Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Владимирский государственный университет
имени Александра Григорьевича и Николая Григорьевича Столетовых»
(ВлГУ)

На правах рукописи

Корнеева Наталья Николаевна

ИССЛЕДОВАНИЕ И РАЗРАБОТКА АЛГОРИТМОВ ДИАГНОСТИКИ КОДИРОВАННЫХ ЦИФРОВЫХ СИГНАЛОВ

Специальность: 05.12.04 – Радиотехника, в том числе системы и устройства телевидения

Диссертация
на соискание ученой степени кандидата технических наук

Научный руководитель:
Доктор технических наук, профессор Никитин Олег Рафаилович

Владимир, 2016

Оглавление

Введение

Актуальность темы исследования. В настоящее время в подавляющем большинстве цифровых систем передачи информации применяются различные виды помехоустойчивого кодирования. Это обусловлено различными факторами, в частности, ростом общего количества радиоизлучающих средств, создающих фон разнообразных внешних помех, а также тем, что требования на качество передачи информации имеют тенденцию к повышению. В такой ситуации использование помехоустойчивых кодов в большинстве случаев достаточно эффективно решает соответствующие проблемы.

Основы помехоустойчивого кодирования при передаче информации были заложены работами К. Шеннона в 1948 году. Им было показано, что если скорость передачи информации меньше пропускной способности канала, то в принципе могут быть подобраны код и способ его декодирования, которые обеспечивают восстановление поврежденной последовательности со сколь угодно большой точностью. С тех пор было разработано и разрабатывается большое число разнообразных кодов, обладающих различными свойствами по восстановлению исходной переданной информационной последовательности, а также скоростью работы, объемом необходимых вычислений, и т.д.

В нашей стране среди специалистов в области применения помехоустойчивого кодирования известны работы таких ученых, как Б.А. Котельников, Э.Л. Блох, А.Н. Колмогоров, В.В, Зяблов, Л.М. Финк, В.В. Золотарев, Г.Н. Овечкин, О.Р Никитин., А.Г.Самойлов, П.А.Полушин, и др. Среди выдающихся зарубежных специалистов можно выделить Дж. Мэсси, А. Витерби, Р. Галлагера, Р. Блейхута, У. Питерсона, Э. Берлекампа, Д. Форни и др.

При осуществлении на приемной стороне процедуры декодирования предполагается, что все параметры применяемого кода и структура кодера заранее полностью известны. На основе этого осуществляется процедура соответствующего декодирования. В то же время в реальной обстановке условия

работы системы передачи информации могут быть достаточно разнообразны. Возможны ситуации, когда параметры используемого кода, необходимые для декодирования, известны не полностью, либо неизвестны вообще, например, если происходит их быстрая смена, а передача информации в приемник об этом задерживается. Информация о структуре кодера важна для систем радиомониторинга и систем адаптивного кодирования. При смене канала передачи или системы передачи требуемая для декодирования информация может отсутствовать. В случаях радиопротиводействия такую информацию получить проблематично в принципе. В таких ситуациях восстанавливающие свойства помехоустойчивых кодов резко ухудшаются, либо прием кодированных сигналов становится принципиально невозможным.

В то же время в передатчике в процессе кодировании вносятся определенные связи между формируемыми кодовыми символами. Анализируя эти связи, можно во многих случаях получить отсутствующую информацию о параметрах используемого кода. Это позволить восстановить исправляющую способность кода и обеспечить требуемый уровень помехоустойчивости, что обуславливает актуальность решения подобных диагностических задач. Задачи особенно актуальны для наиболее часто используемых сверточных и блоковых кодов.

Объект исследования. Методы помехоустойчивого кодирования цифровых сигналов.

Предмет исследования. Алгоритмы диагностики параметров сверточных и блоковых двоичных кодов на основе принимаемых цифровых кодовых последовательностей.

Цель и задачи исследования. Повышение эффективности использования помехоустойчивого кодирования за счет предварительной диагностирования кодовых последовательностей путем анализа принимаемых цифровых сигналов. И восстановления утраченной информации.

Для достижения этой цели необходимо решить следующие **задачи.**

1. Провести обзор различных методов получения сверточных и блоковых кодов, включая модифицированные и перфорированные коды, и проанализировать особенности, которые возможно использовать для целей диагностики.

2. Разработать алгоритмы диагностики сверточных кодов, включая различные их варианты, в том числе «быстрые» алгоритмы диагностики.

3. Разработать алгоритмы диагностики различных вариантов блоковых кодов, включая циклические коды.

4. Разработать диагностические алгоритмы для различных видов модификации кодов (укороченных кодов, расширенных кодов, перфорированных кодов).

5. Создать программные средства, с помощью которых исследовать характеристики предложенных алгоритмов.

Методы исследования. В диссертационной работе используется теория вероятностей и математической статистики, теория кодирования, теория матриц, математическое и компьютерное моделирование.

Научная новизна. В рамках диссертационной работы были получены следующие научные результаты.

1. Проведен анализ методов формирования сверточных и блоковых кодов и выделены особенности, которые возможно использовать для их диагностики на основе принимаемых кодовых последовательностей.

2. Предложены алгоритмы диагностики сверточных кодов, включая «быстрые» алгоритмы диагностики.

3. Разработаны алгоритмы диагностики блоковых кодов, позволяющие определять параметры циклических и линейных блоковых кодов.

4. Предложены алгоритмы диагностики модифицированных кодов, включая укороченные, расширенные и перфорированные коды.

Практическая значимость.

1. Предложенные алгоритмы обработки цифровых сигналов позволяют за счет восстановления исправляющей способности кодов увеличить помехоустойчивость приема.

2. Разработанные алгоритмы дают возможность обеспечить вероятность неправильной диагностики при вероятности битовой ошибки менее 10^{-3} (при малых уровнях шума) до величины не хуже $10^{-4} \div 10^{-6}$ за 20-30 циклов анализа. При повышении вероятности ошибки для обеспечения такого же результата длительность анализа увеличивается в 2-4 раза.

3. Применение «быстрых» алгоритмов анализа сокращает время диагностики в 5-10 раз.

4. Разработаны и исследованы программные средства диагностики сверточных и блоковых кодов и их модификаций, дающие возможность определить параметры кодеров путем анализа принимаемой кодовой последовательности.

На защиту выносятся:

1. Алгоритмы определения кодового ограничения и структуры сверточных кодов, включая «быстрые» алгоритмы диагностики.

2. Алгоритмы диагностики линейных блоковых и циклических кодов.

3. Алгоритм диагностики укороченных и расширенных кодов и алгоритмы определения структуры и параметров перфорированных кодов.

Внедрение научных результатов диссертационной работы проведено в учебный процесс на кафедре радиотехники и радиосистем Владимирского государственного университета, а также в ОАО «Владимирский завод «Электроприбор», г. Владимир, о чем получены акты внедрения.

Апробация работы проведена в форме научных докладов по основным результатам работы и дискуссий, которые проходили на следующих научных конференциях:

- 11-я МНТК «Перспективные технологии в средствах передачи информации – ПТСПИ-2015»;

- 12-я МНТК «Физика и радиоэлектроника в медицине и экологии (ФРЭМЭ 2016);
-2-я МНТК «Наука и образование: проблемы и стратегии развития»-Уфа,15-16 ноября2016г.

Личный вклад автора. Все основные результаты диссертации получены автором лично.

Публикации. По теме диссертации опубликовано 12печатных научных работ, в том числе3 статьи в изданиях, входящих в список ВАК РФ; 5материалов докладов на конференциях различного уровня, включая международные; 3 свидетельства о государственной регистрации программ для ЭВМ;1 положительное решение по заявке на изобретение.

Структура и объем диссертации. Диссертационная работа состоит из введения, четырех глав, заключения. списка литературы и приложений. Общий объем диссертационной работы с приложениями составляет 191страниц, в том числе 180 страниц основного текста. Работа содержит 115 рисунков, 1 таблицу, список использованной литературы содержит 105 наименований.

1. Обзор методов кодирования и характеристик передачи цифровых сигналов

В данной главе ставится задача диагностики цифровых кодированных последовательностей. Для этого производится обзор основных методов кодирования и параметров помехоустойчивости передачи. Анализируются характеристики процедур диагностики и факторы, влияющие на них.

1.1. Предпосылки диагностики кодированных цифровых последовательностей

Значительное количество эксплуатируемых и проектируемых систем передачи информации в настоящее время использует цифровые сигналы [1-20]. Это в определенной степени облегчает построение приемопередающей аппаратуры и дает возможности повышения качества передачи. Если раньше факторов, приводящих к сбоям передачи и появлению ошибок, было сравнительно немного, то современный рост числа радиоизлучающих средств практически во всех диапазонах ведет к возрастанию количества и разнообразия помех от внешних источников. Кроме этого, увеличение объема и скоростей передачи информации приводит к необходимости использования все более широкополосных каналов, в которых проявляются нестационарные свойства, вызывающие искажения сигналов.

Традиционные методы и средства повышения помехоустойчивости, такие как совершенствование антенно-фидерного тракта, увеличение излучаемой мощности, снижение собственного шума приемника и т.д. зачастую оказываются неприменимыми или технически и экономически невыгодными [21-24]. В то же время за счет искусственно вносимой избыточности ресурса системы удается достаточно эффективно повышать качественные показатели систем передачи информации. Определенное расширение используемой полосы частот или снижение скорости передачи, которое используется различными кодами, дает

возможность значительно увеличивать помехоустойчивость и достоверность передачи, что и обуславливает их широкое использование [3,4,7,23,25].

К настоящему времени известны много методов кодирования, имеющих различные возможности по исправлению ошибок и требующие при практической реализации разного уровня усложнения аппаратуры приемников и передатчиков. Однако предполагается, что вид и параметры кодирования, используемые в передатчике, на приемной стороне полностью известны. На их основе осуществляется соответствующая процедура декодирования [3-7, 33-56].

В то же время условия возможного функционирования системы передачи могут быть очень разнообразными. Нетрудно представить ситуации, когда требуемые для декодирования параметры используемых кодов известны не полностью или неизвестны вообще, например, в случае быстрой смены передатчиком характеристик используемого кода, передача информации о которых в приемник задерживается. Также при смене канала передачи или системы передачи требуемая информация может быть утеряна. В некоторых случаях, например в рамках радиопротиводействия ([57-62]), такую информацию заранее получить невозможно в принципе.

В этом случае декодирование либо невозможно в принципе, либо возможно только частичное декодирование. Последствия зависят от уровня остаточной информации и от вида используемого кода. Например, в случае применения систематических кодов исходную передаваемую информационную последовательность часто можно восстановить, но исправление ошибок невозможно. При использовании несистематического кодирования восстановление передаваемой информационной последовательности (пусть с низкими показателями помехоустойчивости) невозможно, при этом передаваемая информация может потеряться безвозвратно.

Все это может значительно ухудшить в целом показатели помехоустойчивости передачи, снижая их существенно ниже допустимых норм.

В то же время процесс кодирования вносит определенные связи между передаваемыми закодированными символами. При этом последовательность

ранее независимых символов становится структурированной. Анализируя эти связи можно восстановить информацию об используемых параметрах кодера, которая раньше отсутствовала [63-66]. В результате возникает возможность декодирования принятой последовательности даже без знания параметров кодера, и при этом восстановить возможность исправления ошибок и сохранения требуемого уровня помехоустойчивости и качества передачи.

Требуемая для этого процедура диагностики зависит от применяемого вида кодера. Для разработки возможных методов диагностики требуется рассмотрение основных методов кодирования, которое проведено в последующих параграфах.

1.2. Обзор основных методов кодирования

Код представляет собой определенную форму представления сообщения, которая может не зависеть от физической сути сигналов, хотя практически она обычно связана с их физическими параметрами [3, 4, 26, 33-56]. Многие сообщения исходно обладают внутренними корреляционными связями, позволяющими устранять часть возможных ошибок, что, например, делает понятной даже сильно зашумленную речь. Однако в общем случае символы исходного информационного сигнала следует полагать взаимно независимыми, т.к. такая последовательность несет наибольший объем информации. Тем более, если исходные информационные связи выражены слабо, или неизвестны, их затруднительно использовать для повышения помехоустойчивости. В этом случае с помощью кодирования различными методами вводят искусственные связи за счет увеличения числа символов. Степень возникающей избыточности определяет исправляющие свойства кода. (Известны также коды без избыточности, которые используют заранее известный ограниченный объем предаваемых информационных сообщений, однако они, как правило, узкоспециализированы и имеют ограниченное применение).

В настоящее время известно большое количество различных кодов. Основные распространенные виды условно представлены на схеме, приведенной

на рисунке 1.1. Важным для диагностики различием выступает разделение на систематические и несистематические коды. В систематических кодах можно в кодовой последовательности выделить исходную информационную последовательность символов. Она может быть непрерывной, либо разделенной на фрагменты. Избыточные новые символы по определенному принципу просто добавляются к исходной последовательности.

В несистематических кодах формируемая в кодере последовательность не содержит неизмененную исходную последовательность. Все символы кодовой последовательности получаются исходя из совокупности исходных символов с использованием определенных функций или правил, определяющих вид и параметры кода. В двоичных кодах каждый символ содержит один информационный бит, символы недвоичных кодов содержат несколько бит. Различие фактически состоит лишь в усложнении аппаратуры кодирования.

Линейные коды отличаются от нелинейных замкнутостью получаемого множества кодовых слов относительно оператора, с помощью которого реализуется кодирование. Оператор в этом случае является линейным, а кодовые слова можно описать в виде векторов некоторого пространства. Линейность кода упрощает его реализацию и при большой длине кодовых слов практически применяются только линейные коды. Линейные коды образуют обширные классы. Однако нелинейные коды в среднем обладают лучшими характеристиками.

Различные коды удобно разделять на относительно простые (немодифицированые) и на модифицированные. Работа посвящена диагностике кодов первого вида, т.к. эти коды выступают как некоторая основа, на базе которой строятся коды второго вида.

Рассматриваемые коды в зависимости от основного правила их получения делятся на сверточные и блоковые. Несмотря на их существенные различия, после определенной модификации каждой группе можно придать некоторые свойства другой группы. Подробнее рассмотрим принципы формирования и свойства каждой из этих групп. Поскольку для целей диагностики имеют основное

значение операции процедуры кодирования, то основное внимание будет уделено именно им.

Рисунок 1.1.

Сверточные коды. Сверточные коды впервые введены в [67] и в настоящее время нашли широкое применение. Они иногда называются непрерывными

кодами, потому что используют непрерывную (последовательную) обработку символов. Кроме того они являются линейными кодами. Кодер обладает памятью в том смысле, что каждый выходной кодовый символ зависит не только от текущего входного информационного символа, но и от нескольких предыдущих входных символов [4,18].

Избыточность кода, определяется соотношением количества кодовых символов, требующихся для передачи определенного числа информационных символов. Если для передачи k информационных символов требуется n кодовых символов, то кодовая скорость равна $R=k/n$. Обобщенная структурная схема сверточного кодера приведена на рисунке 1.2.

На его вход поступает последовательность информационных символов $\mathbf{m}=m_1,m_2,\ldots$ На выходе образуется последовательность кодированных символов $\mathbf{u}=u_1,u_2,\ldots$ Кодер состоит из k сдвиговых регистров, содержащих по K–1 ячеек. Входной коммутатор (Комм.1) распределяет символы входной последовательности по последовательным входам регистров. При каждом такте содержимое регистров сдвигается в соседнюю ячейку. Сигналы с параллельных выходов регистров подаются на входы n многовходовых сумматоров $n>k$. В сумматорах над входными сигналами производится логическая операция сложения по модулю 2 («исключающее или»). Входы каждого сумматора подключены к своему определенному набору ячеек сдвиговых регистров. Эти наборы различаются у всех сумматоров и определяют структуру конкретного используемого кодера.

Выходная кодовая последовательность образуется последовательным поочередным подключением с помощью коммутатора (Комм.2) выходных сигналов сумматоров на общий выход кодера. Таким образом, при поступлении k исходных информационных символов образуется n кодовых символов. Варьируя параметры k и n, можно регулировать кодовую скорость R. Однако чаще используются коды со скоростью $1/n$, а скорость регулируется применением перфорации (выкалывания) [4,26], впервые описанного в [68].

В формировании каждого кодового символа в данный момент времени участвует входной символ и K–1 предыдущих входных символов. Таким образом кодовое ограничение K определяет память кодовой последовательности. Кодер, приведенный на рисунке 1.2, представляет собой систему с конечным откликом.

Если при k=1 пропускать через кодер исходную последовательность символов, содержащую только одну единицу, а остальные – нули, то формируемая кодовая последовательность будет представлять собой набор импульсных откликов $g_1 \div g_n$ каждого из n сумматоров. Наборы коэффициентов $g_1 \div g_n$ описываются в виде векторов **g**. Размеры векторов составляют максимум K двоичных элементов и полностью определяют структуру кодера. Каждый разряд соответствующего двоичного кода соответствует одному отводу регистра, и единица в нем устанавливает факт наличия связи данного разряда регистра с сумматором. Двоичные последовательности **g** (кодовых генераторов) обычно записывают в десятичной форме или разделяют на группы по три символа и записывают в восьмеричной форме. Также используется запись **g** в виде полинома (порождающего полинома).

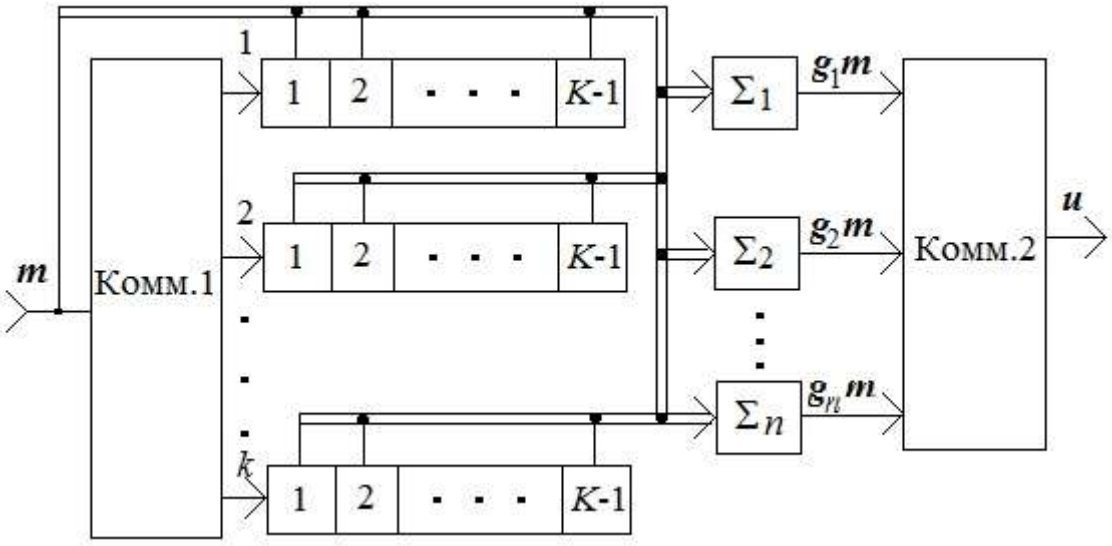

Рисунок 1.2.

Поскольку выходной сигнал каждого сумматора представляет собой свертку соответствующего порождающего полинома и фрагмента входной информационной последовательности **m**, состоящей и текущего входного символа и K–1 предыдущих входных символов, то формируемые группы по n кодовых символов, соответствующих одному входному символу, можно записать в векторной форме: $\mathbf{u}_0=u_1,\ldots,u_n$, где $u_1=\mathbf{g}_1\mathbf{m}$, $u_2=\mathbf{g}_2\mathbf{m},\ldots,u_n=\mathbf{g}_n\mathbf{m}$ (произведением обозначена свертка векторов). Формируемую кодовую последовательность также можно получить с помощью некоторой матрицы **G** (порождающей матрицы), имеющей ленточную структуру.

Исправляющие свойства сверточных кодов с разными наборами порождающих полиномов различаются. Для удобного графического исследования работы кодера и декодера эффективно использование решетчатых диаграмм. После достижения глубины анализа, равной K шагов, структура диаграммы становится постоянной. Решетчатая диаграмма используется при декодировании. Правильно декодированная кодовая последовательность должна соответствовать на каждом шаге только одному из нескольких разрешенных путей по решетке. Суммарное количество отличий принятой последовательности от разрешенных вариантов, измеренное в определенной метрике, служит указанием для восстановления передаваемого информационного сообщения и освобождения его от ошибок. Для каждого набора исходных данных (кодовой скорости, кодового ограничения) известны наиболее эффективные в смысле исправления ошибок коды, хотя могут использоваться и коды с близкой структурой, несколько уступающие им по эффективности.

Структура систематических сверточных кодеров незначительно отличается от структуры несистематических кодеров, представленной на рисунке 1.2. Отличия заключаются в том, что один из входов выходного коммутатора подключен не к выходу одного из сумматоров по модулю 2, а непосредственно ко входу кодера. В результате одна из компонент формируемого кодовой последовательности совпадает со входной информационной последовательностью, хотя ее символы следуют не один за другим, а разделены

n–1 другими кодовыми символами. Систематические сверточные коды используются редко, т.к. их характеристики в среднем хуже, чем у соответствующих несистематических кодов.

Кроме требования высокой эффективности исправления ошибок на вид полиномиальных генераторов накладываются и другие ограничения. Кодовые генераторы удобно представлять в виде полиномов вида:

$$\mathbf{g}(X)=a_0+a_1X+a_2X^2+\ldots+a_nX^n,$$

где переменная X обозначает сдвиг по времени на один такт, X^2 – сдвиг по времени на два такта, и т.д.; коэффициенты $a_0 \div a_n$ могут принимать нулевое или единичное значение. Важное ограничение на вид полиномов связано с возможностью распространения катастрофических ошибок при декодировании, когда конечное число ошибок в кодовых словах вызывает бесконечное число ошибок в декодированных данных.

В [4, 26] показано, что возможность для появления катастрофических ошибок появляется, если при разложении используемых порождающих полиномов на более простые полиномы-множители у любых двух из них в разложении будет присутствовать хотя бы один одинаковый простой полином-множитель.

Известны также рекурсивные сверточные коды, которые обычно бывают систематическими. Они отличаются тем, что характеризуются бесконечным импульсным откликом и сумматоры могут находиться не только на выходах, но и на входах регистров. Несистематический кодер и систематический сверточный кодер имеют одно и т же множество кодовых последовательностей, но с другим соответствие между информационными и кодовыми словами [26].

Блоковые коды. Первоначально рассмотрим двоичные блоковые коды. Виды известных блоковых кодов характеризуются исключительным разнообразием, связанным с особенностями кодирования и декодирования, сложностью реализации, быстродействием и помехоустойчивостью. Однако для

целей диагностики большинство методов их получения можно объединить в две связанные между собой группы. В одной из них используются порождающие матрицы, в другой группе – порождающие полиномы.

Любой блоковый код основан на том, что исходная информационная последовательность символов разбивается на группы, как правило, одинаковой длины k. По определенному правилу на основе значений информационных символов в группе вычисляется последовательность из b проверочных символов и добавляется к информационной последовательности, формируя выходной кодовый блок длиной n.

Если длина информационной части блока невелика, то наиболее простым методом кодирования является использование таблицы соответствия. Таблица соответствия содержит все варианты информационных последовательностей и соответствующие им варианты кодовых блоков. При кодировании на основе каждой новой информационной части для передачи просто выбирается нужный кодовый блок. Однако размер таблицы пропорционален 2^k, поэтому при большой длине информационных частей величина таблицы и время кодирования могут стать недопустимо большими.

В этом случае задачу можно упростить, если не хранить в памяти кодовые блоки, соответствующие поступающим информационным группам, и не отыскивать их в таблице, а каждый раз генерировать вновь. Пусть $\mathbf{m}$ – вектор, содержащий k двоичных элементов и составленный из группы исходных информационных символов. Тогда, если имеется произвольный базис из k линейно независимых двоичных векторов, тогда любой информационный вектор можно записать, как линейную комбинацию некоторых векторов этого базиса. (При составлении линейной комбинации под умножением понимается операция «и», под сложением – операция «исключающее или»). Таким образом, если имеется набор k двоичных линейно независимых векторов $\mathbf{V}_1, \mathbf{V}_2,\ldots, \mathbf{V}_k$, каждый из которых состоит из n элементов, то вектор $\mathbf{u}$, описывающий любой кодовый блок, можно записать, как: $\mathbf{u}=m_1\mathbf{V}_1+m_2\mathbf{V}_2+\ldots+m_k\mathbf{V}_k$.

Порождающая матрица **G** имеет размер $k \times n$ и представляет собой набор векторов $\mathbf{V}_1 \div \mathbf{V}_k$, как строк:

$$\mathbf{G} = \begin{Vmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \vdots \\ \mathbf{V}_k \end{Vmatrix} = \begin{Vmatrix} v_{1,1}, v_{1,2}, \ldots, v_{1,n} \\ v_{2,1}, v_{2,2}, \ldots, v_{2,n} \\ \ldots\ldots\ldots \\ v_{k,1}, v_{k,2}, \ldots, v_{k,n} \end{Vmatrix},$$

где $v_{i,j}$ – элементы с индексом j вектора $\mathbf{V}_i$.

Генерация кодового слова в матричной форме описывается уравнением **u**=**mG**. При этом при кодировании в памяти необходимо сохранять только k векторов $\mathbf{V}_1 \div \mathbf{V}_k$, а не 2^k, как при использовании таблицы соответствия.

Матрицу **G** можно представить, как состоящую из двух блоков, первый блок $\mathbf{G}_1$ размером $k \times k$, второй блок $\mathbf{G}_2$ размером $k \times (n-k)$, т.е.: $\mathbf{G} = \|\mathbf{G}_1 \vdots \mathbf{G}_2\|$. Матрица $\mathbf{G}_1$ определяет вид информационной части кодового слова, матрица $\mathbf{G}_2$ определяет вид проверочной части кодового слова. При использовании несистематического кодирования вид матрицы $\mathbf{G}_1$ может быть произвольным (при сохранении условия линейной независимости строк). При использовании систематического кодирования $\mathbf{G}_1$ является единичной матрицей, $\mathbf{G}_1$=**E**. Несистематическое кодирование обеспечивает такие же характеристики помехоустойчивости, как и систематическое, однако при использовании систематического кодирования процесс кодирования существенно упрощается и ускоряется ([26]), поэтому практически используется систематическое кодирование. К тому же порождающая матрица для несистематического кодирования легко преобразуется в матрицу для систематического кодирования умножением на обратную матрицу $\mathbf{G}_1^{-1}$.

После переобозначения части матрицы, отвечающей за формирование проверочной части кодового слова, $\mathbf{G}_2$=**P**, порождающая матрица приобретает вид: $\mathbf{G} = \|\mathbf{E} \vdots \mathbf{P}\|$. При подробном рассмотрении структуры генерируемого кодового слова видно, что каждый проверочный бит имеет свое происхождение и является

«индивидуальной» комбинацией каких-либо информационных битов. При этом очевидно, что по сравнению с контролем четности методом дублирования разряда или применением одного бита четности, применяемый метод обеспечивает более широкие возможности для исправления возникающих при передаче ошибок.

Циклические коды представляют класс блоковых кодов, кодирование и декодирование которых основано на использовании полиномиальных представлений, более простых, чем матричные процедуры. Широкое распространение этот класс кодов получил в связи с удобством практической реализации на основе достаточно простой современной цифровой элементной базы. Линейный код называется циклическим, если при любом циклическом сдвиге некоторого кодового слова получается другое кодовое слово, которое может быть получено тем же кодером, но из другой входной информационной группы.

Значения символов кодового слова можно рассматривать, как коэффициенты полинома: $\mathbf{u}(X)=u_0+u_1X+u_2X^2+\ldots+u_{n-1}X^{n-1}$. Наличие или отсутствие каких-либо членов в полиноме означает наличие нулей в соответствующих разрядах кодового слова. Если $u_{n-1}\neq 0$, то порядок полинома равен $n-1$.

Циклический код создается с помощью полиномиального генератора $\mathbf{g}(X)$. Для заданного циклического кода (n, k) вид полиномиального генератора является единственным:

$$\mathbf{g}(X)=g_0+g_1X+g_2X^2+\ldots+g_bX^b \ .$$

Предполагается, что коэффициенты g_0 и g_b – ненулевые, $b=n-k$. Каждый полином, описывающий какое-либо кодовое слово, имеет вид $\mathbf{u}(X)=\mathbf{m}(X)\mathbf{g}(X)$, т.е. кодовое последовательность двоичных символов действительно является кодовым словом, сформированным данным кодером, если оно делится без остатка на $\mathbf{g}(X)$. Полиномиальный генератор является одним простым множителем или произведением нескольких простых множителей полинома X^n+1. Использование циклического кодирования в описанном виде формирует несистематические

коды, т.е. в кодовом слове $\mathbf{u}(X)$ нет фрагмента, все символы которого совпадали бы с вектором $\mathbf{m}(X)=m_0+ m_1X+m_2X^2+m_3X^3+\ldots+m_{k-1}X^{k-1}$.

Однако чаще используется систематическая форма кодов, упрощающая процедуры обработки. При систематическом кодировании вектор $\mathbf{m}(X)$ совпадает с фрагментом выходного кодового слова $\mathbf{u}(X)$ и используется как его часть. Для этого символы информационного сообщения первоначально сдвигаются в сторону бóльших степеней X на $n–k$ позиций с помощью умножения на X^{n-k}. При этом формируется полином:

$$X^{n-k}\,\mathbf{m}(X)= m_0X^{n-k}+ m_1X^{n-k+1}+m_2X^{n-k+2}+m_3X^{n-k+3}+\ldots+m_{k-1}X^{n-1}.$$

Далее он делится на выбранный порождающий полином $\mathbf{g}(X)$. Результат деления можно записать в виде:

$$X^{n-k}\,\mathbf{m}(X)=\mathbf{q}(X)\mathbf{m}(X)+\mathbf{p}(X), \qquad (1.1)$$

где $\mathbf{q}(X)$ – частное от деления; $\mathbf{p}(X)$– остаток от деления, равный:

$$\mathbf{p}(X)=p0+p1X+p2X^2+\ldots+p^{n-k-1}\,X^{n-k-1}.$$

Остаток от деления суммируется по модулю 2 с обеими частями уравнения (1.1), в результате получается полином:

$$\mathbf{u}(X)=X^{n-k}\,\mathbf{m}(X)+\mathbf{p}(X)=\mathbf{q}(X)\mathbf{m}(X).$$

Полученный таким образом полином $\mathbf{u}(X)$ является кодовым словом, поскольку делится на $\mathbf{g}(X)$, но его фрагмент из k символов, стоящий в области больших степеней, теперь уже совпадает с информационным вектором $\mathbf{m}(X)$, а фрагмент, стоящий в области $n–k$ меньших степеней, является проверочной частью кодового слова, т.е.:

$$\mathbf{u}(X)=p_0+p_1X+p_2X^2+\ldots+p_{n-k-1}X^{n-k-1}+m_0X^{n-k}+m_1X^{n-k+1}+\ldots+m_{k-1}X^{n-1}.$$

Процедура кодирования иллюстрируется структурной схемой, приведенной на рисунке 1.3. Схема состоит из последовательно включенных элементов памяти $Э_1 \div Э_{n-k-1}$. Фактически она представляет собой регистр сдвига с обратными связями. В момент прихода тактового импульса каждый элемент памяти запоминает значение символа на своем входе и хранит его до поступления следующего тактового импульса. Между ними находятся сумматоры по модулю 2. На их входы поступают выходные сигналы с элементов памяти и с элементов $g_1 \div g_{n-k-1}$. Если коэффициент $g_1 \div g_{n-k-1}$. равен единице, то выходной сигнал ключа (Кл.) подается на соответствующий сумматор, если равен нулю, то суммирование не производится.

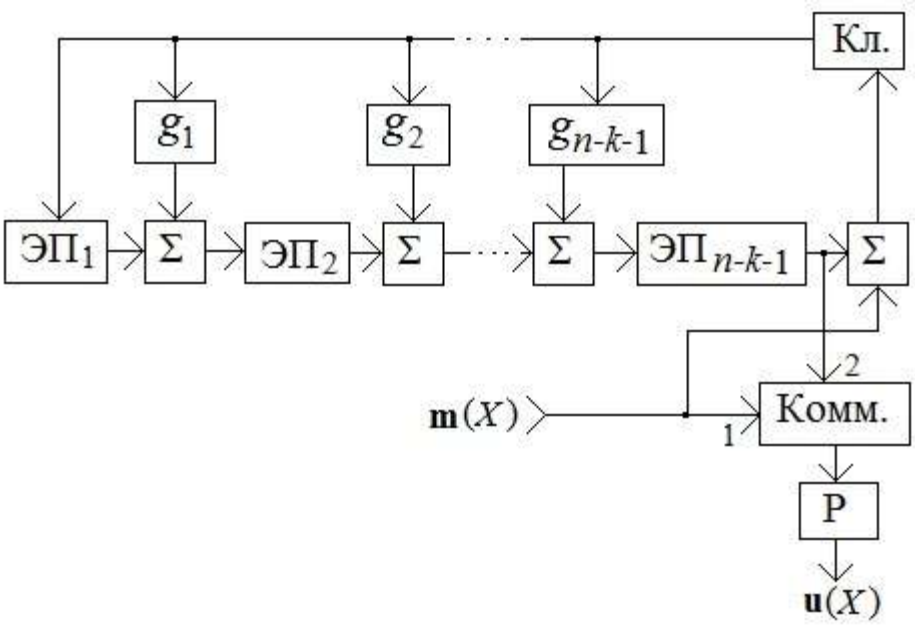

Рисунок 1.3.

Схема работает следующим образом. Моменты появления тактовых импульсов совпадают с появлением информационных символов. При первых символах последовательности $\mathbf{m}(X)$ ключ закрыт, а коммутатор (Комм.) подключает на выход сигнал со своего первого входа, т.е. непосредственно

исходную информационную последовательность. После передачи k-го информационного символа ключ открывается, а коммутатор на свой выход подключает сигнал со второго входа. В следующих $n–k$ тактах происходит формирование проверочных символов. После окончания n тактов обработки в выходном регистре (Р) оказываются записаны все сформированные символы кодового слова, которые далее поступают в передатчик. Для повышения скорости работы кодера иногда используют комбинацию табличного метода и регистра с обратными связями [4, 26].

Некоторые виды блоковых кодов в силу различных причин используются особенно часто. Рассмотрим некоторые из них.

Коды Хемминга. Это достаточно простой вид блоковых кодов, параметры которых выбираются из следующих условий: $n=2^m–1$; $k=2^m–1–m$, где m=2, 3, 4, …

Они способны исправлять все одиночные ошибки в кодовом слове. Декодирование кодов Хемминга производится достаточно просто [4, 26]. Коды Хемминга несмотря на относительно небольшую эффективность по сравнению с другими кодами, достаточно привлекательны тем, что они требуют при заданной блине блока минимальную избыточность для исправления одной ошибки. Кодирование производится обычно с использованием соответствующей матрицы.

Код Голея. Это линейный блоковый код с параметрами (n,k)=(23,12). Он допускает исправление до трех ошибок в кодовых словах длиной 23 символа. Из-за относительно небольшой его длины кодирование и декодирование двоичного кода Голея может быть выполнено даже с использованием соответствующих таблиц [26]. В этом случае таблица содержит список из 2^{12}=4096 кодовых слов, пронумерованных непосредственно двоичной совокупностью соответствующих информационных частей блока. Код Голея имеет циклическую природу и может быть получен и помощью циклического кодирования. Его порождающий полином равен $\mathbf{g}(X)=1+X^2+X^4+X^5+X^6+X^{10}+X^{11}$.

Степень кодирования кода равна 11/23, что не всегда удобно в практическом осуществлении, поэтому часто применяется близкий к нему расширенный код Голея с параметрами (24,12), получаемый добавлением к

кодовому слову еще одного проверочного символа. Его кодовая скорость равна ½ и практическая реализация проще. Код может исправлять в кодовом слове кроме всех трехсимвольных ошибок также некоторое число четырехсимвольных ошибок и является существенно более мощным, чем код Хемминга.

Коды Рида-Малера также представляют собой линейные блоковые коды. Для кодирования используются соответствующие матричные процедуры. Используются коды с разными наборами параметров. Привлекательность кодов основана на достаточно простой процедуре декодирования, базирующейся на мажоритарной логике, при которой решение о значении символов принимается по большинству результатов, полученных из соответствующих проверочных уравнений.

Коды БЧХ (Боуза-Чоудхури-Хоквенгема). Эти коды можно считать обобщением кодов Хэмминга, позволяющим исправлять несколько ошибок в блоке. Они позволяют относительно просто менять параметры кодирования, варьируя длину блока, кодовую скорость и возможности коррекции ошибок. Порождающий полином всегда имеет длину $n–k$. Коды бывают как двоичные, так и недвоичные. В наиболее часто используемых кодах применяются блоки длиной $n=2^m–1$, m=2,3,4,… . Известны таблицы порождающих полиномов наиболее эффективных кодов [26, 69]. Коды задаются корнями порождающего полинома. Иногда эффективность кодов с большой избыточностью хуже, чем кодов с меньшей избыточностью. Наибольшая эффективность достигается в зависимости от кодовой скорости при фиксированном n находится ориентировочно в интервале значений R от 1/3 и 3/4.

Коды Рида-Малера, БЧХ и Рида-Соломона относятся к полиномиальным кодам, которые отличаются тем, что на корни их порождающих полиномов накладываются дополнительные условия.

Недвоичные коды. Недвоичные коды требуют более сложной обработки, чем двоичные, однако реализуют бóльшую эффективность. В качестве символов недвоичных кодов обычно рассматриваются группы бит. Наиболее часто используются коды Рида-Соломона, являющиеся недвоичными БЧХ кодами.

Коды Рида-Соломона являются множеством кодовых слов, компоненты которых равны значениям некоторых определенных полиномов. Обработка значений символов производится в соответствии с правилами вычислений в полях Галуа. Для их описания применяются специальные символы α, обозначающие дополнительные однозначные элементы алгебры конечных полей используемой размерности. Коды позволяют исправлять не только несколько одиночных ошибок, но и пакеты ошибок. Обычно используется систематическая форма кодов. Кодирование при этом осуществляется схемой, сходной со схемой на рисунке 1.3 для кодирования двоичных кодов.

Элементы памяти запоминают не двоичный символ, содержащий один бит информации, а символ, несущий несколько информационных бит. Вместо двоичных коэффициентов g_i используются коэффициенты, образованные из степеней α, которые в данном случае также представляют несколько бит информации. Сложение, как и умножение, производится по правилам операций в конечных полях.

Коды Рида-Соломона позволяют исправить любой набор из $[(n–k)/2]$ ошибок в слове или любой набор $n–k$ стираний (т.е. определить место в блоке, в котором произошла ошибка без последующего ее исправления). Также имеется возможность одновременной коррекции определенного количества ошибок и стираний. Коды обладают существенными преимуществами перед двоичными кодами при воздействии коротких импульсных помех. Поскольку исправляется не бит, а символ с несколькими битами, то длина помехи может составлять и несколько бит, если она не выходит за длительность символа.

Нелинейные коды. Известны различные виды нелинейных кодов (Коды с контрольной суммой, инверсные, коды на основе перестановок, с повторением и без повторения символов, и т.д.). Построение нелинейных кодов сложнее, чем линейных, границы возможных значений параметров обычно связаны сложным образом и более узкие, чем у линейных кодов. В настоящее время коды находят ограниченное применение.

В следующем параграфе будут рассмотрены более сложные коды, основой для которых являются, как правило, сверточные и блоковые коды.

1.3. Виды сложных кодов

Под сложными кодами часто понимаются коды, которые получаются различными операциями над простыми кодами, при этом под простыми кодами можно понимать полученные с помощью методов сверточного и блокового кодирования. Здесь будут рассмотрены коды, модифицированные добавлением или удалением символов; комбинации, получаемые совместным использованием кодов; коды, предназначенные для декодирования, проводимого повторением нескольких последовательных процедур (итеративно декодируемые коды); коды, использующие особенности модуляции сигналов (сигнально-кодовые конструкции) [3, 4, 26, 70-92].

Модифицированные коды. Они строятся на основе некоторого исходного кода, к которому либо добавляют дополнительные символы (расширенный код), либо сокращают часть информационных символов без изменения расстояния между кодовыми символами (укороченный код), либо выбрасывают часть символов (перфорация или выкалывание). Таким образом, достигается определенная гибкость в получении нужных параметров передачи информации. Например, одной из ситуаций, когда эффективно укорочение, является потребность применить код Хэмминга, но при этом требуется число кодовых символов не равное 2^m, m – целое.

Укорочение кодов. Укорочение производится отбрасыванием определенного числа s позиций в информационной части кодового слова (s – глубина укорочения). В слове на передачу информации теперь отводится только $k–s$ символов, но длина проверочной части $n–k$ остается прежней. Укороченное кодовое слово формируется посредством того, что в некоторых фиксированных позициях информационной части исходного (неукороченного) кодового слова устанавливаются нули. Остальные позиции могут принимать произвольные

значения в зависимости от информации, передаваемой в этом кодовом слове. Положение нулевых позиций принципиального значения не имеет, однако для практического удобства кодирования и декодирования обычно выбирается *s* старших позиций. Таким образом, информационная часть кодового слова приобретает вид:

$$\mathbf{m}(X)=m_0+ m_1X+m_2X^2+m_3X^3+\ldots+m_{k-1}X^{k-1-s}.$$

Далее оно также систематическим кодером сдвигается на *n*–*k* разрядов, делится на порождающий полином и остаток от деления записывается в младшие пустые разряды. Образуется укороченный линейный код, степень которого не превышает *n*–*s*–1. В общем случае такой код не остается циклическим.

Если для кодирования используется порождающая матрица $\mathbf{G} = \|\mathbf{E}:\mathbf{P}\|$, то она также должна быть модифицирована. Для этого из нее удаляются s выбранных столбцов и одновременно s строк, соответствующих ненулевым элементам удаленных столбцов.

Пусть исходную матрицу можно записать в виде:

$$\mathbf{G} = \begin{Vmatrix} 1,0,0,\ldots..0, g_{1,k+1}, g_{1,k+2},\ldots..,g_{1,n} \\ 0,1,0,\ldots..0, g_{2,k+1}, g_{2,k+2},\ldots..,g_{2,n} \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ 0,0,0,\ldots..1, g_{k,k+1}, g_{k,k+2},\ldots..g_{k,n} \end{Vmatrix} = \begin{Vmatrix} 1,0,0,\ldots..0, p_{1,k+1}, p_{1,k+2},\ldots..,p_{1,n} \\ 0,1,0,\ldots..0, p_{2,k+1}, p_{2,k+2},\ldots..,p_{2,n} \\ \ldots\ldots\ldots\ldots\ldots\ldots \\ 0,0,0,\ldots..1, p_{k,k+1}, p_{k,k+2},\ldots..p_{k,n} \end{Vmatrix}.$$

Если при укорочении удаляется *s* первых столбцов (и соответственно, *s* первых строк), то размер новой порождающей матрицы становится равным (*k*–*s*)×(*n*–*s*). Единичная информационная матрица становится размером (*k*–*s*)×(*k*–*s*). Элементы новой укороченной проверочной матрицы: $g_{i,j} = p_{i-s,j-s}$. Операция укорочения уменьшает длину информационной части, в то время как число проверочных символов остается прежним. В результате может быть исправлено большее число комбинаций ошибок, и помехоустойчивость передачи информации

возрастает. Таким образом, укороченные коды более эффективны, чем исходные неукороченные коды.

Важное свойство укороченных кодов состоит в том, что для работы с ними могут быть использованы те же самые кодеры и декодеры, что и для неукороченных кодов. Однако для них обычно не выполняется правило циклического сдвига. При практической реализации слова обычно дополняются нулями в откинутых старших разрядах и используются те же самые алгоритмы кодирования и декодирования. А при передаче сообщений включенные нули не передаются.

Расширение кодов. Расширение кодов означает добавление в кодовое слово проверочных символов при сохранении в нем количества информационных символов. В результате получается код (n+s,k). Расширенная порождающая матрица может быть получена из исходной матрицы добавлением к ней соответствующих s столбцов.

Ее размер становится равным $k\times(n+s)$. Если вводимые новые элементы обозначить через $q_{i,j}$, то расширенная матрица приобретает вид:

$$\mathbf{G} = \begin{Vmatrix} 1,0,0,.....,0, g_{1,k+1}, g_{1,k+2},....., g_{1,n}, q_{1,n+1},..., q_{1,n+s} \\ 0,1,0,.....,0, g_{2,k+1}, g_{2,k+2},....., g_{2,n}, q_{2,n+1},..., q_{2,n+s} \\ ..... \\ 0,0,0,.....,1, g_{k,k+1}, g_{k,k+2},.....g_{k,n}, q_{k,n+1},..., q_{k,n+s} \end{Vmatrix}.$$

Расширение кодов также увеличивает помехоустойчивость и изменяет кодовую скорость.

Иногда рассматривается объединенная операция расширения, модифицирующая исходный код и состоящая в добавлении в порождающую матрицу и новых строк, и новых столбцов. Ее размер становится равным $(k+s)\times(n+s)$. При этом ее часть, имеющая вид единичной матрицы, занимает не одну из половин, а меньшую ее часть. Матрица приобретает вид:

$$\mathbf{G} = \begin{Vmatrix} 1,0,0,.....,0, g_{1,k+1}, g_{1,k+2}, ..., g_{1,n}, q_{1,n+1}, ..., q_{1,n+s} \\ 0,1,0,.....,0, g_{2,k+1}, g_{2,k+2}, ..., g_{2,n}, q_{2,n+1}, ..., q_{2,n+s} \\ ..... \\ 0,0,0,.....,1, g_{k,k+1}, g_{k,k+2}, ...g_{k,n}, q_{k,n+1}, ..., q_{k,n+s} \\ q_{k+1,1}, q_{k+1,2}, q_{k+1,3}, ..., q_{k+1,n+s} \\ ..... \\ q_{k+s,1}, q_{k+s,2}, q_{k+s,3}, ..., q_{k+s,n+s} \end{Vmatrix}.$$

Кроме рассмотренных вариантов модификации известны также варианты, когда к множеству всех возможных кодовых слов добавляются новые кодовые слова (операция пополнения) или из этого множества удаляются некоторые кодовые слова (операция выбрасывания) [26, 75]. Для осуществления операции пополнения при сохранении линейности кода в порождающую матрицу добавляются новые линейно независимые строки.

Операция выбрасывания в общем случае приводит к нелинейному коду. Для сохранения линейности возможно удалить часть строк исходной порождающей матрицы. Для систематических кодов операции выбрасывания и укорочения совпадают.

Перфорация кодов. Перфорация сверточных кодов применяется в основном для сверточных кодов и состоит в том, что из процесса передачи в канал удаляются некоторые символы с выхода кодера. Поскольку при этом структура собственно кодера не меняется, то и количество информационных символов не меняется. В результате формируется перфорированный код с более высокой кодовой скоростью.

Правило удаления символов задается, как правило, матрицей (маской) перфорации. Матрица перфорации P задает правило удаления выходных символов. Обычно процесс перфорации является периодическим и сходный (неперфорированный) код имеет скорость $R=1/n$, n – целое. В этом случае количество N_p кодовых символов, через которое процесс повторяется, должно быть кратно n.

Элементы матрицы P равны нулю или единице. Ноль указывает, что соответствующий двоичный символ будет удален, единица – что он будет оставлен. Размеры матрицы P равны $n \times N_p$. Если она содержит q нулей, то кодовая скорость после перфорации станет равной $R=N_p/(nN_p-q)$.

Формирование перфорированного кода производится следующим образом. После прихода на исходный кодер N_p информационных символов он вырабатывает N_p групп кодовых символов, каждая из которых соответствует одному из исходных символов. Эти группы можно записать в виде векторов размера n. Далее символы каждого из векторов соотносятся с соответствующим столбцом маски перфорации. Если элемент маски равен нулю, то символ с таким номером из вектора удаляется. (После этого размеры векторов становятся неравными.) Всего будет удалено q символов. Далее из оставшихся элементов всех векторов последовательно составляется перфорированная последовательность длиной nN_p-q. После этого аналогично производится следующий период обработки.

При записи структуры перфорированного кода правило получения каждого кодового символа как обычно записывается в форме восьмеричного кода, а место перфорированных символов обозначается буквой X. Перфорация используется достаточно широко, поскольку для различных вариантов перфорированного кода, полученных из одного и того же исходного кода, можно использовать один и тот же декодер, предназначенный для исходного кода.

Перфорация применяется также и к линейным блоковым кодам. Она состоит в удалении из кодового слова p проверочных символов. Код при этом остается линейным блоковым, количество информационных символов k в кодовом слове не изменяется, а общее количество кодовых символов уменьшается до $n–p$. Избыточность уменьшается, а скорость кода возрастает.

Комбинированные коды. Известны различные способы комбинирования кодов. Рассмотрим метод последовательного соединения (concatenation – конкатенация). Если используются блоковые коды C_1 и C_2, то их последовательное соединение эквивалентно поочередной передаче кодовых слов,

полученных с помощью первого кода и второго кода. В результате последовательного соединения $i=1\div m$ кодов образуется комбинированный код с параметрами n_0 и k_0:

$$n_0 = \sum_{i=1}^{m} n_i , \quad k_0 = \sum_{i=1}^{m} k_i ,$$

где n_i и k_i – параметры i-го кода. Если $\mathbf{G}_i$ – исходные порождающие матрицы кодов, составляющих последовательное соединение, то его порождающая матрица $\mathbf{G}_0$ последовательного соединения кодов будет равна:

$$\mathbf{G}_0 = \begin{Vmatrix} \mathbf{G}_1,0,0,\ldots,0 \\ 0,\mathbf{G}_2,0,\ldots,0 \\ \ldots\ldots\ldots\ldots \\ o,o,o,\ldots,\mathbf{G}_m \end{Vmatrix}.$$

Величины n_i и k_i у объединяемых кодов чаще не совпадают, т.к. последовательно соединенные коды используют в случае необходимости реализации разного уровня защиты от ошибок. Если же все n_i и k_i одинаковы, это эквивалентно m-кратной повторной передаче одного кодового слова.

Другим типом комбинированных кодов являются *произведения кодов*. В простейшем виде они представляют собой последовательное (повторное) кодирование. При этом выход первого кодера является входом второго кодера, и т.д. (рисунок 1.4.). Несмотря на простоту, метод формирует достаточно помехоустойчивые коды, особенно для кодов с низкой кодовой скоростью. Если используются два кодера, то код первого из них обычно называется внешним кодом, второй – внутренним кодом.

Рисунок 1.4.

Если **G** и **H** – порождающие матрицы двух кодов с размерами $m \times n$ и $p \times r$, то порождающая **Q** матрица произведения этих кодов есть кронекерово произведение этих матриц, т.е.:

$$\mathbf{Q} = \begin{Vmatrix} g_{1,1}\mathbf{H}, g_{1,2}\mathbf{H}, \ldots, g_{1,n}\mathbf{H} \\ g_{2,1}\mathbf{H}, g_{2,2}\mathbf{H}, \ldots, g_{2,n}\mathbf{H} \\ \ldots\ldots\ldots\ldots\ldots \\ g_{m,1}\mathbf{H}, g_{m,2}\mathbf{H}, \ldots, g_{m,n}\mathbf{H} \end{Vmatrix},$$

где gij– элементы матрицы **G**. Матрица **Q** имеет размер $mp \times nr$. Если параметры исходных кодов равны (n_1,k_1) и (n_2,k_2), то при генерации такого кода кодовые слова длиной n_1 символов, уже закодированные внешним кодом, записывают по строкам прямоугольного массива с n_1 столбцами, размещая один символ в один элемент массива. Так заполняют k_2 строк, а оставшиеся $n_2–k_2$ строк заполняются проверочными символами внутреннего кода. При этом k_2 элементов каждого столбца считаются информационными символами внутреннего кода. Общее кодовое слово полученного кода передается из массива по столбцам.

Каскадные коды. Каскадные коды предложены в [77-81]. Чаще всего используются каскадные коды, использующие в качестве внешнего недвоичные блоковые коды Рида-Соломона. Символы полученных кодовых слов подвергаются перемежению, в результате чего порядок следования символов изменяется. Далее они подвергаются кодированию с помощью внутреннего кодера, который может быть как блоковым, так и сверточным. Если внешний, и внутренний коды C_1 и C_2 имеют параметры (n_1, k_1) и (n_2, k_2), то параметры сформированного каскадного кода равны (n_1n_2, k_1k_2) . Когда внутренний код – двоичный линейный блоковый, тогда и полученный каскадный код также является двоичным линейным блоковым.

В [80,81] были предложены обобщенные каскадные коды, как усложнение каскадных кодов. Они способны исправлять как случайные ошибки, так и случайные пакеты ошибок. Обобщение заключается в том, что вводится

разбиение внутреннего кода на подкоды со своей иерархией, а также используется несколько внешних кодов в соответствии с иерархией. При этом если необходимо, то с помощью выбора параметров внутренних и внешних кодов легко получается блоковый или сверточный код с неравной защитой от ошибок. Сообщения, закодированные на верхнем уровне иерархии, имеют усиленную корректирующую способность по сравнению с нижними уровнями.

Турбокоды. Турбокоды могут быть отнесены как к классу комбинированных составных кодов, так и к итеративно декодируемым кодам, так как несут в себе особенности обоих классов. Они предложены в [82,83] и образуются компоновкой нескольких (чаще двух) кодов, создаваемых простыми кодерами (компонентными) и получаемых по-разному из одной и той же информационной последовательности.

Считается, что основной выигрыш при декодировании обусловлен следующим. Обычный декодер и при «жестком», и при «мягком» декодировании выдает «жесткие» двоичные решения. Если же применен каскадный код и декодирование производится в две ступени, то первый декодер не должен обязательно выдавать «жесткие» решения, т.к. его выходной результат не является окончательным результатом декодирования, а имеет промежуточный характер. Поэтому он тоже может выдавать «мягкие» решения, что значительно повышает помехоустойчивость. Точно также, если общий код скомпонован из двух различных кодов, то при параллельной обработке результат декодирования каждого из кодов поступают в целях коррекции результата декодирования на один из входов декодера другого кода и учитываются им. Причем такой результат выдается с «мягком» виде на основе метрики, построенной использованием отношения правдоподобия.

Достаточно эффективные коды получаются даже при простых используемых компонентных кодерах. Если такие кодеры осуществляют сверточное кодирование, то более эффективны систематические рекурсивные коды. Они формируются кодерами с обратной связью, имеющими бесконечную импульсную характеристику. На рисунке 1.5 приведен пример структурной схемы

подобного турбокодера, содержащего два компонентных рекурсивных сверточных кодера, хотя на их количество ограничений нет. Входная информационная последовательность **m** поступает на вход коммутатора (Комм.), а также на первый кодер C_1 и через блок чередования (БЧ) на второй кодер C_2. Оба кодера имеют сдвиговые регистры, содержащие по К–1 ячеек. Обратная связь обеспечивается тем, что на входы регистров с помощью сумматоров Σ_1 в каждом такте подается сумма входного символа и символов из всех ячеек регистра. На сумматоры Σ_2, так же, как и в нерекурсивном варианте кодера (рисунок 1.2), подаются символы только с некоторых ячеек регистра, определяемых видом используемого порождающего полинома. Коммутатор поочередно подключает на выход символы последовательности **m** и вырабатываемых сумматорами Σ_2 последовательностей $\mathbf{v}_1$ и $\mathbf{v}_2$.

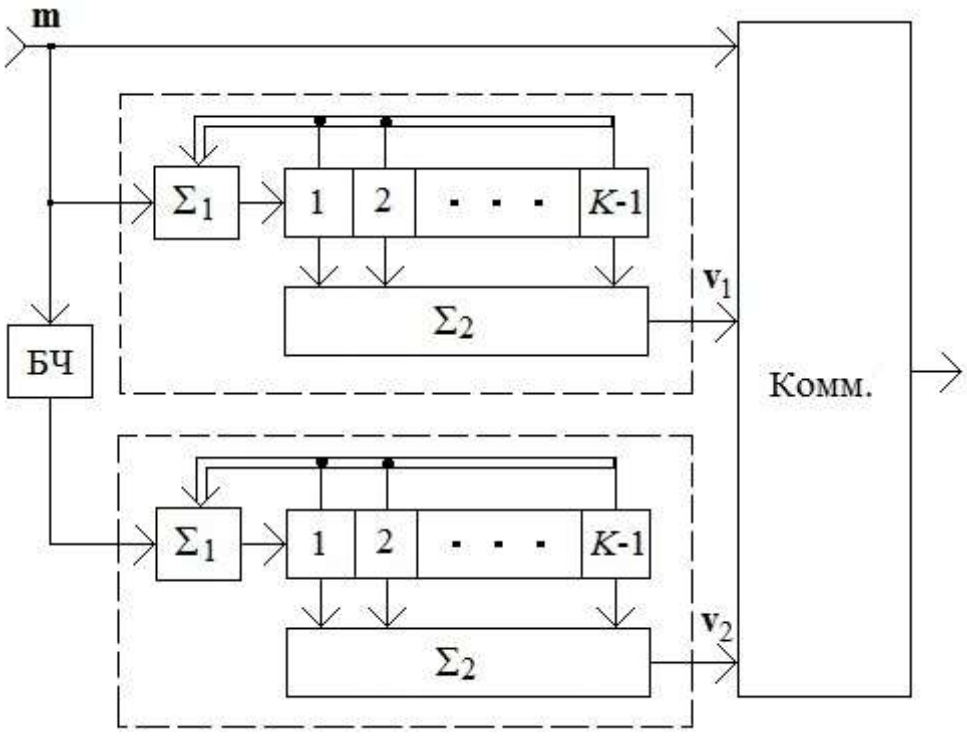

Рисунок 1.5.

