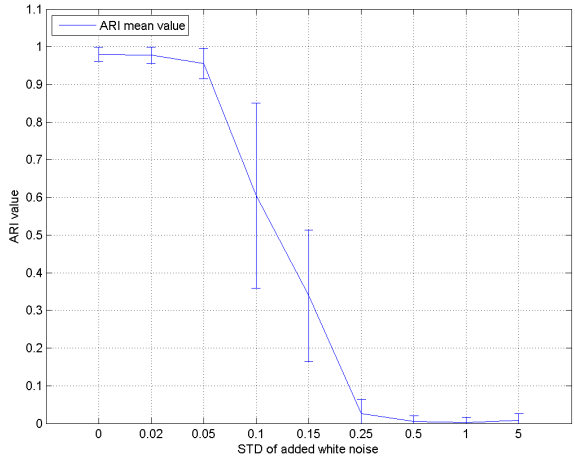


(b) FADDIS-m CEMR mean/avg for $K = 7$ and $N = 50$

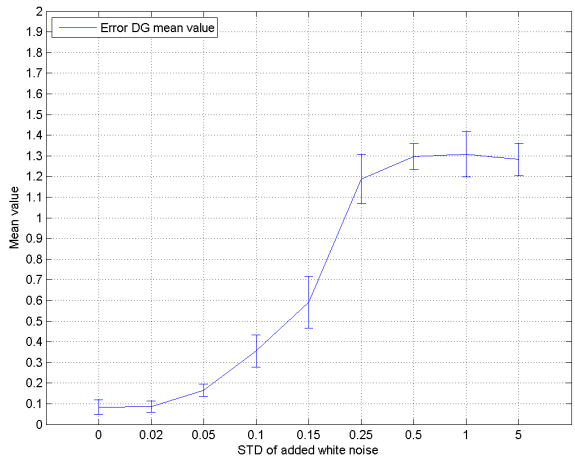


(c) FADDIS-m REI mean/avg for $K = 7$ and $N = 50$

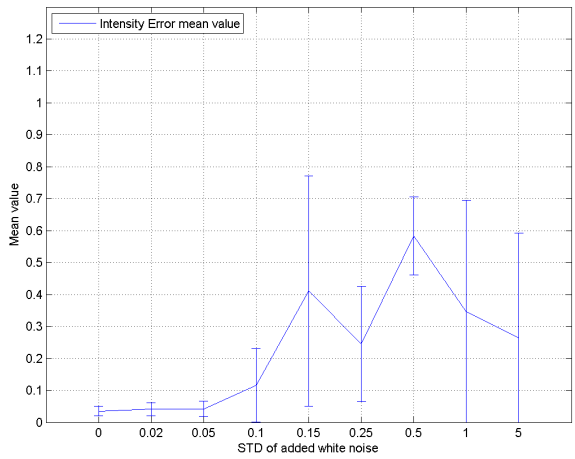
Figure B.26 FADDIS-m results with $K = 7$ and $N = 50$



(a) FADDIS-m ARI mean/avg for $K = 7$ and $N = 100$

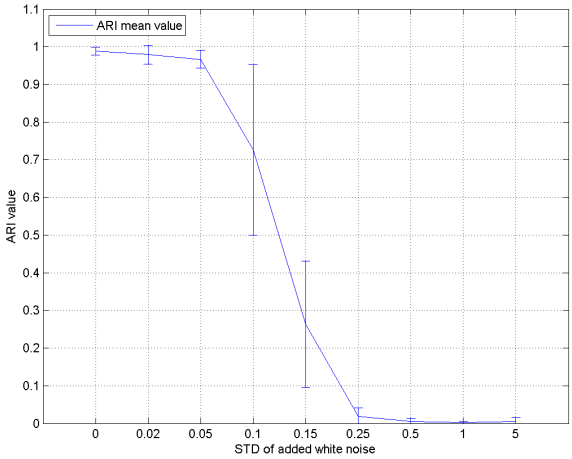


(b) FADDIS-m CEMR mean/avg for $K = 7$ and $N = 100$

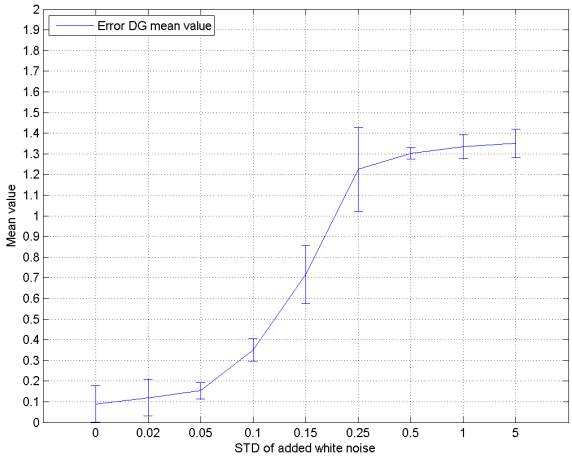


(c) FADDIS-m REI mean/avg for $K = 7$ and $N = 100$

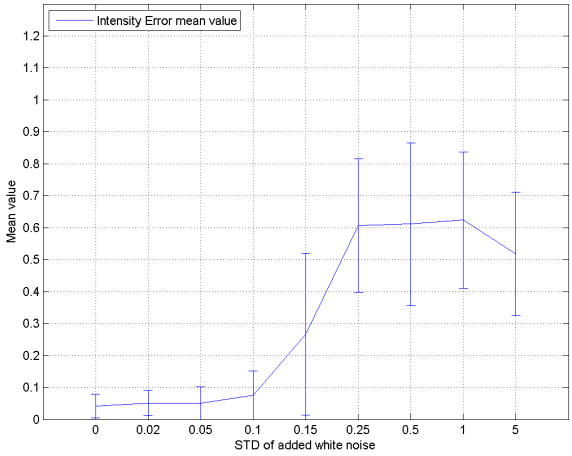
Figure B.27 FADDIS-m results with $K = 7$ and $N = 100$



(a) FADDIS-m ARI mean/avg for $K = 7$ and $N = 200$

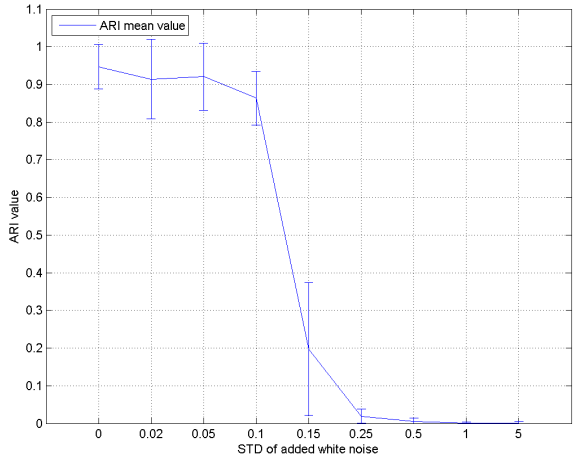


(b) FADDIS-m CEMR mean/avg for $K = 7$ and $N = 200$

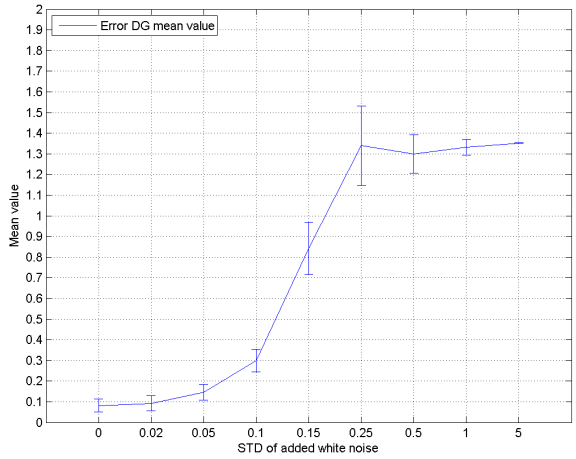


(c) FADDIS-m REI mean/avg for $K = 7$ and $N = 200$

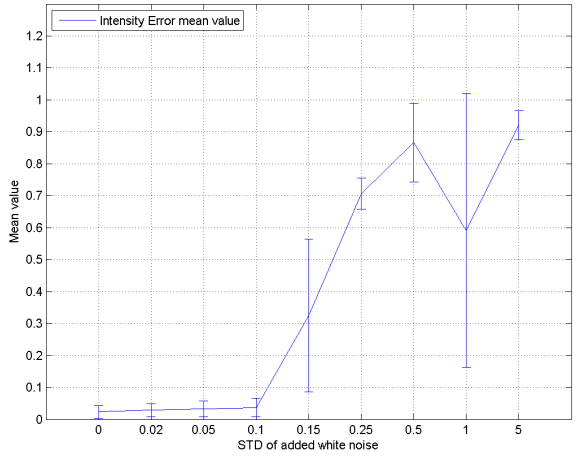
Figure B.28 FADDIS-m results with $K = 7$ and $N = 200$



(a) FADDIS-m ARI mean/avg for $K = 7$ and $N = 400$

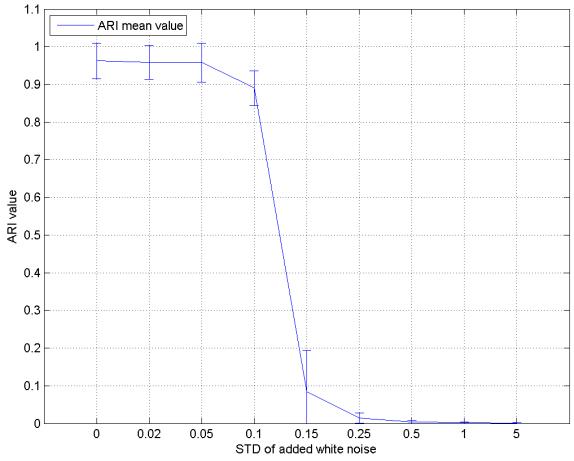


(b) FADDIS-m CEMR mean/avg for $K = 7$ and $N = 400$

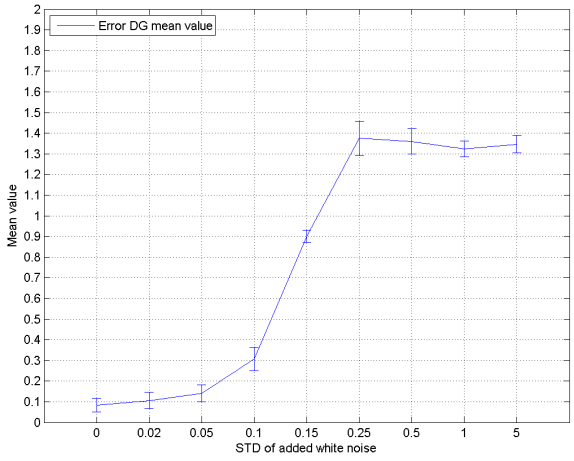


(c) FADDIS-m REI mean/avg for $K = 7$ and $N = 400$

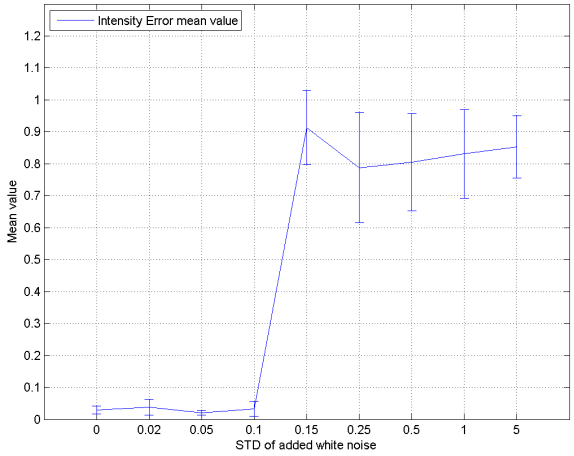
Figure B.29 FADDIS-m results with $K = 7$ and $N = 400$



