Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Национальный исследовательский университет «МЭИ»

Институт: *ИРЭ* Кафедра: *Радиотехнических систем*

Специальность: *11.04.01 – Радиотехника*

ОТЧЕТ по практике

Наименование практики: Производственная практика: научно-исследовательская работа

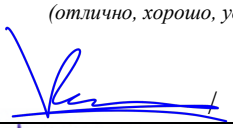
СТУДЕНТ

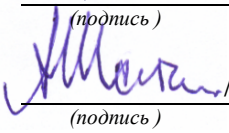
 / Масалкова Н.В. /
(подпись) (*Фамилия и инициалы*)

Группа ЭР-12м-20
(номер учебной группы)

ПРОМЕЖУТОЧНАЯ АТТЕСТАЦИЯ ПО ПРАКТИКЕ

Отлично
(отлично, хорошо, удовлетворительно, неудовлетворительно, зачтено, не зачтено)

 / Корогодин И.В. /
(подпись) (*Фамилия и инициалы члена комиссии*)

 / Шатилов А.Ю. /
(подпись) (*Фамилия и инициалы члена комиссии*)

Москва
2021

Оглавление

Введение

В современном мире от устройств радиолокации требуется, чтобы они работали в условиях, для которых характерны высокая скорость изменения внешней обстановки и большое количество объектов, подлежащих обнаружению [1]. Обработка столь большого количества информации в требуемые сжатые сроки не может быть эффективно осуществлена человеком-оператором [3]. Поэтому перед радиолокационными станциями (РЛС), являющимися основным источником информации, ставится задача автоматизации процессов обработки информации. Актуальность решения данной задачи в том, что в отличие от применявшихся ранее систем обработки, в которых конечное решение принимал человек-оператор, в этих образцах конечное решение принимается определенными алгоритмами [2].

Для разработки подобных систем потребовалось создание теории автоматической обработки радиолокационной информации (РЛИ). Существуют методы, способные осуществлять автоматическую обработку РЛИ [3-8]. Однако большинство данных методов базируются на положениях классической теории радиолокации, которая опирается на предположения, многие из которых не выполняются на практике. Необходимость использования этих предположений обусловлена трудностью формализации и математического описания всевозможных воздействий помех и других факторов, отсутствием единой методологии оценки систем обработки в различных воздушных и помеховых условиях. И оказалось, что при создании образцов РЛС с автоматической обработкой информации, классическая теория часто оказывается неприменима [8]. До сих пор нет универсальной теории для создания автоматической обработки РЛИ.

Действительно, современные радиолокационные средства в достаточно простых условиях успешно справляются со своими задачами, а в сложных условиях (нестационарных и негауссовых помех) их эффективность может резко снижаться. Снижается достоверность выдаваемой РЛС информации за счет

появления большого числа ложных отметок и появления ложных траекторий. Их число может намного превышать число целей в зоне обзора, а использование недостоверной информации ведет к снижению эффективности, например, радиолокационных стрельбовых средств в несколько раз [6].

Непредсказуемость внешней обстановки и высокая динамика её изменения создают значительные трудности для формулировки алгоритмов и для обеспечения высокого качества их работы. Появляются и новые задачи, например, распознавание, кластеризация, классификация и анализ ситуаций. И классических методов решения таких задач нет [10].

В настоящее время актуальными являются исследования по повышению качества обработки информации за счет использования, например, статистики и машинного обучения на определенных подсистемах обработки в зависимости от физической структуры входных и выходных сигналов, места их применения, характера изменения внешних условий. В этих условиях встает задача разделения подобных между собой этапов обработки информации в разнородных системах, эффективность которых могла бы быть резко повышена за счет применения алгоритмов машинного обучения. В данной работе будет рассмотрен один из этапов обработки информации с применением к нему алгоритмов машинного обучения.

1. Этапы обработки радиолокационной информации

Обработка РЛИ представляет собой наиболее важный комплекс задач РЛС. Назначение обработки подготовить к выдаче в требуемом виде полную, достоверную и современную информацию для потребителя о состоянии воздушной обстановки, появлении и местоположении воздушных средств объектов, параметрах их движения, возможных вариантах развития динамики изменения воздушно-помеховой обстановки. В ходе обработки РЛИ решается целый ряд задач, которые можно сгруппировать по близости объектов и применяемых методов в основные этапы. К основным этапам обработки радиолокационной информации можно отнести [8-9]:

- Формирование диаграммы направленности антенны для фазирования антенных решеток;
- Первичная (спектральная) обработка;
- Обнаружение радиолокационных объектов
- Вторичная (траекторная) обработка
- Распознавание объектов и ситуаций
- Анализ обстановки и адаптация

Решаемые на этих этапах задачи служат выполнению главной цели, для которой предназначен радиолокатор – выдаче полной, достоверной и своевременной информации потребителю о воздушных целях.

Большого внимания заслуживает такой этап обработки, как обнаружение отметок от целей и их кластеризация для дальнейшей вторичной (траекторной) обработки целей. Обнаружение радиолокационных целей обычно включает в себя два подэтапа: обнаружение импульсов, отраженных от целей, и их последующего пакетирования в ходе которого объединяются решения, принятые по каждому отдельному импульсу. Обнаружение целей должно проводиться автоматически, сохраняя высокие характеристики при работе в различны тактических условиях, при воздействии разнотипных помех, в том числе нестационарных, т.е. в существенно неопределенных условиях.

Рассмотрим некоторые методы, которые применяются для автоматического обнаружения воздушных целей в РЛС, а именно использование адаптивного порога обнаружения [11], использование фазовых соотношений в многобазовых сигналах (знаковые методы) и ранговые методы [4].

Данным методам присущи крупные недостатки, которые определяют практическую полную непригодность для РЛС, ведущих работу в реальных условиях. Рассмотрим подробнее: в реальных условиях при воздействии на РЛС помех нестационарного типа, а также хотя и стационарных, но негауссовых помех таких, как отражения от подвижных и местных предметов, пассивные помехи, уровень ложных тревог может возрасти на 2…3 порядка, что недопустимо. Кроме того, при наличии в пределах опорной выборки сигналов, отраженных от соседних с обнаруживаемой целью, что характерно для обнаружения целей в группах, энергетические потери могут составлять 10…15 дБ и более [11]. Исключение составляют ранговые методы, обеспечивающие практически полную стабилизацию уровня ложных тревог, но эти методы при малой пачке имеют очень высокие потери даже при обнаружении одиночных целей, а при обнаружении групп целей эти потери катастрофически возрастают.

Для синтеза эффективных методов автоматического обнаружения необходимо привлечь статическую теорию различения гипотез. Синтезируемые методы должны эффективно работать в условиях воздействия основных помех. При этом необходимо определить совокупность параметров помех и обеспечить устойчивость и совокупность параметров.

Пакетирование импульсов проводится для формирования отметок от целей, на основе полученных нескольких импульсов от цели. Эта задача аналогична задаче кластеризации: необходимо из множества одиночных отметок выбрать несколько центров группирования, которые соответствуют обнаруживаемым целям. В итоге проведение группирования отметок в кластеры позволит улучшить алгоритмы вторичной обработки сигналов, а именно сопровождение траекторий групп целей. Группирование позволит сократить количество ложных траекторий примерно на порядок.

2. Постановка задачи

Глобальная цель данной работы является возможность повышения эффективности обработки информации в системах пассивной радиолокации за счет отождествления сигналов с целями путем обработки данных сигналов алгоритмами кластеризации.

В предыдущих работах были рассмотрены существующие алгоритмы кластеризации, подробное описание каждого алгоритма, сравнение между собой.

Задача данной работы продолжить наработки по данной теме, а именно смоделировать модель, которая будет имитировать реальные измерения, чтобы определить кластеры сложных сигналов (паттернов), состоящих из нескольких отметок.

3. Кластерный анализ

Чтобы решить поставленную задачу для начала требуется дать определение понятию кластеризация. *Кластеризация или кластерный анализ* — это задача разбиения множества объектов на группы, называемые кластерами. Внутри каждой группы оказываются объекты с подобными параметрами, а объекты разных групп имеют максимальные отличия друг от друга.

Применение кластерного анализа в общем виде сводится к следующим этапам:

1. Отбор выборки объектов для кластеризации.
2. Определение множества переменных, по которым будут оцениваться объекты в выборке. Перед запуском алгоритма чаще всего требуется *нормализовать* данные или по-другому *стандартизировать* данные. Нормализация предполагает замену номинальных признаков так, чтобы каждых из них находился в диапазоне от нуля до единицы. Стандартизация же подразумевает предобработку данных, после которой каждый признак имеет среднее значение равное нулю и среднеквадратичное отклонение (СКО) равное единице. В итоге нормализация или стандартизация дает равную важность переменным во время кластеризации.
3. Вычисление значений *меры сходства* между объектами.
4. Применение метода кластерного анализа для создания групп сходных объектов (кластеров).
5. Представление результатов анализа.

После получения и анализа результатов возможна корректировка выбранной метрики и метода кластеризации до получения оптимального результата.

Кроме того, кластерный анализ является общим методом машинного обучения без учителя для поиска шаблонов в наборе данных. Обучение без учителя используется, чтобы формировать группы из наборов данных, которые обычно заранее не размечены. Например, используются кластерный анализ для

исследовательского анализа данных, чтобы найти спрятанные шаблоны или группы объектов в непомеченных входных данных.

Кластерный анализ создает группы или по-другому *кластеры*, данных. Объекты, которые принадлежат тому же кластеру, похожи друг на друга и отличны от объектов, принадлежащих другим кластерам. Чтобы определить сколько таких "подобных" и "отличных", используют меру по несходству или по-другому метрику расстояний. Кроме того, в зависимости от имеющегося набора данных требуется использовать масштабирование (или стандартизацию) переменных, чтобы дать равную оценку во время кластеризации.

Категоризация точек данных на основе их расстояния до соседних точек в этом же наборе данных является простым и эффективным способом кластеризации точек.

Решение данной задачи будет проводится в среде `Matlab`, поэтому следует привести описание функций, с которыми будет проводиться работа.

В среде `Matlab` попарное расстояние между точками набора данных находится с помощью функции `pdist2`. Синтаксис данной функции:

`D = pdist2(X,Y,Distance)` - возвращает расстояние между каждой парой наблюдений в *X* и *Y* с используемой метрикой, заданной в Distance.

`D = pdist2(X,Y,Distance,DistParameter)` - возвращает расстояние с помощью метрики, заданной `Distance` и `DistParameter`. Можно задать `DistParameter` только, когда `Distance` `'seuclidean'`, `'minkowski'`, или `'mahalanobis'`.

`D = pdist2(___,Name,Value)` - задает дополнительную опцию с помощью одного из аргументов пары "имя-значение" `'Smallest'` или `'Largest'` в дополнение к любому из аргументов в предыдущих синтаксисах.

Например,

`D = pdist2(X,Y,Distance,'Smallest',K)` - вычисляет расстояние с помощью метрики, заданной `Distance` и

возвращает K наименьшие попарные расстояния до наблюдений в X для каждого наблюдения в Y в порядке возрастания.

`D = pdist2(X,Y,Distance,'Largest',K` вычисляет расстояние с помощью метрики, заданной `Distance` и `DistParameter` и возвращает K самые большие попарные расстояния в порядке убывания.

`[D,I] = pdist2(___,Name,Value)` - также возвращает матричный I. Матричный I содержит индексы наблюдений в X соответствие расстояниям в D.

Требуется определить такое понятие как метрика расстояний. Метрика расстояний является функцией, которая задает расстояние между двумя наблюдениями. В среде `Matlab` функция попарного-расстояния `pdist2` поддерживает следующие метрики расстояния: Евклидово расстояние, квадратичное Евклидово расстояние, стандартизированное Евклидово расстояние, расстояние Махаланобиса, расстояние городского квартала (манхэттенское расстояние), расстояние Минковского, расстояние Чебышева, расстояние косинуса, корреляционное расстояние, расстояние Хемминга, расстояние Жаккара и расстояние Спирмена. Описание данных метрик сведено в Таблицу 1 и продолжение в Таблице 2.

Таблица 3-1. Метрики расстояний.

