Санкт-Петербургский государственный университет

ДУЛЕТОВ Дмитрий Евгеньевич

Выпускная квалификационная работа

Сплайновые словари и алгоритмы разреженной аппроксимации

Уровень образования: бакалавриат

Направление *09.03.04 «Программная инженерия»*

Основная образовательная программа *СВ.5080.2017 «Программная инженерия»*

Научный руководитель:
д.ф.-м.н., проф. А.А. Макаров

Рецензент:
инженер-программист Ю.В. Каменев

Санкт-Петербург
2022

Saint Petersburg State University

Duletov Dmitrii Evgenevich

Bachelor's Thesis

Spline dictionaries and sparse approximation algorithms

Education level: bachelor

Speciality *09.03.04 «Software Engineering»*

Programme *CB.5080.2017 «Software Engineering»*

Scientific supervisor:
Sc.D., prof. A.A. Makarov

Reviewer:
engineer-programmer Y.V. Kamenev

Saint Petersburg
2022

Оглавление

1. Введение

Что такое разреженная аппроксимация? Прежде чем ответить на этот вопрос, введём термин *разреженность*. Разреженность — количество ненулевых элементов вектора определяемое следующим образом:

$$||x||_0 = \{k : x_k \neq 0, k = 1, ..., n\}.$$

Разреженные векторы более удобны для хранения и обработки, так как их можно записывать как позицию и значение, что ускоряет все операции над ними. В связи с этим рассмотрим следующую задачу: дан вектор $b \in \mathbb{C}^m$ и матрица $A \in \mathbb{C}^{m+n}$, тогда надо найти вектор $x \in \mathbb{C}^n$, при этом $Ax = b$. В общем случае $m \ll n$. Эта задача не имеет единственного решения, и в качестве оного может быть выбран разреженный вектор. Можно рассматривать и более общую версию задачи. Для определённых выше A и b, $\epsilon \in \mathbb{R}, \epsilon > 0$ найти x такой, что $||Ax - b||_2 < \epsilon$. Для этой задачи нахождение вектора x становится проще, так как нам больше не нужно точное решение уравнение. Таким образом мы можем малые значения в x приравнять к нулю для увеличения разреженности [7].

Разреженная аппроксимация начала активно развиваться в последние десятилетия (см., например, библиографию в работах [1]-[5]). В контексте аппроксимации сплайнами матрицу A называют *сплайновым словарём*, а каждую её строку, представленную сплайном, называют *атомом*. Зачастую выбор наиболее важных атомов из словаря осуществляется по некоторому, как правило, жадному, алгоритму [6]. Разреженная аппроксимация является одним из наиболее динамично развивающихся и перспективных методов представления сигналов, поскольку не зависит от его частоты и представления.

Разреженная аппроксимация получила множество применений в самых разных областях, таких как обработка аудио-, видео-, графической информации, задачах сжатия и восстановления сигнала, очистки от шумов [1, 7, 5]. Статья [3], в которой описываются методы обработки гравитационных сигналов, широко описывает используемые алгоритмы

разреженной аппроксимации. Также нельзя не упомянуть публикации по обработке импульсных сигналов геоакустической эмиссии [1], в основании которых лежат алгоритмы разреженной аппроксимации. Помимо этого, в упомянутой обзорной статье [7] приведены примеры некоторых библиотек, основанных на разреженной аппроксимации. Примером такой библиотеки может послужить inpaint. Библиотека GlobalBioIm [8] также представляет интерес в связи с открытым кодом и обилием использования методов разреженной аппроксимации. Создатели OMP использовали его для улучшения качества сжатия изображений [9] и в области радиолокационного синтезирования апертуры [25].

На данный момент известные алгоритмы использовали B-сплайны для построения сплайновых словарей [16, 17]. При этом в последние десятилетия активно развиваются подходы к построению сплайнов, сохраняющих свойства B-сплайнов, но имеющих неполиномиальный характер. Один из таких подходов — минимальные сплайны — исследовался в ряде работ [23, 26]. Такие сплайны могут дать более точную аппроксимацию, поскольку они получаются из аппроксимационных соотношений. Исследованию применения тех или иных сплайновых словарей в различных прикладных задачах разреженной аппроксимации и посвящена настоящая выпускная квалификационная работа.

2. Постановка задачи

Целью работы является улучшение характеристик (точности приближения и скорости работы) разреженной аппроксимации, получаемой применением ортогональных жадных алгоритмов для сплайновых словарей. Построенные приближения при этом анализируются на различных сигналах типовых задач разреженной аппроксимации, рассмотренных в недавних публикациях [3, 6].

