Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский физико-технический институт (государственный университет)»

Факультет радиотехники и кибернетики
Кафедра теоретической и прикладной информатики

Направление подготовки/специальность: 03.04.01 Прикладные математика и физика
Направленность (профиль) подготовки: Математические и информационные технологии
Форма обучения: очная

Допущен к ГИА
Руководитель учебного подразделения
_____/_____
(подпись) *расшифровка подписи*
«____» _____20____г.

Исследование эффективности алгоритмов акустического эхоподавления

(магистерская диссертация)

Обучающийся: Кайда Александр Васильевич

_____
(подпись обучающегося)

Научный руководитель: Пашков Руслан Анатольевич, к.ф.-м.н.

_____
(подпись научного руководителя)

Москва 2018

Аннотация

В данной работе исследуются алгоритмы акустического эхоподавления с точки зрения производительности и эффективности. В частности, адаптивные фильтры в частотной области на основе алгоритма наименьших средних квадратов (LMS FDAF). Основной целью является проверка предположения о целесообразности использования эхоподавления в составе гипервизорного виртуализационного решения.

Оглавление

1. Введение 4
   1.1. Особенности виртуальной звуковой подсистемы . . . 9
   1.2. Постановка задачи 11
2. Решение 13
   2.1. Алгоритмы эхоподавления 13
      2.1.1. LMS 16
      2.1.2. FDAF 18
      2.1.3. Многоблочный FDAF 21
   2.2. Проблемы алгоритмов 24
      2.2.1. Двойной разговор 24
      2.2.2. Рассинхронизация 26
   2.3. Анализ эффективности эхоподавления 28
   2.4. Требования к производительности 30
   2.5. Реализация 32
3. Тестирование и результаты 35
   3.1. Тестовый стенд 35
   3.2. Производительность 37
   3.3. Эффективность эхоподавления 39
4. Заключение 48

Список литературы 49

1. Введение

Виртуализация компьютерных систем является важной и востребованной областью информационных технологий. В наше время активно развивается и используется множество виртуализационных решений. Использование таких решений позволяет разделить компьютерные ресурсы между несколькими изолированными экземплярами операционных систем. Такими ресурсами могут выступать процессорное время, память, диск или другие устройства.

Следует выделить два основных типа виртуализации компьютера: уровня платформы и уровня операционной системы (контейнерная виртуализации). Рассмотрим виртуализацию уровня платформы, основанную на гипервизоре.

В дальнейшем будет использоваться такое понятие как виртуальная машина. Под этим следует понимать окружение, которое с точки зрения гостевой ОС является аппаратной платформой. Но на самом деле этот эффект достигается эмуляцией оборудования, выполняемой специальным програмным комплексом - гипервизором. Эмуляция должна достаточно точно имитировать поведение оборудования для нормального функционирования гостевой ОС и являться совершенно прозрачной.

Гипервизор — это программа, позволяющая на одном хост-компьютере запускать несколько виртуальных машин, в которых работают операционные системы, называемые гостевыми. Название это возникло из того, что по уровню привилегий гипервизор находится на уровень выше супервизора (операционной системы, управляющей запуском обычных пользовательских приложений).

Гипервизор предоставляет гостевым операционным системам вирту-

альную аппаратную платформу, а так же обеспечивает изоляцию операционных систем друг от друга, защиту и безопасность, разделение и управление ресурсами компьютера. Гипервизор также может предоставлять средства связи между гостевыми операционными системами, например внутреннюю компьютерную сеть.

Гипервизоры разделяют на два типа: запускаемые непосредственно на аппаратуре как операционные системы (“bare metal”) и запускаемые под обычной операционной системой в качестве пользовательского приложения с особыми привилегиями. Примером гипервизора первого типа является VMware ESX Server, второго — Parallels Desktop. При использовании гипервизора второго типа низкоуровневое взаимодействие с устройствами выполняется хостовой операционной системой.

Проиллюстрируем работу системы виртуализации, основанной на гипервизоре второго типа.

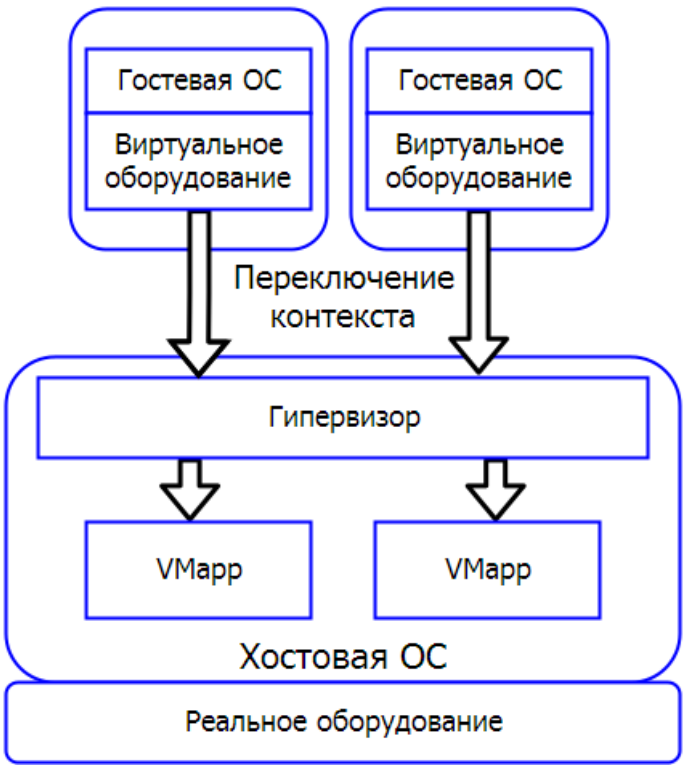

Рис. 1. Виртуализая на гипервизоре второго типа

Как было сказано, одной из главных задач гипервизора является виртуализации различных устройств компьютера. Для пользователя имеет значение, что бы гостевая ОС имела доступ к устройствам реального хост компьютера и поддерживала аналогичный функционал.

Устройства компьютерной платформы можно разделить на несколько классов: системные и периферию. К системным устройствам, без которых невозможно функционирование виртуальных машин с современными ОС, можно отнести жёсткий диск, ACPI, таймера, контроллеры прерываний и другие. К периферийным устройствам следует отнести мышь, клавиатуру, сетевую и звуковую карту, графический ускоритель и т.д.

Одним из важнейших мультимедийных устройств для любого компьютера является звуковая карта. Она служит для ввода и вывода звуковых потоков через устройства записи и воспроизведения соответственно. Звуковые карты бывают внутренние, встроенные в компьютер, и внешние, работающие по протоколам USB, беспроводной связи и другим. Иногда в состав звуковых карт включают аппаратные устройства обработки сигналов, которые улучшают качество звука: удаляют эхо, нежелательные шумы, производят эквализацию звука. Как правило, это устройства высокого класса и ценового сегмента, которые называют HI-FI. Эмуляция звуковой карты является одной из задач гипервизора виртуальных машин.

В то же время высокое распространение получают системы для конференций, громкой связи, звуковых и видео коммуникаций. Такие системы часто реализованы в виде программного обеспечения для персональных компьютеров. Значительной проблемой для таких систем является возникновение эха, которое создается громкоговорителями (колонками) и записывается микрофоном. Такое явление получило название возвратного эха. Его наличие значительно ухудшает качество записи полезного звукового сигнала и может приводить к крайне неприятному эффекту кольцевого

эха.

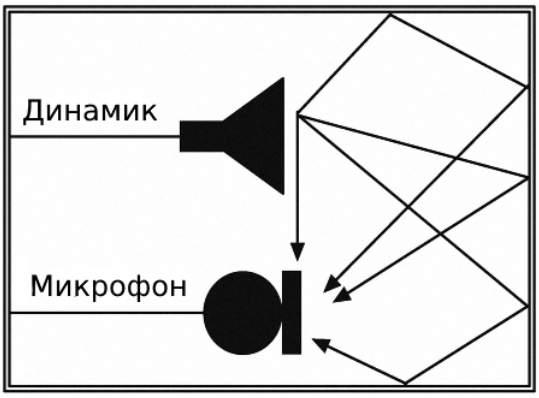

Рис. 2. Возникновение эха

Отметим непростой характер возникающего акустического эха. В частности, эхо состоит из многократно отраженного исходного сигнала, который приходит с различными задержками, с различными амплитудами и испытывает разнородные искажения. Путь, пройденный звуковой волной в воздушной среде от источника к приемнику будем называть эхо путем. Как видно, каждый эхо путь имеет различную длину, характеристику затухания и искажения исходного эхо сигнала. Эти факторы существенно усложняют эхоподавление.

Задача устранения эффекта акустического эха имеет большое значение и может быть решена на разных уровнях:

- Использование независимых источника и приемника звука (гарнитура, наушники);
- Подавления эха программно на уровне конкретного ПО;
- Подавление эха на аппаратном уровне;
- Подавление эха на уровне виртуального звукового устройства.

Разберем представленные решения подробнее. Системы коммуникации без использования громкой связи, например, с наушниками, являют-

ся, безусловно, самым надежным и качественным решением. Однако, это решение имеет частный характер, и не всегда удовлетворяет потребностям пользователей. Поэтому необходимо удалять эхо сигнал из записанного звукового потока, если он там присутствует. Это позволит полноценно использовать громкую связь в случае такой необходимости.

К сожалению, не каждое прикладное ПО заботится о качестве передаваемого звука. Более того, даже операционные системы не всегда обеспечивают качественную обработку звука и удаление эхо сигнала. Более универсальным решением является подавление эха на аппаратном уровне компьютера, в случае, если в звуковом устройстве есть такая возможность. Однако эта возможность ограничивается рамками конкретного звукового устройства: в случае использования нескольких различных устройств для ввода и вывода такое решение работать не может.

Логичным выходом из ситуации может быть использование виртуальной аппаратной платформы для эхоподавления. Это решение не зависит от конкретных звуковых устройств, способов ввода и вывода звука и от аппаратной конфигурации хост системы, обеспечивая виртуальные машины качественным звуком. Отметим также и то, что такое решение является абсолютно прозрачным для гостевой системы.

Таким образом, перед гипервизором виртуальных машин ставится задача не только эмуляции звуковой карты и передачи сигнала с реального звукового устройства, но и обработки этого сигнала для обеспечения его высокого качества. При этом гостевая система и все ПО внутри неё не будет страдать от эха и других негативных звуковых эффектов.

1.1. Особенности виртуальной звуковой подсистемы

Гипервизор виртуальных машин использует эмуляцию звуковой карты для ввода/вывода звука. Обработка звуковых потоков должна осуществляться на этапе транспорта между API звуковой подсистемы хостовой ОС и эмулированной звуковой картой. С точки зрения виртуальной машины, это является аппаратным решением, к которому нет доступа программно. Обратимся к диаграмме виртуализации.