Значение	Формула	Описание
`'euclidean'`	$\rho(x,x') = \sqrt{\sum_{i=1}^{N}(x_i - x_i')^2}$ Где x и x' точки в N-мерном пространстве	Значение по умолчанию. Наиболее распространенная функция для определения расстояния. Представляет собой геометрическое расстояние в многомерном пространстве между двумя точками. Евклидово расстояние является особым случаем расстояния Минковкого, где $p = 2$

'squaredeuclidean'	$\rho(x,x') = \sum_{i=1}^{N}(x_i - x_i')^2$	Применяется для придания большего вес более отдаленных друг от друга объектов.
'seuclidean'	$\rho(x,x') = \sqrt{\sum_{i=1}^{N} S_i(x_i - x_i')^2}$	Каждое координатное различие между наблюдениями масштабируется путем деления на соответствующий элемент стандартного отклонения, $S = std(x,'omitnan')$
'mahalanobis'	$\rho(x,x') = (X_i - X_i')C^{-1}(X_i - X_i')^T$	Предполагает, что точки множества эллипсоидально (частный случай – сферически) распределены вокруг центра масс – расстояние безразмерно и масштабно-инвариантно. Ковариация – это численное выражение свойства ковариантности двух признаков точек. Свойство ковариантности означает, что признаки имеют тенденцию изменяться совместно. C – ковариационная матрица
'cityblock'	$\rho(x,x') = \sum_{i=1}^{N}\lvert x_i - x_i'\rvert$	Это расстояние является средним разностей по координатам. Для этой меры влияние отдельных больших разностей (выбросов) уменьшается (т.к. они не возводятся в квадрат). Расстояние городского квадрата является особым случаем расстояния Минковского, где $p = 1$
'minkowski'	$\rho(x,x') = \sqrt{\sum_{i=1}^{N}\lvert x_i - x_i'\rvert^p}$	Параметрическая метрика на евклидовом пространстве, которую можно рассматривать как обобщение евклидова

		расстояния и расстояния городских кварталов.
'chebychev'	$\rho(x, x') = \max(\sum_{i=1}^{N} \lvert x_i - x_i' \rvert)$	Это расстояние может оказаться полезным, когда нужно определить два объекта как «различные», если они различаются по какой-либо одной координате. Расстояние городского квадрата является особым случаем расстояния Минковского, где $p = \infty$
'cosine'	$\rho(x, x') = \frac{\sum_{i=1}^{N} x_i \cdot x_i'}{\sqrt{\sum_{i=1}^{N} (x_i)^2} \cdot \sqrt{\sum_{i=1}^{N} (x_i')^2}}$	Метод сходства между двумя ненулевыми векторами пространства данных. Определен как косинус угла между ними, которых также совпадает с внутренним произведением тех же векторов, нормализованных к обоим, имеющим длину 1.

Таблица 3-2. Продолжение таблицы 1. Метрики расстояний.

Функция	Описание
'correlation'	Ещё называют скорректированным косинусным сходством. Использование данного метода компенсирует недостаток предыдущего метода путем вычитания среднего значения соответствующе точки набора.
'hamming'	Число позиций, в которых соответствующие точки двух наборов одинаковой длины различны.
'jaccard'	Используется для калибрования сходства и разнообразия из образцов наборов. Основан на пересечении и объединении данных
'spearman'	Является методом ранговой корреляции и используется для выявления и оценки тесноты связи между двумя рядами сопоставляемых количественных показателей.

4. Методы решения задачи кластеризации

Приведем обзор различных алгоритмов кластеризации, чтобы решить, какой из алгоритмов лучше всего подойдет для решения поставленной задачи.

- **Иерархическая кластеризация**

Иерархическая кластеризация группирует данные в различных масштабах, создавая дерево кластеров или дендрограмму. Дерево – это не единственный набор кластеров, а многоуровневая иерархия, в которой кластеры на одном уровне объединяются в кластеры на следующем. Это позволяет решить какой уровень кластеризации подходит для решения поставленной задачи.

Среди алгоритмов иерархической кластеризации выделяют два основных типа: восходящие и нисходящие алгоритмы. Нисходящие алгоритмы работают по принципу «сверху-вниз»: в начале все объекты помещаются в один кластер, который затем разбивается на все более мелкие кластеры. Восходящие же алгоритмы работают по обратному алгоритму «снизу-вверх»: в начале работы каждый объект помещается в отдельный кластер, а затем объединяются во все более крупные кластеры, пока все объекты выборки не будут содержатся в одном кластере. В итоге строится система вложенных разбиений.

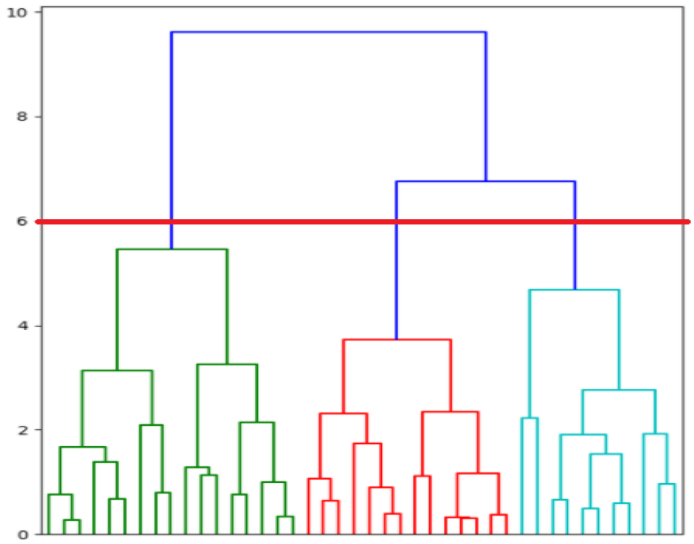


Рис. 4-4-1. Пример иерархической кластеризации – дендрограмма. Уровнями можно задать количество кластеров разбиения.

- **Метод k-средних**

Данный метод кластеризации считается наиболее простым, но в то же время недостаточно точным. Суть данного алгоритма в следующем: метод разбивает множество элементов векторного пространства на заранее известное число кластеров k. Алгоритм стремится минимизировать среднеквадратическое отклонения на точках каждого кластера. На каждой итерации перевычисляется центр масс для каждого кластера, полученного на предыдущей итерации, затем векторы разбиваются на кластеры вновь в соответствии с тем, какой из новых центров оказался ближе по выбранной метрике. Алгоритм завершается, когда на какой-то итерации не происходит изменения внутрикластерного расстояния. Это происходит за конечное число итераций, так как количество возможных разбиений конечного множества конечно, а на каждом шаге суммарное квадратичное отклонение уменьшается, поэтому зацикливание невозможно.

- **Смешанная гауссовская модель**

Смешанная гауссовская модель формирует кластеры по принципу суперпозиции многомерных гауссовских законов распределения плотности вероятности. Для каждого наблюдения данный метод определяет апостериорные плотности вероятности, которые указывают, что наблюдение имеет некоторую вероятность принадлежности к каждому кластеру.

По смешанной гауссовской модели можно выполнить *hard-кластеризацию* (жесткая кластеризация), которая гарантирует высокую точность группировки, где все запросы привязываются друг к другу и в один кластер попадают только на сто процентов совместимые запросы, путем выбора компонента, который максимизирует апостериорную вероятность. Hard-кластеризация определяет точку из данных только в один кластер. И в итоге кластер образуется там, где наибольшая апостериорная вероятность.

Кроме hard-кластеризации можно использовать смешанную гауссовскую модель для так называемой *soft-кластеризации* (нечетной кластеризации). При данном методе присваивается вероятность попадания точки данных для каждого кластера, и эта вероятность указывает на силу связи точки данных с кластером.

В отличие от методов hard-кластеризации, методы soft-кластеризации являются гибкими, поскольку они могут назначать точку данных нескольким кластерам.

- **Спектральная кластеризация**

Спектральная кластеризация использует спектр (собственных значений) матрицы сходства данных для осуществления снижения размерности перед кластеризацией в пространство меньших размерностей. Матрица сходства подается в качестве входа и состоит из количественных оценок относительно схожести каждой пары точек в данных. В низкой размерности данные в кластерах разделяются более широко, позволяя использовать далее алгоритмы кластеризации, например, k-средних. Низкая размерность данных основана на значимых собственных векторах матрицы Кирхгофа. Данная матрица является одним из способов представить график подобия, который отображает локальные отношения между точками данных, как неориентированный граф.

- **DBSCAN**

DBSCAN (Density-based spatial clustering of applications with noise, плотностной алгоритм пространственной кластеризации с присутствием шума) является основанным на плотности алгоритмом, который идентифицирует кластеры произвольной формы и выбросы (шум) в данных. Во время кластеризации DBSCAN определяет точки, которые не принадлежат ни одному из кластеров и относит к шуму. Для этого метода не требуется предварительное знание количества кластеров.

Далее требуется определить какой метод кластеризации использовать для решения поставленной задачи. Напомним, что решается задача проведения группирования отметок в кластеры [6]. В свою очередь количество кластеров заранее не известно, поэтому подойдут алгоритмы, которые не учитывают количество кластеров, как входной параметр. Также радиолокационные отметки имеют зашумленные данные, поэтому нужно найти такой алгоритм, который бы учитывал случайные выбросы. Тогда учитывая все высказывания подходит

алгоритм DBSCAN. Этим алгоритмом в данной работе будет реализована кластеризация отметок.

4.1. Описание алгоритма DBSCAN

Имея дело с пространственными кластерами различной плотности, размера и формы, может быть сложно обнаружить группу точек. Задача может быть еще более сложной, если данные содержат шум и выбросы. Для работы с большими пространственными базами данных обычно применяется алгоритм DBSCAN [15]. Основных причины использования алгоритма:

1. Требует минимальных знаний предметной области.
2. Он может обнаружить кластеры произвольной формы.
3. Эффективен для большой базы данных, то есть размер выборки превышает несколько тысяч.

Рассматривая данные причины убеждаемся, что с помощью данного алгоритма возможно решить поставленную задачу.

Основная концепция алгоритма DBSCAN состоит в том, чтобы найти области высокой плотности, которые отделены друг от друга областями низкой плотности. И чтобы правильно применить этот алгоритм кластеризации для решения поставленной задачи, требуется подробнее рассмотреть его параметры и характеристики. На вход метод просит матрицу близости и два параметра – радиус $epsilon$-окрестности и $minpts$-минимальное количество соседей.

Для определения $epsilon$ и $minpts$ введем несколько определений. Пусть задана некоторая симметричная функция расстояния $\rho(x, x')$ и константы ε и m. Тогда:

1. Назовем область $E(x)$, для которой $\forall x$: $\rho(x, x') \leq \varepsilon$, где ε – окрестность объекта x
2. Центральным объектом или ядерным объектом степени m называется объект, ε-окрестность которого содержит не менее m объектов: $|E(x)| \geq m$

3. Объект p непосредственно плотно-достижим из объекта q, если $p \in E(q)$ и q – корневой объект

4. Объект p плотно-достижим из объекта q, если $\exists p_1, p_2 \dots p_n, p_1 = q, p_n = p$, такие что $\forall i \in 1 \dots n-1: p_i + 1$ непосредственно плотно-достижим из p_i

Следуя определению плотной области, точка может быть классифицирована как основная точка если $|E(x)| \geq m$. Центральные точки, как следует из названия, обычно находятся внутри кластера. Пограничная точка (достижимая по плотности точка) имеет меньше, чем m в своей $E(x)$ области, но лежит в окрестности другой центральной точки. Шум (выпадающая точка) – это любая точка данных, которая не является ни основной, ни пограничной.

Достижимость не является симметричным отношением, поскольку, по определению, никакая точка не может быть достигнута из неосновной точки, независимо от расстояния (так что неосновная точка может быть достижимой, но ничто не может быть достигнуто из неё). Поэтому дальнейшее понятие *связности* необходимо для формального определения области кластеров, найденных алгоритмом DBSCAN. Две точки p и q *связаны по плотности*, если имеется точка o, такая что и p, и q достижимы из o. Связность по плотности *является* симметричным отношением.

Тогда кластер удовлетворяет двум свойствам:

1. Все точки в кластере попарно связны по плотности.

2. Если точка достижима по плотности из какой-то точки кластера, она также принадлежит кластеру.

Этапы работы алгоритма DBSCAN.

Выше приведены определения и теперь можно описать шаги алгоритма DBSCAN:

1. Алгоритм начинается с произвольной точки, которая ещё не была просмотрена.

2. Выбирается ε-окрестность выбранной точки и, если она содержит m точек, то начинается формирование кластера, в противном случае точка помечается как шум. Эта точка может быть позже найдена в ε-окрестности другой точки и, таким образом, может стать частью кластера.
3. Если точка найдена как основная точка, то точки в ε-окрестности также являются частью этого кластера. Таким образом, все точки найденные в ε-окрестность, добавляются вместе с их собственной ε-окрестностью, если они также являются основными точками.
4. Вышеописанный процесс продолжается, пока не будет найден связный по плотности кластер.
5. Процесс возобновляется. Выбирается и обрабатывается новая непосещённая точка, что ведёт к обнаружению следующего кластера или шума.

Введя определения и описание работы алгоритма DBSCAN перейдем к недостаткам. Приведем два самых больших недостатков алгоритма:

1. Если в базе данных есть точки, которые образуют кластеры различной плотности, то с помощью данного алгоритма не удается хорошо раскластеризовать точки, поскольку кластеризация зависит от параметров ε и m, они не могут быть выбраны отдельно для всех кластеров.
2. Если данные и функции не так хорошо понятны специалисту в области, то может быть настроить параметры ε и m. Тогда требуется сравнение нескольких итераций с различными значениями ε и m.