Для достижения поставленной цели были сформулированы следующие задачи:

1. Изучение существующих работ в области разреженной аппроксимации и использующихся в них алгоритмах.

2. Подготовка набора тестовых сигналов для проведения экспериментов и реализация соответствующего модуля обработки.

3. Ускорение используемых алгоритмов при помощи параллельных вычислений.

4. Разработка нового алгоритма разреженной аппроксимации для использования преимуществ сплайновых словарей.

5. Проведение численных экспериментов на подготовленных сигналах и оценка эффективности использования словарей из минимальных сплайнов в задачах разреженной аппроксимации.

3. Обзор

3.1. Алгоритмы разреженной аппроксимации

Задачей алгоритма разреженной аппроксимации является восстановления x из уравнения $y = \Phi x$, где y - известные данные, а Φ будем называть словарём. Существует два основных подхода к построению разреженной аппроксимации — Basis Pursuit и Matching Pursuit, которые были разработаны в 90-е годы XX века и активно развиваются и по сей день. Основополагающими алгоритмами обоих семейств являются Lasso и Orthogonal Matching Pursuit (далее OMP). Первый алгоритм является l_1 регуляризованным МНК, а OMP выбирает на каждом шаге наиболее коррелированный атом (подробнее см. в [7]). В ходе исследования, проведённого в предыдущих работах [31], было принято решение пользоваться алгоритмом OMP. Помимо этих семейств алгоритмов, существуют и другие подходы, например, CoSAMP и Iterative TreshHolding, однако в данной работе предпочтение отдано алгоритму OMP, так как его эффективность была доказана в работе [25] при помощи анализа свойства ограниченной изометрии.

Данный алгоритм проходит следующие этапы.

1. Инициализация $x_0 = 0, r_0 = y$ - остаток, $k = 1, \Lambda_0 = \emptyset$ - множество выбранных индексов.
2. Оценка остатка $r_k = y - \Phi x_{k-1}$.
3. Вычисление $\Lambda_K = \underset{k}{argmax} |\langle r_k, \varphi_i \rangle|$.
4. Оценка ненулевых величин вектора $x_k = \underset{k}{argmin} ||\Phi_k x_k - y||_2$. Компоненты с номерами не из Λ_k считать равными нулю.
5. $k := k + 1$, если $||r_k||_2 >$ порог, перейти к шагу 2.

Также интерес представляет алгоритм Stagewise Orthogonal Matching Pursuit (далее StOMP) [11], разработанный создателем OMP. Подобно

OMP, приближение сигнала строится итеративно, но на третьем этапе выбирает сразу множество номеров, для которых

$$\underset{k}{argmax}|\langle r_k, \varphi_i \rangle| > treshhold_k,$$

где $treshhold_k$ — задаваемый на каждой итерации порог.

Алгоритм StOMP представляет собой цикл, условие останова которого зависит от второй нормы остатка. На каждом шаге алгоритма пересчитывается разность текущего результата и исходного сигнала, назовём её r_k. Если $||r_k|| <$ порога, который задаётся в начале программы, то цикл прерывается. На каждом этапе алгоритма выбирается нефиксированное количество атомов, влияние которых превосходит лямбду, и параметр, зависящий от r_k. Эти атомы добавляются в набор уже выбранных ранее атомов и для этого набора находится псевдообратная матрица Мура-Пенроуза. Эта операция имеет схожий результат с решением системы линейных уравнений. При помощи этой матрицы находятся коэффициенты для соответствующих набору атомов. После этого пересчитывается остаток и цикл повторяется.

3.2. Словари

Выбор словаря не менее важен, чем выбор метода разреженной аппроксимации и является вторым ключевым вопросом на пути построения качественного приближения. Существует обзорная статья [2], имеющая более тысячи цитирований в Scopus, широко описывающая свойства и особенности применения многих известных словарей. На основании приведённых в ней сведений было решено выбрать словарь Габора, как давно успевший подтвердить свою значимость инструмент. Элемент словаря Габора формируется по следующей формуле [32]:

$$atom(n) = \frac{cos\left(2\pi f n + p\right) \cdot exp\left(\frac{-\pi \cdot n^2}{s^2}\right)}{\sqrt{s}},$$

где f — частота от 0 до 0.5, n — номер атома от 1 до N, s — масштаб, $2^s < N$, p — фаза от 0 до π.

Также из множества словарей, наиболее активно использующихся в современных прикладных задачах разреженной аппроксимации для проведения экспериментов, был выбран словарь DCT, основанный на дискретном косинусном преобразовании [7]. Элемент словаря DCT имеет следующий вид:

$$cos(\frac{\pi \cdot (j + 0.5) \cdot i}{N}).$$

Генерация происходит итеративным перебором j и i, где i — это номер атома, который принимает значения от 1 до N, а j — позиция внутри атома, принимающая значения от 1 до k.