Блок чередования производит перемежение символов, изменяя порядок их следования. Последовательности символов с выходов обоих кодеров не совпадают, при этом эффективность кодирования зависит от того, как часто в

обеих последовательностях будут встречаться одинаковые фрагменты. Это во многом зависит от правила перемежения. В качестве компонентных кодеров могут также использоваться линейные блоковые кодеры.

Формируемый турбокод во многом похож на выше рассмотренное произведение кодов. Отличие заключается в том, что если раньше заполнялся весь прямоугольный массив размера $n_1 \times n_2$, то теперь из него удалены $(n_1 - k_1) \times (n_2 - k_2)$ символов, т.е. все проверочные символы одного кода, полученные из проверочных символов от другого кода.

Итеративно декодируемые коды. К таким кодам обычно относят такие коды, в которых многократно используется «мягкое» декодирование, и достигаются высокие показатели помехоустойчивости при относительно небольших объемах вычислений [84-92]. Кроме турбокодов и близких к ним, в настоящее время получили распространение коды с низкой плотностью проверки на четность (LDPC – low density parity control) и коды для многопорогового декодирования.

LDPC-коды – это линейные коды, у которых малое число ненулевых элементов в порождающей матрице [4,26]. В принципе его можно рассматривать, как турбокод, составными кодами является совокупность простейших кодов. Во многих работах указывается, что в определенных условиях по эффективности они могут превосходить турбокоды. Регулярный LDPC-код является линейным кодом, у которого количество единичных элементов в каждой строке и столбце проверочной матрицы, связанной с порождающей матрицей, много меньше количества символов в кодовом слове (Матрица сильно разрежена, ориентировочно содержание единиц может быть порядка 1% и ниже). Кроме этого почти всегда накладывается ограничение, состоящее в том, чтобы в матрице не было двух строк или столбцов, одновременно содержащих единицы более чем в одной позиции.

В нерегулярных LDPC-кодах распределение единичных элементов в строках и столбцах матрицы выбирается в соответствии с некоторым

неравномерным распределением вероятности. Они могут обеспечить несколько бóльшую помехоустойчивость, чем регулярные коды.

Многопороговые методы кодирования и декодирования [90-93]. Многопороговое декодирование также основано на итеративных процедурах и может быть применено как при использовании достаточно простых сверточных и блоковых кодеров, так и для сложных каскадных кодеров. Эффективность кодов сравнима с эффективностью LDPC-кодов, но практическая реализация при близких характеристиках, как правило, проще.

Количество информационных символов в блоковом коде для кодовой скорости R=1/2 равно как минимум $2q$+1, где q – наибольшая степень порождающего полинома. Для кодовых скоростей R=1/n для уменьшения риска размножения ошибок иногда от кодов вида (n, k) переходят к кодам вида (qn, qk), где q – небольшое целое число. Общая кодовая скорость R=qk/qn при этом сохраняется.

В кодере структура кодового слова образуется следующим образом. Группа символов входного информационного потока первоначально разделяется на q групп и из них формируется q проверочных групп символов, далее все группы передаются поочередно, складывая общее кодовое слово. Для формирования каждой проверочной группы одновременно используются все информационные группы. Пример структуры подобного кодера для q=2 и кодовой скорости R=2/4 приведен на рисунке 1.6.

Информационная группа символов $\mathbf{m}$ разбивается на две группы $\mathbf{m}_1$ и $\mathbf{m}_2$ одинаковой длины K и синхронно поступает на входы 1 двух коммутаторов (Комм.). Первоначально в течение K тактов оба коммутатора с выходом соединяют свои первые входы. При этом K информационных символов подаются на выходы кодера и одновременно заполняют K ячеек сдвиговых регистров. Далее оба коммутатора на выходы подключают свои вторые входы 2, после этого вычисляются K проверочных символов двух проверочных последовательностей $\mathbf{v}_1$ и $\mathbf{v}_2$ и подаются на выходы кодера. Ко входам каждого из сумматоров по модулю 2 (Σ) подключены некоторые ячейки как первого, так и второго регистров. Номера

этих ячеек определяются используемыми порождающими полиномами. Таким образом, сформированное кодовое слово образуется группами символов $\mathbf{m}_1$, $\mathbf{m}_2$, $\mathbf{v}_1$, $\mathbf{v}_2$.

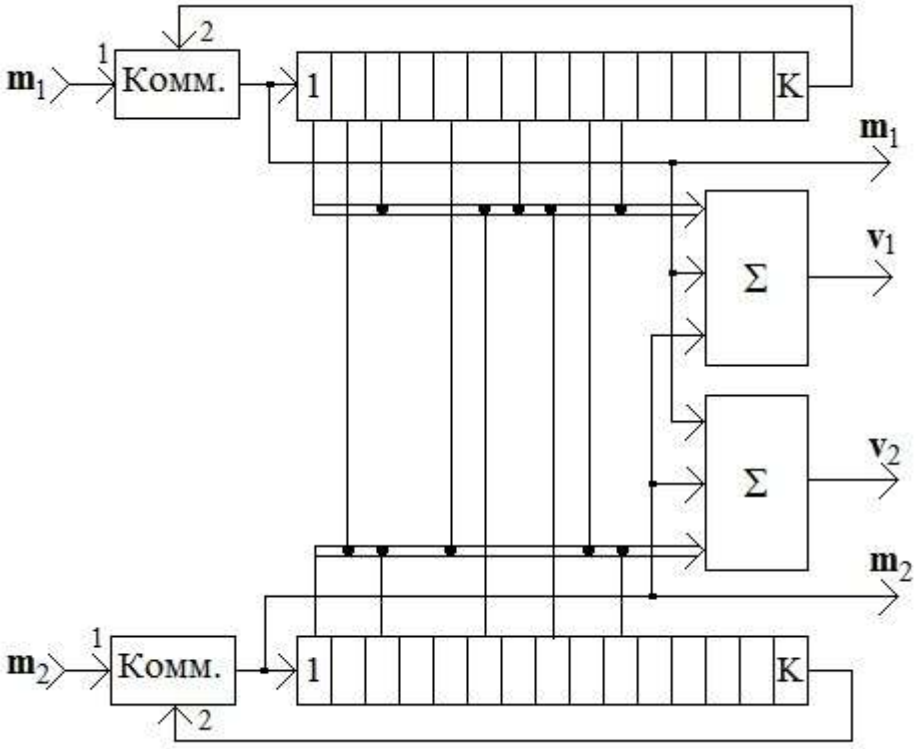

Рисунок 1.6.

Сигнально-кодовые конструкции. Они представляют собой совместное взаимоувязанное использование модуляции и кодирования (решетчатое кодирование – trellis coded modulation, TCM) и предназначаются для повышения эффективности использования ограниченного частотного диапазона [70-76]. В случае использования цифровой модуляции сигнальное множество представляет собой набор сигналов с определенными параметрами (как правило, амплитудно-фазовыми). Каждому сигналу из набора приписывается определенное сочетание нескольких бит кодового слова. Основной результат достигается за счет следующих факторов. Переход от двоичной модуляции к использованию набора сигналов позволяет повысить скорость передачи в той же полосе частот. Однако увеличение набора сигналов снижает помехоустойчивость при ограничении предельной мощности передатчика. Однако часть ресурса увеличения скорости

передачи можно потратить на введение кодирования, которое компенсирует снижение помехоустойчивости.

Этот же фактор можно использовать и при увеличении количества уже имеющихся символов в наборе. Дополнительные символы дают использовать тот или иной метод кодирования, что в свою очередь повышает помехоустойчивость передачи. При правильном выборе параметров модуляции и кодирования этот фактор преобладает, что дополнительно повышает помехоустойчивость.

Кроме этого, большое значение имеет правильный выбор соотнесения сигналов из используемого набора и вариантов последовательности битов кодового слова. В частности, если набор сигналов представляет собой «созвездие» на фазовой плоскости, то близко расположенным сигнальным точкам присваиваются такие сочетания символов, которые из-за применения кодирования следовать одно за другим не могут. Этот фактор различного уровня вероятности ошибок дополнительно повышает помехоустойчивость.

Формирование передаваемых сигналов в подобных системах передачи информации фактически разбивается на два этапа. На первом производится собственно кодирование сигналов в их логическом представлении с учетом последующего соответствия используемому набору сигналов. А на втором этапе фрагменты кодированной последовательности символов приписываются передаваемым сигналам в их аналоговом представлении. При этом первый этап использует известные схемы кодеров.

Таким образом, современное состояние методов кодирования характеризуется большим разнообразием применяемых кодов, однако подавляющее большинство использует базовые принципы, сводящиеся к методам сверточного и блокового кодирования.

1.4. Критерии помехоустойчивости передачи сигналов и качества диагностики кодеров

Для оценки помехоустойчивости передачи цифровых сигналов в основном применяется вероятность ошибки и достоверность передачи [1-15]. Вероятность ошибки относится к принятым битам или символам и понимается, как среднее число принятых бит или символов, отличающихся от переданных. Основной причиной ошибок служит шум каскадов приемника в основном теплового происхождения. Сумма шумовой составляющей и составляющей полезного сигнала в какие-то моменты времени будет соответствовать не переданному символу, а другому символу из используемого алфавита, в результате чего детектором будет принято ошибочное решение. Ошибки также могут возникать за счет помеховых сигналов, приходящих от внешних источников.

Вероятность ошибки определяется соотношением между уровнями полезной и мешающей составляющих в принятом сигнале. При передаче по нестационарному каналу сигнал претерпевает различного рода искажения. Колебания коэффициента передачи вызывают быстрые и медленные замирания. Многолучевый характер распространения приводит к дисперсии энергии сигнала по времени. Если временной интервал дисперсии соизмерим с длительностью символов, это ведет к появлению межсимвольной интерференции сигналов из-за наложения напряжения предыдущим символов на последующие. Такие искажения также повышают вероятность ошибки.

Поскольку в нестационарных каналах передачи сигналов изменения параметров канала носят случайный характер, то и изменения вероятности ошибки также случайны по времени. Для учета этого фактора используется достоверность передачи. Под этим обычно понимается процент времени сеанса передачи, в течение которого вероятность ошибки не хуже определяемой соответствующими нормами на качество передачи.

Диагностическое решение о параметрах используемого кода принимается на основе анализа принимаемых символов, и текущее состояние канала передачи

непосредственно на него не влияет, а влияние оказывается косвенно через вероятность битовой (символьной) ошибки. Основой для диагностики является факт, что сущность кодирования состоит в добавлении дополнительных проверочных символов, определенным образом связанных с передаваемыми информационными символами. Именно выявление этих связей позволяет сделать выводы о параметрах кодера.

Для выявления связей необходимо определенное количество символов в анализируемой выборке. Требуемый объем выборки может достаточно сильно различаться в разных конкретных условиях. Поэтому диагностическая ошибка может быть обусловлена двумя различными причинами.

Одна из них состоит в том, что используемого объема конкретной выборки недостаточно для вынесения решения даже при незначительном уровне шумов по сравнению с уровнем полезного сигнала. В результате возможны следующие ситуации. Диагностический алгоритм по окончании процедуры анализа сообщает, что правильное решение не может быть получено вообще. В другой ситуации алгоритм формирует несколько вариантов решения и в удачном варианте сообщает вероятность каждого из решений. Выходом может служить либо повторение процедуры анализа с другой выборкой, либо увеличение объема данной выборки до тех пор, пока не останется единственное решение или вероятность одного из возможных решений будет настолько преобладать над вероятностью других вариантов, что они могут быть отброшены.

Другая причина диагностической ошибки появляется в случае заметного уровня вероятности битовой или символьной ошибки, возникающей по различным причинам. В этом случае принимаемая последовательность символов также может заметно отличаться от передаваемой. Это может привести к необходимости значительного увеличения объема анализируемой выборки. Но даже в этом случае при получении единственного диагностического решения оно может оказаться неверным, т.к. носит статистический характер. Поэтому здесь кроме собственно получения значений параметров кода требуется определять вероятность того, что полученный результат соответствует действительному.

Исходя из этого, в дальнейших главах предлагаются различные алгоритмы диагностики и исследуются их свойства.

1.5. Выводы

1. При эксплуатации систем передачи цифровых сигналов, использующих помехоустойчивое кодирование, могут возникать ситуации, когда информация о параметрах кодеров либо утрачена, либо неполная, либо отсутствует изначально. При этом в одних случаях невозможно исправление ошибок, возникающих при передаче по каналу, а в других в принципе невозможна передача информации. В таких ситуациях требования на качество и помехоустойчивость не выполняются.
2. В указанных ситуациях, тем не менее, возможно получение данных о параметрах кодирования. Основой для этого служат связи между принимаемыми символами, появляющиеся в передатчике при осуществлении кодирования.
3. Связи между кодированными символами достаточно разнообразны и зависят от используемых кодов. Анализ характера этих связей позволяет разработать диагностические алгоритмы для определения параметров кодов. В результате успешно осуществляется декодирование с исправлением ошибок даже в отсутствие сведений о параметрах кодера и обеспечивается требуемая помехоустойчивость и качество передачи информации.

2. Разработка алгоритмов диагностики сверточных кодов

В данной главе предложены и исследованы алгоритмы диагностики сверточных кодов. Излагается общий принцип, который позволяет определять структуру кодеров на основе связей между символами кодовых последовательностей [63-66, 99]. Предлагается вариант его реализации в форме матричного алгоритма. Исследуются его детерминированный и статистический варианты для каналов с различным уровнем поражения шумами, а также предлагаются «быстрые» алгоритмы диагностики. Диагностика позволяет принимать закодированные цифровые сигналы даже в случае отсутствия или утери информации о параметрах кодера и декодировать сигнал с исправлением ошибок, что существенно повышает помехоустойчивость передачи информации.

2.1. Принципиальные основы диагностики сверточных кодов

Как известно, сверточное кодирование в общем виде заключается в последовательной передаче нескольких новых кодовых символов после поступления каждого нового исходного информационного символа. Каждый из этих кодовых символов образуется с помощью свертки нового символа и нескольких предыдущих информационных символов с соответствующим порождающим полиномом.

Все кодовые символы, соответствующие одному информационному символу, образуются с помощью различных порождающих полиномов $\mathbf{g}_1 \div \mathbf{g}_q$. Набор таких полиномов полностью определяет структуру кодера и свойства генерируемого кода. Определение их вида и составляет задачу диагностики.

Пусть исходная информационная последовательность двоичных символов m_0, m_1, m_2, m_3, …, m_i,… будет представлена, как вектор $\mathbf{m}$, либо как полином $\mathbf{m}(X)=m_0+m_1X+m_2X^2+m_3X^3+\ldots+m_iX^i+\ldots$, где символы m_i принимают значения, равные либо 1, либо 0 в зависимости от передаваемой информации; X – оператор,

описывающий задержку по времени на интервал T длительности одного символа (X^2 означает задержку по времени на $2T$, X^3 означает задержку на $3T$, и т.д.).

Принципиальная основа диагностических алгоритмов сверточных кодов базируется на следующем. Основная сгенерированная кодовая последовательность $\mathbf{u}_0$ может быть разбита на группы символов, образованных при появлении на входе кодера одного нового кодового символа. Количество символов в группе определяется кодовой скоростью R и равно $q=1/R$. Для целей диагностики рассмотрим основную кодовую последовательность $\mathbf{u}_0$, как совокупность q частных кодовых последовательностей: $\mathbf{u}_1 \div \mathbf{u}_q = \mathbf{g}_1\mathbf{m} \div \mathbf{g}_q\mathbf{m}$ (произведение понимается, как двоичная свертка по модулю 2). Каждая частная кодовая последовательность образована своим порождающим полиномом.

Естественно, реализации частных последовательностей $\mathbf{u}_1 \div \mathbf{u}_q$ не совпадают (чем больше проявляется такое несовпадение, тем «лучше» применяемый вариант порождающих полиномов). Однако, для целей диагностики важны два свойства этих последовательностей. Одно из них заключается в том, что все эти частные кодовые последовательности образованы из *одной и той* же исходной информационной последовательности $\mathbf{m}(X)$. Второе базируется на предположении, что за время наблюдения структура кодера остается постоянной. Следовательно, хотя все частные кодовые последовательности $\mathbf{u}_1 \div \mathbf{u}_q$ различаются между собой, тем не менее, между ними существуют скрытые внутренние связи. Анализируя последовательности, можно выявить эти связи, а в результате определить вид порождающих полиномов $\mathbf{g}_1 \div \mathbf{g}_q$ и искомую структуру кодера.

Представление частных кодовых последовательностей в форме двоичной свертки дает возможность предложить три различных алгоритма диагностики. Рассмотрим их и оценим возможные особенности реализации. Будем считать, что перфорация кодовой последовательности не производится. Исходно рассмотрим кодовую скорость $R=1/2$, далее расширим рассмотрение на произвольную скорость.

При кодовой скорости $R=1/2$ кодер генерирует две частные кодовые последовательности: $\mathbf{u}_1(X)=\mathbf{g}_1(X)\mathbf{m}(X)$ и $\mathbf{u}_2(X)=\mathbf{g}_2(X)\mathbf{m}(X)$. Все далее

рассматриваемые алгоритмы заключаются в нахождении определенных преобразований этих последовательностей. Преобразования различающихся исходных частных последовательностей символов $\mathbf{u}_1$ и $\mathbf{u}_2$ должны в результате формировать одинаковую выходную последовательность **m**. Если это достигнуто, то задача диагностики решена.

Все алгоритмы построены по единому принципу. Используются некоторые «поисковые» полиномы $\mathbf{h}_1$ и $\mathbf{h}_2$, и в процессе диагностики их вид изменяется. По достижению вышеизложенной цели эти полиномы отождествляются с искомыми порождающими полиномами $\mathbf{g}_1$ и $\mathbf{g}_2$.

Алгоритм 1. При осуществлении операций этого алгоритма осуществляется двоичное деление частных кодовых последовательностей $\mathbf{u}_1(X)$ и $\mathbf{u}_2(X)$ на «поисковые» полиномы $\mathbf{h}_1(X)$ $\mathbf{h}_2(X)$, в результате получаются последовательности:

$$\mathbf{y}_1(X)=\mathbf{u}_1(X)/\mathbf{h}_1(X)=[\mathbf{g}_1(X)/\mathbf{h}_1(X)]\mathbf{m}(X), \qquad (2.1.)$$

$$\mathbf{y}_2(X)=\mathbf{u}_2(X)/\mathbf{h}_2(X)=[\mathbf{g}_2(X)/\mathbf{h}_2(X)]\mathbf{m}(X).$$

Вид полиномов $\mathbf{h}_1(X)$ и $\mathbf{h}_2(X)$ независимо изменяется, естественно, при этом меняются и последовательности $\mathbf{y}_1$ и $\mathbf{y}_2$. В случае, когда символы последовательностей $\mathbf{y}_1(X)$ и $\mathbf{y}_2(X)$ станут совпадать, то при условии одинакового значения порядков «поисковых» и искомых полиномов будет совпадать вид «поисковых» полиномов $\mathbf{h}_1(X)$, $\mathbf{h}_2(X)$ и вид искомых порождающих полиномов $\mathbf{g}_1(X)$, $\mathbf{g}_2(X)$, т.е.: $\mathbf{h}_1(X)=\mathbf{g}_1(X)$ и $\mathbf{h}_2(X)=\mathbf{g}_2(X)$.

Алгоритм 2. При осуществлении операций этого алгоритма обрабатывается одна из частных кодовых последовательностей. (Номер последовательности принципиального значения не имеет, будем рассматривать первую последовательность $\mathbf{u}_1(X)$). Другая последовательность при этом не изменяется.

Первая последовательность подвергается операции $\mathbf{h}_2(X)/\mathbf{h}_1(X)$, т.е., двоичному умножению на полином $\mathbf{h}_2(X)$ и двоичному делению на полином $\mathbf{h}_1(X)$

(порядок осуществления этих операций значения не имеет). Фактически, умножение на полином является повторным сверточным кодированием последовательности в соответствии с порождающим полиномом $\mathbf{h}_2(X)$.

В результате формируется последовательность символов:

$$\mathbf{y}_1(X)=[\mathbf{h}_2(X)/\mathbf{h}_1(X)]\mathbf{u}_1(X)=[\mathbf{h}_2(X)/\mathbf{h}_1(X)]\mathbf{g}_1(X)\mathbf{m}(X). \qquad (2.2)$$

Она сравнивается с неизмененной второй последовательностью $\mathbf{y}_2(X)=\mathbf{g}_2(X)\mathbf{m}(X)$. Случай их совпадения соответствует соблюдению равенств: $[\mathbf{h}_2(X)/\mathbf{h}_1(X)]\mathbf{g}_1(X)=\mathbf{g}_2(X)$, что при заранее известном кодовом ограничении определяет соблюдение равенств: $\mathbf{h}_1(X)=\mathbf{g}_1(X)$; $\mathbf{h}_2(X)=\mathbf{g}_2(X)$.

Алгоритм 3. При осуществлении операций этого алгоритма каждая из частных кодовых последовательностей $\mathbf{u}_1(X)$ и $\mathbf{u}_2(X)$ домножается на «поисковые» полимомы $\mathbf{h}_1(X)$ и $\mathbf{h}_2(X)$. (То есть, осуществляется повторное сверточное кодирование в соответствии с этими порождающими полиномами).

При этом формируются последовательности символов: $\mathbf{y}_1(X)=\mathbf{h}_1(X)\mathbf{u}_1(X)=\mathbf{h}_1(X)\mathbf{g}_1(X)\mathbf{m}(X)$ и $\mathbf{y}_2(X)=\mathbf{h}_2(X)\mathbf{u}_2(X)=\mathbf{h}_2(X)\mathbf{g}_2(X)\mathbf{m}(X)$. Производится поиск вида полиномов $\mathbf{h}_1(X)$ и $\mathbf{h}_2(X)$ до тех пор, пока символы последовательностей $\mathbf{y}_1(X)$ и $\mathbf{y}_2(X)$ не станут совпадать. Такое возможно, если выполняется условие: $\mathbf{h}_1(X)\mathbf{g}_1(X)=\mathbf{h}_2(X)\mathbf{g}_2(X)$, или $\mathbf{h}_1(X)=\mathbf{g}_2(X)$ и $\mathbf{h}_2(X)=\mathbf{g}_1(X)$. При этом $\mathbf{y}_1(X)=\mathbf{g}_2(X)\mathbf{g}_1(X)\mathbf{m}(X)$ и $\mathbf{y}_2(X)=\mathbf{g}_1(X)\mathbf{g}_2(X)\mathbf{m}(X)$. Поскольку $\mathbf{g}_1(X)\mathbf{g}_2(X)=\mathbf{g}_2(X)\mathbf{g}_1(X)$, то независимо от значения символов последовательности $\mathbf{m}(X)$ будет выполняться равенство $\mathbf{y}_1(X)=\mathbf{y}_2(X)$.

При внешнем сходстве эти алгоритмы имеют существенные различия в практическом осуществлении, определяющем их достоинства и недостатки. Алгоритмы 1 и 2 включают операцию двоичного деления. Ее стандартная реализация [4, 18], осуществляется несколько сложнее, чем сверточное кодирование (поразрядное умножение по модулю 2). Алгоритм 2 требует двух последовательных операций (умножение и деление), что затрудняет параллелизацию вычислений. В этих условиях алгоритм 3 обладает некоторыми

преимуществами при его осуществлении. По этой причине, несмотря на определенные особенности, которые будут рассмотрены в дальнейшем, он будет выбран, как основа для диагностики кодов.

При других кодовых комбинациях рассмотренные алгоритмы должны применяться следующим образом. Структура кодера в этом случае определяется q порождающими полиномами: $\mathbf{g}_1 \div \mathbf{g}_q$.

Группа кодовых символов, формируемая при появлении каждого нового информационного символа, в этом случае содержит q символов. Таким образом, общая кодовая последовательность может быть разделена на q частных кодовых последовательностей $\mathbf{u}_1 \div \mathbf{u}_q$. Значения их символов также обладают внутренней взаимосвязью, т.к. образованы из одной и той же исходной информационной последовательности $\mathbf{m}$. Для диагностики используется q «поисковых» полиномов $\mathbf{h}_1 \div \mathbf{h}_q$.

В соответствии с алгоритмом 1 вид поисковых полиномов будет совпадать с видом искомых порождающих полиномов, если выполняются равенства, аналогичные (2.1):

$$\mathbf{u}_1/\mathbf{h}_1 = \mathbf{u}_2/\mathbf{h}_2 = \ldots = \mathbf{u}_q/\mathbf{h}_q, \qquad (2.3)$$

т.е. все символы с одинаковыми номерами из новых последовательностей $\mathbf{y}_1 \div \mathbf{y}_q$, которые образованы двоичным делением частных кодовых последовательностей $\mathbf{u}_1 \div \mathbf{u}_q$ на соответствующие полиномы $\mathbf{h}_1 \div \mathbf{h}_q$, должны совпадать.

Могут быть предложены различные пути практического осуществления алгоритма. Один из них состоит в том, что все q частных кодовых последовательностей разбиваются на пары, и для всех пар по отдельности определяется вид пар «поисковых» полиномов в соответствии с алгоритмом для кодовой скорости R=1/2.

Другой путь заключается в том, что выбирается одна из частных кодовых последовательностей (скажем, последовательность $\mathbf{u}_1$) и пары составляются поочередным включением в них этой выбранной кодовой последовательности и остальных q–1 частных кодовых последовательностей $\mathbf{u}_2 \div \mathbf{u}_q$.

Представляет интерес более быстрый путь поиска вида порождающих полиномов. Первоначально на основе анализа какой-либо пары частных кодовых последовательностей (скажем, $\mathbf{u}_1$ и $\mathbf{u}_2$) определяются «поисковые» полиномы $\mathbf{h}_1$ и $\mathbf{h}_2$. Далее учитывается тот факт, что любое из q равенств $\mathbf{m}(X)=\mathbf{u}_i(X)/\mathbf{g}_i(X)$, $i=1\div q$ можно переписать в виде: $\mathbf{g}_i(X)=\mathbf{u}_i(X)/\mathbf{m}(X)$. После определения вида порождающих полиномов $\mathbf{g}_1=\mathbf{h}_1$ и $\mathbf{g}_2=\mathbf{h}_2$ «автоматически» оказывается определенным и вид части исходной информационной последовательности **m**. Поэтому уже эту полученную последовательность можно использовать для непосредственного вычисления всех остальных порождающих полиномов, минуя длительную процедуру поиска их вида.

При сравнении свойств изложенных путей реализации алгоритма можно прийти к следующим выводам. Первый путь представляется более привлекательным, чем второй. Во втором пути при неудачном выборе частной кодовой последовательности, которая включается во все пары, или ошибки при нахождении вида $\mathbf{g}_1$, могут возникнуть ошибки при определении вида и остальных порождающих полиномов $\mathbf{g}_2\div\mathbf{g}_q$.

Однако время решения диагностической задачи по второму пути значительно короче. Действительно, если длина кодового ограничения равна K, то для каждой пары по первому пути возможно $2^K\times2^K=2^{2K}$ вариантов сочетаний различных поисковых полиномов. С учетом того, что нужно перебрать $q/2$ пар, общее число исследуемых вариантов полиномов будет составлять $N_1=(q2^{2K})/2$.

А по второму пути число вариантов сочетаний первой пары равно 2^{2K}. При поиске вида остальных порождающих полиномов исследуется только один полином в паре (т.к. вид другого полинома уже известен). Количество вариантов в этом случае будет равно 2^K. Таким образом, общее число исследуемых вариантов будет равно $N_2=2^{2K}+(q-2)2^K$. Нетрудно убедиться, что $N_2<<N_1$.

При использовании третьего пути время анализа сокращается в еще большей степени. Вначале на основе двух последовательностей (пусть $\mathbf{u}_1$ и $\mathbf{u}_2$) определяется вид двух «поисковых» полиномов $\mathbf{h}_1$ и $\mathbf{h}_2$ последовательности **m**. Затем производится деление оставшихся последовательностей $\mathbf{u}_3\div\mathbf{u}_q$ на **m**.

Поскольку раньше сравнение получаемых последовательностей $\mathbf{y}_1 \div \mathbf{y}_q$ происходило также после деления соответствующих полиномов, то число операций по рассматриваемому третьему пути будет равно: $N_3=2^{2K}+(q-2)$, очевидно, что $N_3<<N_2$.

Платой за ускорение операции анализа по второму и третьему пути будет увеличение возможности ошибочной диагностики. При неправильном определении хотя бы одной составляющей «поискового» полинома $\mathbf{h}_1$ по последовательности $\mathbf{u}_1$ частично неправильными будут и все остальные «поисковые» полиномы, определенные на его основе. При использовании третьего пути вероятность неправильной диагностики будет еще выше, так как сюда добавляется возможность неправильного восстановления последовательности $\mathbf{m}$.

Были рассмотрены три возможных пути диагностики для кодовых скоростей $R<1/2$, однако для двух других алгоритмов и последовательность операций, и выводы о сравнительных свойствах будут такими же.

На основе проведенного анализа возможных методов диагностики в последующих разделах для дальнейшего исследования выбран алгоритм 3.

2.2. Алгоритм диагностики при малом уровне шумов

При исследовании различных алгоритмов диагностики необходимо учитывать, что любое диагностическое решение вырабатывается всегда с определенной погрешностью, зависящей от различных обстоятельств. Погрешность выражается в том, что та структура порождающих полиномов, которая была получена после определенной обработки кодовых сигналов, не совпадает с истинной структурой используемых порождающих полиномов.

В последующем использовании полученного диагностического решения это должно привести к определенным потерям в зависимости от того, насколько полученное решение отличается от истинного. Однако это выходит за пределы собственно диагностических алгоритмов и в данной работе не рассматривается. В

качестве неправильного будет расцениваться такое диагностическое решение, в котором полученный вид порождающих полиномов имеет любые отличия от истинного их вида, независимо от количества этих отличий и их расположения.

Можно выделить две основные причины возможного появления погрешностей. Одна из них заключается в том, что времени анализа (используемой длины рассматриваемой выборки **m**) оказывается недостаточно. Вторая выражается в том, что при значительном уровне аддитивных шумов в основном теплового происхождения в реальной приемной аппаратуре после демодуляции возможно появления ошибочных символов. При этом какие-то двоичные символы будут менять свои значения на инверсные. Если такие события достаточно частые, то алгоритм может неправильно восстановить какие-либо части порождающих полиномов. Несмотря на то, что обе причины обусловлены случайными свойствами обрабатываемых сигналов, однако источники появления случайности в обоих случаях различаются. В первом варианте в общем случае значения символов исходной информационной последовательности можно как правило, считать взаимно независимыми и равновероятными. Во втором варианте случайность обусловлена свойствами шума, как статистического процесса.

Если первая причина устраняется необходимым увеличением времени анализа и не зависит от уровня шума, то вторая причина требует определенного изменения структуры диагностических алгоритмов.

Рассмотрим алгоритм, предназначенный для работы в условиях, когда при возникновении погрешностей доминирует первая причина («детерминированная» обработка), а влиянием шума можно пренебречь, а в параграфе 2.3. описывается алгоритм, предназначенный для работы в условиях воздействия значительных шумов. Сравнительная оценка погрешности диагностических решений при использовании этих алгоритмов также рассмотрена в параграфе 2.3.

Процедура алгоритма 3, рассмотренного в предыдущем параграфе, требует осуществления поиска вида полиномов **h**, при котором будет наблюдаться совпадение значений символов, формируемых с их помощью

последовательностей **y**. Реализация алгоритма возможна двумя методами. Будем рассматривать случай использования кодовой скорости R=1/2, необходимые подходы при других значениях кодовой скорости были рассмотрены в предыдущем разделе.

В обоих методах необходимо реализовать перебор 2^{2K} вариантов сочетаний видов «поисковых» полиномов $\mathbf{h}_1$ и $\mathbf{h}_2$, и для каждого устанавливать сходство символов, формируемых с их помощью последовательностей $\mathbf{y}_1$ и $\mathbf{y}_2$. Однако очередность выполнения этих операций в обоих методах различается.

Операции первого метода осуществляются в последовательности, укрупненная структурная схема которой изображена на рисунке 2.1. Первоначально общая кодовая последовательность $\mathbf{u}_0$ разделяется на первую $\mathbf{u}_1$ и вторую $\mathbf{u}_2$ частные кодовые последовательности. После начала процедуры перебора производится 2^{2K} циклов анализа. В первом цикле рассматривается первый вариант сочетания видов «поисковых» полиномов $\mathbf{h}_1$ и $\mathbf{h}_2$ (порядок нумерации вариантов сочетаний принципиальной роли не играет, важно, чтобы были рассмотрены все необходимые сочетания).

С помощью этих полиномов производится последовательное сверточное кодирование частных кодовых последовательностей $\mathbf{u}_1$ и $\mathbf{u}_2$ и в результате формируются последовательности $\mathbf{y}_1$ и $\mathbf{y}_2$. С появлением первых символов этих последовательностей они посимвольно сравниваются. В результате фиксации первого же несовпадения символов обеих последовательностей, расположенных в одинаковых позициях, процесс сверточного кодирования прекращается, алгоритм переходит к следующему циклу и генерируется следующий вариант сочетания видов полиномов $\mathbf{h}_1$ и $\mathbf{h}_2$.

При фиксации факта совпадений проверка с данным вариантом сочетания видов полиномов $\mathbf{h}_1$ и $\mathbf{h}_2$ продолжается по следующим символам последовательностей $\mathbf{u}_1$ и $\mathbf{u}_2$. Это обусловлено тем, что даже при «неправильном» сочетании полиномов $\mathbf{h}_1$ и $\mathbf{h}_2$ (т.е., несовпадающем с видом искомых полиномов $\mathbf{g}_1$ и $\mathbf{g}_2$) значения каких-либо символов сформированных последовательностей $\mathbf{y}_1$ и $\mathbf{y}_2$ могут случайно совпасть из-за независимости символов исходной

последовательности **m**, причем несколько совпадений может происходить подряд. Поэтому по результатам фиксации символов ведется счет количества подряд наблюдаемых совпадений. При «правильном» сочетании полиномов $\mathbf{h}_1$ и $\mathbf{h}_2$ совпадения будут случаться постоянно. А вероятность случайного числа подряд идущих совпадений при «неправильном» сочетании полиномов экспоненциально убывает в зависимости от этого числа. Поэтому когда она уменьшится до уровня, приемлемого для точности диагностики, будет установлено, что вид искомых порождающих полиномов $\mathbf{g}_1$ и $\mathbf{g}_2$ найден.

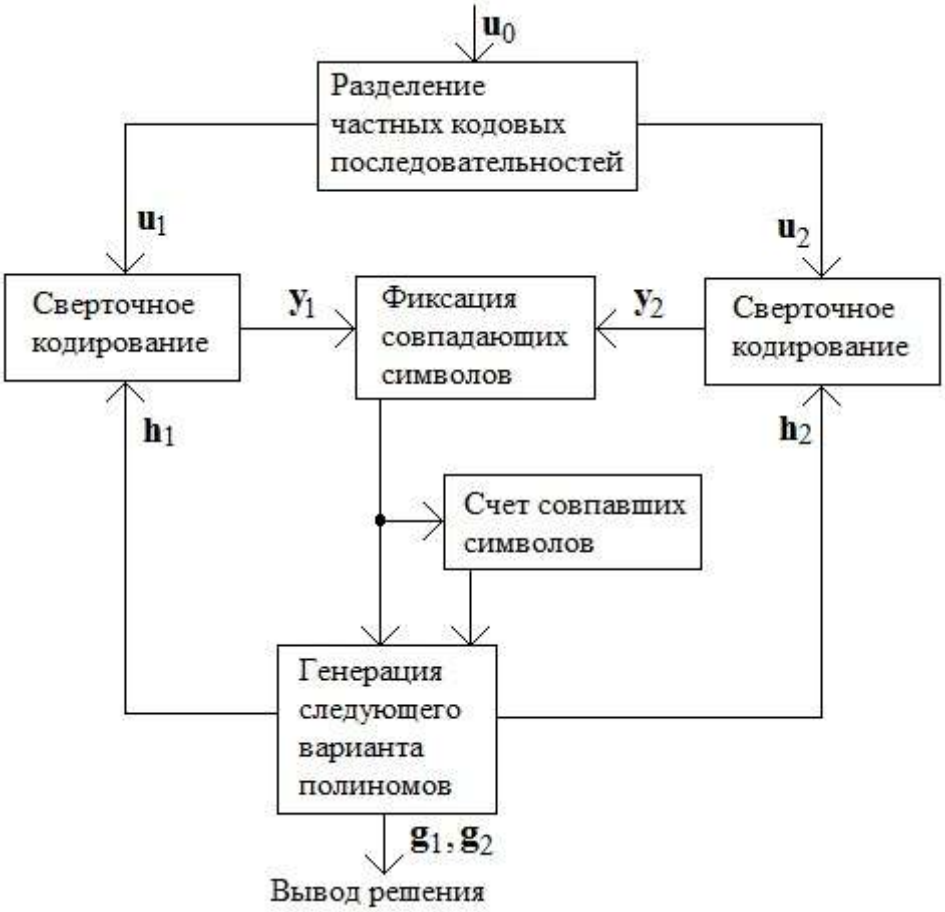

Рисунок 2.1

Таким образом, в этом методе для каждого из 2^{2K} сочетаний видов «поисковых» полиномов в каждом цикле производится перебор вариантов групп по K символов последовательностей $\mathbf{u}_1$ и $\mathbf{u}_2$, пока не будет принято решение о достижении одного из требуемых результатов. При этом варианты таких групп могут быть получены либо простым сдвигом символов этих последовательностей на один символ, либо последовательным сдвигом обеих последовательности сразу

на K символов. Для каждого нового сочетания видов «поисковых» полиномов в отдельных циклах может быть использована одна та же тестовая последовательность таких групп, заранее занесенная в память. Можно также реализовать алгоритм без использования тестовой последовательности, непрерывно подавая для анализа вновь принимаемые приемником символы.

Другой метод основан на другой очередности перебора вариантов. В нем уже для каждой новой группы из K символов последовательностей $\mathbf{u}_1$ и $\mathbf{u}_2$ выполняется свой цикл анализа, в котором перебираются все варианты сочетаний «поисковых» полиномов. При этом обработку результатов сравнения удобно производить в матричной форме. Структура, реализующая подобный метод, приведена на рисунке 2.2.

Полиномы $\mathbf{h}_1(X)$ и $\mathbf{h}_2(X)$ можно записать, как: $\mathbf{h}_1(X)=a_0+a_1X+a_2X^2+\ldots+a_{K-1}X^{K-1}$ и $\mathbf{h}_2(X)=b_0+b_1X+b_2X^2+\ldots+b_{K-1}X^{K-1}$, где коэффициенты a_i и b_i , ($i=1\div K$), могут принимать значения 0 или 1, и наборы значений этих коэффициентов определяют «индивидуальную» структуру каждого полинома.

Работа схемы на рисунке 2.2 по определению вида порождающих полиномов управляется блоком управления (БУ). Алгоритм в целом состоит из операций последовательного выполнения N однотипных циклов. Последовательность перебора вариантов сочетания «поисковых» полиномов может в каждом цикле быть реализована следующим образом.

В начале каждого цикла в сдвиговые регистры последовательно вводится очередная группа из K символов первой частной кодовой последовательности $\mathbf{u}_1(X)$ (в сдвиговый регистр СР1) и второй частной кодовой последовательности $\mathbf{u}_2(X)$ (в сдвиговый регистр СР2). С началом нового цикла вводится новая группа из K символов обеих кодовых последовательностей.

Эти частные кодовые последовательности получаются с помощью блока выделения кодовых последовательностей (БВКП), который определяет во входной кодовой последовательности $\mathbf{u}_0(X)$ пары кодовых символов, относящихся к общему информационному символу. Далее первый и второй символы в каждой

паре коммутатор (Комм.) разделяет по своим двум выходам, создавая кодовые последовательности $\mathbf{u}_1(X)$ и $\mathbf{u}_2(X)$.

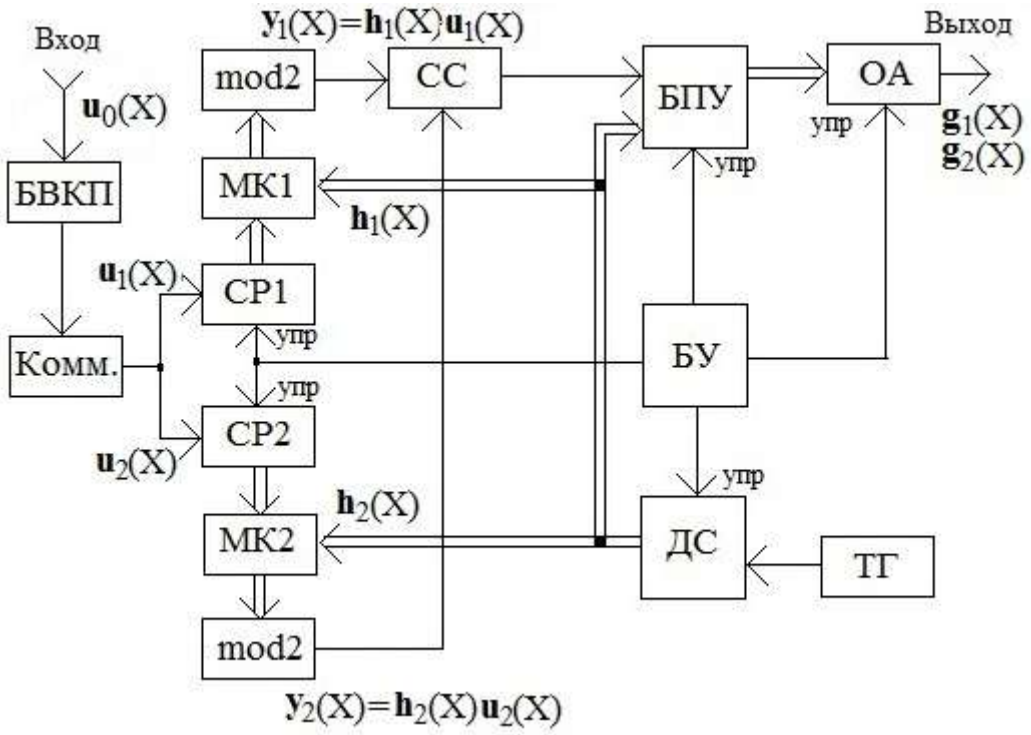

Рисунок. 2.2.

После начала цикла БУ запускает двоичный счетчик (ДС), который начинает последовательно считать импульсы тактового генератора (ТГ) и в двоичном коде выдает результаты счета на многоканальные ключи (МК1 и МК2). Результат выдается в форме двоичного кода с $2K$ разрядами, начиная с единицы, заканчивая числом 2^{2K-1} (т.е., начиная с $\underbrace{000\ldots01}_{2K}$, заканчивая числом $\underbrace{111..11}_{2K}$). Из этого числа K низших разрядов поступает на МК1, другие K высших разрядов поступают на МК2.

Каждый из МК1 и МК2 содержит K параллельных независимо открывающихся ключей. Сигнальные входы этих ключей подсоединены к выходам соответствующих разрядов сдвиговых регистров СД1 и СД2. На управляющий вход каждого ключа подается один из разрядов двоичного числа, вырабатываемого двоичным счетчиком ДС. Если значение этого разряда равно единице, тот ключ открывается, если нулю, то ключ закрыт. Таким образом,

любому сгенерированному ДС двоичному числу, как набору нолей и единиц, соответствует набор открытых и закрытых ключей в МК1 и МК2.

Выходные сигналы всех ключей МК1 и МК2 поступают на сумматоры по модулю 2 (mod2), где вычисляются соответствующие суммы. Набор открытых и закрытых ключей определяет, какие из K символов, записанных в сдвиговых регистрах, участвуют в формировании суммы по модулю 2, т.е. вид полиномов $\mathbf{h}_1$ и $\mathbf{h}_2$. Фактически, значения низших K разрядов и высших K разрядов двоичного числа, вырабатываемого двоичным счетчиком, равны, соответственно, значениям коэффициентов $a_0 \div a_{K-1}$ и $b_0 \div b_{K-1}$ этих полиномов.

Таким образом, в каждом цикле перебираются все возможные варианты сочетания вида «поисковых» полиномов, (кроме варианта, когда все коэффициенты $a_0 \div a_{K-1}$ и $b_0 \div b_{K-1}$ равны нулю, который не имеет смысла). В следующем цикле вновь перебираются эти варианты «поисковых» полиномов, и т.д.

Выходные сигналы сумматоров по модулю 2 ,т.е. результаты свертки двух групп из к символов, взятых из последовательностей $\mathbf{y}_1$ и $\mathbf{y}_2$, сравниваются по величине в схеме сравнения (СС). Она вырабатывает единичный сигнал, если они совпадают, и нулевой сигнал, если различаются, фактически осуществляя логическую операцию: $x_3 = x_1 x_2 \vee \bar{x}_1 \bar{x}_2$, где x_1 и x_2 – входные сигналы схемы сравнения, x_3 – ее выходной сигнал.

Результат сравнения поступает на блок памяти с умножением (БПУ). При запуске процесса диагностики перед первым циклом все ячейки БПУ предварительно заполняются единицами. Далее в каждом цикле сигнал схемы сравнения для каждого нового сочетания вариантов «поисковых» полиномов умножается на число, присутствующее в одной из ячеек БПУ. Адрес этой ячейки равен двоичному числу, сгенерированному ДС для этого сочетания $\mathbf{h}_1$ и $\mathbf{h}_2$.

После проведения N циклов блок управления завершает работу и с помощью блока определения адреса заполненных ячеек (блок ОА) выдает для дальнейшего использования результаты диагностики.

Поскольку в каждом цикле в сдвиговые регистры СД1 и СД2 заводятся новые группы символов, каждый из которых равновероятно принимает значения 1 и 0, то для всех «неправильных» вариантов «поисковых» полиномов, т.е. не совпадающих с видом порождающих полиномов $\mathbf{g}_1$ и $\mathbf{g}_2$, выходной сигнал схемы сравнения будет также равновероятно принимать оба эти значения. И только для искомого «правильного» варианта, определяемого соотношением: $\mathbf{h}_1=\mathbf{g}_2$ и $\mathbf{h}_2=\mathbf{g}_1$ выходной сигнал СС будет всегда равен единице.

В каждой ячейке с новыми циклами накапливается произведение результатов сравнения сочетания «поисковых» полиномов, определяемого адресом этой ячейки, причем для всех «неправильных» полиномов в этом произведении будут равновероятно присутствовать и единицы и нули. Поэтому, даже если появятся подряд несколько единиц, то достаточно быстро в сомножители добавится ноль, и содержимое ячейки обнулится. После каждого цикла количество ячеек БПУ с единичным содержанием будет постоянно убывать и по истечении определенного количества циклов обнулятся все ячейки кроме одной. Адрес этой ячейки определяет «правильный» вариант коэффициентов полиномов $\mathbf{h}_1$ и $\mathbf{h}_2$, т.е. искомую структуру порождающих полиномов $\mathbf{g}_1$ и $\mathbf{g}_2$. Он, как результат процедуры диагностики, выводится для дальнейшего использования.

Для исследования работы данного алгоритма было проведено компьютерное моделирование с помощью программного комплекса [99]. Моделирование проводилось следующим образом. Передаваемый информационный сигнал имитировался случайным двоичным потоком с равновероятным появлением нулей и единиц. Далее эта последовательность символов кодировалась сверточным кодом с заданной структурой $\mathbf{g}_1$ и $\mathbf{g}_2$. После этого осуществлялись операции описанного алгоритма. (В данном параграфе воздействие шумов не учитывалось, т.е. полагалось, что вероятность появления ошибочных символов значительно меньше, чем требуемая точность диагностики.) Когда в БПУ остается один ненулевой элемент, его адрес выводился, как результат диагностики.

Для иллюстрации процесса «очищения» БПУ от неправильных вариантов «поисковых» полиномов в качестве примера приведены рисунки 2.2.3-2.2.7, отражающие текущие состояния БПУ при отыскании «правильного» сочетания «поисковых» полиномов. Для наглядности структура памяти БПУ представлена в виде матрицы, у которой десятичные номера строк и столбцов равны, соответственно *K* низшим и *K* высшим разрядам генерируемого ДС двоичного числа.

Рисунки приведены для сверточных кодов с различными используемыми кодовыми ограничениями. Рисунки 2.3.a- 2.3.e – для кодового ограничения К=3 и кода (5,7).

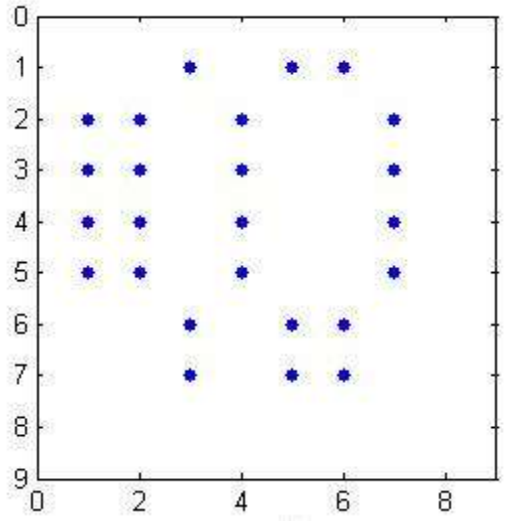

Рисунок. 2.3.a.

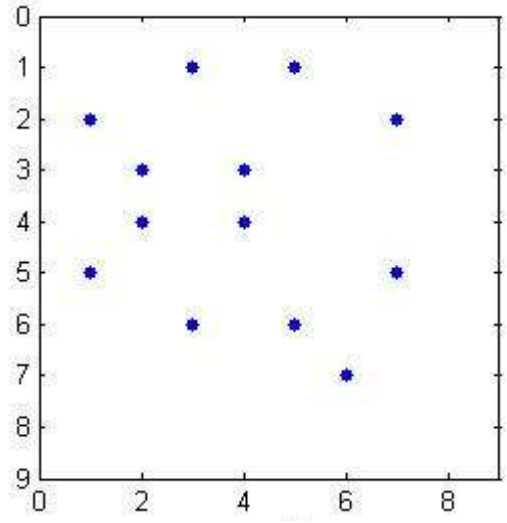

Рисунок. 2.3.b.

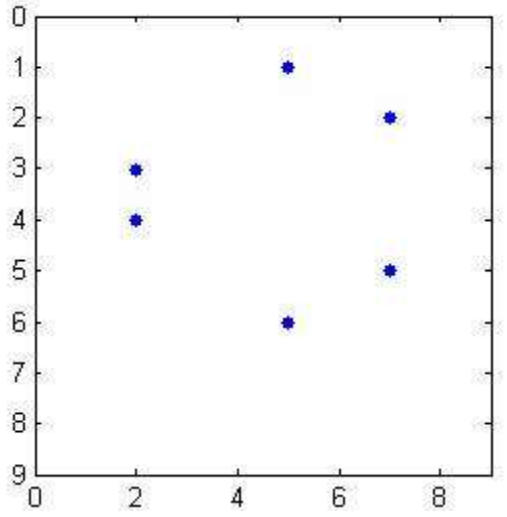

Рисунок. 2.3.c.

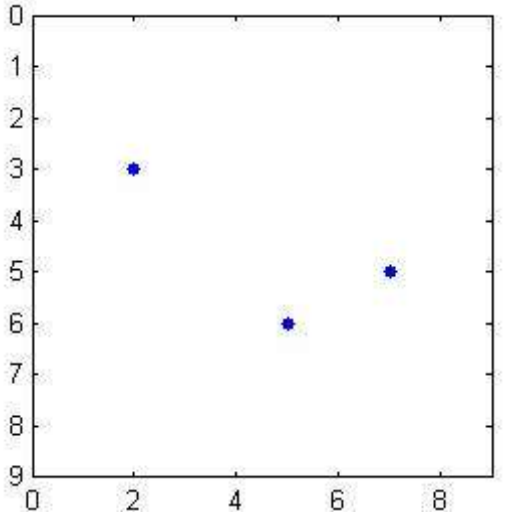

Рисунок. 2.3.d.

Буквы a,b,c,d соответствуют последовательному изменению содержимого ячеек БПУ после каждого нового цикла, начиная с первого. Точки указывают на единичные значения в ячейках соответствующего номера. Номера столбцов и строк для удобства приведены в десятичной системе счисления. В данном примере процедура диагностики заняла всего пять циклов, после чего получен правильный вид используемых для кодирования порождающих полиномов.

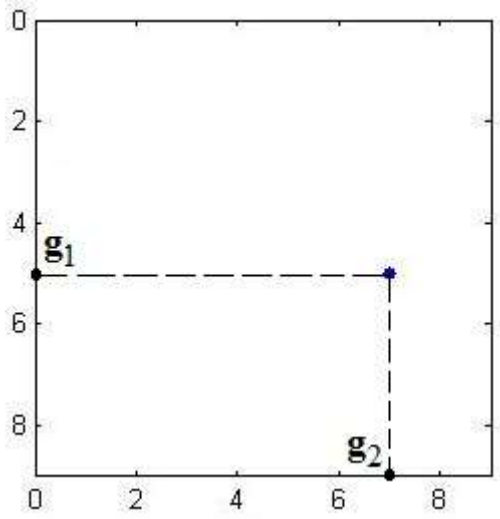

Рисунок. 2.3.e.

В последующих сериях рисунков приведены примеры диагностики кодов с кодовыми ограничениями до величины K=7 для часто используемых при таких ограничении видах порождающих полиномов [4, 18, 26].

Рисунки 2.4.a–2.4.f описывают процедуру диагностики при кодовом ограничении K=4. Здесь использован код (13,15). Результат на рисунке 2.4.f. получается в десятичном обозначении кода, $(11,13)_{10}=(13,15)_8$.

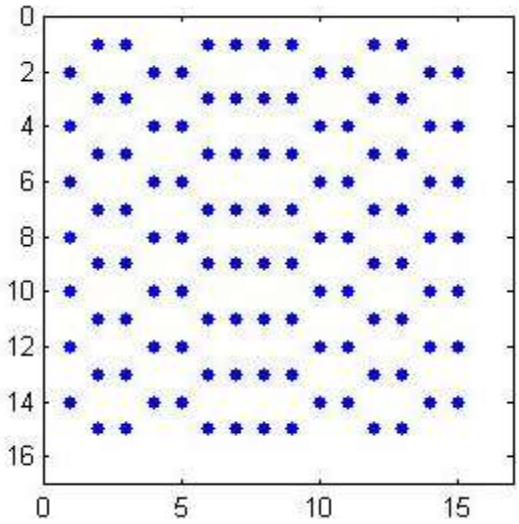

Рисунок 2.4.a.

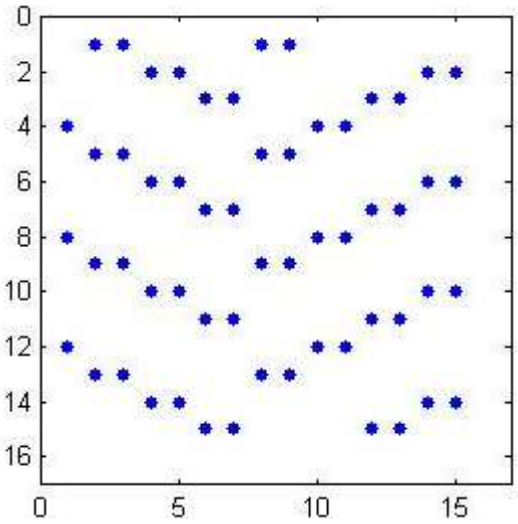

Рисунок.2.4.b.

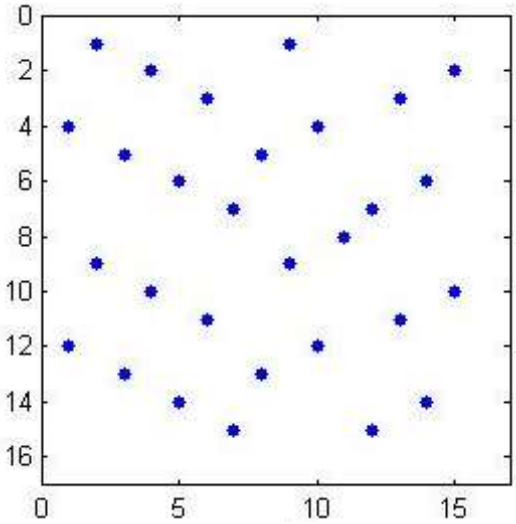

Рисунок. 2.4.c.