В итоге сложность применения алгоритма DBSCAN – это выбор ε *и* m.

Существуют эвристики для подбора ε *и* m. Чаще всего применяется такой метод и его вариации:

1. Выбирается m. Используются значения больше или равные порядку размерности входных данных. Чем более неоднородный датасет, тем соответственно больше уровень шума и тогда следует взять m большим.

2. Для оценки значения ε требуется вычислить среднее расстояние по m для каждой точки входных данных. Далее сортируются полученные значения по возрастанию и строится график.
3. Построится резко возрастающий график. Значение ε выбирается в полосе, где происходит сильный перегиб кривой, это область, где точки начинают затухать в область выбросов [15].

Итог. DBSCAN отлично работает на плотных, хорошо отделенных друг от друга кластерах. При этом их форма совершенно не важна. Алгоритм отлично обнаруживает кластеры малой размерности. Успешно применяется для большого датасета $N = 10^{6\ldots8}$, причем сложность элементов датасета значения не имеет. Количество элементов в кластере может варьироваться, количество выбросов значения не имеет, если они рассеяны по большому объему, а также количество кластеров значения не имеет.

Сведем характеристики данных алгоритмов в таблицу 3.

Таблица 4-1. Характеристики основных методов кластеризации.

Алгоритм кластеризации	**Описание алгоритма кластеризации**	**Входные данные**	**Требование конкретного количества кластеров**	**Кластерные идентифицированные формы**	**Выделение выбросов**
Иерархическая кластеризация	Расстояние между объектами	Попарные расстояния между наблюдениями	Нет	Кластеры произвольной формы	Нет
K-средних	Расстояние между объектами и центроидами	Фактические наблюдения	Да	Сфероидальные кластеры с равной диагональный ковариацией	Нет
Смешанные гауссовские модели	Смесь нормальных распределений	Фактические наблюдения	Да	Сфероидальные кластеры с различными структурами ковариации	Да

Спектральная кластеризация	График, представляющий связи между точками данных	Фактические наблюдения или матрица подобия	Да, но алгоритм также обеспечивает способ оценить количество кластеров	Кластеры произвольной формы	Нет
DBSCAN	**Плотность областей в данных**	**Физические наблюдения или попарные расстояния между наблюдениями**	**Нет**	**Кластеры произвольной формы**	**Да**

5. Моделирование

Для решения задачи и с целью качественной оценки эффективности алгоритма кластеризации DBSCAN на данных, которые представляют собой элементарные радиоимпульсы, требуется описать параметры имитационной модели сигнала.

5.1. Исходные данные моделирования

Для того, чтобы описать, что из себя представляют элементарные импульсы, был проведен анализ реальных записей обнаруживаемых элементарных импульсов с многопозиционного пассивного радиолокационного комплекса. Одиночные импульсы имеют следующие параметры: время прихода импульса $T_{\text{пр}}$, длительность импульса $\tau_{\text{и}}$, период импульса $T_{\text{и}}$, это разность времен между текущим импульсом и предыдущим импульсом, несущая частота F. Анализируя реальные записи было выяснено, что импульсы с некоторыми параметрами встречаются чаще всего. А именно: минимальное значение периода $T_{\text{и.мин}} = 2$ мкс, наиболее часто встречающиеся несущие частоты сигналов представляют собой следующий дискретный набор $F = [1{,}09; 1{,}5; 5{,}48; 9{,}8; 16]$ ГГц, наиболее часто встречающиеся длительности импульса представляют собой дискретный набор $\tau_{\text{и}} = [50; 100; 500; 20000; 65000]$ нс. Параметры этих импульсов взяты за основу сигналов в имитационной модели.

Реальные записи всегда содержат шумы наблюдения, поэтому это следует учесть и для имитационной модели. В итоге подобрались следующие шумы наблюдения: для времени прихода импульса $\Delta T_{\text{пр}} = 50$ нс, по несущей частоте $\Delta F = 1$кГц, для длительности импульса $\Delta\tau_{\text{и}} = 10$ нс.

Кроме шума наблюдения существуют также отклонения от мгновенных значений. Положим следующее: девиация частоты в долях от несущей принимает равновероятные значения в следующем диапазоне целых чисел $\Delta F_{\text{д}} = 0.001 \ldots 0.005$ $(0.1 \ldots 0.5\%)$, девиация длительности импульса в долях $\Delta\tau_{\text{и.д.}} = 0.001 \ldots 0.005$ $(0.1 \ldots 0.5\%)$.

5.2. Формирование выборки

Задача следующая. Требуется сформировать выборку импульсов в которой будут присутствовать повторяющиеся несколько раз сигналы с одинаковыми параметрами – паттерны. Именно эти паттерны должен выделить алгоритм DBSCAN при кластеризации.

Формирование выборки импульсов происходит следующим образом. Задаются параметры импульса:

1. Формируется случайным образом период между импульсами:

$$T = T_{\text{и.мин}} \cdot f([1,10]) + g(0, \Delta T_{\text{пр}}) \tag{5.1}$$

Где $f([1,10])$ – непрерывное равномерное распределение целых чисел, $g(0, \Delta T_{\text{пр}})$ – нормальный закон распределения с нулевым средним значением и дисперсией $T_{\text{пр}}$.

2. Формируется несущая частота импульса:

$$f = f(F) \cdot \left(1 + (-1) \cdot f([0,1]) \cdot f\left(\Delta F_{\text{д}}\right)\right) \tag{5.2}$$

3. Формируется случайным образом длительность импульса:

$$\delta t = f(\tau) \cdot \left(1 + (-1) \cdot f([0,1]) \cdot f\left(\Delta \tau_{\text{и.д.}}\right)\right) \tag{5.3}$$

4. Далее задается длина паттерна L
5. Формируется выборка импульсов из $N = 10000$ элементов, где случайным образом распределены паттерны. К параметрам импульсов добавляется шум наблюдения, распределенный по нормальному закону.

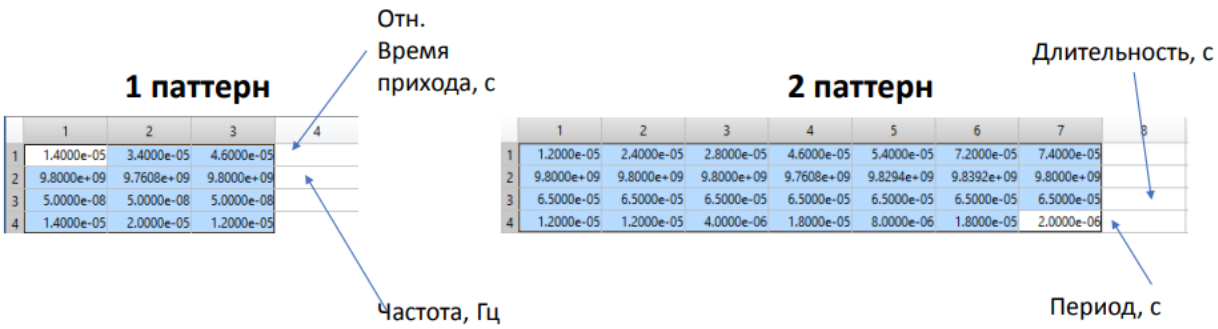
В итоге модель наблюдений описывается следующим образом:

$$Y = \boldsymbol{X} + \boldsymbol{\varepsilon} \tag{5.4}$$

где $\boldsymbol{\varepsilon}$ - матрица шумов наблюдений размерностью (Nx4). $\boldsymbol{X}$ - матрица состояния размерностью (Nx4). $\boldsymbol{Y}$ - матрица наблюдений размерностью (Nx4).

5.3. Результаты эксперимента

Требуется сформировать выборку импульсов и добавить в эту выборку два паттерна случайным образом. Размер 1-го паттерна возьмём равным трем импульсам, причём было принято, что три импульса это минимальный размер паттерна. Размер 2-го паттерна равен семи импульсам.



	1	2	3	4
1	1.4000e-05	3.4000e-05	4.6000e-05	
2	9.8000e+09	9.7608e+09	9.8000e+09	
3	5.0000e-08	5.0000e-08	5.0000e-08	
4	1.4000e-05	2.0000e-05	1.2000e-05	

	1	2	3	4	5	6	7	8
1	1.2000e-05	2.4000e-05	2.8000e-05	4.6000e-05	5.4000e-05	7.2000e-05	7.4000e-05	
2	9.8000e+09	9.8000e+09	9.8000e+09	9.7608e+09	9.8294e+09	9.8392e+09	9.8000e+09	
3	6.5000e-05	6.5000e-05	6.5000e-05	6.5000e-05	6.5000e-05	6.5000e-05	6.5000e-05	
4	1.2000e-05	1.2000e-05	4.0000e-06	1.8000e-05	8.0000e-06	1.8000e-05	2.0000e-06	

Рис. 5-1 Пример паттернов.

Предполагается, что алгоритмом кластеризации DBSCAN можно будет выделить добавленные паттерны в отдельные кластеры, тем самым сформировав два кластера со схожими паттернами. Остальные импульсы он будет считать помехами и шумами (выбросами).

Подробнее проанализируем смоделированную выборку. Для этого построим вероятностные распределение параметров матрицы наблюдений (рис. 5-2) и представление импульсов в выборке (рис. 5-3, 5-4).

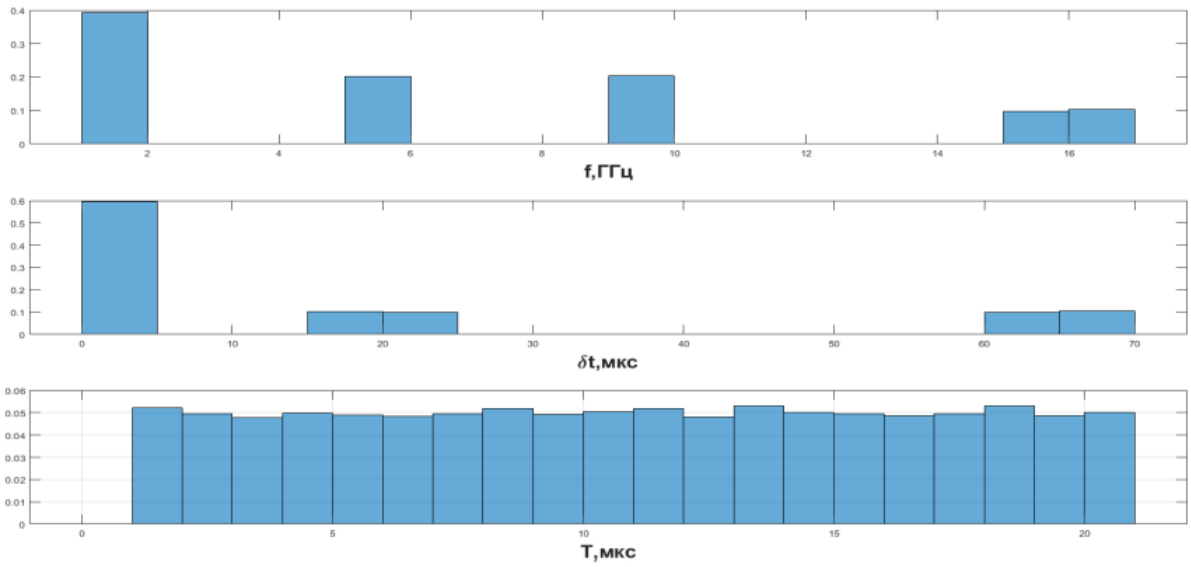


Рис. 5-2 Вероятностные распределения параметров матрицы наблюдений (без времен прихода)