В работе также рассматриваются минимальные сплайны нескольких видов: изучавшиеся ранее словари из B-сплайнов [16, 17], а также тригонометрические [22] и гиперболческие [23]. Построение словаря на основе минимальных сплайнов происходит следующим образом: единичный отрезок делится на n-1 равных отрезков, где n — количество атомов. Также для корректного построения сплайн-функций на концах отрезка слева и справа добавляется по два фиктивных отрезка. Далее составляется массив концов этих отрезков, будем обозначать их x_i и называть этот массив сеткой. На каждой итерации проходим по всем точкам измерения и проверяем, попадают ли они на отрезок $[x_i, x_{i+3}]$ (именно там располагается носитель сплайна, соответствующий i-ому атому). Если точка лежит в указанном отрезке, вычисляем значение в соответствии со сплайн-функцией, в противном случае она считается равной нулю.

Также построим дополнительные сетки путём прореживания оригинальной. Возьмём из сетки все узлы с нечётными номерами и удалим их. Таким образом сетка укрупнилась в два раза. После этого на этой сетке по тем же правилам строятся сплайны, получается ещё один уро-

вень нашего словаря. Повторяя этот процесс итеративно, можно значительно увеличить избытычность словаря, что потенциально может улучшить разреженность решения.

Для вычисления точного значения сплайн-функций на носителе используются следующие формулы:

Формулы вычисления B-сплайнов

$$\omega_j^B(t) = \frac{(t-x_j)^2}{(x_{j+1}-x_j)(x_{j+2}-x_j)}, t \in [x_j, x_{j+1}),$$

$$\omega_j^B(t) = \frac{1}{x_{j+1}-x_j}\left[\frac{(t-x_j)^2}{x_{j+2}-x_j} - \frac{(t-x_{j+1})^2(x_{j+3}-x_j)}{(x_{j+2}-x_{j+1})(x_{j+3}-x_{j+1})}\right], t \in [x_{j+1}, x_{j+2}),$$

$$\omega_j^B(t) = \frac{(t-x_{j+3})^2}{(x_{j+3}-x_{j+1})(x_{j+3}-x_{j+2})}, t \in [x_{j+2}, x_{j+3}).$$

Формулы вычисления гиперболических сплайнов

$$\omega_j^{hyp}(t) = \frac{ch\frac{x_{j+2}-x_{j+1}}{2} sh^2\left(\frac{t-x_j}{2}\right)}{ch\frac{x_{j+1}-x_j}{2} sh\frac{x_{j+2}-x_j}{2}}, t \in [x_j, x_{j+1}),$$

$$\omega_j^{hyp}(t) = \frac{ch\frac{x_{j+2}-x_{j+1}}{2}}{sh\frac{x_{j+1}-x_j}{2}}\left(\frac{sh^2\left(\frac{t-x_j}{2}\right)}{sh\frac{x_{j+2}-x_j}{2}} - \frac{sh\frac{x_{j+3}-x_j}{2} sh^2\left(\frac{t-x_{j+1}}{2}\right)}{sh\frac{x_{j+3}-x_{j+1}}{2} sh\frac{x_{j+2}-x_{j+1}}{2}}\right), t \in [x_{j+1}, x_{j+2}),$$

$$\omega_j^{hyp}(t) = \frac{ch\frac{x_{j+2}-x_{j+1}}{2} sh^2\left(\frac{x_{j+3}-t}{2}\right)}{sh\frac{x_{j+3}-x_{j+1}}{2} sh\frac{x_{j+3}-x_{j+2}}{2}}, t \in [x_{j+2}, x_{j+3}).$$

Формулы вычисления тригонометрических сплайнов

$$\omega_j^T(t) = sin^2\left(\frac{t-x_j}{2}\right) sin^{-1}\left(\frac{x_{j+2}-x_j}{2}\right) sin^{-1}\left(\frac{x_{j+1}-x_j}{2}\right), t \in [x_j, x_{j+1}),$$

$$\omega_j^T(t) = \tfrac{1}{2} \cdot$$
$$\cdot \left[sin\left(\tfrac{x_{j+2}+x_{j+3}}{2} - t\right) cos\tfrac{x_{j+1}-x_j}{2} + sin\left(t - \tfrac{x_j+x_{j+1}}{2}\right) cos\tfrac{x_{j+3}-x_{j+2}}{2}\right] \cdot$$
$$\cdot sin^{-1}\left(\tfrac{x_{j+2}-x_{j+1}}{2}\right) sin^{-1}\left(\tfrac{x_{j+3}-x_{j+1}}{2}\right) sin^{-1}\left(\tfrac{x_{j+2}-x_j}{2}\right), t \in [x_{j+1}, x_{j+2}),$$

$$\omega_j^T(t) = sin^2\left(\frac{t-x_{j+2}}{2}\right) sin^{-1}\left(\frac{x_{j+3}-x_{j+1}}{2}\right) sin^{-1}\left(\frac{x_{j+3}-x_{j+2}}{2}\right), t \in [x_{j+2}, x_{j+3}).$$