(a) FADDIS-m ARI mean/avg for $K = 7$ and $N = 700$



(b) FADDIS-m CEMR mean/avg for $K = 7$ and $N = 700$

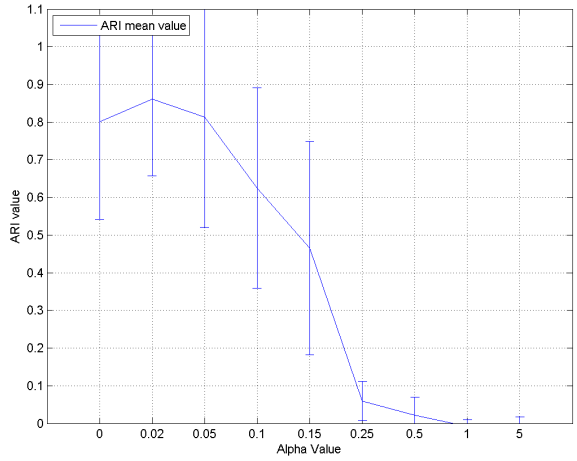


(c) FADDIS-m REI mean/avg for $K = 7$ and $N = 700$

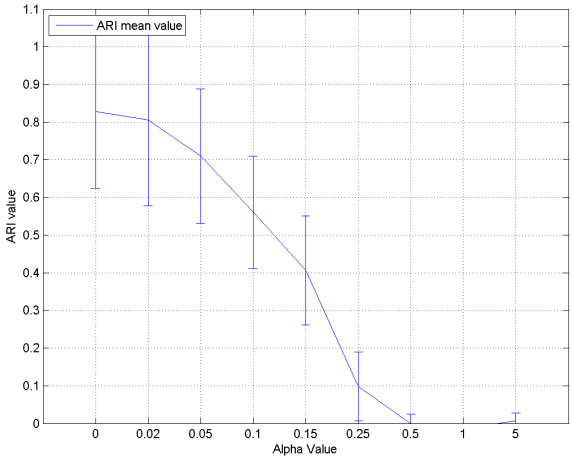
Figure B.30 FADDIS-m results with $K = 7$ and $N = 700$

B.5 FastMap FCM plot results

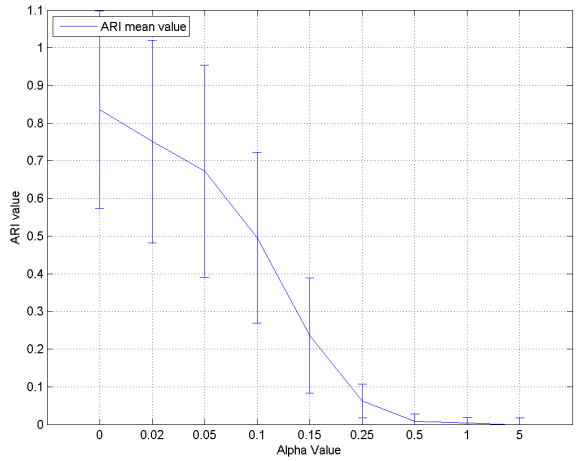
In this section, all the resulting ARI avg/std plots for FMFCM for $K = \{3,5,7\}$ and $N = \{50,100,200,400,700\}$ will be shown.



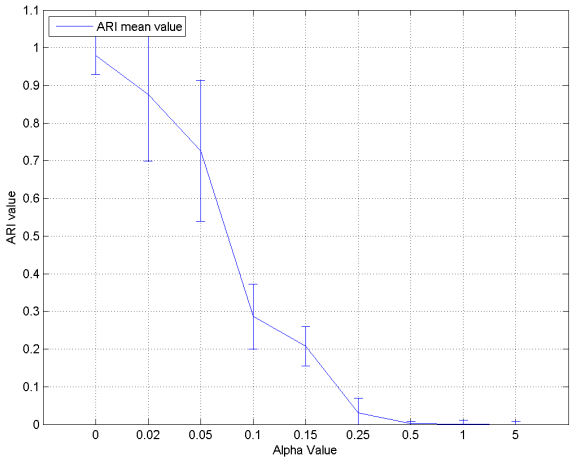
(a) FMFCM ARI mean/avg for $K = 3$ and $N = 50$



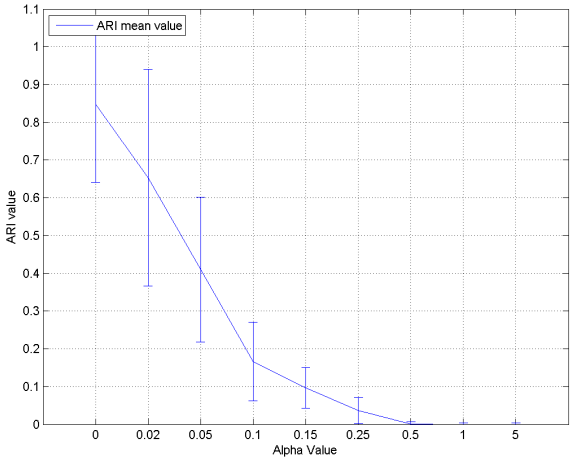
(b) FMFCM ARI mean/avg for $K = 3$ and $N = 100$



(c) FMFCM ARI mean/avg for $K = 3$ and $N = 200$

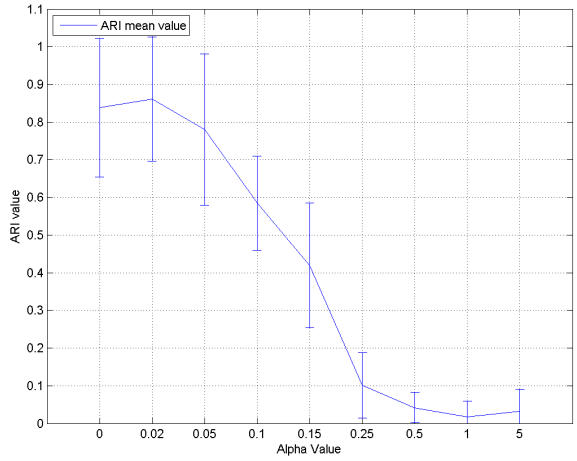


(d) FMFCM ARI mean/avg for $K = 3$ and $N = 400$

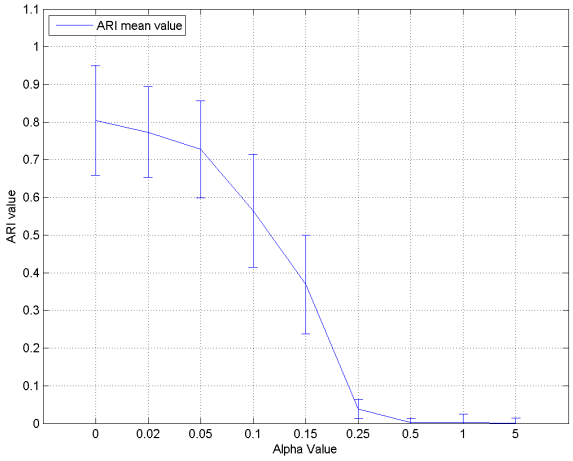


(e) FMFCM ARI mean/avg for $K = 3$ and $N = 700$

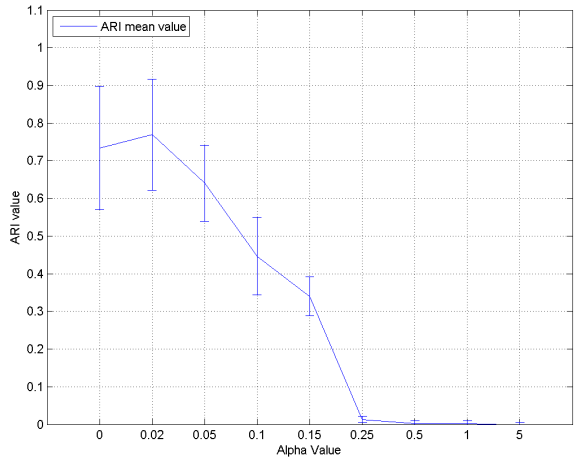
Figure B.31 FastMap FCM ARI mean/avg for $K = 3$



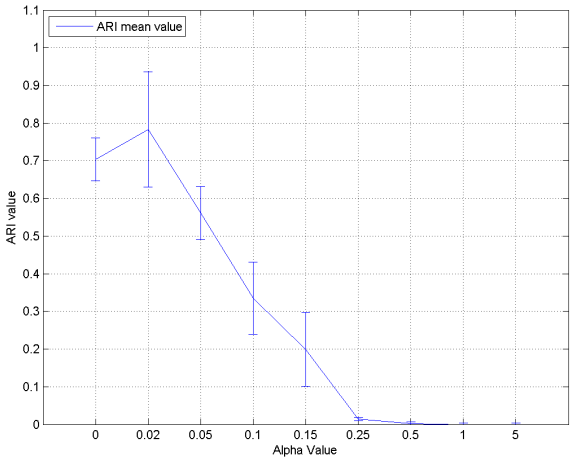
(a) FMFCM ARI mean/avg for $K = 5$ and $N = 50$



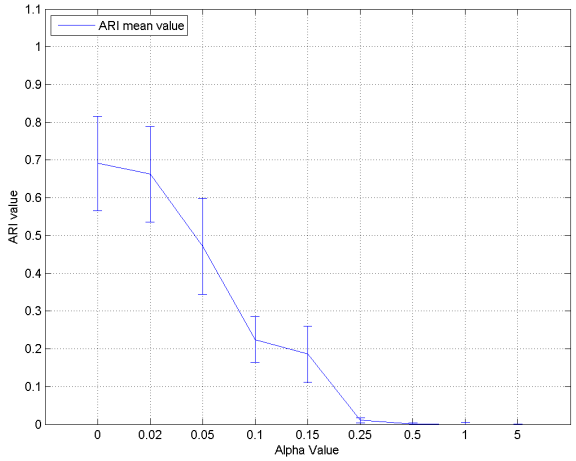
(b) FMFCM ARI mean/avg for $K = 5$ and $N = 100$



(c) FMFCM ARI mean/avg for $K = 5$ and $N = 200$

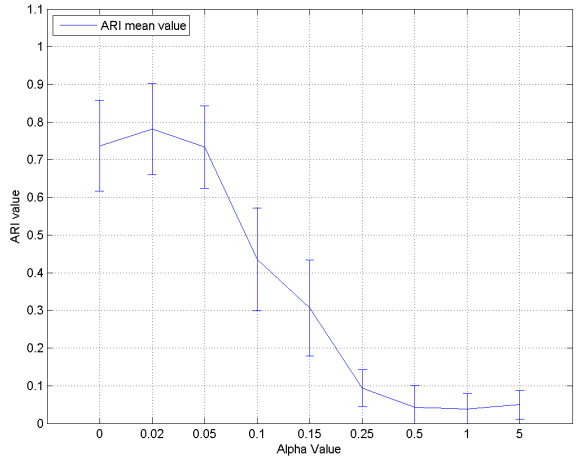


(d) FMFCM ARI mean/avg for $K = 5$ and $N = 400$

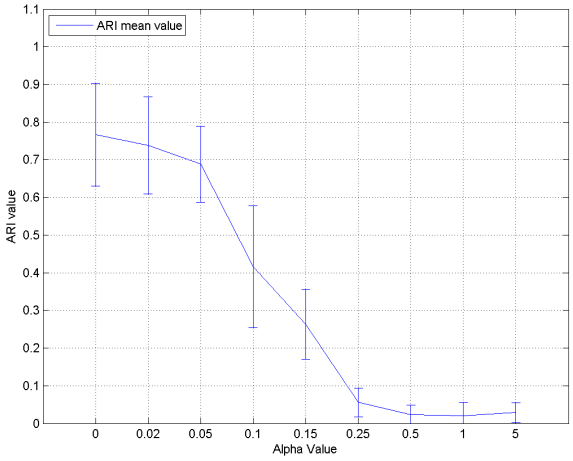


(e) FMFCM ARI mean/avg for $K = 5$ and $N = 700$

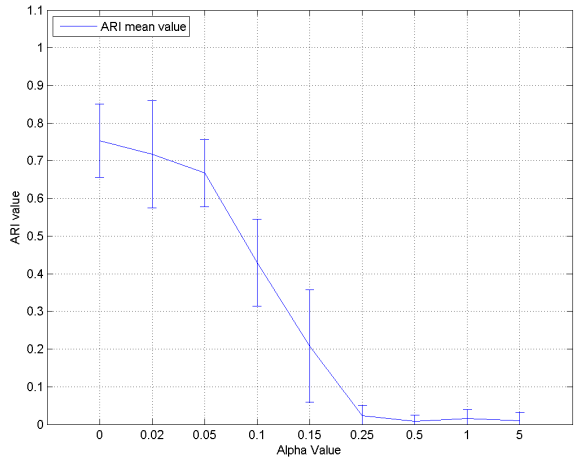
Figure B.32 FastMap FCM ARI mean/avg for $K = 5$