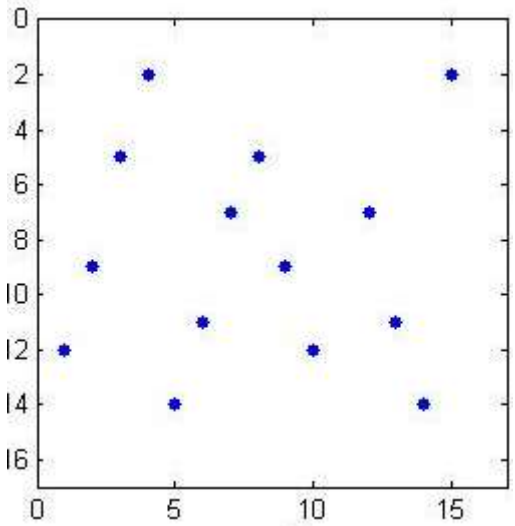

Рисунок. 2.4.d.

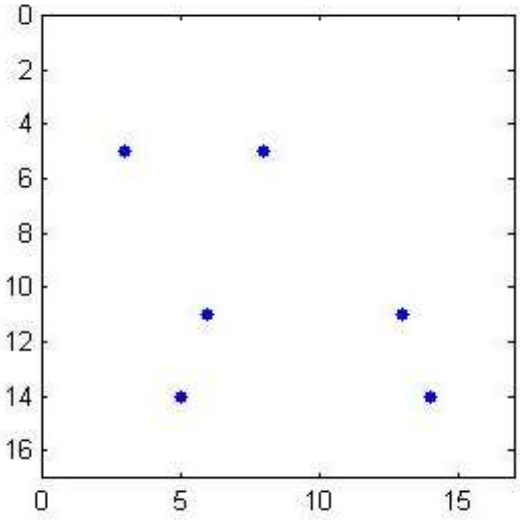

Рисунок. 2.4.e.

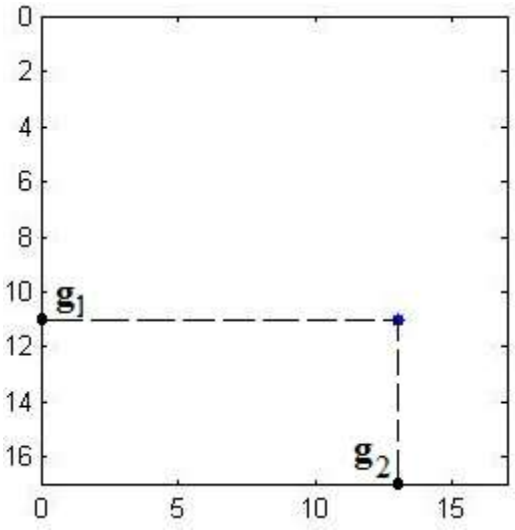

Рисунок. 2.4.f.

На рисунках 2.5.a.–2.5.f. представлен последовательный вид матрицы после разных циклов для кодового ограничения K=5 и кода $(25,37)_8=(21,31)_{10}$.

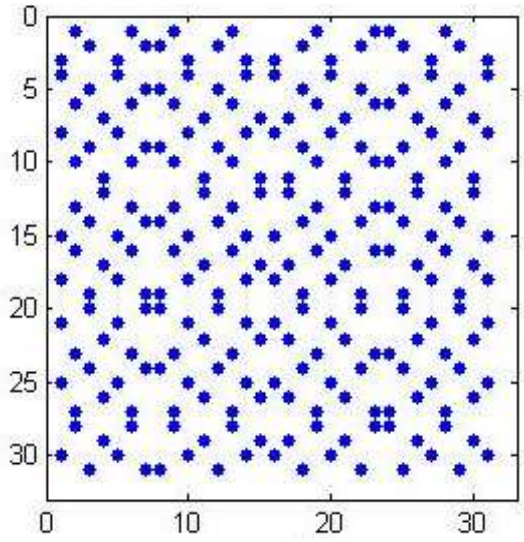

Рисунок. 2.5.a. 2-й цикл

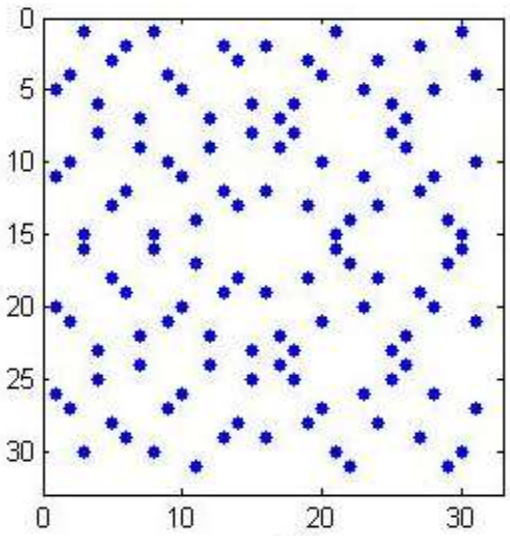

Рисунок. 2.5.b.4-й цикл

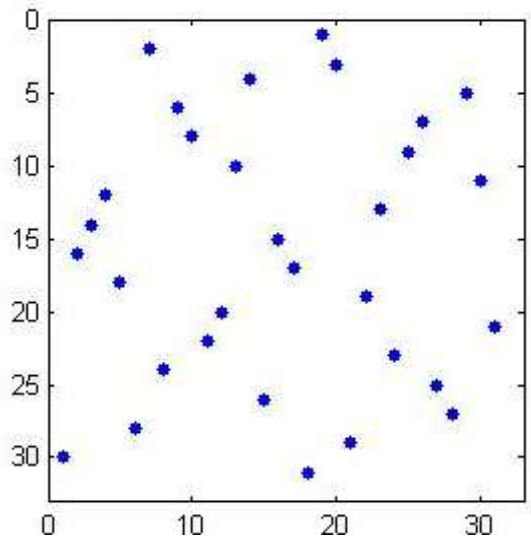

Рисунок. 2.5.c. 6-й цикл

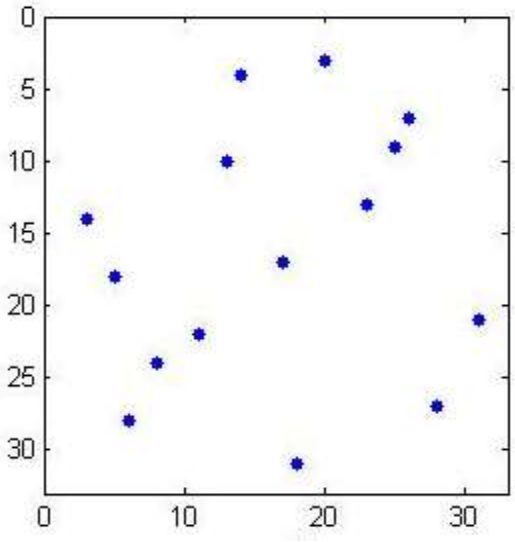

Рисунок. 2.5.d. 7-й цикл

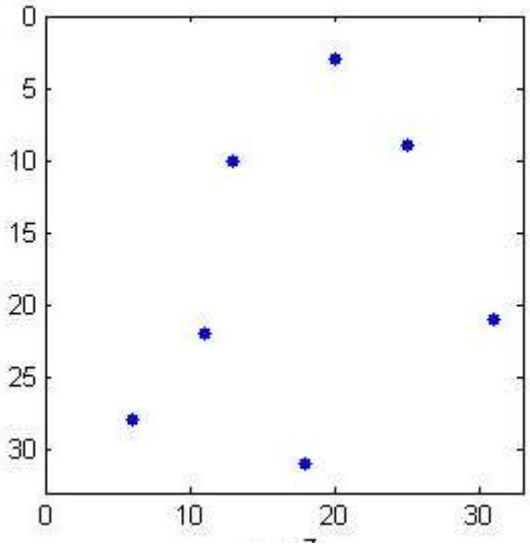

Рисунок. 2.5.e. 8-й цикл

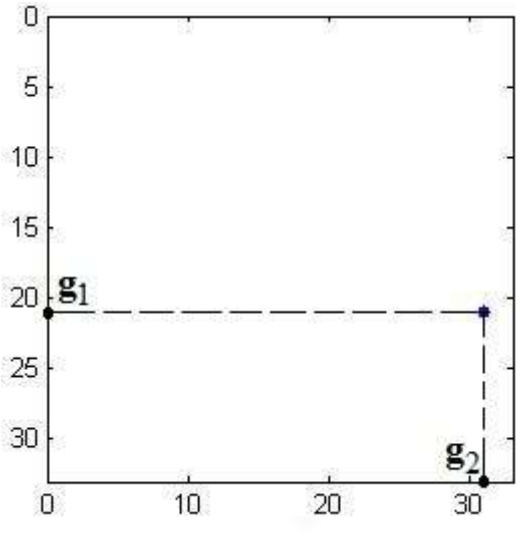

Рисунок. 2.5.f. 10-й цикл

На рисунках 2.6.a.–2.6.f. представлен последовательный вид матрицы после разных циклов для кодового ограничения K=6 и кода $(57,65)_8=(47,53)_{10}$.

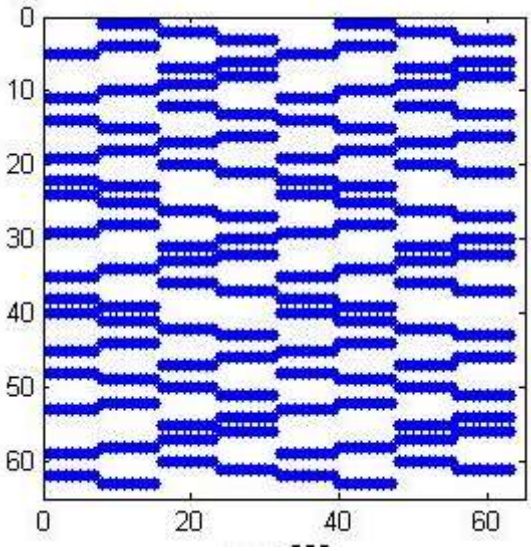

Рисунок. 2.6.a. 2-й цикл

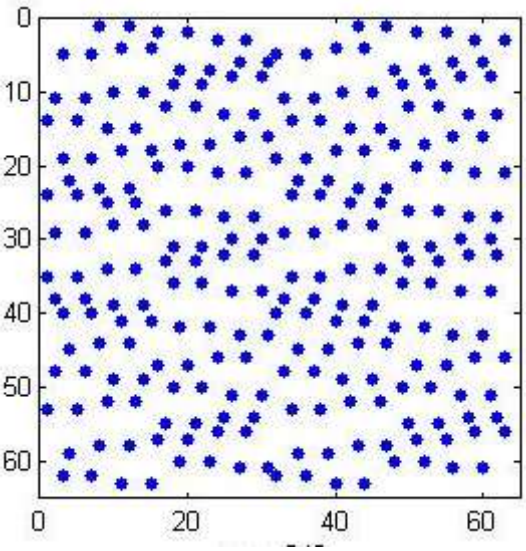

Рисунок. 2.6.b. 4-й цикл

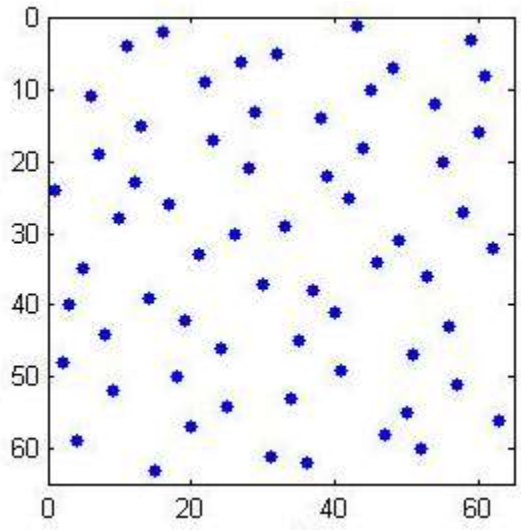

Рисунок. 2.6.c. 7-й цикл

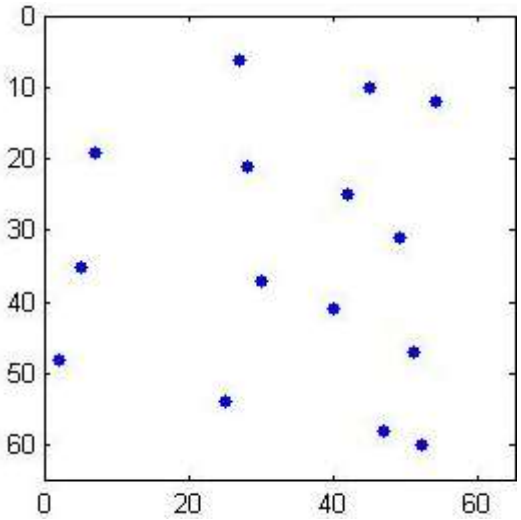

Рисунок. 2.6.d. 9-й цикл

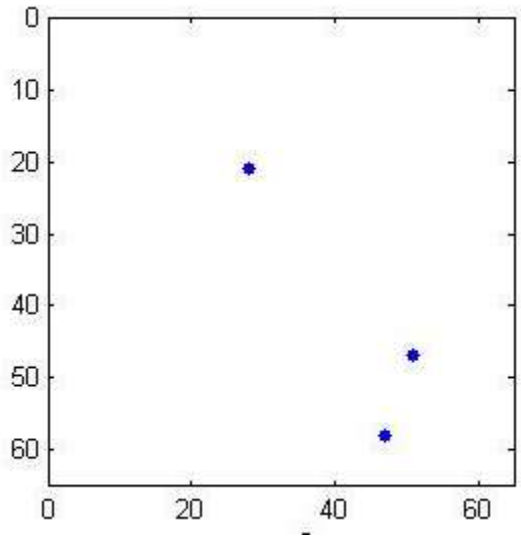

Рисунок. 2.6.e. 11-й цикл

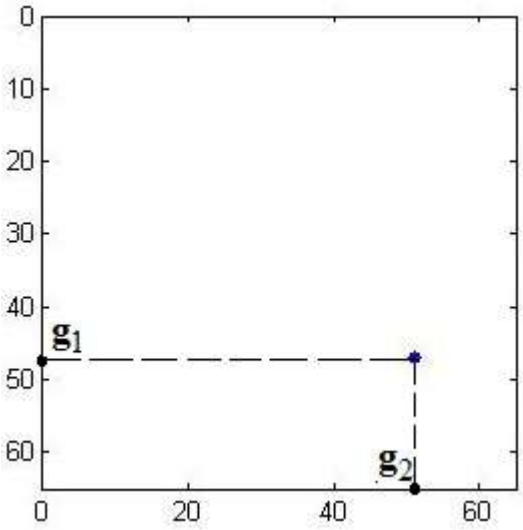

Рисунок. 2.6.f. 12-й цикл

На рисунках 2.7.a.–2.7.f. представлен последовательный вид матрицы после разных циклов для кодового ограничения K=7 и кода $(133,171)_8=(91,121)_{10}$.

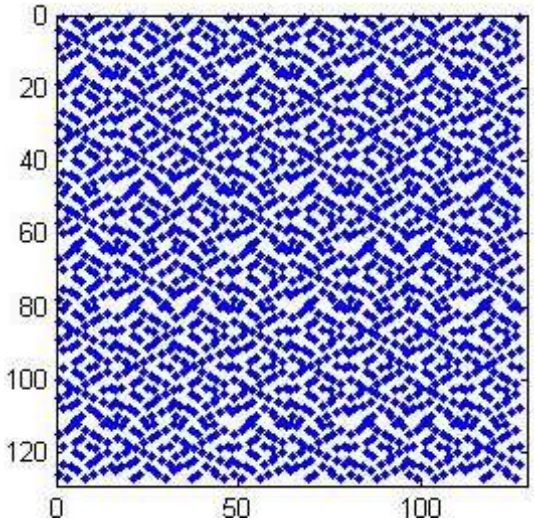

Рисунок. 2.7.a. 3-й цикл

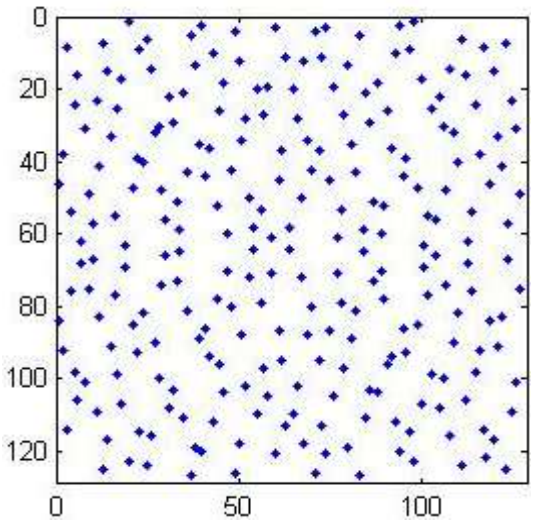

Рисунок. 2.7.b. 7-й цикл

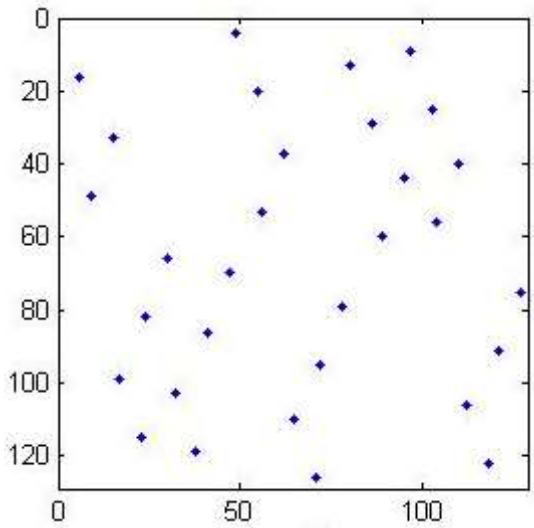

Рисунок. 2.7.c. 9-й цикл

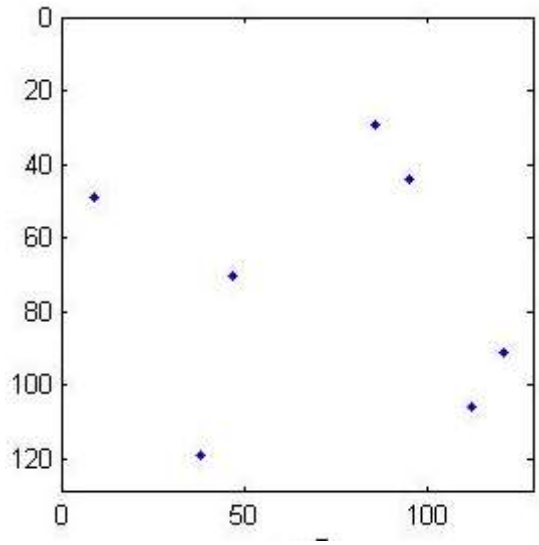

Рисунок. 2.7.d. 11-й цикл

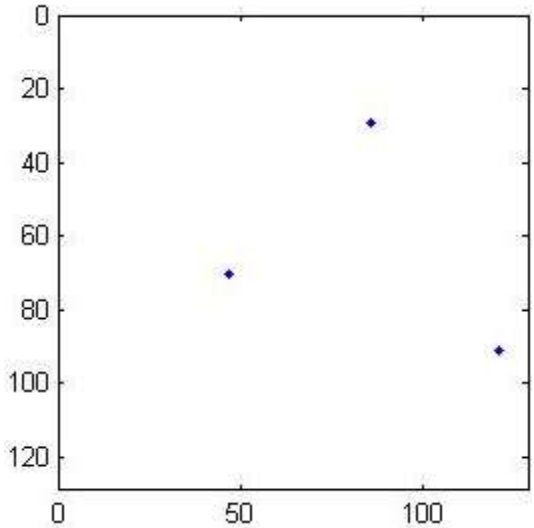

Рисунок. 2.7.e. 13-й цикл

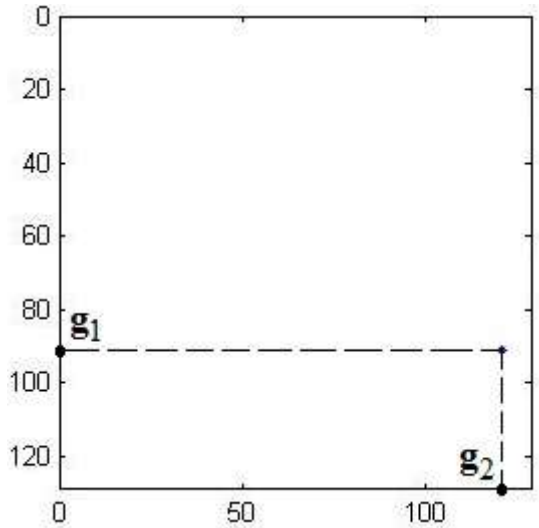

Рисунок. 2.7.f. 14-й цикл

Таким образом, алгоритм находит «правильный» вид «поисковых» полиномов, совпадающий с видом искомых порождающих полиномов, за относительно небольшое количество циклов.

Прерывание работы процедуры может быть организовано двумя путями. Первый путь заключается в том, что осуществляется заранее оговоренное количество циклов, а по их завершении принимается решение об удачной диагностике, если в БПУ осталась только одна ячейка с ненулевым содержимым. Если не осталось ни одной такой ячейки, то произошел сбой, вероятной причиной которого является воздействие шумов (подобная ситуация будет рассмотрена в следующем параграфе). Если в матрице осталось больше одной ячейки с ненулевым содержимым, то либо также причиной является воздействие шумов, либо оговоренного проведенного количества циклов оказалось недостаточно. В этом случае вся процедура диагностики может быть повторена.

Рассмотрим вероятность этих исходов.

Обозначим $Q=2^{2K}-2$ – количество ячеек с «неправильными» полиномами с учетом «правильной» ячейки и ячейки с двоичным числом, равным нулю. Тогда вероятность того, что в какой-то одной «неправильной» ячейке после N циклов останется единица, будет равна $0{,}5^N$, а вероятность необходимости повторной диагностики, т.е. вероятность того, что хотя бы в одной из всех «неправильных» Q ячеек останется единица: $P_{\text{Н}}= [1-(1-0{,}5^N)^Q]$. Если N достаточно велико, то $P_{\text{Н}}\approx Q\cdot 0{,}5^N=(2^{2K}-2)\cdot 0{,}5^N\approx 0{,}5^{(-2K-N)}$. На рисунке 2.8 представлены графики зависимости величины $P_{\text{Н}}$ от количества циклов N при различном кодовом ограничении K.

Количество циклов, после которого будет принято диагностическое решение, также является случайной величиной. Распределение вероятности $P_{\text{С}}$ его величины можно рассчитать следующим образом. Она равна вероятности $P_{\text{С}}=P_1P_2$ совместного совершения событий: того, что за N циклов единственность диагностического решения еще не будет достигнута, $P_1=1-(1-0{,}5^N)^Q$; и того, что за $N+1$ циклов она достигнута уже будет, с вероятностью $P_2=(1-0{,}5^{(N+1)})^Q$. Графики

вероятности P_C необходимого количества циклов, которые нужно провести для выработки диагностического решения при различных значениях кодового ограничения, представлены на рисунке 2.9.

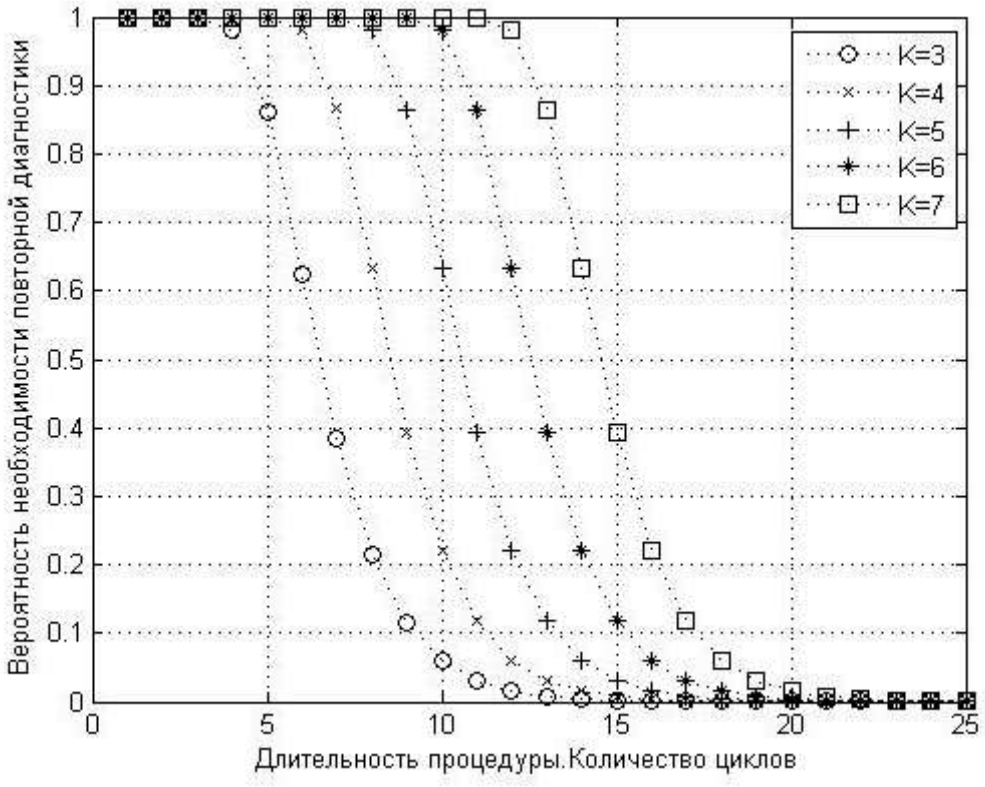

Рисунок. 2.8.

Необходимо сравнить полученные результаты с предыдущим методом реализации алгоритма, рассмотренном в этом параграфе.

В предыдущем методе для каждого из 2^{2K} вариантов сочетаний «поисковых» полиномов последовательно в каждом цикле производилось многоразовое повторное сверточное кодирование различных групп первой и второй кодовых последовательностей и сравнивались результаты кодирования. Эта операция производилась до тех пор, пока результаты не будут совпадать (в случае несовпадения происходит переход к другому варианту полиномов). Если же совпадение будет наблюдаться в течение оговоренного количества N операций, то принимается решение, что искомое сочетание «поисковых» полиномов найдено.

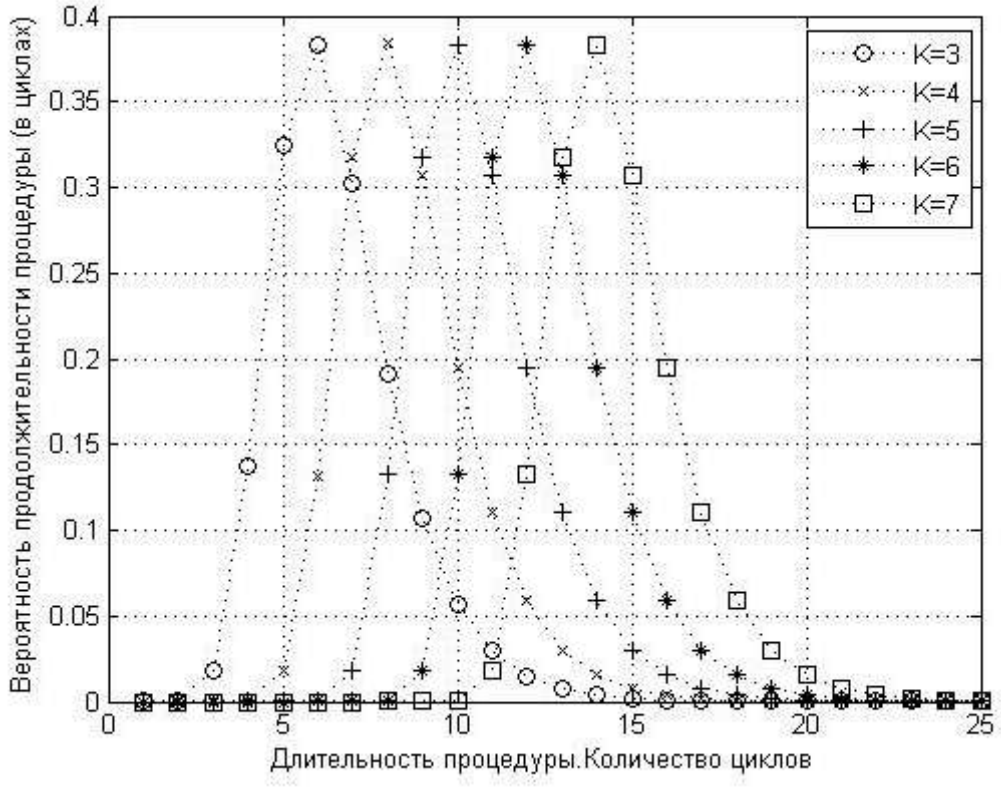

Рисунок. 2.9.

Поскольку результат сравнения для «неправильных» полиномов равновероятно принимает значения 1 и 0, то вероятность ошибки диагностического решения, когда за «правильное» сочетание будет принято «неправильное», будет равна $P_{E1}=0{,}5^N$. Это определяет выбор значения N исходя из допустимой величины диагностической ошибки P_{E0}. Таким образом, количество проверок должно удовлетворять условию: $N \geq \log_2 P_{E0}$.

Среднее количество операций сравнения, приходящееся на каждый вариант сочетаний «поисковых» полиномов равно: $\sum_{i=1}^{N} i 0{,}5^i \approx \sum_{i=1}^{\infty} i 0{,}5^i = 2$. «Правильный» вариант сочетания «поисковых» полиномов может равновероятно оказаться в любом месте массива из Q сочетаний, следовательно, среднее количество операций сравнения при работе с использованием первого метода равно: $N_1=2\cdot 0{,}5Q+N\approx 2^{2K}+N$.

При использовании второго метода в каждом цикле производится Q операций, однако после каждого цикла количество ячеек с единичным

содержанием уменьшается в среднем вдвое. Следовательно, если не прерывается работа после оговоренного количества N циклов, то среднее количество циклов будет равно $\log_2 Q \approx 2K$, а среднее количество операций сравнения $N_2 = 2K \cdot Q \approx K \cdot 2^{(2K+1)}$. Количество операций сравнения по второму методу больше, чем по первому методу, однако второй метод имеет определенные преимущества. Недостатки первого метода заключаются в том, что при любом значении N диагностическое решение все равно выносится с определенной ошибкой. Кроме того, циклы анализа каждого сочетания вариантов «поисковых» полиномов все время включают различное количество операций сравнения, что может быть неудобно технически.

При втором методе, если количество циклов N оказалось недостаточно (ненулевая ячейка осталась не одна), то алгоритм сообщает, что цель диагностики не достигнута, что принципиально отличается от ошибочного решения. В этом случае возможно алгоритм просто продлить или повторить, что, очевидно, должно выглядеть более привлекательным для дальнейшего использования диагностического решения.

Таким образом, с использованием описываемого алгоритма при незначительном уровне шумов возможно получать диагностические решения с учетом требований по точности диагностики.

2.3. Алгоритм диагностики при значительном уровне шумов

При значительном уровне шумов приемной аппаратуры появляются факторы, осложняющие работу диагностического алгоритма. В результате воздействия шумов после детектирования появляются ошибочные символы, которые не соответствуют правильной работе сверточного кодера в передатчике и искажают работу процедуры получения диагностического решения. Однако и в этом случае общая схема алгоритма, реализуемая схемой на рисунке 2.2, остается применимой после модификации некоторых блоков.

Рассмотрим подробнее воздействие шумов при различных операциях алгоритма. При сравнении повторного кодирования частных кодовых последовательностей с помощью «поисковых» полиномов влияние шумов в случае «неправильных» видов полиномов на конечном результате не сказывается. Действительно, пусть вероятность ошибки символа равна $P_{\text{В}}$. Это означает, что нулевые выходные сигналы схемы сравнения будут появляться не с вероятностью 0,5, а с вероятностью $0{,}5–0{,}5P_{\text{В}}$, а вместо них с вероятностью $0{,}5P_{\text{В}}$ будут появляться единицы. (Коэффициент 0,5 перед $P_{\text{В}}$ появляется из-за того, что отличие кодовой группы с ошибочным символом от соответствующей безошибочной такой же кодовой группы может равновероятно как изменить результат свертки и «поисковым» полиномом, так и не менять его.) Но и единичные выходные сигналы схемы сравнения будут появляться не с вероятностью 0,5, а с вероятностью $0{,}5–0{,}5P_{\text{В}}$, а вместо них с вероятностью $0{,}5P_{\text{В}}$ будут появляться нулевые результаты. Таким образом, произойдет взаимная компенсация изменений значения вероятности и алгоритм в части реакции на «неправильные» сочетания «поисковых» полиномов не изменится.

Если алгоритм не будет модифицирован, изменения произойдут в части определения «правильного» сочетания видов полиномов. В предыдущем подразделе сравнение «правильных» полиномов всегда давало единичный результат. Теперь же, если какой-либо символ из группы K символов частных кодовых последовательностей, которые в данный момент обрабатываются, принят ошибочно, то результат сравнения может равновероятно принять и единичный, и нулевой результат. В случае такого нулевого результата обнулится и все единичное произведение, ранее накопленное в ячейке с адресом «правильных» полиномов.

Если окончание работы процедуры производится после выполнения фиксированного числа N циклов, то ее результатом будет либо недостижение диагностического решения, когда все ячейки содержат нулевые результаты, либо ошибочное диагностическое решение, если хотя бы одна ячейка с «неправильным» адресом еще не обнулилась.

Вероятность того, что за N циклов хотя бы один из кодовых символов является ошибочным, равна $P_1=1–(1–P_E)^{KN}$. Поскольку ошибочный символ может равновероятно привести и к единичному, и к нулевому результату сравнения, тот вероятность того, что за N циклов «правильная» ячейка обнулится хотя бы один раз, равна $0{,}5P_1$. Естественно, если обнуление произойдет не в последний цикл, то далее в «правильной» ячейке вновь будет накапливаться единичный результат, но по истечению циклов он будет меньше N.

Возникновение единиц в «неправильных» ячейках может привести к двум ситуациям. Одна из них – неправильная диагностика, если такая ячейка будет единственной. Другая ситуация – отсутствие диагностического решения за количество циклов, равное N, если таких ячеек будет больше одной. Вероятность того, что останется единичной одна из «неправильных» ячеек, равна $P_2=P_1(1–P_1)^{Q–1}$. Тогда вероятность неправильной диагностики равна $P_E=0{,}5P_1P_2$.Вероятность того, что диагностического решения за N циклов не будет выработано вообще, равна $P_3=0{,}5P_1(1–P_2)$.

При значительных уровнях шума, когда P_B составляет заметную величину, вероятность диагностической ошибки также может стать недопустимо большой.

В таких условиях необходимо модифицировать алгоритм. Общая структура алгоритма и последовательность большинства операций остается прежней, как описана в предыдущем подразделе на рисунке 2.2. Модификация касается действий, выполняемых блоком БПУ. В блоке БПУ теперь производится не умножение каждого нового результата сравнения на содержимое ячейки соответствующего номера, а арифметическое суммирование нового результата сравнения с содержимым ячейки.

Таким образом, появление ошибочного результата при сравнении «правильных» полиномов уже не приводит к срыву нормальной работы, а лишь несколько снижает величину суммы в «правильной» ячейке.

После завершения процедуры средняя величина суммы, накопленной в «правильной» ячейке, станет равна $N(1–P_E)\approx N$, а средняя сумма, накопленная во всех остальных ячейках – $0{,}5N$. Если количество циклов значительное, то эти

цифры будут заметно различаться, и блок ОУ без труда выберет ячейку с «правильным» номером.

Возникновение сбоя в работе алгоритма возможно, когда в какой-либо из «неправильных» ячеек накопится сумма, равная N. Значения сумм, накопленных в «неправильных» ячейках, подчиняются биномиальному распределению, которое имеет достаточно большое значение дисперсии.

На рисунках 2.10-2.13 в качестве примера приведены в графическом виде значения содержимого ячеек БПУ для различных кодов и разного количества циклов анализа.

На рисунках по горизонтальным осям отложены в десятичной системе счисления номера строк и столбцов матрицы. По вертикальной оси отложены суммы, которые присутствуют в ячейках матрицы после N циклов анализа. Обозначения использованных кодов приведены в восьмеричной системе счисления. Поскольку среднеквадратическое значение распределения сумм пропорционально квадратному корню из числа циклов, то отнесенное к среднему их значению, оно убывает достаточно медленно. Заметно, что при использовании кода с большим кодовым ограничением необходимо использовать большее количество циклов.

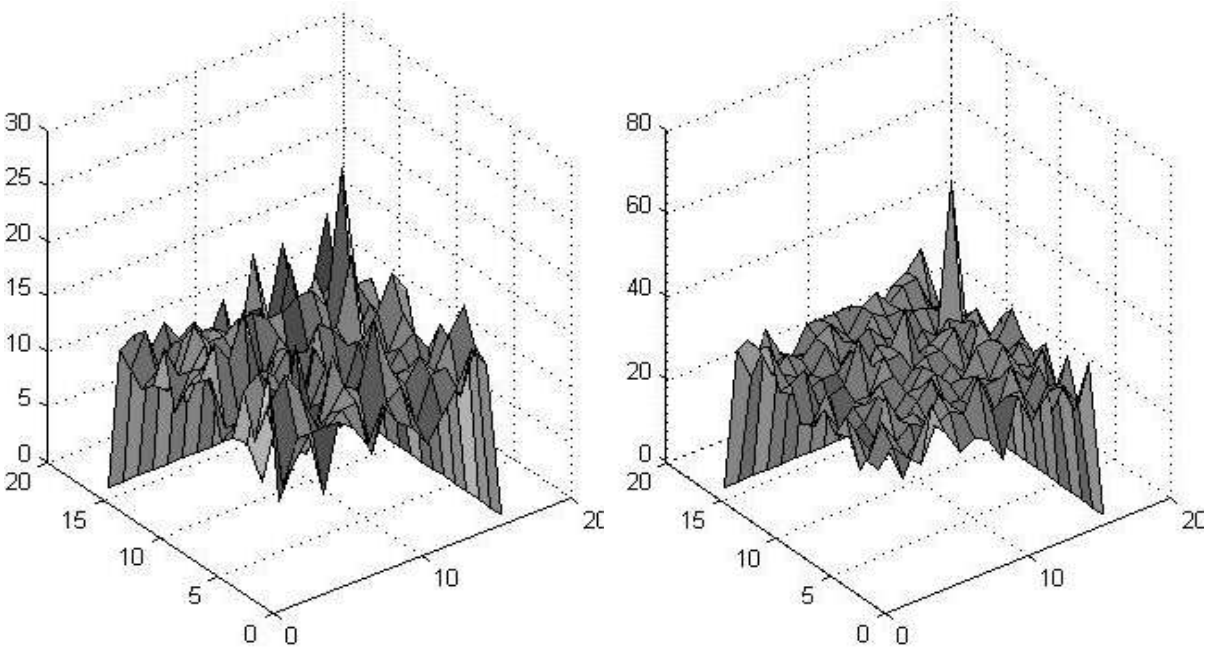

Рисунок. 2.10. N=10, код $(13,15)_8$

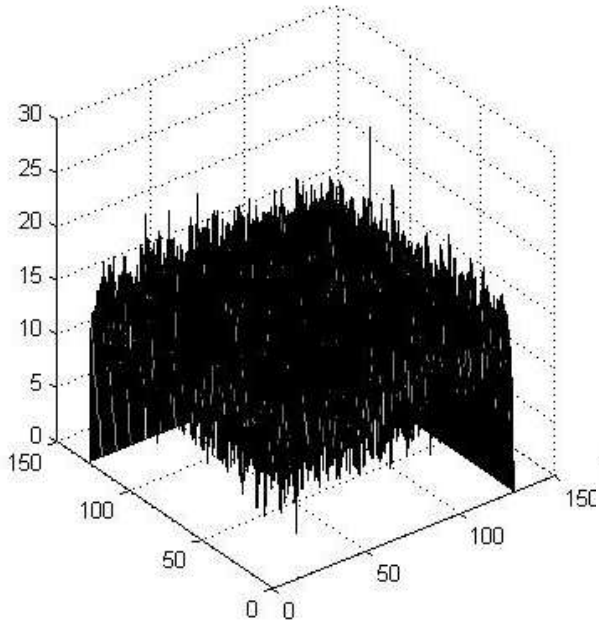

Рисунок. 2.11. N=50, код $(13,15)_8$

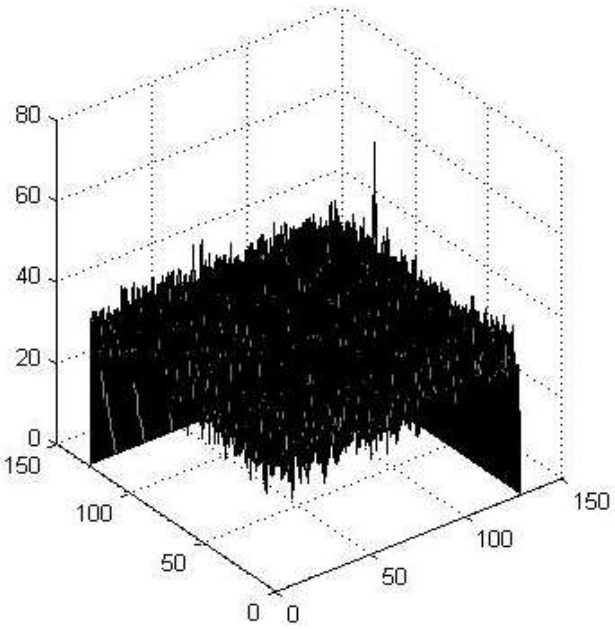

Рисунок. 2.12. N=10, код $(133,171)_8$

Рисунок. 2.13. N=50, код $(133,171)_8$

При этом если в «правильной» ячейке тоже будет число N, то алгоритм сообщит, что решения получить не удалось, вероятность такого исхода равна $(1-P_E)^N \cdot 2^{2K-N}$. Если же в «правильную» ячейку хотя бы один раз будет добавляться ноль, то в ней будет находиться сумма, равная N–1, то есть меньшая, чем в неправильной ячейке и в качестве решения алгоритм укажет ошибочный адрес и вид «поисковых» полиномов. Вероятность этого события приблизительно равна $(N-1)P_E \cdot 2^{2K-N}$. Для того, чтобы вероятность ошибочного решения была малой, количество циклов в процедуре анализа должно быть заметно больше удвоенного кодового расстояния.

Описываемый модифицированный алгоритм требует большего числа циклов анализа, чем алгоритм в его «детерминированном» варианте из подраздела 2.2. В принципе в условиях работы при воздействии шумов оба алгоритма могут быть объединены. Например, процедура анализа по «детерминированному»

алгоритму производится два или три раза, а полученное содержимое ячеек складывается по одинаковым номерам ячеек. В этом случае вероятность сбоя в работе будет равно, соответственно $N^2P_E^2$ $N^3P_E^3$, т.е достаточно малая даже при значительном уровне шумов.

2.4. Алгоритм определения кодового ограничения

Рассмотренные в предыдущих параграфах алгоритмы диагностики предполагали, что величина кодового ограничения K заранее известна, и в своей работе использовали эту информацию. Предварительное знание величины K необходимо, т.к. ее неправильная оценка приведет к неверным решениям. В данном параграфе рассмотрим вопросы определения величины K.

Первоначально исследуем подробнее ситуацию, если используемое значение кодового ограничения K_e отличается от истинного. Если $K_e < K$, то в любом случае полученный вид «поисковых» полиномов будет отличаться от правильного, т.к. искомые порождающие полиномы $\mathbf{g}_1$ и $\mathbf{g}_2$ содержат K слагаемых, а полученные будут содержать всего K_e слагаемых и часть слагаемых полинома пропадет.

Другая ситуация складывается, если $K_e > K$. Описанный в предыдущих подразделах алгоритм предлагает вид поисковых полиномов $\mathbf{h}_1(X)$ и $\mathbf{h}2(X)$, как решение уравнения:

$$\mathbf{h}_1(X)\mathbf{g}_1(X)\mathbf{m}(X)=\mathbf{h}_2(X)\mathbf{g}_2(X)\mathbf{m}(X). \qquad (2.4)$$

Если $K_e=K$, решение будет единственным, но если $K_e>K$ возможно появление нескольких решений. Вид полиномов $\mathbf{g}_1(X)$ и $\mathbf{g}_2(X)$, естественно, должен различаться, при этом для них выбирают «простые» полиномы, т.е. такие, которые не делятся нацело ни на какие полиномы кроме самих себя и единицы. Если же порождающие полиномы представляют собой произведение каких-либо простых полиномов, то необходимо, чтобы у них не было общего делителя

меньшей степени. Действительно, если бы у них был некоторый общий полиномиальный делитель, например, $\mathbf{g}_C(X)$, то их можно было бы записать в виде: $\mathbf{g}_1(X)=\mathbf{g}_C(X)\mathbf{g}_{D1}(X)$, $\mathbf{g}_2(X)=\mathbf{g}_C(X)\mathbf{g}_{D2}(X)$. В этом случае две передаваемые частные кодовые последовательности:

$$\mathbf{u}_1(X)=\mathbf{g}_1(X)\mathbf{m}(X)=\mathbf{g}_C(X)\mathbf{g}_{D1}(X)\mathbf{m}(X)=\mathbf{g}_{D1}(X)\mathbf{m}_C(X),$$
$$\mathbf{u}_2(X)=\mathbf{g}_2(X)\mathbf{m}(X)=\mathbf{g}_C(X)\mathbf{g}_{D2}(X)\mathbf{m}(X)=\mathbf{g}_{D2}(X)\mathbf{m}_C(X),$$

где $\mathbf{m}_C(X)=\mathbf{g}_C(X)\mathbf{m}(X)$, т.е. по каналу передачи передается некоторая модифицированная последовательность $\mathbf{m}_C(X)$, кодированная порождающими полиномами $\mathbf{g}_{D1}(X)$ и $\mathbf{g}_{D2}(X)$. Но кодовые ограничения рассматриваемых полиномов $\mathbf{g}_C$ и $\mathbf{g}_{D1}$, $\mathbf{g}_{D2}$ равны, соответственно, K_C и $K_{D1}=K_{D2}$, причем $K_CK_{D1}=K$, $K_CK_{D2}=K$, при этом $K_{D1}=K_{D2}<K$.

Это означает, что несмотря на то, что передаваемый сигнал технически кодируется сверточным кодером с кодовым ограничением, равным K, фактически это приводит к тому, что получаемая кодовая последовательность оказывается кодированной кодом с меньшим кодовым ограничением, значительно менее помехоустойчивым. Кроме этого, как упоминалось ранее, при этом возможно возникновение катастрофических ошибок.

Если полагать степень «поисковых» полиномов $K_e>K$, то возможно решение уравнения (2.4) в следующем виде: $\mathbf{h}_1(X)=\mathbf{h}_0(X)\mathbf{g}_2(X)$, $\mathbf{h}_2(X)=\mathbf{h}_0(X)\mathbf{g}_1(X)$, где $\mathbf{h}_0(X)$ – некоторый общий дополнительный полином степени $K_a=K_e-K$. В зависимости от величины K_e возможны несколько различных полиномов, дающих несколько вариантов решения уравнения (2.4.1). Например, если $K_a=2$, то возможны полиномы с первой и второй степенью, первая степень соответствует виду $\mathbf{h}_0(X)=1$ и дает правильные решения, т.е. $\mathbf{h}_1(X)=\mathbf{g}_2(X)$ и $\mathbf{h}_1(X)=\mathbf{g}_1(X)$. Вторая степень дает два возможных вида дополнительного полинома: X и $1+X$. И если первый вид фактически приводит также к правильному решению, поскольку всего лишь означает сдвиг по времени на один такт обработки содержимого сдвигового

регистра с теми же полиномами $\mathbf{g}_1$ и $\mathbf{g}_2$, то второй вариант приводит к совсем другому виду порождающих полиномов, т.е. к ошибочному решению.

Если же K_a=3, то возможных различных видов дополнительных полиномов будет еще больше: 1, X, 1+X, X^2, 1+X^2, X+X^2, 1+X+X^2, соответственно, уравнение (2.4) будет иметь семь решений.

Сказанное иллюстрируется рисунками 2.14-2.17.

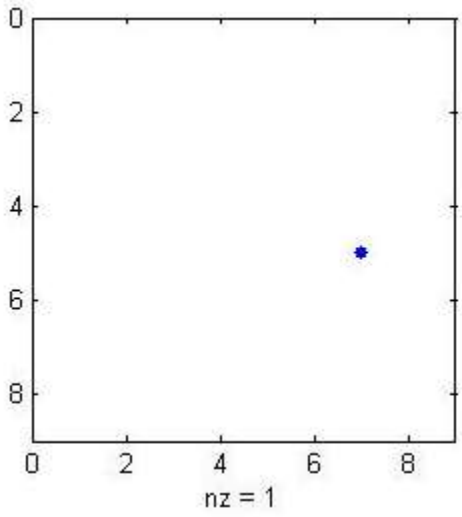

Рисунок. 2.14. К=3

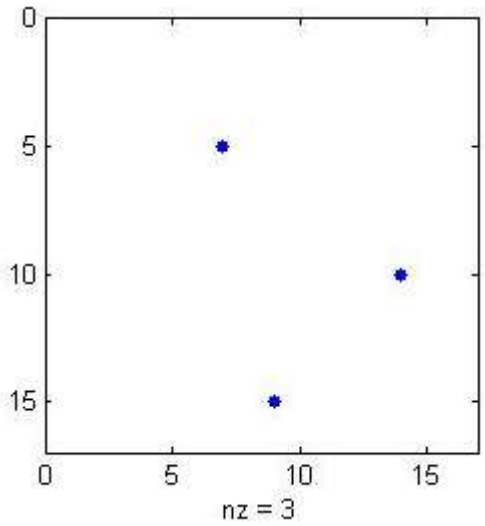

Рисунок. 2.15. $К_е$=4

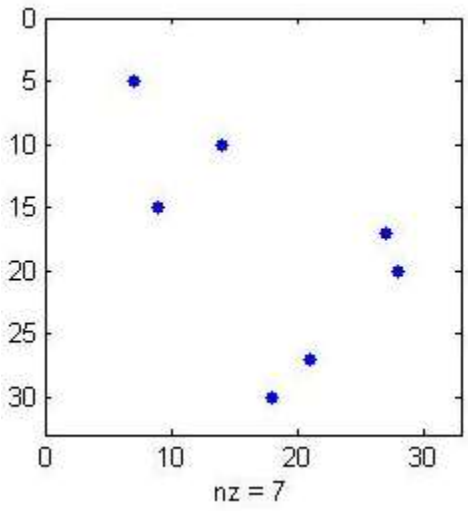

Рисунок. 2.16. $К_е$=5

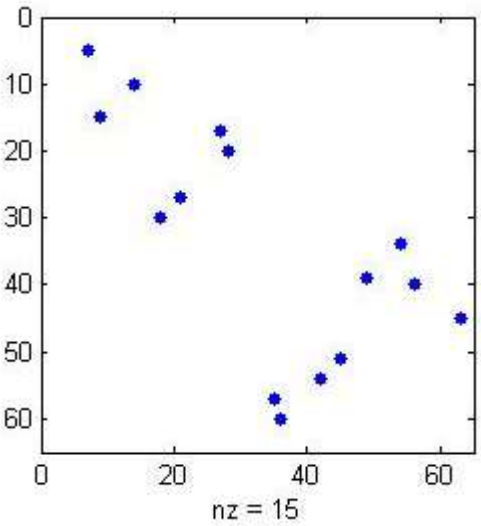

Рисунок. 2.17. $К_е$=6

На рисунках представлен вид матриц, полученных с помощью описываемого алгоритма после завершения процедуры. Пример приведен для кода (5,7), имеющего кодовое ограничение K=3 и использующего порождающие полиномы $\mathbf{g}_1(X)=1+X^2$ и $\mathbf{g}_2(X)=1+X+X^2$. Если именно такое значение применяется в алгоритме, то процедура выдает единственное правильное решение. Если использовано неправильное кодовое ограничение K_e=4, то процедура выдает три решения. Первое решение – правильное: (5,7). Второе решение – (10,14), соответствующее полиномам $X+X^3=X(1+X^2)$ и $X+X^2+X^3=X(1+X+X^2)$, т.е. фактически сдвинутое на один символ правильное первое решение. Третье решение – неправильное: (15,9), соответствующее полиномам $1+X+X^2+X^3=(1+X)(1+X^2)$ и $1+X^3=(1+X)(1+X+X^2)$. В случае K_e=5 количество решений будет равно nz=7 , в случае K_e=6 количество решений: nz=15.

Таким образом, встает обязательная задача предварительного определения кодового ограничения в случае отсутствия такой априорной информации.

Для этой цели может быть использован алгоритм, также обрабатывающий определенную матрицу. (Первоначально рассмотрим ситуацию, когда уровнем шумов можно пренебречь.) Алгоритм базируется на следующих свойствах частных кодовых последовательностей. Взятые по отдельности синхронные символы обеих частных последовательностей некоррелированы, и в общем случае условная вероятность появления единицы или нуля во второй последовательности равна 0,5 как при условии единичного, так и нулевого значения символа первой кодовой последовательности. Однако в случае рассмотрения достаточно длинных групп символов обеих последовательностей с одинаковыми индексами между ними обнаруживаются определенные взаимосвязи. С помощью их использования можно оценить величину используемого кодового ограничения.

Рассмотрим D подряд идущих символов информационной последовательности $\mathbf{m}(X)$. С помощью порождающего полинома из него образуется первая частная кодовая последовательность $\mathbf{u}_1=\mathbf{g}_1\mathbf{m}$. При этом из D символов информационной последовательности можно получить $L1$ символов первой кодовой последовательности, причем $D=L1+K-1$. При этом, если D

символов информационной последовательности однозначно определяют соответствующую группу из $L1$ символов первой кодовой последовательности, то в «противоположном» направлении такой однозначности нет. Кодовая последовательность L1 одного и того же вида может быть получена при разном виде D. Это различие проявляется в разном сочетании значений символов в «остатке» информационной последовательности размером K–1. То есть определенной кодовой последовательности размера $L1$ может соответствовать 2^{K-1} различных вариантов исходной информационной последовательности.

Но с другой стороны, исходная информационная последовательность **m** является общей для получения и первой, и второй частных кодовых последовательностей $\mathbf{u}_1$ и $\mathbf{u}_2$. Значит, та же группа из D символов информационной последовательности однозначно определяет соответствующие $L2$ символов второй кодовой последовательности. (Длина обеих групп одинакова, $L_1=L_2$).

Таким образом, если рассматривать две одновременные приходящие группы из $L_1 = L_2$ символов первой и второй кодовых последовательностей, то общее возможное количество вариантов их сочетаний будет не $2^{L1}\times 2^{L2}$, а *меньше*, так как каждому варианту группы из L_1 символов первой последовательности может соответствовать не 2^{L2} вариантов групп второй последовательности, а *всего лишь* 2^{K-1} вариантов, причем $2^{L1}\times 2^{K-1} << 2^{L1}\times 2^{L2}$. (Тот же результат будет если первоначально рассматривать вторую частную кодовую последовательность). В этом заключается появившаяся взаимосвязь между кодовыми последовательностями достаточной длины, если их длина $L>K$–1.

Использование этой взаимосвязи заключается в наборе достаточной статистики по группам символов частных кодовых последовательностей и определении размера «дефицита» появляющихся вариантов сочетаний. Измерение величины этого «дефицита» позволяет прямо вычислить значение кодового ограничения K.

Соответствующий алгоритм определения значения K может быть реализован структурной схемой, представленной на рисунке 2.18. Для этого

последовательно заполняется матричный блок памяти (МБП). Перед началом процедуры измерения величины K все ячейки МБП обнуляются. После начала последовательно анализируются группы по $L=L_1=L_2$ синхронных символов (т.е. символов с одинаковыми индексами) первой и второй частных кодовых последовательностей $\mathbf{u}_1(X)$ и $\mathbf{u}_2(X)$. Каждая новая пара групп может формироваться как последовательным сдвигом последовательностей $\mathbf{u}_1$ и $\mathbf{u}_2$ на один символ, так и выбором каждый раз полностью новых групп из этих последовательностей.

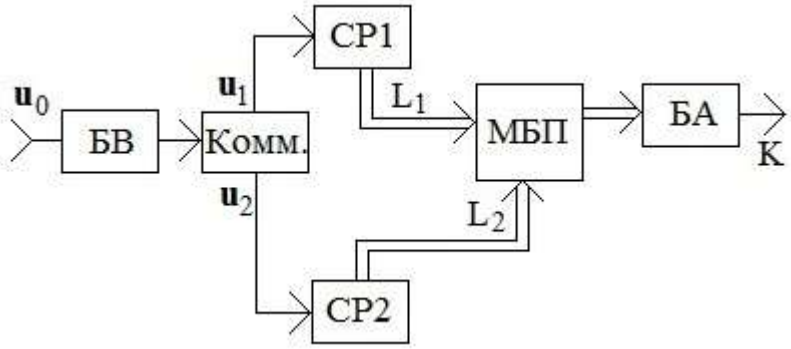

Рисунок. 2.18.