3.3. Приближаемые сигналы

В большей части статей о разреженной аппроксимации приближаются либо случайные математические сигналы, либо данные, характерные для узкой предметной области (как, например, характеристики сопротивления материалов в задачах устойчивости мостов в статье [20] или измерение цифровой линеаризации предискажений [27]). Ниже будет раскрыт список направлений использования разреженной аппроксимации, которые были выбрали для исследования.

Значительная часть статей, посвящённых разреженной аппроксимации, использует в качестве приближаемых функций синтетические математические сигналы. В качестве сигналов из этой категории взяты следующие математические функции:

- x^2
- $arctg(10 \cdot x)$
- $0.2 \cdot ch(5 \cdot x)$

Следующая категория статей, которые хотелось бы отметить, посвящена сжатию аудиофайлов [28]. Сжатие файла происходит при помощи разреженной аппроксимации, при этом в сжатом файле хранится только процедура построения словаря и список коэффициентов, что позволяет добиться очень большого уровня сжатия. В приведённой статье для аудиофайла 64 kbps было получена сжатая версия на 47% меньше, чем для стандарта MP3. Как показало наше прошлогоднее исследование, сплайновые словари хорошо подходят для аппроксимации аудиофайлов, поэтому при помощи них можно ещё улучшить качество сжатия, сохраняя ту же точность.

Множество статей о разреженной аппроксимации описывают задачу удаления шума, эта область очень популярна во многих прикладных задачах, особенно при работе с изображениями. [29]. В этой области успешно применяется алгоритм OMP, и, возможно, использование сплайнового словаря может улучшить эти результаты. Однако, задача

удаления шума также актуальна для аудиофайлов. Эта тема не изучалась в широкоцитируемых источниках, поэтому она может представлять особый интерес.

Последней интересной статьёй хотелось бы выделить изучение разреженной аппроксимации астрономических изображений [30]. В ней с фотографий звёздного неба, которые в высоком качестве занимают много места, при помощи разреженной аппроксимации считывается значимая информация для её компактного хранения. Однако, в контексте данной работы аппроксимация изображений не рассматривалась.

4. Реализация

4.1. Архитектура

Приложение, разработанное для постановки численных экспериментов, было написано на языке программирования C++ 11 версии. Несмотря на то, что большая часть программ в данной области использует MATLAB, указанный язык программирования был выбран из соображений кроссплатформенности, а также исходя из желания предоставить сообществу возможность использования данного инструмента без установки. Также язык C++ позволяет писать эффективные алгоритмы, а также использовать преимущества параллельной обработки. Программный модуль не требует дополнительных библиотек. С исходным кодом решения, опубликованным на Github, можно ознакомиться по ссылке https://github.com/Duletov/Diploma.

Программа компилируется при помощи gcc версии не ниже 4.9.2 следующей командой из основной директории:

```
g++ main.cpp -fopenmp -std=c++11 -o main
```

Запуск программы осуществляется из командной строки. Примерами запуска могут послужить следующие команды:

- `main 10 10 100 1 d o s x`
- `main 1000 1000 10000 1 h o a audio/minor.wav`

В качестве аргументов программы указываются соответственно:

- `nAtoms` — количество атомов;
- `szSignal` — длина тренировочного сигнала;
- `szTest` — длина тестового сигнала;
- `length` — длина отрезка на котором строится входная функция;

- `dictType` — тип используемого словаря;
- `algoType` — тип используемого алгоритма;
- `signalType` — тип входного сигнала;
- `signal` — сигнал.

Аргумент `nAtoms` принимает количество атомов в строящемся словаре. С увеличением количества атомов растёт избыточность и, потенциально, точность аппроксимации.

Аргумент `szSignal` отвечает за количество узлов сетки, на которой будет построена аппроксимация, а `szTest` за количество узлов сетки, на которой будет проверяться её точность. Значение `szTest` должно быть больше или равно значению `szSignal` и не больше 100 000, что связано с ограничениями на размеры создаваемых массивов.