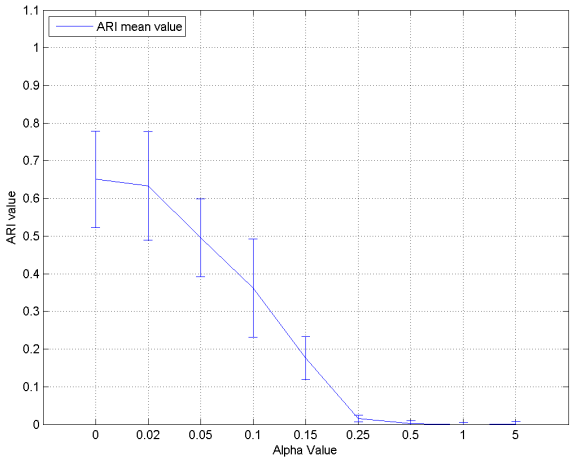
(a) FMFCM ARI mean/avg for $K = 7$ and $N = 50$



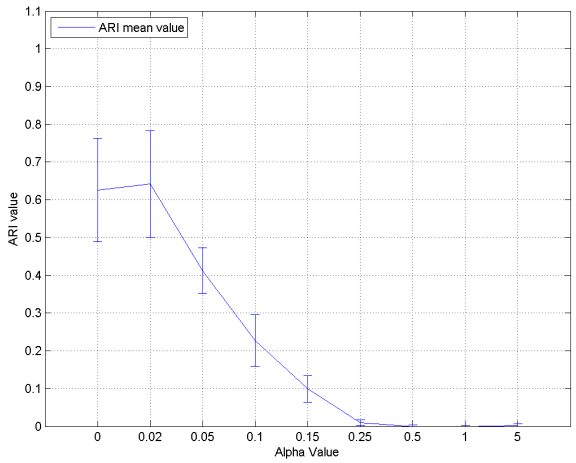
(b) FMFCM ARI mean/avg for $K = 7$ and $N = 100$



(c) FMFCM ARI mean/avg for $K = 7$ and $N = 200$



(d) FMFCM ARI mean/avg for $K = 7$ and $N = 400$

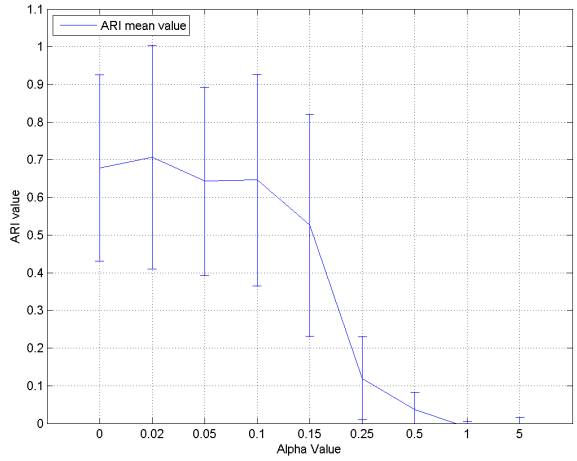


(e) FMFCM ARI mean/avg for $K = 7$ and $N = 700$

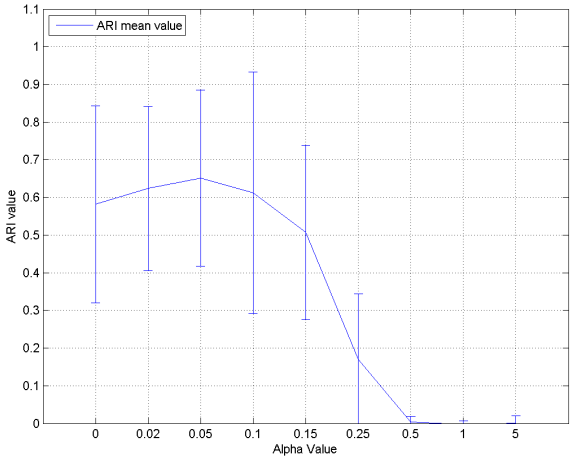
Figure B.33 FastMap FCM ARI mean/avg for $K = 7$

B.6 NERFCM plot results

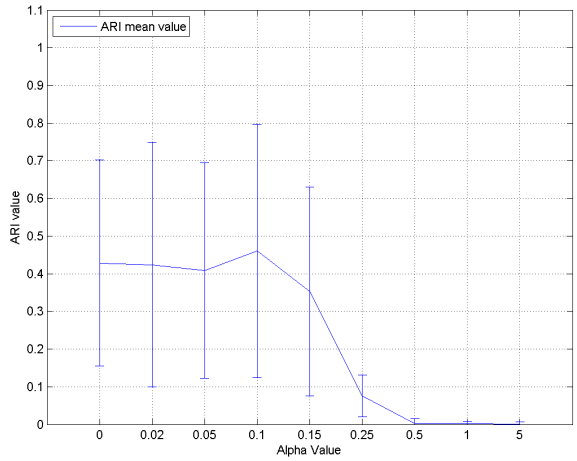
For each $K = \{3,5,7\}$ and $N = \{50,100,200,400,700\}$ the NERFCM resulting plots for the ARI index are shown in this section which will complement the conclusions made in Section 5.4.3.



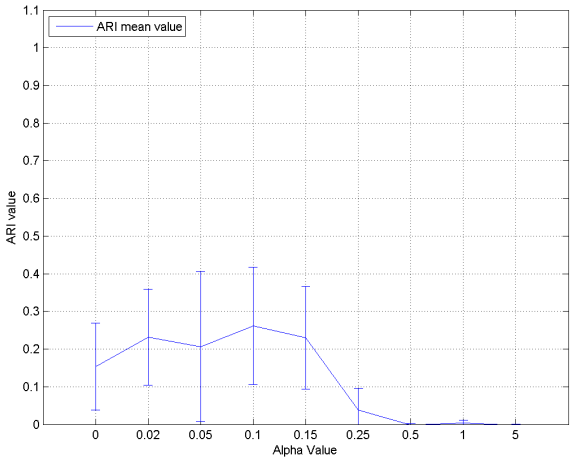
(a) NERFCM ARI mean/avg for $K = 3$ and $N = 50$



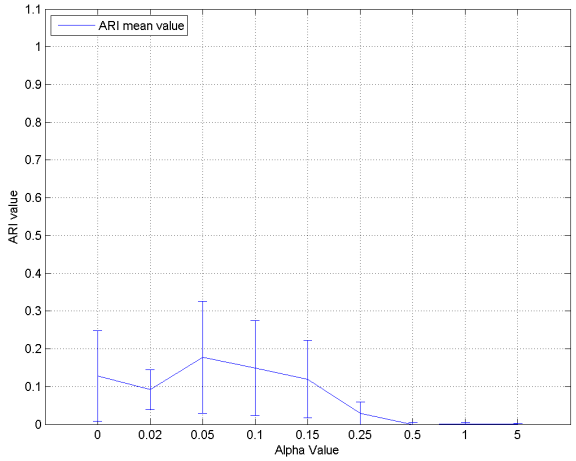
(b) NERFCM ARI mean/avg for $K = 3$ and $N = 100$



(c) NERFCM ARI mean/avg for $K = 3$ and $N = 200$

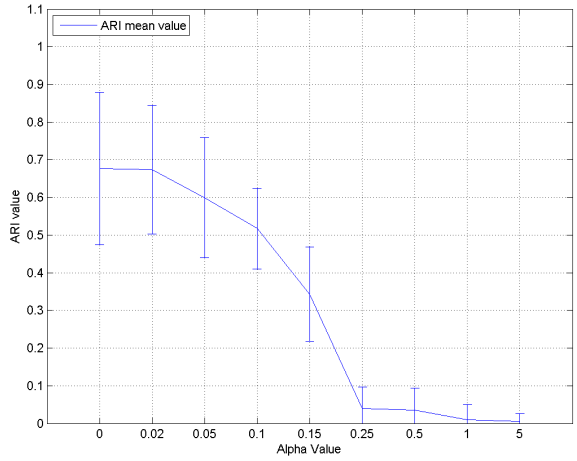


(d) NERFCM ARI mean/avg for $K = 3$ and $N = 400$

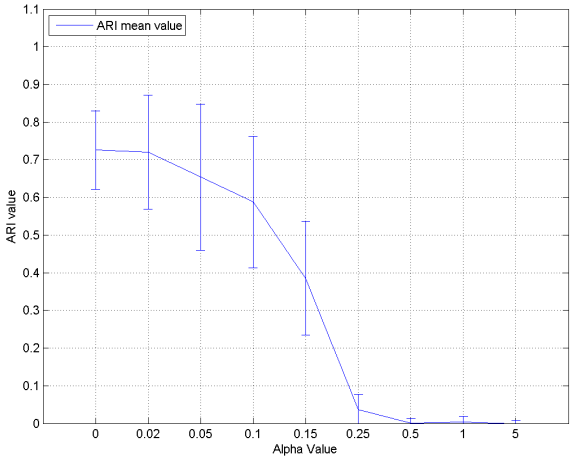


(e) NERFCM ARI mean/avg for $K = 3$ and $N = 700$

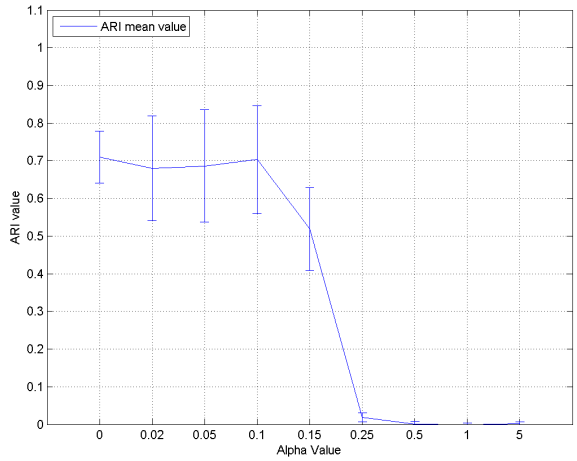
Figure B.34 NERFCM ARI mean/avg for $K = 3$



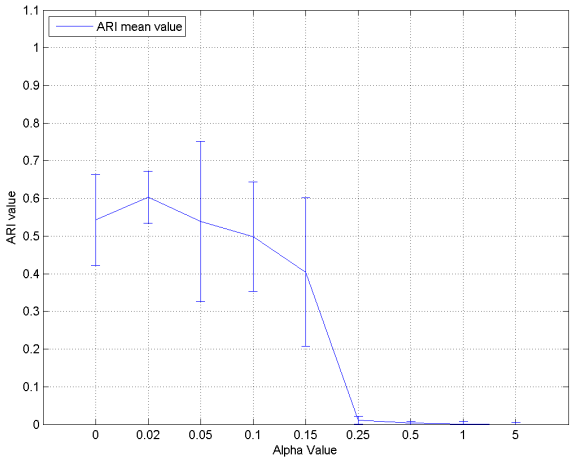
(a) NERFCM ARI mean/avg for $K = 5$ and $N = 50$



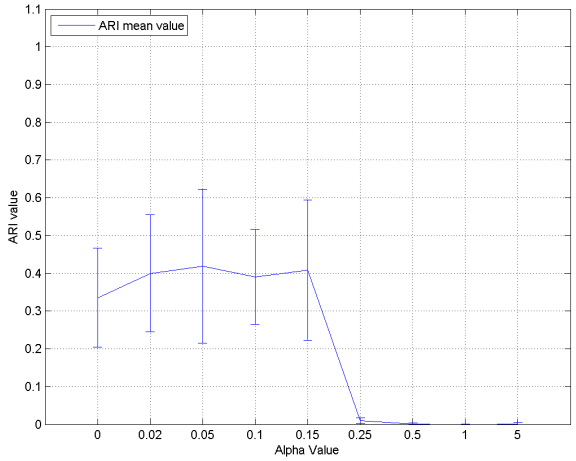
(b) NERFCM ARI mean/avg for $K = 5$ and $N = 100$