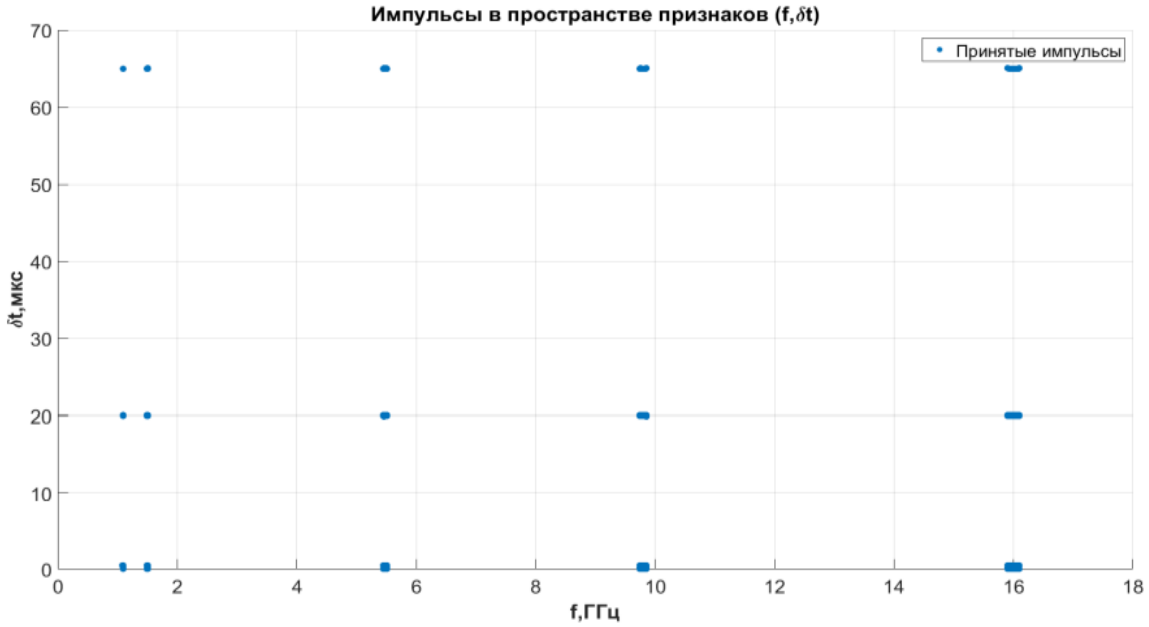


Рис. 5-3 Двумерное представление импульсов

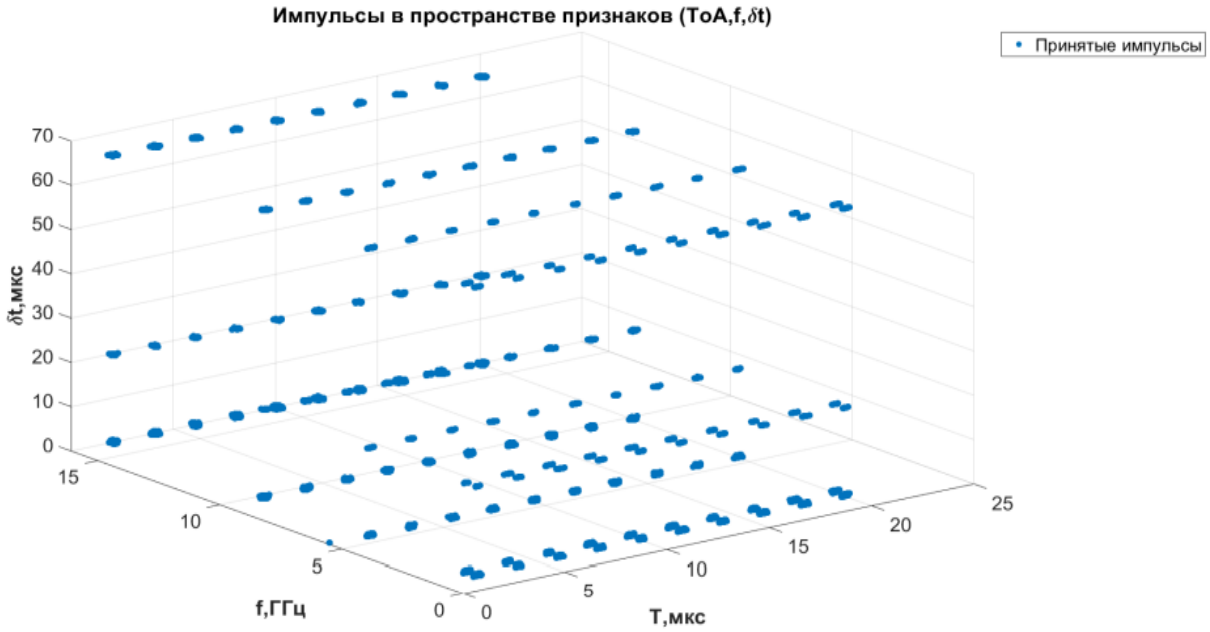


Рис. 5-4 Трехмерное представление импульсов с осью значений периодов

Далее перед запуском алгоритма кластеризации требуется нормализовать, либо стандартизировать выборку импульсов. В данной работе была выбрана Z-стандартизация параметров матрицы наблюдения. Где Z-стандартизация – это такое преобразование данных, которое позволяет перевести шкалу на Z шкалу, где среднее значение будет равняться нулю, а стандартное отклонение равняется единице.

Параметр времени прихода импульса не используется в алгоритме кластеризации, т.к. он является не информативным. Поэтому входными параметрами для алгоритма кластеризации являются следующие три параметра: несущая частота импульса, период между импульсами и длительность импульса.

Перед нами ставилась задача находить паттерны из выборки импульсов. Поэтому требуется так преобразовать входные данные, чтобы алгоритм кластеризации искал именно набор импульсов, сгруппированный в паттерн. Для этого матрицу наблюдений $\boldsymbol{Y}$ требуется видоизменить, расширив её до размерности $(N * (3 * \text{размер\_паттерна} - 1))$. Назовем эту операцию «окном смещения». Данную операцию необходимо проделать для двух случаев, ищется два паттерна разной длины в выборке импульсов.

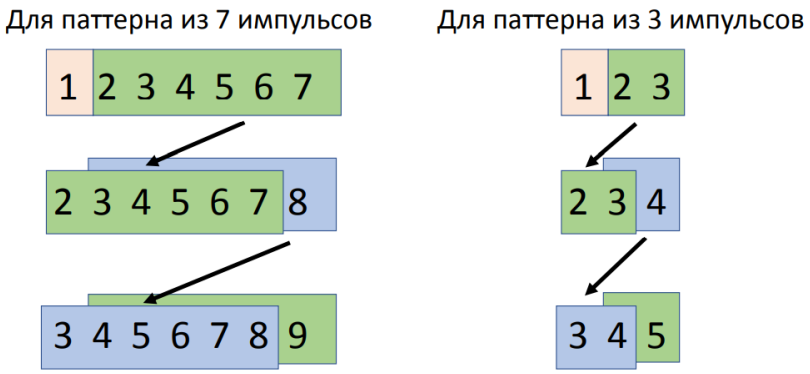


Рис. 5-5 Операция окно смещения

Для работы алгоритма кластеризации DBSCAN требуется задать следующие параметры: число минимальных соседей $minpts$ и радиус поиска соседей $epsilon$. В предыдущей работе было подробно проанализировано, как влияют данные параметры на точность кластеризации и были сделаны следующие выводы: увеличение числа точек $minpts$, ведет к увеличению значения параметра ε-окрестности и при увеличении параметра ε-окрестности алгоритм находит большее количество кластеров. Так-же в предыдущей работе значение ε-окрестности подбиралось каждый раз по графику k-расстояний и не было автоматизировано. Поэтому в данной работе следующий подход к оценке входных параметров алгоритма кластеризации.

Число минимальных соседей $minpts$ для модели выбирается из учета минимального числа случайного повторения одного из двух паттернов в выборке (при формировании выборки импульсов заранее известны индексы паттернов и число их повторений). $Epsilon$ выбирается исходя из отсортированной по возрастанию матрицы попарных расстояний с евклидовой метрикой. Данная метрика была взята из анализа предыдущих работ по данной теме.

В итоге была получена следующая реализация где первый паттерн длинной три импульса повторился двенадцать раз, а второй паттерн длинной семь импульсов семь раз. Тогда $minpts = 7$ берем равным семи.

Подробнее представим выбор ε-окрестности на рис. 5-6, 5-7.

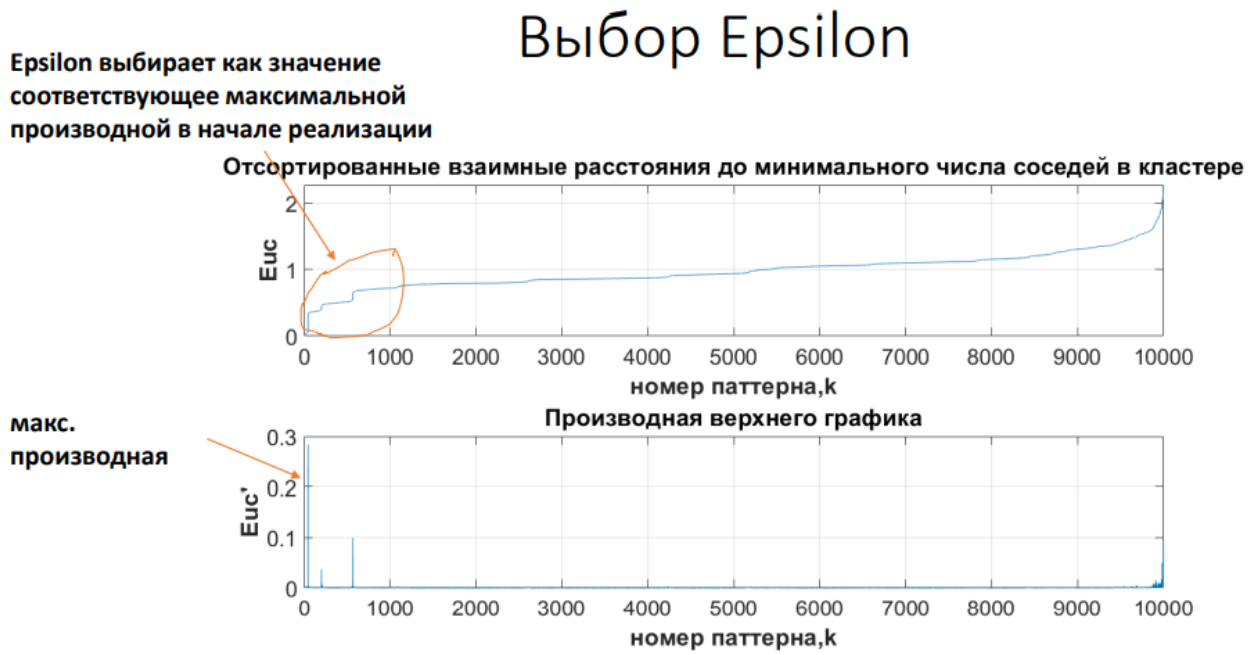


Рис. 5-6 Выбор ε-окрестности.

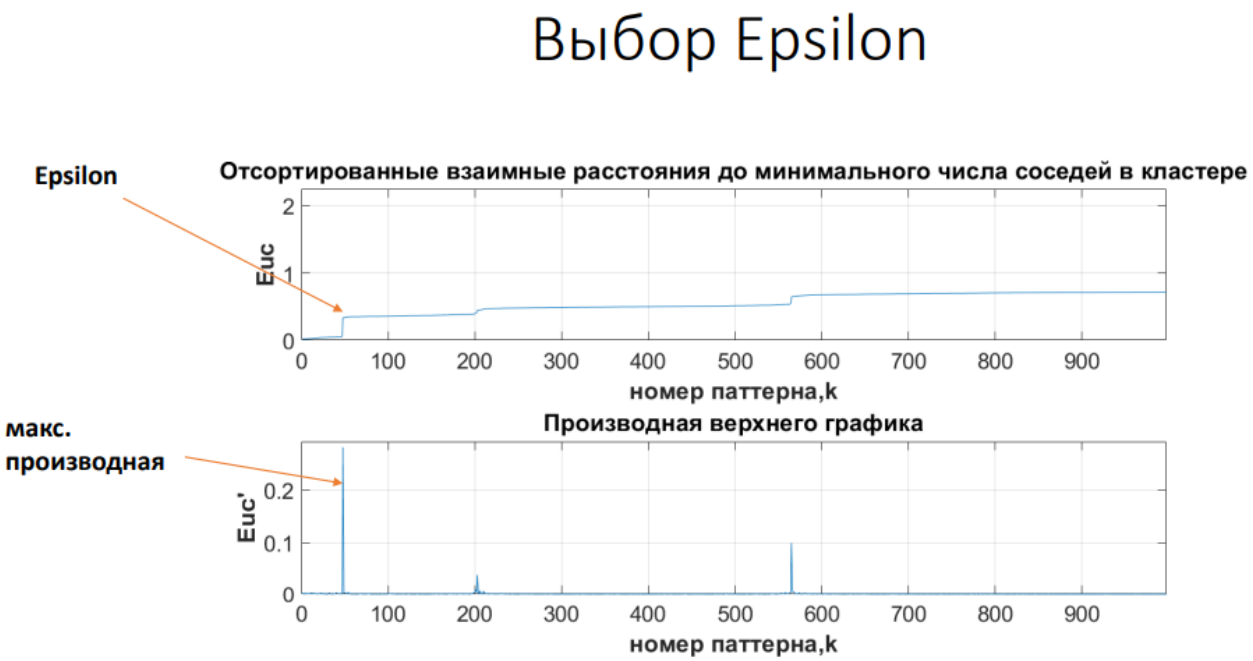


Рис. 5-7 Выбор ε-окрестности в увеличенном масштабе.

Далее представлены результаты алгоритма кластеризации для данной выборки, которая содержит два паттерна разной длины.