Аргумент `length` отвечает за длину отрезка $[a, b]$, на котором строится входная функция и соответствующие словари.

Аргумент `dictType` отвечает за выбор словаря и может принимать следующие значения:

- `g` — словарь Габора;
- `d` — словарь DCT;
- `s` — словарь B-сплайнов;
- `t` — словарь тригонометрических сплайнов;
- `h` — словарь гиперболических сплайнов.

Аргумент `algoType` отвечает за выбор алгоритма и принимает следующие значения:

- `o` — OMP параллельная версия;
- `t` — OMP;
- `s` — StOMP;

- `p` — SOMP.

Перечисленные выше словари и алгоритмы описываются в следующих главах.

Аргумент `signalType` может принимать два значения, если передаётся `s`, то в качестве входного сигнала генерируется синтетическая функция. В этом случае аргумент `signal` может принимать следующие значения:

- `x` — функция x^2
- `ch` — функция $0.2 \cdot ch(5x)$
- `arctg` — функция $arctg(10x)$

Если на вход в аргументе `signalType` передаётся `a`, то программа будет обрабатывать аудиофайл при помощи модуля AudioReader. AudioReader считывает данные из аудиофайла, путь к которому передаётся в аргументе `signal` в виде `audio/minor.wav`, и передаёт данные в виде массива типа double в модуль Algorithms.

Аргумент `signal` в зависимости от значения `signalType` принимает либо строку с выбором синтетической функции, либо относительный путь к аудиофайлу. Модуль Dictionary по первым четырём аргументам и типу генерирует два соответствующих словаря длины `szSignal` и `szTest` и также передаёт их в модуль Algorithms.

В модуле Algorithm полученные данные передаются алгоритму, выбор которого определяется аргументом `algoType`.

Результатом работы выбранного алгоритма является массив коэффициентов и список выбранных атомов словаря, по которым можно восстановить исходный сигнал с максимальной точностью. Аппроксимированный сигнал также выводится в файл `results.txt`.