Поток символов общей кодовой последовательности $\mathbf{u}_0$ с помощью блока выделения кодовых последовательностей (БВ) и коммутатора (Комм.) разделяется по двум выходам коммутатора, создавая частные кодовые последовательности $\mathbf{u}_1(X)$ и $\mathbf{u}_2(X)$. Они последовательно записываются в первый и второй сдвиговые регистры (СР1 и СР2).

Ячейки памяти МБП, как и в предыдущих алгоритмах, расположены в виде матрицы, где пара номеров столбов и строк в двоичной системе счисления указывает адрес ячейки. При записи каждой новой пары групп в СР1 и СР2 их символы считываются с параллельных выходов регистров и используются, как номера столбца и строки новой ячейки. При этом к ее содержимому прибавляется единица.

После проведения достаточно большого числа N таких записей общее число записанных единиц в МБП также равно N, но при этом матрица оказывается

заполненной не вся. Если в течение N записей наблюдались все возможные варианты сочетаний значения символов или в первой, или во второй группах символов, то количество незаполненных ячеек МБП будет точно равно $2^L2^L - 2^L2^{(K-1)}$. В заполненных ячейках будет записано случайное количество единиц. Однако в данном случае величина этого числа значения не имеет, а определяющим является факт, что содержимое ячейки не равно нулю. После завершения процедуры блок анализа (БА) определяет количество ненулевых ячеек и вычисляет искомую величину K.

Обозначим через γ отношение общего числа ячеек в матрице к количеству ячеек с ненулевым содержанием. Это отношение измеряется блоком анализа и используется для вычисления K.

Поскольку $\gamma=(2^{2L})/(2^{L+K-1})$, то после преобразования:

$$K = L - \log_2 \gamma + 1. \qquad (2.5)$$

Работа алгоритма исследовалась с применением [99]. В качестве примера на рисунках 2.19-2.22 приведен вид матрицы памяти при разных величинах L и K. Черными точками обозначены ненулевые ячейки, нумерация строк и столбцов указана в десятичной системе счисления.

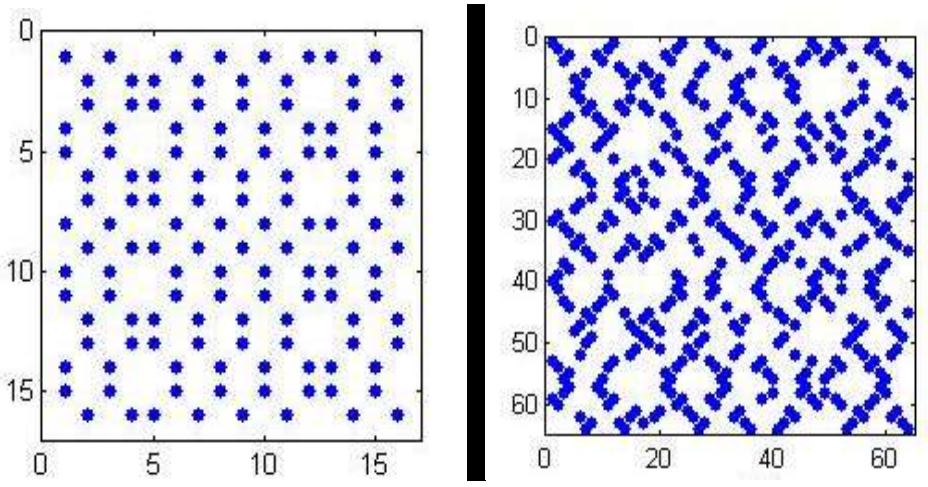

Рисунок. 2.19. K=4, L=4 Рисунок. 2.20. K=4, L=6

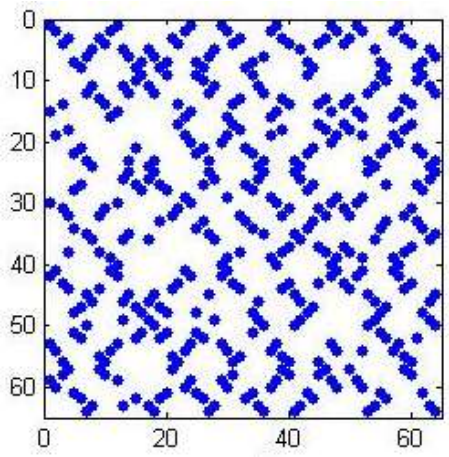

Рисунок. 2.21. K=6, L=6

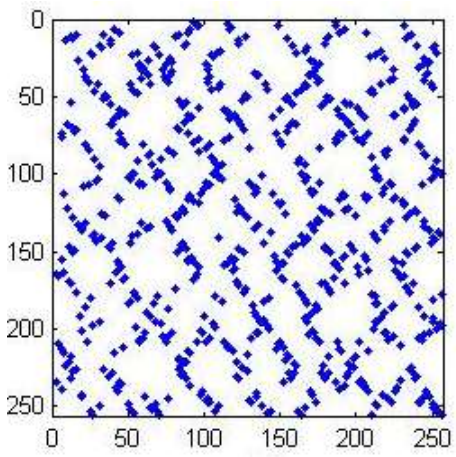

Рисунок. 2.22. K=6, L=8

Формула 2.5 исходит из того, что за N записей реализовались все L возможных вариантов групп символов. Однако в реальности существует вероятность того, что за любое количество записей определенное количество вариантов не появится. В этом случае в некоторых ячейках, в которых должны быть ненулевые суммы, останутся нули. Найдем вероятность того, что такие факты приведут к ошибке определения величины K.

Пусть общее количество «разрешенных» сочетаний групп символов равно $R=2^L\times 2^{K-1}$. Поскольку можно считать, что в этой «разрешенной» области все сочетания групп символов равновероятны, то вероятность того, что в определенную ячейку при одной записи добавится единица, равна $P_1=1/R$, а того, что не попадет, соответственно, $1-1/R$. Тогда вероятность того, что в эту ячейку за N записей единица не добавится ни разу, равна $P_2=(1-P_1)N$. Но с учетом того, что величина R достаточно большая и с использованием свойств «замечательных» пределов это выражение можно преобразовать:

$$P_2=(1-\frac{1}{R})^N=[(1-\frac{1}{R})^R]^{\frac{N}{R}}\approx\lim_{N\to\infty}\{[(1-\frac{1}{R})^R]^{\frac{N}{R}}\}=\{\lim_{N\to\infty}\{[(1-\frac{1}{R})^R]\}^{\frac{N}{R}}=e^{-\frac{N}{R}}.$$

Присутствие нулей в ячейках «разрешенной» области приводит к тому, что вычисленное значение K получается не целым, а дробным. Поскольку для

дальнейшего использования оно должно иметь вид целого числа, то в блоке БА оно округляется до ближайшего целого числа. Таким образом, допустимо, чтобы в определенном количестве V ячеек «разрешенной» области остались нули, при этом алгоритм, тем не менее, правильно определит искомое значение K. Если же количество таких ячеек будет больше, чем V, то будет принято ошибочное решение, что кодовое ограничение равно некоторому K_1. В качестве K_1 могут в принципе оказаться числа K–1, K–2 и т.д. Однако вероятность того, что получатся значения K–2, K–3 и т.д. существенно меньше, чем K–1, поэтому в качестве K_1 будем рассматривать K_1=К–1.

Граничное значение вычисленной дробной оценки K, которое тем не менее приведет к правильному решению, равно K–0,5, а допустимое количество V ячеек с нулевым содержанием, не вызывающее ошибки решения, определится выражением:

$$V = 2^{L} \times 2^{K-1} - 2^{L} \times 2^{K-1,5} = 2^{L+K-2}(2-\sqrt{2}).$$

Таким образом, вероятность неправильного решения при проведении N записей равна:

$$P_K = \sum_{i=V}^{R} C_i^R \exp(-i\frac{N}{R})[1-\exp(-\frac{N}{R})]^{R-i} \quad , \qquad (2.6)$$

где $C_i^R = R!/[i!(R-i)!]$.

На рисунке 2.23 приведены в качестве примера рассчитанные по формуле (2.6) графики зависимости P_K от количества записей N при некоторых сочетаниях параметров алгоритма.

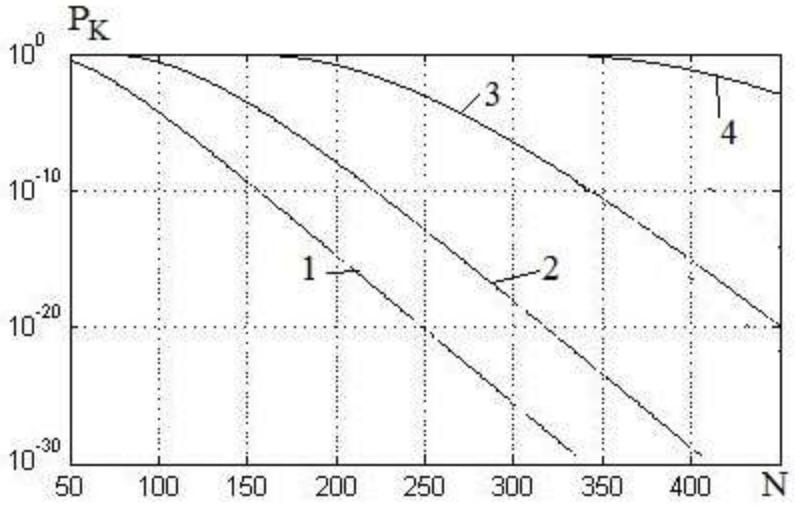

Рисунок. 2.23.

Цифрами обозначены графики, полученные при следующих значениях параметров: график 1 при L=K=3; график 2 при L=4, K=3; график 3 при L=K=4; график 4 при L=5, K=4. Сходные зависимости наблюдаются и при других сочетаниях параметров, при этом наблюдается большое различие порядка величины P_K даже при небольших изменениях L и N.

Полученные зависимости позволяют сделать вывод сделать вывод, что даже небольшое увеличение используемой длины L при условии сохранения такой же величины вероятности ошибочного решения P_K требует значительного возрастания количества записей N и размера матрицы памяти БПУ и вряд ли имеет смысл. В связи с этим можно рекомендовать использование минимально допустимого значения L=K–1.

Оценим возможную ошибку решения в условиях воздействия шумов значительного уровня. Обозначим весь объем ячеек памяти через $S=2^{2L}$ и количество «запрещенных» ячеек памяти, куда при правильной работе алгоритма никогда не должны добавляться единицы, через Z=S–R.

Влияние шумов проявится в том, что добавление единиц будет происходить не в те ячейки, куда следует, а в случайно расположенные. Пусть вероятность появления ошибочного символа равна P_B. В результате в какие-то «разрешенные» ячейки единицы добавляться не будут, а будут добавляться в «запрещенные»

ячейки. Однако на «разрешенные» области это существенного влияния оказывать не будет, т.к. эквивалентно тому, что вместо N записей в эту область было произведено $N(1–P_B)$ записей. Поскольку в современных системах передачи информации величина вероятности появления ошибок P_B, как правило, достаточно невелика, то на работу алгоритма это скажется незначительно.

Считаем, что ошибочная запись может быть равновероятно произведена в любую из S ячеек, тогда вероятность попадания ошибочной записи в любую ячейку «запрещенной» зоны равна RP_B/S. Вероятность, что в конкретную ячейку при проведении N записей будет записана хотя бы одна единица, определится выражением: $P_2=1–(1–RP_B/S)^N$.

Так же, как и в предыдущем случае, часть неверных записей не опасна, т.к. в результате округления вычисленного кодового ограничения до целой величины все равно процедура даст правильный результат. Здесь аналогично предыдущему случаю будем учитывать только ближайшее ошибочное значение K+1. При этом размеры зоны V_2, попадание в которую не опасно, будут равны:

$$V_2 = 2^L \times 2^{(K-0,5)+1} - 2^L \times 2^{K-1} = 2^{L+K-1}(\sqrt{2}-1).$$

Таким образом, вероятность ошибочного определения кодового расстояния по причине воздействия шумов определится выражением:

$$P_V = \sum_{i=V_2}^{Z} C_i^Z P_2^i (1-P_2)^{Z-i}. \qquad (2.7)$$

На рисунке 2.24 приведены зависимости вероятности ошибочного решения для некоторых параметров.

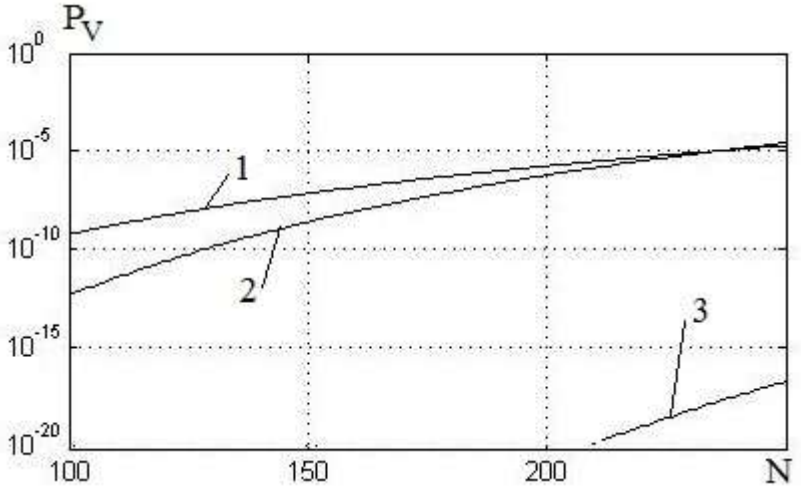

Рисунок. 2.24.

Нумерация графиков соответствует следующим значениям параметров алгоритма: график 1 соответствует L=K=3; график 2 соответствует L=4, K=3; график 3 соответствует L=K=4. Вычисления производились при P_B=10^{-3}.

Следует отметить, что с ростом числа записей N вероятность ошибочного решения также возрастает. Это объясняется тем, что при большом количестве записей кроме добавления единиц в «разрешенные» ячейки также увеличивается число попаданий и в «запрещенную» зону. В результате в ней может накопиться достаточное количество ячеек с ненулевым содержимым, которое может привести к тому что после округления результатов анализа в БА будет получена неверная оценка кодового ограничения, равная K+1.

Описанный в данном подразделе алгоритм также позволяет провести проверку правильности работы блока выделения кодовых последовательностей БВ. Блок должен из входной общей кодовой последовательности $\mathbf{u}_0(X)$ выделять пары символов, которые соответствуют одному общему символу исходной информационной последовательности $\mathbf{m}(X)$. Если в результате его неправильной работы такие пары символов будут определены неверно (например, один взят из пары, получено из предыдущего информационного символа, а другой из последующего информационного символа), то формируемая матрица приобретет другой вид. Поскольку теперь группы символов, помещаемые в сдвиговые

регистры, образованы из разных участков информационной последовательности, то используемая алгоритмом их внутренняя взаимосвязь пропадает. В результате «запрещенная» и «разрешенная» области отсутствуют, и вся матрица заполняется равномерно.

В следующем подразделе будут рассмотрены «быстрые» алгоритмы диагностики, которые за счет усложнения обработки позволяют уменьшить общее время диагностической процедуры.

2.5. «Быстрые» алгоритмы диагностики сверточных кодов

Диагностическая процедура, описанная в подразделах 2.2 и 2.3, производится с помощью достаточно простых алгоритмов, однако состоящих из большого количества операций. Ее можно ускорить за счет усложнения алгоритмов. В данном параграфе будут описаны два «быстрых» алгоритма диагностики сверточных кодов, основанные на их модификации.

Сущность первого алгоритма заключается в следующем. Ранее для нахождения «правильного» варианта сочетания полиномов осуществлялось N циклов, причем в каждом цикле производилось одинаковое количество 2^{2K} операций сравнения. А теперь предлагается в каждом последующем цикле производить разное количество операций сравнения, уменьшая их число в каждом последующем цикле. Уменьшение числа операций возможно за счет исключения тех сочетаний вариантов полиномов, которые в предыдущем цикле были определены, как «неправильные». Количество «неправильных» полиномов, определенных в каждом цикле является случайным, однако общая тенденция снижения их числа сохраняется при любом оставшемся объеме. В результате, хотя общее количество циклов процедуры в среднем сохраняется тем же, однако общее число необходимых операций сравнения значительно сокращается.

На рисунке 2.25 представлен вариант структуры, реализующей данный «быстрый» алгоритм диагностики. Ее можно разбить на два укрупненных блока. Структура блока Б1 совпадает с соответствующей частью структуры,

изображенной на рисунке 2.2 для алгоритма, описанного в подразделе 2.2. Фактически, она осуществляет предварительную обработку поступающей общей кодовой последовательности $\mathbf{u}_0$. Здесь она разделяется на две частные кодовые последовательности $\mathbf{u}_1$ и $\mathbf{u}_2$ с помощью блоков БВПК и Комм.1, далее группы по K символов обеих частных кодовых последовательностей записываются в сдвиговые регистры СР1 и СР2. С помощью многоканальных ключей МК1 и МК2 и сумматоров по модулю 2 осуществляется повторное кодирование и результаты кодирования сравниваются в схеме сравнения СС.

Далее операции данного алгоритма уже отличаются от алгоритма по п. 2.2. Они производятся в общем блоке Б2. Узлы этого блока работают следующим образом.

Все варианты сочетания вида «поисковых» полиномов также записываются с помощью двоичного числа с $2K$ разрядами. Наборы таких вариантов хранятся в блоке FIFO («first input – first output»). Это блок памяти, осуществляющий в свои ячейки циклическую запись данных. Записываемые $2K$ разрядные двоичные числа подаются на его вход и записываются при поступлении управляющего сигнала записи. При записи каждого последующего числа адреса всех предыдущих чисел последовательно сдвигаются.

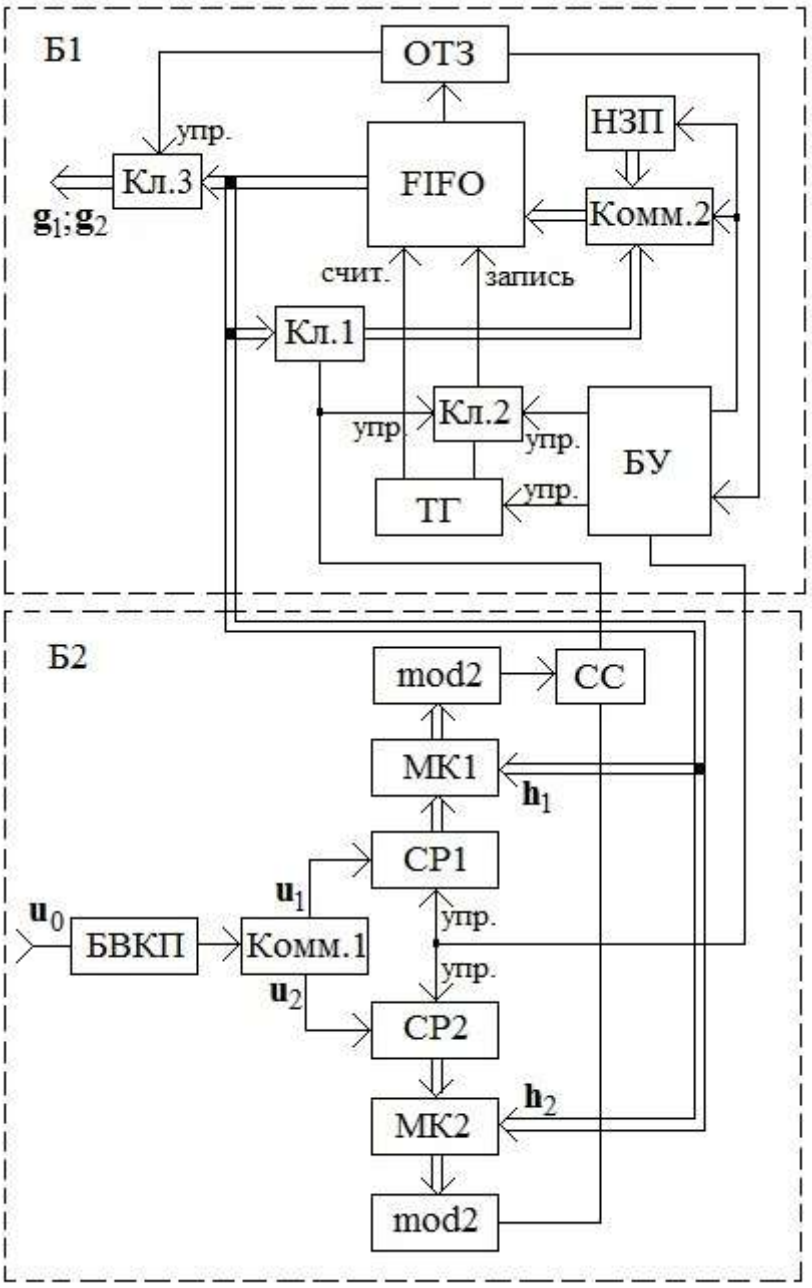

Рисунок. 2.25.

Вывод каждого записанного числа производится по управляющему сигналу считывания. При этом считывается первое в очереди из имеющихся записанных чисел, после чего оно удаляется из блока FIFO, а первым в очереди на считывание становится следующее записанное после него число. Считывание и запись могут осуществляться независимо и с разной скоростью.

Таким образом, адрес записи и адрес считывания перемещаются по «кольцу», при этом могут приближаться или удаляться один от другого. Поскольку общий объем памяти ограничен, то при его полном заполнении блоком выдается определенный сигнал и новая запись произведена быть не может, пока не считано число, стоящее первым в очереди на считывание. Также обычно выдается информация об общем количестве занятых в текущий момент ячеек памяти, что позволяет контролировать ее полное опустошение.

В начале процедуры диагностики блок управления (БУ) осуществляет с помощью блока начальной загрузки памяти (НЗП) запись в блок FIFO всех 2^{2K} вариантов двоичных чисел, определяющих структуру «поисковых» полиномов. Для этого он переключает коммутатор Комм.2 на выход блока НЗП и включает тактовый генератор (ТГ), который вырабатывает соответствующие записывающие импульсы для FIFO и подает их через открытый ключ Кл.2.

После этого предварительного заполнения памяти FIFO начинается собственно процедура анализа. ТГ начинает подавать считывающие импульсы на блок FIFO. При каждом считывании соответствующее двоичное число поступает на МК1 и МК2 и производится повторное кодирование с «поисковыми» полиномами, определяемыми этим числом. Далее результаты кодирования сравниваются в СС.

Если они совпадают, то по сигналу схемы сравнения открываются первый и второй ключи (Кл.1 и Кл.2). В результате на блок FIFO поступает записывающий импульс, а данное двоичное число вновь перезаписывается в этот блок, но уже оказываясь в конце всего записанного массива чисел («в хвосте» очереди). Если же схема сравнения определяет, что результаты кодирования не совпадают, то перезапись данного двоичного числа не происходит. Оно безвозвратно удаляется из памяти. После чего с помощью следующего считывающего импульса происходит переход к следующему двоичному числу, т.е. к анализу следующего варианта сочетания «поисковых» полиномов.

Таким образом, в первом цикле процедуры будут проанализированы все 2^{2K} вариантов полиномов. В результате часть «неправильных» полиномов удалятся из

памяти и количество вновь перезаписанных чисел – вариантов полиномов, уменьшится. В следующем цикле из них опять удалится некоторая часть, и т.д. Работа прекращается, когда в памяти остается только одно двоичное число, которое определит «правильный» вариант «поисковых» полиномов. Текущее оставшееся количество чисел в блоке FIFO определяется блоком определения текущей загрузки (ОТЗ). Когда в памяти останется единственное число, он открывает ключ Кл.3 и это число, определяющее структуру искомых порождающих полиномов $\mathbf{g}_1$ и $\mathbf{g}_2$, подается как результат диагностики на выход для дальнейшего внешнего использования.

Количество удаляемых из памяти чисел в каждом цикле является случайной величиной, ее свойства были исследованы в п. 2.2. В среднем в каждом цикле количество оставшихся в памяти чисел уменьшается вдвое и средняя продолжительность анализа с каждым циклом также уменьшается вдвое. При этом среднее количество циклов будет равно $2K$, как и в алгоритме по п.2.2. Но в алгоритме по п.2.2 среднее количество операций было равно $2K{\cdot}2^{2K}$. А в рассматриваемом алгоритме среднее общее количество операций за всю процедуру составит величину $\sum_{i=0}^{2K} 2^i = 2^{2K+1} - 1 \approx 2 \cdot 2^{2K}$. Таким образом, процедура по описываемому «быстрому» алгоритму производится в среднем в K раз быстрее.

Однако кроме достоинств этот алгоритм имеет и недостатки. Один из них состоит в необходимости использования более сложной схемы. Другой недостаток более значителен и заключается в повышенной чувствительности алгоритма к символьным ошибкам, возникающим при значительном уровне шумов. Здесь не происходит накопления результата, и если случится символьная ошибка, то она удаляет «правильную» комбинацию из памяти. Вероятность такого события равна $1–(1–P_B)^{2K}\approx 2KP_B$. Выходом может служить повторение процедуры и сравнение ее результатов, однако при этом теряется выигрыш в общем времени диагностики. Поэтому при значительном уровне шумов

необходимо для диагностики использовать более длительную процедуру, реализуемую алгоритмом, изложенным в п. 2.3.

Второй «быстрый» алгоритм основывается на свойствах полиномов $\mathbf{g}_1$ и $\mathbf{g}_2$, которые используются при кодировании. Как уже упоминалось, эти полиномы не должны содержать одинаковых полиномиальных множителей. Такое правило приводит к существенному сокращению возможных вариантов сочетания видов «поисковых» полиномов.

Сокращение количества возможных вариантов рассмотрим на примере кодового ограничения K=6. Соответствующие данные приведены в таблице 2.1.

Таблица 2.1.

	$\mathbf{g}_1(X)$	Разложение полинома $\mathbf{g}_1(X)$ на простые множители	Возможные варианты полинома $\mathbf{g}_2(X)$
1.	$33_{10}=41_8$	$1+X^5=(1+X)(1+X+X^2+X^3+X^4)$	35_{10}, 37_{10}, 41_{10}, 47_{10}, 49_{10}, 51_{10}, 55_{10}, 59_{10}, 61_{10}
2.	$35_{10}=43_8$	$1+X+X^5=(1+X+X^2)(1+X+X^3)$	33_{10}, 37_{10}, 41_{10}, 43_{10}, 47_{10}, 51_{10}, 53_{10}, 55_{10}, 57_{10}, 59_{10}, 61_{10}
3.	$37_{10}=45_8$	$1+X^2+X^5$	$33_{10} - 63_{10}$
4.	$39_{10}=47_8$	$1+X+X^2+X^5=$ $=(1+X)^2(1+X+X^3)$	37_{10}, 41_{10}, 47_{10}, 51_{10}, 55_{10}, 59_{10}, 61_{10}
5.	$41_{10}=51_8$	$1+X^3+X^5$	$33_{10} - 63_{10}$
6.	$43_{10}=53_8$	$1+X+X^3+X^5=$ $=(1+X)(1+X^3+X^4)$	35_{10}, 37_{10}, 41_{10}, 47_{10}, 49_{10}, 51_{10}, 53_{10}, 55_{10}, 59_{10}, 61_{10}
7.	$45_{10}=55_8$	$1+X^2+X^3+X^5=$ $=(1+X)^3(1+X+X^2)$	37_{10}, 41_{10}, 47_{10}, 51_{10}, 55_{10}, 59_{10}, 61_{10}
8.	$47_{10}=57_8$	$1+X+X^2+X^3+X^5$	$33_{10} - 63_{10}$
9.	$49_{10}=61_8$	$1+X^4+X^5=$	33_{10}, 37_{10}, 41_{10}, 43_{10},

		$=(1+X+X^2)(1+X+X^3)$	47_{10}, 51_{10}, 53_{10}, 55_{10}, 57_{10}, 59_{10}, 61_{10}
10.	$51_{10}=63_8$	$1+X^3+X^5$	$33_{10} - 63_{10}$
11.	$53_{10}=65_8$	$1+X+X^3+X^5=$ $=(1+X)(1+X+X^4)$	35_{10}, 37_{10}, 41_{10}, 47_{10}, 49_{10}, 51_{10}, 55_{10}, 59_{10}, 61_{10}
12.	$55_{10}=67_8$	$1+X+X^2+X^4+X^5$	$33_{10} - 63_{10}$
13.	$57_{10}=71_8$	$1+X^3+X^4+X^5=$ $=(1+X)^2(1+X^2+X^3)$	35_{10}, 37_{10}, 41_{10}, 47_{10}, 49_{10}, 51_{10}, 55_{10}, 59_{10}, 61_{10}
14.	$59_{10}=73_8$	$1+X+X^3+X^4+X^5$	$33_{10} - 63_{10}$
15.	$61_{10}=75_8$	$1+X^2+X^3+X^4+X^5$	$33_{10} - 63_{10}$
16.	$63_{10}=77_8$	$1+X+X^2+X^3+X^4+X^5=$ $=(1+X)(1+X+X^2)^2$	37_{10}, 41_{10}, 47_{10}, 51_{10}, 55_{10}, 59_{10}, 61_{10}

Во втором столбце таблицы приведены все возможные варианты полинома $\mathbf{g}_1$ при K=6. В третьем столбце таблицы показано разложение в соответствии с правилами конечной алгебры полинома $\mathbf{g}_1$ на простые полиномы. Поскольку полином $\mathbf{g}_2$ в своем разложении не может содержать ни одного простого полинома из разложения $\mathbf{g}_1$, то количество его допустимых вариантов сокращается. В третьем столбце таблицы показано, какие варианты полинома $\mathbf{g}_2$ допустимы при конкретном варианте полинома $\mathbf{g}_1$.

Общее количество вариантов сочетаний «поисковых» полиномов при данном значении кодового ограничения K=6 равно 196. Если рассматривать все возможные варианты вида полинома $\mathbf{g}_2$ и допустимые для них варианты полинома $\mathbf{g}_1$, то, естественно, результаты будут аналогичными. На практике используются обычно не все сочетания, а обладающие лучшими свойствами помехоустойчивости (максимальным «свободным» расстоянием [4,26]). Однако заранее неизвестно, используются ли именно такие полиномы.

Реализация данного алгоритма возможна по схеме алгоритма по п. 2.2, если вместо двоичного счетчика (ДС) использовать генератор допустимых сочетаний

видов «поисковых» полиномов по таблице, аналогичной таблице 2.1 для используемого в данный момент кодового ограничения. Это приведет к существенному сокращению объема перебираемых вариантов. В частности, для рассмотренного примера при K=6 вместо необходимости перебора $2^{2\cdot6}$=4096 вариантов будет необходимо перебирать всего 196+196=392 варианта.

Недостатком данного алгоритма можно считать факт, что мы заранее ограничиваемся предполагаемыми вариантами комбинаций, а в случае использования нестандартных вариантов, их диагностика невозможна.

Данный алгоритм может быть еще более ускорен, если дополнительно использовать принцип работы первого алгоритма, изложенного в этом подразделе. После каждой проведенной проверки «неправильные» сочетания полиномов исключаются при проверке в следующем цикле. Однако, как указывалось и для первого алгоритма, такой подход применим только при незначительном уровне шумов.

Таким образом, в зависимости от параметров диагностируемого кода общий объем вычислений может быть сокращен (для типовых значений кодового расстояния) в 5-10 раз.

2.6. Выводы

1. При осуществлении диагностической процедуры сверточных кодов наиболее удобным является алгоритм, использующий повторное кодирование частных кодовых последовательностей.
2. Алгоритм с повторным кодированием может быть реализован в матричной форме, при этом при малом уровне шумов более быстродействующим является вариант с перемножением, а при значительном уровне шумов – с накоплением результатов анализа.
3. Кодовое ограничение может быть оценено с помощью соотношения количества наблюдаемых и возможных комбинаций частных кодовых последовательностей.

4. При часто используемых параметрах кодирования и малых уровней шума для обеспечения вероятности неправильной диагностики ниже $2\cdot 10^{-3}$ достаточно произвести около 25 циклов анализа, а для снижения этого уровня до $2\cdot 10^{-5}$ достаточно 30 циклов анализа.

5. При большом уровне шумов (вероятности битовой ошибки до 10^{-1}) количество циклов анализа необходимо увеличить в 2-4 раза.

6. Алгоритм может быть реализован в «быстрых» модификациях путем исключения части «неправильных» сочетаний видов порождающих полиномов, но его реализация имеет смысл лишь при малых уровнях шумов. В этом случае время анализа может сократиться в 5-10 раз.

3. Разработка алгоритмов диагностики блоковых кодов

В данной главе предложены и исследованы алгоритмы диагностики двоичных блоковых кодов. Рассмотрены ситуации использования как систематических, так и несистематических кодов. Поскольку блоковые коды реализуются в основном в циклическом варианте, получаемом с помощью порождающих полиномов, то основное внимание уделено именно циклическим кодам. Из нециклических кодов описан алгоритм диагностики кодов, получаемых с помощью порождающей матрицы. Исследована работа алгоритмов в условиях воздействия шумов значительного уровня. Поскольку время диагностики существенным образом зависит от размеров кодовых блоков, то рассмотрены «быстрые» алгоритмы, позволяющие значительно сократить длительность диагностики.

3.1. Принципиальные основы диагностики блоковых кодов

Причины необходимости диагностики блоковых кодов отличаются от причин диагностики сверточных кодов, изложенных в предыдущей главе. Поскольку сверточное кодирование применяется в основном в несистематической форме, то отсутствие сведений о параметрах кодера приводит к невозможности раскодирования и приема информации. Блоковое же кодирование применяется в основном в систематической форме. При этом в блоке можно выделить информационную часть и часть, содержащую проверочные символы. Поэтому в принципе можно с низким качеством определить передаваемую информационную последовательность. Однако декодирование с исправлением ошибок без знания параметров кодера невозможно, поэтому и помехоустойчивость передачи оказывается низкой, в основном, значительно ниже допустимых норм качества. Если же здесь используется несистематическое кодирование, как правило, реализуемое с помощью порождающей матрицы, то без знания параметров кодера даже выделение из блоков информационной части и в этом случае невозможно.

Тем не менее, поскольку все блоки сформированы по одним и тем же правилам и с помощью одного и того же кодера, то независимо от передаваемой информации, взаимосвязи между проверочными и информационными символами каждого блока имеют один и тот же характер. Это дает возможность на основе анализа необходимого количества блоков выявить эти взаимосвязи и на их основе восстановить структуру кодера. А это, в свою очередь, позволит повысить помехоустойчивость передачи информации. В то же время, принципы диагностики зависят от вида блоковых кодов.

Поскольку подавляющее большинство блоковых кодов используется в циклической форме, то в первую очередь рассмотрены принципы диагностики циклических кодов.

В этом случае кодирование заключается в применении порождающего полинома $\mathbf{g}(X)$. Как известно, при несистематическом кодировании каждый i-й кодовый блок может быть описан в виде произведения двух полиномов: $\mathbf{y}_i(X)=\mathbf{m}_i(X)\mathbf{g}(X)$, где $\mathbf{g}(X)$ – порождающий полином используемого кодера, общий для всех кодовых слов; $\mathbf{m}_i(X)$ – часть i–го кодового слова, содержащая передаваемую в нем информацию. То есть, полиномы $\mathbf{y}_i(X)$, описывающие все кодовые блоки, имеют как минимум один общий полином $\mathbf{g}(X)$.

Пусть длина кодовых блоков составляет n символов. Блоки содержат k информационных символов, добавляемые к ним b проверочных символов сформированы на основе правил применяемого кодирования, n=k+b. В этом случае максимальная степень полинома $\mathbf{g}(X)$ равна b. Максимальная степень полинома $\mathbf{m}_i(X)$ зависит от передаваемой в данном блоке информационной части и может достигать максимальной величины, равной k.

При систематическом циклическом кодировании кодовый блок в кодере формируется по-другому. Для этого исходный двоичный информационный полином $\mathbf{m}_i(X)$ первоначально домножается на X^b, что соответствует сдвигу на b разрядов в сторону увеличения. В результате получается полином $\mathbf{m}_i(X)X^b$ с максимальной степенью, в общем случае равной $n=k+b$. Далее он делится на порождающий полином $\mathbf{g}(X)$. Максимальная степень образующегося остатка $\mathbf{r}_i(X)$

равна b, т.е. равна количеству нулей в записи $\mathbf{m}_i(X)X^b$, начиная с первого разряда. После этого остаток $\mathbf{r}_i(X)$ складывается по модулю 2 с полиномом $\mathbf{m}_i(X)X^b$, образуя передаваемый по каналу передачи кодовый блок $\mathbf{y}_i(X)=\mathbf{m}_i(X)X^b+\mathbf{r}_i(X)$. Таким образом, и в этом случае кодовый блок кратен полиному $\mathbf{g}(X)$, т.е. порождающий полином является одним из множителей полинома $\mathbf{y}_i(X)$. Можно записать: $\mathbf{y}_i(X)=\mathbf{M}_i(X)\mathbf{g}(X)$, где множитель $\mathbf{M}_i(X)$ зависит от информационных символов, использованных при формировании данного блока. Именно этот факт используется при декодировании в приемнике для вычисления синдромов ошибок и их исправления.

Поскольку в исходной информационной последовательности символы в общем случае взаимно независимы, то и информационные части $\mathbf{M}_i(X)$ полиномов различных блоков также независимы между собой. А части $\mathbf{g}(X)$ полиномов *одинаковые* во всех кодовых блоках. Поэтому при сравнении достаточного числа кодовых блоков можно определить общую часть их полиномов. Именно она описывает порождающий полином $\mathbf{g}(X)$, используемый при кодировании в передатчике.

Если при передаче применяются нециклические коды, (имеются в виду линейные блоковые коды в общем виде) то кодовые блоки формируются по-другому. Для этого используется порождающая матрица **G** [4]. Для систематических кодов, а из линейных блоковых кодов используются практически только они, процесс кодирования завершается существенно быстрее.

Пусть порождающая матрица имеет размер $k\times n$. Все ее векторы-строки являются линейно-независимыми. В простом виде ее можно записать, как состоящую из двух матричных блоков **E** и **P**:

$$\mathbf{G}=\|\mathbf{E}\vdots\mathbf{P}\|=\begin{Vmatrix} 1,0,0,0,\ldots,0, p_{1,k+1}, p_{1,k+2},\ldots,p_{1,k+b} \\ 0,1,0,0,\ldots,0, p_{2,k+1}, p_{2,k+2},\ldots,p_{2,k+b} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ 0,0,0,0,\ldots,1, p_{k,k+1}, p_{k,k+2},\ldots,p_{k,k+b} \end{Vmatrix},$$

где **E** – единичная матрица размера $k \times k$, у которой элементы главной диагонали равны единице, все остальные элементы равны нулю; **P** – проверочная матрица размера $k \times b = k \times (n-k)$, состоящая из элементов p_{ij} . Все вектор-строки единичной матрицы, естественно, линейно-независимы. Для осуществления эффективного декодирования с исправлением ошибок все вектор-строки матрицы **P** также должны быть линейно-независимы.

Обозначим вектор-строки матрицы **G** через $\mathbf{V}_1$, $\mathbf{V}_2$,…,$\mathbf{V}_k$. Тогда любое кодовое слово можно записать в виде:

$$\mathbf{y}=\mathbf{mG}=m_1\mathbf{V}_1+m_2\mathbf{V}_2+\ldots+m_k\mathbf{V}_k.$$

Оно представляет собой сумму по модулю 2 части строк порождающей матрицы, взятых с весовыми коэффициентами, определяемыми наличием нулей и единиц соответствующей информационной последовательности.

При декодировании должна использоваться соответствующая проверочная матрица **H** размером $(n-k) \times n = b \times n$, такая, что ее строки ортогональны строкам матрицы **G**. Матрица **H** также состоит из двух блоков: $\mathbf{H} = \left\| \mathbf{P}^T \vdots \mathbf{E} \right\|$, причем размеры единичной матрицы **E** здесь равны $b \times b$. Для дальнейшего исправления ошибок с помощью синдромов используется свойство $\mathbf{GH}^{\mathrm{T}}=\mathbf{0}$, где **0** – нулевая матрица размера $b \times b$.

Таким образом, задачей диагностики в данном случае является определении вида матрицы **P** на основе анализа принимаемых кодовых блоков.

Принципиальная основа диагностического алгоритма основана на следующем. Информационные части различных кодовых блоков в общем случае независимы между собой. Поэтому среди принятых блоков можно подобрать k блоков $y_1 \div y_k$, информационные части $m_1,\ldots,m_k$ которых между собой все являются взаимно независимыми. Из них, как из строк, составляется матрица $\mathbf{Y} = \left\| \mathbf{M} \vdots \mathbf{S} \right\|$, в которой часть **M** имеет размер $k \times k$, часть **S** имеет размер $k \times b$. Строки матрицы **M**

– это информационные последовательности $m_1,\ldots,m_k$ выбранных кодовых блоков $y_1 \div y_k$. Строки матрицы **S** – это проверочные части этих кодовых блоков.

Далее вычисляется обратная ей матрица $\mathbf{M}^{-1}$ (Все операции сложения-вычитания производятся по модулю 2). Произведение матриц $\mathbf{M}^{-1}\mathbf{M}=\mathbf{E}$ образует единичную матрицу. Поэтому, если матрицу **Y** умножить на матрицу $\mathbf{M}^{-1}$, то получается матрица:

$$\mathbf{M}^{-1}\mathbf{Y} = \mathbf{M}^{-1}\left\|\mathbf{M}\vdots\mathbf{S}\right\| = \left\|\mathbf{M}^{-1}\mathbf{M}\vdots\mathbf{M}^{-1}\mathbf{S}\right\| = \left\|\mathbf{E}\vdots\mathbf{M}^{-1}\mathbf{S}\right\| .$$

Таким образом, получается порождающая матрица $\mathbf{M}^{-1}\mathbf{Y}=\mathbf{G}$, в которой часть $\mathbf{M}^{-1}\mathbf{S}=\mathbf{P}$, т.е. фактически задача диагностики оказывается решенной.

При реализации описанного принципа в форме алгоритма необходимо решить предварительные задачи. В частности, требуется из принятых сигналов выбрать такой набор кодовых блоков $y_1 \div y_k$, у которых информационные части представляют собой линейно-независимые векторы. Кроме этого, могут оказаться неизвестными заранее размеры информационной k и проверочной b частей блоков. В этом случае их также необходимо определять заранее.

Таким образом, на основе анализа определенной совокупности принятых блоков символов можно в результате диагностики определить структуру используемого кодера для циклических и линейных блоковых кодов. Решению этих задач посвящены следующие параграфы данной главы. Первоначально рассмотрены условия работы при отсутствии влияния шумов, затем решения расширены на условия работы в шумах значительного уровня.

3.2. Алгоритм диагностики на основе непосредственного вычисления простых полиномов

Как было рассмотрено в предыдущем параграфе, задача диагностики циклических кодов, как наиболее распространенного варианта блоковых кодов, решается нахождением общего множителя всех рассматриваемых кодовых

блоков. С вычислительной точки зрения она может решаться с различных позиций. Наиболее простая из них – это определение для каждого полинома, описывающего кодовый блок, всех его полиномов, как сомножителей.

Двоичные полиномы в полях Галуа обладают рядом свойств, схожих со свойствами чисел натурального ряда. В частности, как и любое число натурального ряда, двоичный полином можно представить в виде произведения некоторых простых полиномов. Эти простые полиномы уже не являются произведениями каких-то еще более простых полиномов с меньшей максимальной степенью, т.к. выступают в роли простых множителей натурального числа. Таким образом, любой сложный полином можно выразить как совокупность произведения простых полиномов (своего рода, как «спектр», из них составленный).

Как уже говорилось, каждый кодовый блок, как последовательность двоичных символов, может быть описан в виде полинома, а он в свою очередь, разложен на простые множители – простые полиномы. Тот набор простых полиномов, которые описывают изменяющуюся информационную часть блока, различается в каждом блоке, а тот набор полиномов, произведение которых составляет искомый порождающий полином, является общим для всех кодовых блоков.

Таким образом, диагностический алгоритм представляет собой совокупность операций, представленных на рисунке 3.1.

Он состоит из вычисления для каждого кодового блока набора простых полиномов, как сомножителей, на которые разлагается этот блок, далее накопления информации о частоте появления различных простых полиномов, и после анализа определенного количества блоков выбор набора простых сомножителей, встречающихся во всех блоках. Произведение этих простых полиномов и составляет тот порождающий полином, который является целью диагностического анализа.

Первоначально производится отбор N кодовых блоков $y_1 \div y_N$. В общем виде эти сложные полиномы представляют произведения простых полиномов.

(Исключение имеют место лишь при появлении ошибочных символов, которые случайно могут преобразовать полином кодового блока в простой полином, размером, аналогичным размеру блока). Далее полиномы этих блоков поочередно подаются на процедуру разделения на простые полиномы.

Процедура разделения на простые полиномы носит частично циклический характер. Поочередно перебираются значения полиномов γ_k, начиная с наиболее простого, равного X (двоичное число 2) до $X^{n-1}+X^{n-2}+\ldots X+1$ (двоичное число, равное 2^n-1). Полином $\mathbf{y}_i$ очередного кодового слова проверяется на делимость на полином γ_k, и если он делится нацело, то делитель запоминается (деление производится в соответствии с правилами операций в полях Галуа). После этого частное от деления проверяется на равенство единице, и в случае неравенства опять повторяется деление на γ_k. Если же устанавливается, что полученный полином $\mathbf{y}_i$ / γ_k нацело на γ_k не делится, то происходит проверка делимости полинома $\mathbf{y}_i$ на следующий делитель γ_{k+1}.

После декомпозиции на сомножители всех выбранных N кодовых блоков определяются наиболее часто встречающиеся простые полиномы, как делители. И хотя в информационных блоках некоторые простые полиномы (например, полином, X, что будет исследовано далее) тоже будут встречаться часто, но тем не менее, не во всех кодовых блоках. А те полиномы, которые составляют порождающий полином, будут встречаться во всех блоках, поэтому чаще, чем все возможные другие. В результате именно их количество после окончания всей процедуры будет запомнено наибольшим. Таким образом, последняя операция анализа определит именно их, и представит, как результат диагностики.

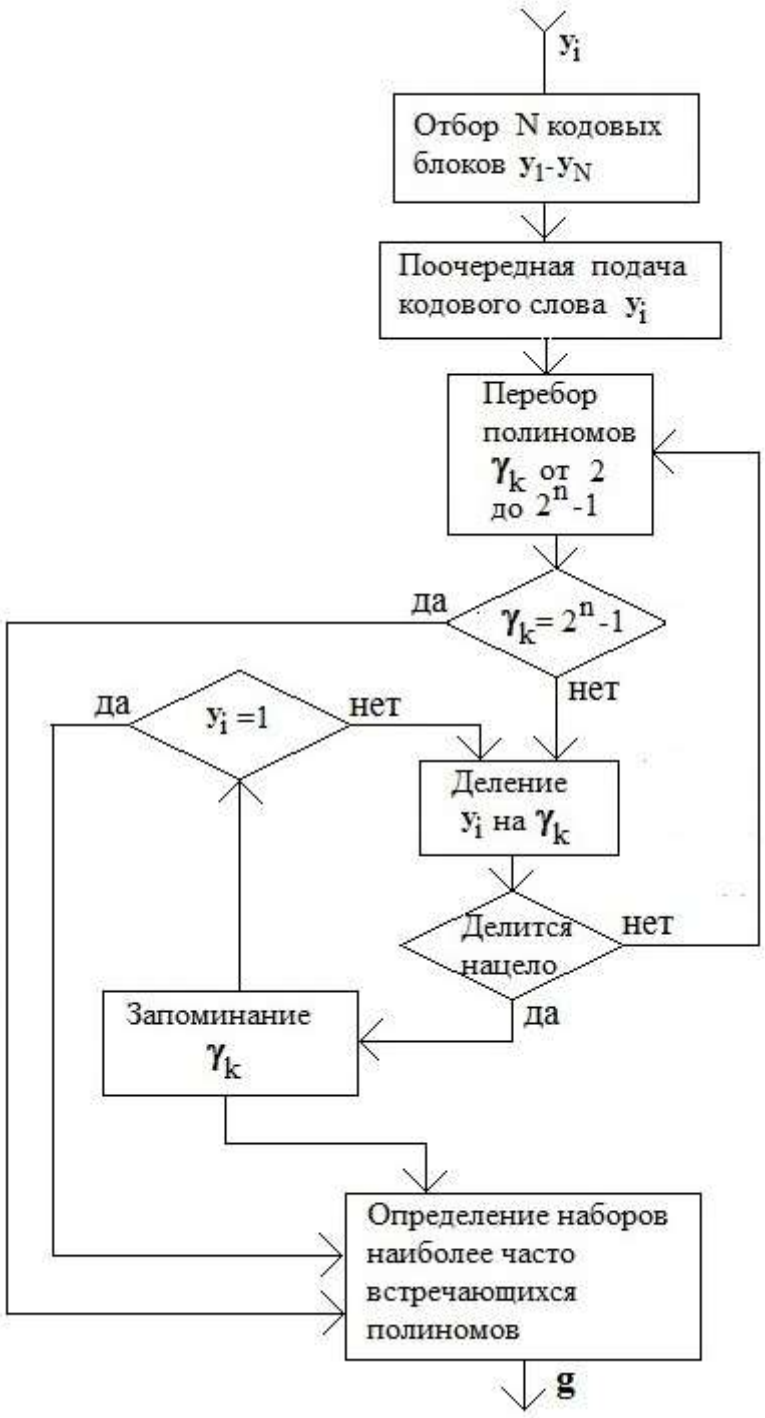

Рисунок 3.1.

Данный алгоритм реализован, как часть, в программном комплексе [100].

При исследованиях работы алгоритма задается длина k информационной части кодовых блоков и вид порождающего полинома g. Также задается различное число N кодовых блоков в анализируемом наборе. После этого информационная часть блоков имитируется случайной последовательностью

нулей и единиц длиной k, в которой появление обоих вариантов значения символов равновероятно.

После этого производится блоковое кодирование, и полученный набор кодовых слов подвергается обработке с помощью описываемого алгоритма. Выявляются общие для всех блоков простые множители и из их произведения формируется оценка порождающего полинома.

Некоторые примеры результатов исследования его свойств иллюстрированы рисунками 3.2-3.25. Они сгруппированы в три группы, соответствующие разным исходным параметрам кодера.

Во всех группах количество наборов кодовых блоков равняется N=6. Такое небольшое количество обуславливается тем, что уже при этом значении выявляются простые множители, составляющие порождающий полином, и увеличение кодовых блоков в анализируемом наборе не приводит к изменению результата.

Рисунки 3.2-3.9. относятся к значению порождающего полинома: $\mathbf{g}=11_{10}=1011=X^3+X+1$. Рисунки 3.10-3.25 относятся к значению порождающего полинома: $\mathbf{g}=127_{10}=1111111=X^6+X^5+X^4+X^3+X^2+X+1$. Рисунки 3.2-3.17 относятся к значению длины информационной последовательности, равной k=5 рисунки 3.18-3.25 относятся к значению длины кодовой последовательности, равной k=7.

Рисунки каждой группы скомпонованы по одинаковому принципу. Первые шесть рисунков в каждой группе (соответственно, 3.2-3.7, 3.10-3.15 и 3.18-3.23) показывают, сколько и каких полиномов-делителей обнаружено в каждом из шести кодовых слов. По горизонтальной оси отложена в десятичном выражении величина полиномов, По вертикальной оси отложено количество полиномов-делителей каждого значения.

Рисунки 3.8, 3.16 и 3.24 показывают полученные результаты в наглядной форме. Они сведены в матрицу, которая интерпретирована, как изображение. По горизонтально оси отложены значения полиномов-делителей, соответствующие номерам столбцов матрицы, по вертикальной оси отложены номера кодовых блоков в наборе, соответствующие номерам строк матрицы. Цвет

соответствующего элемента матрицы показывает количество полинома данного вида (Потемнение от белого к черному указывает на возрастание значения элемента матрицы.)

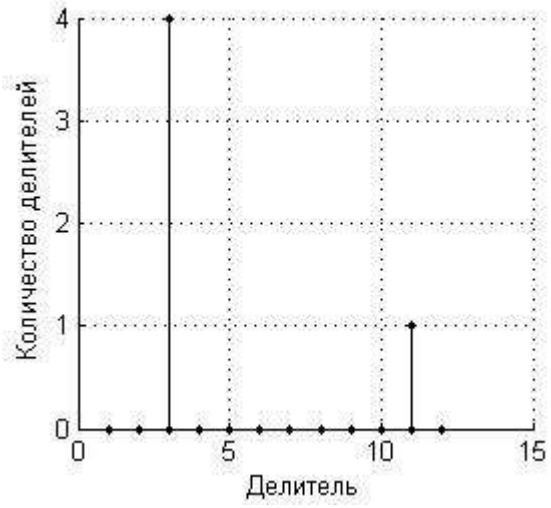

Рисунок 3.2.

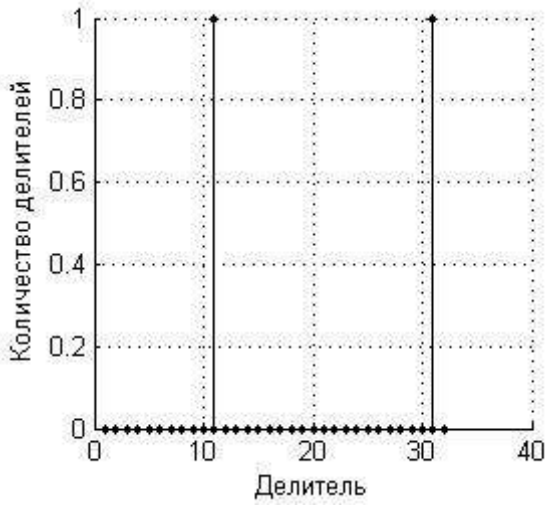

Рисунок 3.3.

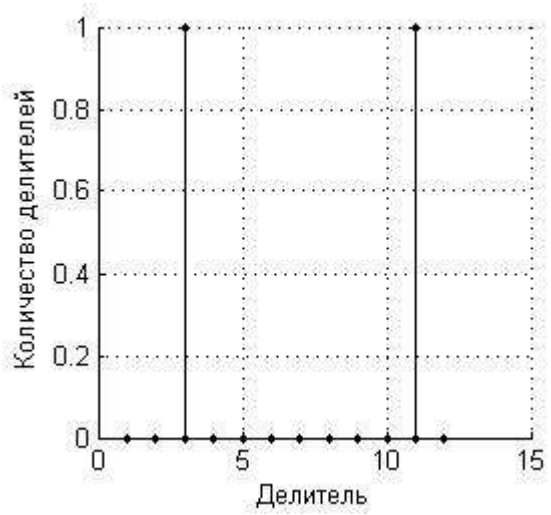

Рисунок 3.4.

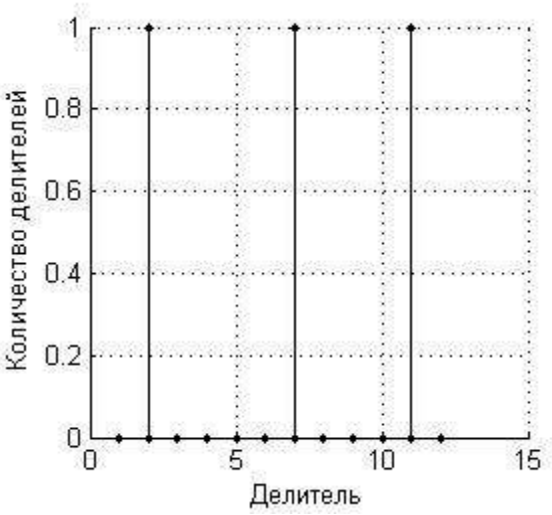

Рисунок 3.5.

Рисунки 3.9, 3.17 и 3.25 показывают обнаруженные общие для всех блоков множители, как конечный результат анализа. Рисунки указывают на правильный результат диагностики. В частности, полином $\mathbf{g}=11_{10}$ является простым и на более короткие полиномы не разлагается.

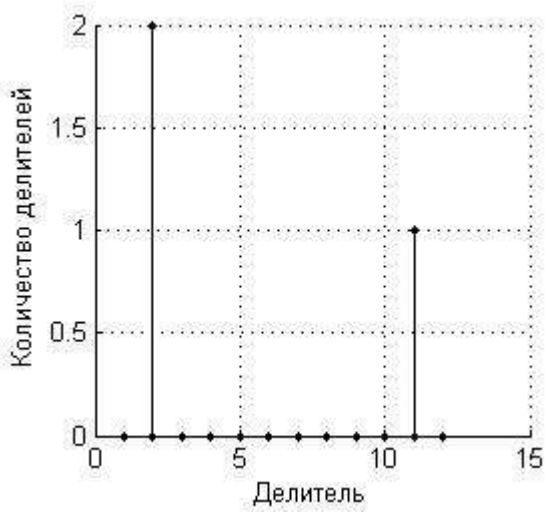

Рисунок 3.6.