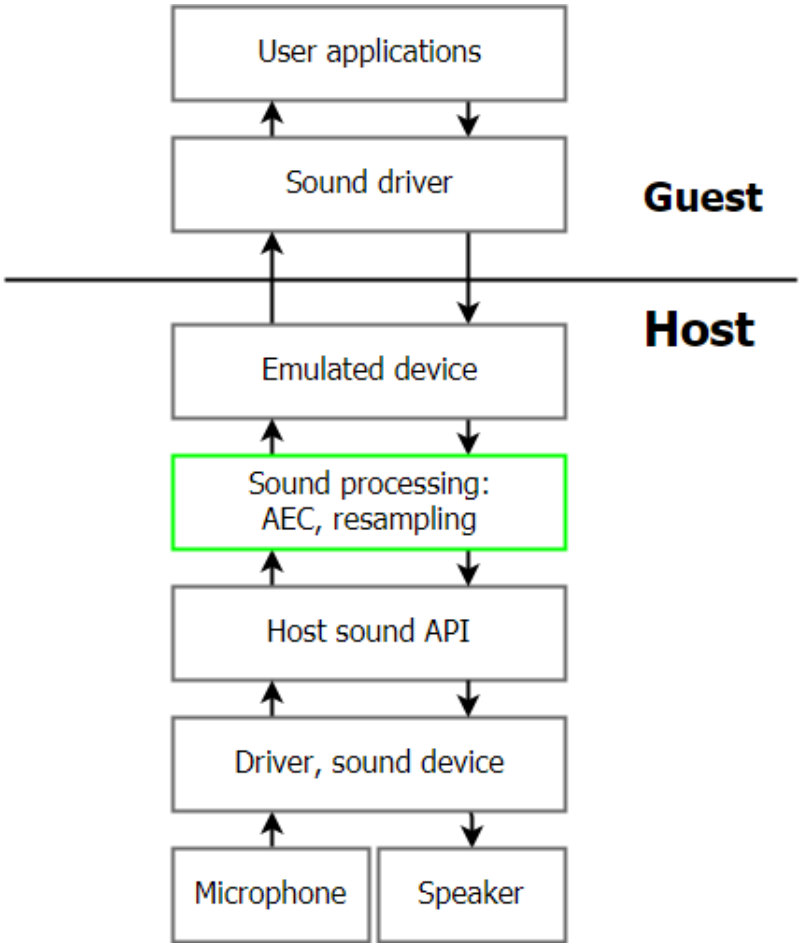

Рис. 3. Схема виртуализации звука

Как видно из диаграммы, обработка записанного микрофоном звукового потока должна происходить на этапе между получением этого потока от хост API и передачей его эмулированной звуковой карте. При этом задержка, вносимая механизмом эхоподавления, увеличивает общую задержку получения записанного потока гостевой системой. Существенной зада-

чей является минимизация добавленной задержки для обеспечения работы аудио подсистемы в реальном времени.

Заметим также то, что в данной работе рассматривается подавление возвратного эха, то есть такого эхо сигнала, прообраз которого был ранее подан гостевой системой на воспроизведение через громкоговорители.

В качестве основных требований к алгоритму эхоподавления можно выделить:

- Работа в режиме реального времени;
- Низкое потребление ресурсов;
- Низкая задержка (latency);
- Быстрая адаптация к меняющемуся окружению;
- Высокая эффективность вне зависимости от характера звукового потока (речь/музыка/посторонние звуки).

Сложностью в данной схеме является отсутствие информации о назначении звуковых потоков, которые исходят и поступают в гостевую систему через эмулированную звуковую карту. Это может быть как конференция в реальном времени, так и прослушивание окружающего фона голосовыми помощниками во время простоя компьютера. В первом случае приоритетом обладает качественное подавление эха, а вот во втором — высокая нагрузка на хост систему совершенно неприемлема. Таким образом необходимо минимизировать нагрузку от применения эхоподавления.

Другой проблемой является свободный выбор параметров звуковых потоков гостевой системой. Вследствие этого потоки записи и воспроизве-

дения на этапе эмуляции звуковой карты могут иметь различную частоту дискретизации и формат данных, что потребует предварительной передискретизации и изменения формата на промежуточный, подходящий для обработки. Т.к. реальное звуковое оборудование может работать с параметрами, отличными от промежуточных, для ввода и вывода звука может потребоваться ещё один этап передискретизации и изменения формата звукового потока.

Важным требованием для всех алгоритмов эхоподавления является строгая синхронизация входящего и исходящего звуковых потоков. В самом деле, рассинхронизация всего на один отсчет приведет к рассогласованию коэффициентов фильтра. Адаптивный алгоритм с этим сможет справиться, но отставание или потеря одного полного кадра воспроизведения может привести к полному сбросу фильтра. Синхронизация потоков на уровне эмуляции звуковой карты и гостевого времени не составляет труда, однако управлять синхронизацией хостовых источников звука чаще всего невозможно, т.к. это самостоятельные устройства со своими источниками тактовых сигналов.

Таким образом, крайне желательно что бы на хостовой ОС ввод и вывод звука осуществлялся через звуковые устройства с одним и тем же источником времени, что не всегда выполняется для, например, внешних звуковых карт. В противном случае необходимо проводить дополнительные действия для поддержания принудительной синхронизации, что требует дополнительных вычислительных ресурсов.

1.2. Постановка задачи

Акустическое эхоподавление является важным этапом обработки звука в коммуникационных системах и может быть реализовано на уровне

виртуального звукового устройства в составе гипервизора виртуальных машин. Однако работа эхоподавления на программном уровне является вычислительно сложной операцией, которая может существенно нагрузить хост компьютер. Важно выбрать наиболее оптимальный с точки зрения производительности подход.

Отметим так же то, что подробного изучения требует и эффективность эхоподавления в реальной системе, учитывая возможные сложности.

Исходя из сказанного выше, перед данной работой ставится задача исследования эффективности и производительности алгоритмов акустического эхоподавления, работающих в реальном времени.

2. Решение

2.1. Алгоритмы эхоподавления

Основным требованием к алгоритму эхоподавления является самостоятельная адаптация к конкретной звуковой системе. В реальном применении конфигурация эхо путей неизвестна и может активно меняться во времени. Для таких задач существует класс алгоритмов из области цифровой обработки сигналов, получивший название адаптивных фильтров. Фильтром в широком смысле слова называется система, которая выделяет или улучшает определенную информацию, которая содержится в сигнале.

Цифровые адаптивные фильтры очень популярны в настоящее время благодаря их высокой эффективности при обработке сигналов и обучаемой природе. Обучаемость достигается благодаря наличию отрицательной обратной связи через сигнал ошибки. Обычно адаптивный фильтр имеет в своем составе определенный алгоритм автоподстройки, который изменяет коэффициенты фильтра на основании сигнала ошибки. Состояние фильтра задается через некоторый вектор коэффициентов $\mathbf{w}(n)$.

Разработано большое количество адаптивных фильтров, все они имеют различные достоинства и недостатки. Важно отметить, что области применения таких фильтров так же значительно различаются.

Приведем схематическое представление звуковой системы, в которой адаптивный фильтр удаляет возвратное эхо. На Рис. 4 видно, что цифровой сигнал $x(n)$ проигрывается динамиками и проходя по эхо пути подвергается воздействию неизвестного преобразования $\mathbf{w}'(n)$. Это происходит под влиянием акустики конкретного помещения. В дальнейшем этот звуковой сигнал смешивается с полезным сигналом разговора и шумом $v(n)$. В таком виде сигнал попадает в микрофон и происходит его оцифровка в

соответствующем звуковом устройстве. Результирующий цифровой сигнал обозначим как $d(n)$.

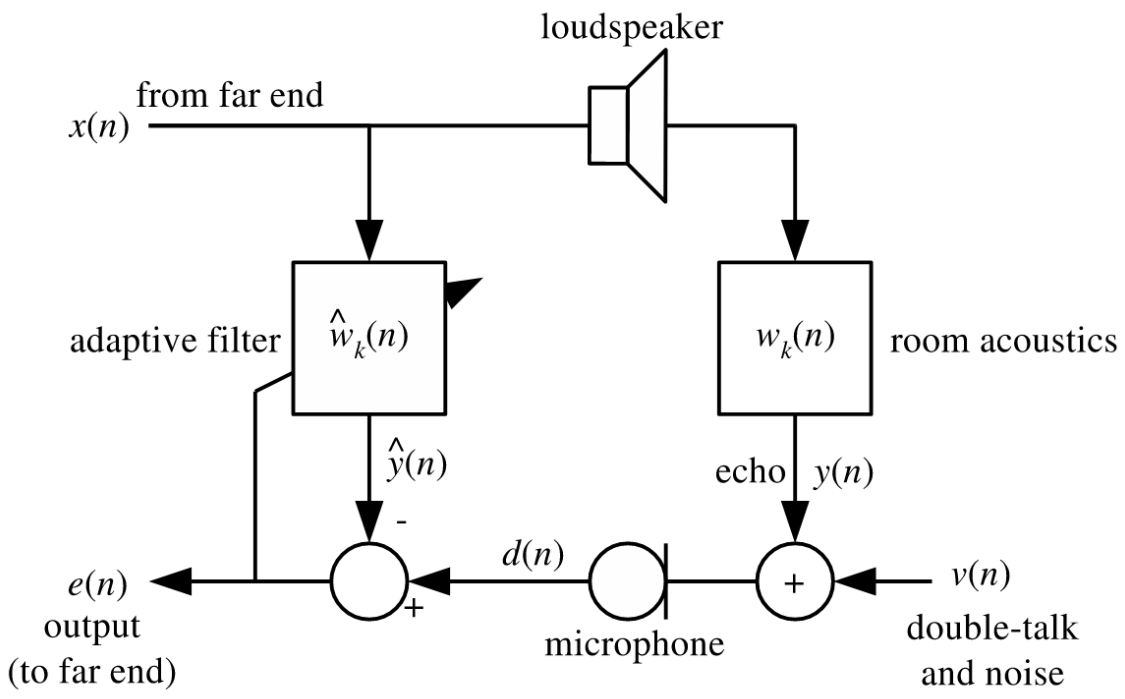

Рис. 4. Схема эхоподавления адаптивным фильтром

Процесс работы адаптивного фильтра является итеративным, на каждой из итераций происходят два основных действия: фильтрация эхо сигнала и адаптация коэффициентов фильтра $\mathbf{w}(n)$ с помощью алгоритма автоподстройки.

Первый этап работы фильтра на очередной итерации — предсказание эхо сигнала $y(n)$ в данный момент времени (отсчет под номером n). Это предсказание основано на хранимой истории $\mathbf{x}(n)$ сигнала, кореллирующего с эхом. В данном случае это сигнал из динамиков. Длина данного вектора обозначается как N и называется длиной фильтра.

$$\mathbf{x}(n) = [x(n), x(n-1), \ldots, x(n-N+1)]^T,$$

где обозначение $(.)^T$ является операцией транспонирования.

Вычитая предсказанное эхо из записанного микрофоном сигнала $d(n)$, получаем сигнал ошибки на данном отсчете:

$$e(n) = d(n) - y(n) = v(n) + \Delta y(n)$$

Этот процесс и является фильтрацией нежелательного сигнала. Вычисленный сигнал ошибки содержит в себе как полезный сигнал $v(n)$, так и $\Delta y(n) = y'(n) - y(n)$, то есть разницу между настоящим эхом и вычисленным фильтром.