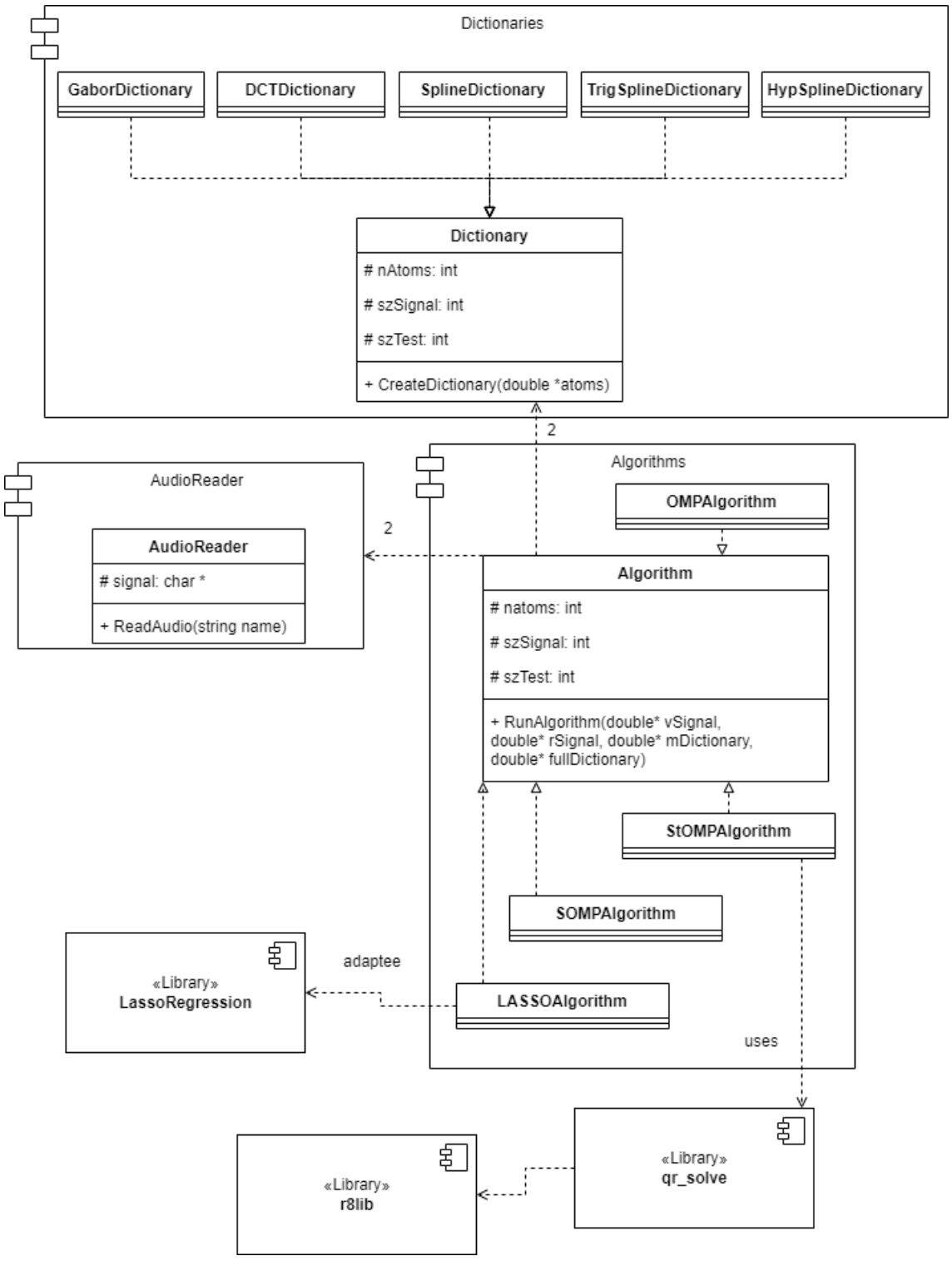

Рисунок 1 - Диаграмма классов приложения

4.2. Модуль AudioReader

Поскольку аудиофайл не требует никакой специальной обработки, была использована библиотека AudioFile, предоставляющая простой

способ считывания аудиофайлов формата wav. Считывание идёт последовательно из всех аудиоканалов, для входного сигнала берётся каждая десятая точка, что отражает дискретность считываемых данных. После считывания аудиофайла данные упаковываются в массивы типа double и передаются в модуль Algorithm.

4.3. Модуль Dictionaries

Данный модуль по количеству атомов — `nAtoms`, длине сигнала — `szSignal` и выбранному виду словаря `dictType` последовательно генерирует все атомы и формирует из них словарь.

Элемент словаря Габора в данном модуле формируется по следующей формуле

$$\frac{cos(2 \cdot \pi \cdot f \cdot (j-i) + p) \cdot exp(\frac{-\pi \cdot (j-i)^2}{s})}{s},$$

где f — частота, s — масштаб, p — фаза, i — номер текущего атома, j — позиция внутри атома. Генерация происходит итеративным перебором по j от 1 до k и по i от 1 до N. Подробное описание назначения каждого из параметров, а также мотивация к выбору возможных значений приведены в статье [18].

Элемент словаря DCT имеет следующий вид:

$$cos(\frac{\pi \cdot (j + 0.5) \cdot i}{N}).$$

Генерация происходит итеративным перебором j и i, где i — это номер атома, принимает значения от 1 до N, а j - позиция внутри атома, принимает значения от 1 до k.

Построение словаря на основе сплайнов происходит поэтапно. Весь словарь разделён на несколько уровней, каждый из которых имеет свою сетку разной степени разреженности, на которой и строятся сплайны. Первый уровень имеет n точек сетки, где n — количество атомов. По-

строение словаря происходит следующим образом:

1. На отрезке $[a, b]$ с одинаковыми промежутками выбираются n узлов текущей сетки, также по два фиктивных узла добавляется слева и справа для корректного построения сплайн-функций на концах отрезка. Будем обозначать эти точки x_i.

2. Далее происходит n итераций, на каждой из которых происходит построение одного атома. Для этого проходим по всем точкам генерируемого атома и проверяем, попадают ли они на отрезок $[x_i, x_{i+3}]$ (именно там располагается носитель сплайна, соответствующий i-ому атому). Если точка лежит в указанном отрезке, вычисляем значение сплайн-функции (формулы представлены в обзоре), в противном случае она считается равной нулю.

3. После окончания построения уровня новая сетка строится удалением половины узлов и передаётся в пункт 1, если $n > 4$.

Программа строит два словаря, один для построения аппроксимации и второй для проверки точности.

4.4. Модуль Algorithms

В этом модуле принимаются на вход словари и сигналы, полученные на предыдущих этапах, после чего в зависимости от переменной `algoType` запускается один из алгоритмов разреженной аппроксимации.

Далее подробнее рассмотрим схему работы и псевдокод каждого из них.