(c) NERFCM ARI mean/avg for $K = 5$ and $N = 200$

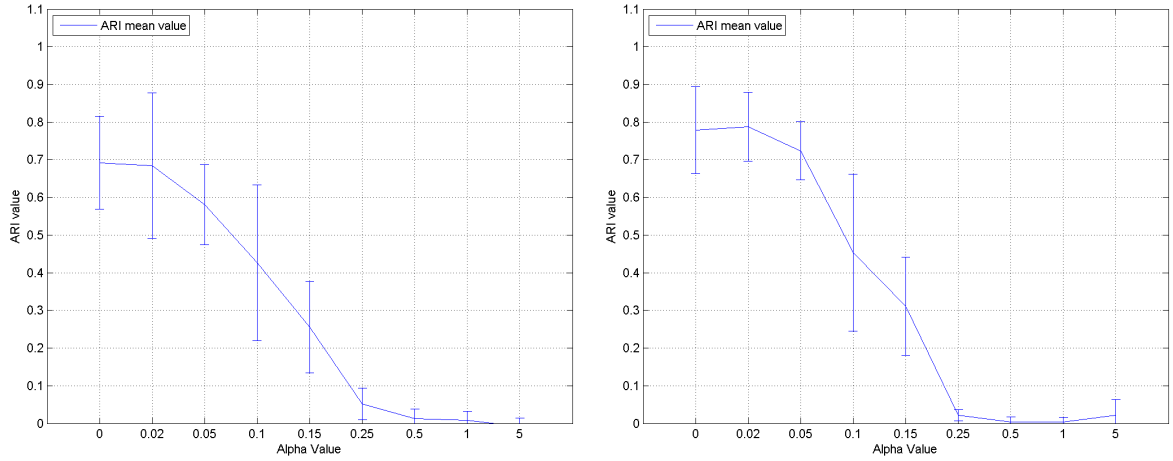


(d) NERFCM ARI mean/avg for $K = 5$ and $N = 400$



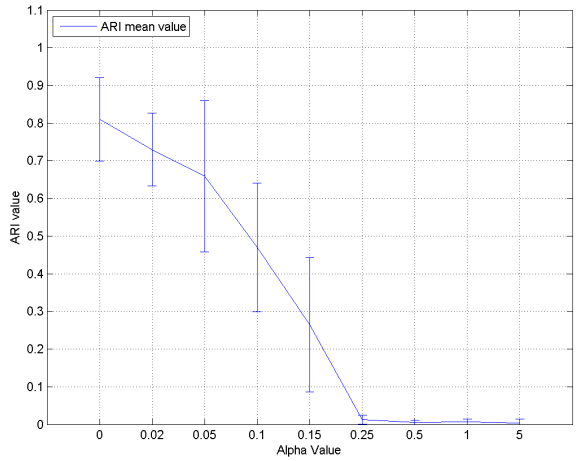
(e) NERFCM ARI mean/avg for $K = 5$ and $N = 700$

Figure B.35 NERFCM ARI mean/avg for $K = 5$

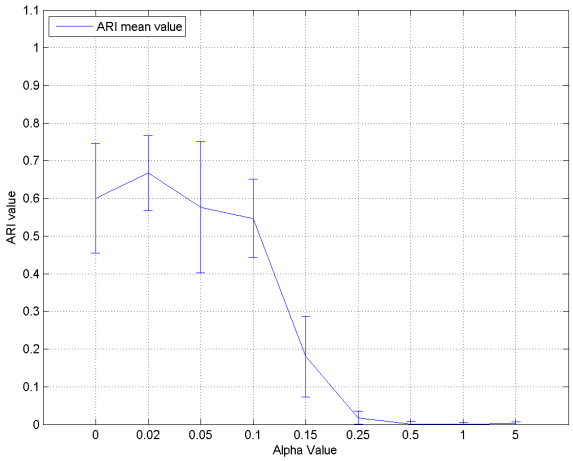


(a) NERFCM ARI mean/avg for $K = 7$ and $N = 50$

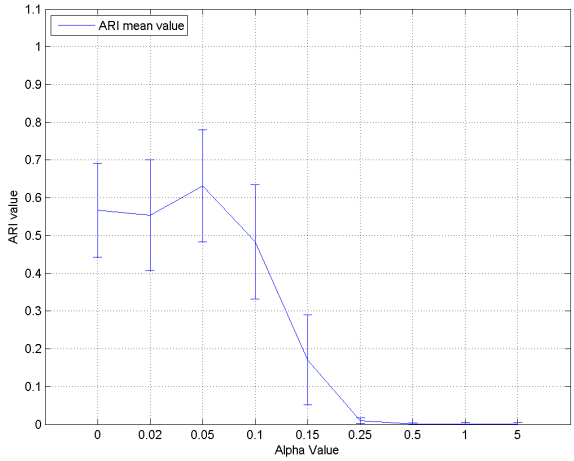
(b) NERFCM ARI mean/avg for $K = 7$ and $N = 100$



(c) NERFCM ARI mean/avg for $K = 7$ and $N = 200$



(d) NERFCM ARI mean/avg for $K = 7$ and $N = 400$



(e) NERFCM ARI mean/avg for $K = 7$ and $N = 700$

Figure B.36 NERFCM ARI mean/avg for $K = 7$

C. Results of Experiments with Benchmark Datasets

In the following subsections, for each dataset, the results for the other benchemark datasets will be presented.

For the datasets where a relational matrix is provided (i.e Fat Oil, Country and Microcomputer data), both FADDIS algorithms will be used directly with the matrices and also applying the Gaussian Kernel with and without the Lapin transformation pre-processing.

C.1 Cancer Data

This dataset has 2 clusters and 699 objects. One cluster has 458 objects and the other 241. Some objects have *NaN* in some attributes, and since the similarity matrix cannot be determined with these values, the corresponding objects were removed. After removing some objects, the total entities were 683, having the first cluster 444 objects and the second 239 objects.

The original data classification plot can be found in figure C.1.

C.1.0.1 Conventional Proximity

The obtained results for FMFCM and NERFCM using $K = 2$ and $K = 3$, as the input parameter for the number of clusters, are presented, respectively, in tables C.1 and C.2.

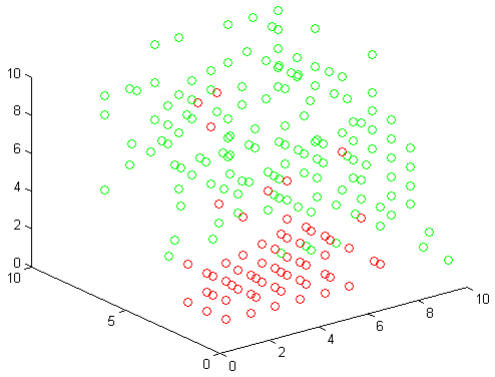


Figure C.1 Original Cancer Dataset Classification

<table>
<tr><td>K = 2</td><td>K=3</td></tr>
<tr><td>Confusion Matrix:
<table><tr><td></td><td></td><td>Original Classes</td><td></td></tr><tr><td></td><td></td><td>1</td><td>2</td></tr><tr><td>Predicted Classes</td><td>1</td><td>434</td><td>20</td></tr><tr><td></td><td>2</td><td>10</td><td>219</td></tr></table>
Adjusted Rand Index: 0.83
Mismatch Error: 0.04</td><td>Confusion Matrix:
<table><tr><td></td><td></td><td>Original Classes</td><td></td></tr><tr><td></td><td></td><td>1</td><td>2</td></tr><tr><td>Predicted Classes</td><td>1</td><td>430</td><td>8</td></tr><tr><td></td><td>2</td><td>14</td><td>116</td></tr><tr><td></td><td>3</td><td>0</td><td>115</td></tr></table>
Adjusted Rand Index: 0.77
Mismatch Error: 0.2</td></tr>
</table>

Table C.1 FMFCM results for Cancer Data

<table>
<tr><td>K = 2</td><td>K=3</td></tr>
<tr><td>Confusion Matrix:
<table><tr><td></td><td></td><td>Original Classes</td><td></td></tr><tr><td></td><td></td><td>1</td><td>2</td></tr><tr><td>Predicted Classes</td><td>1</td><td>432</td><td>9</td></tr><tr><td></td><td>2</td><td>12</td><td>230</td></tr></table>
Adjusted Rand Index: 0.88
Mismatch Error: 0.03</td><td>Confusion Matrix:
<table><tr><td></td><td></td><td>Original Classes</td><td></td></tr><tr><td></td><td></td><td>1</td><td>2</td></tr><tr><td>Predicted Classes</td><td>1</td><td>430</td><td>4</td></tr><tr><td></td><td>2</td><td>2</td><td>184</td></tr><tr><td></td><td>3</td><td>12</td><td>51</td></tr></table>
Adjusted Rand Index: 0.74
Mismatch Error: 0.1</td></tr>
</table>

Table C.2 NERFCM results for Cancer Data

C.1.0.2 Gaussian Kernel

FADDIS Stop Criteria Tuning:

- FADDIS-a: $\varepsilon = 0.001$
- FADDIS-m: $\varepsilon = 0.001$

FADDIS-a (Gaussian Kernel without Lapin)

· Number of clusters: 4
· Stop Condition: 'SC_2' - Cluster contribution is too small
· Confusion Matrix in table C.3 with Mismatch error: 0.47
· Unclustered data: 0%
· Adjusted Rand Index: 0.199
· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.4

		Original Classes	
		1	2
Predicted Classes	1	120	0
	2	94	239
	3	114	0
	4	116	0

Table C.3 Confusion Matrix Obtained from original classification and clustering results of Cancer Data with FADDIS-a applying the Gaussian Kernel

		Contribution	Intensity
Extracted Clusters	1	0.9001	229.13
	2	0.0240	37.444
	3	0.0169	31.428
	4	0.0007	6.2318

Table C.4 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-a for the Cancer Data using the Gaussian Kernel

FADDIS-m (Gaussian Kernel without Lapin)

· Number of clusters: 3

· Stop Condition: 'SC_1' - No positive weights at spectral clusters

· Confusion Matrix in table C.5 with Mismatch error: 0.47

· Unclustered data: 0%

· Adjusted Rand Index: 0.091

· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.6

		Original Classes	
		1	2
Predicted Classes	1	136	0
	2	173	237
	3	135	2

Table C.5 Confusion Matrix Obtained from original classification and clustering results of Cancer Data with FADDIS-m applying the Gaussian Kernel

		Contribution	Intensity
Extracted Clusters	1	0.9001	229.13
	2	0.0240	37.444
	3	0.0169	31.428

Table C.6 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-m for the Cancer Data using the Gaussian Kernel

The resulting plots for the Gaussian Kernel tests are in figure C.2.

FADDIS-a (Gaussian Kernel with Lapin)

· Number of clusters: 4

· Stop Condition: 'SC_3' - Residual is too small

· Confusion Matrix in table C.7 with Mismatch error: 0.03

· Unclustered data: 0.73%

· Adjusted Rand Index: 0.87

· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.8

		Original Classes	
		1	2
Predicted Classes	1	424	4
	2	17	231
	3	0	1
	4	0	1

Table C.7 Confusion Matrix Obtained from original classification and clustering results of Cancer Data with FADDIS-a with Lapin applying the Gaussian Kernel

		Contribution	Intensity
Extracted Clusters	1	0.7809	319.67
	2	0.1379	134.34
	3	0.0456	77.264
	4	0.0099	36.131

Table C.8 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-a with Lapin for the Cancer Data using the Gaussian Kernel

FADDIS-m (Gaussian Kernel with Lapin)

· Number of clusters: 4

· Stop Condition: 'SC_3' - Residual is too small

· Confusion Matrix in table C.9 with Mismatch error: 0.03

· Unclustered data: 0.73%

· Adjusted Rand Index: 0.87

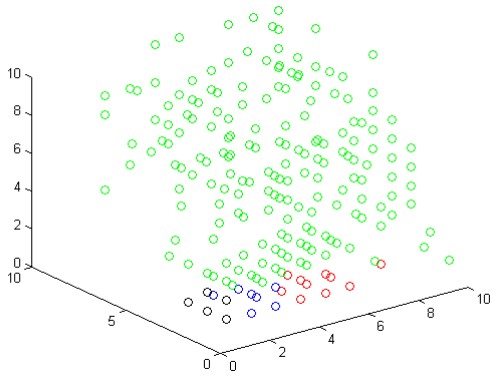
· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.10