Обратим внимание на то, что сигнал ошибки является результатом работы фильтра, в дальнейшем поступает к потребителю и по сути является сигналом с подавленным возвратным эхом.

Второй этап работы фильтра заключается в адаптации его коэффициентов на основании алгоритма автоподстройки. Для этого сигнал ошибки $e(n)$ поступает на вход алгоритма. Это служит отрицательной обратной связью, так как основной задачей фильтра с математической точки зрения является минимизация $|\Delta y(n)|$, а значит и ошибки $|e(n)|$. Выразим это через среднеквадратичную ошибку $E\{e^2(n)\}$, которая будет критерием эффективности фильтра.

Существует два основных адаптивных алгоритма: алгоритм наименьших средних квадратов (LMS - Least Mean Square) и алгоритм рекурсивных наименьших квадратов (RLS - Recursive Least Squares). В данной работе будем рассматривать LMS, т.к. RLS имеет значительно большую вычислительную сложность, хотя и обладает лучшей сходимостью.

2.1.1. LMS

Алгоритм наименьших средних квадратов (LMS) - стохастический алгоритм градиентного спуска, наиболее часто применяемый алгоритм адаптивной фильтрации. Его целевым критерием является минимизация мгновенной квадратичной ошибки $e^2(n)$. Адаптивный фильтр на основе LMS алгоритма реализуется с помощью следующих уравнений:

$$y(n) = \mathbf{w}^T(n)\mathbf{x}(n)$$

$$e(n) = d(n) - y(n)$$

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu e(n)\mathbf{x}(n),$$

где $\mathbf{w}(n)$ — это вектор весов фильтра длиной N, рекуррентно обновляемый на каждом шаге

$\mathbf{x}(n)$ — это вектор, состоящий из последних N отсчетов дальнего сигнала (который создает эхо),

μ — это коэффициент шага автоподстройки, который влияет на скорость схождения и его точность.

Отметим, что данный фильтр работает во временной области - вычисляет эхо для каждого отсчета отдельно, и по сути является фильтром с конечной импульсной характеристикой (КИХ-фильтр). Так же следует заметить, что обновление коэффициентов происходит после их использования на данном шаге и является рекурсивным процессом.

Устойчивость алгоритма сильно зависит от величины шага автоподстройки μ. В самом деле - малый шаг приведет к чрезмерно долгой автоподстройке, а слишком большой - к расходимости фильтра. Для сходимости шаг выбирается следующим образом:

$$0 < \mu < \frac{2}{\lambda_{max}},$$

где λ_{max} это максимальное собственное значение матрицы автокорелляции входного сигнала $x(n)$.

Необходимо заметить, что сила звукового сигнала $\mathbf{x}(n)$ значительно влияет на второе слагаемое в уравнении автоподстройки, и шаг автоподстройки должен быть обратно пропорционален мощности этого сигнала. Соответственно мы можем улучшить алгоритм LMS введением переменного коэффициента нормализации шага, принимая в расчет мощность входного сигнала. Полученный алгоритм получил название Normalized LMS и обладает лучшими характеристиками устойчивости при сравнимой вычислительной сложности. По этой причине он используется очень широко.

В случае NLMS уравнение автоподстройки принимает следующий вид:

$$\mathbf{w}(n+1) = \mathbf{w}(n) + \mu \frac{\mathbf{x}(n)}{\mathbf{x}^T(n)\mathbf{x}(n) + \epsilon} e(n),$$

где размер шага теперь $0 < \mu < 2$ и ϵ это малое слагаемое для предотвращения ошибки деления на ноль при нулевом входном векторе.

Заметим, что основные преимущества алгоритма NLMS над его предшественником проявляются при фильтрации нестационарных сигналов, таких как речь, где мощность звукового потока варьируется в широких пределах. В таких случаях достигается значительно более быстрая сходимость фильтра к оптимальному состоянию, в котором среднеквадратичная ошибка аналогична.

Существенным недостатком описанного выше фильтра является высокая вычислительная сложность, составляющая $O(N)$ умножений и сложений на каждый дискретный звуковой отсчет. Для акустического эхоподавления в режиме реального времени необходима большая длина фильтра N, составляющая сотни миллисекунд (тысячи отсчетов). Цифровые адап-

тивные фильтры, основанные на временных рядах оказываются излишне ресурсозатратными в таких случаях.

Остановимся подробнее на необходимой длине фильтра. Эта величина должна соответствовать максимально возможной длине эхо пути в реальном применении. В то же время эхо путь можно разделить на две составляющие:

- время прохождения колебаний воздушной среды через комнату или другое пространство от динамиков к микрофону
- время обработки сигнала звуковой картой, драйверами и другими программными прослойками хостовой операционной системы.

Оценим максимальное расстояние, которое проходит звуковой сигнал в комнате величиной 20 метров. Это соответствует габаритам большой комнаты. Время прохождения волны при этом составляет до 60мс. Время обработки звукового сигнала компьютером оценим сверху в 100мс[1]. Результирующая полная длина эхо пути составляет 160мс и может быть округлена с запасом до 200мс.

2.1.2. FDAF

Приведенный выше LMS фильтр алгоритмически представляет собой линейную свертку во временной области (фильтрация) и линейную корелляцию (обучение фильтра). По теореме о свертке, перейдя к круговой свертке можно значительно уменьшить вычислительную сложность алгоритма. Круговая свертка производится над блоками дискретных отсчетов в частотной области, переход в которую происходит с помощью дискретного

[1] Время на вывод и ввод звука через внешнюю звуковую карту

преобразования Фурье. ДПФ преобразует N вещественных отсчетов в N комплексных коэффициентов, представляющих исходный сигнал в виде гармоник. Общий вид преобразования:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N} kn} \qquad (k = 0, \ldots, N-1).$$

Как известно, алгоритм ДПФ обладает вычислительной сложностью $O(N^2)$. В данной работе будет использоваться быстрое преобразование Фурье, которое применяется повсеместно в цифровой обработке сигналов. Напомним, БПФ является вычислительно эффективной вариацией дискретного преобразования Фурье и обладает асимтотической сложностью $O(N \log(n))$ с размерами блока, которые являются степенями числа 2: $N = 2^l$. В противном случае его эффективность падает до $O(N(p_1 + \cdots + p_n))$ действий при размере блока $N = p_1 p_2 \cdots p_n$. В данной работе ограничимся выбором размеров блоков степенями двойки для достижения максимальной производительности.

Описанный выше фильтр работает поблочно над входными сигналами, преобразованными в частотную область, и носит название Frequency Domain Adaptive Filter (FDAF). Алгоритм автоподстройки - тот же алгоритм наименьших средних квадратов (LMS) или его вариации с переменным шагом μ. Его целевым критерием в частотной области также является минимизация мгновенной квадратичной ошибки. Такой тип фильтров хорошо себя зарекомендовал и используется в реальных системах.

К сожалению, использование круговой свертки и круговой корреляции приводит к возникновению дополнительных ошибок. Для борьбы с ними применяются сложные техники наложения входных сигналов, которые приводят к корректным результатам.

Разберем алгоритм работы фильтра более подробно. Вектора входных данных $\mathbf{x}(j), \mathbf{d}(j)$ имеют длину N, а обозначение FFT представляет из себя БПФ длины 2N.

Здесь и далее будем использовать обозначение $\mathbf{I}_N, \mathbf{0}_N$ - единичная и нулевая матрица размера N.

На итерации j происходит вычисление вектора сигнала ошибки $\mathbf{e}(j)$ и его частотного образа:

$$\mathbf{X}(j) = diag\{FFT[\mathbf{x}(j-1), \mathbf{x}(j)]\}$$

$$\mathbf{y}(j) = [\mathbf{0}_N, \mathbf{I}_N] FFT^{-1}[\mathbf{X}(j)\mathbf{W}(j)]$$

$$\mathbf{e}(j) = \mathbf{d}(j) - \mathbf{y}(j)$$

$$\mathbf{E}(j) = FFT[\mathbf{0}_N, \mathbf{e}(j)_N]$$

Заметим, что прямое и обратное БПФ всегда выполняется над двойными блоками фильтра. Обновление вектора весов длины 2N с помощью алгоритма LMS происходит следующим образом:

$$\mathbf{W}(j+1) = \mathbf{W}(j) + \mu * FFT(\mathbf{G}^{10} FFT^{-1}[\mathbf{X}^*(j)\mathbf{E}(j)]),$$

где $\mathbf{X}^*(j)$ - комплексно сопряжённый вектор, а $\mathbf{G}^{10} = \begin{bmatrix} \mathbf{I}_N & \mathbf{0}_N \\ \mathbf{0}_N & \mathbf{0}_N \end{bmatrix}$

Операция обратного и прямого преобразования Фурье в последней формуле является градиентным ограничением и производится для соответствия алгоритму линейной корреляции. В случае экономии на указанных двух БПФ алгоритм называется неограниченным (unconstrained) FDAF. К сожалению, при таком упрощении значительно ухудшается сходимость

алгоритма. Проводя аналогию с адаптивным фильтром во временной области это соответствует обновлению весов не на каждом отсчете, а на каждом блоке в N отсчетов.

Данный фильтр имеет хорошую сходимость и значительно меньшую вычислительную сложность чем обыкновенный адаптивный фильтр на основе LMS алгоритма. Асимптотическая сложность алгоритма составляет $O(\log(N))$ умножений в пересчёте на один обработанный отсчет звукового сигнала.

Такой тип адаптивных фильтров получил широкое распространение как в программной, так и в аппаратной реализации.

Отметим и основной недостаток FDAF фильтра: размер блока N должен быть равным длине фильтра, а длина фильтра - это максимально возможное время запаздывания эха. Отсюда следует что задержка фильтра составляет сотни миллисекунд, что недопустимо долго для использования в реальном времени. Необходимо найти способ существенного снижения задержки фильтра до приемлемой.

2.1.3. Многоблочный FDAF

Для решения проблемы большой задержки предложен фильтр, разбивающий большой блок на M меньших частей, каждая из которых имеет длину N отсчетов. При этом на каждой итерации работы фильтра обрабатывается только один блок входных данных. Такой фильтр получил название Partitioning Block FDAF (PBFDAF), или Multidelay Block FDAF (MDF). В данном фильтре размер блока может варьироваться в широких пределах при фиксированной общей длине фильтра и это позволяет выбрать приемлемую задержку обработки при низкой вычислительной сложности.