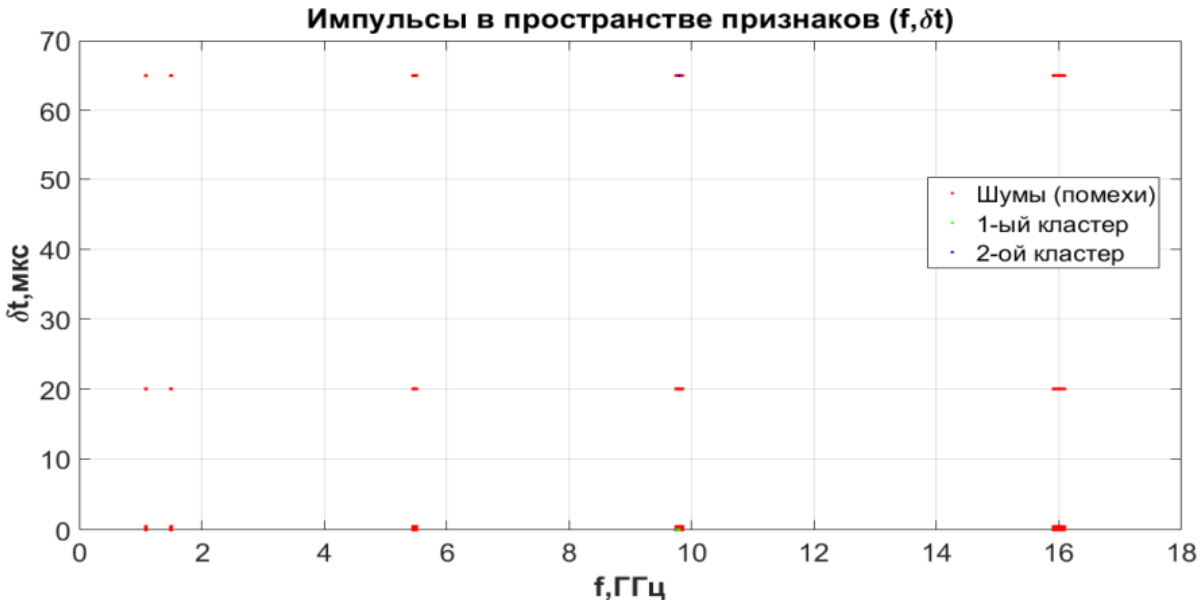


Рис. 5-8 Импульсы в пространстве признаков частоты и длительности импульса.

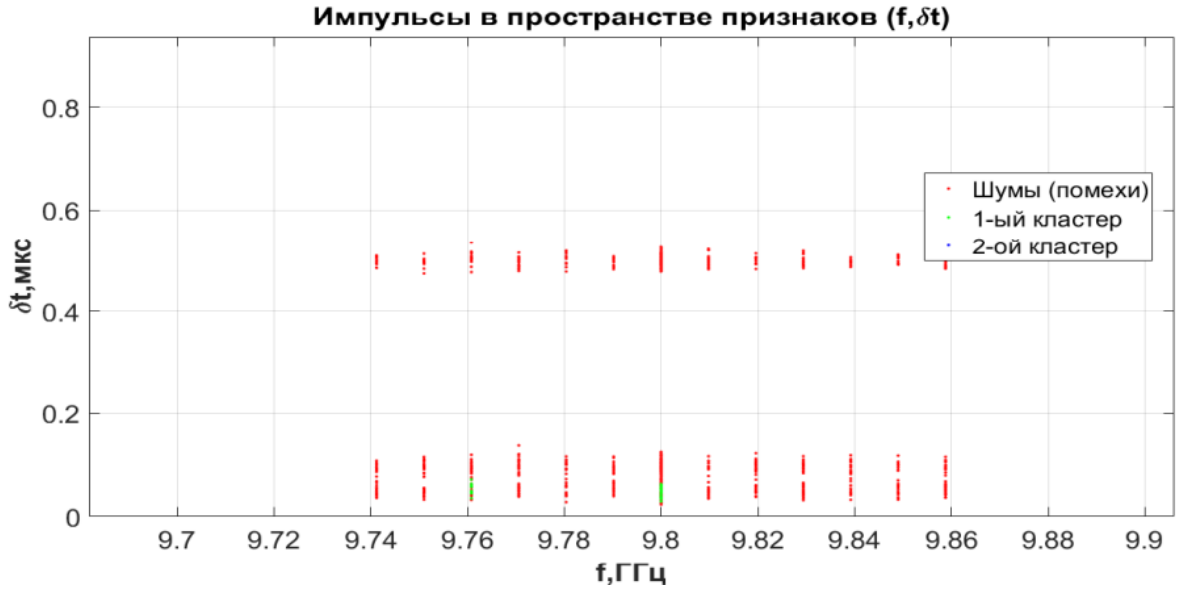


Рис. 5-9 Импульсы в пространстве признаков частоты и длительности импульса в увеличенном масштабе первый кластер.

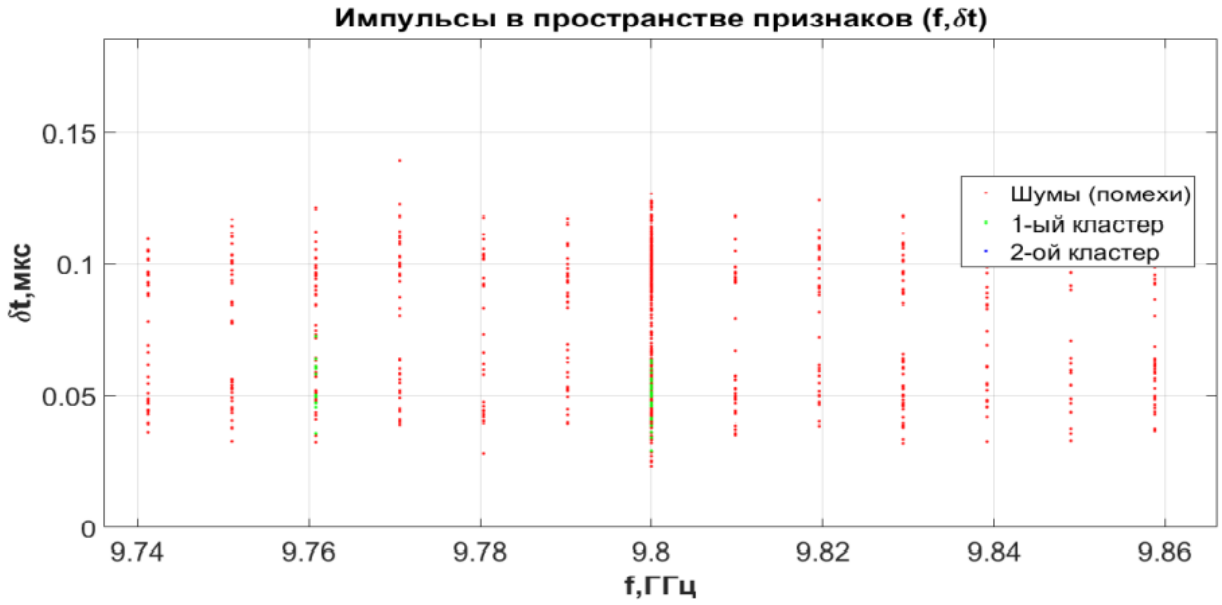


Рис. 5-10 Импульсы в пространстве признаков частоты и длительности импульса в увеличенном масштабе первый кластер.

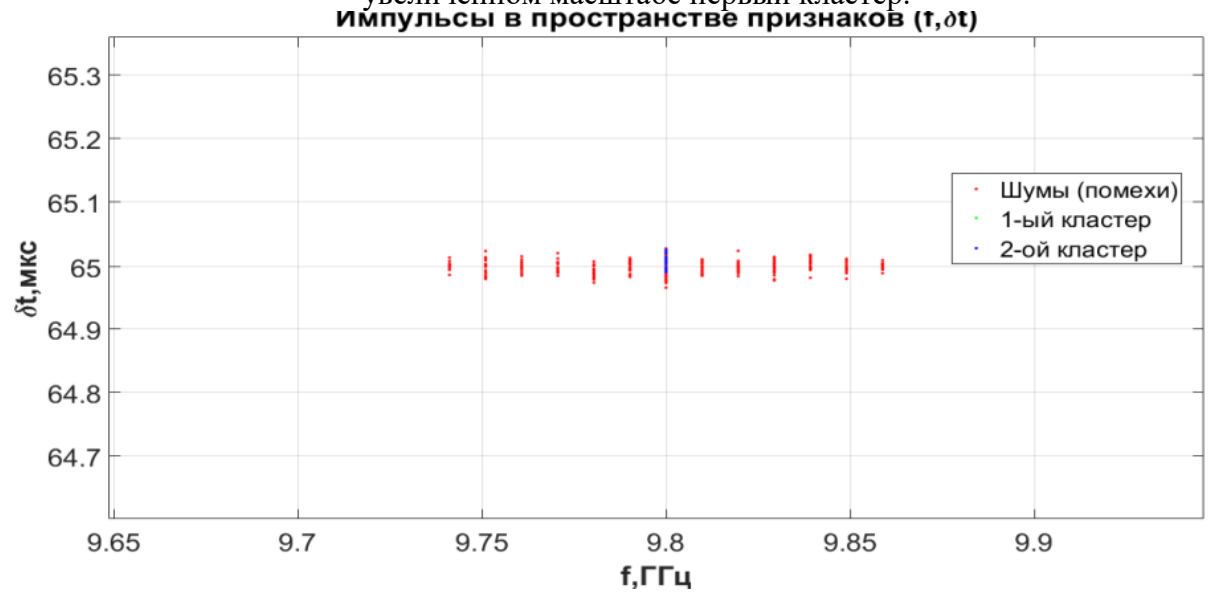


Рис. 5-11 Импульсы в пространстве признаков частоты и длительности импульса в увеличенном масштабе второй кластер.

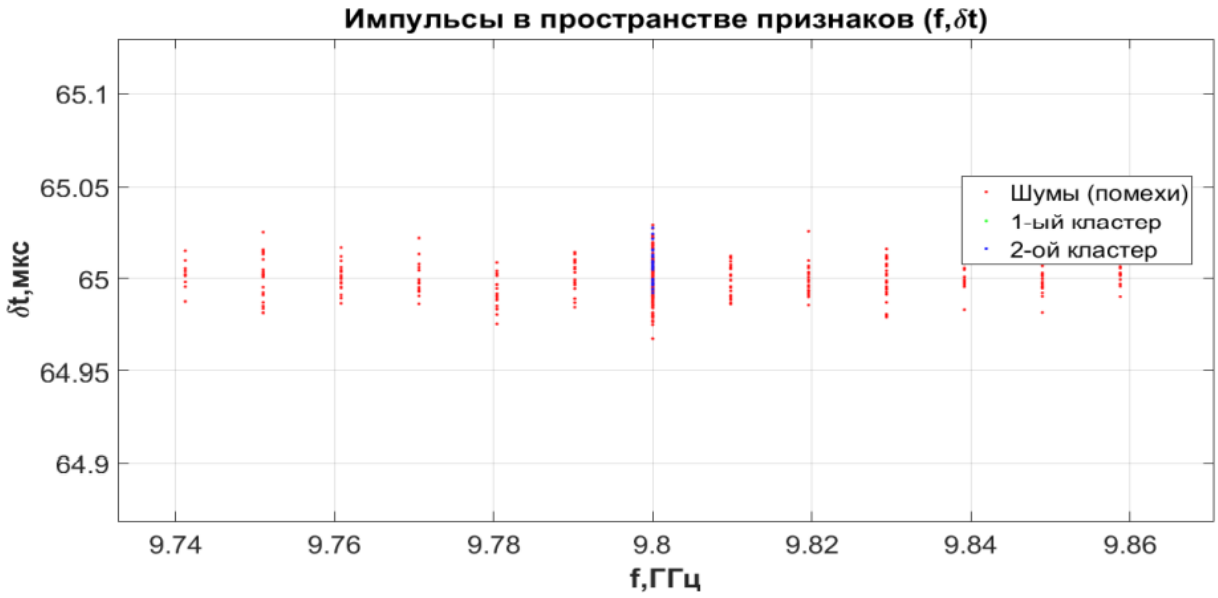


Рис. 5-12 Импульсы в пространстве признаков частоты и длительности импульса в увеличенном масштабе второй кластер.

Алгоритм DBSCAN явно выделил два кластера. Теперь требуется проверить точность оценки качества кластеризации. Чтобы это сделать нужно ввести обозначения и формулы по котором будет происходить пересчет точности.

Точность правильного распознавания – проверка оцененных положений паттернов в выборке к истинным положениям:

$$accuracy = \frac{\widehat{N}_{\text{истинных паттернов}}}{N_{\text{истинных паттернов}}} \tag{5.5}$$

Ложное распознавание:

$$falseAlarm = \frac{\widehat{N}_{\text{ложных паттернов}}}{N - max(\text{размер паттерна}) + 1 - N_{\text{истинных паттернов}}} \tag{5.6}$$

Число оцененных кластеров:

$$\widehat{N}_{\text{кластеров}} \tag{5.7}$$

Итого, получаем следующие результаты кластеризации на одной реализации.