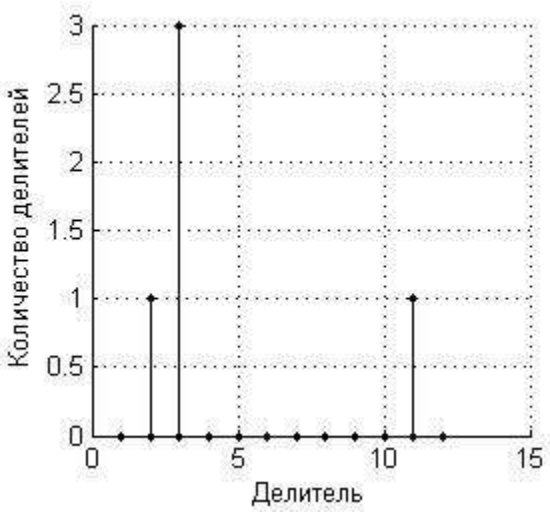

Рисунок 3.7.

Также и полином :

$$\mathbf{g}=127_{10}=X^6+X^5+X^4+X^3+X^2+X+1=(X^3+X^2+1)(X^3+X+1)=11_{10}\times 13_{10}.$$

Он является произведением двух более коротких простых полиномов 11_{10} и 13_{10}, что видно на рисунках 3.17 и 3.25.

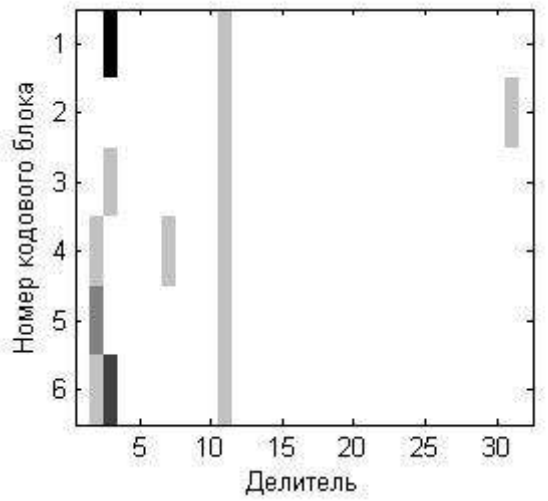

Рисунок 3.8.

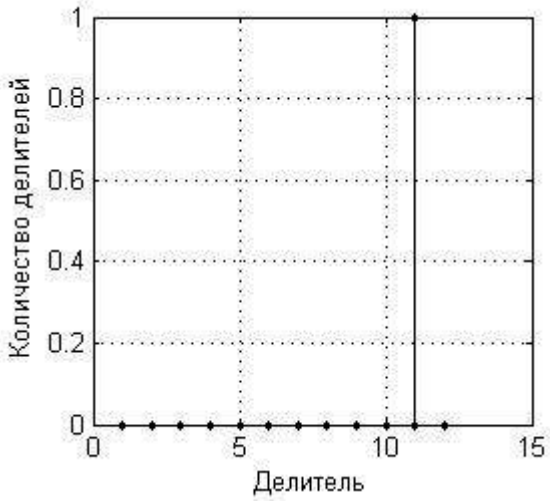

Рисунок 3.9.

Оценим вероятность неправильной диагностики. Поскольку в литературе не описана точная формула появления простых полиномов в возрастающем ряде полиномов, (как и не известна соответствующая формула, относящаяся к натуральному ряду), используется грубая оценка.

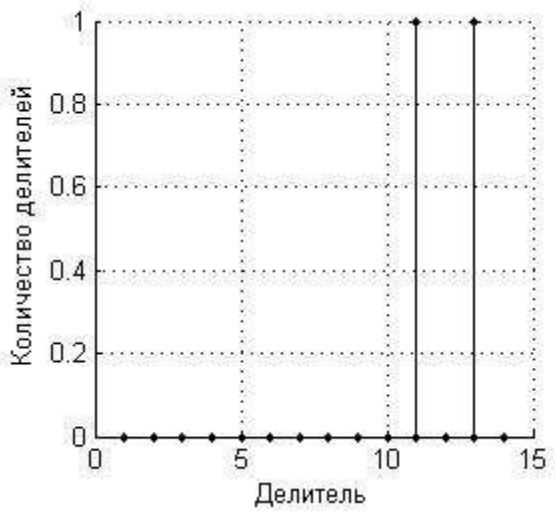

Рисунок 3.10.

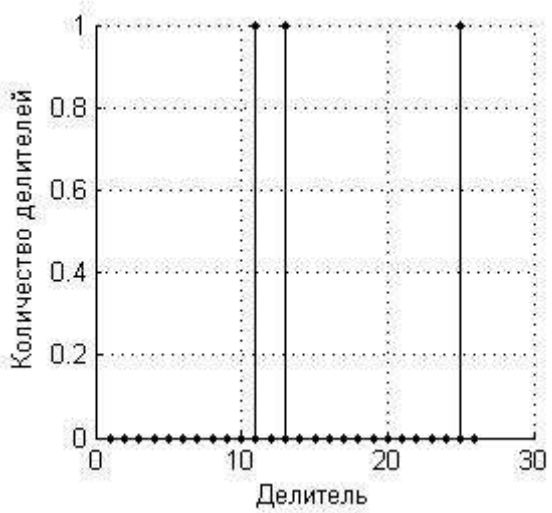

Рисунок 3.11.

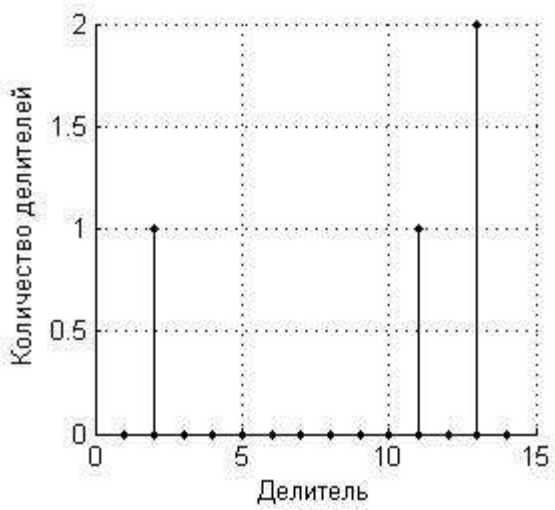

Рисунок 3.12.

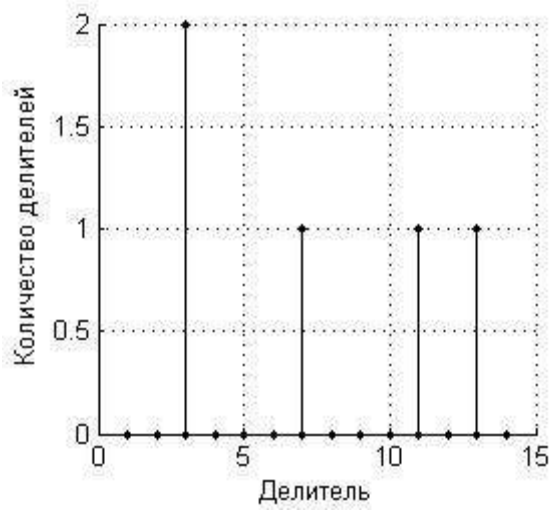

Рисунок 3.13.

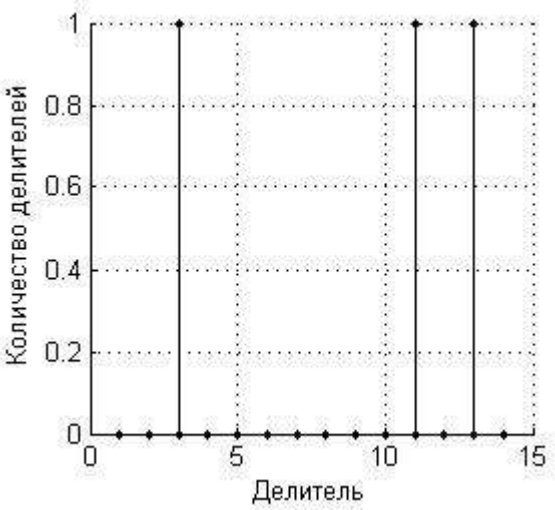

Рисунок 3.14.

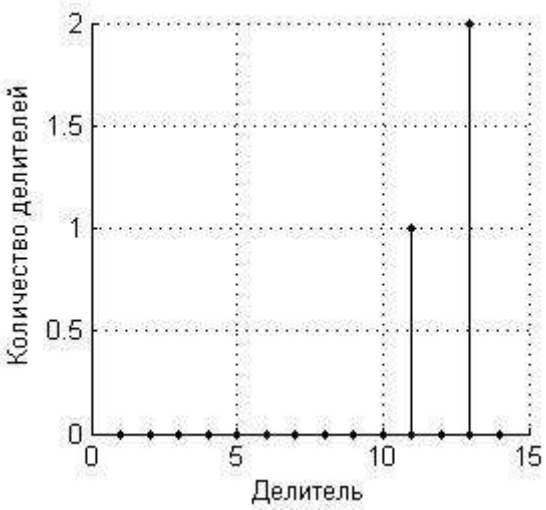

Рисунок 3.15.

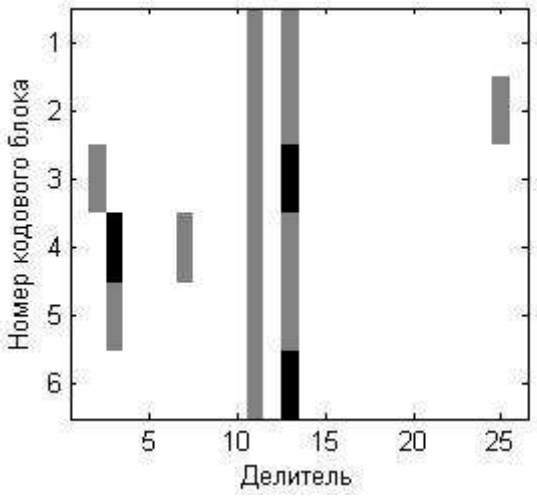

Рисунок 3.16.

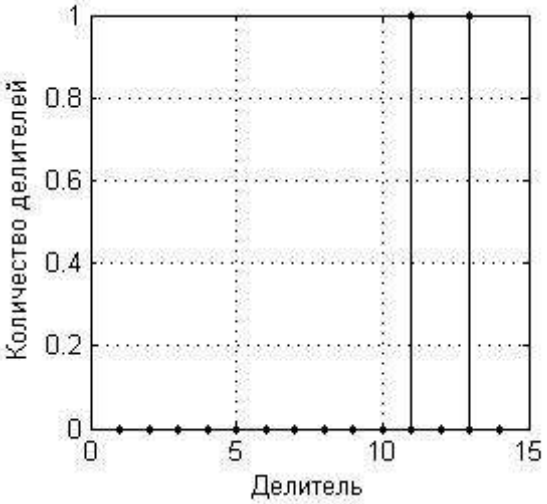

Рисунок 3.17.

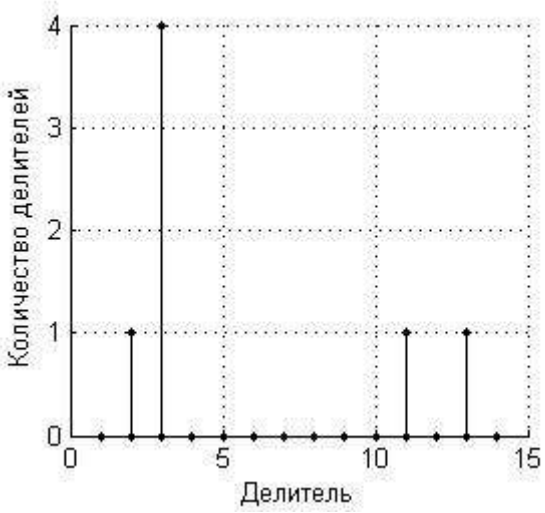

Рисунок 3.18.

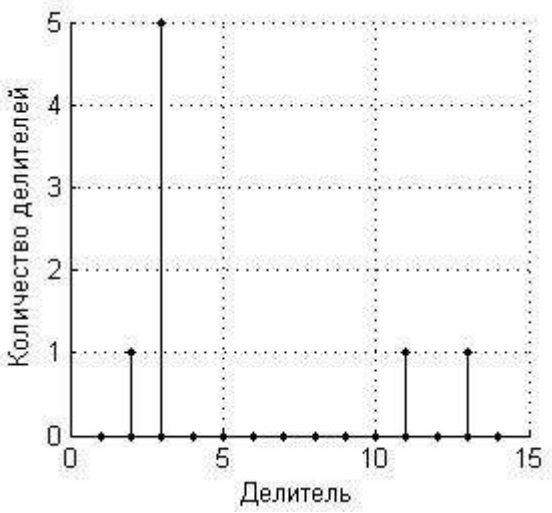

Рисунок 3.19.

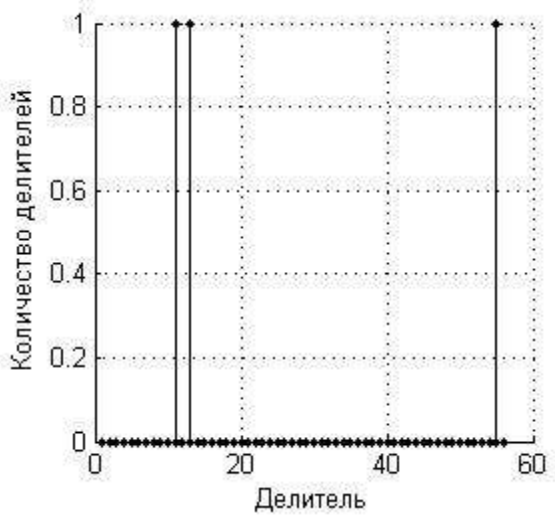

Рисунок 3.20.

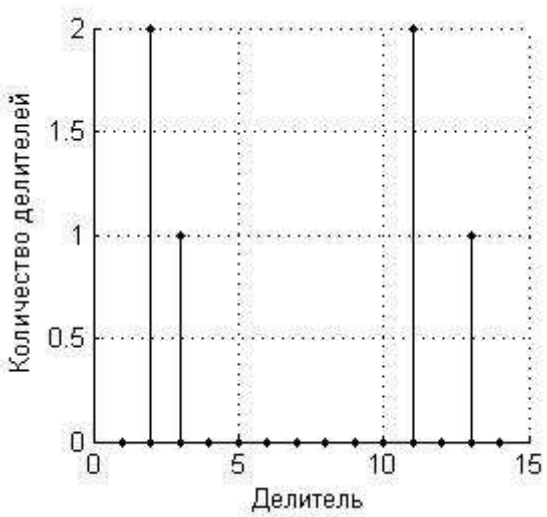

Рисунок 3.21.

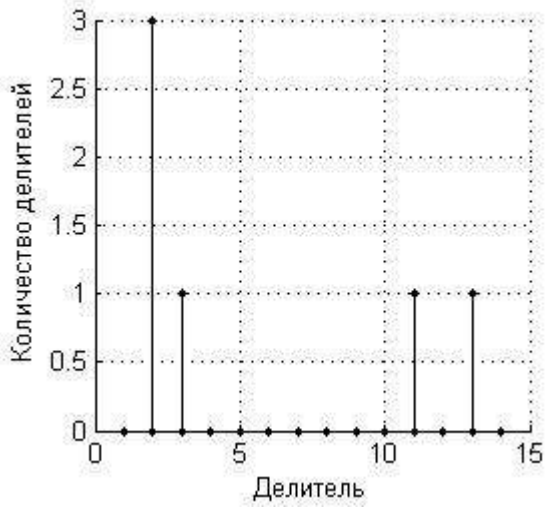

Рисунок 3.22.

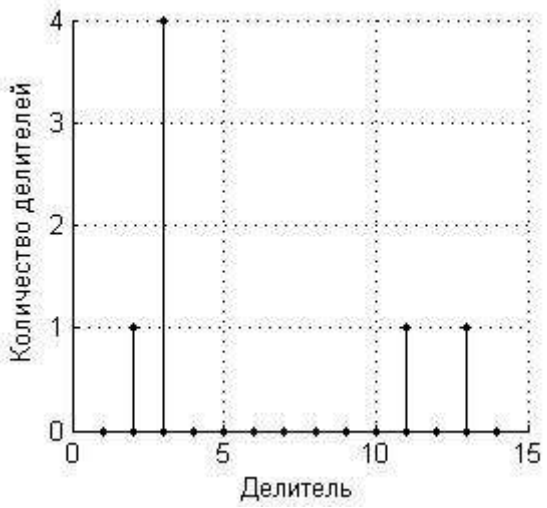

Рисунок 3.23.

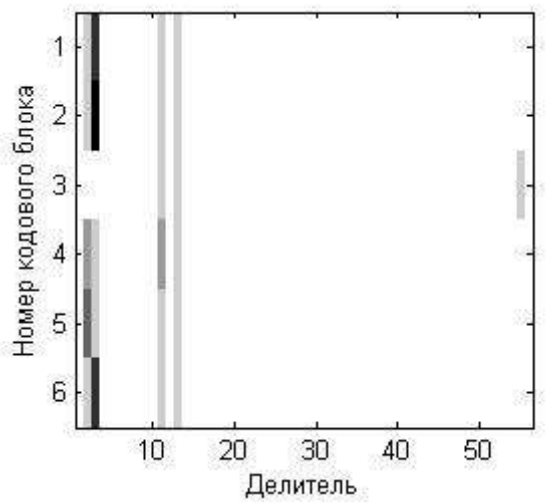

Рисунок 3.24.

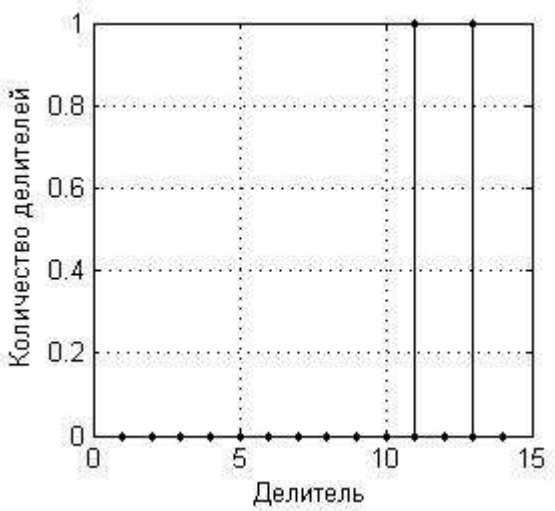

Рисунок 3.25.

Ошибка может появиться, если во всех информационных частях кодовых блоков появится один и тот же полином, не входящий в порождающий полином. Поскольку очевидно, что самым распространенным полиномом является полином, равный X (половина всех кодовых блоков делится на 2), то рассмотрим этот полином. Вероятность, что информационная часть конкретного кодового бока окажется четной, равна 1/2. Поэтому вероятность, что в информационных частях N рассматриваемых блоков появится множитель 2, равна 2^{-N}. При увеличении количества анализируемых кодовых блоков эта величина быстро убывает, становясь меньше типового значения допустимой символьной ошибки.

Множители, большие, чем 2, появляются в полиномах, описывающих информационную часть кодовых блоков, с меньшей вероятностью, поэтому их влияние на появление диагностической ошибки существенно меньше.

Главным недостатком описываемого диагностического алгоритма с непосредственным вычислением простых множителей является существенное возрастание длительности вычислений при увеличении величины кодового слова n. Каждый кодовый блок длиной n символов должен быть проанализирован на декомпозицию на простые множители. Перебор чисел M от 2 до $M=2^n$ в отношении разложения на простые множители (число M представляется в двоичном варианте) покажет, что некоторые числа будут простыми (их полиномы являются простыми и на более короткие полиномы не разлагаются), а некоторые будут сложными, разлагаемыми на более простые полиномы. Эта операция с ними занимает более длительное время. Но даже если считать, что все операции занимают одинаковое время, то анализ каждого числа займет минимум $M=2^n$ элементарных операций разложения на множители.

Таким образом, с учетом сказанного, время обработки результатов возрастает минимально в пропорции к величине 2^n от длины n кодового блока. В практическом применении время обработки при увеличении длины кодового блока растет очень быстро.

Выше рассматривалась ситуация, когда уровень шумов, вызывающих символьные ошибки, был незначительным. Но во многих случаях, как говорилось

выше, системы передачи информации работают при значительном уровне шумов, вызывающих появление ошибочных символов. На работе описываемого алгоритма диагностики это скажется следующим образом.

Появление ошибочного символа – случайное событие, и место искаженного символа в кодовом блоке также случайно. В результате кодовый блок уже не будет представлять произведение порождающего полинома на полином информационного блока, а некоторый случайный полином. В результате его разложение на простые полиномы также будет случайно, и простых множителей, из которых составлен порождающий полином, вероятно, такой кодовый блок содержать не будет.

Таким образом, если в данном канале передачи информации вероятность символьной ошибки равна P_B, то после окончания процедуры анализа N кодовых блоков в блоке памяти, относящихся к правильным результатам, вместо N результатов будет зафиксировано $P_B N$ результатов, а во всех других ячейках памяти результат, меньший $0{,}5N$, не изменится. Поэтому описанная операция выбора максимального элемента памяти также выдаст правильный результат, несмотря на появление ошибочных символов.

Описанный алгоритм, который является естественным простым решением диагностической задачи, имеет смысл применять только при относительно коротких кодовых блоках. Для более длинных кодовых блоков необходимо использовать другие алгоритмы, укорачивающие время анализа. Описанию их существа и особенностей посвящены следующие параграфы.

3.3. «Быстрые» алгоритмы диагностики циклических кодов

Рассматриваемые в этом параграфе алгоритмы 1, 2 и 3 основаны на выборочном использовании полиномов-множителей с учетом известных свойств делимого. Быстрый алгоритм 4 базируется на попарном сравнении кодовых блоков без учета их информационного содержания. Ускорение анализа основано на различных свойствах полиномов кодовых блоков. Может быть предложено несколько алгоритмов.

Алгоритм 1. Как известно, полиномы в полях Галуа проявляют много свойств, присущих математическим объектам линейной алгебры. В частности, если некоторое двоичное число q, которое описывается полиномом Q, разлагается на простые полиномы, то максимальным полиномом в его разложении, кроме самого числа q, может быть полином, не больший, чем $q/2$. Поэтому, при использовании алгоритма, описанного в предыдущем параграфе, не обязательно анализировать разложимость очередного числа-полинома M на все возможные сомножители. Достаточно проверить все полиномы до $M/2$ и отдельно полином, равный M.

Действительно, пусть в разложении числа M есть какой-либо множитель M_1, такой, что $M/2 < M_1 < M$. Тогда $M=M_1M_2$, где $M_2<M/2$. Как уже говорилось, если описанный алгоритм обнаруживает какой-либо делитель, он делит анализируемое число на этот делитель и далее анализирует полученное от деления частное. Поэтому делитель M_2 будет обнаружен не доходя до $M/2$, и там же после этого будет найден делитель M_1. Анализ ведется последовательно, поэтому все возможные варианты делителей до $M/2$ будут рассмотрены, и после него остается проверить лишь принадлежность самого M к простым полиномам.

Эта модификация позволяет уменьшить общее количество требуемых операций примерно в два раза.

Алгоритм 2. Этот алгоритм предполагает использование достаточно быстродействующей памяти большого объема. В ней для каждого возможного числа M должны быть записаны все простые полиномы, на которые оно разлагается. При поступлении очередного кодового блока из памяти сразу

извлекаются все его множители-полиномы. Быстродействие такого алгоритма определяется быстродействием устройств памяти. Однако и здесь имеется ограничение длины кодовых блоков, т.к. уже когда их длина превышает тридцать символов, размеры памяти могут составлять десятки гигабайт и более.

Алгоритм 3. Отличие от предыдущего алгоритма состоит в том, что в памяти хранятся не разложение на множители чисел от 2 до 2^k–1, а только простые полиномы. Когда поступает очередной кодовый блок, то проверяется делимость его полинома, равного *M*, не на все числа от 2 до *M*, а только на имеющиеся в памяти полиномы в порядке возрастания их величины. Если установлена делимость числа, то процедура повторяется уже с частным от деления.

На рисунке 3.26 показана относительная частота появления различных простых полиномов, полученная с помощью [100]. Исследованы простые полиномы от 2 до 120, их величина отложена по горизонтальной оси. Частота появления нормирована относительно частоты появления полинома *X*, т.е. равного 2. Зависимость частоты появления простого полинома-множителя от величины сложного полинома может относительно точно аппроксимирована степенной зависимостью с отрицательным показателем степени. Это поясняется с помощью рисунка 3.27, где рассмотрены простые полиномы до полинома, равного 1001.

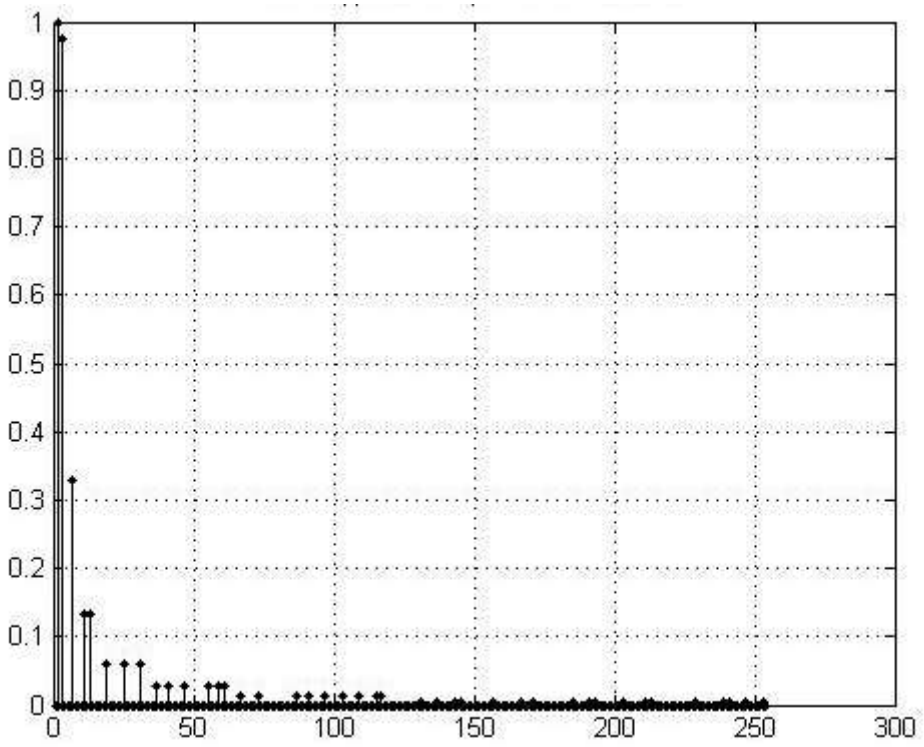

Рисунок 3.26.

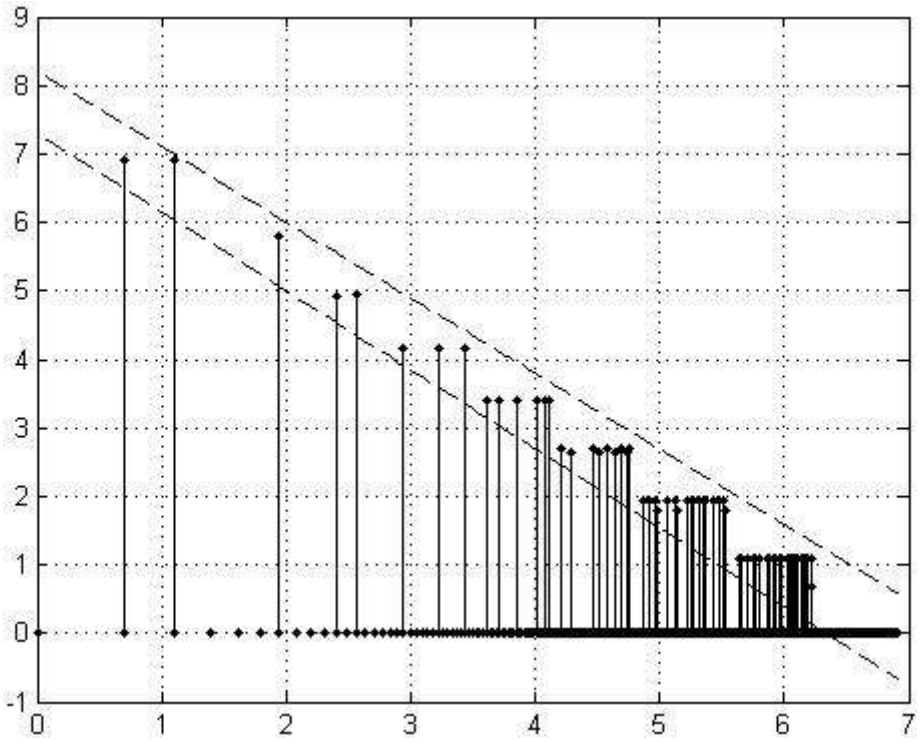

Рисунок 3.27.

Для удобства оценки зависимости частоты появления полинома от его значения величины по обеим шкалам отложены в логарифмической системе координат. В такой системе координат частоты появления простых полиномов

лежат между двумя прямыми. Наклон прямых равен, соответственно, –1,1 и –1.15. Это означает, что без логарифмического преобразования частоты появления убывают примерно обратно пропорционально величине полиномов. (Верхняя штриховая полоса означает зависимость, пропорциональную $1/M^{1,1}$, нижняя штриховая полоса – зависимость, пропорциональную $1/M^{1,15}$). Общая формула, определяющая точное количество простых полиномов, на которые разлагается произвольное число, описываемое составным полиномом M, неизвестна, однако количество таких простых полиномов существенно меньше, чем величина M. Это ослабляет требования на объем памяти по сравнению с алгоритмом 2, но удлиняет время обработки.

Алгоритм 4. Алгоритм основан на том, что для собственно диагностики не имеет значения информационная часть блоков, поэтому она может быть для удобства изменена любым образом при условии, что проверочная часть блоков останется прежней. Это позволяет с помощью определенной предварительной обработки блоков существенно сократить время анализа.

Как уже рассматривалось, при несистематическом кодировании каждый i-й кодовый блок может быть описан в виде произведения двух полиномов: $\mathbf{y}_i(X)=\mathbf{m}_i(X)\mathbf{g}(X)$, где $\mathbf{g}(X)$ – порождающий полином используемого кодера, общий для всех кодовых слов; $\mathbf{m}_i(X)$ – часть i–го кодового слова, содержащая передаваемую в нем информацию. То есть, все кодовые блоки имеют как минимум один общий полином $\mathbf{g}(X)$. Максимальная степень полинома $\mathbf{g}(X)$ равна b. Степень полинома $\mathbf{m}_i(X)$ зависит от передаваемой в данном блоке информации и может достигать максимальной величины, равной k.

При систематическом циклическом кодировании кодовый блок в кодере формируется по-другому. Для этого исходный двоичный информационный полином $\mathbf{m}_i(X)$ первоначально домножается на X^b, что соответствует сдвигу на b разрядов в сторону увеличения. В результате получается полином $\mathbf{m}_i(X)X^b$ с максимальной степенью, в общем случае равной $n=k+b$. Далее он делится на порождающий полином $\mathbf{g}(X)$. Максимальная степень образующегося остатка $\mathbf{r}_i(X)$ равна b, т.е. равна количеству нулей в записи $\mathbf{m}_i(X)X^b$, начиная с первого разряда.

После этого остаток $\mathbf{r}_i(X)$ складывается с полиномом, образуя передаваемый по каналу передачи кодовый блок $\mathbf{y}_i(X)=\mathbf{m}_i(X)X^b+\mathbf{r}_i(X)$. Поскольку операции сложения и вычитания по модулю 2 эквивалентны, то и в этом случае кодовый блок кратен полиному $\mathbf{g}(X)$, т.е. порождающий полином является одним из множителей полинома $\mathbf{y}_i(X)$, и можно записать $\mathbf{y}_i(X)=\mathbf{M}_i(X)\mathbf{g}(X)$. При стандартном декодировании именно этот факт используется в приемнике для вычисления синдромов ошибок и их исправления.

Если различные кодовые блоки сравнивать попарно, то наборы всех множителей, на которые разлагаются их полиномы, будут различаться, но множитель $\mathbf{g}(X)$ будет присутствовать *во всех* кодовых блоках. Естественно, иногда могут появляться одинаковые множители и в информационной части $\mathbf{M}_i(X)$ разных кодовых слов.

Рассмотрим два каких-либо различных кодовых блока. Обозначим одинаковые множители в информационной части сравниваемых кодовых блоков через $\mathbf{M}_C(X)$, а различающиеся части через $\mathbf{M}_{d1}(X)$ и $\mathbf{M}_{d2}(X)$. Если взять другую пару кодовых блоков, то в ней часть $\mathbf{M}_C(X)$ и части $\mathbf{M}_{d1}(X)$ и $\mathbf{M}_{d2}(X)$ будут в общем случае отличаться от таких же частей в первой паре, в третьей паре отличаться от первых двух, и т.д. Таким образом, если сравнивать достаточно большое количество пар, то, несмотря на то, что в каждой паре по отдельности кроме полинома $\mathbf{g}(X)$ иногда будут общими и другие полиномы, но во всей совокупности анализируемых кодовых блоков общим множителем будет только искомый порождающий полином $\mathbf{g}(X)$.

Обобщенная структурная схема операций, реализующих данный принцип, представлена на рисунке 3.28.

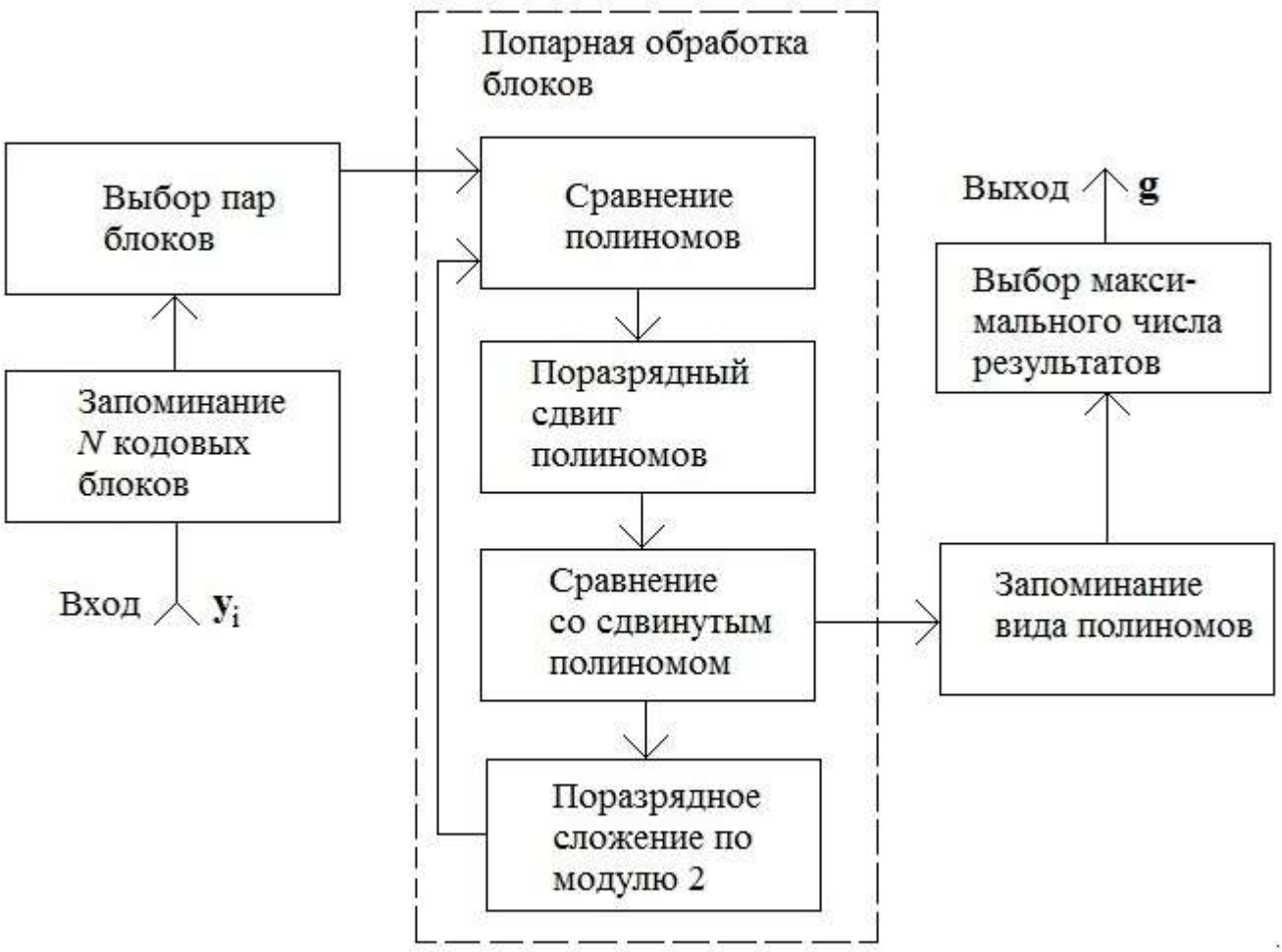

Рисунок 3.28.

Обработка начинается с выбора и запоминания N кодовых блоков $\mathbf{y}_i$. Поскольку все кодовые блоки в общем случае одинаковые по свойствам, которые используются при диагностике, то могут быть выбраны любые блоки из принятых. Количество N должно быть достаточно большим.

Поскольку информационные части разных блоков в подавляющем большинстве случаев различаются, то и появление одинаковых множителей в информационной части достаточно маловероятно, значительно вероятнее, что общая часть будет состоять только из порождающего полинома. Большое количество N необходимо, чтобы при анализе вероятность получения именно искомого порождающего полинома была значительно больше, чем других полиномов. При увеличении N эта вероятность также возрастает, что дает возможность уменьшить ошибку диагностики до требуемых уровней.

Как и при использовании алгоритмов главы 2 присутствие помех заметного уровня при передаче по каналу связи не приводит к существенному ухудшению работы диагностического алгоритма. Появление ошибочных символов лишь проявляется в том, что такие блоки разлагаются на другие полиномы-множители, в наборе которых скорее всего сам полином **g** будет отсутствовать.

Из запомненных кодовых блоков можно составить $N(N–1)/2$ различающихся пар. Пусть P_1 – вероятность появления ошибочных символов в одном кодовом блоке. Если уровень помех незначителен, то после анализа «правильный» вид искомого порождающего полинома будет получен $N(N–1)/2$ раз. А в случае воздействия помех значительного уровня «правильный» полином будет обнаружен в среднем $(1–P_1)\ N(N–1)/2$ раз. Изменение вида полиномов из-за воздействия помех, относящихся к информационной части блока, влияния на работу алгоритма не оказывает, поскольку от конкретного вида информационной части его работа не зависит. Поскольку в практических условиях величина P_1 достаточно мала, то снижение количества раз появления «правильного» вида полинома будет также незначительно, и заметного влияния в целом помехи не оказывают.

Далее производится попарная обработка выбранных блоков. В ее результате определяется и запоминается вид полинома-множителя, общего для анализируемой пары блоков. После завершения анализа всех сочетаний кодовых блоков осуществляется выбор вида полиномов, запомненного максимальное число раз. Вероятность появления в каждом цикле анализа только порождающего полинома $\mathbf{g}(X)$ гораздо выше, и именно он будет определен максимальное число раз, а все другие результаты, появляющиеся случайно, будут зафиксированы меньшее число раз. Поэтому данная операция определит именно полином $\mathbf{g}(X)$, являющийся искомым результатом диагностики.

Операция попарной обработки блоков состоит из нескольких последовательных операций. Первоначально производится сравнение полиномов двух анализируемых кодовых блоков, и определяются максимальные степени

полиномов обоих блоков. Пусть полиномы первого и второго кодовых блоков имеют вид, соответственно:

$$\mathbf{y}_1(X)=a_S X^S+a_{S-1}X^{S-1}+a_{S-2}X^{S-2}+\ldots+a_0,$$

$$\mathbf{y}_2(X)=b_T X^T+b_{T-1}X^{T-1}+b_{T-2}X^{T-2}\ldots+b_0,$$

где $a_S=b_T=1$; S и T – максимальные степени полиномов. Фактически, сравниваются величины S и T.

Предположим, оказалось, что $S>T$, т.е. полином $\mathbf{y}_1$ больше, чем полином $\mathbf{y}_2$. Тогда после этого производится поразрядный сдвиг меньшего полинома вверх на $S–T$ разрядов. Для этого второй полином домножается на X^{S-T}. Если же второй полином больше первого, то тогда первый полином домножается на необходимую величину разности максимальных степеней, т.е. сдвигается на соответствующее разностное число разрядов. В том случае, если максимальные степени обоих полиномов равны, (т.е. $S=T$), то никакого поразрядного сдвига не производится. Таким образом, после данной операции максимальные степени обоих полиномов станут совпадать.

После этого сравнивается величина большего полинома и результата сдвига меньшего полинома. При этом возможны две ситуации. Если в результате сдвига меньший полином становится в точности равным большему полиному, то попарная обработка анализируемых блоков прекращается. Вид меньшего полинома до сдвига запоминается и начинается анализ следующей пары кодовых блоков.

Если же сравниваемые полиномы не равны между собой, то далее производится их поразрядное сложение (вычитание) по модулю 2. Поскольку их максимальные степени до этого были выровнены, то после осуществления этой операции максимальная степень разности уменьшается на единицу. После этого алгоритм вновь возвращается к сравнению, но уже двух других полиномов, из которых один равен упомянутому меньшему полиному до его поразрядного

сдвига, а другой – результату сложения по модулю 2. После нее вновь выполняются описанные операции, пока не будет достигнуто точное равенство обрабатываемых полиномов.

Поясним работу алгоритма подробнее. Когда процедура попарной обработки блоков заканчивается, это приводит к тому, что в результате остаются общие для обоих анализируемых в ней кодовых блоков полиномы. Действительно, пусть сравниваемые кодовые блоки описываются полиномами: $\mathbf{y}_1(X)=\mathbf{M}_{d1}(X)\mathbf{M}_C(X)\mathbf{g}(X)$ и $\mathbf{y}_2(X)=\mathbf{M}_{d2}(X)\mathbf{M}_C(X)\mathbf{g}(X)$.

Как известно, поразрядные сложение и вычитание двоичных чисел по модулю 2 являются эквивалентными операциями, т.к. приводят к одинаковым результатам. Поэтому разностный полином равен:

$$\mathbf{y}_3=\mathbf{y}_1-\mathbf{y}_2=(\mathbf{M}_{d1}+\mathbf{M}_{d2})\mathbf{M}_C\mathbf{g}= (\mathbf{M}_{d1}-\mathbf{M}_{d2})\mathbf{M}_C\mathbf{g}=\mathbf{M}_3\mathbf{M}_C\mathbf{g}.$$

Он будет содержать те же совпадающие общие множители, что и исходные полиномы до сложения, а изменяются только различающиеся в $\mathbf{y}_1$ и $\mathbf{y}_2$ части полиномов.

Поскольку перед сложением (вычитанием) максимальный порядок меньшего полинома был временно приравнен к порядку большего, то после этой операции порядок результата сложения всегда уменьшается на единицу по сравнению с порядком максимального из анализируемых полиномов. Если после сложения результат оказывается не равен меньшему из полиномов, то вновь повторяется выравнивание порядков двух анализируемых полиномов и их вычитание. Порядок различающейся части ($\mathbf{M}_3$) вновь уменьшается на единицу, и т.д. Число повторяющихся операций определяется конкретным видом полиномов $\mathbf{M}_{d1}$ и $\mathbf{M}_{d2}$ и равно порядку максимального их них.

Для каждой пары кодовых блоков производится постепенное итеративное уменьшение порядка соответствующих полиномов до тех пор, пока оба полинома различающихся частей не станут равными единице. Оставшиеся части обоих исзодных полиномов при этом равны между собой, т.е. равны общему полиному

для каждой пары. Если максимальная степень различающихся частей равна k, то в общем случае общий полином будет получен максимум за k повторений описанного набора операций.

Таким образом, после проведения определенного числа описанных повторяющихся итераций полином $\mathbf{M}_3$ становится равным единице, т.е остается только полином вида $\mathbf{M}_C\mathbf{g}$, если и в информационных частях кодовых блоков есть одинаковые полиномы, или остается полином вида $\mathbf{g}$, если состав полиномов информационных частей полностью различается.

На рисунках 3.29-3.31 в качестве примера приведены графики, отражающие поведение итерационного процесса получения общего полинома при разных исходных параметрах кода. (Цель приводимого примера – не определение вида именно порождающего полинома, а иллюстрация протекания итеративного процесса.) По горизонтальным осям обоих графиков этих рисунков отложены номера итераций, по вертикальным осям – значения двух сравниваемых полиномов в десятичной записи. Графики приведены парами. В каждой паре одинаковые исходные параметры. У графиков, расположенных слева, масштаб вертикальной шкалы – линейный, у графиков, расположенных справа, масштаб вертикальной шкалы – логарифмический. Логарифмический масштаб использован для наглядности, поскольку пределы изменения величины полиномов меняются в достаточно широких пределах.

Графики, описывающие поведение исходного большего полинома, обозначены цифрой 1, исходного меньшего полинома – цифрой 2. Цифрой 3 обозначен достигаемый уровень общего полинома $\mathbf{M}_C\mathbf{g}$.

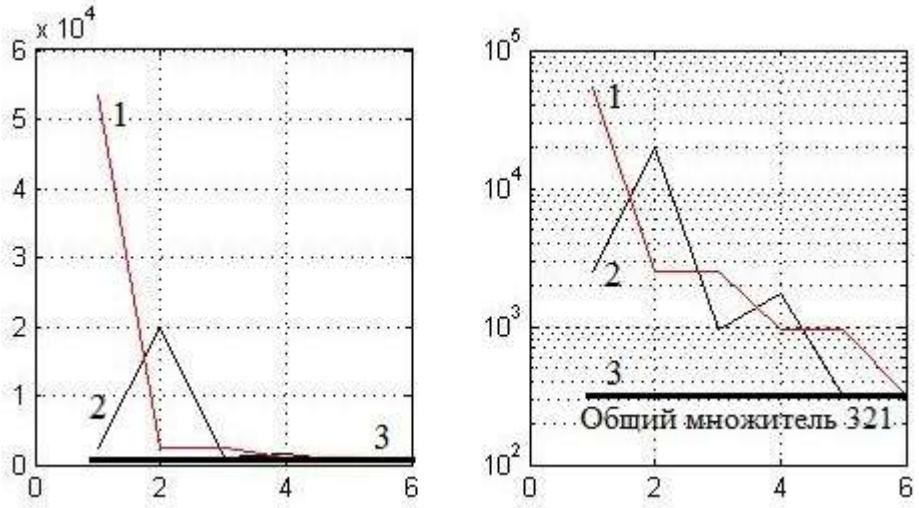

Рисунок 3.29 – n=18, $\mathbf{M}_C\mathbf{g}$=321.

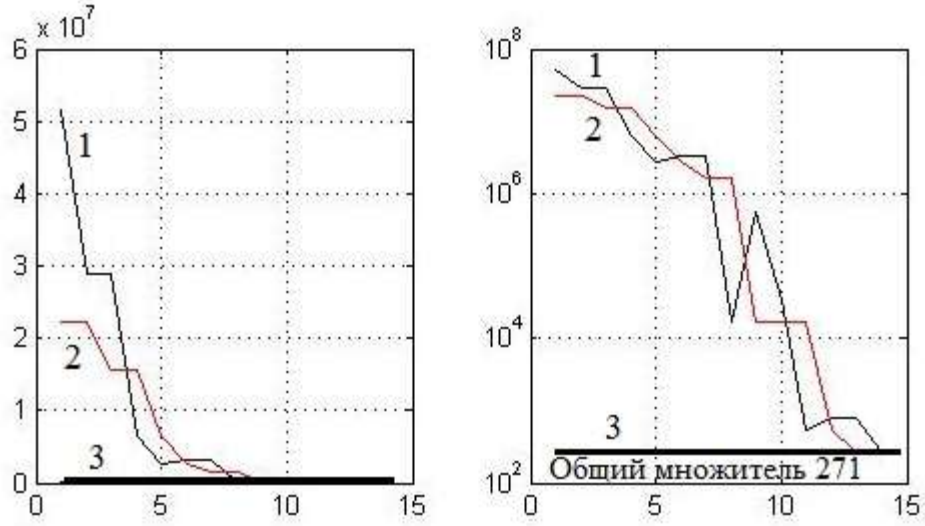

Рисунок 3.30 – n=28, $\mathbf{M}_C\mathbf{g}$ =271.

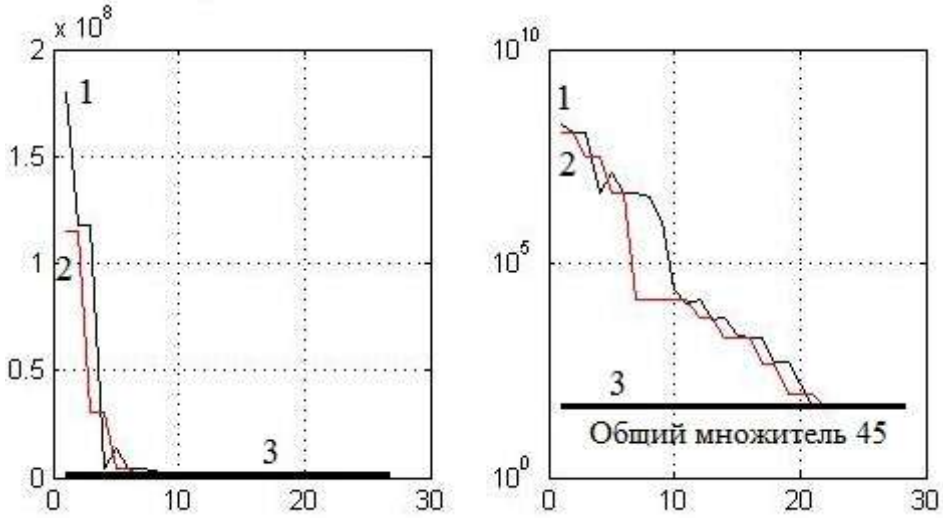

Рисунок 3.31 – n=30, $\mathbf{M}_C\mathbf{g}$ =45.

Графики показывают, что к общему полиному решение сходится достаточно быстро. Скорость схождения зависит от длины кодовых блоков. После завершения всех $N(N-1)/2$ циклов попарной обработки оказываются запомненными вид полиномов $\mathbf{g}$ и какие-то виды полиномов $\mathbf{M}_C\mathbf{g}$, относящиеся к различным парам. (Сравнительные вероятностные характеристики этих результатов будут рассмотрены далее).

В случае, если не наблюдается значительного преобладания количества одного вида запомненных полиномов, (который можно отнести к **g)** над другими запомненными видами, то операции алгоритма могут вновь повторяться, но не над полиномами $\mathbf{y}_i$, а над вновь полученными запомненными полиномами $\mathbf{g}$ и $\mathbf{M}_C\mathbf{g}$. При этом после каждого сравнения полиномы $\mathbf{g}$ также всегда остаются, а чтобы остались полиномы $\mathbf{M}_C\mathbf{g}$ других видов, одинаковые полиномы должны наблюдаться в информационных частях уже четырех различных кодовых блоков, что является существенно менее вероятными событиями. При необходимости общая процедура может аналогично повторяться и далее.

Для примера рассмотрим подробно данный набор операций с двумя произвольными полиномами, состоящими из общей части $\mathbf{g}(X)$ и различающихся частей. Вид различающихся частей значения не имеет. В качестве анализируемой пары полиномов выберем: $\mathbf{y}_1=(X^4+X+1)\mathbf{g}(X)$ и $\mathbf{y}_2=(X^3+X^2)\mathbf{g}(X)$.

В операциях происходят следующие преобразования полиномов. В выбранной виде паре полиномов первый полином $\mathbf{y}_1$ больше, чем второй полином $\mathbf{y}_2$. После первого сравнения этих двух полиномов определяется, что второй полином должен быть умножен на X, т.е. двоичное число, его описывающее, должно быть сдвинуто вверх на один разряд. После такого сдвига полиномы $\mathbf{y}_1$ и $\mathbf{y}_2X$ не становятся точно равными между собой, поэтому после сравнения они последующей операцией складываются (вычитаются). Получается полином:

$$\mathbf{y}_3=\mathbf{y}_1-\mathbf{y}_2X=(X^4+X+1)\mathbf{g}(X)+X(X^3+X^2)\mathbf{g}(X)=(X^4+X+1+X^4+X^3)\mathbf{g}(X)=(X^3+X+1)\mathbf{g}(X).$$

Результат сложения – полином $\mathbf{y}_3=(X^3+X+1)\mathbf{g}(X)$. Он вновь сравнивается с меньшим (до сдвига) полиномом $\mathbf{y}_2=(X^3+X^2)\mathbf{g}(X)$. Устанавливается, что полученный полином $\mathbf{y}_3$ меньше, чем исходный меньший полином $\mathbf{y}_2=(X^3+X^2)\mathbf{g}(X)$, однако порядки их одинаковы (порядки равны 3). Поэтому поразрядный сдвиг не производится и они складываются (вычитаются) без сдвига. В результате получается полином:

$$\mathbf{y}_4=[(X^3+X^2)\mathbf{g}(X)]+[(X^3+X+1)\mathbf{g}(X)]=(X^2+X+1)\mathbf{g}(X).$$

Теперь вновь сравниваются полиномы $\mathbf{y}_3=(X^3+X+1)\mathbf{g}(X)$ и $\mathbf{y}_4=(X^2+X+1)\mathbf{g}(X)$. Поскольку степень первого из них больше на единицу, то полином $\mathbf{y}_4$ теперь домножается на X. В результате получается полином $X\mathbf{y}_4=X(X^2+X+1)\mathbf{g}(X) = X^3+X^2+X$. Он также не равен полиному $\mathbf{y}_3$. Поэтому дальше производится суммирование по модулю 2.. После суммирования получается полином:

$$\mathbf{y}_5=\mathbf{y}_3-X\mathbf{y}_4=(X^3+X+1)\mathbf{g}(X)+X(X^2+X+1)\mathbf{g}(X)= (X^2+1)\mathbf{g}(X).$$

Далее он сравнивается с меньшим из складываемых (до домножения на X) полиномов, т.е. с полиномом $\mathbf{y}_4=(X^2+X+1)\mathbf{g}(X)$).

Полученный полином $\mathbf{y}_5$ меньше меньшего из складываемых полиномов $\mathbf{y}_4$, но их степени совпадают. В результате теперь поразрядного сдвига не производится, но сами полиномы не равны, поэтому производится их сложение (вычитание), после чего получается результат – полином

$$\mathbf{y}_6=\mathbf{y}_4-\mathbf{y}_5=(X^2+X+1)\mathbf{g}(X)+(X^2+1)\mathbf{g}(X)=X\mathbf{g}(X).$$

После этого он сравнивается с меньшим из складываемых полиномов, т.е. с полиномом $\mathbf{y}_5=(X^2+1)\mathbf{g}(X)$. Степень полученного полинома $\mathbf{y}_6=X\mathbf{g}(X)$ меньше, чем степень меньшего из складываемых полиномов (имеется в виду полином

$\mathbf{y}_5=(X^2+1)\mathbf{g}(X)$), поэтому полином $\mathbf{y}_6=X\mathbf{g}(X)$ умножается на X. После такого умножения и после вычитания получается результат:

$$\mathbf{y}_7=(X^2+1)\mathbf{g}(X)+X^2\mathbf{g}(X)=\mathbf{g}(X).$$

После этого устанавливается, что степень предыдущего меньшего полинома $\mathbf{y}_6=X\mathbf{g}(X)$ больше, чем степень полученного результата $\mathbf{y}_7=\mathbf{g}(X)$. В следствие этого меньший полином $\mathbf{y}_7$ в операции 6 домножается на соответствующий множитель, определяемый разностью степеней, т.е. на X. После этого сравниваемые полиномы приобретают вид: $\mathbf{y}_6=X\mathbf{g}(X)$ и $\mathbf{y}_7=X\mathbf{g}(X)$, т.е. становятся точно равными один другому.

Устанавливается факт этого получившегося теперь равенства, после чего цикл попарной обработки данной пары кодовых блоков завершается. Вид меньшего до сдвига полинома $\mathbf{g}(X)$ передается к операции запоминания вида полученных полиномов. В ней фиксируется выработанный при анализе данной пары кодовых блоков результат, т.е. найденное значение общего полинома.

Обработка с помощью описанного диагностического анализа может осуществляться с помощью структурной схемы, приведенной на рисунке 3.32.