Опишем работу фильтра подробнее. В данном случае мы имеем M векторов $\mathbf{X}$ и $\mathbf{W}$, которые представляют исходный эхо сигнал и вектор весов в частотной области поблочно. На каждой итерации j старший $\mathbf{X}$ задается в виде матрицы таким образом:

$$\mathbf{X}_M(j) = diag\{FFT[\mathbf{x}(j-1), \mathbf{x}(j)]\}$$

При этом остальные блоки эхо сигнала смещаются на один, не требуя вычислений:

$$\mathbf{X}_m(j) = \mathbf{X}_{m+1}(j-1) \qquad (m = 1, \ldots, M-1).$$

Фильтрация осуществляется с помощью смешения линейной и круговой свертки, что является главной особенностью этого фильтра:

$$\mathbf{y}(j) = [\mathbf{0}_N, \mathbf{I}_N] FFT^{-1} \sum_{m=1}^{M} \mathbf{X}_m(j) \mathbf{W}_m(j)$$
$$\mathbf{e}(j) = \mathbf{d}(j) - \mathbf{y}(j)$$

Сигнал ошибки также переводим в частотную область:

$$\mathbf{E}(j) = FFT[\mathbf{0}_N, \mathbf{e}(j)_N]$$

Обновление весов фильтра $\mathbf{W}_m(j)$ происходит аналогично FDAF фильтру над каждым из M векторов:

$$\mathbf{W}_m(j+1) = \mathbf{W}_m(j) + \mu * FFT(\mathbf{G}^{10} FFT^{-1}[\mathbf{X}_m^*(j)\mathbf{E}(j)])$$

Важной оптимизацией в последней формуле по сравнению с FDAF является применение двух БПФ с целью градиентного ограничения только на старшем блоке $m = M$. Это позволяет сэкономить $2(M-1)$ БПФ, что довольно незначительно сказывается на конечной эффективности алгоритма но даёт существенный прирост производительности. Полный отказ от градиентного ограничения приводит к существенному ухудшению сходимости фильтра и в таком случае фильтра носит название неограниченного (unconstrained) MDF.

Таким образом, с помощью многоблочного подхода достигается существенное снижение задержки фильтра при обработке звукового потока при сохранении эффективности исходного метода.

Вычислительная сложность такого алгоритма составляет $O(\log_2(N)+M)$ в пересчете на один обработанный дискретный звуковой отсчет. Этот показатель аналогичен сложности FDAF фильтра при использовании небольших M. Однако главное преимущество данного фильтра - малая задержка фильтра при обработке звукового потока.

Отметим требования к размеру блока N в многоблочном фильтре. Возвращаясь к свойствам прямого ДПФ, следует заметить, что дискретные отсчеты в частотной области описывают гармоники с частотами $\frac{k}{T}$, где $T = \frac{N}{f}$ - период блока ДПФ. Соответственно для качественной фильтрации низкочастотного ($\approx$ 20Гц) эха необходимо, что бы размер ДПФ соответствовал периоду в 50мс. Так как преобразование Фурье происходит над двойными блоками, то размер блока фильтра, а значит и минимальная задержка, должны составлять 25мс. Это соответствует размеру блока $N = 1100$ отсчетов. Ввиду необходимости использования степеней двойки в качестве размера блока целесообразно принять $N = 1024$ отсчета. Полученное значение вполне подходит для удаления эха в звуковом потоке виртуальных машин в режиме реального времени.

2.2. Проблемы алгоритмов

2.2.1. Двойной разговор

Описанные адаптивные фильтры на основе LMS алгоритма имеют общее требование: для достижения высокого качества подавления эха полезный сигнал не должен коррелировать с подавляемым эхо сигналом. К сожалению, это не всегда так.

Часто встречаемый пример — когда полезный сигнал содержит речь одного человека, а эхо состоит из разговора другого человека. Такое происходит при общении людей средствами связи. В данном случае, несмотря на различие конкретного произносимого текста и разный тембр голоса, возникает некоторая корреляция между этими сигналами. Это приводит к возможной расходимости алгоритмов автоподстройки фильтра и замедляет обучение.

Такой тип сигналов получил название "двойной разговор"(double-talk) несмотря на то, что эта проблема не ограничивается человеческой речью. Любой сигнал, подверженный взаимной корреляции приводит к ухудшению эффективности адаптивного фильтра.

Приведем пример подобного сигнала на Рис. 5. В данном случае представлены монологи двух мужчин. Видно, что сигналы имеют подобное спектральное распределение и имеют общие шаблоны.

В настоящий момент предложено множество способов борьбы с указанным явлением. Часть из них строится на попытке сторонними средствами распознать двойную речь в конкретный момент времени и приостановить обучение адаптивного фильтра. С помощью такого метода можно предотвратить расхождение фильтра, однако обучение значительно замедляется.

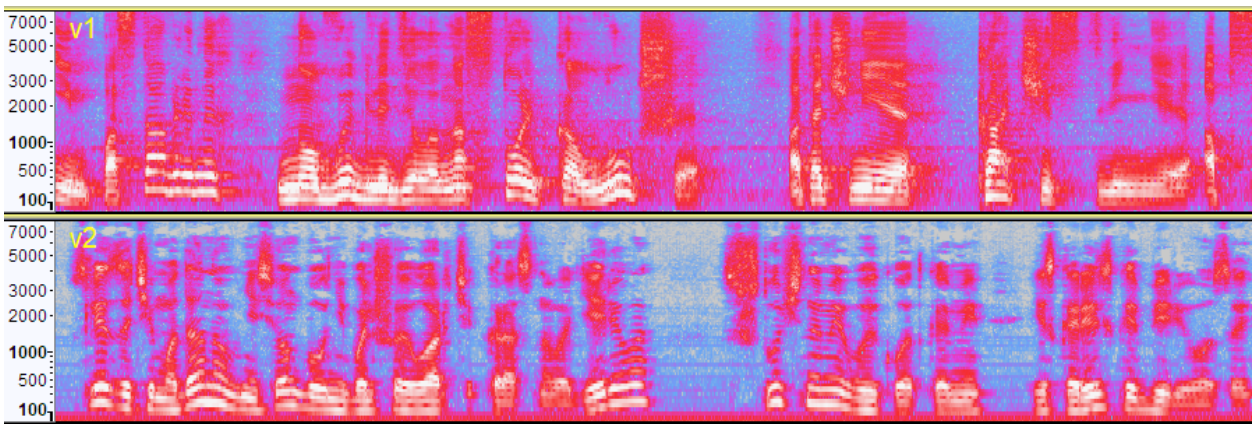

Рис. 5. Спектральное представление двойного разговора

Другим недостатком описанного подхода является низкая точность распознавания двойной речи когда фильтр не находится в оптимальном режиме.

В данной работе предлагается использовать метод, основанный на вариации скорости автоподстройки в зависимости от наличия интерференции между входными сигналами и на эффективности схождения фильтра в данный момент. При этом не требуется стороннего определения наличия двойного разговора, а только лишь изменение коэффициента шага автоподстройки μ.

Приведем краткое описание метода. В данном случае предлагается сделать коэффициент шага автоподстройки μ переменным для разных дискретных частот.

$$\mu_{opt}(k,j) = min\left(\eta(j)\frac{|Y(k,j)|^2}{|E(k,j)|^2}, \mu_{max}\right),$$

где k — номер дискретного частотного отсчета,

μ_{max} — заданная постоянная величина максимально допустимой скорости подстройки,

$\eta(j)$ — коэффициент линейной регрессии между мощностями эхо сигнала и выходного сигнала на заданной частоте.

Такой подход позволяет замедлять подстройку алгоритма на частотах,

которые интерферируют в двух сигналах.

Данный алгоритм обеспечивает хорошую скорость адаптации в общем случае и не позволяет фильтру расходиться при наличии двойного разговора.

2.2.2. Рассинхронизация

Как было показано выше, разработано множество адаптивных фильтров, которые позволяют эффективно удалять нежелательное эхо из входного сигнала. Однако, важным требованием для работы этих фильтров является точная синхронизация звуковых преобразователей на входном и выходном тракте. К сожалению, как показывает практика, эффективность эхоподавления значительно снижается при рассинхронизации потоков.

Это происходит ввиду двух факторов: несоответствия количества дискретных отсчетов проигранного и записанного сигналов за некоторый промежуток времени и смещения частот в записанном сигнале относительно проигранного.

Для синхронной работы потоков необходимо что бы цифро-аналоговый и аналого-цифровой преобразователь, которые выводят и вводят звуковые потоки, работали на совпадающей частоте. К сожалению, это не всегда так даже для одного аппаратного источника звука. При использовании разных звуковых устройств расхождение частот еще более вероятно ввиду использования ими разных источников тактовых сигналов. Такая ошибка обусловлена дефектами, производственными погрешностями и разными температурными режимами разных устройств. Для нескольких дешевых звуковых устройств низкого качества относительное расхождение частот дискретизации может достигать единиц процентов (500Гц для частоты дискретизации 44100Гц).

Важно отметить, что разность частот дискретизации для определенной аппаратной конфигурации остается постоянной на протяжении длительного времени. Такого времени достаточно для совершения звонков и конференций. Значит от алгоритма подстройки частоты не требуется высокой скорости переобучения, а при каждом изменении конфигурации звуковых потоков можно сбрасывать накопленные алгоритмом данные о характерном отставании одного потока от другого.

Соответственно, важным вызовом для задачи эхоподавления в режиме реального времени является принудительная синхронизация звуковых сигналов $\mathbf{x}(n)$ и $\mathbf{d}(n)$.

Простейшее решение основано на вычислении разности частот Δf некоторым способом и точной передискретизации сигнала $\mathbf{x}(n)$. Высокая точность требует применения полифазных фильтров. К сожалению, точная передискретизация является вычислительно затратной операцией и не всегда приемлема. Однако это решение обладает простотой и надежностью при качественном вычислении разности частот входящих сигналов.

Существует и другое решение данной проблемы, основанное на изученном выше PBFDAF фильтре. Предлагается адаптивно варьировать шаг блока фильтра в зависимости от величины отставания на данной итерации. Таким образом, последующий блок $\mathbf{x}(n)$ размера N будет сформирован из входящего прообраза эхо сигнала с шагом $(M - k)$, где k - величина отставания в дискретных отсчетах одного звукового потока от другого на данной итерации. При этом происходит частичное перекрытие нескольких блоков. Важно то, что фильтр сохраняет способность эффективно удалять эхо сигнал в частотной области при $k \ll M$.

Рассинхронизация менее чем на 1 отсчет корректируется с помощью вращения фаз при быстром преобразовании Фурье, которое выполняется

над входными сигналами при работе PBFDAF фильтра. Вычисление разности частот Δf происходит с помощью отдельного алгоритма LMS в частотной области.

Данный подход хорошо подходит в качестве надстройки над вычислительно эффективным PBFDAF фильтром и показал меньшую вычислительную сложность по сравнению с точной передискретизацией сигнала.

2.3. Анализ эффективности эхоподавления

Важным критерием выбора одного из адаптивных фильтров для задачи эхоподавления является его эффективность, которая выражается в следующих факторах:

- скорость выхода на оптимальный режим;
- величина остаточного эхо сигнала в оптимальном режиме;
- устойчивость фильтра к изменению эхо пути и характера звуковых потоков;
- вычислительная сложность.

Проанализируем эти факторы. Оптимальным режимом назовем режим работы фильтра в устойчивом стационарном состоянии, при котором средний уровень остаточного эха остается постоянным. Время выхода на такой режим от начала работы фильтра является важным показателем качества его работы и зависит напрямую от скорости адаптации фильтра.

Устойчивость фильтра в разных режимах работы зависит от точности алгоритма автоподстройки. Как видно, к алгоритму автоподстройки предъявляются высокие требования: он должен быть быстрым при расстроенном

фильтре, но точным и аккуратным вблизи оптимального решения. Упомянутый выше нормализованный алгоритм наименьших средних квадратов (NLMS) хорошо подходит для этого, но у него есть целый ряд недостатков, что привело к разработке множества улучшений.