Алгоритм:

- Число кластеров - 2
- Позиции 1-ого кластера - 1388 1776 2739 2943 3656 4770 5761 7984 8327 8529 9537 9745
- Позиции 2-ого кластера - 1806 2326 4045 4896 5576 6276 8874

Истина:

- Число кластеров - 2
- Позиции 1-ого кластера - 1388 1776 2739 2943 3656 4770 5761 7984 8327 8529 9537 9745
- Позиции 2-ого кластера - 1806 2326 4045 4896 5576 6276 8874

Реальных кластеров - 2 Выявили – 2.

Точность распознавания паттернов алгоритмом на данной реализации составила - 100% Процент определенных ложных паттернов - 0%.

На данном примере алгоритм точно определил все паттерны и их положения в выборке. Сто процентная точность могла быть не во всех выборках, поэтому проведем ещё несколько экспериментов.

Проведем следующие эксперименты, чтобы посмотреть, как меняется точность, при различных входных параметрах. Будем увеличивать число реализаций, например, $realization = 100, 1000$ и повторять моделирование. Увеличим размер выборки импульсов до $N = 100000$. И увеличим число паттернов в выборке $n = 3; 5$. Результаты экспериментов сведем в таблицу 4.

Таблица 4 Результаты моделирования

Размер выборки импульсов N	Число паттернов в выборке n	Число реализаций r	Средняя точность определения паттернов $accuracy$	Средний процент определения ложных паттернов $falseAlarm$	Среднее число выявляемых кластеров $\hat{n}$
10000	2 [3; 7]	1	100%	0%	2
10000	2 [3; 7]	100	99.5297%	0.0020038 %	2
10000	2 [3; 7]	1000	99.2537%	0.017998%	2.04
100000	2 [3; 7]	1	100%	0.0010019 %	2
100000	2 [3; 7]	10	100%	0.0065132 %	2.8
10000	3 [3; 7; 15]	1	100%	0%	3
10000	3 [3; 7; 15]	100	97.8412%	0.075268%	3.15
10000	3 [3; 7; 15]	500	99.2151%	0.042529%	3.138
10000	3 [3; 7; 15]	1000	99.0613%	0.039426%	3.124
10000	5 [3; 7; 10; 15; 20]	1	100%	0.010056%	5

10000	5 [3; 7; 10; 15; 20]	100	99.3053%	0.043205%	5.21

По результатам эксперимента получили высокую точность распознавания паттернов в смоделированной импульсной выборке.

Было рассмотрено различное число паттернов в выборке, различная длина импульсной выборки и различное число реализаций.

Выводы

В ходе выполнения научно-исследовательской работы были рассмотрены этапы обработки радиолокационной информации. Было сказано, что применение применение алгоритмов статистики и машинного обучения к системам обработки радиолокационной информации считается актуальным и многообещающим направлением. В данной работе было предложено применить алгоритмы кластеризации на этапе обнаружения отметок от целей. Решение такое задачи позволит в будущем подбирать более эффективные алгоритмы вторичной (траекторной) обработки целей.

Для решения задачи кластеризации отметок от цели был выбран алгоритм кластеризации DBSCAN – плотностному алгоритму пространственной кластеризации с присутствием шума.

В данной работе была смоделирована имитационная модель элементарных радиоимпульсов, основанная на анализе реальных записей обнаруживаемых элементарных импульсов с многопозиционного пассивного радиолокационного комплекса. Применение алгоритма кластеризации к данным, сформированным на модели, показало, что алгоритм кластеризации DBSCAN верно определяет паттерны в выборке с высокой точностью больше 90%.

Дальнейшая работа в данной направлении – это кластеризовать реальные записи с многопозиционного пассивного радиолокационного комплекса с целью выделения повторяющихся паттернов и дальнейшего анализа записей.

Список литературы

1. Галушкин А.И. Нейрокомпьютеры (Нейрокомпьютеры и их применение. Кн. 3). М.: ИПРЖР, 2000. 528 с.
2. Barton D.K. Radar system analysis and modeling. – Artech House, Boston, MA, 2005.
3. Попов Г.П. Инженерная психология в радиолокации. М.: Сов. Радио, 1971. 143с.
4. Кузьмин С.З. Цифровая обработка радиолокационной информации. М.: Сов. радио,1967. 400 с.
5. Левин Б.Р. Теоретические основы статистической радиотехники. 3-е изд. перераб. идоп. М.: Радио и связь, 1989. 656 с.
6. Перов А.И. Оптимальный алгоритм дискретного сопровождения многих целей сидентификацией измерений // Радиотехника, No1, 2003.
7. Петухов СИ., Степанов А.Н. Эффективность ракетных средств ПВО. М.:Воениздат, 1976. 104 с.
8. Радиолокационные системы: Основы построения и теория. Справочник / Под ред.Я.Д. Ширмана. М.: ЗАО "МАКВИС", 1998. 828 с.
9. Татузов А.Л., Чухлеб Ф.С. Использование нейросетевой технологии при обработкерадиолокационной информации // Информационные Технологии. 1999, No 1, С. 25-33.
10. Татузов А.Л. Нейронные сети в задачах радиолокации. Кн. 28. – М.: Радиотехника, 2009. – 432 с.: ил. (Научная серия «Нейрокомпьютеры и их применение)
11. Dillard G.M. Mean-level detection of nonfluctuating signals // IEEE Trans., 1974, v.AES-10, no.6 (Nov. 1974), pp.795-799.