		Original Classes	
		1	2
Predicted Classes	1	424	4
	2	17	231
	3	0	1
	4	0	1

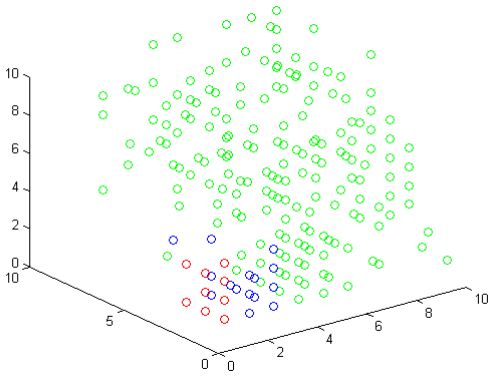
Table C.9 Confusion Matrix Obtained from original classification and clustering results of Cancer Data with FADDIS-m with Lapin applying the Gaussian Kernel

		Contribution	Intensity
Extracted Clusters	1	0.7809	319.67
	2	0.1379	134.34
	3	0.0456	77.264
	4	0.0099	36.131

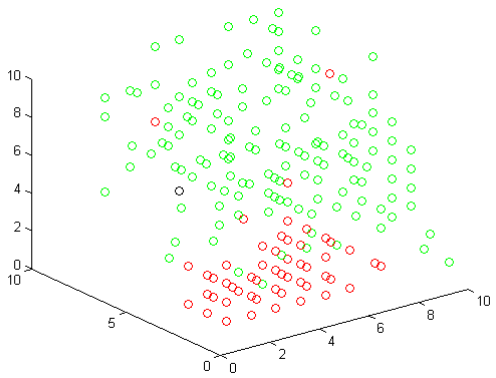
Table C.10 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-m with Lapin for the Cancer Data using the Gaussian Kernel



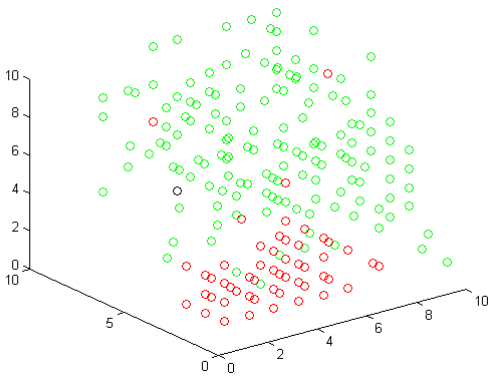
(a) FADDIS-a



(b) FADDIS-m



(c) FADDIS-a with Lapin



(d) FADDIS-m with Lapin

Figure C.2 Cancer Data Clustering Results for FADDIS with Gaussian Kernel

C.2 Wine Data

This dataset has 3 clusters, and the number of entities for each one is 59, 71 and 48, making a total of 178 entities.

The original classification data plot is in figure C.3.

C.2.0.3 Conventional Proximity

The obtained results for FMFCM and NERFCM using $K = 2$, $K = 3$ and $K = 4$, as the input parameter for the number of clusters, are presented, respectively, in tables C.11 and C.12.

K = 2

Confusion Matrix:

		Original Classes		
		1	2	3
Predicted Classes	1	50	4	2
	2	9	67	46

Adjusted Rand Index: 0.36
Mismatch error: 0.34

K=3

Confusion Matrix:

		Original Classes		
		1	2	3
Predicted Classes	1	45	1	0
	2	0	50	21
	3	14	20	27

Adjusted Rand Index: 0.35
Mismatch error: 0.35

K=4

Confusion Matrix:

		Original Classes		
		1	2	3
Predicted Classes	1	27	4	2
	2	0	45	16
	3	9	22	30
	4	23	0	0

Adjusted Rand Index: 0.27
Mismatch error: 0.43

Table C.11 FMFCM results for Wine Data

K = 2

Confusion Matrix:

		Original Classes		
		1	2	3
Predicted Classes	1	53	4	5
	2	6	67	43

Adjusted Rand Index: 0.40
Mismatch error: 0.33

K=3

Confusion Matrix:

		Original Classes		
		1	2	3
Predicted Classes	1	46	2	0
	2	0	49	17
	3	13	20	31

Adjusted Rand Index: 0.37
Mismatch error: 0.35

K=4

Confusion Matrix:

		Original Classes		
		1	2	3
Predicted Classes	1	29	4	4
	2	0	45	14
	3	7	22	30
	4	23	0	0

Adjusted Rand Index: 0.29
Mismatch error: 0.42

Table C.12 NERFCM results for Wine Data

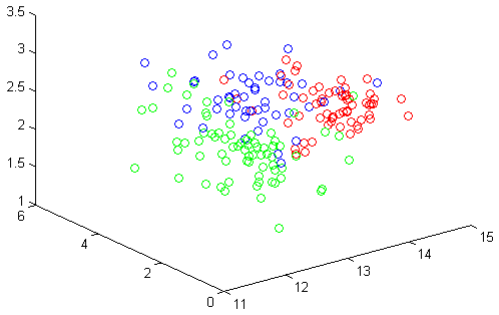


Figure C.3 Original Wine Dataset Classification

C.2.0.4 Gaussian Kernel

Stop Criteria Tuning:

- FADDIS-a: $\varepsilon = 0$
- FADDIS-m: $\varepsilon = 0$

FADDIS-a (Gaussian Kernel without Lapin)

· Number of clusters: 5
· Stop Condition: 'SC_4' - Maximum number of clusters reached
· Confusion Matrix in table C.13 with Mismatch error: 0.43
· Unclustered data: 0%
· Adjusted Rand Index: 0.343
· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.14

The respective plots are in figure C.4.

		Original Classes		
		1	2	3
Predicted Classes	1	50	4	2
	2	0	31	2
	3	9	11	20
	4	0	14	10
	5	0	11	14

Table C.13 Confusion Matrix Obtained from original classification and clustering results of Wine Data with FADDIS-a using the Gaussian Kernel

		Contribution	Intensity
Extracted Clusters	1	0.1749	13.489
	2	0.1504	12.508
	3	0.0871	9.521
	4	0.0612	7.977
	5	0.0594	7.859

Table C.14 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-a for the Wine Data using Gaussian Kernel

FADDIS-m (Gaussian Kernel without Lapin)

· Number of clusters: 6

· Stop Condition: 'SC_4' - Maximum number of clusters reached

· Confusion Matrix in table C.15 with Mismatch error: 0.45

· Unclustered data: 0%

· Adjusted Rand Index: 0.326

· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.16

		Original Classes		
		1	2	3
Predicted Classes	1	50	4	2
	2	0	29	2
	3	9	11	19
	4	0	12	8
	5	0	6	9
	6	0	9	8

Table C.15 Confusion Matrix Obtained from original classification and clustering results of Wine Data with FADDIS-m using the Gaussian Kernel

		Contribution	Intensity
Extracted Clusters	1	0.17488	13.489
	2	0.15037	12.508
	3	0.08712	9.520
	4	0.06116	7.977
	5	0.00991	3.212
	6	0.05936	7.859

Table C.16 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-m for the Wine Data using Gaussian Kernel

FADDIS-a (Gaussian Kernel with Lapin)

· Number of clusters: 5

· Stop Condition: 'SC_4' - Maximum number of clusters reached

· Confusion Matrix in table C.17 with Mismatch error: 0.45

· Unclustered data: 0%

· Adjusted Rand Index: 0.236

· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.18

		Original Classes		
		1	2	3
Predicted Classes	1	40	2	0
	2	12	45	33
	3	1	24	15
	4	1	0	0
	5	5	0	0

Table C.17 Confusion Matrix Obtained from original classification and clustering results of Wine Data with FADDIS-a with Lapin using the Gaussian Kernel

		Contribution	Intensity
Extracted Clusters	1	0.99875	4.08e+7
	2	3.890e-7	25467
	3	5.809e-9	311
	4	1.079e-9	1341
	5	9.98e-12	129

Table C.18 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-a with Lapin for the Wine Data using Gaussian Kernel

FADDIS-m (Gaussian Kernel with Lapin)

· Number of clusters: 2

· Stop Condition: 'SC_3' - Residual is too small

· Confusion Matrix in table C.19 with Mismatch error: 0.6

· Unclustered data: 0%

· Adjusted Rand Index: 0.0004

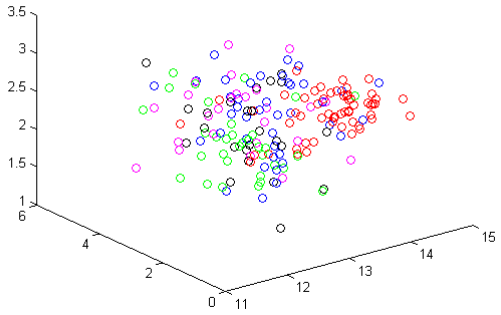
· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.20

		Original Classes		
		1	2	3
Predicted Classes	1	1	0	0
	2	58	71	48

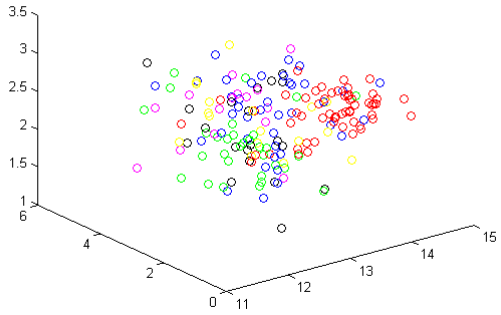
Table C.19 Confusion Matrix Obtained from original classification and clustering results of Wine Data with FADDIS-m with Lapin using the Gaussian Kernel

		Contribution	Intensity
Extracted	1	0.99875	4.08e+7
Clusters	2	3.890e-7	25467

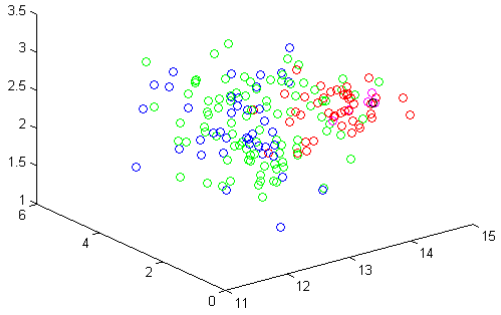
Table C.20 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-m with Lapin for the Wine Data using Gaussian Kernel



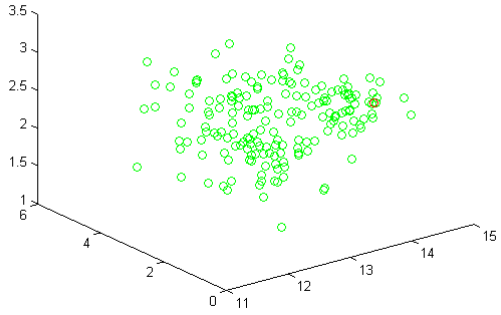
(a) FADDIS-a



(b) FADDIS-m



(c) FADDIS-a with Lapin



(d) FADDIS-m with Lapin

Figure C.4 Wine Data Clustering Results for FADDIS with Gaussian Kernel

C.3 Fat Oil Data

This dataset only has 8 entities and FRC used with 2 clusters. The data is given as a similarity matrix, where the diagonal is 0. To compute this with FADDIS, the diagonal values were changed to the matrix maximum value with the method $S_2 = S_1 + diag(\max S_1)$, where S_1 is the given similarity matrix and S_2 the resulting matrix that will be used with the algorithm. This step was made since the diagonal of a similarity matrix is supposed to have the maximum similarity value for the used algorithms. Also, a dissimilarity matrix is obtained with $D = \max S_2 - S_2$. D will be used for the FastMap and NERFCM, since both need a dissimilarity matrix as input.

In the second part, the Gaussian Kernel is applied to the D in order to continue testing with FADDIS. The clusters formed with the FRC algorithm are:

- **Cluster 1**: Linseed Oil, Perilla Oil, Cotton-Seed Oil, Sesame Oil, Camelia Oil, Olive Oil
- **Cluster 2**: Beef-tallow, Lard

The results obtained from using FMFCM and NERFCM with $K = 2$ and $K = 3$ are presented, respectively, in tables C.21 and C.22.