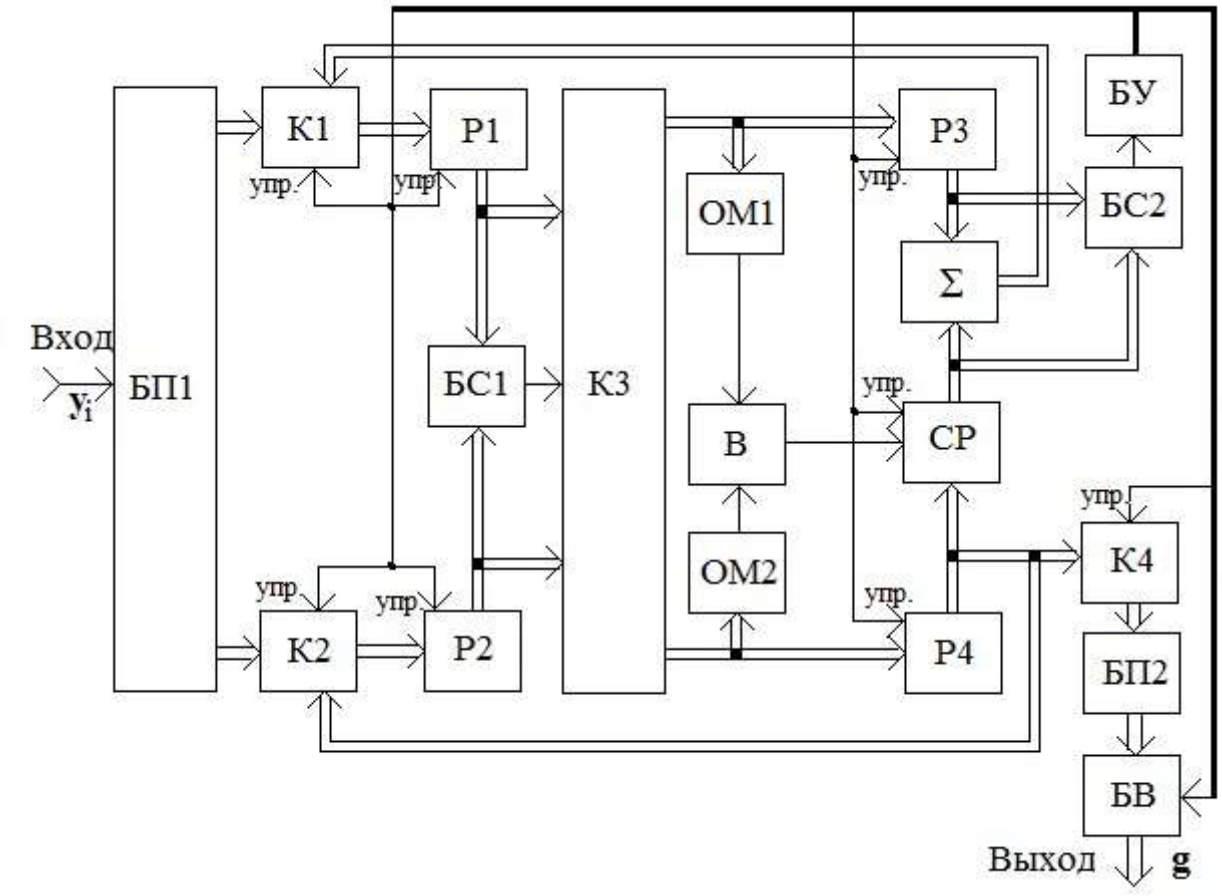

Рисунок 3.32.

Блоки этой структурной схемы, как примера реализации описанного алгоритма, работают следующим образом. На первый блок памяти (БП1) поступают принятые кодовые блоки в форме последовательностей двоичных символов $\mathbf{y}_i$. В нем запоминается N различных кодовых блоков. (Последовательность запомненных блоков значения не имеет, это могут быть либо подряд следующие блоки, либо расположенные другим образом.)

Из этого блока памяти извлекаются пары различных блоков, один из этих блоков подается на вход первого коммутатора (К1), другой блок подается на вход второго коммутатора (К2). Последовательность извлечения из памяти пар блоков также значения не имеет, текущие номера извлекаемых блоков задаются блоком управления (БУ). В первом блоке памяти занесено N кодовых блоков, значит, осуществляя $N(N–1)/2$ циклов, имеется возможность анализировать до $N(N–1)/2$

различных пар блоков. В каждом цикле анализа производится одинаковый набор действий.

Цикл анализа очередной пары кодовых блоков состоит в следующем. В самом начале цикла первый и второй коммутаторы подключают первый и второй многоканальные выходы первого блока памяти БП1 на многоканальные параллельные входы, соответственно, первого и второго регистров (Р1 и Р2), где анализируемая в данный момент пара кодовых блоков записывается в форме двоичного кода.

В первом блоке сравнения (БС1) сравнивается величина кодовых блоков, записанных в регистрах Р1 и Р2. Выходной сигнал этого блока сравнения управляет третьим коммутатором (К3).

Если двоичные полиномы (кодовые блоки), записанные в регистрах Р1 и Р2, не равны между собой, то блок сравнения БС1 с помощью коммутатора К3 подает больший кодовый блок на вход первого определителя максимальной степени (ОМ1) и к параллельному входу третьего регистра (Р3), а меньший блок подается коммутатором К3 на вход второго определителя максимальной степени (ОМ2) и к параллельному входу четвертого регистра (Р4). Поступившие сигналы записываются в эти регистры. Полином, записанный в регистре Р4 также перезаписывается в сдвиговый регистр (СР).

Если же окажется, что записанные в регистрах Р1 и Р2 двоичные полиномы равны между собой, то на оба выхода коммутатора К3 подаются эти одинаковые полиномы.

В определителях максимальной степени ОМ1 и ОМ2 определяются порядки полиномов.

Далее в вычитателе (В) находится арифметическая разность их порядков (разность максимальных степеней полиномов) и на основе выходного сигнала вычитателя в сдвиговом регистре производится сдвиг всего записанного полинома в сторону увеличения степени на полученную величину этой арифметической разности. Таким образом, после этого сдвига порядки (максимальные степени) полиномов, записанных в третьем регистре Р3 и в

сдвиговом регистре СР становятся одинаковыми. Эти полиномы подаются на второй блок сравнения (БС2) и на сумматор Σ, где производится их поразрядное сложение по модулю 2.

Во втором блоке сравнения БС2 сравниваются полиномы, записанные в регистрах Р3 и СР. Если они равны между собой, то этот блок сравнения сообщает об этом блоку управления БУ. После чего данный цикл заканчивается, блок управления открывает четвертый коммутатор (К4), и сигнал с выхода регистра Р4 подается на второй блок памяти БП2. В этом втором блоке памяти каждому возможному виду полиномов соответствует своя ячейка памяти. Первоначально до начала процедуры диагностики все ячейки обнуляются. Когда в очередном цикле определен очередной общий полином, то в ячейку, ему соответствующую, прибавляется единица к той сумме, которая там уже была записана ранее.

Если блок сравнения БС2 не зафиксировал равенства полиномов в третьем и четвертом регистрах, то результат сложения в сумматоре по модулю 2 подается на другой вход первого коммутатора К1, а на другой вход второго коммутатора К2 подается полином, записанный в регистре Р4. В результате на выходы обоих коммутаторов подключаются не сигналы с блока памяти БП1, а вновь поступившие сигналы с их других входов (т.е. с сумматора и с регистра Р4). После этого вся последовательность действий повторяется. Работа коммутаторов, регистров и блоков памяти управляется блоком управления.

Когда перебраны все $N(N-1)/2$ сочетаний рассматриваемых кодовых блоков, в блоке памяти БП2 в различные ячейки оказывается записанными разное количество единиц. После этого блок выделения максимальной суммы определяет, в какой ячейке записано максимальное число. Полином, соответствующий этой ячейке, подается на выход, как конечный результат всей процедуры диагностики.

Использование описанного алгоритма 4 позволяет значительно сократить время диагностики по сравнению с алгоритмом, описанным в главе 2. Это, в свою

очередь, при ограниченных вычислительных ресурсах расширяет диапазон параметров кодера, которые могут быть успешно диагностированы.

При анализе пары кодовых блоков при каждой итерации внутри попарной обработки блоков (см. рисунок 3.3.1) максимальная степень полиномов или уменьшается, или остается прежней. Причем две итерации подряд прежней она остаться не может. Таким образом, общее максимальное количество итераций в общем случае лежит между числами k и $1{,}5k$. При грубой оценке и с учетом числа анализируемых пар общее количество итераций в среднем равно $0{,}75kN(N–1)$. Это существенно меньше необходимого количества операций декомпозиции по алгоритму главы 2, т.к. здесь в отношении длины информационной части блоков наблюдается приблизительно линейный рост времени обработки, а не экспоненциальный. К тому же каждая процедура итерации проще, чем операция декомпозиции на простые множители.

Недостатками описываемого алгоритма являются бóльшая чувствительность к символьным ошибкам, а также необходимость вычислений в полях Галуа с двоичными числами достаточно большой длины.

Для изучения свойств описываемого алгоритма были проделаны компьютерные эксперименты с большим количеством различных исходных параметров с использованием разработанного программного комплекса [100]. Информационная часть кодовых блоков моделировалась различающимися для каждого блока случайными последовательностями нулей и единиц с равной вероятностью появления. Согласно выбранным параметрам кодирования формировались N кодовых блоков. Далее из них составлялось $N(N–1)/2$ вариантов пар, которые анализировались согласно описанному алгоритму. Общий множитель каждой пары фиксировался, и по окончанию общей процедуры выбирался множитель, зафиксированный большее число раз, и выводился в качестве результата диагностики.

Далее представлены в качестве примера несколько групп рисунков, позволяющих проиллюстрировать свойства алгоритма при изменении различных параметров, и отражающих общие свойства, отмеченные во всем объеме

экспериментов. Во всех рисунках по горизонтальной оси отложены значения обнаруженных общих множителей (в десятичной записи). По вертикальной оси отложено количество таких обнаруженных множителей. Под рисунками указаны параметры, при которых они получены. Следует отметить, что каждый эксперимент даже при одних и тех же параметрах дает несколько отличающиеся результаты, т.к. информационные части анализируемого набора блоков являются независимыми случайными последовательностями, тем не менее общие свойства, позволяющие сделать соответствующие выводы, сохраняются.

Группа рисунков 3.33-3.37 иллюстрирует влияние возрастания количества символов в информационной части блоков. Возрастание количества информационных символов в блоках не приводит к существенному изменению свойств накопленных результатов определения общих множителей, сохраняется преобладание вида, соответствующего исходному порождающему полиному.

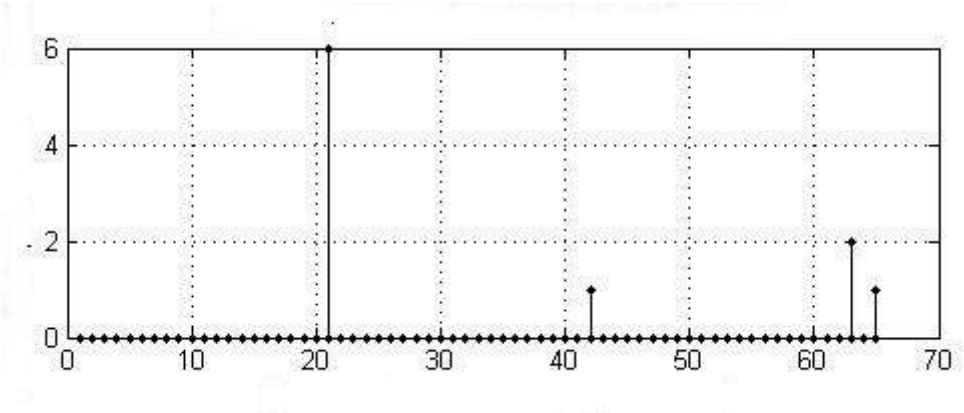

Рисунок 3.33 – k=5, N=5, **g**=21.

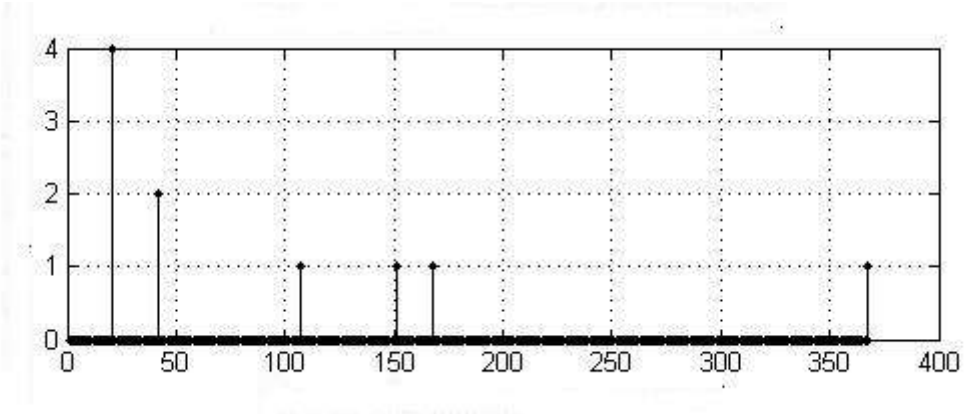

Рисунок 3.34 – k=10, N=5, **g**=21.

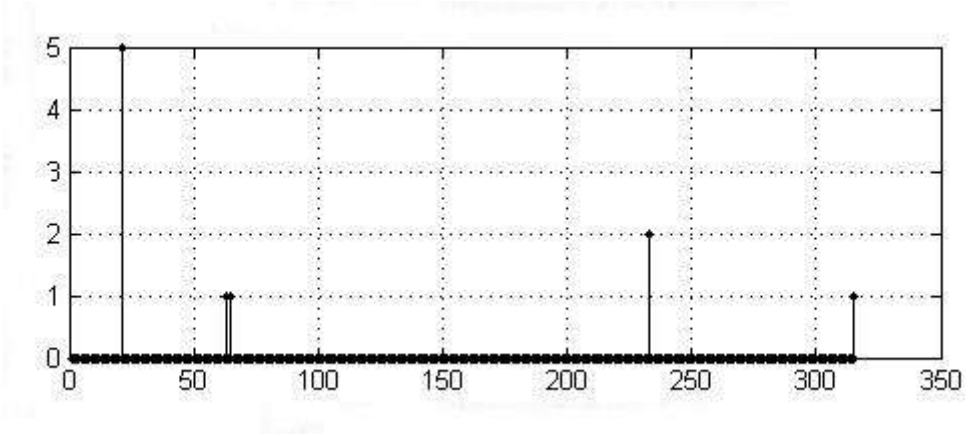

Рисунок 3.35 – k=15, N=5, **g**=21.

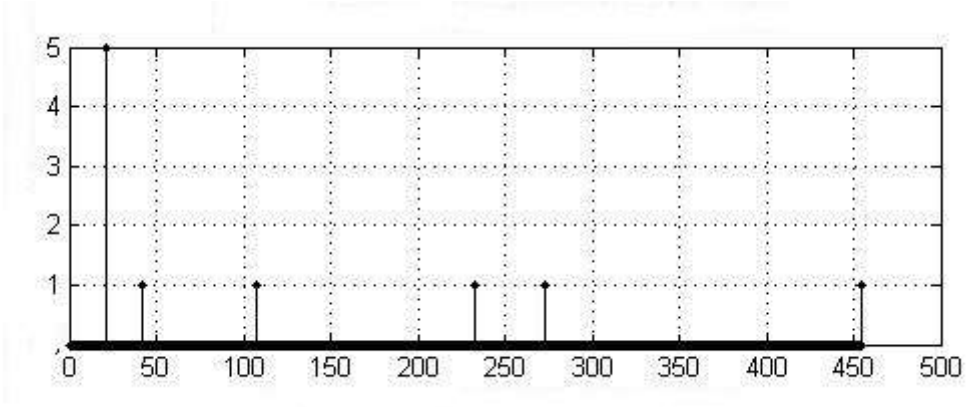

Рисунок 3.36 – k=20, N=5, **g**=21.

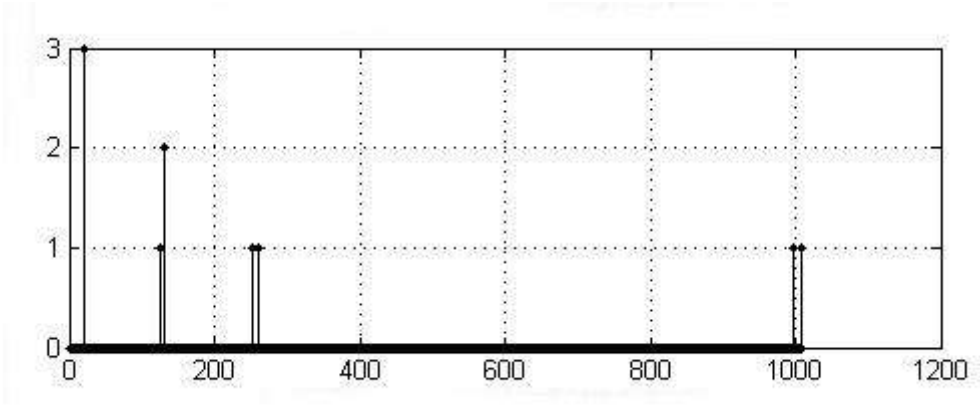

Рисунок 3.37 – k=25, N=5, **g**=21.

Графики рисунков 3.38-3.40 также приведены для разных значений длины информационной части кодовых блоков, но уже для другого значения порождающего полинома. Общее свойство преобладания правильного значения искомого полинома также сохраняется.

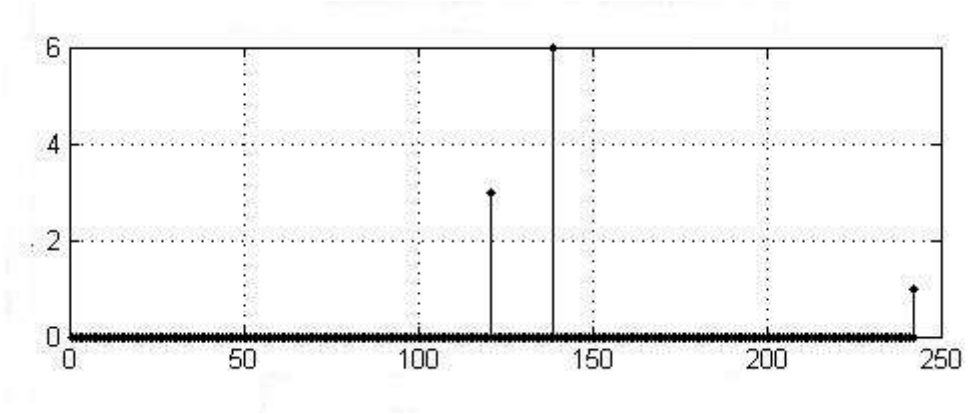

Рисунок 3.38 – k=5, N=5, **g**=121.

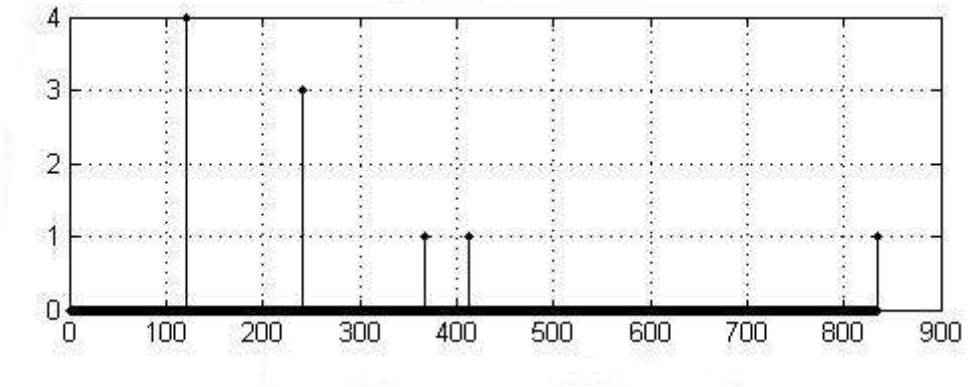

Рисунок 3.39 – k=10, N=5, **g**=121.

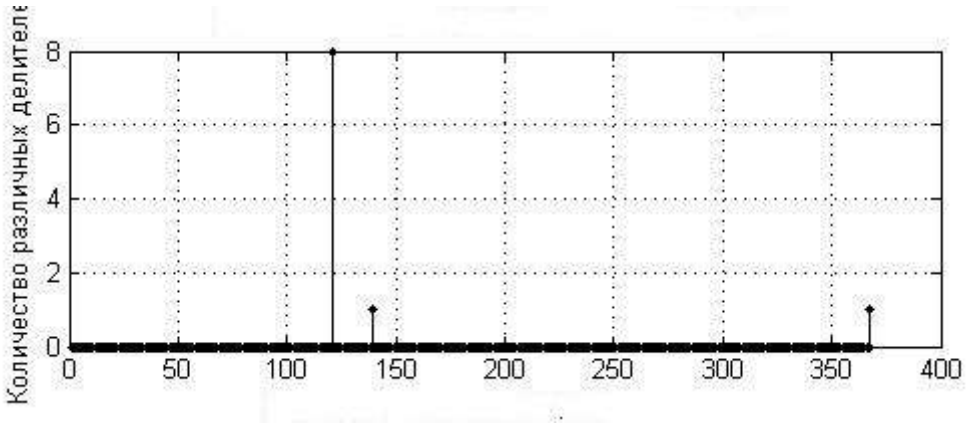

Рисунок 3.40 – k=20, N=5, **g**=121.

Графики на рисунках 3.41-3.44 показывают изменения свойств полученной совокупности результатов при увеличении набора исходных анализируемых кодовых слов при сохранении постоянными остальных параметров. При увеличении набора усиливается преобладание правильного результата диагностики перед другими результатами.

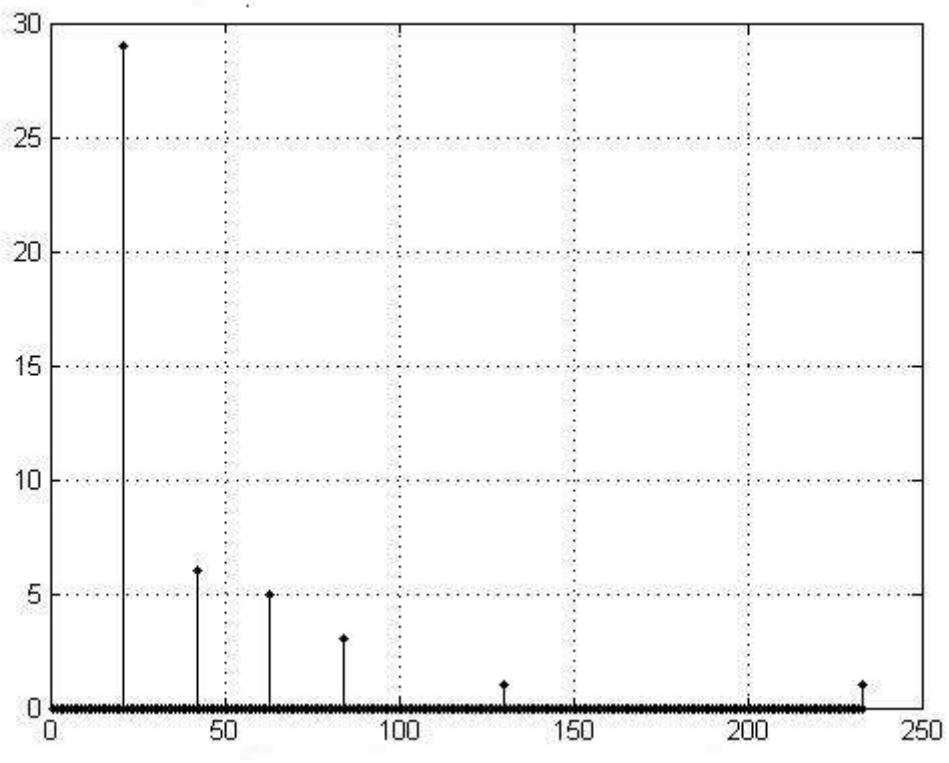

Рисунок 3.41 – k=5, N=10, **g**=21.

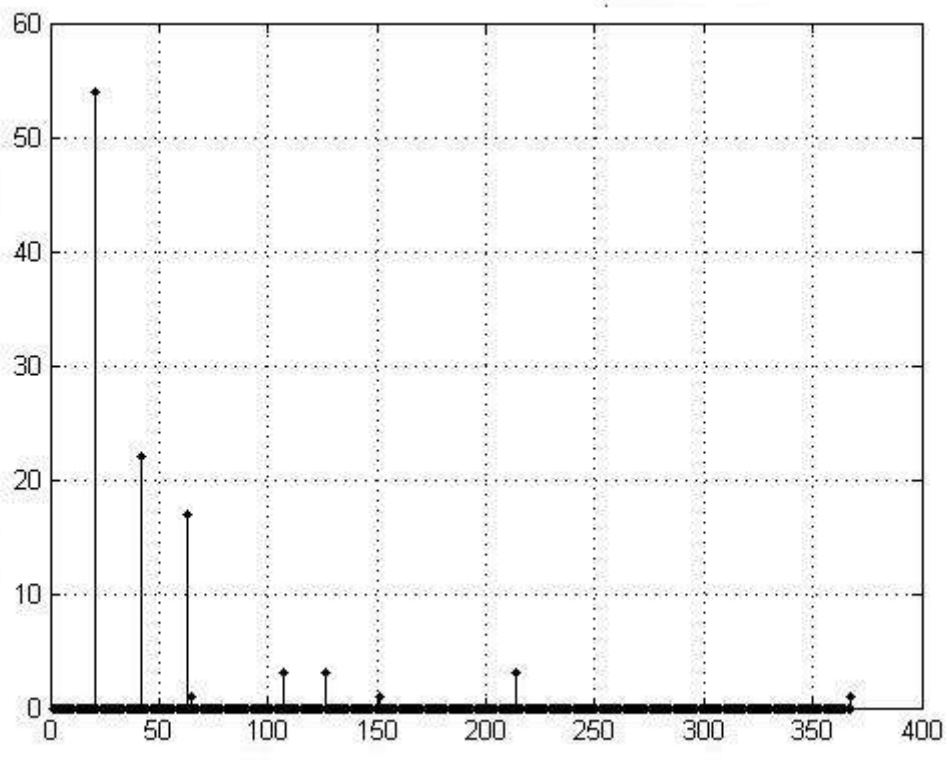

Рисунок 3.42 – k=5, N=15, **g**=21.

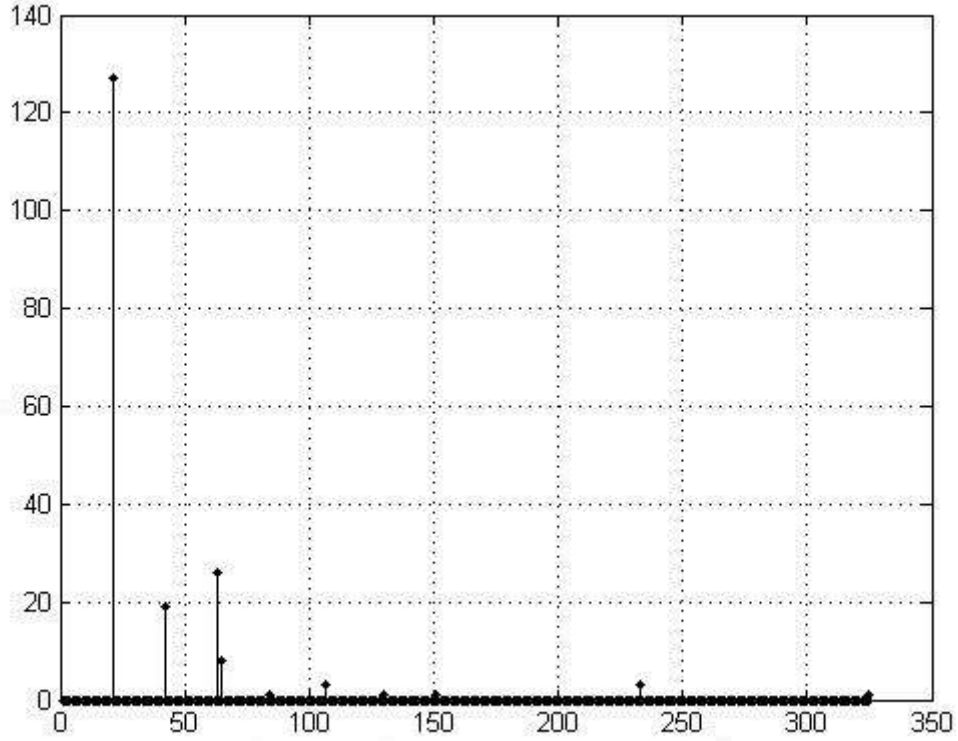

Рисунок 3.43 – k=5, N=20, **g**=21.

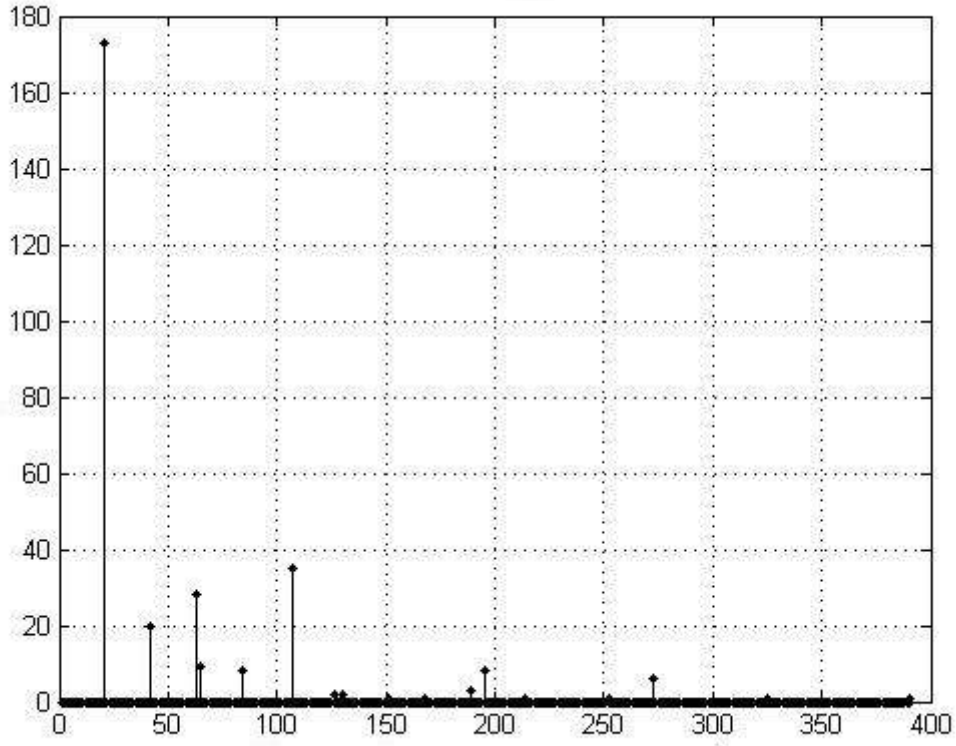

Рисунок 3.44 – k=5, N=25, **g**=21.

Графики 3.45-3.48 иллюстрируют свойства алгоритма при значительном увеличении длины проверочной части кодовых блоков и ее преобладании над длиной информационной части. И в этом случае анализ даже на относительно небольшом количестве кодовых блоков позволяет определить правильный вид порождающего полинома.

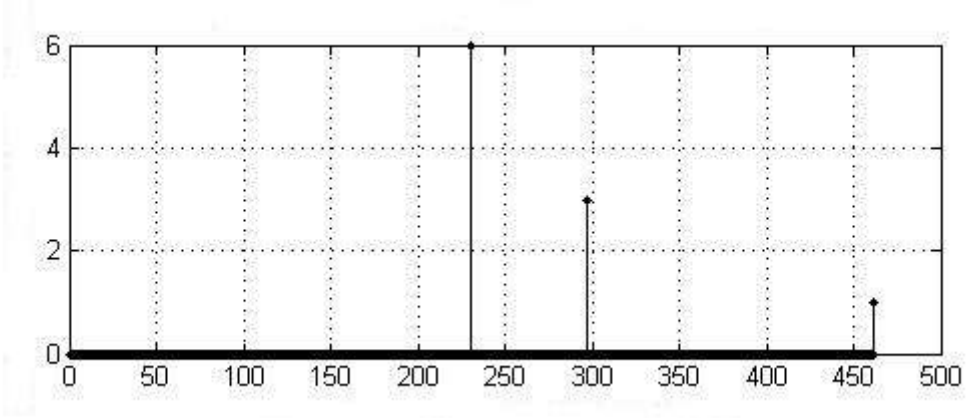

Рисунок 3.45 – k=5, N=5, **g**=231.

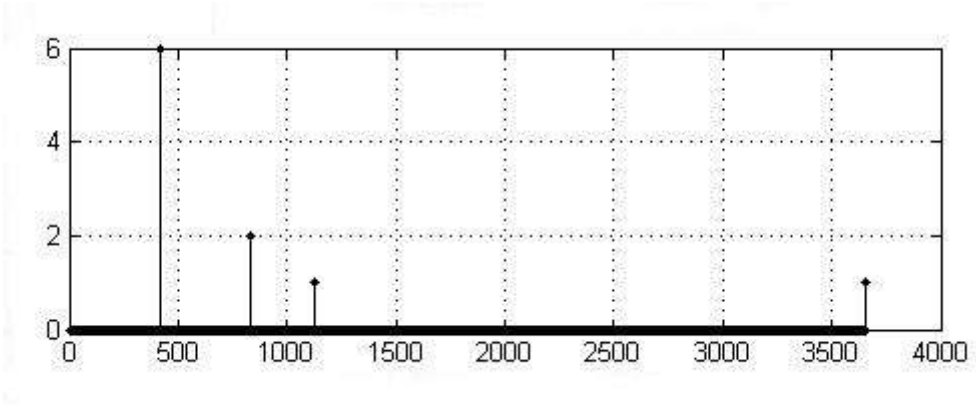

Рисунок 3.46 – k=5, N=5, **g**=417.

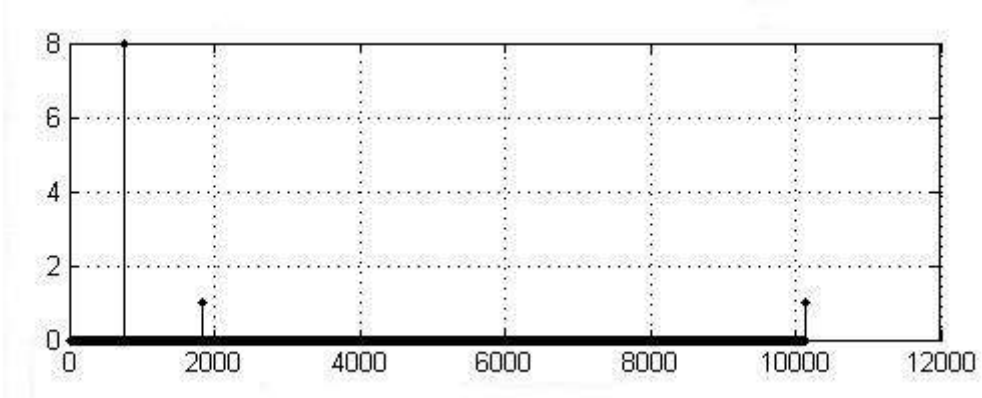

Рисунок 3.47 – k=5, N=5, **g**=751.

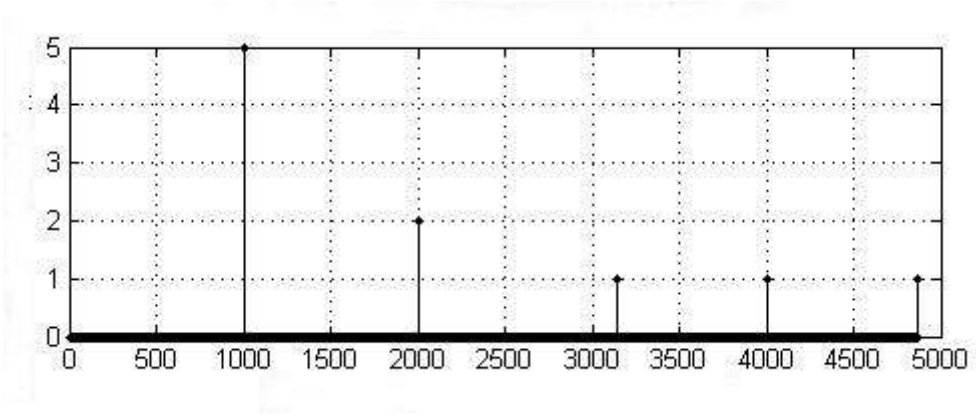

Рисунок 3.48 - k=5, N=5, **g**=1001.

Графики на рисунках 3.49-3.52 получены при одинаковых параметрах и иллюстрируют упоминавшийся результат, что полученные формы наборов выявленных общих полиномов, хотя и носят достаточно случайный характер, однако также обладают свойствами, необходимыми для диагностики.

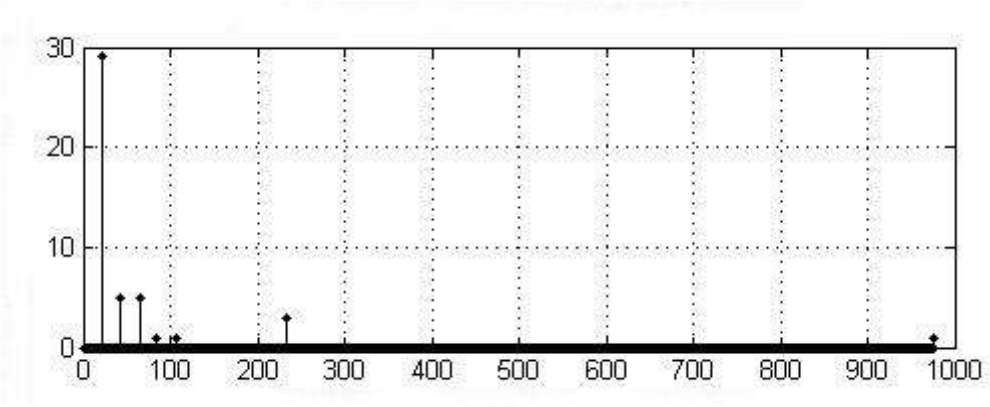

Рисунок 3.49 – k=5, N=5, **g**=21.

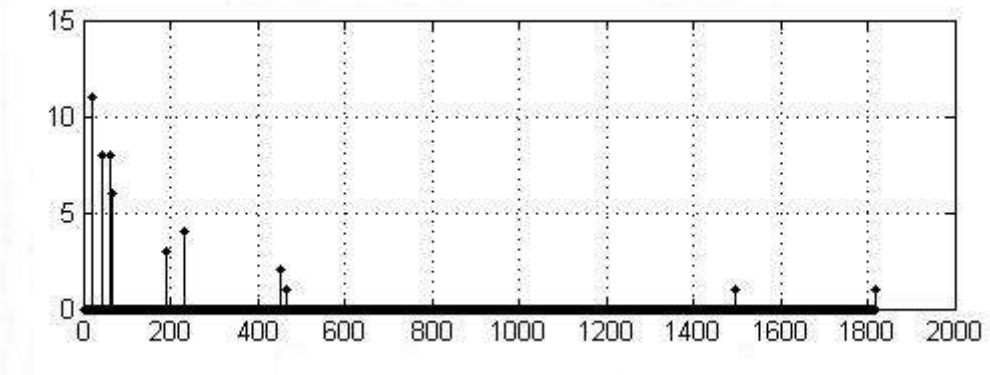

Рисунок 3.50 – k=5, N=5, **g**=21.

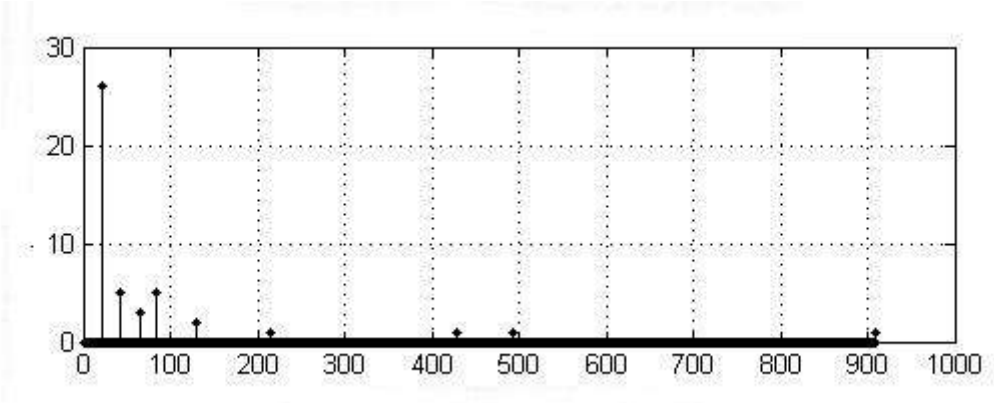

Рисунок 3.51 – k=5, N=5, **g**=21.

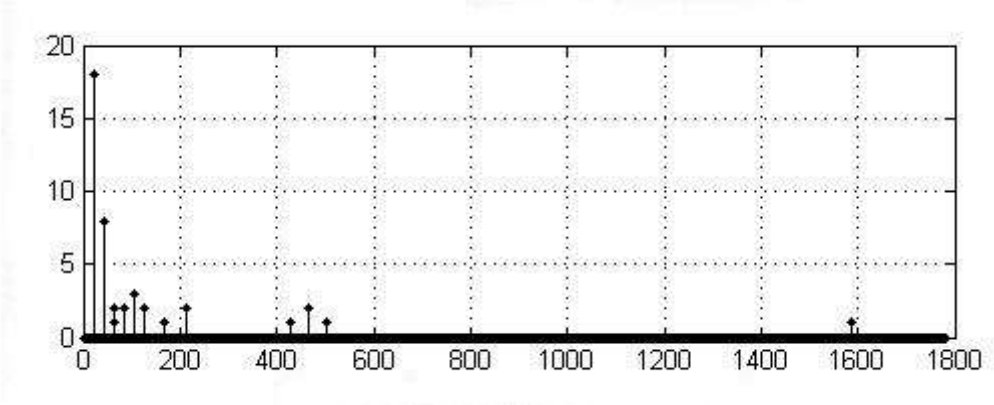

Рисунок 3.52 – k=5, N=5, **g**=21.

Оценим вероятность принятия ошибочного диагностического решения при использовании описываемого алгоритма. Правильное решение означает, что в результате проведения всей процедуры анализа величина полинома $\mathbf{M}_C$ стала равной 1. Любое другое значение этого полинома приведет к ошибочному решению.

Информационная часть может равновероятно принимать любое значение до 2^{k-1}. Будем считать, что в этом интервале чисел присутствует L простых полиномов и вероятность появления каждого полинома равна P_i, i=1÷L. Тогда вероятность, что в какой-либо паре анализируемых кодовых блоков будет одновременно присутствовать i-й простой полином, равна P_i^2. Таким образом, вероятность того, что при сравнении пары кодовых блоков будет определен

правильный вид порождающего полинома, т.е. вероятность того, что пар одинаковых простых полиномов в них не будет вообще, равна:

$$p_0 = \prod_{i=1}^{L}(1 - P_i^2). \tag{3.1}$$

Соответственно, вероятность того, что в паре анализируемых кодовых блоков будет присутствовать j-й простой полином, равна:

$$p_i = P_i^2 \prod_{\substack{j=1 \\ j\neq i}}^{L}(1 - P_j^2) = \frac{P_i^2}{1 - P_i^2}\prod_{j=1}^{L}(1 - P_j^2) = p_0 \frac{P_i^2}{1 - P_i^2}. \tag{3.2}$$

После завершения процедуры анализа среднее значение содержимого ячейки памяти, относящейся к правильному виду порождающего полинома, будет равно: $S_0=p_0N(N–1)/2$. Соответственно, среднее значение в ячейке, относящейся к i-му простому полиному, будет равно: $S_i=p_iN(N–1)/2$.

Ошибка диагностики будет возникать в случае, если $S_i \geq S_0$ при любых значениях индекса i, $i=1 \div L$. Рассмотрим вероятность ошибки диагностики P_{Di}, которая может возникнуть из-за i-го простого полинома. Она равна

$$P_{Di}\{S_i \geq S_0\} = \sum_{k=1}^{L}[C_k^L p_i^k (1 - p_i)^{L-k} \sum_{j=1}^{k} C_j^L p_0^j (1 - p_0)^{L-j}]. \tag{3.3}$$

Как уже упоминалось ранее, точная формула для определения вероятности P_i неизвестна, поэтому будем использовать грубую оценку, по которой величина этой вероятности обратно пропорционально значению простого полинома. Первыми простыми полиномами в используемых полях Галуа (по модулю 2) являются: 2, 3, 7, 11, и т.д. Соответственно, значения вероятности P_i будут равны: P_1=0,5; P_2=0,33; P_3=0,14; P_4=0,09,… Поскольку значения величины P_i убывают быстро и значения разностей $(1–Pi^2)$ очень быстро приближаются к единице, поэтому имеет смысл рассматривать вклад в величину p_0 только первых полиномов. В частности с учетом только первых четырех простых полиномов-множителей соответствующее произведение (1–0,25)(1–0,109)(1–0,02)(1–0,008)≈0,656. Также следует учесть, что значения вероятности появления простых полиномов убывают несколько быстрее, чем обратно-пропорциональная

зависимость, соответственно разности $(1–P_i^2)$ будут больше, а величина, определяемая формулой 3.1 – выше. Для грубой оценки «с запасом» примем значение p_0=0,65. В соответствии с этим: p_1≈0,22; p_2≈0,08; p_3≈0,013; p_4≈0,007.

Графики зависимостей вероятности P_{Di} ошибки диагностики от количества анализируемых кодовых слов N, рассчитанные на основе формулы (3.3), представлены на рисунке 3.53 для первых трех простых полиномов, i=1÷3. Нумерация графиков соответствует номерам простых полиномов.

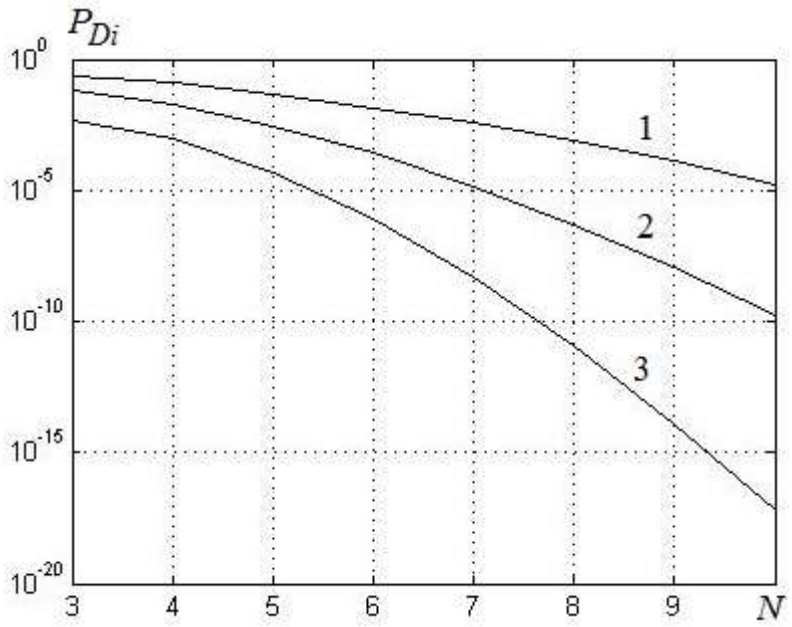

Рисунок 3.53.

Общая ошибка диагностики будет складываться из ошибок диагностики, обусловленной всеми простыми полиномами, однако с ростом номера полинома вероятность с ним связанной ошибки уменьшается в очень большой степени и на общую ошибку практически не влияет. Таким образом, путем не слишком большого увеличения набора анализируемых кодовых блоков можно снижать общую ошибку диагностики до требуемого уровня.

Предложенный и описанный алгоритм быстрой процедуры диагностики циклических кодов позволяет за приемлемое время анализа определять вид порождающего полинома в значительно более широком диапазоне параметров

кодера. В следующем параграфе рассмотрен алгоритм диагностики линейных блоковых кодов, получаемый с использованием порождающей матрицы.

3.4. Алгоритм диагностики линейных блоковых кодов

В краткой форме алгоритм диагностики линейных блоковых кодов, которые формируются с помощью порождающей матрицы **G**, описан в параграфе 3.1. Также указано, какие дополнительные задачи необходимо решить для ее определения. В данном параграфе алгоритм описан подробно и исследованы его свойства с применением программного комплекса [100].

Исходно порождающая матрица состоит из матричных блоков **E** и **P**, $\mathbf{G} = \|\mathbf{E}\vdots\mathbf{P}\|$, где **E** – единичная матрица размера $k\times k$, у которой элементы главной диагонали равны единице, все остальные элементы равны нулю; **P** – проверочная матрица размера $k\times b=k\times(n-k)$. Задача диагностики состоит в определении элементов матрицы **P**.

Структурная схема диагностического алгоритма приведена на рисунке 3.54. Операции алгоритма состоят в следующем. Первоначально из потока принимаемых кодовых блоков выбирается q кодовых блоков и из них составляется матрица **Y** размером $q\times n$. Для дальнейшей обработки из нее выделяется квадратная матрица **Q** размером $q\times q$, расположенная, как блок, либо в начале матрицы **Y** (если информационные символы предшествуют проверочным символам), либо в ее конце (если информационные символы следуют за проверочными символами).

Если известно заранее, сколько символов находится в информационной части блока, то сразу устанавливается $q=k$. Если же это количество неизвестно, то производится его поиск и определение. Кроме этого, необходимо, чтобы матрица **Q** не была вырожденной. Однако матрица, сформированная из взятых наугад q

кодовых блоков, вполне может оказаться вырожденной, поэтому алгоритмом также производится перебор блоков, пока не будет выполняться условие невырожденности матрицы **Q**.

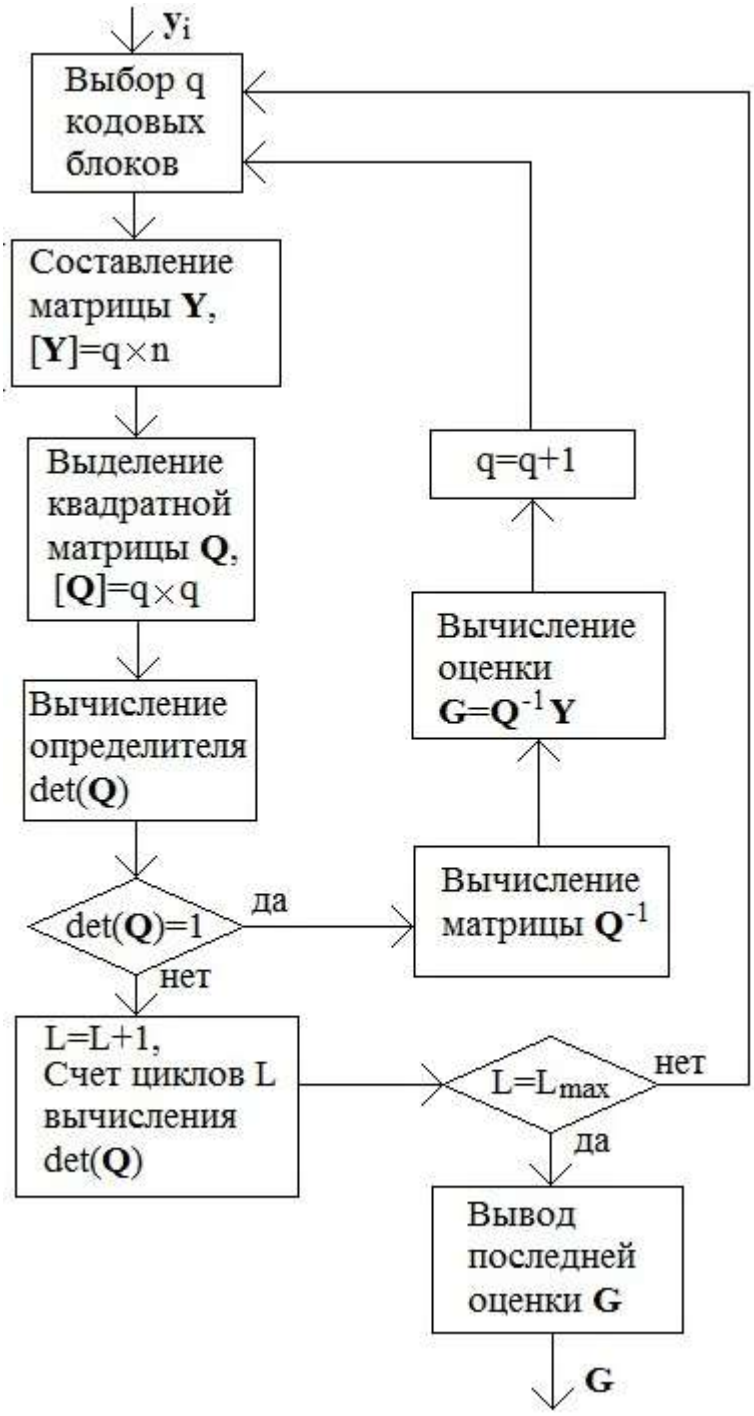

Рисунок 3.54.

В соответствии с этими условиями алгоритм совершает следующие операции. Первоначально значение q устанавливается минимальным, равным 2. Полученная матрица **Q** проверяется на невырожденность путем вычисления определителя от нее. (Следует заметить, что все матричные вычисления производятся не по правилам линейной алгебры, а по правилам оперирования в полях Галуа.) Если оказалось, что определитель не равен нулю, то производится вычисление обратной матрицы $\mathbf{Q}^{-1}$. Далее вычисляется и запоминается оценка порождающего полинома $\mathbf{G}=\mathbf{Q}^{-1}\mathbf{Y}$, полученная для данного значения q. После этого величина значения q увеличивается на единицу, и вновь повторяются все операции, начиная с выбора q кодовых блоков и составления матрицы **Y** с новым размером.

Если же значение вычисленного определителя окажется равным нулю, то выбираются q новых блоков с тем же значением q и цикл всех операций повторяется. При этом подсчитывается количество циклов L с одинаковым значением q, при которых определитель имел нулевое значение. Если оно превысит заранее выбранное значение L_{max}, то принимается решение, что последняя оценка матрицы **G**, полученная при наибольшем из рассмотренных значений q, соответствует истинному виду используемой при кодировании порождающей матрицы. Это значение выводится, как результат диагностики, для последующего использования. Если же при каком-то цикле $L<L_{max}$ определитель станет равным единице, то, как и описывалось, вычисляется соответствующая ему оценка матрицы **G**, значение q увеличивается на единицу, и т.д.

Таким образом, алгоритм производит оценку невырожденности матрицы **Q**, начиная с ее минимального размера. При этом используется свойство, что пока q не превысит истинное неизвестное заранее значение k, формируемые матрицы **Q** будут являться либо вырожденными, либо невырожденными. А если q превысит величину k, то матрицы **Q** будут вырожденными *всегда*, при этом «индикатором» вырожденности служит нулевое значение определителя. При постепенном увеличении значения q последним значением, при котором определитель может

принимать единичные значения, будет $q=k$. Именно при этом значении вычисляется обратная матрица $\mathbf{Q}^{-1}$, и оценка **G** считается правильной.

Естественно, поскольку значения элементов матрицы случайны, то даже при $q \le k$ эта матрица может так же случайно оказаться вырожденной. Поэтому проверка вырожденности, если определитель принимает нулевые значения, производится L_{max} раз. Величина L_{max} выбирается заранее и зависит от допустимых требований на ошибку диагностики. Этот вопрос будет подробно рассмотрен далее.

Работа описываемого алгоритма исследовалась с помощью компьютерного моделирования. Некоторые результаты представлены на рисунках 3.55-3.60. Рисунки представляют собой визуализацию оценок матрицы **G** при возрастающих значениях q. На рисунке 3.4.2 представлена матрица **G**, используемая для кодирования, параметры которой равны: k=6, b=3.

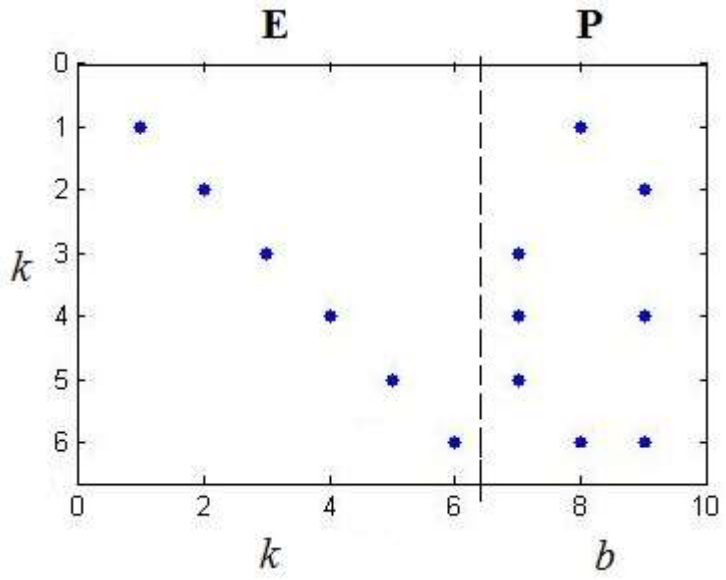

Рисунок 3.55.