Необходимо строго задать метрику для определения величины остаточного эхо сигнала $\Delta y(n) = y'(n) - y(n)$.

В качестве такого показателя предлагается взять логарифмическую величину снижения мощности эха, выраженную в децибелах. В дальнейшем будем обозначать её аббревиатурой echo return loss enhancement (ERLE).

$$\mathbf{ERLE} = 10lg\frac{\Delta\mathbf{y}^T(j)\Delta\mathbf{y}(j)}{\mathbf{y}^T(j)\mathbf{y}(j)}$$

Это значение целесообразно вычислять поблочно, усреднением на нескольких блоках. Использование скользящего среднего позволит получить непрерывные графики без резких колебаний значений на отдельных блоках. Такие колебания обусловлены резкими изменения мощности входного эхо сигнала и эффективности его подавления ввиду других факторов. Будем использовать простое арифметическое скользящее среднее. Формула вычисления среднего значения некоторой величины p на шаге n с размером окна k элементов:

$$\overline{p}_n = \frac{1}{k}\sum_{i=0}^{k-1} p_{n-i}$$

Размер окна будем подбирать эмпирически в зависимости от размера блока и силы зашумленности графика.

Остановимся подробнее на методике вычисления сигнала $\Delta y(n)$. Так как в реальной акустической системе действительный эхо сигнал $y'(n)$ неиз-

вестен, приходится прибегнуть к симуляции, которая программно смешивает полезный сигнал речи $v(n)$ с записанным ранее эхо сигналом. Это позволяет после отработки алгоритма эхоподавления простым алгебраическим вычитанием получить искомую ошибку:

$$\Delta y(n) = e(n) - v(n)$$

Отметим некорректность подхода, при котором сигнал речи $v(n)$ отсутствует во входящем потоке $d(n)$. Дело в том, что сходимость адаптивного фильтра значительно зависит от величины сигнала ошибки $e(n)$, которая, в свою очередь, зависит от наличия не коррелирующего с эхом сигнала $v(n)$. В самом деле, такой сигнал значительно влияет на поведение алгоритма и его отсутствие внесёт большую погрешность в полученные результаты. Условия симуляции должны быть максимально приближены к реальным.

2.4. Требования к производительности

Основной проблемой при программной реализации эхоподавления является высокая вычислительная сложность. Точнее говоря, низкая энергоэффективность таких решений.

Современные компьютеры вполне способны программно обрабатывать звуковые потоки высокого качества в реальном времени. Однако для использования этой технологии в повседневном режиме необходимо обеспечить максимально низкую "фоновую"нагрузку, которая не будет значительно влиять на доступные ресурсы и энергопотребление.

В настоящее время обработка звука, в том числе и эхоподавление, часто интегрирована в звуковую подсистему пользовательских устройств на

аппаратном уровне. Это обеспечивает высокое качество коммуникации с использованием таких устройств и минимальное влияние на общую производительность системы. Однако в рассматриваемом случае обработка звукового сигнала с целью эхоподавления происходит на уровне эмуляции аппаратуры и ложится на центральный процессор хостовой системы.

Очень важно добиться высокого уровня быстродействия системы во время работы алгоритма в одной, и даже в нескольких гостевых ОС. Высокая нагрузка на ЦП так же приведёт к нежелательному высокому энергопотреблению и тепловыделению. А работа алгоритма во время простоя компьютера не позволит ему перейти в режим частичного сна для уменьшения потребления энергии.

Несмотря на всю важность качества эхоподавления, выбор математического алгоритма эхоподавления основывается во многом на компромиссе качества и производительности. Из таких же соображений будут выбраны оптимальные параметры для выбранного алгоритма.

Суммарную нагрузку на центральный процессор можно вычислить исходя из соотношения времени обработки одного блока адаптивным фильтром к длительности этого блока во времени. Эту величину можно получить исходя из количества отсчетов в блоке и частоты дискретизации используемого устройства.

Так как звуковое устройство работает равномерно, адаптивный фильтр будет получать на выход блоки равной длины с постоянным периодом. Легко заметить, что нагрузка на ЦП будет равномерной и однопоточной:

$$CPU = \frac{\Delta T f}{N}$$

Целесообразно обозначить планку потребления вычислительных ре-

сурсов центрального процессора хостовой системы порядка 10%. Это число выбрано исходя из соображений наименьшего негативного влияния на компьютер и его энергопотребление.

2.5. Реализация

В результате проведенного исследования алгоритмов эхоподавления был выбран многоблочный адаптивный фильтр в частотной области (MDF фильтр). Такой выбор обусловлен следующими факторами:

- Низкая вычислительная сложность;
- Малая вносимая в звуковой поток задержка;
- Высокая скорость адаптации и хорошая устойчивость на реальных сигналах.

Указанные преимущества замечательно согласуются с поставленными задачами.

Для повышения эффективности эхоподавления при коррелирующих входных сигналах (двойной разговор) используется описанная выше техника, основанная на вариации скорости автоподстройки фильтра в зависимости от величины интерференции входных сигналов на разных частотах.

Для принудительной синхронизации потоков применён метод точной передискретизации сигналов. Это позволяет использовать алгоритм вне зависимости от наличия синхронизации между выбранными аппаратными источниками звука. Подчеркнём, что для тестирования эхоподавления выбирались синхронизированные источники звука. Это сделано намеренно для того, что бы не вносить дополнительные погрешности в измерения.

В качестве основы для реализации эхоподавления взят исходный код открытого звукового кодека Speex. Исходные коды были доработаны для соответствия поставленным задачам.

Система эхоподавления была интегрирована в звуковой стек гипервизорного виртуализационного решения Parallels Desktop. В качестве прообраза эхо сигнала в алгоритм поступает звуковой поток из гостевой системы, а в качестве полезного сигнала — запись с микрофона хост компьютера. После этапа эхоподавления сигнал отправляется в гостевую систему. Отметим, что эхоподавление может работать с произвольным количеством гостевых систем.

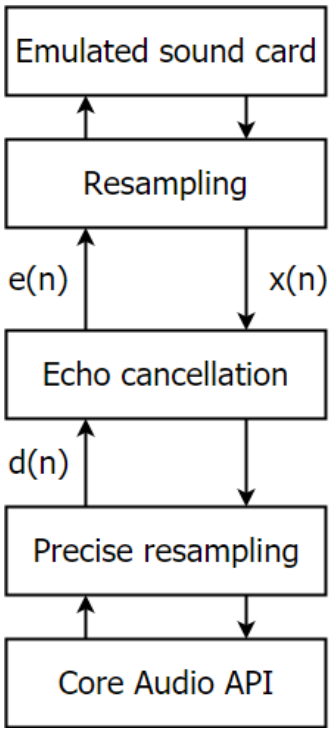

Рис. 6. Схема работы аудио подсистемы

На Рис. 6 представлена схема работы алгоритма в составе стека виртуальной аудио подсистемы. Стрелками обозначены направления движения звуковых потоков, которые на этапе работы эхоподавления подписаны в соответствии с принятыми обозначениями.

Для передачи звука на реальное аппаратное обеспечение использует-

ся фреймворк Core Audio, который является стандартным средством для ввода/вывода звука в операционной системе OS X.

Отметим также то, что представленный аудио стек работает на блочной основе: звуковой поток между разными слоями передаётся блоками, или, иначе говоря, пакетами дискретных отсчётов. Это происходит равномерно во времени. Использование многоблочного адаптивного фильтра в частотной области (MDF) лишь наложило особое требование на размер блока, который должен быть степенью числа 2.

3. Тестирование и результаты

3.1. Тестовый стенд

Для замеров производительности использовалось реальной окружение. Напомним, эхоподавление реализовано в составе гипервизорного виртуализационного решения Parallels Desktop. В качестве источника и приемника звука выступает пользовательское приложение в гостевой ОС. Также используются внешние динамики и микрофон.

В качестве хост компьютера используется Apple MacMini Late 2012. Приведем характеристики компьютера :

- CPU: Intel Core i7-3615 4+4 cores;
- RAM: 16Gb 1333 MHz DDR3;
- OS: OS X 10.13 "High Sierra".

В качестве компилятора используется clang в составе IDE Xcode 9.2. Это является стандартным тулчейном для OS X. Тесты проводятся в двух вариантах: без оптимизаций компилятора (опция -O0) и с включенными оптимизациями (опция -O2). Этот шаг не вносит большого вклада в данную работу и интересен исключительно с точки зрения эффективности современных компиляторов. Так как алгоритм эхоподавления имеет большое количество циклов и математических вычислений, оптимизации могут существенно ускорить его работу, обеспечив максимальное использование возможностей современного процессора.

Для измерений эффективности эхоподавления использовалась другая программная конфигурация, основанная на симуляции. Такой выбор обу-

словлен необходимостью точно вычислять величину остаточного эхо сигнала, как было указано в соответствующем разделе. Симулятор представляет собой отдельную программу, работающую независимо от звуковых устройств и гипервизорного виртуализационного решения. Опишем процесс её работы подробнее.

На вход программе подаётся:

- прообраз эхо сигнала $x(n)$;
- реальная запись (образ) эхо сигнала $y'(n)$;
- полезный сигнал речи $v(n)$.

Последние два сигнала смешиваются и поступают на вход адаптивного фильтра в виде суммы $d(n) = v(n) + y'(n)$.

Результатом работы фильтра является сигнал $e(n)$. Результирующее остаточное эхо вычисляется как $\Delta y(n) = e(n) - v(n)$. Исходя из этого вычисляется эффективность эхоподавления по метрике ERLE.

Все сигналы, участвующие в симуляции подготовлены заранее и хранятся в виде файлов. Результат работы также сохраняется в файл. Симуляция работает без привязки к реальному времени.

Данная система максимально приближена к реальному процессу эхоподавления, так как образ эхо сигнала представляет из себя реальную запись микрофоном проигранного прообраза эхо сигнала. Это обеспечивает реалистичность эхо пути и наличие в образе искажений, характерных для реальных систем звукозаписи.

3.2. Производительность

В результате измерений было получено время обработки одного блока адаптивным фильтром в зависимости от размера этого блока. Напомним, что размер блока должен быть степенью двойки для оптимальной работы быстрого преобразования Фурье. Представим на графике полученные результаты для машинного кода, скомпилированного без оптимизаций и с ними.