K = 2

Confusion Matrix:

		Original Classes	
		1	2
Predicted Classes	1	6	0
	2	0	2

Adjusted Rand Index: 1
Mismatch error: 0

K=3

Confusion Matrix:

		Original Classes	
		1	2
Predicted Classes	1	4	0
	2	0	2
	3	2	0

Adjusted Rand Index: 0.46
Mismatch error: 0.25

Table C.21 FMFCM results for Fat Oil Data

K = 2

Confusion Matrix:

		Original Classes	
		1	2
Predicted Classes	1	6	0
	2	0	2

Adjusted Rand Index: 1
Mismatch error: 0

K=3

Confusion Matrix:

		Original Classes	
		1	2
Predicted Classes	1	5	0
	2	0	2
	3	1	0

Adjusted Rand Index: 0.65
Mismatch error: 0.13

Table C.22 NERFCM results for Cancer Data

FADDIS Stop criteria tuning:

- FADDIS-a: $\varepsilon = 0.001$
- FADDIS-m: $\varepsilon = 0.005$

FADDIS-a)

· Number of clusters: 2
· Stop Condition: 'SC_3' - Residual is too small
· Confusion Matrix in table C.23 with Mismatch error: 0
· Unclustered data: 0%
· Adjusted Rand Index: 1
· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.24
· Retrieved Clusters:
– **Cluster 1**: Linseed Oil, Perilla Oil, Cotton-Seed Oil, Sesame Oil, Camelia Oil, Olive Oil
– **Cluster 2**: Beef-tallow, Lard

		Original Classes	
		1	2
Predicted Classes	1	6	0
	2	0	2

Table C.23 Confusion Matrix Obtained from FRC classification and clustering results of Fat Oil Data with FADDIS-a

		Contribution	Intensity
Extracted Clusters	1	0.93515	39.534
	2	0.02601	6.593

Table C.24 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-a for the Fat Oil Data

FADDIS-m

· Number of clusters: 2

· Stop Condition: 'SC_3' - Residual is too small

· Confusion Matrix in table C.25 with Mismatch error: 0

· Unclustered data: 0%

· Adjusted Rand Index: 1

· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.26

· Retrieved Clusters:

– **Cluster 1**: Linseed Oil, Perilla Oil, Cotton-Seed Oil, Sesame Oil, Camelia Oil, Olive Oil

– **Cluster 2**: Beef-tallow, Lard

		Original Classes	
		1	2
Predicted Classes	1	6	0
	2	0	2

Table C.25 Confusion Matrix Obtained from FRC classification and clustering results of Fat Oil Data with FADDIS-m

		Contribution	Intensity
Extracted	1	0.93515	39.534
Clusters	2	0.02601	6.593

Table C.26 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-m for the Fat Oil Data

C.3.0.5 Gaussian Kernel

FADDIS Stop Criteria Tuning:

- FADDIS-a: $\varepsilon = 0.001$
- FADDIS-m: $\varepsilon = 0.005$

FADDIS-a (Gaussian Kernel without Lapin)

· Number of clusters: 5
· Stop Condition: 'SC_3' - Residual is too small
· Confusion Matrix in table C.27 with Mismatch error: 0
· Unclustered data: 0%
· Adjusted Rand Index: 0.22
· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.28
· Retrieved Clusters:
– **Cluster 1:** Cotton-Seed Oil, Sesame Oil, Olive Oil
– **Cluster 2:** Beef-tallow, Lard
– **Cluster 3:** Linseed Oil
– **Cluster 4:** Perilla Oil
– **Cluster 5:** Camelia Oil

		Original Classes	
		1	2
Predicted Classes	1	3	0
	2	0	2
	3	1	0
	4	1	0
	5	1	0

Table C.27 Confusion Matrix Obtained from FRC classification and clustering results of Fat Oil Data with FADDIS-a with the Gaussian Kernel

		Contribution	Intensity
Extracted Clusters	1	0.55005	2.836
	2	0.24837	1.906
	3	0.06836	0.999
	4	0.06243	0.955
	5	0.03414	0.707

Table C.28 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-a for the Fat Oil Data with Gaussian Kernel

FADDIS-m (Gaussian Kernel without Lapin)

· Number of clusters: 5

· Stop Condition: 'SC_3' - Residual is too small

· Confusion Matrix in table C.29 with Mismatch error: 0

· Unclustered data: 0%

· Adjusted Rand Index: 0.22

· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.30

· Retrieved Clusters:

– **Cluster 1:** Cotton-Seed Oil, Sesame Oil, Olive Oil

– **Cluster 2:** Beef-tallow, Lard

– **Cluster 3:** Linseed Oil

– **Cluster 4:** Perilla Oil

– **Cluster 5:** Camelia Oil

		Original Classes	
		1	2
Predicted Classes	1	3	0
	2	0	2
	3	1	0
	4	1	0
	5	1	0

Table C.29 Confusion Matrix Obtained from FRC classification and clustering results of Fat Oil Data with FADDIS-m with the Gaussian Kernel

		Contribution	Intensity
Extracted Clusters	1	0.55005	2.836
	2	0.24837	1.906
	3	0.06252	0.956
	4	0.06789	0.996
	5	0.03407	0.706

Table C.30 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-m for the Fat Oil Data with Gaussian Kernel

FADDIS-a (Gaussian Kernel with Lapin)

· Number of clusters: 4

· Stop Condition: 'SC_2' - Cluster contribution is too small

· Confusion Matrix in table C.31 with Mismatch error: 0

· Unclustered data: 0%

· Adjusted Rand Index: 0.4

· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.32

· Retrieved Clusters:

– **Cluster 1:** Perilla Oil, Cotton-Seed Oil, Camelia Oil, Olive Oil

– **Cluster 2:** Beef-tallow, Lard

– **Cluster 3:** Linseed Oil

– **Cluster 4:** Sesame Oil

		Original Classes	
		1	2
Predicted Classes	1	4	0
	2	0	2
	3	1	0
	4	1	0

Table C.31 Confusion Matrix Obtained from FRC classification and clustering results of Fat Oil Data with FADDIS-a with Lapin and the Gaussian Kernel

		Contribution	Intensity
Extracted Clusters	1	0.60467	4944
	2	0.04872	1403
	3	0.00283	338
	4	6.02e-6	16

Table C.32 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-a with Lapin for the Fat Oil Data with Gaussian Kernel

FADDIS-m (Gaussian Kernel with Lapin)

· Number of clusters: 2

· Stop Condition: 'SC_1' - No positive weights at spectral clusters

· Confusion Matrix in table C.33 with Mismatch error: 0

· Unclustered data: 0%

· Adjusted Rand Index: 1

· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.34

· Retrieved Clusters:

– **Cluster 1:** Linseed Oil, Perilla Oil, Cotton-Seed Oil, Sesame Oil, Camelia Oil, Olive Oil

– **Cluster 2:** Beef-tallow, Lard

		Original Classes	
		1	2
Predicted Classes	1	6	0
	2	0	2

Table C.33 Confusion Matrix Obtained from FRC classification and clustering results of Fat Oil Data with FADDIS-m with Lapin and the Gaussian Kernel

		Contribution	Intensity
Extracted Clusters	1	0.60467	4944
	2	0.04972	1403

Table C.34 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-m with Lapin for the Fat Oil Data with Gaussian Kernel

C.4 Country data

This dataset has 12 entities and FRC was used with 3 clusters. The data is given as a dissimilarity matrix, where the diagonal is 0. To compute this with FADDIS, a similarity matrix was obtained with $S = max(D) - D$, where D is the given dissimilarity matrix and S the resulting similarity matrix that will be used with FADDIS in a first stage. The Gaussian Kernel is applied to D. As for the FastMap and NERFCM algorithms, D is used.

The clusters formed with the FRC algorithm are:

- **Cluster 1**: China, Cuba, USSR, Yugoslavia
- **Cluster 2**: Belgium, France, Israel, USA
- **Cluster 3**: Brazil, Egypt, India, Zaire

The results obtained from using FMFCM and NERFCM with $K = 2$, $K = 3$ and $K = 4$ are presented, respectively, in tables C.35 and C.36.

K = 2

Confusion Matrix:

		Original Classes		
		1	2	3
Predicted Classes	1	4	0	1
	2	0	4	3

Adjusted Rand Index: 0.41
Mismatch error: 0.33

K=3

Confusion Matrix:

		Original Classes		
		1	2	3
Predicted Classes	1	4	0	0
	2	0	4	1
	3	0	0	3

Adjusted Rand Index: 0.74
Mismatch error: 0.08

K=4

Confusion Matrix:

		Original Classes		
		1	2	3
Predicted Classes	1	4	0	0
	2	0	4	0
	3	0	0	2
	4	0	0	2

Adjusted Rand Index: 0.84
Mismatch error: 0.17

Table C.35 FMFCM results for Country Data

K = 2	K=3	K=4
Confusion Matrix:	Confusion Matrix:	Confusion Matrix:
Adjusted Rand Index: 0.41	Adjusted Rand Index: 1	Adjusted Rand Index: 0.88
Mismatch error: 0.33	Mismatch error: 0	Mismatch error: 0.08

		Original Classes		
		1	2	3
Predicted Classes	1	4	0	1
	2	0	4	3

		Original Classes		
		1	2	3
Predicted Classes	1	4	0	0
	2	0	4	0
	3	0	0	4

		Original Classes		
		1	2	3
Predicted Classes	1	4	0	0
	2	0	4	0
	3	0	0	3
	4	0	0	1

Table C.36 NERFCM results for Country Data

FADDIS Stop criteria tuning:

- FADDIS-a: $\varepsilon = 0.001$
- FADDIS-m: $\varepsilon = 0.005$

FADDIS-a

· Number of clusters: 5

· Stop Condition: 'SC_1' - Maximum number of clusters reached

· Confusion Matrix in table C.37 with Mismatch error: 0

· Unclustered data: 0%

· Adjusted Rand Index: 0.88

· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.38

· Retrieved Clusters:

– **Cluster 1**: China, Cuba, USSR, Yugoslavia

– **Cluster 2**: Belgium, France, Israel, USA

– **Cluster 3**: Brazil, India, Zaire

– **Cluster 4**: Egypt

		Original Classes		
		1	2	3
Predicted Classes	1	4	0	0
	2	0	4	0
	3	0	0	3
	4	0	0	1
	5	0	0	0

Table C.37 Confusion Matrix Obtained from FRC classification and clustering results of Country Data with FADDIS-a

		Contribution	Intensity
Extracted Clusters	1	0.73384	38.281
	2	0.06870	11.713
	3	0.02518	7.0915
	4	0.03158	7.9414
	5	0.00182	1.9065

Table C.38 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-a for the Country Data

FADDIS-a

· Number of clusters: 5

· Stop Condition: 'SC_1' - Maximum number of clusters reached

· Confusion Matrix in table C.39 with Mismatch error: 0

· Unclustered data: 0%

· Adjusted Rand Index: 0.88

· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.40

· Retrieved Clusters:

– **Cluster 1**: China, Cuba, USSR, Yugoslavia

– **Cluster 2**: Belgium, France, Israel, USA

– **Cluster 3**: Brazil, India, Zaire

– **Cluster 4**: Egypt

		Original Classes		
		1	2	3
Predicted Classes	1	4	0	0
	2	0	4	0
	3	0	0	3
	4	0	0	1
	5	0	0	0

Table C.39 Confusion Matrix Obtained from FRC classification and clustering results of Country Data with FADDIS-m

		Contribution	Intensity
Extracted Clusters	1	0.73384	38.281
	2	0.06870	11.713
	3	0.02518	7.0915
	4	0.03158	7.9414
	5	0.00182	1.9065

Table C.40 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-m for the Country Data

C.4.0.6 Gaussian Kernel

FADDIS Stop Criteria Tuning:

- FADDIS-a: $\varepsilon = 0.001$
- FADDIS-m: $\varepsilon = 0.005$

FADDIS-a (Gaussian Kernel without Lapin)

· Number of clusters: 5

· Stop Condition: 'SC_4' - Maximum number of clusters reached

· Confusion Matrix in table C.41 with Mismatch error: 0.33

· Unclustered data: 0%

· Adjusted Rand Index: 0.645

· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.42

· Retrieved Clusters:

– **Cluster 1**: Cuba, USSR

– **Cluster 2**: Belgium, France, Israel, USA

– **Cluster 3**: Brazil, Zaire

– **Cluster 4**: Egypt, India

– **Cluster 5**: China, Yugoslavia

		Original Classes		
		1	2	3
Predicted Classes	1	2	0	0
	2	0	4	0
	3	0	0	2
	4	0	0	2
	5	2	0	0

Table C.41 Confusion Matrix Obtained from FRC classification and clustering results of Country Data with FADDIS-a with Gaussian Kernel