4.4.1. OMP

Алгоритм OMP имеет следующий вид:

На вход данный алгоритм получает массив с входным сигналом `vSignal` и массив со словарём `mDictionary`. На первой стадии идёт подготовка всех данных для начала основного хода работы. Создаётся

Algorithm 1 OMP

for $k < nAtoms$ **do**
  $iChosenAtom \leftarrow chooseAtom(residue)$
  $newDictionary[k] \leftarrow oldDictionary[iChosenAtom]$
  reorthognalize(orthogonalDictionary)
  normalize(orthogonalDictionary[k])
  calcBiorthogonal(mBiortogonal, newDictionary[k], orthogonalDictionary[k])
  calcResidue(vSignal, newDictionary)
  $normresidue \leftarrow \sqrt{\langle residue, residue \rangle}$
  if $normresidue < tolerance$ **then**
    break
  end if
end for

массив `mNewDictionary` — копия `mDictionary`, который станет основой для выбора новых элементов, массив `iOldDictionary` для запоминания нумерации, а также вычисляется первый остаток - `residue`, равный входному сигналу. Далее происходит `N` итераций, где максимальное значение `N` — это минимум из `nAtoms` и `szSignal`.

1. Для каждого атома вычисляется его влияние на уменьшение ошибки и берётся лучший из них. За этот пункт отвечает функция `chooseAtom`.

2. Выбранный атом меняется местами с атомом под номером `k`, где `k` - номер шага. При этом произошедшая смена отмечается в массиве `iOldDictionary`, где хранится номер атома в оригинальном словаре.

3. Функция `reorthognalize` ортогонализирует словарь `orthogonalDictionary`.

4. Нормализируется k-ый атом в `orthogonalDictionary`

5. В массиве `mBiorthogonal` при помощи функции `calcBiorthogonal`

рассчитываются биортогональные функции, необходимые для вычисления коэффициентов.

6. Пересчитывается остаток residue при помощи функции `calcResidue`.

7. Норма остатка `normresidue` сравнивается с требуемой точностью, и если первое значение выше, цикл повторяется.

После вычисления всех итераций рассчитывается массив коэффициентов и программа проверяет успешность найденного решения, сравнивая с образцом.

Асимптотика алгоритма OMP составляет $O(N^2 \cdot M)$, где N — это длина входного сигнала, а M — количество атомов. Алгоритм совершает максимум N итераций с учётом того, что атомов обычно больше. На каждой итерации совершаются следующие действия - выбор атома $O(N \cdot M)$, копирование элементов $O(N)$, реортогонализация и биортогонализация — $O(N \cdot M)$, нормализация $O(N)$, пересчёт остатка $O(N)$.

4.4.2. StOMP

Алгоритм StOMP работает следующим образом:

Algorithm 2 StOMP

while $\|residue\| > tolerance$ **do**
  $M \leftarrow dictionary \cdot residue$
  $lambda \leftarrow random(2,3) \cdot \|residue\|_{L_2} \cdot \sqrt{szSignal}$
  $I \cup find(M \geq lambda)$
  $StOMP \leftarrow svd\_solve(I)$
  $residue \leftarrow vSignal - dictionary \cdot StOMP$
end while

Каждая итерация алгоритма происходит следующим образом:

- Рассчитывается `M` - массив влияния того или иного атома при включении его в набор.

- Вычисляется значение $\lambda = \frac{rand(2:3) \cdot \|residue\|_2}{\sqrt{szSignal}}$.
- Каждое значение массива `M`, которое выше λ, добавляется в множество выбранных индексов.
- При помощи псевдоинверсии Мура-Пенроуза, реализация которой была имплементирована при помощи функции `svd_solve` из библиотеки `qr_solve`, находятся коэффициенты для выбранных атомов и записываются в массив `s_StOMP`.
- Норма остатка `check` сравнивается с требуемой точностью, и если первое значение выше, цикл повторяется.

Асимптотика алгоритма StOMP зависит от реализации поиска псевдообратной матрицы, сложность которой составляет $O(N \cdot M)$, и общая сложность алгоритма $O(N \cdot M)$. Реализация самой вычислительно сложной части алгоритма взята из библиотеки, что делает невозможным её улучшение. Как показывают результаты прикладных испытаний, алгоритм работает заметно дольше стандартного OMP и в сравнение по скорости входить не будет.

4.5. Параллельная версия OMP

Для распараллеливания алгоритма OMP была выбрана библиотека OpenMP, которая предназначена для удобной работы с циклами for. Для них библиотека сама создаёт нужное количество потоков и распределяет итерации цикла между ними, однако нужно следить за гонками данных и границами видимости переменных. На отдельных шагах алгоритма были произведены следующие действия:

1. В функции `chooseAtom` происходит произведение транспонированной матрицы на вектор, где значение каждого элемента итогового вектора считается в отдельном потоке.
2. Во время переноса атома вызывается функция его копирования, однако время на создание потоков нивелирует выигрыш, получае-

мый от параллельной обработки переноса значений. В связи с чем здесь дополнительные потоки не создаются.