Код программы

```
clear all
clc
close all

% Начало замера тестов
tic

% --------------------------------------------------------------------
% TODO Инициализация данных.
rng(10)


rr = 100; % Число реализаций
stat_matrix = nan(rr,3); % Матрица статистики всех реализаций

time = datestr(now,'ddmmyyhhMMss');
fid = fopen([time '_logs.txt'], 'wt'); % Открыли файл для записи

freq = [1.09 1.5 5.48 9.8 16]*1e9; % Несущая частота
dur = [50 100 500 20000 65000]*1e-9; % Длительность импульса
T_min = 2e-6;

for ll=1:rr % Прогоняем много реализаций



N_signal1 = 3; % Число импульсов в паттерне 1.
N_signal2 = 7; % Число импульсов в паттерне 2.
N_signal3 = 10; % Число импульсов в паттерне 3.
N_signal4 = 15;
N_signal5 = 20;

% Формирование 1-го паттерна.
prev_T = 0;
pattern_freq = freq(randi(length(freq)));
pattern_dur = dur(randi(length(dur)));
pattern_signal1 = nan(4,N_signal1); %4 параметра импульса.
for i = 1:N_signal1
   pattern_signal1(:,i) =  make_impulse( pattern_freq, pattern_dur, T_min, prev_T );
   prev_T = pattern_signal1(1,i);
end


% Формирование 2-го паттерна.
prev_T = 0;
pattern_freq = freq(randi(length(freq)));
pattern_dur = dur(randi(length(dur)));
pattern_signal2 = nan(4,N_signal2); % 4 параметра импульса.
for i = 1:N_signal2
   pattern_signal2(:,i) =  make_impulse( pattern_freq, pattern_dur, T_min, prev_T );
   prev_T = pattern_signal2(1,i);
end


%Формирование 3-го паттерна.
prev_T = 0;
pattern_freq = freq(randi(length(freq)));
pattern_dur = dur(randi(length(dur)));
```

```
pattern_signal3 = nan(4,N_signal3); % 4 параметра импульса.
for i = 1:N_signal3
   pattern_signal3(:,i) =  make_impulse( pattern_freq, pattern_dur, T_min, prev_T );
   prev_T = pattern_signal3(1,i);
end

%Формирование 4-го паттерна.
prev_T = 0;
pattern_freq = freq(randi(length(freq)));
pattern_dur = dur(randi(length(dur)));
pattern_signal4 = nan(4,N_signal4); % 4 параметра импульса.
for i = 1:N_signal4
   pattern_signal4(:,i) =  make_impulse( pattern_freq, pattern_dur, T_min, prev_T );
   prev_T = pattern_signal4(1,i);
end

%Формирование 5-го паттерна.
prev_T = 0;
pattern_freq = freq(randi(length(freq)));
pattern_dur = dur(randi(length(dur)));
pattern_signal5 = nan(4,N_signal5); % 4 параметра импульса.
for i = 1:N_signal5
   pattern_signal5(:,i) =  make_impulse( pattern_freq, pattern_dur, T_min, prev_T );
   prev_T = pattern_signal5(1,i);
end

% prev_T = clock;
% prev_T = prev_T(6);
prev_T = 19.3680000000000;
N = 10000;

% Закидывание паттерна во всю выборку.
k1=0;
k2=0;
k3=0;
k4=0;
k5=0;

imp = nan(4,N);
positions1=0;
positions2=0;
positions3=0;
positions4=0;
positions5=0;

i = 1;
% TODO Добавление паттеринов в матрицу импульсов.
while i < N+1
   imp(:,i) =  make_impulse( freq, dur, T_min, prev_T );
   if (randi(1000) == 300) && (i<(N-N_signal1))
       k1=k1+1;
       %        i
       positions1(k1)= i;
       for j = 1:N_signal1
           imp(:,i) = pattern_signal1(:,j);
           imp(1,i) = imp(1,i) + prev_T;
           i = i + 1;
       end
       prev_T = imp(1,i-1);

   elseif (randi(1000) == 100) && (i<(N-N_signal2))
```

```
        k2=k2+1;
        %       i
        positions2(k2)= i;
        for j = 1:N_signal2
            imp(:,i) = pattern_signal2(:,j);
            imp(1,i) = imp(1,i) + prev_T;
            i = i + 1;
        end
        prev_T = imp(1,i-1);
    elseif (randi(1000) == 10) && (i<(N-N_signal3))
        k3=k3+1;
        %       i
        positions3(k3)= i;
        for j = 1:N_signal3
            imp(:,i) = pattern_signal3(:,j);
            imp(1,i) = imp(1,i) + prev_T;
            i = i + 1;
        end
        prev_T = imp(1,i-1);
    elseif (randi(1000) == 10) && (i<(N-N_signal4))
        k4=k4+1;
        %       i
        positions4(k4)= i;
        for j = 1:N_signal4
            imp(:,i) = pattern_signal4(:,j);
            imp(1,i) = imp(1,i) + prev_T;
            i = i + 1;
        end
        prev_T = imp(1,i-1);
    elseif (randi(1000) == 10) && (i<(N-N_signal5))
        k5=k5+1;
        %       i
        positions5(k5)= i;
        for j = 1:N_signal5
            imp(:,i) = pattern_signal5(:,j);
            imp(1,i) = imp(1,i) + prev_T;
            i = i + 1;
        end
        prev_T = imp(1,i-1);
    else
        prev_T = imp(1,i);
        i = i + 1;
    end

end

% Истинные позиции паттернов.
true_positions.position1 = positions1';
true_positions.position2 = positions2';
true_positions.position3 = positions3';
true_positions.position4 = positions4';
true_positions.position5 = positions5';


vector_k = [k1 k2 k3 k4 k5];
% vector_k = [k1 k2 k3];

% Добавление шума.
imp(1,:) = imp(1,:) + normrnd(0, 50e-9,[1, N]);
imp(2,:) = imp(2,:) + normrnd(0, 1e3,[1, N]);
imp(3,:) = imp(3,:) + normrnd(0, 10e-9,[1, N]);
```

```
imp(4,1) = 0;

for i=2:N
    imp(4,i) = imp(1,i) - imp(1,i-1);
end
% ------------------------------------------------------------------------



% ------------------------------------------------------------------------

%TODO графики пространства признаков.
%пространство признаков до нормировки
% figure(1)
% scatter(imp(2,:)*1e-9,imp(3,:)*1e6,50,'filled')
% legend('Принятые импульсы');
% grid on
% title('Импульсы в пространстве признаков (f,\deltat)')
% xlabel('f,ГГц','FontSize',20,...
%        'FontWeight','bold')
% ylabel('\deltat,мкс','FontSize',20,...
%        'FontWeight','bold')
% set(gca, 'FontSize', 20)
%
%
% figure(2)
% scatter3(imp(1,:),imp(2,:)*1e-9,imp(3,:)*1e6,50,'filled')
% grid on
% legend('Принятые импульсы');
% title('Импульсы в пространстве признаков (ToA,f,\deltat)')
% ylabel('f,ГГц','FontSize',20,...
%        'FontWeight','bold')
% zlabel('\deltat,мкс','FontSize',20,...
%        'FontWeight','bold')
% xlabel('t,c','FontSize',20,...
%        'FontWeight','bold')
% set(gca, 'FontSize', 20)
%
% figure(22)
% scatter3(imp(4,:)*1e6,imp(2,:)*1e-9,imp(3,:)*1e6,50,'filled')
% grid on
% legend('Принятые импульсы');
% title('Импульсы в пространстве признаков (ToA,f,\deltat)')
% ylabel('f,ГГц','FontSize',20,...
%        'FontWeight','bold')
% zlabel('\deltat,мкс','FontSize',20,...
%        'FontWeight','bold')
% xlabel('T,мкс','FontSize',20,...
%        'FontWeight','bold')
% set(gca, 'FontSize', 20)
%
%
% %построим распределения параметров импульсов
% figure(23)
% subplot(3,1,1)
% histogram(imp(2,:)*1e-9,'Normalization','probability')
% xlabel('f,ГГц','FontSize',20,...
%        'FontWeight','bold')
% subplot(3,1,2)
% histogram(imp(3,:)*1e6,'Normalization','probability')
% xlabel('\deltat,мкс','FontSize',20,...
```

```
%        'FontWeight','bold')
% subplot(3,1,3)
% histogram(imp(4,:)*1e6,'Normalization','probability')
% xlabel('T,мкс','FontSize',20,...
%        'FontWeight','bold')
% grid on


% Другой способ нормировки (z-стандартизация).

mean_z = mean(imp,2);
std_z = [std(imp(1,:)) std(imp(2,:)) std(imp(3,:)) std(imp(4,:)) ]';
imp_norm = (imp - mean_z)./std_z;


imp_norm = imp_norm';


%imp_norm = imp_norm(:,2:3);
imp_norm = imp_norm(:,2:4); %с учетом периода.



% ------------------------------------------------------------------------
% Проход окном.
answer_DBSCAN = struct();

% Окно в два паттерна.
%vector_N = [N_signal1 N_signal2];

% Окно в один паттерн.
vector_N = [N_signal1];

for p=1:length(vector_N)
    N_signal = vector_N(p);
    IMP = nan(1,N_signal*length(imp_norm(1,:)));
    for i = 1:(N-N_signal+1)
        iii = [];
        for j = 1:N_signal
            iii = [iii imp_norm((i-1) + j,:)];
        end
        IMP(i,:) = iii;
    end
    % ------------------------------------------------------------------------



    % ------------------------------------------------------------------------
    % Предобработка перед кластеризацией (удаление начального значения периода в
    % строках


    IMP(:,3) = [];

    %$crutch$
    minpts = min(vector_k); %число соседей (1. берет равный или больше числа feature+1)
(2. число паттернов)
    kD = pdist2(IMP,IMP,'euc','Smallest',minpts); % The minpts smallest pairwise
distances

    kD(minpts+1,:) = sort(kD(end,:));
```

```
    kD(minpts+2,1) = 0;

    for z=2:length(kD)
        kD(minpts+2,z) = kD(minpts+1,z)-kD(minpts+1,z-1);
    end
    % --------------------------------------------------------------------



    % --------------------------------------------------------------------

    % Графики
    %     figure(1)
    %     subplot(2,1,1)
    %     plot(kD(end-2,:))
    %     ylabel('Euc','FontSize',20,...
    %            'FontWeight','bold')
    %     xlabel('номер паттерна,k','FontSize',20,...
    %            'FontWeight','bold')
    %     grid on
    %     subplot(2,1,2)
    %     plot(kD(end-2,:),'.','MarkerSize',10)
    %     ylabel('Euc','FontSize',20,...
    %            'FontWeight','bold')
    %     xlabel('номер паттерна,k','FontSize',20,...
    %            'FontWeight','bold')
    %     title('Взаимные расстояния до минимального числа соседей в кластере')
    %     set(gca, 'FontSize', 20)
    %     grid on

    % Графики
%     figure(2)
%     subplot(2,1,1)
%     plot(kD(end-1,:))
%     ylabel('Euc','FontSize',20,...
%            'FontWeight','bold')
%     xlabel('номер паттерна,k','FontSize',20,...
%            'FontWeight','bold')
%     title('Отсортированные взаимные расстояния до минимального числа соседей в
кластере')
%     grid on
%     set(gca, 'FontSize', 20)
%     subplot(2,1,2)
%     plot(kD(end,:))
%     ylabel("Euc'",'FontSize',20,...
%            'FontWeight','bold')
%     xlabel('номер паттерна,k','FontSize',20,...
%            'FontWeight','bold')
%     title('Производная верхнего графика')
%     grid on
%     set(gca, 'FontSize', 20)

    [val,idx] = max(kD(end,1:round(end/2)));

    %     epsilon = 0.01; % Равный колену графика (чем меньше espsilon, тем больше
вероятность появляения нескольких кластеров или выбросов)
    %     epsilon = max(kD(end,1:(end-1)))/2; % end-1, так как end точно не подходит

    % TODO добавил / 5
    epsilon = kD(end-1,idx)/5; %$crutch$
```

```
        if ll == 17
            aaaaaaa = 5;
            ffff = aaaaaaa;
        end

    labels = dbscan(IMP,epsilon,minpts); %класстеризуем через DBSCAN

    % disp(['Число кластеров - ' num2str(max(labels))])
    curr_str = ['=== Номер реализации -- ', num2str(ll), ' ===='];
    fprintf(fid,'%s\n',curr_str);
    curr_str = 'Кластеризация DBSCAN:';
    fprintf(fid,'%s\n',curr_str);
    for v=1:max(labels)
        % disp(['Позиции ' num2str(v) '-ого кластера - ' num2str(find(labels==v)')])
        evalc(['answer_DBSCAN.claster', num2str(p), num2str(v), ' = find(labels==v)']);
        curr_str = ['Позиции ' num2str(v) '-ого кластера - '
num2str(find(labels==v)')];
        fprintf(fid,'%s\n',curr_str);
    end
end
% ------------------------------------------------------------------------



% ------------------------------------------------------------------------
% Сопоставление паттернов со сдвигами в отдельные массивы.
fieldnames_ans = fieldnames(answer_DBSCAN);
data_set = [];
for v=1:length(fieldnames(answer_DBSCAN))
        work_field_name = fieldnames_ans(v);
        work_field=eval(strcat('answer_DBSCAN.',work_field_name{1,1}));
        data_set = [data_set work_field'];
end

data_set = sort(data_set);

post_proc1 = struct();
temp_v = 0;
index=1;
index_internal = 2;
temp_v(1) = data_set(1);
for i=2:length(data_set)

    left = data_set(i-1);
    right = data_set(i);

    if left==right-1
      temp_v(index_internal) = right;
      index_internal = index_internal + 1;
      if data_set(i)==data_set(end)
           evalc(['post_proc1.index', num2str(index), ' = temp_v']);
      end
    else
        evalc(['post_proc1.index', num2str(index), ' = temp_v']);
        index = index + 1;
        index_internal = 2;
        temp_v = 0;
        temp_v(1) = data_set(i);
        if data_set(i)==data_set(end)
           evalc(['post_proc1.index', num2str(index), ' = temp_v']);
```

```
        end
    end
end
% ------------------------------------------------------------------------

if ll==17
    aaa=5;
end


% ------------------------------------------------------------------------
% Новая пост-обработка.
new_post_proc = struct();
threshold_freq = 2 * 20*1e9 * 0.001 * 6; %$crutch$
threshold_dur = 2 * 0.1*1e-6; %$crutch$

fieldnames_ans = fieldnames(post_proc1);
count = 1;
for i=1:length(fieldnames(post_proc1))
    temp_post_proc = struct();
    work_field_name = fieldnames_ans(i);
    work_field = eval(strcat('post_proc1.',work_field_name{1,1}));

    if size(work_field, 2) ~= 1
        for ii = 1:size(work_field, 2)
           flag = true;
           fieldnames_ans_in_temp_post_proc = fieldnames(temp_post_proc);
           for iii = 1:numel(fieldnames_ans_in_temp_post_proc)
               work_field_name_new = fieldnames_ans_in_temp_post_proc(iii);
               left_ind = eval(strcat('temp_post_proc.',work_field_name_new{1,1}));
               right_ind = work_field(ii);

               left_freq = mean(imp(2, left_ind));
               right_freq = imp(2, right_ind);

               left_dur = mean(imp(3, left_ind));
               right_dur = imp(3, right_ind);

               if i == 4
                ffffff = 5;
               end

               if abs(left_freq - right_freq) < threshold_freq &&  abs(left_dur -
right_dur) < threshold_dur
                   str = strcat('temp_post_proc.', work_field_name_new{1,1});
                   temp = [left_ind, right_ind];
                   evalc([str ' = temp']);
                   flag = false;
                   break;
               end


           end
           if flag
               temp_v = work_field(ii);
               evalc(['temp_post_proc.index', num2str(count), ' = temp_v']);
               count = count + 1;
           end

        end
```

```
    else
        temp_v = work_field(1);
        evalc(['temp_post_proc.index', num2str(count), ' = temp_v']);
        count = count + 1;
    end


    fieldnames_ans_in_new_post_proc = fieldnames(temp_post_proc);
    for ii=1:length(fieldnames_ans_in_new_post_proc)
        work_field_name_temp_post_proc = fieldnames_ans_in_new_post_proc(ii);
        work_field_temp_post_proc =
eval(strcat('temp_post_proc.',work_field_name_temp_post_proc{1,1}));
        evalc(['new_post_proc.', work_field_name_temp_post_proc{1,1}, ' =
work_field_temp_post_proc']);

    end
end

post_proc1 = new_post_proc;
% -------------------------------------------------------------------------




%$crutch$ сортировка структуры по длине каждого поля
% -------------------------------------------------------------------------
post_proc2 = struct();
massiv_lengths = nan(length(fieldnames(post_proc1)),3);
fieldnames_ans = fieldnames(post_proc1);
for i=1:length(fieldnames(post_proc1))
    work_field_name = fieldnames_ans(i);
    work_field = eval(strcat('post_proc1.',work_field_name{1,1}));
    temp_var = length(work_field);
    massiv_lengths(i,1) = temp_var;
end