		Contribution	Intensity
Extracted Clusters	1	0.08455	1.0073
	2	0.08339	1.0004
	3	0.08334	1
	4	0.08333	1
	5	0.08333	1

Table C.42 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-a for the Country Data with Gaussian Kernel

FADDIS-m (Gaussian Kernel without Lapin)

· Number of clusters: 6

· Stop Condition: 'SC_4' - Maximum number of clusters reached

· Confusion Matrix in table C.43 with Mismatch error: 0.33

· Unclustered data: 0%

· Adjusted Rand Index: 0.48

· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.44

· Retrieved Clusters:

– **Cluster 1**: Cuba, USSR

– **Cluster 2**: Belgium, France, USA

– **Cluster 3**: Brazil, Zaire

– **Cluster 4**: Israel

– **Cluster 5**: Egypt, India

– **Cluster 6**: China, Yugoslavia

		Original Classes		
		1	2	3
Predicted Classes	1	2	0	0
	2	0	3	0
	3	0	0	2
	4	0	1	0
	5	0	0	2
	6	2	0	0

Table C.43 Confusion Matrix Obtained from FRC classification and clustering results of Country Data with FADDIS-m with Gaussian Kernel

		Contribution	Intensity
Extracted Clusters	1	0.08455	1.0073
	2	0.08339	1.0004
	3	0.08334	1
	4	0.08210	0.9926
	5	0.08333	1
	6	0.08333	1

Table C.44 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-m for the Country Data with Gaussian Kernel

FADDIS-a (Gaussian Kernel with Lapin)

· Number of clusters: 4

· Stop Condition: 'SC_2' - Cluster contribution is too small

· Confusion Matrix in table C.45 with Mismatch error: 0.58

· Unclustered data: 0%

· Adjusted Rand Index: 0.24

· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.46

· Retrieved Clusters:

– **Cluster 1**: Yugoslavia

– **Cluster 2**: Belgium, China, Egypt, France, Israel, USA

– **Cluster 3**: Brazil, India, USSR, Zaire

– **Cluster 4**: Cuba

		Original Classes		
		1	2	3
Predicted Classes	1	1	0	0
	2	1	4	1
	3	1	0	3
	4	1	0	0

Table C.45 Confusion Matrix Obtained from FRC classification and clustering results of Country Data with FADDIS-a with Lapin and Gaussian Kernel

		Contribution	Intensity
Extracted Clusters	1	0.54584	5.8264e+6
	2	0.06358	1.9884e+6
	3	0.00582	6.0181e+5
	4	0.00086	2.3147e+5

Table C.46 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-a with Lapin for the Country Data with Gaussian Kernel

FADDIS-m (Gaussian Kernel with Lapin)

· Number of clusters: 2

· Stop Condition: 'SC_1' - No positive weights at spectral clusters

· Confusion Matrix in table C.47 with Mismatch error: 0.42

· Unclustered data: 0%

· Adjusted Rand Index: 0.24

· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.48

· Retrieved Clusters:

– **Cluster 1**: Brazil, Cuba, India, USSR, Yugoslavia, Zaire

– **Cluster 2**: Belgium, China, Egypt, France, Israel, USA

		Original Classes		
		1	2	3
Predicted Classes	1	3	0	3
	2	1	4	1

Table C.47 Confusion Matrix Obtained from FRC classification and clustering results of Country Data with FADDIS-m with Lapin and Gaussian Kernel

		Contribution	Intensity
Extracted Clusters	1	0.54584	5.8264e+6
	2	0.06358	1.9884e+6

Table C.48 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-m with Lapin for the Country Data with Gaussian Kernel

C.5 Microcomputer Data

This dataset has 12 entities and FRC was used with 4 clusters. The data is given as a similarity matrix, where the diagonal is 0. To compute this with FADDIS, a new similarity matrix was obtained with $S_2 = S_1 + diag(\max S_1)$, where S_1 is the given similarity matrix and S_2 the resulting similarity matrix that will be used. This step was made since the diagonal of a similarity matrix is supposed to have the maximum similarity value for the used algorithms. The second section will obtain a dissimilarity matrix with $D = \max S_2 - S_2$, and then apply the Gaussian Kernel in case of FADDIS. D will also be used with FastMap and NERFCM.

The clusters formed with the FRC algorithm are:

- **Cluster 1**: HP-85
- **Cluster 2**: Zenith H8, Zenith H89, Horizon, TRS-80 III
- **Cluster 3**: Com. VIC 20, O.S. Challenger, O.S.II Series
- **Cluster 4**: Apple II, Atari 800, Ex. Sorcerer, TRS-80 I

The results obtained from using FMFCM and NERFCM with $K = 3$, $K = 4$ and $K = 5$ are presented, respectively, in tables C.49 and C.50.

FADDIS Stop criteria tuning:

- FADDIS-a: ε = *0.005*
- FADDIS-m: ε = *0.005*

K = 3

Confusion Matrix:

		Original Classes			
		1	2	3	4
Predicted Classes	1	1	0	0	0
	2	0	4	0	0
	3	0	0	3	4

Adjusted Rand Index: 0.6
Mismatch error: 0.33

K = 4

Confusion Matrix:

		Original Classes			
		1	2	3	4
Predicted Classes	1	1	0	0	0
	2	0	4	0	0
	3	0	0	2	0
	4	0	0	1	4

Adjusted Rand Index: 0.75
Mismatch error: 0.08

K = 5

Confusion Matrix:

		Original Classes			
		1	2	3	4
Predicted Classes	1	1	1	0	0
	2	0	3	0	0
	3	0	0	2	1
	4	0	0	0	3
	5	0	0	1	0

Adjusted Rand Index: 0.46
Mismatch error: 0.17

Table C.49 FMFCM results for Microcomputer Data

K = 3

Confusion Matrix:

		Original Classes			
		1	2	3	4
Predicted Classes	1	1	0	2	0
	2	0	4	0	0
	3	0	0	1	4

Adjusted Rand Index: 0.68
Mismatch error: 0.5

K = 4

Confusion Matrix:

		Original Classes			
		1	2	3	4
Predicted Classes	1	0	0	1	2
	2	0	3	0	0
	3	1	0	2	0
	4	0	1	0	2

Adjusted Rand Index: 0.3
Mismatch error: 0.42

K = 5

Confusion Matrix:

		Original Classes			
		1	2	3	4
Predicted Classes	1	1	0	1	0
	2	0	3	0	0
	3	0	0	1	0
	4	0	0	1	3
	5	0	1	0	1

Adjusted Rand Index: 0.33
Mismatch error: 0.33

Table C.50 NERFCM results for Microcomputer Data

FADDIS-a

· Number of clusters: 4

· Stop Condition: 'SC_2' - Cluster contribution is too small

· Confusion Matrix in table C.51 with Mismatch error: 0.25

· Unclustered data: 0%

· Adjusted Rand Index: 0.28

· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.52

· Retrieved Clusters:

– **Cluster 1**: HP-85

– **Cluster 2**: Zenith H8, Zenith H89, Horizon

– **Cluster 3**: Apple II, O.S. Challenger, O.S.II Series

– **Cluster 4**: Atari 800, Com. VIC 20, Ex. Sorcerer, TRS-80 I, TRS-80 III

		Original Classes			
		1	2	3	4
Predicted Classes	1	1	0	0	0
	2	0	3	0	0
	3	0	0	2	1
	4	0	1	1	3

Table C.51 Confusion Matrix Obtained from FRC classification and clustering results of Microcomputer Data with FADDIS-a

		Contribution	Intensity
Extracted Clusters	1	0.95821	48.917
	2	0.01122	5.292
	3	0.00736	4.288
	4	0.00299	2.735

Table C.52 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-a for the Microcomputer Data

FADDIS-m

· Number of clusters: 4

· Stop Condition: 'SC_2' - Cluster contribution is too small

· Confusion Matrix in table C.53 with Mismatch error: 0.33

· Unclustered data: 0%

· Adjusted Rand Index: 0.28

· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.54

· Retrieved Clusters:

– **Cluster 1**: Com. VIC 20, HP-85

– **Cluster 2**: Zenith H8, Zenith H89, Horizon

– **Cluster 3**: Apple II, O.S. Challenger, O.S.II Series, TRS-80 I

– **Cluster 4**: Atari 800, Ex. Sorcerer, TRS-80 III

C.5.0.7 Gaussian Kernel

FADDIS Stop Criteria Tuning:

		Original Classes			
		1	2	3	4
Predicted Classes	1	1	0	1	0
	2	0	3	0	0
	3	0	0	2	2
	4	0	1	0	2

Table C.53 Confusion Matrix Obtained from FRC classification and clustering results of Microcomputer Data with FADDIS-m

		Contribution	Intensity
Extracted Clusters	1	0.95821	48.917
	2	0.00823	4.534
	3	0.01224	5.528
	4	0.00308	2.775

Table C.54 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-m for the Microcomputer Data

- FADDIS-a: $\varepsilon = 0.001$
- FADDIS-m: $\varepsilon = 0.001$

FADDIS-a (Gaussian Kernel without Lapin)

· Number of clusters: 5

· Stop Condition: 'SC_4' - Maximum number of clusters reached

· Confusion Matrix in table C.55 with Mismatch error: 0.17

· Unclustered data: 0%

· Adjusted Rand Index: 0.658

· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.56

· Retrieved Clusters:

– **Cluster 1:** HP-85

– **Cluster 2:** Zenith H8, Zenith H89, TRS-80 III

– **Cluster 3:** Com. VIC 20, O.S. Challenger, O.S.II Series

– **Cluster 4:** Apple II, Atari 800, TRS-80 I

– **Cluster 5:** Ex. Sorcerer, Horizon

		Original Classes			
		1	2	3	4
Predicted Classes	1	1	0	0	0
	2	0	3	0	0
	3	0	0	3	0
	4	0	0	0	3
	5	0	1	0	1

Table C.55 Confusion Matrix Obtained from FRC classification and clustering results of Microcomputer Data with FADDIS-a with Gaussian Kernel

		Contribution	Intensity
Extracted Clusters	1	0.41173	2.9162
	2	0.24667	2.2572
	3	0.05373	1.0535
	4	0.04842	1.0000
	5	0.04089	0.9190

Table C.56 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-a for the Microcomputer Data with Gaussian Kernel

FADDIS-m (Gaussian Kernel without Lapin)

· Number of clusters: 6

· Stop Condition: 'SC_4' - Maximum number of clusters reached

· Confusion Matrix in table C.57 with Mismatch error: 0.33

· Unclustered data: 0%

· Adjusted Rand Index: 0.362

· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.58

· Retrieved Clusters:

– **Cluster 1**: HP-85

– **Cluster 2**: Zenith H8, Zenith H89, TRS-80 III

– **Cluster 3**: O.S. Challenger, O.S.II Series

– **Cluster 4**: Apple II, TRS-80 I

– **Cluster 5**: Ex. Sorcerer, Horizon

– **Cluster 6**: Atari 800, Com. VIC 20

		Original Classes			
		1	2	3	4
Predicted Classes	1	1	0	0	0
	2	0	3	0	0
	3	0	0	2	0
	4	0	0	0	2
	5	0	1	0	1
	6	0	0	1	1

Table C.57 Confusion Matrix Obtained from FRC classification and clustering results of Microcomputer Data with FADDIS-m with Gaussian Kernel

		Contribution	Intensity
Extracted Clusters	1	0.41173	2.9162
	2	0.24667	2.2572
	3	0.05373	1.0535
	4	0.04089	0.9191
	5	0.03877	0.8948
	6	0.04842	1.0000

Table C.58 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-m for the Microcomputer Data with Gaussian Kernel

FADDIS-a (Gaussian Kernel with Lapin)

· Number of clusters: 5

· Stop Condition: 'SC_4' - Cluster contribution is too small

· Confusion Matrix in table C.59 with Mismatch error: 0.33

· Unclustered data: 0%

· Adjusted Rand Index: 0.49

· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.60

· Retrieved Clusters:

– **Cluster 1:** Atari 800, HP-85, TRS-80 I

– **Cluster 2:** Zenith H8, Zenith H89, Horizon, TRS-80 III

– **Cluster 3:** Com. VIC 20

– **Cluster 4:** Apple II, Ex. Sorcerer, O.S.II Series

– **Cluster 5:** O.S. Challenger

		Original Classes			
		1	2	3	4
Predicted Classes	1	1	0	0	2
	2	0	4	0	0
	3	0	0	1	0
	4	0	0	1	2
	5	0	0	1	0