В процессе работы алгоритма перебираются возрастающие значения q, и для каждого значения при единичной величине определителя вычисляется обратная матрица $\mathbf{Q}^{-1}$ и оценка матрицы **G**. Оценки матрицы **G** приведены на рисунках 3.56-3.60 в графической форме (единичные элементы матриц

обозначены точками, по вертикали отложены номера строк, по горизонтали отложены номера столбцов). Значение L_{max} в экспериментах составляло L_{max}=10. После того, как значение q начинало превышать k, во всех экспериментах получался det(**Q**)=0 и формирование оценки **G** становилось невозможным.

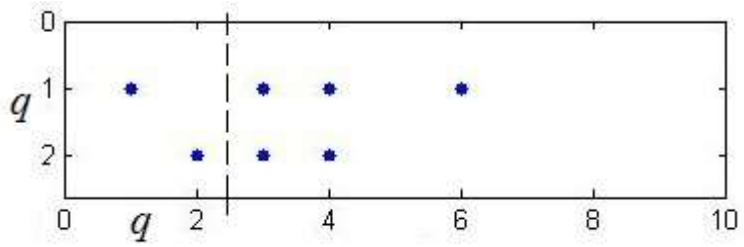

Рисунок 3.56 – q=2.

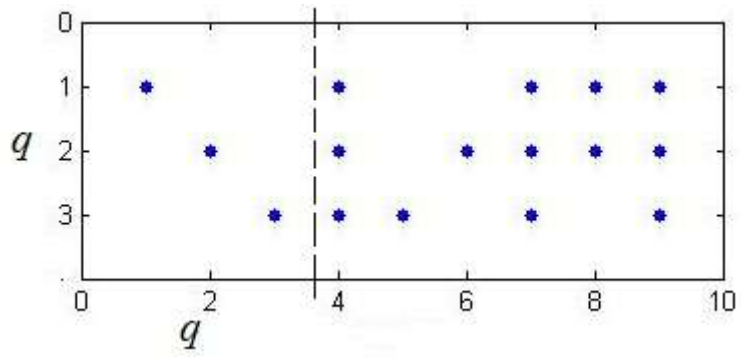

Рисунок 3.57 – q=3.

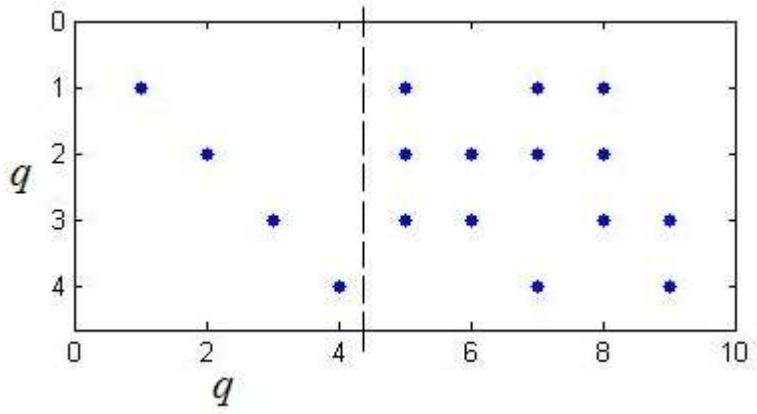

Рисунок 3.58 – q=4.

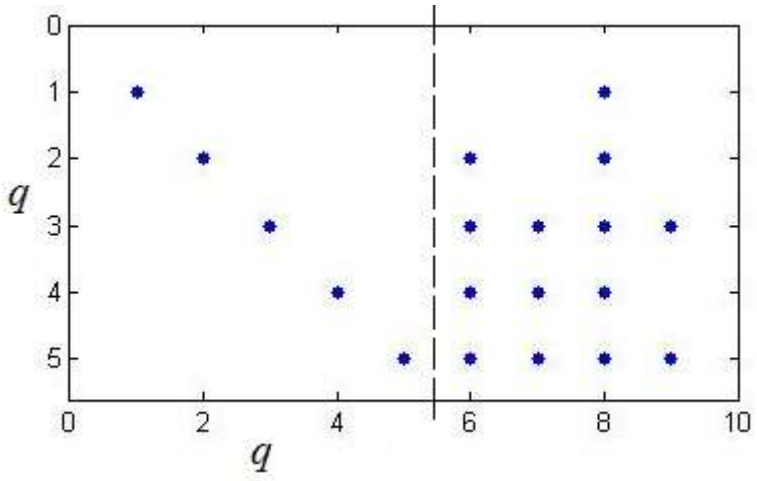

Рисунок 3.59 – q=5.

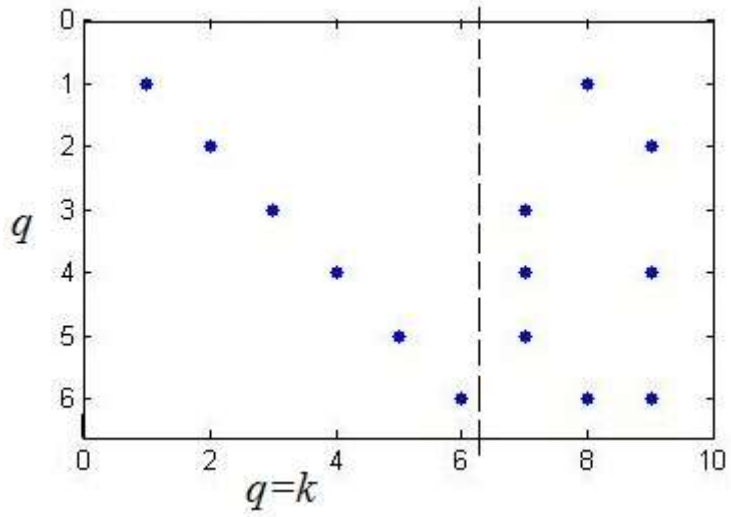

Рисунок 3.60 – q=6.

Во всех проведенных экспериментах, один из примеров которых представлен на рисунках, наблюдалась одинаковая картина. Она заключалась в том, что при $q>k$ обратная матрица $\mathbf{Q}^{-1}$ вычислялась достаточно легко, однако модель порождающей матрицы **G**, естественно, не соответствовала использованной при кодировании. По достижении $q=k$ формировалась «правильная» модель порождающей матрицы. После превышения величиной q этого значения анализируемые вектора-строки матрицы **Q** всегда оказывались линейно зависимыми, определитель такой матрицы был равен нулю, что делало невозможным продолжение операций алгоритма и позволяло подать на выход сформированное правильное решение диагностической задачи.

Далее оценим факторы, влияющие на неудачную попытку диагностики и на появление ошибки диагностики из-за воздействия помех. Первоначально рассмотрим вероятность неудачи при диагностике по причине того, что при выбранном предельном количестве циклов L_{max} будут появляться только вырожденные матрицы, которые не дадут получить оценку **G.**

Как уже упоминалось, при $q\le k$ матрицы **Q**, формируемые из кодовых блоков, могут случайно оказаться как вырожденными, так и не вырожденными. Вероятность появления вырожденных матриц определяет, какое значение L_{max} необходимо выбирать, чтобы обеспечить требуемый уровень вероятности диагностической ошибки. Эту вероятность появления вырожденной матрицы **Q** будем определять из следующих соображений. Пусть ее размеры составляют $[\mathbf{Q}]=q\times q$. Определим, сколько вариантов этой матрицы будет вырожденными, и какую долю они составят из всех возможных вариантов.

Будем рассматривать матрицу **Q** построчно, т.е. постепенно добавляя в рассмотрение к первой строке вторую и определяя, какие ее варианты приведут к вырожденности, далее добавляя к первым двум строкам третью строку и определяя, сколько вариантов третьей строки приведут к вырожденности, и т.д.

Первая строка может быть выбрана произвольно. Количество остальных элементов матрицы равно $q(q-1)$. Поскольку появление значения единицы или

нуля у каждого элемента можно считать равновероятным, то всего возможно $2^{q(q-1)}$ различных вариантов матрицы, причем появление каждого из них случайно и равновероятно.

Чтобы матрица стала вырожденной, вторая строка должна быть в точности равна первой, т.е. для условия вырожденности возможен лишь один вариант второй строки. Значения же элементов остальных строк могут принимать произвольные значения, при любых их вариантах матрица останется вырожденной. Здесь возможно $2^{q(q-2)}$ вариантов матрицы.

Рассмотрим третью строку при условии, что первые две строки различаются между собой (случай их одинаковости уже рассмотрен). Матрица станет вырожденной при условии, что третья строка равна либо первой строке, либо второй строке, либо их сумме (по модулю 2), т.е. возможны три ее варианта. Элементы остальных же q-3 строк могут независимо принимать одно из двух значений, значит всего будет $3q2^{q(q-3)}$ вариантов матрицы, приводящих к ее вырожденности.

Проводя аналогичные рассуждения для четвертой строки, пятой строки и т.д., можно получить количество вариантов, приводящих к вырожденности матрицы, соответственно, $7q2^{q(q-4)}$, $15q2^{q(q-5)}$, и т.д. Для i-й строки аналогичные рассуждения покажут, что количество вариантов матрицы, приводящих к ее вырожденности, равно $(2^{i-1}-1)2^{q(q-i)}$. Таким образом, для всей матрицы количество S вариантов ее вида, когда она является вырожденной, равно:

$$S = \sum_{i=2}^{q}(2^{i-1}-1)2^{q(q-i)} = \sum_{j=0}^{q-2}(2^{j+1}-1)2^{q(q-j-2)}.$$

Преобразуем данное выражение:

$$S = 2^{q(q-2)}[\sum_{j=0}^{q-2}2^{(1-q)j+1} - \sum_{j=0}^{q-2}2^{-qj}] = 2^{q(-2)}\{\sum_{j=0}^{q-2}2[2^{(1-q)}]^j - \sum_{j=0}^{q-2}[2^{-q}]^j\} =$$

$$= 2^{q(q-2)}[2\frac{1-2^{-(1-q)^2}}{1-2^{1-q}} - \frac{1-2^{-q(q-1)}}{1-2^{-q}}].$$

При данной первой строке будет количество всех возможных вариантов матрицы, равное $2^{k(k-1)}$. Таким образом, вероятность p_q, что построенная матрица размера $q \times q$, окажется вырожденной, равна:

$$p_q = 2^{-q}[2\frac{1-2^{-(1-q)^2}}{1-2^{1-q}} - \frac{1-2^{-q(q-1)}}{1-2^{-q}}].$$

Общая вероятность P_V того, что при матрице размера $k \times k$ и предельном количестве L_{max} циклов проверки диагностическое решение так и не будет найдено, равна:

$$P_V = 1 - \sum_{q=2}^{k}(1 - p_q^{L_{\max}}).$$

На рисунке 3.61 приведена зависимость p_q от q (прерывистой линией) и зависимости P_V от L_{max} при различных k (сплошными линиями). По горизонтальной оси отложены значения q и L_{max}, по вертикальной оси в логарифмическом масштабе отложены значения p_q и P_V. График 1 соответствует значению k=3; график 2 соответствует значению k=5; график 3 соответствует значению k=10.

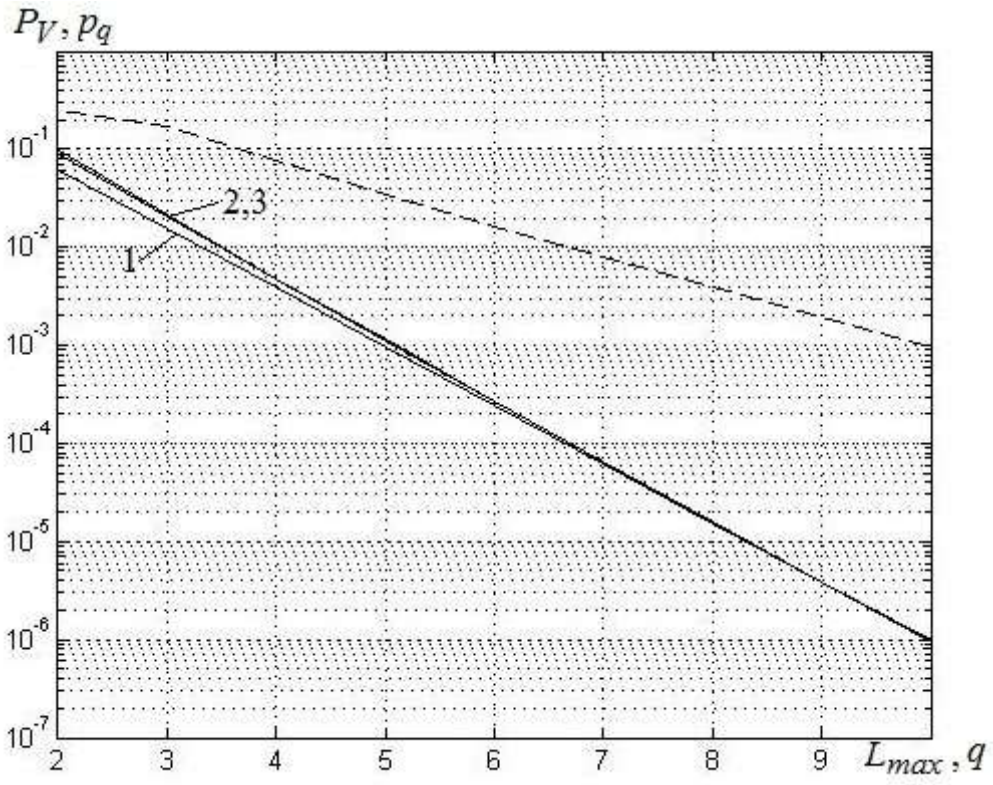

Рисунок 3.61.

С увеличением k, начиная со значения k=5, графики практически сливаются. При увеличении L_{max} величина вероятности P_V уменьшается практически экспоненциально, поэтому несложно за счет относительно небольшого такого увеличения достичь требуемого малого уровня P_V .

Рассмотрим воздействие помех на описываемый алгоритм диагностики. Возникающие из-за этого символьные ошибки могут привести к различным негативным результатам. Один из них состоит в том, что матрицы, ранее бывшие невырожденными, могут стать вырожденными. Определим вероятность такого события. Если вероятность символьной ошибки равна P_B, то вероятность того, что в матрице размером $q \times q$ появится хотя бы одна ошибка, будет равна $1-(1-P_B)^{q^2}$, а вероятность того, что такая ошибка невырожденную матрицу сделает вырожденной, равна, соответственно: $P_2 = p_q[1-(1-P_B)^{q^2}]$. Это в общем числе испытаний уменьшит долю невырожденных матриц и увеличит долю вырожденных до $(P_V+P_2)L_{max}$. Однако увеличением числа циклов L_{max} в $(1-P_V)/(1-$

P_V–P_2) раз при необходимости обеспечивается такой же уровень вероятности диагностической ошибки. Технически это сделать несложно, поэтому данный вариант негативного воздействия помех может быть без особенных проблем скомпенсирован.

Другой негативный результат может проявиться в том, что «правильная» невырожденная матрица превратится в «неправильную», но тоже невырожденную. Это опасно в последнем цикле, когда величина q достигнет значения k. Вероятность такого события равна: $P_3 = (1-p_k)\,p_k[1-(1-P_B)^{k^2}]$. Определитель станет равным единице, и если в следующих циклах при $q>k$ он, будет равным нулю, значит «неправильная» невырожденная матрица будет принята за «правильную». От нее будет вычислена обратная матрица, которая после умножения на **Y** даст неверный вид порождающей матрицы **G.**

По этой причине для работы в условиях значительного уровня шумов и помех алгоритм должен быть модифицирован. Модификация заключается в следующем. Решения о различных вариантах действий предпринимаются не по одному факту равенства единице вычисленного значения определителя при конкретной величине q, а по нескольким единичным его значениям (например, по l_{max} единичным значениям), полученным при одной и той же величине q. В каждом из таких случаев далее попутно вычисляется оценка порождающей матрицы **G**. Вероятность ошибки при этом равна $P_3^{l_{\max}}$ и быстро падает с ростом l_{max}. Однако и общее время анализа увеличивается в l_{max} раз. К схожим последствиям приведет ситуация, когда при $q=k+1$ вырожденная матрица **Q** из-за символьной ошибки примет вид невырожденной, которая дальше будет обрабатываться, как правильная и для их устранения также решение предпринимается по l_{max} одинаковым результатам.

Когда будет констатировано, что $q=k$, то сравниваются между собой полученные оценки матрицы **G**. Далее используется мажоритарный принцип и выбирается в качестве результата диагностики тот вид матрицы, который получится наибольшее число раз.

Таким образом, предложенный алгоритм позволяет эффективно диагностировать вид используемых линейных блоковых кодов, полученных с помощью порождающей матрицы. При работе в условиях значительного уровня помех и шумов алгоритм может быть модифицирован, и для сохранения малого уровня вероятности диагностической ошибки достаточно увеличения общего времени анализа.

3.5. Выводы

1. При осуществлении диагностики систематических и несистематических циклических кодов может быть использован алгоритм, использующий сравнение набора простых полиномов, на которые разлагаются кодовые блоки. Однако, несмотря на достаточно универсальный общий подход, предлагаемый алгоритмом, длительность вычислений быстро растет с увеличением размера кодовых блоков.

2. «Быстрый» диагностический алгоритм, основанный на попарном сравнении кодовых блоков позволяет существенно сократить время анализа, но более чувствителен к помехам и шумам.

3. При диагностике линейных блоковых кодов, получаемых с помощью порождающей матрицы, как более общего случая блоковых кодов, достаточно эффективен алгоритм, основанный на построении невырожденных матриц от фрагментов набора принимаемых кодовых блоков. В условиях работы при значительном уровне помех и шумов предложена модификация алгоритма, позволяющая сохранить его качественные показатели за счет приемлемого увеличения продолжительности анализа.

4. Разработка алгоритмов диагностики модифицированных кодов

В данной главе рассмотрены возможности диагностики модифицированных кодов, получаемых из исходных простых кодов с помощью уменьшения количества информационных символов, увеличения числа проверочных символов, либо отбрасывания части символов после предварительного кодирования. В качестве основы использованы принципы и алгоритмы диагностики, описанные во второй и третьей главах, а также решены возникающие дополнительные диагностические задачи. Для перфорированных сверточных кодов предложены и описаны новые алгоритмы для оценки дополнительных новых параметров, используемых при перфорации кодов. Приведены результаты по исследованию эффективности алгоритмов диагностики модифицированных кодов.

4.1. Принципиальные основы диагностики модифицированных кодов

Как было описано в первой главе, модификация кодов заключается в изменении кодовой скорости, получаемой путем использования простых методов кодирования, и сводящегося к изменению количественного соотношения между исходными информационными символами и формируемыми с их помощью проверочными символами. При этом для блоковых кодов это осуществляется с помощью отбрасывания определенного числа s позиций в информационной части кодового слова (укорочение кодов), либо добавлением в кодовое слово проверочных символов при сохранении в нем количества информационных символов (расширение кодов).

Модификация сверточных кодов заключается в периодическом удалении символов ранее сформированной кодовой последовательности путем выкалывания некоторых символов (перфорации). Поскольку принципы модификации для блоковых и сверточных кодов существенно различаются, то

также различаются и принципы работы соответствующих диагностических алгоритмов.

Первоначально рассмотрим укорочение и расширение блоковых кодов. Если для формирования исходного блокового кода используется порождающая матрица размером $k \times n$, то после модификации размеры новой порождающей матрицы становятся равными $(k–s)\times(n–s)$ в случае ее укорочения. Диагональная часть матрицы с единичными элементами, соответствующая информационной части кодовых блоков, приобретает размер $(k–s)\times(n–s)$. Наборы элемент строк матрицы, определяющие вид проверочной части блоков, однозначно определяются номером строк, т.е. номером символов информационной части. Поэтому для диагностики может без существенных изменений быть использован алгоритм диагностики, предложенный для немодифицированных блоковых кодов, формируемых с помощью порождающих матриц и описанный в разделах 3.1 и 3.4.

Если размер $k–s$ сокращенной информационной части кодовых блоков известен заранее, то среди принятых блоков подбираются $k–s$ блоков $y_1 \div y_{k–s}$, информационные части $m_1,\ldots,m_{k–s}$ которых между собой все являются взаимно независимыми. Из них, как из строк, составляется матрица $\mathbf{Y} = \|\mathbf{M}\vdots\mathbf{S}\|$, в которой часть $\mathbf{M}$ имеет размер $(k–s)\times(k–s)$, часть $\mathbf{S}$ имеет размер $(k–s)\times b$. Строки матрицы $\mathbf{M}$ – это информационные последовательности $m_1,\ldots,m_{k–s}$ выбранных кодовых блоков $y_1 \div y_{ks}$, соответственно, строки матрицы $\mathbf{S}$ – это проверочные части этих кодовых блоков. Далее вычисляется обратная ей матрица $\mathbf{M}^{-1}$. Матрица $\mathbf{Y}$ умножается на матрицу $\mathbf{M}^{-1}$: $\mathbf{M}^{-1}\mathbf{Y} = \mathbf{M}^{-1}\|\mathbf{M}\vdots\mathbf{S}\| = \|\mathbf{M}^{-1}\mathbf{M}\vdots\mathbf{M}^{-1}\mathbf{S}\| = \|\mathbf{E}\vdots\mathbf{M}^{-1}\mathbf{S}\|$.

Таким образом, получается порождающая матрица $\mathbf{M}^{-1}\mathbf{Y}=\mathbf{G}$, в которой часть $\mathbf{M}^{-1}\mathbf{S}=\mathbf{P}$, т.е. фактически задача диагностики оказывается решенной. Естественно, требуется из принятых сигналов выбрать такой набор кодовых блоков $y_1 \div y_{k–s}$, у которых информационные части представляют собой линейно-независимые векторы.

Однако, так же, как и при отсутствии модификации кода, могут оказаться заранее неизвестными размер $k–s$ укороченной информационной части и размер b проверочной части блоков. В этом случае производится поиск последовательным увеличением количества q выбранных кодовых блоков, из первых q символов которых составляется квадратная матрица и проверяется на вырожденность. Поскольку, как и в алгоритме в параграфах 3.1, 3.4, устойчивая вырожденность будет наблюдаться при условии $q>k–s$, то именно такое значение q принимается за $k–s$.

В случае же использования расширения кодов алгоритм диагностики еще более похож на алгоритм диагностики соответствующего немодифицированного блокового кода. Действительно, устойчивая вырожденность формируемой матрицы также имеет место при $q>k$, что позволяет определить величину k и вид части порождающей матрицы, формирующей проверочные символы кодовых слов. Отличия наблюдаются только в том, что количество элементов строк в этой части больше, чем случае отсутствия модификации, что никак не сказывается на работе диагностического алгоритма.

Если модифицируются циклические блоковые алгоритмы, то отличия появляются в укорочении информационной части $\mathbf{m}_i(X)$ кодовых блоков. Однако, фактически это проявляется лишь в уменьшении числа простых полиномов, из которых состоит информационная часть каждого блока. Поэтому в этом случае могут быть применены диагностические алгоритмы, описанные в параграфах 3.2 и 3.3. Более того, их работа облегчается, поскольку уменьшается количество полиномов информационной части, мешающих диагностике.

Таким образом, если модификации подвергаются блоковые коды, то для их диагностики могут быть использованы практически те же алгоритмы, что и для немодифицированных кодов. Однако при использовании перфорации сверточных кодов задача диагностики существенно усложняется, что требует использования значительно отличающихся алгоритмов. Это обусловлено следующими причинами.

В исходном сверточном коде каждый символ формируется на основе K информационных символов, т.е. связан определенными совместными взаимными корреляционными зависимостями с K/R предыдущими кодовыми символами. При этом подобные зависимости заранее неизвестны и носят статистический характер, известно лишь, что они постоянны при неизменной структуре кодера. Для их выявления при диагностике может потребоваться использование достаточного объема выборки принятой кодовой последовательности. А перфорация удаляет зачастую близко расположенные символы, при этом взаимные корреляционные зависимости существенно ослабляются. Это может потребовать для их выявления и последующей обработки значительного увеличения необходимого объема выборок и усложнения алгоритмов обработки.

Кроме этого, возрастает количество параметров кодирования, которые необходимо диагностировать. В частности остается прежняя задача определения величины кодового ограничения и вида порождающих полиномов. Появившаяся новая задача заключается в определении параметров перфорации.

Как описывалось в параграфе 1.3, процесс перфорации задается матрицей или маской перфорации. Она указывает, какие из сформированных кодовых символов должны быть удалены. Перфорация производится периодически через определенное количество символов T_P. Период перфорации должен быть кратным $n=1/R$, где R – кодовая скорость исходного сверточного кода. Маска перфорации указывает, какие из кодовых символов внутри периода T_P должны быть удалены (пусть их будет s символов). Таким образом, новые дополнительные параметры, которые должны быть определены в результате диагностики, заключаются в оценке величины T_P и структуры маски перфорации.

Эта комплексная диагностическая задача может быть решена с помощью двух предлагаемых алгоритмов. Первый из них определяет период перфорации и величину кодового ограничения примененного кодера. Второй алгоритм на этой основе определяет вид используемых порождающих полиномов и структуру маски перфорации. В следующих параграфах приведено описание этих алгоритмов и исследование их свойств.

4.2. Алгоритм определения периода перфорации и величины кодового ограничения

Величина кодового ограничения может быть определена одновременно с периодом перфорации, поэтому диагностика обеих величин может быть осуществлена в одном алгоритме.

Структуру перфорированных сверточных кодов принято записывать в форме своего рода таблички, где цифрами обозначена запись применяемых порождающих полиномов в восьмеричном коде, а буквой X обозначено расположение удаляемых символов. Например, запись $\begin{pmatrix} 7,7 \\ 5, X \end{pmatrix}$ означает, что использован код с порождающими полиномами, равными 7_8 и 5_8. При этом одна пара кодовых символов передается без изменений, а из второй пары кодовых символов удаляется второй символ, сформированный с помощью полинома 5_8. Таким образом, период перфорации равен четырем, используется два сумматора по модулю 2. Кодовая скорость до перфорации составляла ½, а после перфорации она стала равной 2/3. Соответственно, например, запись $\begin{pmatrix} 171, X, 171 \\ 133, 133, X \end{pmatrix}$ означает, что период перфорации равен шести и использован исходный сверточный код (171,133). Первые два символа кода передаются без изменений, из второй пары символов удаляется первый символ, а из третьей пары символов удаляется второй символ. До перфорации кодовая скорость составляла 1/2, а после перфорации она стала равной 3/4 .

Однако, правило осуществления перфорации может быть записано и в «одномерном» виде в форме строки символов расположенных последовательно длиной в период перфорации. Для приведенных примеров это будет выглядеть в виде: (7, 5, 7, *X*) и (171, 133, *X*, 133, 171, *X*). Поскольку символы, обозначенные буквой *X*, по каналу не передаются, то после перфорации период повторения станет равным, соответственно, 3 и 4, а повторяющиеся фрагменты будут иметь

вид (7, 5, 7) и (171, 133, 133, 171). Если при формировании исходного сверточного кода использовалось три и более сумматора по модулю 2, то соответствующая табличка будет иметь три и более строки.

Для удобства изложения далее будем называть запись правила перфорации в форме таблички матрицей перфорации, а запись правила в форме одной строки – маской перфорации.

Основу описываемого алгоритма диагностики перфорированных кодов составляет алгоритм определения кодового ограничения для неперфорированных кодов, описанный в параграфе 2.4, но здесь он видоизменен определенным образом. Рассмотрим ситуацию, когда кодовая скорость сверточного кода равна 1/2. В исходном виде в параграфе 2.4 алгоритм разделяет принимаемую общую кодовую последовательность символов $\mathbf{u}_0$ на две частные кодовые последовательности $\mathbf{u}_1(X)$ и $\mathbf{u}_2(X)$. Из них формируются пары групп символов одинаковой длины $L=L_1=L_2$. При этом для сокращения необходимой общей длины выборки рекомендуется, чтобы каждая новая пара групп формировалась последовательным сдвигом последовательностей $\mathbf{u}_1$ и $\mathbf{u}_2$ на один символ, хотя возможен выбор каждый раз полностью новых групп из этих последовательностей.

Результат анализа каждой пары заносится в первоначально обнуленную корреляционную матрицу. Если величина соответствующего элемента была равна нулю, то туда в данном цикле записывается единица, если же элемент уже был равен единице, то никаких действий не производится. Номер строки и столбца, где расположен рассматриваемый элемент, определяется по значениям групп символов из первой и второй частных последовательностей. Возможность оценки значения кодового ограничения с помощью подобного алгоритма основывалась на том, что обе частные кодовые последовательности, пусть с помощью разных порождающих полиномов, но образовывались из одной и той же исходной информационной последовательности. При этом вид полиномов не менялся, т.е. на протяжении сеанса диагностики был один и тот же для всех пар групп. Это приводило к тому, что при определенном сочетании L двоичных символов в

одной группе, сочетания двоичных символов из другой пары могли принимать не любые 2^L возможных вариантов, а значительно меньше. Величина этого «дефицита» определялась соотношением значения кодового ограничения K и значения L. В результате даже при достаточно длинной анализируемой выборке матрица заполнялась не полностью и в ней оставались элементы с нулевым содержанием, причем рост объема выборки сохранял такие элементы нулевыми. По соотношению их количества и количества ненулевых элементов определялась величина искомого кодового ограничения.

Предлагаемый алгоритм определения периода перфорации осуществляет схожую обработку кодовой последовательности, но реализует поисковый принцип нахождения решения. Для этого принятая перфорированная кодовая последовательность предварительно видоизменяется. Видоизменение заключается в том, что в нее периодически (с некоторым периодом T_Q) вставляются идущие подряд дополнительные s_Q символов. Значения вставляемых символов все берутся равными предыдущему символу перед вставкой. Величина s_Q – «предположительное» количество удаляемых символов в периоде перфорации. Таким образом, величина T_Q+s_Q равна «предположительному» значению периода перфорации T_P.

Последовательно перебираются различные значения величин T_Q и s_Q. Для каждого сочетания строится соответствующая корреляционная матрица по выборке достаточно большого объема. Определяется количество ненулевых элементов в ней, на основе чего делаются выводы о величине периода перфорации и возможного количества удаляемых символов внутри одного периода перфорации. (Но не их расположения, оно будет определяться алгоритмом, описанным в следующем параграфе).

При любой структуре маски перфорации внутри периода обязательно должна присутствовать подряд идущая пара неудаленных символов исходного кода, а в любой паре символов нельзя удалять сразу оба символа. Кроме этого, поскольку минимальное значение периода перфорации равно T_P=3 (что соответствует простейшему коду), то возможные значения величины s могут

находиться только внутри интервала $2 \le s \le T_P - 2$. Кроме этого, поскольку при скорости R=1/2 величина периода перфорации всегда четное число, то выбранные текущие значения T_Q и s_Q должны быть либо одновременно четными числами, либо одновременно нечетными числами. В качестве вставляемых s_Q дополнительных символов используются s_Q предыдущих принятых символов перфорированной последовательности. Далее полученная дополненная последовательность подвергается обработке аналогично алгоритму по параграфу 2.4, т.е. она разбивается на две частные последовательности, из них выбираются пары групп по L символов, строится корреляционная матрица, и т.д. Должно соблюдаться условие $L>T_P=T_Q+s_Q$.

Кроме описанного вставления дополнительных символов, отличие от алгоритма, описанного в параграфе 2.4, состоит в том, что здесь первый символ, начиная с которого формируется каждая последующая пара групп, по которым заполняется корреляционная матрица, сдвинут относительно первого символа предыдущей пары не на произвольную величину, а *точно* на величину T_Q+s_Q. При этом не имеет значения, в каком месте группы расположены вновь вставленные дополнительные символы.

Возможность при подобном переборе параметров сделать правильные выводы об оцениваемой величине T_P основывается на следующих особенностях предлагаемого алгоритма.

Рассмотрим первый из примеров перфорированных кодов, приведенных в данном параграфе, соответствующие параметры которого равны T_P=4 и s=1. Если обозначить через соответствующий порождающий полином формируемый с его помощь символ (т.е., 7 или 5), то произвольный отрезок перфорированной кодовой последовательности будет иметь вид: …, 7, 5, 7, 7, 5, 7, 7, 5, 7, 7, 5, 7, 7, 5, 7,…, и т.д.

Предположим, при переборе возможных значений T_Q и s_Q выбраны «правильные» значения s_Q=s=1, T_Q=T_P–s_Q=3. И пусть оказалось, что дополнительный символ вставляется через S=2 символов по отношению к

первому из пары символов исходного кода, не подвергшегося перфорации. Обозначим дополнительный символ через *Y*. Тогда дополненная кодовая последовательность будет иметь вид: $\mathbf{u}_0$=…,7, 5, 7, *Y*, 7, 5, 7, *Y*, 7, 5, 7, *Y*, 7, 5, 7, *Y*,…, и т.д.

Для формирования координат элемента матрицы необходимо по *L* символов для каждой из двух частных последовательностей, т.е. набор из $2L$ подряд идущих символов. Пусть используется величина L=5 и какой-либо конкретный набор $\mathbf{U}_i$ начинается с пары неперфорированных символов. Тогда он будет иметь вид: $\mathbf{U}_i$=7, 5, 7, *Y*, 7, 5, 7, *Y*, 7, 5. Поскольку следующий набор сдвинут по отношению к нему на T_Q+s_Q символов, то он будет иметь вид $\mathbf{U}_{i+1}$=7, 5, 7, *Y*, 7, 5, 7, *Y*, 7, 5. Также и следующий набор $\mathbf{U}_{i+2}$=7, 5, 7, *Y*, 7, 5, 7, *Y*, 7, 5 , и т.д. Таким образом, из-за кратности сдвига величине T_Q+s_Q, структура всех наборов будет одинакова.

После разделения в соответствии с алгоритмом на две частные последовательности: $\mathbf{u}_1$ и $\mathbf{u}_2$ из них будут также сформированы наборы $\mathbf{V}_1$=7, 7, 7, 7, 7 и $\mathbf{V}_2$= 5, *Y*, 5, *Y*, 5, являющиеся в двоичной системе исчисления координатами соответствующего элемента корреляционной матрицы. Для удобства пронумеруем последовательность символов с помощью индексов, т.е.: $\mathbf{V}_1$=7_1, 7_2, 7_3, 7_4, 7_5 и $\mathbf{V}_2$= 5_1, *Y*, 5_3, *Y*, 5_5.

Согласно правилам алгоритма, в качестве вставляемого дополнительного символа используется предыдущий символ общей кодовой последовательности, т.е. $\mathbf{V}_2$= 5_1, 7_2, 5_3, 7_4, 5_5. А если бы перфорации не производилось (как в п. 2.4), то второй набор имел бы вид: $\mathbf{V}_2$*= 5_1, 5_2, 5_3, 5_4, 5_5.

Основой результатов по алгоритму п. 2.4 являлось то, что любому конкретному набору $\mathbf{V}_1$ могут соответствовать не произвольные наборы $\mathbf{V}_2$* , а только определенное ограниченное их число, поскольку они формируются из общей исходной информационной последовательности с помощью *постоянного* правила (определяемого порождающим полиномом) и зависят от кодового ограничения кодера. Однако и наборы $\mathbf{V}_2$, формируемые с помощью предлагаемого алгоритма, также формируются из общей исходной

информационной последовательности с помощью *постоянного правила*. В результате любому конкретному набору двоичных символов вектора $\mathbf{V}_1$ могут соответствовать не произвольные наборы символов вектора $\mathbf{V}_2$, а также только определенное *ограниченное* их число. А, следовательно, и корреляционная матрица будет заполнена единицами не полностью при любой длине используемой выборки.

Нетрудно заметить, что величина сдвига S при этом не имеет значения. В частности, если окажется, что S=1, то соответствующие вектора будут иметь вид:

$\mathbf{u}_0$=…,7, 5, *Y*, 7, 7, 5, *Y*, 7, 7, 5, *Y*, 7, 7, 5, *Y*, 7,…, и т.д.

В результате структура (т.е., расположение вставляемого символа относительно других символов в группе, определяющей положение элемента в корреляционной матрице) будет также постоянным:

$\mathbf{U}_i$=7, 5, Y, 7, 7, 5, Y, 7, 7, 5,

$\mathbf{V}_1$=7_1, Y, 7_3, Y, 7_5.

После подстановки:

$\mathbf{V}_1$=7_1, 5_1, 7_3, 5_3, 7_5,

$\mathbf{V}_2$= 5_1, 5_2, 5_3, 5_4, 5_5.

Относительная структура $\mathbf{V}_1$ и $\mathbf{V}_2$ в этом случае также остается постоянной при всех индексах i. При этом любому конкретному набору $\mathbf{V}_2$ также могут соответствовать не произвольные наборы $\mathbf{V}_1$, а также только определенное *ограниченное* их число, такое же, как и при S=2, и корреляционная матрица также будет заполнена единицами не полностью при любой длине используемой выборки.

Аналогичные выводы можно сделать и при значениях S=0. Величина используемого L при соблюдении условия $L>T_P=T_Q+s_Q$ также не имеет принципиального значения, однако, как и в случае, рассмотренном в п. 2.4. ее неоправданное увеличение приводит к значительному затягиванию диагностической процедуры.

Теперь рассмотрим ситуацию, когда выбрано «неправильное» значение T_Q=4 (s по-прежнему равно единице). В этом случае кодовая последовательность

будет иметь вид: $\mathbf{u}_0$=…,7, 5, 7, 7, *Y*, 5, 7, 7, 5, *Y*, 7, 7, 5, 7, *Y*, 7, 5, 7, 7, *Y*,…, и т.д. Величину *L* при этом в соответствии с условием $L>T_P=T_Q+s_Q$ необходимо увеличить. Тогда, пусть при *L*=6 некоторый конкретный набор из 2*L* символов будет иметь вид: $\mathbf{U}_i$=7, 5, 7, 7, *Y*, 5, 7, 7, 5, *Y*, 7, 7. Однако каждый из последующих наборов уже будет отличаться по структуре: $\mathbf{U}_{i+1}$=5, 7, 7, 5, *Y*, 7, 7, 5, 7, *Y*, 7, 5 ; $\mathbf{U}_{i+2}$=7, 7, 5, 7, *Y*, 7, 5, 7, 7, *Y*, 5, 7 ; и т.д.

Соответственно, из них будут сформированы такие группы символов, определяющих положение элементов в корреляционной матрице:

$\mathbf{V}_{1,i}$=7, 7, *Y*, 7, 5, 7;

$\mathbf{V}_{2,i}$=5, 7 ,5, 7, *Y*, 7.

$\mathbf{V}_{1,i+1}$=5, 7, *Y*, 7, 7, 7;

$\mathbf{V}_{2,i+1}$= 7, 5, 7, 5 ,*Y*, 5.

$\mathbf{V}_{1,i+2}$=7, 5, *Y*, 5, 7, 5;

$\mathbf{V}_{2,i+2}$=7, 7, 7, 7, *Y*, 7.

После замены *Y* на предыдущий символ исходной последовательности $\mathbf{u}_0$ группы символов примут следующий вид:

$\mathbf{V}_{1,i}=7_1, 7_2, 7_3, 7_4, 5_5, 7_6$;

$\mathbf{V}_{2,i}=5_1, 7_3 ,5_3, 7_5, 5_5, 7_7$.

$\mathbf{V}_{1,i+1}=5_3, 7_4, 5_5, 7_7, 7_8, 7_9$;

$\mathbf{V}_{2,i+1}=7_4, 5_5, 7_6, 5_7 ,7_8, 5_9$.

$\mathbf{V}_{1,i+2}=7_6, 5_7, 7_8, 5_9, 7_{11}, 5_{11}$;

$\mathbf{V}_{2,i+2}=7_7, 7_8, 7_9, 7_{10}, 7_{11}, 7_{12}$.

Для удобства сравнения структуры пар векторов на различных шагах, нумерацию индексов на шагах *i*+1 и *i*+2 сдвинем к единице. Тогда на этих шагах структуры будет иметь вид:

$\mathbf{V}^{**}_{1,i+1}=5_1, 7_2, 5_3, 7_5, 7_6, 7_7$;

$\mathbf{V}^{**}_{2,i+1}=7_2, 5_3, 7_4, 5_5, 7_6, 5_7.$

$\mathbf{V}^{**}_{1,i+2}=7_1, 5_2, 7_3, 5_4, 7_6, 5_6;$

$\mathbf{V}^{**}_{2,i+2}=7_2, 7_3, 7_4, 7_5, 7_5, 7_7.$

Из сравнения структуры пар векторов на различных шагах следует вывод, что на каждом последующем шаге эти структуры *различны* и постоянно меняются. Это обусловлено тем, что данный вариант T_Q+s_Q не совпадает с истинным значением периода перфорации, в результате из-за некратности периодов не возникает постоянства относительной структуры векторов $\mathbf{V}_1$ и $\mathbf{V}_2$.

Таким образом, если значения параметров выбраны неправильно, то конкретному двоичному числу, определяемому вектором $\mathbf{V}_1$, может соответствовать любое двоичное число вектора $\mathbf{V}_2$, а не ограниченное количество их вариантов. В результате при выборке достаточно большого объема соответствующая корреляционная матрица будет заполнена единичными элементами практически полностью. Поэтому индикатором правильности выбранного варианта параметров служит заполненность корреляционной матрицы: если при достаточно большой выборке доля заполненных ячеек стремится к определенной величине, заметно меньшей единицы, то оценка значений параметров правильная. (При этом, чем больше значение L, тем эта величина меньше). Если же при возрастании объема выборки матрица стремится полностью заполниться, то оценка параметров неверна.

Правило соблюдения кратности справедливо и при $s>1$. Проиллюстрируем это на примере. Для общности обозначим код первой частной последовательности через *a*, код второй частной последовательности через *b*. Пусть матрица перфорации имеет вид: $\begin{pmatrix} a, X, a \\ b, b, X \end{pmatrix}$. Тогда маска перфорации имеет вид: *a,b,X,b,a,X*. После перфорации некоторый отрезок последовательности примет вид: …,*a,b,b,a,a,b,b,a,a,b,b,a*,… В случае использования «правильных» параметров перфорации (T_Q=4, s_Q=2) после добавления двух дополнительных символов последовательность приобретет вид: …, *a,b,b,a,Y,Y,a,b,b,a,Y,Y,a,b,b,a,Y,Y*,… Пусть

используется L=7, но выборки длиной 2L берутся через T_Q+s_Q =6 символов. Тогда структура всех выборок будет одинакова: *a,b,b,a,Y,Y,a,b,b,a,Y,Y,a,b*. Соответственно, будет постоянна и относительная структура первой и второй групп символов, определяющих положение элементов корреляционной матрицы: *a,b,Y,a,b,Y,a* и *b,a,Y,b,a,Y,b*.

Если же используется «неправильная» оценка параметров, например, T_Q=5, s_Q=1, то после добавления дополнительного символа последовательность приобретет вид: …,*a,b,b,a,a,Y, b,b,a,a,b,Y,b,a,a,b,b,Y,a*,… При том же значении L структура различных необходимых выборок длиной 2L будет различаться: *a,b,b,a,a,Y,b,b,a,a,b,Y,b,a*; *b,b,a,a,b,Y,b,a,a,b,b,Y,a,a*, и т.д. Соответственно будут различаться и различные относительные структуры первой и второй групп символов: *a,b,a,b,a,b,b* и *b,a,Y,b,a,Y,a*; *b,a,a,b,a,b,a* и *b,a,Y,a,b,Y,a*; и т.д. В результате соответствующая корреляционная матрица заполнится полностью. Приведенный пример иллюстрирует, что дополнительным условием «правильности» кроме T_Q+s_Q=T_P является и необходимость совпадения числа удаляемый символов в периоде перфорации, т.е.: s_Q=s.

Следует отметить, что правило укажет на правильность решения и в случаях, когда величина T_P кратна T_Q+s_Q, поэтому алгоритм перебора следует начинать с небольших значений T_Q и s_Q, далее их увеличивая до достижения «правильного» решения. После получения «правильного» решения возможно также определить величину используемого кодового ограничения, что будет рассмотрено далее.

В соответствии с этим был разработан алгоритм определения периода перфорации и величины кодового ограничения [101]. Укрупненная схема алгоритма представлена на рисунке 4.1.

При осуществлении операций алгоритма перебор значений T_Q и s_Q начинается с минимального значения T_Q=3 и соответствующего ему значения s=1. Вставляются дополнительные символы, и определяется вид корреляционной матрицы и ее заполненность. Если при достаточно большой выборке матрица оказывается практически заполненной, то сочетание значений и признается

неудачным. После этого последовательно увеличивается значение T_Q до некоторого T_{Qmax} и при различных s каждый раз вновь производится описанная процедура анализа.

Рисунок 4.1.

Из всех вариантов только при правильном варианте параметров корреляционная матрица оказывается не заполненной полностью. Этот вариант принимается в качестве оценки периода перфорации.

Количество перебираемых вариантов может быть уменьшено, т.к., согласно [ ], максимальная величина T_{Qmax} и соответствующее ей значение s стоит выбирать не более T_{Qmax}=8 при s=3. Это обеспечивает увеличение кодовой скорости от R=1/2 до R=7/8, хотя эти величины не являются принципиальными ограничениями алгоритма. Кроме этого, следует учитывать, что при кодовой скорости R=1/2 исходного сверточного кода значение T_Q+s_Q должно быть четным числом.

Далее на рисунках 4.2-4.5 представлены в качестве примера некоторые результаты, иллюстрирующие работу алгоритма. Рисунки отражают заполненность корреляционной матрицы. Точки указывают на присутствие единицы в соответствующем элементе корреляционной матрицы. Во всех экспериментах использовалось 10^4 пар групп символов. Результаты приведены для исходного сверточного кода (13, 15). Период перфорации составлял T_P=4. Кодовая скорость преобразовывалась с R=1/2 до R=3/4, т.е. величина s составляла s=1. Рисунки 4.2, 4.3 и 4.5 получены при одинаковом «правильном» значении параметра T_Q=3. Рисунок 4.4 получен при «неправильном» значении параметра T_Q=4. Рисунки 4.2 и 4.3 получены при значении сдвига, равном, соответственно, S=0 и S=1. Рисунки 4.2, 4.3 и 4.4 получены при L=5, рисунок 4.5. получен при L=6.

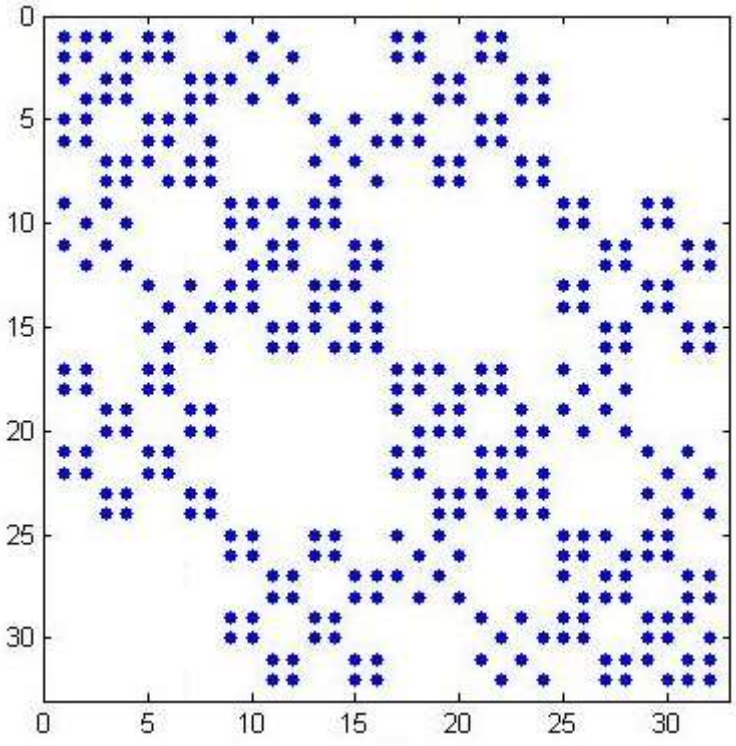

Рисунок 4.2.

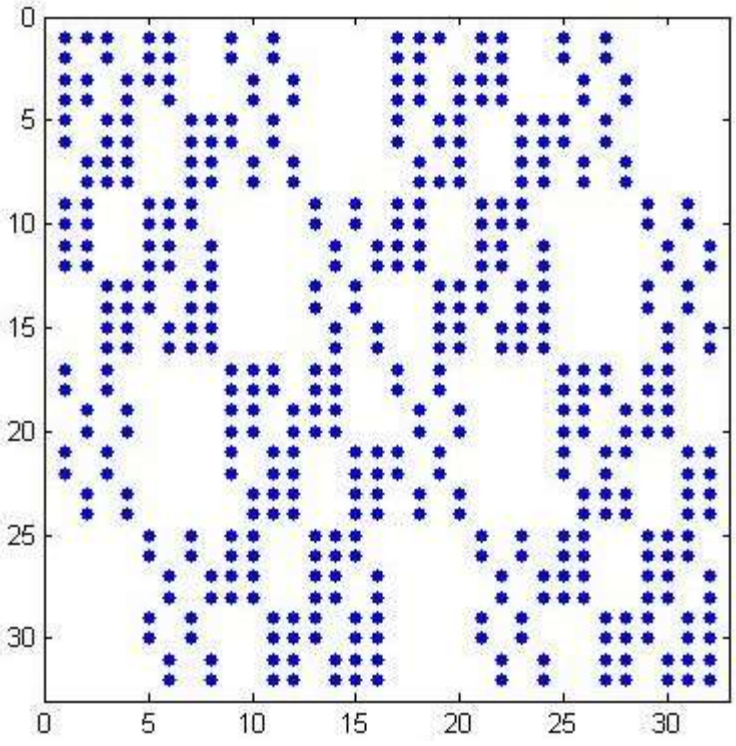

Рисунок 4.3.

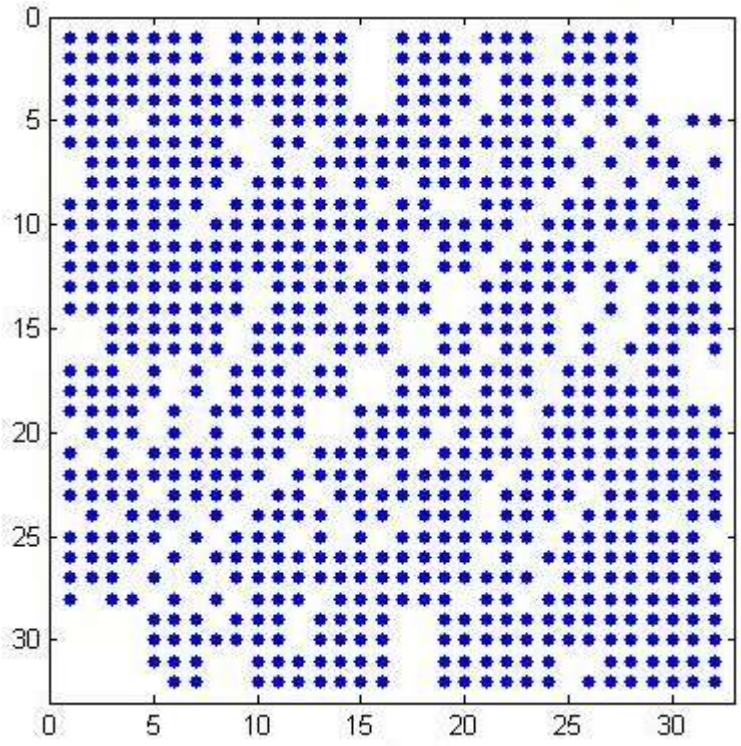

Рисунок 4.4.

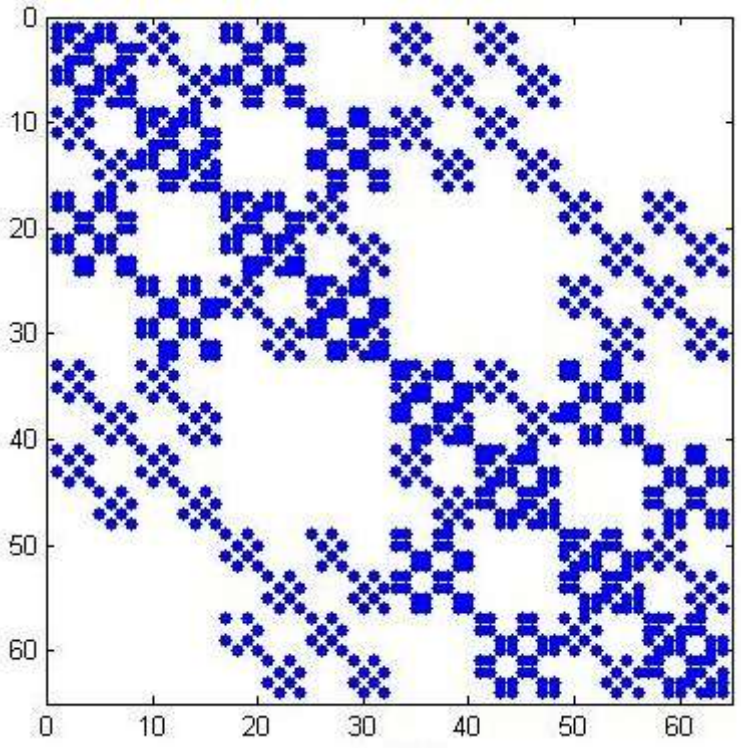

Рисунок 4.5.

Рисунки отражают заполненность корреляционной матрицы. Точки указывают на присутствие единицы в соответствующем элементе корреляционной матрицы. Во всех экспериментах использовалось 10^4 пар групп

символов. Результаты приведены для исходного сверточного кода (13, 15). Период перфорации составлял T_P=4. Кодовая скорость преобразовывалась с R=1/2 до R=3/4, т.е. величина s составляла s=1. Рисунки 4.2, 4.3 и 4.5 получены при одинаковом «правильном» значении параметра T_Q=3. Рисунок 4.4 получен при «неправильном» значении параметра T_Q=4. Рисунки 4.2 и 4.3 получены при значении сдвига, равном, соответственно, S=0 и S=1. Рисунки 4.2, 4.3 и 4.4 получены при L=5, рисунок 4.5. получен при L=6.

Все рисунки отражают факт, что при «правильном» решении корреляционные матрицы оказываются заполненными лишь частично, причем наблюдается хорошо выраженная структурированность рисунка заполнения матриц. С ростом длины групп символов L эта структурированность выражена сильнее. При «неправильном» выборе параметров матрица заполнена практически полностью.

Заполненность матрицы при «правильном» решении используется для определения кодового ограничения. При этом с некоторой коррекцией применима формула (2.4.2), полученная в параграфе 2.4. Поскольку теперь конкретной группе символов одной частной последовательности может соответствовать меньшее количество вариантов символов другой частной последовательности, что отражается на заполненности матрицы, то величина кодового ограничения теперь должна оцениваться по формуле (в тех же обозначениях):

$$K=L-\log_2\gamma+1+s. \qquad (4.1)$$

В случае, когда при кодировании используется другая кодовая скорость (например, R=1/3) , т.е. применяются три сумматора по модулю 2, то здесь также может быть применен подход, описанный в главе 2. В этом случае три частные кодовых последовательностей попарно объединяются тремя вариантами, и для каждого производится оценка согласно описанному алгоритму. При этом получаемые результаты частично взаимно дублируются, что повышает достоверность оценки.

Как и в случае отсутствия перфорации, работа при значительном уровне битовых ошибок в принятом сигнале ухудшает свойства алгоритма. Это приводит

к искажению корреляционной матрицы. В такой ситуации применимы решения, аналогичные решениям, описанным в главе 2. Одно из них, не требующее изменения алгоритма, заключается в проведении нескольких повторных оценок по различным выборкам и выработке решения по мажоритарному принципу. Другое решение – более простое, однако, требующее значительно большего общего объема обрабатываемой последовательности символов и времени анализа.

Оно заключается в формировании корреляционной матрицы по «арифметическому» принципу. Элементы матрицы принимают не бинарные значения, а получаются последовательным поцикловым накоплением. При получении в каждом цикле координат очередного элемента матрицы к его содержимому прибавляется единица независимо от его предыдущего содержания.

При этом после анализа выборки достаточного объема содержание практически всех элементов корреляционной матрице может быть ненулевым. В каждом элементе будет находиться некоторое число, равное количеству повторений сочетания символов групп первой и второй частных последовательностей. Однако, если сравнить со случаем незначительных шумов при тех же параметрах кодирования и перфорации, то там, где раньше были нули, накопленные суммы будут иметь незначительную величину, а там, где раньше были единицы, в элементах матрицы будут накоплены достаточно большие числа. При использовании выборки достаточно большого объема соотношение среднего значения накопленных сумм в ячейках первого и второго вида буде приближаться к величине битовой ошибки в данной системе передачи. После сравнения с некоторым пороговым уровнем ячейки с малым накопленным уровнем обнуляются, а ячейкам с большим накопленным уровнем присваиваются единичные значения. Далее процедура анализа производится аналогично предыдущей.