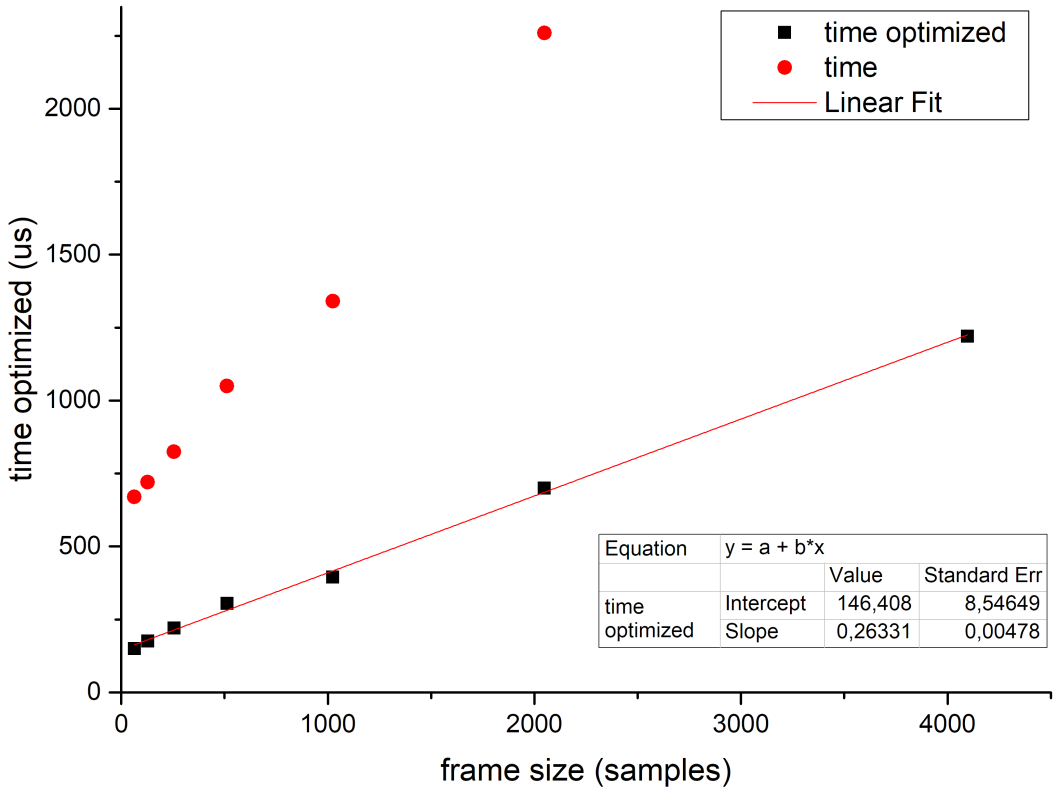

Рис. 7. Время обработки блока звукового сигнала

Хорошо видно, что оптимизации компилятора решительно влияют на конечную производительность алгоритма. Время обработки одного блока отличается примерно в 4 раза. Это подчеркивает высокую эффективность оптимизаций в современных компиляторах и важность использования оптимизаций в реальном применении. В дальнейшем анализ строим на оптимизированной версии машинного кода.

Легко заметить, что точки хорошо ложатся на прямую линию со смещением в нуле, что говорит о наличии нагрузки постоянного характера, не зависящей от N. На графике зависимость вида $O(nlog(n))$ не прослеживается. Это может говорить о преобладании линейной сложности на блоках малого размера. Другими словами, на данных размерах блока время обработки не достигло своей асимптотики.

Увеличивать размер блока и дальше не представляется целесообразным ввиду значительных задержек, вносимых таким алгоритмом в звуковой тракт. В самом деле, в случае с блоком размером в 4096 отсчета задержка составляет 93мс, что не всегда допустимо.

Напомним, что задачей является найти оптимальный режим эхоподавления, который будет иметь приемлемую задержку и низкую вычислительную сложность. Для наглядности приведем эти же данные, представленные в виде зависимости нагрузки на центральный процессор от размера блока.

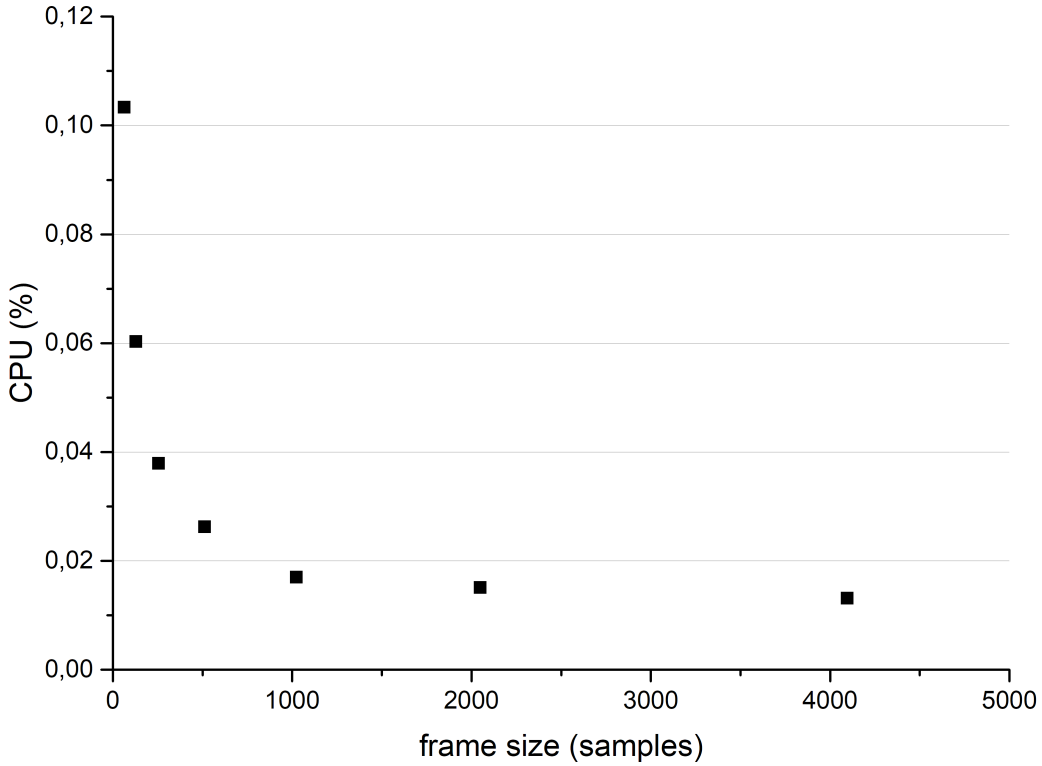

Рис. 8. Нагрузка на ЦП в зависимости от размера блока

Из представленного графика видно, что наиболее оптимальным является размер кадра в 1024 отсчета. При этом задержка составляет 23,2 мс, что является вполне приемлемым значением для реальных применений. Бóльшие размеры блока не способствуют значительному снижению нагрузки, однако повышают задержку.

Напомним, что полученная нагрузка на ЦП является однопоточной и рассчитывается как соотношение времени обработки блока на очередной итерации к периоду итерации фильтра. Полученная нагрузка составляет чуть меньше 2%, что является крайне хорошим результатом. Такая нагрузка очень мало влияет на общее энергопотребление и тепловыделение хост компьютера, а значит, эхоподавление может работать во время простоя компьютера. Заметим, что не составляет проблемы работы эхоподавления одновременно с несколькими виртуальными машинами.

3.3. Эффективность эхоподавления

Для исследования эффективности эхоподавления проводились серии тестов в симуляторе, который работал с заранее подготовленными звуковыми потоками.

В качестве входных данных использовались несколько типов звуковых сигналов:

- Человеческая речь;
- Плотная музыка с обилием инструментов;
- Синтетическая электронная музыка;
- Сигнал тишины в качестве тестового.

Эффективность эхоподавления будем измерять метрикой ERLE, изученной выше. Напомним, это логарифмическая величина снижения мощности эха, которая показывает, насколько остаточное эхо слабее его записанного прообраза. Эта численная метрика является общепринятой для задачи эхоподавления.

Для начала ознакомимся с работой адаптивного фильтра на несколько синтетическом примере: монолог человека на фоне эха, состоящего из синтезированной электронной музыки. Такой сигнал имеет довольно четкую спектральную картинку, совершенно не коррелирует с полезным сигналом и фильтр должен с ним хорошо справляться.

На Рис. 9 расположены амплитуды следующих сигналов: полезный речевой сигнал, эхо сигнал, ошибка вычисления эха, посчитанная синтетически, и сигнал на выходе из фильтра. Как видно на этих графиках, результирующий сигнал весьма близок к исходному, и влияние эхо сигнала с течением времени сходит на нет. Обратимся к более строгим численным результатам по величине ERLE ().

На графике силы подавления эха представлено несколько экспериментальных наборов, которые отличаются использованным размером блока фильтра N. Возьмем за основу, как было решено ранее, размер блока $N = 1024$ отсчета. Напомним, это обусловлено малой задержкой (порядка 23мс) и низкой нагрузкой на систему. Как видно, алгоритму необходимо существенное время для обучения. Выход на оптимальный режим происходит после 20 секунд. Существенный спад на 33 секунде работы обусловлен резким изменением характера эхо сигнала: в данный момент мелодия становится значительно более агрессивной и меняет спектральные характеристики. Впрочем, фильтр быстро адаптируется и восполняет потери. В результате в оптимальном режиме сила подавления эха $ERLE \approx 30$дБ.