Table C.59 Confusion Matrix Obtained from FRC classification and clustering results of Microcomputer Data with FADDIS-a with Lapin and Gaussian Kernel

		Contribution	Intensity
Extracted Clusters	1	0.41173	2.9162
	2	0.24667	2.2572
	3	0.05373	1.0535
	4	0.04842	1.0000
	5	0.04089	0.9190

Table C.60 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-a with Lapin for the Microcomputer Data with Gaussian Kernel

FADDIS-m (Gaussian Kernel with Lapin)

· Number of clusters: 2

· Stop Condition: 'SC_1' - No positive weights at spectral clusters

· Confusion Matrix in table C.61 with Mismatch error: 0.42

· Unclustered data: 0%

· Adjusted Rand Index: 0.5

· Contribution to the data scatter and Intensity of the extracted clusters are presented in Table C.62

· Retrieved Clusters:

−**Cluster 1:** Zenith H8, Zenith H89, HP-85, Horizon, TRS-80 III

−**Cluster 2:** Apple II, Atari 800, Com. VIC 20, Ex. Sorcerer, O.S. Challenger, O.S.II Series, TRS-80 I

		Original Classes			
		1	2	3	4
Predicted Classes	1	1	4	0	0
	2	0	0	3	4

Table C.61 Confusion Matrix Obtained from FRC classification and clustering results of Microcomputer Data with FADDIS-m with Lapin and Gaussian Kernel

		Contribution	Intensity
Extracted Clusters	1	0.39412	10.904
	2	0.09219	5.274

Table C.62 Values for the contribution of the data scatter and intensity of the extracted clusters for FADDIS-m with Lapin for the Microcomputer Data with Gaussian Kernel

Bibliography

[1] Altschul, S., Gish, W., Miller, W., Myers, E., Lipman, D., *Basic Local Alignment Search Tool*, Journal of Molecular Biology, Vol 215, p.403-410, 1990

[2] Bezdek, J., *Cluster validity with fuzzy sets*, Journal of Cybernetics, Vol 3, p.58-73, 1974

[3] Bezdek, J., *Pattern Recognition with Fuzzy Objective Function Algorithms*, Plenum Press, 1981

[4] Bezdek, J., Ehrlich, R., Full, W., *FCM: The fuzzy c-means clustering algorithm*, Computers and Geosciences, Vol 10, Issues 2-3, p.191-203, 1984

[5] Bezdek, J., Pal, M., Keller, J., Krisnapuram, R., *Fuzzy Models and Algorithms for Pattern Recognition and Image Processing*, Springer, 1999

[6] Bezdek, J., Havens, T., Keller, J., Popescu, M., Rehrig, E., Appel, H., Schultz, J., *Fuzzy cluster analysis of bioinformatics data composed of microarray expression data and gene ontology annotations* , Proc. North American Fuzzy Information Processing Society, p.1-6, 2008.

[7] Bezdek, J., Hathaway, R., *VAT: A tool for visual assessment of (Cluster) Tendency*, Joint Conference on Neural Networks, IEEE Press, p.2225-2230, 2002.

[8] Bezdek, J., Sledge, I., Havens, T. and Keller, J., *Relational Generalizations of Cluster Validity Indices*, Journal IEEE Transactions on Fuzzy Systems, Vol 18, Issue 4, p.771-786, 2010.

[9] Brouwer, R., *A method of relational fuzzy clustering based on producing feature vectors using FastMap*, Journal Information Sciences: an International Journal, Vol 179, Issue 20, p.3561-3582, 2009.

[10] Carter, T. A., *Eigenvalues and Eigenvectors*, Linear Algebra - An Introduction to Linear Algebra for Pre-Calculus Students, p.114-131, 1995.

[11] Corsini, P., Lazzerini, B., Marcelloni, F., *A new fuzzy relational clustering algorithm based on the fuzzy C-means algorithm*, Soft Computing - A Fusion of Foundations, Methodologies and Applications, Vol 9, Issue 6, p.439-447, 2005.

[12] Davé, R., Sen, S., *Robust Fuzzy Clustering of Relational Data*, Proc. of the IEEE Conference on Fuzzy Systems, Vol 10, Issue 6, p.713-727, 2002.

[13] Ding, C., He, X., *K-means Clustering via Principal Component Analysis*, Proc. International Conference on Machine Learning 2004, p.225-232, 2004.

[14] Du, W., Urahama, K., *Unsupervised and Semi-Supervised Graph-Spectral Algorithms for Robust Extraction of Arbitrarily Shaped Fuzzy Clusters*, IEICE Transactions on Information Systems, E89-D, p.2315-2318, 2006.

[15] Faloutsos, C., Lin, K., *FastMap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets*, ACM SIGMOD'95 Conference on Management of Data, p.163-174, 1995.

[16] Halkidi, M., Batistakis, Y., Vazirgiannis, M., *On Clustering Validation Techniques*, Journal of Intelligent Information Systems 17, Issue 2-3, p.107-145, 2001.

[17] Hathaway, R., Huband, J., Bezdek, J., *A Kernelized Non-Euclidean Relational Fuzzy c-Means Algorithm*, Fuzzy Systems 2005 - The 14th IEEE International Conference, p.414-419, 2005.

[18] Hathaway, R., Davenport, J., Bezdek, J., *On relational data versions of c-means algorithms*, Pattern Recognition Letters, Vol 17, Issue 6, p.607-612, 1996 .

[19] Hathaway, R., Bezdek, J., Sabin, M., Tucker, W., *Convergence theory for fuzzy C-means: counterexamples and repairs*, IEEE Transactions on Systems, Man and Cybernetics, Vol 17, Issue 5, p.873-877, 1987.

[20] Hathaway, R., Davenport, J., Bezdek, J., *Relational dual of the c-means clustering algorithms*, Pattern Recognition, Vol 22, Issue 2, p.205-212, 1989.

[21] Hathaway, R., Bezdek, J., *NERF c-means: Non-Euclidean relational fuzzy clustering*, Pattern Recognition, Vol 27, Issue 3, p.429-437, 1994.

[22] Inoue, K., Urahama, K., *Sequential fuzzy cluster extraction by a graph spectral method*, Pattern Recognition, Vol 20, Issue 7, p.699-705, 1999.

[23] Jordan, M., Ng, A., Weiss, Y., *On spectral clustering: analysis and an algorithm*, Advances in Neural Information Processing Systems, Vol 14, p.849-856, 2002.

[24] Jordan, M. *Lecture 5: Properties of Kernels and the Gaussian Kernel*, Advanced Topics in Learning & Decision Making, 2004.

[25] Klein, D. J., Randic, M., *Resistance Distance*, Journal of Mathematical Chemistry, Vol 12, p.81-95, 1993

[26] Luxburg, U., *A tutorial on spectral clustering*, Statistics and Computing, Vol 17, p.395-416, 2007.

[27] Mirkin, B., Nascimento, S., *Analysis of Community Structure, Affinity Data and Research Activities using Additive Fuzzy Spectral Clustering*, Technical Report 6, School of Computer Science, Birkbeck University of London, 2009.

[28] Mirkin, B., Nascimento, S., *Additive Spectral Method for Fuzzy Cluster Analysis of Similarity Data Including Community Structure and Affinity Matrices*, Information Sciences, Elsevier, ISSN 0020-0255, to appear, 2011.

[29] Mirkin, B., Nascimento, S., Fenner, T., Pereira, L., *Building Fuzzy Thematic Clusters and Mapping Them to Higher Ranks in a Taxonomy*, International Journal of Software and Informatics, Vol. 4 (special issue of KSEM2010), No. 3, p.257-275, 2010.

[30] Mirkin, B., Nascimento, S., Fenner, T., Pereira, L.M., *Constructing and Mapping Fuzzy Thematic Clusters to Higher Ranks in a Taxonomy*, Proc. 4th International Conference on Knowledge Science, Engineering & Management (KSEM 2010), p.329-340, 2010.

[31] Mitchell, T., *Machine Learning*, McGraw-Hill, 1997.

[32] Nadler, B., Galun, M., *Fundamental Limitations of Spectral Clustering*, Advanced in Neural Information Processing Systems, Vol 19, p.1017-1024, 2006.

[33] Nascimento, S., *Fuzzy Clustering Via Proportional Membership Model*, Frontiers in Artificial Intelligence and Applications, Vol 119, 2005.

[34] Nascimento, S., *Introdução à Aprendizagem não Supervisionada*, Aprendizagem Automática e Data Mining lectures, FCT-UNL, MEI 2007/2008.

[35] Nascimento, S., Franco, P., *Segmentation of Upwelling Regions in Sea Surface Temperature Images via Unsupervised Fuzzy Clustering*, Lecture Notes in Computer Science, Vol 5788, p.543-553, 2009.

[36] Ott, T., Kern, A., Steeb, W., Stoop, R., *Sequential clustering: tracking down the most natural clusters*, Journal of Statistical Mechanics: Theory and Experiment, Issue 11, p.11014, 2005.

[37] Perona, P., Zelnik-Manor, L., *Self-Tuning Spectral Clustering*, Advances in Neural Information Processing Systems, Vol 17, p.1601-1608, 2004.

[38] Popescu, M., Bezdek, J., Keller, J., Havens, T., Huband, J., *A new cluster validity measure for bioinformatics relational datasets*, Proc. IEEE Conference on Fuzzy Systems 2008, p.726-731, 2008.

[39] Roubens, M., *Pattern classification problems and fuzzy sets*, Fuzzy Sets and Systems, Vol 1, Issue 4, p.239-253, 1978.

[40] Shlens, J., *A Tutorial on Principal Component Analysis*, 2009.

[41] Shi, J., Malik, J., *Normalized cuts and image segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 22, Issue 8, p.888-905, 2000.

[42] Song, J., Nicolae, D., *A sequential clustering algorithm with applications to gene expression data*, Journal of the Korean Statistical Society, Vol 38, Issue 2, p.175-184, 2009.

[43] Suryavanshi, B., Shiri, N., Mudur, S., *An Efficient Technique for Mining Usage Profiles using Relational Fuzzy Subtractive Clustering*, Proc. IEEE International Workshop on Challenges in Web Information Retrieval and Integration (WIRI'05), p.23-29, 2005.

[44] Suryavanshi, B., Shiri, N., Mudur, S., *Incremental Relational Fuzzy Subtractive Clustering for Dynamic Web Usage Profiling*, Proc. WEBKDD Workshop on Taming Evolving, Expanding and Multi-faceted Web Clickstreams, p.21-24, 2005.

[45] Tan, P., Steinbach, M., Kumar, V., *Cluster Analysis: Basic Concepts and Algorithms*, Lecture Notes for Chapter 8: Introduction to Data Mining, 2006.

[46] Vapnyarskii, I., *Lagrange Multipliers*, Springer Online Reference Works, http://eom.springer.de/L/l057190.htm

[47] Wanga, W., Zhanga, Y., *On fuzzy cluster validity indices*, Fuzzy Sets and Systems, Vol 158, Issue 19, p.2095-2117, 2007.

[48] Weisstein, E., *Laplacian Matrix*, MathWorld - A Wolfram Web Resource, http://mathworld.wolfram.com/LaplacianMatrix.html

[49] Xie, X., Beni, G., *A cluster validity index for fuzzy clustering*, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 13, p.841-847, 1991.

[50] Yeung, K., Ruzzo, W., *Details of the Adjusted Rand index and Clustering algorithms Supplement to the paper "An empirical study on Principal Component Analysis for clustering gene expression data"*, Bioinformatics, Vol 17, p.763-774, 2001.

[51] Yeung, K., Ruzzo, W., *An empirical study of principal component analysis for clustering gene expression data*, Technical Report UW-CSE-00-11-03, 2000.

[52] *The 1998 ACM Computing Classification System*, http://www.acm.org/about/class/1998

[53] *Gene Ontology*, http://www.geneontology.org

[54] *Basic Alignment Local Search Tool*, http://blast.ncbi.nlm.nih.gov/

[55] *UCI Machine Learning Repository*, http://archive.ics.uci.edu/ml/

[56] Windham, M., *Numerical classification of proximity data with assignment measures*, Journal of Classification, Vol 2, p.157-172, 1985.