[sort_seq,ss_idx] = sort(massiv_lengths(:,1));
massiv_lengths(:,2) = sort_seq;
massiv_lengths(:,3) = ss_idx;

j=1;
% for i=drange(massiv_lengths(:,3))
for i=1:length(fieldnames(post_proc1))
    num = massiv_lengths(i,3);
    work_field_name = fieldnames_ans(num);
    work_field = eval(strcat('post_proc1.',work_field_name{1,1}));
    evalc(['post_proc2.index', num2str(j), ' = work_field']);
    j=j+1;
end
% -------------------------------------------------------------------------




% -------------------------------------------------------------------------
% Избавление от сдвигов.

post_proc3 = struct();


if ll==17
```

```
    aaa=5;
end

fieldnames_ans = fieldnames(post_proc2);
for v=1:length(fieldnames(post_proc2))
        work_field_name = fieldnames_ans(v);
        work_field = eval(strcat('post_proc2.',work_field_name{1,1}));
        len = work_field(length(work_field))-work_field(1);

        if isfield(post_proc3, ['index' num2str(len-1)]) %$crutch$
            temp=eval(strcat('post_proc3.',['index' num2str(len-1)]));
            temp = [temp work_field(1)];
            evalc(['post_proc3.index', num2str(len-1), ' = temp']);

        elseif isfield(post_proc3, ['index' num2str(len+1)]) %$crutch$
            temp=eval(strcat('post_proc3.',['index' num2str(len+1)]));
            temp = [temp work_field(1)];
            evalc(['post_proc3.index', num2str(len+1), ' = temp']);

        elseif ~isfield(post_proc3, ['index' num2str(len)])
            evalc(['post_proc3.index', num2str(len), ' = work_field(1)']);

        else
            temp = eval(strcat('post_proc3.',['index' num2str(len)]));
            temp = [temp work_field(1)];
            evalc(['post_proc3.index', num2str(len), ' = temp']);
        end
end
% ------------------------------------------------------------------------


% ------------------------------------------------------------------------
% Защита от паттернов одинаковой длинны.
% Откоментить, если не нужен нижележащий блок.
% answer_postproc = post_proc3;


% ------------------------------------------------------------------------
% Закоментить этот блок, если он не нужен.
count = 1;
answer_postproc = struct();

% TODO выбираем threshold.
threshold_freq = 2 * 20*1e9 * 0.001 * 6; %$crutch$
threshold_dur = 2 * 0.1*1e-6;

fieldnames_ans = fieldnames(post_proc3);
for v=1:length(fieldnames_ans)
    time_answer = struct();
    work_field_name = fieldnames_ans(v);

    work_field = eval(strcat('post_proc3.',work_field_name{1,1}));
    c = 1;
    for k=1:size(work_field,2)
        new_i = work_field(k);
%     for new_i = drange(work_field)
        flag = true;
        if ~isempty(time_answer)
            fieldnames_ans_1 = fieldnames(time_answer);
            for i_in_time_answer = 1:length(fieldnames_ans_1)
```

```
                work_field_in_time_answer = eval(strcat('time_answer.',
fieldnames_ans_1{i_in_time_answer}));

                left_freq = imp(2, new_i);
                left_dur = imp(3, new_i);

                index = strsplit(work_field_name{1,1}, 'x');
                index = str2double(index(2));

                pattern_freq_temp = imp(2,
work_field_in_time_answer(1):(work_field_in_time_answer(1) + vector_N(1) + index-1));
                pattern_dur_temp = imp(3,
work_field_in_time_answer(1):(work_field_in_time_answer(1) + vector_N(1) + index-1));

                right_freq = mean(pattern_freq_temp);
                right_dur = mean(pattern_dur_temp);

                if abs(right_freq - left_freq) <= threshold_freq && abs(right_dur -
left_dur) <= threshold_dur
                    temp = [work_field_in_time_answer new_i];
                    evalc([strcat('time_answer.', fieldnames_ans_1{i_in_time_answer}) '
= temp']);
                    flag = false;
                    break;
                end
            end
        end
        if flag
            str = strcat('time_answer.', work_field_name{1,1}, '_', num2str(count));
            temp = new_i;
            evalc([str ' = temp']);
            count = count + 1;
        end
        c = c + 1;
    end

    fieldnames_ans_1 = fieldnames(time_answer);
    count = 1;
    for i_in_time_answer = 1:length(fieldnames_ans_1)
        work_field_name = fieldnames_ans_1(i_in_time_answer);
        work_field = eval(strcat('time_answer.',work_field_name{1,1}));
        str = strcat('answer_postproc.', work_field_name{1,1});
        evalc([str ' = work_field']);
    end
end
% -------------------------------------------------------------------------




% -------------------------------------------------------------------------
    % Обратная нормировка.
    IMP_out(:,1) = IMP(:,1).* std_z(2)' +  mean_z(2)';
    IMP_out(:,2) = (IMP(:,2).* std_z(3)' +  mean_z(3)').*1e6 ;
    for j=1:(length(IMP(1,:))-2)
        if mod(j,3)==1 %частота
            IMP_out(:,2+j) = IMP(:,2+j).* std_z(2)' + mean_z(2)';
        end
        if mod(j,3)==2 %длительность импульса
```

```
            IMP_out(:,2+j) = ( IMP(:,2+j).* std_z(3)' + mean_z(3)' ).*1e6; %переводим в
мкс
        end
        if mod(j,3)==0 %период
            IMP_out(:,2+j) = ( IMP(:,2+j).* std_z(4)' + mean_z(4)' ).*1e6; %переводим в
мкс
        end
    end
% ------------------------------------------------------------------------



% ------------------------------------------------------------------------
    % Вывод результатов в виде наглядных паттернов в кластерах
    total_proc = struct();
    imp=imp';
    for z =1:numel(fieldnames(answer_postproc))
        fieldnames_ans = fieldnames(answer_postproc);
        work_field_name = fieldnames_ans(z);
        index = strsplit(work_field_name{1,1}, 'x');
        index = strsplit(index{1,2}, '_');
        index = str2double(index(1));
        if index == 0 %паттерн из 3 импульсов (8 параметров)
            work_field=eval(strcat('answer_postproc.',work_field_name{1,1}));
            j=1;
            input_data_new=nan(length(work_field),length(IMP_out(1,:))+2);
            for i=1:length(work_field)
                input_data_new(j,1) = work_field(i);
                input_data_new(j,2) = imp(work_field(i),1);
                input_data_new(j,3:end) = IMP_out(work_field(i),:);
                j=j+1;
            end
            evalc(['total_proc.claster', num2str(z) ' = input_data_new']);
        else %больше 3 импульсов
            work_field=eval(strcat('answer_postproc.',work_field_name{1,1}));
            j=1;
            input_data_new=nan(length(work_field) ,length(IMP_out(1,:)) +
index*length(imp_norm(1,:))+2);
            for i=1:length(work_field)
                input_data_new(j,1) = work_field(i);
                input_data_new(j,2) = imp(work_field(i),1);
                input_data_new(j,3:length(IMP_out(1,:))+2) = IMP_out(work_field(i),:);
                result =
IMP_out((work_field(i)+1:work_field(i)+index),(length(IMP_out(1,:))-
length(imp_norm(1,:))+1:length(IMP_out(1,:))));
                result = reshape(result.',1,[]);
                input_data_new(j,length(IMP_out(1,:))+1+1+1:end) = result;
                j=j+1;
            end
            evalc(['total_proc.claster', num2str(z) ' = input_data_new']);
        end
    end
    imp=imp';
%------------------------------------------------------------------------
```

```
% ------------------------------------------------------------------------
% Формирование ответа
total = struct();
answer_postproc = struct();
fieldnames_ans = fieldnames(total_proc);

count = 1;
for v=1:length(fieldnames_ans)
    work_field_name = fieldnames_ans(v);
    work_field = eval(strcat('total_proc.',work_field_name{1,1}));
    work_field = sortrows(work_field, 1);


    %$crutch$
    if size(work_field, 1) >= 4
        work_field_answer_postproc = work_field(:, 1);
        work_field_total = work_field(:, 2:end);
        evalc(['answer_postproc.claster', num2str(count) ' =
work_field_answer_postproc']);
        evalc(['total.claster', num2str(count) ' = work_field_total']);
        count = count + 1;
    end
end
%------------------------------------------------------------------------



fieldnames_ans = fieldnames(answer_postproc);

count = 1;
for v=1:length(fieldnames_ans)
    work_field_name = fieldnames_ans(v);
    work_field = eval(strcat('answer_postproc.',work_field_name{1,1}));

    work_field_answer_postproc = work_field(:, 1);
    work_field_total = work_field(:, 2:end);
    evalc(['answer_postproc.claster', num2str(count) ' = work_field_answer_postproc']);
    evalc(['total.claster', num2str(count) ' = work_field_total']);
    count = count + 1;
end




%------------------------------------------------------------------------
% Вывод позиций обработанных кластеров

fprintf(fid,'%s\n','');

% Показываем ответ алгоритма.
total_answer_mas = 0;
fieldnames_ans = fieldnames(answer_postproc);
disp(['Номер реализации - ' num2str(ll)])
disp('Результат с выхода алгоритма(DBSCAN+постобработка):  ')
curr_str1 = 'Результат с выхода алгоритма(DBSCAN+постобработка): ';
disp(['Число кластеров - ' num2str(length(fieldnames_ans))])
```

```
curr_str2 = ['Число кластеров - ' num2str(length(fieldnames_ans))];
fprintf(fid,'%s\n',curr_str1);
fprintf(fid,'%s\n',curr_str2);
    for v=1:length(fieldnames(answer_postproc))
        work_field_name = fieldnames_ans(v);
        work_field=eval(strcat('answer_postproc.',work_field_name{1,1}))';
        total_answer_mas(v,1:length(work_field)) = work_field;
        disp(['Позиции ' num2str(v) '-ого кластера - ' num2str(work_field)])
        curr_str = ['Позиции ' num2str(v) '-ого кластера - ' num2str(work_field)];
        fprintf(fid,'%s\n',curr_str);
    end

% Показывает истинный ответ.
true_positions_mas = 0;
fieldnames_ans = fieldnames(true_positions);
disp('Истиное расположение паттернов в кластерах:   ')
curr_str1 = 'Истиное расположение паттернов в кластерах:   ';
disp(['Число кластеров - ' num2str(length(fieldnames_ans))])
curr_str2 = ['Число кластеров - ' num2str(length(fieldnames_ans))];
fprintf(fid,'%s\n',curr_str1);
fprintf(fid,'%s\n',curr_str2);

    for v=1:length(fieldnames(true_positions))
        work_field_name = fieldnames_ans(v);
        work_field=eval(strcat('true_positions.',work_field_name{1,1}));
        true_positions_mas(v,1:length(work_field)) = work_field;
        disp(['Позиции ' num2str(v) '-ого кластера - ' num2str(work_field') ])
        curr_str = ['Позиции ' num2str(v) '-ого кластера - ' num2str(work_field') ];
        fprintf(fid,'%s\n',curr_str);
    end
fprintf(fid,'%s\n','');

%-------------------------------------------------------------------------
```

```
%-------------------------------------------------------------------------
% Блок оценки точности результатов.
    fprintf("\n");
    fprintf(fid,'%s\n','');
    disp(['Реальных кластеров - ' num2str(length(true_positions_mas(:,1))) ' Алгоритм
выявил - ' num2str(length(total_answer_mas(:,1)))])
    curr_str = ['Реальных кластеров - ' num2str(length(true_positions_mas(:,1))) '
Алгоритм выявил - ' num2str(length(total_answer_mas(:,1)))];
    fprintf(fid,'%s\n',curr_str);
    stat_matrix(ll,3) = length(total_answer_mas(:,1));

    total_answer_mas = sort(reshape(total_answer_mas,1,[]));
    true_positions_mas = sort(reshape(true_positions_mas,1,[]));
    j=1;
    % Блок обработки (исключение нулей).
    for i=1:length(total_answer_mas)
        if total_answer_mas(j) == 0
           total_answer_mas(j)=[];
           if j>length(total_answer_mas)
               break
```

```
            end
        else
            j=j+1;
            if j>length(total_answer_mas)
               break
            end
        end
    end
    j=1;


    % Блок обработки (исключение нулей).
    for i=1:length(true_positions_mas)
        if true_positions_mas(j) == 0
           true_positions_mas(j)=[];
           if j>length(true_positions_mas)
               break
           end
        else
            j=j+1;
            if j>length(true_positions_mas)
               break
            end
        end
    end

    C1 = intersect(total_answer_mas,true_positions_mas);
    if length(total_answer_mas)>length(true_positions_mas)
        C2 = setdiff(total_answer_mas,true_positions_mas);
    else
        C2 = setdiff(true_positions_mas,total_answer_mas);
    end

    FA1 = 100*length(C2)/(N-max(vector_N)+1-sum(vector_k)); %False Alarm (не совсем
правильно)
    accuracy = length(C1)/length(true_positions_mas)*100;
    disp(['Точность распознавания паттернов алгоритмом на данной реализации составила -
' num2str(accuracy) '%'])
    disp(['Процент определенных ложных паттернов - ' num2str(FA1) '%'])
    curr_str1 = ['Точность распознавания паттернов алгоритмом на данной реализации
составила - ' num2str(accuracy) '%'];
    curr_str2 = ['Процент определенных ложных паттернов - ' num2str(FA1) '%'];
    fprintf(fid,'%s\n',curr_str1);
    fprintf(fid,'%s\n',curr_str2);

    stat_matrix(ll,1) = accuracy;
    stat_matrix(ll,2) = FA1;
    fprintf("\n");
    fprintf(fid,'%s\n','');
    fprintf(fid,'%s\n','');
    fprintf(fid,'%s\n','');



end
%--------------------------------------------------------------------------
```

```
%-------------------------------------------------------------------------
% Общая статистика по всем реализациям.
Mean_accuracy = sum(stat_matrix(:,1))/rr;
Mean_FAl = sum(stat_matrix(:,2))/rr;
Mean_clusters = sum(stat_matrix(:,3))/rr;
fprintf("\n");
fprintf(fid,'%s\n','');
disp(['Результат кластеризации на ' num2str(rr) ' случаях, средняя точность определения
паттернов составила - ' num2str(Mean_accuracy) '%'])
disp(['Средний процент определенных ложных паттернов составил - ' num2str(Mean_FAl)
'%'])
disp(['Среднее число выявляемых кластеров составило - ' num2str(Mean_clusters)])
curr_str1 = ['Результат кластеризации на ' num2str(rr) ' случаях, средняя точность
определения паттернов составила - ' num2str(Mean_accuracy) '%'];
curr_str2 = ['Средний процент определенных ложных паттернов составил - '
num2str(Mean_FAl) '%'];
curr_str3 = ['Среднее число выявляемых кластеров составило - ' num2str(Mean_clusters)];
fprintf(fid,'%s\n',curr_str1);
fprintf(fid,'%s\n',curr_str2);
fprintf(fid,'%s\n',curr_str3);
%-------------------------------------------------------------------------



% Закрытие файлов.
fclose(fid);
fclose all;


% Конец замеров тестов
toc


%-------------------------------------------------------------------------
% Функция формирования имплуьсов
function [ imp ] = make_impulse( freq, dur, T_min, prev_T )

    delta = T_min * randi(10);
    ToA = prev_T + delta; % Формирование времени прихода импульса.

    freq1 = freq(randi(length(freq))); % Формирование несущей импульса.
    f = freq1 + randi([0 1])*(-1)^randi([0 1]) * 0.001 * randi(6) * freq1;

    dur1 = dur(randi(length(dur))); % Формирование длительности импульса.
    d = dur1 + 0*randi([0 1])*(-1)^randi([0 1]) * 0.1 * randi(6) * dur1;
    imp = [ToA; f; d; delta];
end
```