В данном параграфе изложены пути диагностики части необходимых параметров – периода перфорации и величины кодового ограничения. Для полной диагностики необходимо еще определить структуру маски перфорации внутри

периода перфорации, а также вид используемых для кодирования порождающих полиномов. Эти задачи решаются в следующем параграфе.

4.3. Алгоритм определения маски перфорации и порождающих полиномов

Как уже упоминалось, в задачу диагностики параметров перфорированного сверточного кода кроме периода перфорации и кодового ограничения входит определение структуры маски перфорации и вида порождающих полиномов, используемых при кодировании. Для этого используется алгоритм, схожий с алгоритмом, описанным в параграфе 2.2, но с применением поискового метода.

Алгоритм предполагает, что величина периода перфорации уже определена и количество удаляемых символов в периоде также известно. При этом в процессе поиска производится перебор возможных вариантов расположения удаленных символов. При «правильном» расположении одновременно определяется структура полиномов, «неправильное» расположение характеризуется тем, что в процессе анализа ни одной комбинации порождающих полиномов получить не удается.

В алгоритме, предложенном в параграфе 2.2,, процесс поиска вида порождающих полиномов заключается в том, что каждая частная кодовая последовательность $\mathbf{u}_1$ и $\mathbf{u}_2$ повторно кодируется «поисковыми» полиномами различного значения. При этом для каждой новой группы из K символов последовательностей $\mathbf{u}_1$ и $\mathbf{u}_2$ выполняется свой цикл анализа, в котором перебираются все варианты сочетаний «поисковых» полиномов. Фиксируются факты совпадения значений символов после повторного кодирования и запоминаются варианты полиномов, при которых это происходит. Фиксация результатов сравнения производится в матричной форме, при этом каждый вариант сочетания в двоичной форме представляет собой координаты элемента матрицы, в который заносится единичной значение.

Когда производится следующий цикл, то вырабатывается следующая матрица. Далее она поэлементно перемножается с предыдущей матрицей, и.т.д. Элемент с «правильным» сочетанием полиномов будет получать единичные значения в каждом цикле, а вероятность получения единичного значения для всех «неправильных» сочетаний близка к 0,5. В результате количество единичных элементов в матрице будет убывать с каждым циклом, пока не останется единственный ненулевой элемент. Координаты его (номера строки и столбца в двоичной записи) принимаются за результат диагностики.

В случае перфорации часть символов удалена, поэтому их значения неизвестны. Но период перфорации уже определен, поэтому внутри него количество вариантов расположения удаленных символов ограничено. Варианты удаления последовательно перебираются. Вместо удаленных символов вставляются *дополнительные* символы, значения которых устанавливаются по определенному правилу, после чего производится процедура анализа по алгоритму параграфа 2.2. В случае «правильного» расположения дополнительных символов, совпадающем с расположением удаленных символов, после анализа остается единственное решение (единственный ненулевой элемент матрицы), координаты которого соответствуют искомым порождающим полиномам, а расположение этих удаленных символов укажет на искомую структуру маски перфорации.

В случае «неправильного» расположения дополнительных символов, не совпадающих с расположением удаленных символов, после определенного количества циклов матрица обнуляется полностью и в ней не остается ни одного ненулевого элемента. Это является индикатором неправильного решения, после чего необходимо переходить к анализу следующего варианта расположения дополнительных символов.

Дополнительные вставляемые символы заменяют удаленные, но о значении их на приемной стороне ничего не известно, т.е. они могут принимать любые из двух возможных значений. Поэтому, если для нахождения решения необходимо произвести C циклов анализа, и в полученной последовательности из C символов

присутствуют D вставленных символов, то существует 2^D вариантов сочетаний значений вставляемых символов. Один из этих вариантов, соответствующий правильному значению удаленных символов. В остальных сочетаниях вставляемых символов матрицы, получаемые в каждом цикле, не будут иметь постоянного ненулевого элемента, и после их полного обнуления эти неудачные варианты будут отвергнуты.

В общем случае процедуру, описанную в параграфе 2.2, необходимо производить 2^D раз, однако практически можно значительно уменьшить время анализа, генерируя варианты по мере перехода к каждому следующему циклу, и сразу отвергая неудачные.

Покажем на примере, что описанная процедура при «правильном» выборе расположения вставляемых символов сформирует правильное решение, а при «неправильном» выборе соответствующая матрица обнуляется.

Обозначим символы в исходной кодовой последовательности буквой A и пронумеруем их индексы с первого символа от начала рассмотрения, т.е.:
…, A_1, A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9, A_{10},… Пусть при перфорации удаляется каждый четвертый символ, перфорацию, как принято, обозначим буквой X:
…, A_1, A_2, A_3, X, A_5, A_6, A_7, X, A_9, A_{10}, A_{11}, X, A_{13},… После удаления символов последовательность примет вид: …, A_1, A_2, A_3, A_5, A_6, A_7, A_9, A_{10}, A_{11}, A_{13}, A_{14},…
После «правильного» вставления дополнительных символов: …, A_1, A_2, A_3, Y, A_5, A_6, A_7, Y, A_9, A_{10}, A_{11}, Y, A_{13},… Разделение на две частные последовательности:

…A_1, A_3, A_5, A_7, A_9, A_{11}, A_{13}, A_{15},… (4.2)
…A_2, Y, A_6, Y, A_{10}, Y, A_{14}, Y,…

Если бы не было перфорации, то соответствующие частные последовательности имели бы вид:

…A_1, A_3, A_5, A_7, A_9, A_{11}, A_{13}, A_{15},… (4.3)
…A_2, A_4, A_6, A_8, A_{10}, A_{12}, A_{14}, A_{16},…

Удаленные при перфорации символы …, A_4, A_8, A_{12}, A_{16}… имели определенные двоичные значения, и при переборе всех вариантов значений последовательности Y один из вариантов будет соответствовать правильному и даст оценку вида порождающих полиномов. Любой другой вариант последовательности Y означал бы, что при генерации второй последовательности вид порождающего полинома иногда меняется, а раз в матрице не будет неизменной позиция элемента, где всегда появляется единица, то матрица вскоре полностью обнулится.

Теперь рассмотрим вариант «неправильного» размещения вставляемых дополнительных символов: …, A_1, A_2, Y, A_3, A_5, A_6, Y, A_7, A_9, A_{10}, Y, A_{11}, A_{13}, A_{14},… Разделение на две частные последовательности в этом случае будет иметь вид:

$$\begin{aligned} &\ldots, A_1, Y, \ A_5, Y, \ A_9, \ \ Y, \ \ A_{11}, Y, \ldots \\ &\ldots, A_2, A_3, A_6, A_7, A_{10}, A_{11}, A_{13}, A_{14}, \ldots \end{aligned} \qquad (4.4)$$

Нетрудно видеть, что «правильная» вторая частная последовательность, согласно формуле (4.3) должна была бы содержать только четные номера символов. Но вторая частная последовательность по формуле (4.4) содержит и четные и нечетные номера, т.е. чередующиеся символы из первой и второй исходных частных последовательностей. Следовательно, она не является сформированной из исходной информационной последовательности сверткой с *постоянным* одним и тем же порождающим полиномом. Поэтому ненулевого элемента с постоянным расположением в матрице не будет при всех возможных вариантах последовательности значений дополнительных элементов Y и матрица постепенно обнулится.

Другой вариант расположения вставляемых элементов может иметь вид: …, A_1, Y, A_2, A_3, A_5, Y, A_6, A_7, A_9, Y, A_{10}, A_{11}, A_{13}, Y, A_{14},… Разделение на частные последовательности:

…, A_1, A_2, A_5, A_6, A_9, A_{10}, A_{13},A_{14}, …

…, Y, A_3, Y, A_7, Y, A_{11}, Y, A_{15}, …

Здесь уже у сформированной первой частной последовательности идет чередование символов из первой и второй исходных частных последовательностей, следовательно, она не является продуктом свертки исходной информационной последовательности с постоянным порождающим полиномом, и в матрице также не будет единичного элемента с постоянным расположением, и она обнулится.

Сходные выводы можно получить, рассматривая другие периоды и маски перфорации. Для сокращения времени анализа варианты последовательности Y можно обрабатывать попарно, таким образом, чтобы в пределах L для одного из вставляемых дополнительных символов оценивать сразу два бинарных варианта значения. На рисунках 4.6 – 4.22 приведены в качестве примера результаты поцикловой работы алгоритма для случаев «правильного» (рисунки 4.6 – 4.14) и «неправильного» (рисунки 4.15 – 4.22) расположения вставляемых дополнительных символов, полученные с использованием [101]. Представлены эксперименты для кода (13, 15), количество L в этих экспериментах равнялось L=5.

Все рисунки построены по одинаковой схеме. Они объединены в тройки. В каждой тройке левый и центральный рисунки показывают две матрицы, полученные в каждом цикле для двух вариантов значений одного из вставляемых дополнительных символов. Единичные элементы матриц обозначены точками. В каждом цикле одинаковые элементы этих двух матриц объединяются операцией «или». Правая матрица иллюстрирует накопление результатов анализа, т.е. она показывает элементы, оставшиеся единичными по всем предыдущим циклам.

Левая и центральная матрицы во всех циклах разные, поэтому с каждым следующим циклом общих элементов остается все меньше, пока на цикле 9 не останется единственный ненулевой элемент, координаты которого и определят оценку порождающих полиномов кодера (11_{10},13_{10}=13_8,15_8).

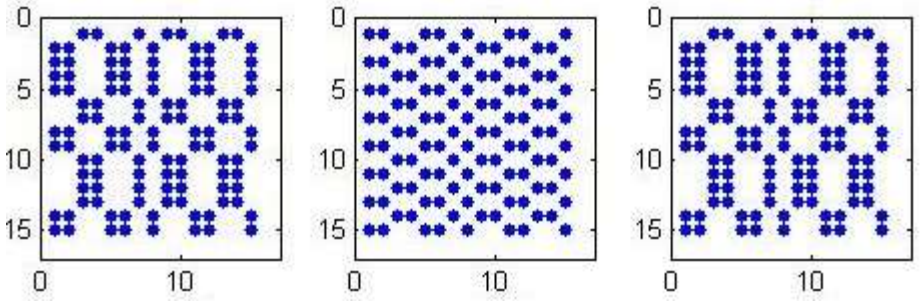

Рисунок 4.6 – Цикл 1.

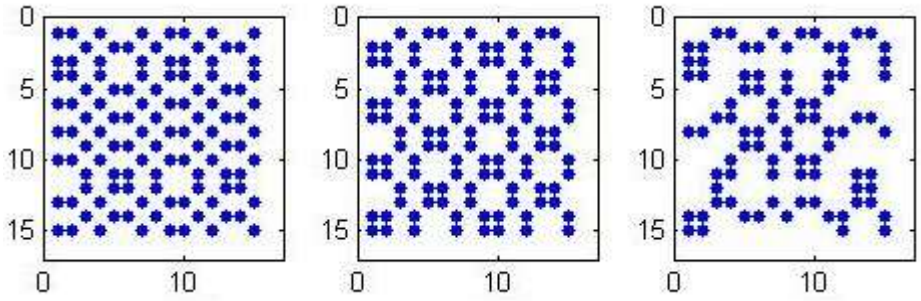

Рисунок 4.7 – Цикл 2.

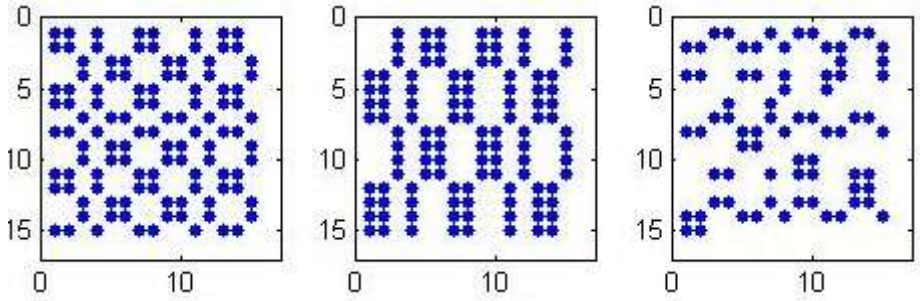

Рисунок 4.8 – Цикл 3.

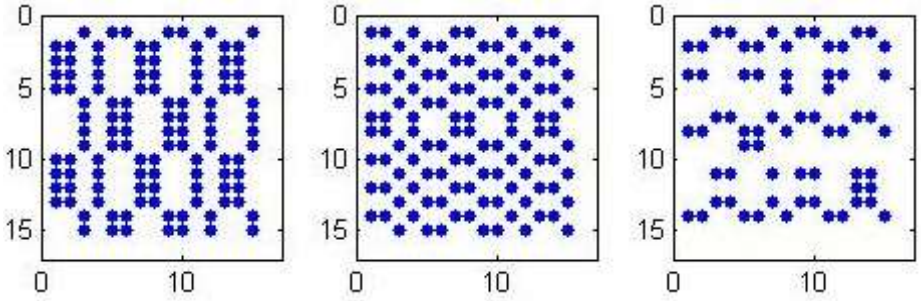

Рисунок 4.9 – Цикл 4.

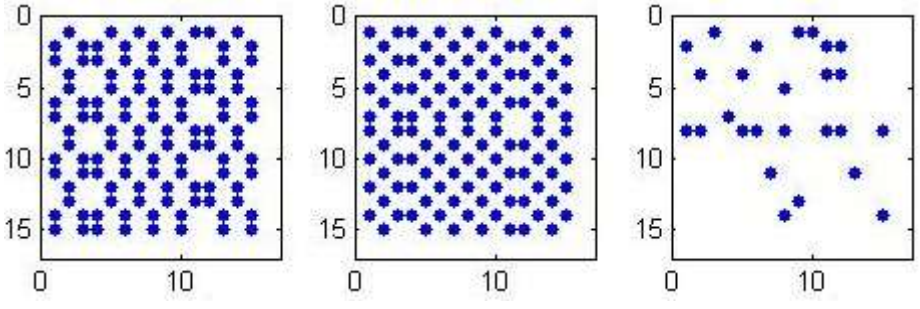

Рисунок 4.10 – Цикл 5.

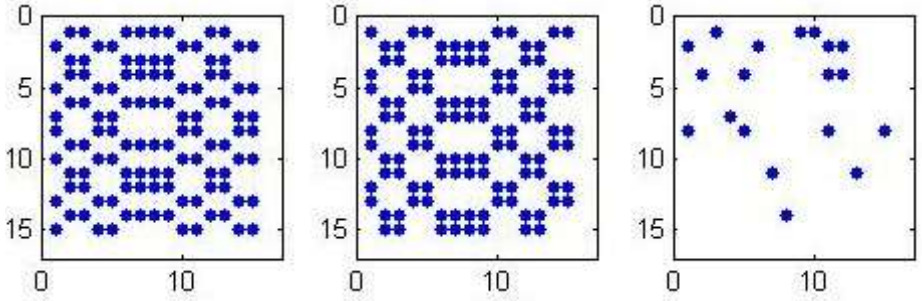

Рисунок 4.11 – Цикл 6.

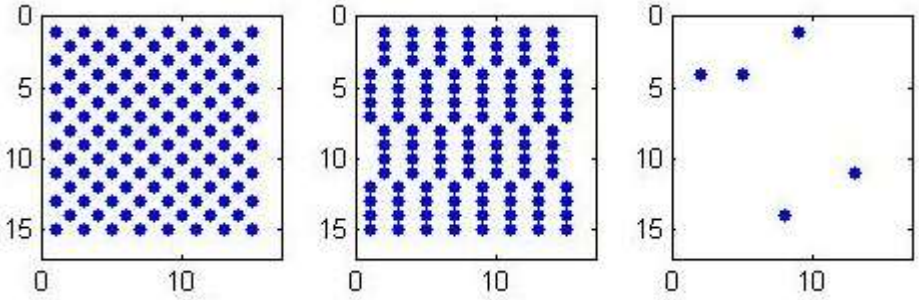

Рисунок 4.12 – Цикл 7.

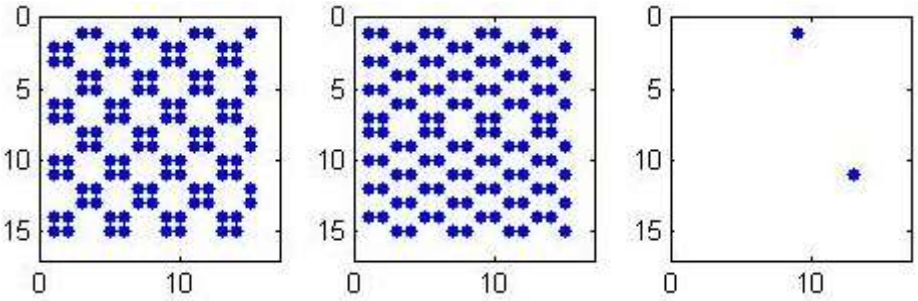

Рисунок 4.13 – Цикл 8.

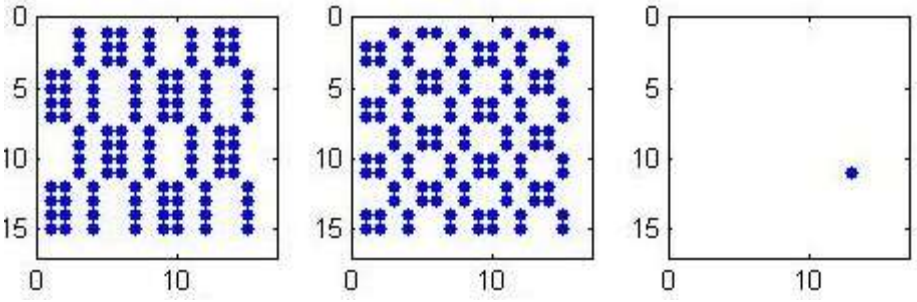

Рисунок 4.14 – Цикл 9.

На рисунках 4.15– 4.22 показана эволюция структуры общей матрицы при «неправильном» варианте маски перфорации. С каждым циклом количество

ненулевых элементов в результирующей матрице также уменьшается, пока не останутся только нулевые элементы.

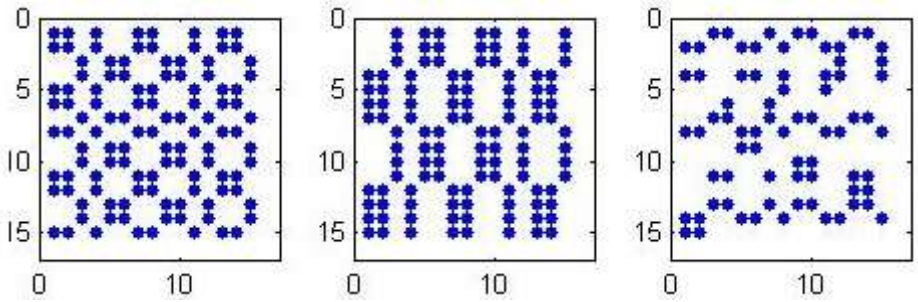

Рисунок 4.15.– Цикл 1.

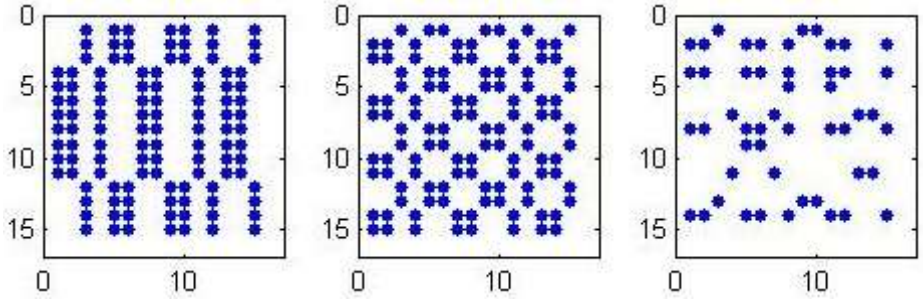

Рисунок 4.16.– Цикл 2.

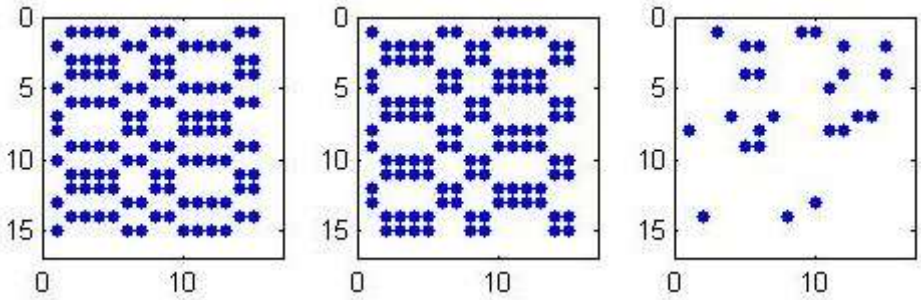

Рисунок 4.17.– Цикл 3.

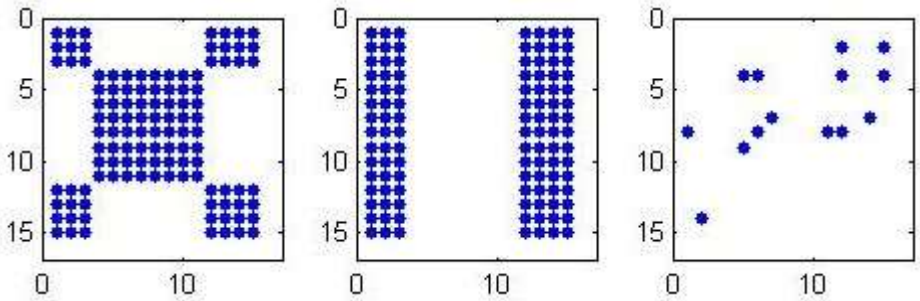

Рисунок 4.18.– Цикл 4.

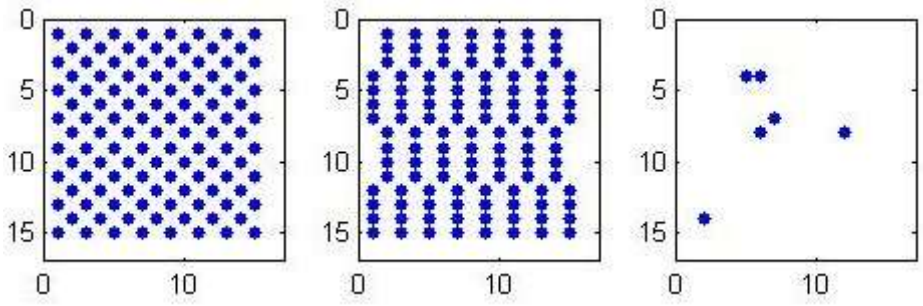

Рисунок 4.19.– Цикл 5.

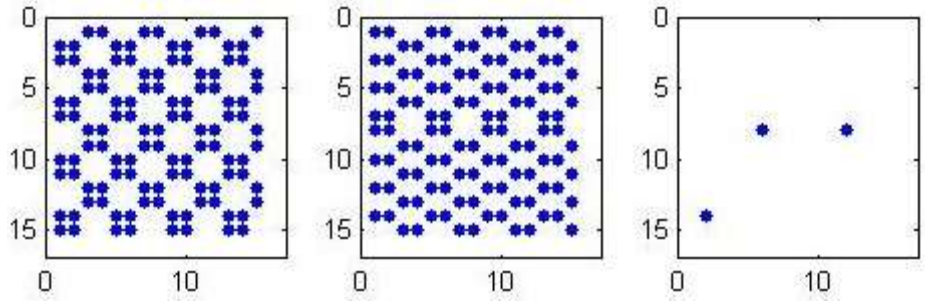

Рисунок 4.20.– Цикл 6.

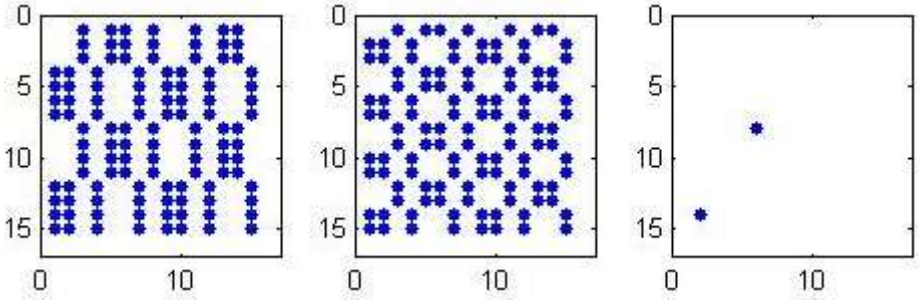

Рисунок 4.21.– Цикл 7.

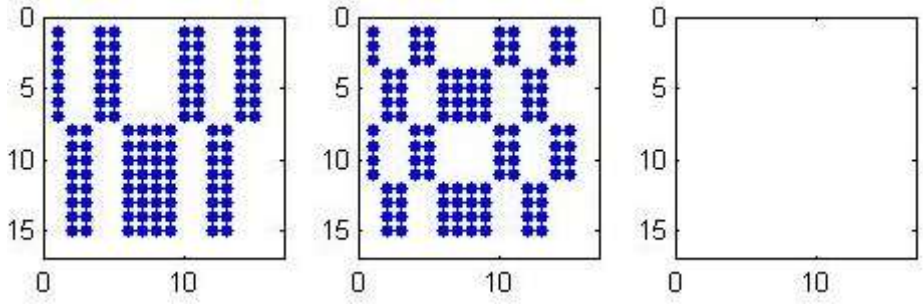

Рисунок 4.22.– Цикл 8.

Таким образом, общий алгоритм анализа имеет вид, представленный на рисунке 4.23. Он использует результаты работы алгоритма, описанного в параграфе 4.2, в частности уже определенные значения периода перфорации T_P и кодового ограничения K. После их ввода и ввода символов анализируемой

последовательности производится перебор вариантов маски перфорации, и на ее основе значений последовательности вводимых дополнительных символов *Y*. Для каждого из вариантов этой последовательности производится поцикловое формирование матрицы и анализ ее содержания. В случае появления одного ненулевого элемента, (т.е выбран «правильный» вариант маски перфорации) и его сохранения в течение нескольких циклов, принимается решение об удачной диагностике, и оценка вида порождающих полиномов, а также маска перфорации выводятся, как результат диагностики. Если этого на некотором цикле не случилось, то анализ продолжается далее.

Если выбран «неправильный» вариант маски перфорации, то матрица через определенное количество циклов полностью обнуляется. В этом случае производится переход к следующему варианту значений дополнительных символов *Y*. Если же все возможные варианты последовательности символов *Y* исчерпаны, то производится переход к следующему возможному варианту маски перфорации. Если и возможные варианты маски перфорации также уже исчерпаны, то алгоритм констатирует, что решение не принято. Это может быть обусловлено либо недостаточностью объема анализируемой выборки, либо появлением в ней ошибочных символов.

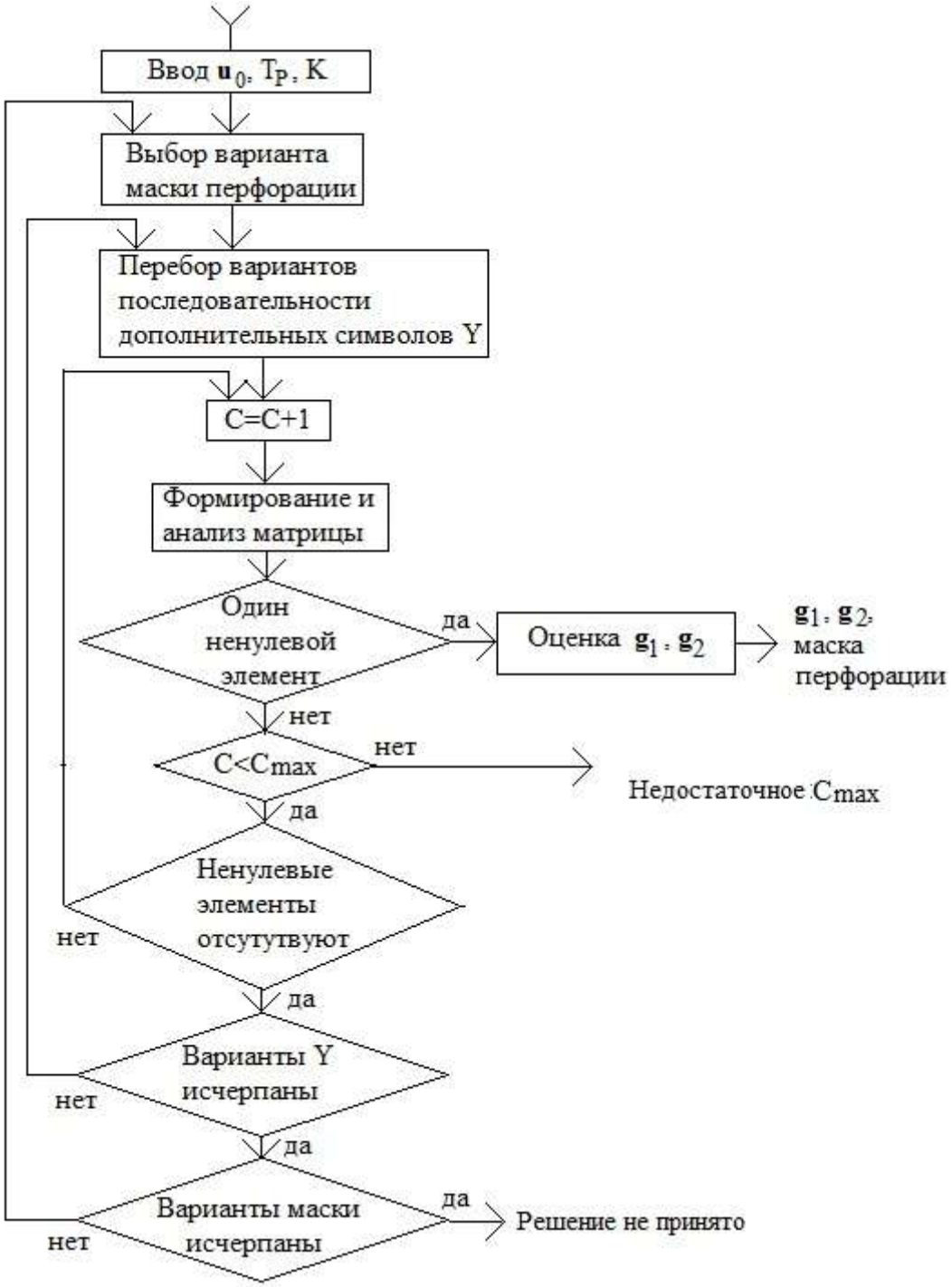

Рисунок 4.23.

Работа алгоритма также прекращается, если после осуществления заранее выбранного максимального количества циклов C_{max} никакого решения не получено. В этом случае вводимое значение C_{max} может быть увеличено, однако это приведет к замедлению работы алгоритма.

Когда кодовая скорость исходного сверточного кода отличается от R=1/2, то подобный анализ, как и в главе 2, производится попарно для каждой пары частных кодовых последовательностей. В случае работы при значительном уровне шумов всю процедуру анализа необходимо производить несколько раз, т.к. появление ошибок может привести к тому, что диагностическое решение принято не будет.

Таким образом, совместное последовательное использование алгоритмов, описанных в параграфах 4.2 и 4.3, позволяет определить искомую структуру перфорированных кодов и оценить вид используемых порождающих полиномов.

4.4. Выводы

1. Диагностика модифицированных кодов, построенных на исходных блоковых кодах, может производиться на основе алгоритмов, предложенных в главе 3. Диагностика перфорированных кодов, полученных путем выкалывания символов исходного сверточного кода, требует использования специализированных алгоритмов.
2. Предложены два алгоритма диагностики перфорированных кодов, предполагающих совместное последовательное использование. Один из алгоритмов позволяет определить период перфорации и величину кодового ограничения. Другой алгоритм дает возможность найти структуру маски перфорации и вид порождающих полиномов, используемых при кодировании.

Заключение

1. При эксплуатации систем передачи цифровых сигналов, использующих помехоустойчивое кодирование, могут возникать ситуации, когда информация о параметрах кодеров либо утрачена, либо неполная, либо отсутствует изначально. Однако при этом возможно восстановление данных о параметрах кодирования, так как при его осуществлении появляются определенные взаимосвязи между кодовыми символами. Их анализ позволяет произвести диагностику, на основе которой определить требуемые структуру и параметры кодеров.

2. Предложены алгоритмы для диагностики сверточных кодов, основанные на том, что различные последовательные кодовые символы принимаемой последовательности образованы из одной и той же исходной информационной последовательности, при этом в результате диагностики определяется величина кодового ограничения и структура используемого кодера.

3. Исследованы свойства алгоритмов диагностики для незначительного и для существенного уровня шумов, а также «быстрые» виды алгоритмов. При часто используемых параметров кодирования и малых уровнях шума для обеспечения вероятности неправильной диагностики ниже $2\cdot 10^{-3}$ достаточно произвести около 25 циклов анализа, а для снижения этого уровня до $2\cdot 10^{-5}$ достаточно 30 циклов анализа.

4. При большом уровне шумов (при вероятности битовой ошибки до 10^{-1}) для сохранения таких же показателей качества диагностики количество циклов анализа необходимо увеличить в 2-4 раза. Использование «быстрых» алгоритмов анализа позволяет в целом сократить время диагностики в 5-10 раз.

5. Для диагностики циклических кодов предложены алгоритмы, основанные на различных вариантах сравнения наборов простых полиномиальных множителей, полученных путем разложения принимаемых кодовых блоков.

6. Для диагностики линейных блоковых кодов, полученных с помощью использования порождающей матрицы, предложены алгоритмы с использованием вспомогательных невырожденных матриц.

7. Эффективность использования диагностики может быть оценена сравнением показателей помехоустойчивости передачи закодированного и незакодированного сигналов, при этом разница показателей при различных значениях параметров может составлять до 3 дБ.

8. На основе модификации предложенных выше алгоритмов разработаны процедуры диагностики укороченных, расширенных и перфорированных кодов.

9. Разработаны специализированные программные комплексы, с помощью которых проведено исследование предложенных диагностических алгоритмов и показана высокая эффективность их использования.

10. Учитывая вышесказанное, можно заключить, что создана методика диагностирования наиболее распространенных и важных видов кодов, что позволяет восстановить утерянную или поврежденную информацию об используемом кодере и повысить помехоустойчивость передачи сигналов.

Список литературы

1. Волков, Л.Н. Системы цифровой радиосвязи [Текст] / Л.Н.Волков, М.С.Немировский, Ю.С.Шинаков. – М.: Экотрендз, 2005. – 392 с.

2. Немировский, А.С. Системы связи и радиорелейные линии [Текст] / А.С. Немировский, Е.В. Рыжков. – М.: Связь, 1980. – 432 с.

3. Прокис, Дж. Цифровая связь [Текст] / Дж.Прокис; пер. с англ. – М.: Радио и связь, 2000. – 800 с.

4. Скляр, Б. Цифровая связь. Теоретические основы и практическое применение [Текст] / Б. Скляр; пер. с англ. – М.: Изд. дом “Вильямс”, 2003. – 1104 с.

5. Телекоммуникационные системы и сети [Текст] / Под ред. В.П. Шувалова.– М.: Горячая линия – Телеком, 2003, т. 1 – 647 с.; 2004, т. 2 – 672 с.

6. Телекоммуникационные системы и сети.. Радиосвязь, радиовещание и телевидение [Текст] / Под ред. В.П. Шувалова. – М.: Горячая линия-Телеком, 2004 т.3. – 673 с.

7. Феер, К. Беспроводная цифровая связь. Методы модуляции и расширения спектра [Текст] / К. Феер: пер. с англ.; под редакцией В.И.Журавлева – М.: Радио и связь, 2000. – 520 с.

8. Шмалько, А.В. Цифровые сети связи: Основы планирования и построения [Текст] / А.В. Шмалько. – М.: Эко-Трендз, 2001. – 282 с.

9. Основы построения телекоммуникационных систем и сетей: Учебник для вузов [Текст] / Под ред. Н.В. Гордиенко и В.В. Крухмалева. – М.: Горячая линия-Телеком, 204. – 510 с.

10. Галкин, В.А. Цифровая мобильная связь. Учебное пособие для вузов [Текст] / В.А. Галкин. – М.: Горячая линия–Телеком, 2007. – 432 с.

11. Цифровые и аналоговые системы передачи. Учебник для вузов [Текст] / Под ред. В.И. Иванова. – М.: Горячая линия–Телеком, 2003. – 232 с.

12. Системы мобильной связи. Учебное пособие для вузов [Текст] / Под ред. В.П. Ипатова. – М.: Горячая линия–Телеком, 2003. – 272 с.

13. Ратынский, М.В. Основы сотовой связи [Текст] / М.В. Ратынский – М.: Радио и связь, 1998. – 392 с.

14. Помехоустойчивость и эффективность систем передачи информации [Текст] / А.Г. Зюко и др.; под ред. А.Г. Зюко. – М.: Радио и связь, 1985. – 272 с.

15. Левин, Л.С. Цифровые системы передачи информации [Текст] / Л.С. Левин, М.А. Плоткин. – М.: Радио и связь, 1982. – 216 с.

16. Ли, У.К. Техника подвижных систем связи. [Текст] / У.К. Ли. – М.: Радио и связь, 1985. – 390 с.

17. **Нефедов, В И**. Основы радиоэлектроники и связи: учебное пособие для вузов [Текст] / В. И. Нефедов, А. С. Сигов. — М.: Высшая школа, 2009. — 735 с.

18. Радиотехнические системы передачи информации [Текст] / В.А.Борисов, В.В. Калмыков, Я.М.Ковальчук. – М.:Радио и связь, 1990. – 304 с.

19. Нефедов, В.И. Общая теория связи : учебник для бакалавриата и магистратуры [Текст] / В.И.Нефедов, А.Г.Сигов. – М.: Юрайт, 2016. – 495 с.

20. Атражев, М.П. Борьба с радиоэлектронными средствами [Текст] / М.П. Атражев, В.А. Ильин, Н.П. Марьин. – М.: Воениздат, 1972. – 272 с.

21. Бородич, С.В. Искажения и помехи в многоканальных системах радиосвязи с частотной модуляцией [Текст] / С.В. Бородич. – М.: Связь, 1976. – 256 с.

22. Немировский, А.С. Борьба с замираниями при передаче аналоговых сигналов [Текст] . – М.: Радио и связь, 1984. – 208 с.

23. Полушин, П.А. Избыточность сигналов в радиосвязи [Текст] / П.А. Полушин, А.Г. Самойлов. – М.: Радиотехника, 2007. – 256 с.

24. Электромагнитная совместимость радиоэлектронных средств и непреднамеренные помехи: пер. с англ. [Текст] / Дональд Р.Ж. Уайт – М.: Сов. радио, 1977, Т. 1 – 348 с.; 1978, Т.2 – 272 с.; 1979, Т. 3 – 464 с.

25. Использование избыточности в системах передачи информации [Текст] / П.А. Полушин. – LAP LAMBERT Academic Publishing, Saarbrucken, Deuschland, 2011. – 341 p.

26. Морелос – Сарагоса, Р. Искусство помехоустойчивого кодирования. Методы, алгоритмы, применения: пер. с англ. [Текст] / Р. Морелос – Сарагоса. – М.: Техносфера, 2005. – 320 с.

27. Системы мобильной связи. Учебное пособие для вузов [Текст] / Под ред. В.П. Ипатова. – М.: Горячая линия–Телеком, 2003. – 272 с.

28. Сергиенко, А.Б. Цифровая обработка сигналов [Текст]/ А.Б. Сергиенко. – СПб.: Питер, 2003. – 604 с.

29. Рихтер, С.Г. Цифровое радиовещание [Текст] / С.Г. Рихтер. – М.: Горячая линия-Телеком, 2004 – 352 с.

30. Тяпичев, Г.А. Спутники и цифровая радиосвязь [Текст] / Г.А. Тяпичев. – М.: ТехБук, 2008. – 288 с.

31. Лей Э. Цифровая обработка сигналов для инженеров и технических специалистов; практическое руководство [Текст] / Э. Лей : пер. с англ.. – М.: ООО «Группа ИДТ», 2007. – 336 с.

32. Айчфишер, Э. Цифровая обработка сигналов: практический подход [Текст] / Э. Айчфишер, Б. Джервис: пер. с англ. – М.: Изд. дом «Вильямс», 2004. – 992 с.

33. Берлекамп Э.Р. Алгебраическая теория кодирования. Пер. с англ. [Текст] – М.: Мир, 1971. – 176 с.

34. Блейхут, Р. Теория и практика кодов, исправляющих ошибки: пер. с англ. [Текст] / Р.Блейхут. – М.: Мир, 1986. – 576 с.

35. Витерби А.Д. Принципы цифровой связи и кодирования [Текст] / А.Д. Витерби, Дж.К. Омура : пер. с англ. – М. : Радио и связь, 1982. – 536 с.

36. Князев, А.Д. Элементы теории и практики обеспечения электромагнитной совместимости радиоэлектронных средств [Текст] / А.Д. Князев. – М.: Радио и связь, 1984. – 336 с.

37. Коржик, В.И. Помехоустойчивое кодирование дискретных сообщений в каналах со случайной структурой [Текст] / В.И. Коржик, Л.М. Финк – М.: Связь, 1979.– 272 с.

38. Питерсон, У. Коды, исправляющие ошибки [Текст] / У. Питерсон, Э. Уэлдон: пер. с англ.; под ред. Р.Д. Добрушина и С.И. Самойленко. – М.: Мир, 1976. – 593 с.

39. Теория кодирования [Текст] / Т. Касами, И. Токура, Е. Ивадари : пер. с япон. под. ред. С.И. Гельфанда и Б.С. Цыбакова. – М.: Мир, 1978. – 576 с.

40. Финк, Л.М. Теория передачи дискретных сообщений [Текст] / Л.М.Финк. – М.: Советское радио, 1970. – 728 с.

41. Фано, Р. Статистическая теория связи [Текст] / Р.Фано: пер. с англ. – М.: Мир, 1965. – 375 с.

42. Форни, Д. Каскадные коды [Текст] / Д.Форни: пер. с англ. под ред. С.И. Самойленко.. – М.: Мир, 1970. – 207 с.

43. Шеннон, К. Работы по теории информации и кибернетике [Текст] / К.Шеннон : пер. с англ. – М.: Изд-во иностранной литературы, 1963. – 829 с.

44. Кларк, Дж., мл. Кодирование с исправлением ошибок в системах цифровой связи: пер. с англ.; под. редакцией Б.С. Цыбакова [Текст] / Дж. Кларк, мл., Дж. Кейн. – М.: Радио и связь, 1987. – 392 с.

45. Мак-Вильямс, Ф.Дж. Теория кодов, исправляющих ошибки. [Текст] / Ф.Дж.Мак-Вильямс, Н.Дж.А.Слоэн. – М.: Связь, 1979. – 307 с.

46. Самсонов, Б.Б. Теория информации и кодирования [Текст] / Б.Б. Самсонов, Е.М. Плохов, А.И.Филиненко, Т.В.Кречет . – Ростов н/Д, Феникс, 2002. – 267 с.

47. Квашенников, В.В. Адаптивное помехоустойчивое кодирование в технике связи [Текст] / В.В.Квашенников, А.Д.Кухарев. – Калуга: Изд-во научной литературы Н.Ф.Бочкаревой, 2007. – 147 с.

48. Марков А.А. Введение в теорию кодирования [Текст] /А.А.Марков. – М.: Наука, 1982. – 201 с.

49. Кудряшов, В.Б. Теория информации: Учебник для вузов [Текст] / В.Б. Кудряшов. – СПб, Питер, 2009. – 320 с.

50. Кловский, Д.Д. Передача дискретных сообщений по радиоканалам. [Текст] / Д.Д. Кловский. – М.: Радио и связь, 1982. – 304 с

51. Cain, J.B. Punctured convolutional codes of rate *(n-1)/n* and simplified maximum likelihood decoding. / J.B. Cain, G.C. Clark, J.M. Geist // IEEE Trans. Inf. Theory. 1979. - IT – 25 - P. 97-100.

52. Fano, R.M. A heuristic discussion of probabilistic decoding / R.M.Fano // IRE Trans. Inf. Theory. - 1963, vol. IT9, n.2. - P. 64-74.

53. Forney, G.D. Burst – correcting codes for the classic bursty channel. / G.D. Forney // IEEE Trans. Commun. Technol. – 1971. - vol. COM – 19, October. - P. 772 - 781.

54. Heller, J.A. Feedback decoding for convolutional codes / J.A.Heller // in advances in communication system, J.Viterbi (ed.) - New York : Academic, 1975. - vol.4.A

55. Lin, S. Error control coding. / S. Lin, D.J. Costello. – Englewood: Prentice – Hall, 1983.

56. Viterbi, A. Convolutional codes and an their performance in communication systems / A.J.Viterbi // IEEE Trans. Commun. Technol. – 1971. - vol. COM19, n.5, October. - P. 751-772.

57. Защита от радиопомех [Текст] / Под ред. М.В. Максимова. – М.: Сов. радио, 1976. – 496 с.

58. Папалекси, Н.Д. Радиопомехи и борьба с ними [Текст] / Н.Д. Папалекси. – М.: ОГИЗ, Государственное издательство технико-экономической литературы, 1942. – 187 с.

59. Гохберг, А.П. Режекция комплекса сосредоточения помех [Текст] / А.П. Гохберг // Радиотехника. – 1989. – №6. – С. 3–9.

60. Адаптивная компенсация помех в каналах связи [Текст] / Под ред. Ю.И. Лосева. – М.: Радио и связь, 1988. - 208 с.

61. Полушин, П.А. Воздействие сосредоточенных помех на системы передачи сигналов со сверточным кодированием [Текст] / П.А. Полушин, Д.В. Синицин, И. Джулани, Ж.Л. Гомес // Радиотехнические и телекоммуникационные системы.– 2014. – №3(15).– С. 69-73.

62. Никитин, О.Р. Арифмологический алгоритм сверточного кодирования цифровых сигналов при воздействии узкополосной помехи [Текст] / О.Р. Никитин, П.А. Полушин, Д.В. Синицин, И. Джулани // Вестник Рязанского гос. радиотехнического университета. – 2014. –№4 (вып. 50).– часть 1. – С. 45-50.

63. Корнеева, Н.Н. Возможности диагностики параметров сверточных кодов [Текст] / Н.Н.Корнеева, О.Р.Никитин // 11-я МНТК «Перспективные технологии в средствах передачи информации – ПТСПИ-2015» – Владимир–ВлГУ–2015. – С. 154-156.

64. Корнеева, Н.Н. Декодирование циклических кодов при неизвестной структуре кодера [Текст] / Н.Н.Корнеева, О.Р.Никитин // 11-я МНТК «Перспективные технологии в средствах передачи информации – ПТСПИ-2015» – Владимир–ВлГУ–2015 (СТРАНИЦЫ!) – С.156-158.

65. Корнеева, Н.Н. Разработка алгоритмов диагностики сверточных кодов / Н.Н. Корнеева, О.Р. Никитин // 12-я МНТК «Физика и радиоэлектроника в медицине и экологии (ФРЭМЭ 2016)».– Владимир-Суздаль, 5-7 июля 2016. – книга 1.– С. 354-356.

66. Корнеева, Н.Н. Разработка алгоритмов диагностики сверточных кодов [Текст] / Н.Н. Корнеева, О.Р. Никитин, П.А.Полушин // Радиотехнические и телекоммуникационные системы. – 2016. – №1. – С. 31-36.

67. Elias P. Coding for Noisy Channels // IRE Com. Rec., 1955. –vol. 3, pt. 4. – pp. 37-46.

68. Clark G., Cain J. Error Correction Coding for Digital Communication // Plenum Press, 1981. – pp. 21-26.

69. Lin S., Costello D.J. Error Control Coding: Fundamentals and Applications. // Prentice-Hall, 1983. – 101 p.

70. Ungerboeck, G. Trellis-Coded modulation with Redundant Signal Sets. // IEEE Communication Magazine. – part 1, part2, vol 25, February, 1987. – pp.5-21.

71. Ungerboeck, G. Channel Coding with Multilevel/Phase. // IEEE Trans. Inform.Theory. – vol. IT-28, January, 1982. – pp. 55-67.

72. Imai H., Hirakawa S. A New Multilevel Coding Method Using Error-Correcting Codes // IEEE Trans. Info. Theory. – vol IT-234, no. 5, May, 1977. – pp. 371-377.

73. Viterbi, A.J., Wolf, J.K., Zehavi E., Padovani R. A Pragmatic Approach to Trellis-Coded Modulation. // IEEE Comm. Mag., July, 1989. – pp. 11-19.

74. Zehavi, E., Wolf J.K. P2 Codes: Pragmatic Trellis Codes Utilizing Punctured Convolutional Codes // IEEE Communication Magazine, Feb., 1995. – pp. 26-36.

75. Brouwer, A.E., Verhoeff, T. // An Updated Table for for Minimum-Distance Bounds for Binary Linear Codes // IEEE Trans. Info Theory. Vol 39, no. 2, Mar., 1993. – pp. 662-677.

76. Schlegel, C. Trellis Codes. – IEEE Press, 1997. – 141p.

77. Forney, G.D. Concatenated Codes. – MIT Press Research Monograf 37/ – 1966. – 203 p.

78. Форни, Д.Д. Каскадные коды. – М.: Мир, 10970. – 212 с.

79. Блох, Э.Л., Зяблов, В.В. Линейные каскадные коды. – М.: Наука, 1982. – 134 с.

80. Блох, Э.Л., Зяблов, В.В. Обобщенные каскадные коды [Текст] / Э.Л. Блох, В.В. Зяблов – М.:Связь, 1976. – 134 с.

81. Zinoviev, V.A. Generalized Cascade Codes. //Probl. Pered. Inform. – vol.12, no. 1 , 1976. – pp. 5-15.

82. Berou C., Glavieux A., Thitimajshima P. New Shannon Limit Error-Cjrrecting Coding and Decoding: Turbo Codes. // IEEE Proceedings of the Int. Conf. on Communications, Geneva, Switzerland, May, 1993. – pp. 1064-1070.

83. Berou C., Glavieux A. Near Optimum Error Correcting Coding and Decoding: Turbo Codes. // Ieee Trans. On Communications. – vol. 4,no. 10, October, 1996. – pp.1261-1271.

84. Galager R.G. Low-Density Parity-Check Codes. – IRE Trans. Info. Theory. – vol.8, no. 1, Januaru, 1962. – pp. 21-28.

85. Галлагер, Р. Коды с малой полотностью проверки на четность [Текст] / Р.Галлагер. – М.: Мир, 1966. – 167 с.

86. Галлагер, Р. Теория информации и надежная связь [Текст] / Р. Галлагер. – М.: Советское радио, 1974. – 306 с.

87. Massey, J.L. Threshold Decoding. – MIT Press, 1963. – 278 p.

88. Мэсси, Дж. Пороговое декодирование [Текст] / Дж. Мэсси. – М.: Мир, 1966. – 287 с.

89. Берлекэмп, Э.Р. Техника кодирования с исправлением ошибок [Текст] / Э.Р. Берлекэмп // ТИИЭР. - 1980, т. 68. - № 5. – С. 24 -58.

90. Золотарев, В.В. Помехоустойчивое кодирование [Текст] / В.В. Золотарев, Г.В. Овечкин. – М.: Горячая линия-Телеком, 2004. – 126 с.

91. Золотарев, В.В. Алгоритмы многопорогового декодирования для гауссовых каналов [Текст] / В.В. Золотарев, Г.В. Овечкин // Информационные процессы. – т. 8, №1, 2008. – С. 68-83.

92. Зубарев, Ю.Б. Многопороговые декодеры для высокоскоростных спутниковых каналов связи: новые перспективы [Текст] / Ю.Б.Зубарев, В.В.Золотарев, Г.В.Овечкин, В.В.Строков // Электросвязь, №5, 2002. – С. 10-12.

93. Справочник по теории вероятности и математической статистике [Текст] / В.С. Королюк, Н.И. Портенко, А.В. Скороход, А.Ф. Турбин.– М.: Наука. Главная редакция физико-математической литературы, 1985. - 640 с.

94. Левин, Б.Р. Теоретические основы статистической радиотехники [Текст] / Б.Р. Левин. – М.: Сов. радио, 1974, Т. 1 – 552 с; 1975, Т. 2 –392 с.; 1976, Т. 3 – 288 с.

95. Тихонов, В.И. Статическая радиотехника. [Текст] / В.И.Тихонов. – М.: Сов. радио. 1966. – 678 с.

96. Справочник по математике (для научных работников и инженеров) [Текст] / Г.Корн, Т. Корн – М.: Наука, 1977. – 832 с.

97. Воеводин, В.В. Матрицы и вычисления [Текст] / В.В. Воеводин, Ю.А.Кузнецов. – М.: Наука. Главная редакция физико-математической лимтературы, 1984. – 320 с.

98. Денисенко, А.Н. Сигналы. Теоретическая радиотехника: Справочное пособие [Текст] / А.Н. Денисенко. – М.: Горячая линия–Телеком, 2005.. – 704 с.

99. Корнеева Н.Н., Полушин П.А., Никитин О.Р. Программный комплекс для исследования матричного метода диагностики сверточных кодов. //Свидетельство РФ о государственной регистрации программы для ЭВМ №2016610459: рег.12.01.2016.– Заявлено 17.11.2015., №2015661085.-Опубл.20.02.2016.

100. Корнеева Н.Н., Полушин П.А., Никитин О.Р. Программный комплекс для исследования алгоритмов диагностики циклических блоковых кодов. //Свидетельство РФ о государственной регистрации программы для ЭВМ №2016614189,– рег.18.04.2016. – Заявлено 29.02.2016.,№ 2016611632.-Опубл.20.05.2016.

101. Корнеева Н.Н., Полушин П.А., Никитин О.Р. Программный комплекс для исследования алгоритмов диагностики перфорированных сверточных кодов. //Свидетельство РФ о государственной регистрации программы для ЭВМ №2016618221,– рег. 25.07.2016. – Заявлено 06.06.2016., №2016615858.-Опубл.20.08.2016.

102. Корнеева, Н.Н. Диагностика циклических кодов [Текст] / Н.Н. Корнеева // 2-я МНТК «Наука и образование: проблемы и стратегии развития»-Уфа,15-16 ноября2016г.-С.181-184.

103. Korneeva, N.N. On choice of admissible number of the data transmission channels in dts with usage of a statisties of natural intervals of speech [Text] / Smorshevskiy V.S., Penzeev A.A., Korneeva N.N. // «Proceedings TEIC Actual problems of telecom Part2» - Antwerp-1999. -P.49-51.

104. Корнеева, Н.Н. Алгоритм диагностики циклических кодов на основе непосредственного вычисления простых полиномов [Текст] / Н.Н. Корнеева, О.Р. Никитин, П.А.Полушин // Радиотехнические и телекоммуникационные системы. – 2017. – №1. – С. 62-68.

105. Заявка на изобретение № 2015151382,дата подачи заявки 30.11.2015, Корнеева Н.Н., Полушин П.А., Никитин О.Р., Способ диагностики сверточных кодов.Решение о выдаче патента 19.01.2017.

«УТВЕРЖДАЮ»
Первый проректор, проректор по
научной и инновационной работе ВлГУ
_____В.Г.Прокошев
«07» 04 2017г.

АКТ ВНЕДРЕНИЯ

результатов диссертационной работы Корнеевой Натальи Николаевны на тему «Исследование и разработка алгоритмов диагностики кодированных цифровых сигналов».

Настоящий акт составлен о том, что материалы диссертационной работы Корнеевой Н.Н. внедрены в учебный процесс на кафедре радиотехники и радиосистем ВлГУ по направлениям магистратуры 11.04.01 «Радиотехника» и бакалавриата 11.03.02 «Инфокоммуникационные технологии и системы связи» используются в дисциплинах:

- Методы и устройства приема сигналов
- Современные системы подвижной связи
- Современные радиоэлектронные системы

Заведующий кафедрой
радиотехники и радиосистем О.Р. Никитин

«УТВЕРЖДАЮ»

Заместитель генерального директора-

главный инженер

ОАО «Владимирский завод «Электроприбор»

_____В.А.Павловский

«07» «04»2017

АКТ ВНЕДРЕНИЯ

результатов диссертационной работы Корнеевой Натальи Николаевны на тему «Исследование и разработка алгоритмов диагностики кодированных цифровых сигналов».

Настоящий акт подтверждает, что ОАО «Владимирский завод «Электроприбор» использует разработанные в диссертационной работе Корнеевой Н.Н. методы и алгоритмы обработки радиосигналов при создании новой радиоэлектронной аппаратуры для передачи и приема информации. Они позволяют улучшить помехоустойчивость и другие качественные показатели данной аппаратуры.

Заместитель главного инженера

по новой технике

ОАО «Владимирский завод «Электроприбор» _____ А.А. Илюхин