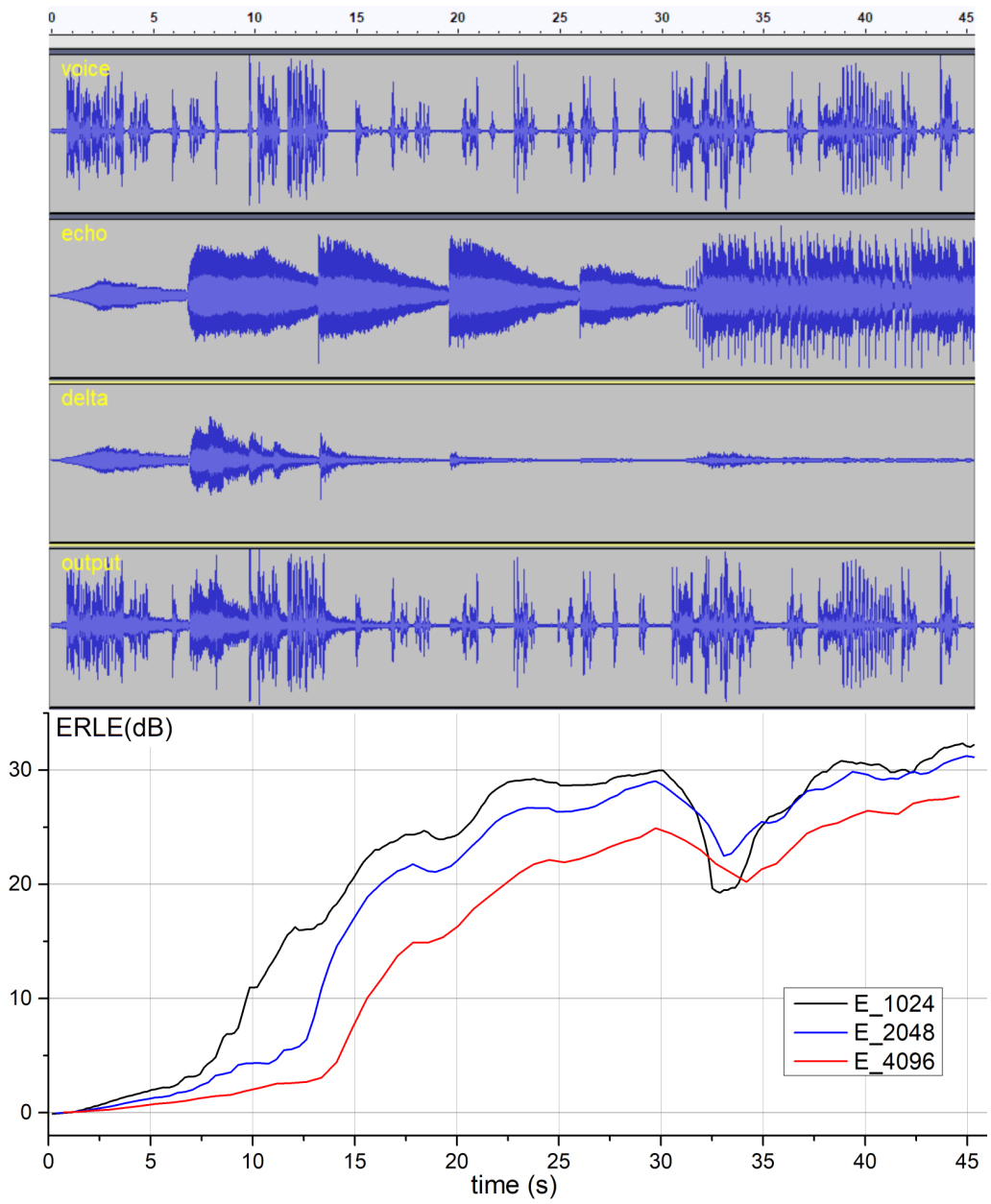

Рис. 9. Амплитуды сигналов и подавление эха

Проанализируем зависимость силы подавления эха от размера блока фильтра. На графике отчетливо видно, что скорость обучения фильтра существенно зависит от размера блока. Причина этого явления заключается в том, что обучение фильтра происходит итеративно, а количество итераций фильтра обратно пропорционально размеру одного блока. В дальнейшем наблюдается рост силы подавления и постепенное сближение всех трех кривых. Существенный спад на 33 секунде - так же общее явление, эффективность всех фильтров падает примерно до 20дБ.

Исходя из представленных данных можно сделать предварительный вывод, что размер блока $N = 1024$ отсчета является оптимальным и обеспечивает наилучшую скорость автоподстройки адаптивного фильтра.

Проведем аналогичный эксперимент в отсутствии речевого сигнала. В данном случае во входном потоке находится исключительно эхо сигнал. Приведем график силы подавления эха в этом случае и сравним его с графиком из предыдущего эксперимента.

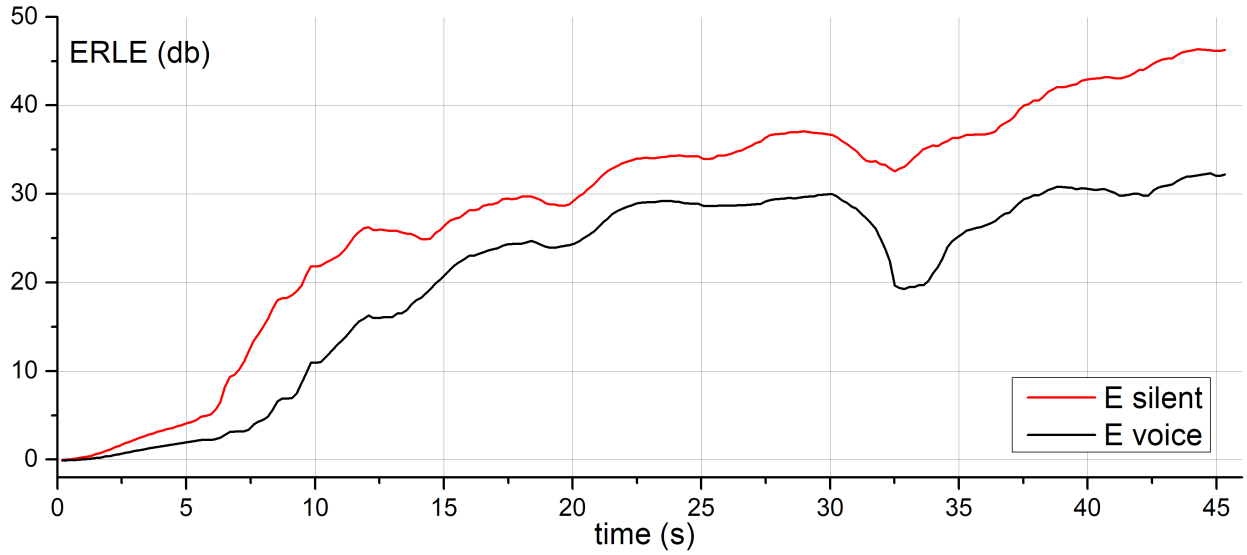

Рис. 10. Подавление эхо в сигнале, не содержащем полезный

Как видно на Рис. 10, сила подавления эха значительно выше в случае отсутствия речевого сигнала и достигает 45дБ. Это обусловлено тем, что наличие полезного, не кореллирующего с эхом сигнала увеличивает сигнал ошибки, тем самым накладывая шум на алгоритм автоподстройки. Этот фактор приводит к более долгой сходимости алгоритма. В отсутствии постороннего звука алгоритм обучается в идеальных условиях и сходится в наиболее оптимальной точке.

К сожалению, такой тест является синтетическим, так как в реальности всегда есть шумы и нежелательные сигналы. Однако это демонстрирует устройство и слабые стороны выбранного адаптивного фильтра.

Проведем эксперимент с тестовым сигналом, содержащим двойной разговор. В таком случае эхо сигнал содержит речь одного собеседника, а полезный сигнал — другого. Этот пример интересен тем, что часто встречается в реальных коммуникационных системах при общении людей. Особенность заключается в том, что такой набор сигналов коррелирует между собой. Как было указано в предыдущих главах, такие условия является существенной сложностью для адаптивных фильтров на основе LMS алгоритма, т.к. корреляция сигналов мешает правильной адаптации фильтра.

Приведем краткий отрезок амплитудного графика (Рис. 11). На нем видно, что на 10 секунде процесса эффективность резко повышается, и остаточный эхо сигнал становится малозаметен.

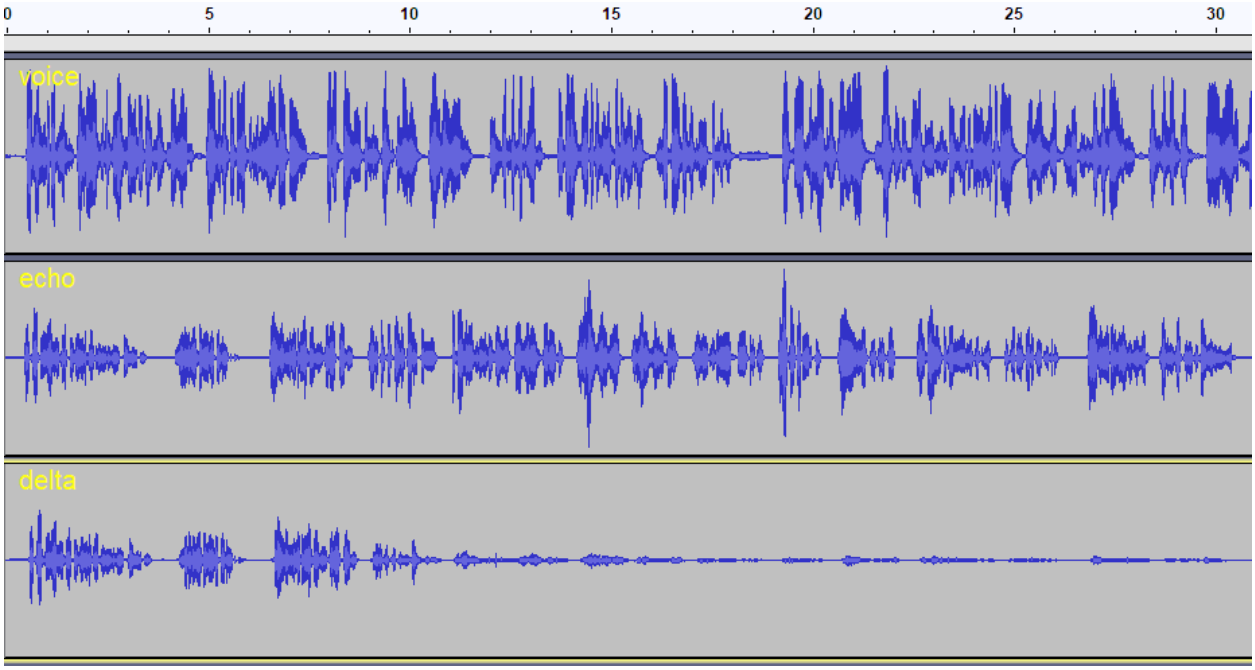

Рис. 11. Работа фильтра в случае двойного разговора

Обратимся к Рис. 12 — графику затухания мощности эхо сигнала (ERLE). Этот график построен для всей длины сигнала, которая составляет 85 секунд. На нем отчетливо видно, что на 10 секунде работы адаптивного фильтра эффективность кардинально повышается и сила подавления эха возрастает до 20дБ. В дальнейшем, фильтр выходит на оптимальный ре-

жим, в котором эхо подавляется на уровне 25-35дБ. Конкретная величина силы затухания меняется в зависимости от особенностей входных сигналов. Так, например, эффективность резко падает в случае тихой паузы в эхо сигнале, т.к. эхо сигнал либо отсутствует, либо представлен сложными нехарактерными для этого сигнала шумами, которые сложно подавить.

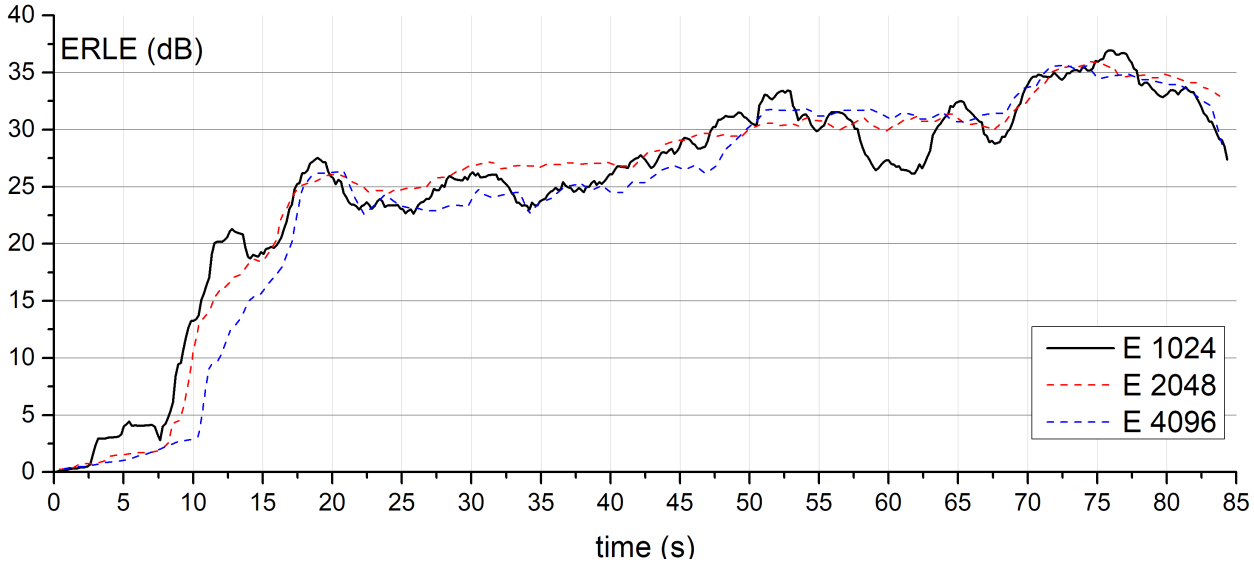

Рис. 12. Подавление эха в случае двойного разговора

На данном графике пунктирными линиями обозначены кривые силы эхоподавления для бóльших размеров блока: 2048 и 4096 отсчетов. Хорошо видно, что в оптимальном режиме различие между тремя кривыми незначительно и выбор лидера неочевиден. Однако важно то, что скорость выхода на оптимальный режим отличается в пользу малого блока, хотя и незначительно в данном испытании. Этот эксперимент подтверждает эффективность выбора размера блока $N = 1024$ отсчета.

Исходя из последних результатов, можно сделать вывод о высокой эффективности фильтра даже в случае с сигнала с двойным разговором. Это обусловлено применением метода вариации скорости автоподстройки при наличии интерференции между сигналами.

Исследуем зависимость эффективности подавления эхо сигнала от раз-

мера всего фильтра. Для этого, попробуем варьировать количество блоков в фильтре M при заданном размере блока N. Для этого возьмем новую пару сигналов: речь в качестве полезного сигнала и музыкальную композицию с плотной спектральной характеристикой в качестве эхо сигнала. На Рис. 13 представлены амплитудные характеристики входных сигналов и остаточного эха.

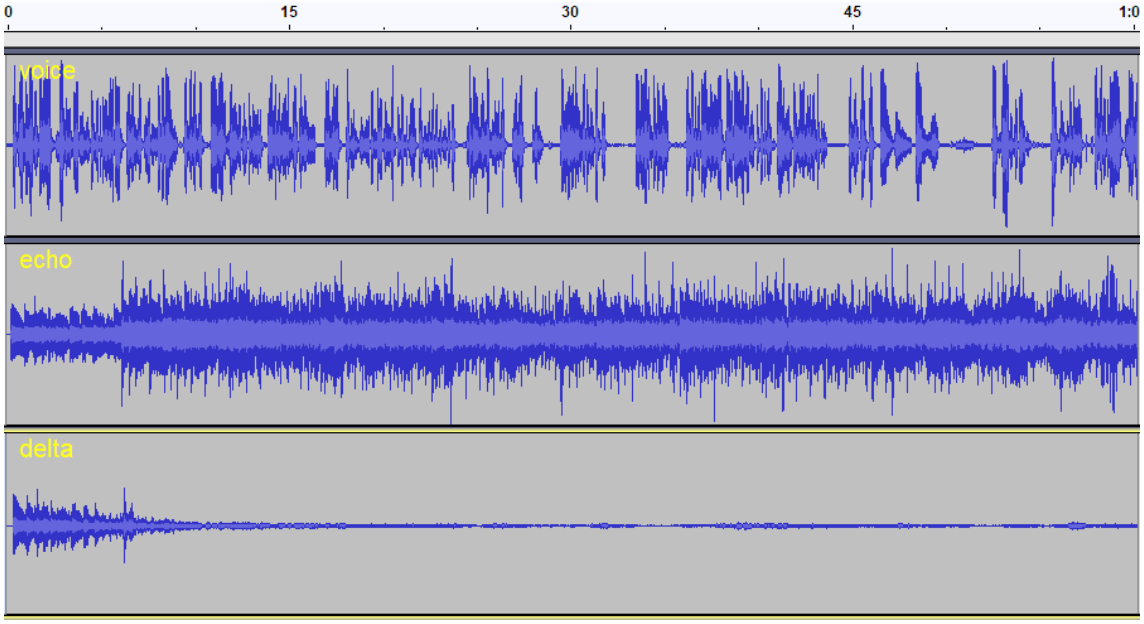

Рис. 13. Работа фильтра на типичном примере

На графиках хорошо видно что алгоритм очень быстро адаптируется и сила остаточного эха резко уменьшается. Приведем графики зависимости силы подавления эха ERLE от времени.

На Рис. 14 отображены 4 кривые, которые имеют длины фильтра по времени от 130 до 750мс. Размер блока при этом $N = 1024$ отсчета, соответственно количество блоков в фильтре варьируется от 6 до 33. При этом эхо путь намеренно остается неизменным: его максимальная длина составляет ≈ 120мс.

Легко заметить, что с ростом длины фильтра существенно замедляется адаптация фильтра. Соответственно, увеличивается время выхода на

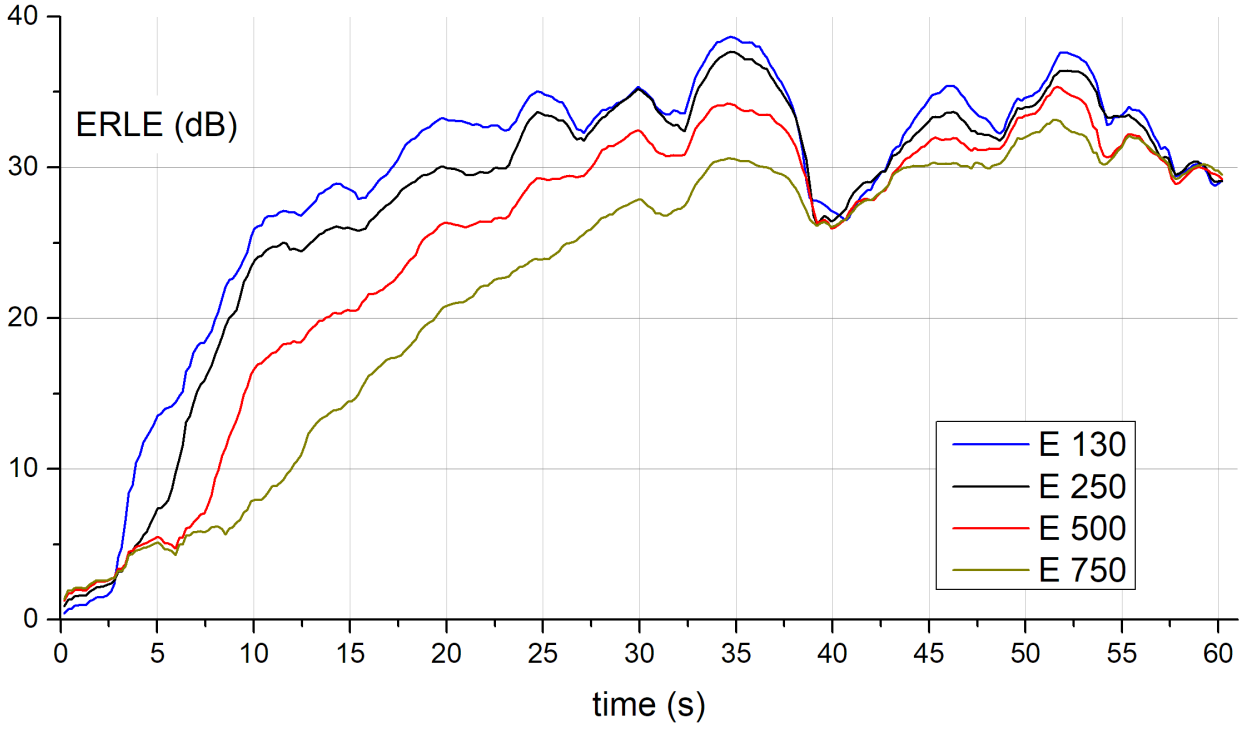

Рис. 14. Подавление эха в зависимости от длины фильтра

оптимальный режим. В целом, все кривые ведут себя аналогично и испытывают синхронный рост и падение. В дальнейшем они сближаются, обеспечивая аналогичный уровень подавления эхо в оптимальном режиме. Стоит также заметить, что на 38-й секунде наблюдается резкий синхронный спад, обусловленный резким изменением спектральных характеристик эхо сигнала и корреляцией с сигналом голоса.

Исходя из полученных результатов, сделаем вывод что целесообразно выбирать наименьшую приемлемую длину фильтра для повышения эффективности работы фильтра. Нижней планкой длины фильтра должна служить максимальная длина эхо пути в звуковой системе. Эта величина не поддается вычислению в общем случае, однако может быть оценена исходя из логических соображений и характерных задержек звукового оборудования в компьютере.

На всех представленных примерах прослеживается хорошая эффективность эхоподавления. Скорость выхода на оптимальный режим состав-

ляет 20-30 секунд, при этом сила подавления мощности эхо сигнала составляет порядка 30дБ. При субъективном анализе полученных результатов остаточный эхо сигнал не слышен на фоне полезного сигнала умеренной громкости, что говорит о достаточном качестве удаления эха. Оптимальный размер блока составляет 1024 отсчета.

4. Заключение

В результате данной работы были изучены различные алгоритмы акустического эхоподавления, основанные на цифровых адаптивных фильтрах. Наиболее эффективный алгоритм был реализован в составе гипервизорного виртуализационного решения и протестирован с точки зрения производительности и эффективности.

Проведённые тесты показали высокую производительность выбранного алгоритма, которая позволяет использовать алгоритм в повседневной жизни. Также на основе тестов были выбраны оптимальные параметры для алгоритма. Тесты алгоритма в симуляторе показали хорошую эффективность эхоподавления и высокую скорость выхода на оптимальный режим при различных использованных входных сигналах.

Полученные результаты однозначно говорят о целесообразности применения акустического эхоподавления в составе виртуальных звуковых устройств для обеспечения гостевых систем качественным звуком.

Список литературы

1. Bernard Widrow, Samuel D. Stearns. *Adaptive Signal Processing.* Prentice Hall, 1985.
2. Sabri M. Hanshi, Yung-Wey Chong. *Review of acoustic echo cancellation techniques for voice over IP.* Journal of Theoretical and Applied Information Technology, 2015.
3. Abhishek Deb, Asutosh Kar, Mahesh Chandra. *A technical review on adaptive algorithms for acoustic echo cancellation.* International Conference on Communication and Signal Processing, 2014.
4. Simon S. Haykin, Bernard Widrow. *Least-Mean-Square Adaptive Filters.* Wiley, 2003.
5. Jia-Sien Soo, Khee K. Pang. *Multidelay Block Frequency Domain Adaptive Filter.* IEEE, 1990.
6. Jean-Marc Valin. *On Adjusting the Learning Rate in Frequency Domain Echo Cancellation With Double-Talk.* IEEE, 2007.
7. Feiran Yang, Ming Wu. *A Computationally Efficient Delayless Frequency-Domain Adaptive Filter Algorithm.* IEEE, 2013.
8. Jose Manuel Gil-Cacho. *Adaptive Filtering Algorithms for Acoustic Echo Cancellation and Feedback Control.* Belgium, 2013.