3. В функции реортогонализации на каждом шагу изменяется k-ая строчка массива `orthogonalDictionary`, в связи с чем эффективное распараллеливание невозможно.

4. Во время нормализации атома происходит N операций, однако их параллельная обработка не даёт выигрыша по времени.

5. В функции `calcBiorthogonal` при пересчёте старых значений функций, каждое из них вычисляется в отдельном потоке.

5. Реализация собственного алгоритма

5.1. Описание алгоритма

Алгоритм OMP на каждом шагу выбирает всего один атом, что не оптимально, и многие улучшения этого алгоритма пытаются решить эту проблему. Однако, если использовать словари, основанные на сплайнах, то можно использовать их локальность, чтобы обойти это узкое место. Каждый сплайн построен всего на трёх отрезках сетки и пересекается с 4 другими сплайнами. Значит, если на шаге алгоритма выбирать каждый пятый сплайн, то они не будут влиять друг на друга, и появится возможность обрабатывать их параллельно. Если использовать словари, построенные на одной сетке, то можно было бы сократить асимптотику алгоритма до $O(N \cdot M)$, однако наш словарь состоит из разных уровней и, соответственно, сплайнах построенных на разных сетках, которые между собой пересекаются, а значит, каждый уровень придётся обрабатывать отдельно. Всего у нас уровней $log(M) + 1$, что делает нашу асимптотику равной $O(N \cdot M \cdot log(M))$

На каждой итерации идёт обработка всех уровней, количество которых подсчитывается до начала работы алгоритма. Один уровень обрабатывается по следующему сценарию:

- Выбор максимального количества из оставшихся сплайнов. Используемый метод выбора описывается ниже.
- Добавление сплайнов в обновлённый словарь.
- Реортогонализация добавленных в словарь атомов при помощи функции `reorthognalize`.
- Нормализация выбранных атомов.
- Пересчёт биортогональных функций при помощи `calcBiorthogonal`.
- Обновление остатка `calcResidue`.

Algorithm 3 SOMP

for $k < nIterations$ **do**
  while $level < szSignal$ **do**
    $chosenAtoms \leftarrow chooseAtom(residue)$
    $newDictionary[k] \leftarrow oldDictionary[chosenAtoms]$
    reorthognalize(newDictionary)
    for $atominchosenAtoms$ **do**
      normalize(newDictionary[atom])
    end for
    calcBiorthogonal(mBiortogonal, newDictionary)
    calcResidue(vSignal, newDictionary)
    $level \leftarrow level \cdot 2$
  end while
  $normresidue \leftarrow \sqrt{\langle residue, residue \rangle}$
  if $normresidue < tolerance$ **then**
    break
  end if
end for

- Сравнение нормы остатка `normresidue` с требуемой точностью; если первое значение выше, цикл повторяется.

Приступив к очередному уровню, необходимо выбрать максимальное количество из оставшихся сплайнов. Для этого считается вектор $c = \langle$ `mNewDictionary`, `residue` $\rangle$. Далее необходимо из этого вектора выбрать наибольшее количество сплайнов с большими значениями, для этого используется метод трёх указателей. Правый указатель перебирает элементы атома, и если текущий элемент больше всех предыдущих, то обновляется средний указатель на него. Левый указатель отмечает крайний сплайн, который теоретически может быть взят, и если средний указатель удалился от него дальше, чем на два, то этот сплайн выбирается, если его полезность превышает требуемую точность. Также, если разница между правым и средним маркером превышает два, то средний маркер является локально оптимальным и также попадает в разложение.

В оригинальном OMP на каждой итерации значения выбранного

атома должно быть изменено в соответствии со всеми предыдущими, в новом алгоритме атомы поступают группой и расчёты для них можно вести отдельно, а значит, и на разных потоках. Также самой тяжеловесной операцией всего алгоритма является chooseAtom, а количество использований этой операции снижается с N до $log(N)$.

6. Эксперименты

В рамках проведения экспериментов использовался персональный компьютер со следующими характеристиками: процессор Intel core i7, оперативная память DDR4 12 Гб.
Для измерения точности разреженной аппроксимации были поставлены эксперименты на синтетических функциях и на аудиофайлах.

6.1. Синтетические функции

На этом этапе тестирования для каждой функции из приведённого ниже списка строилась равномерная сетка на единичном отрезке, и для этой сетки в каждом узле бралось значение функции, после чего по этим измерениям строилась аппроксимация. Результаты прошлых работ показали, что значения погрешности в работе OMP и StOMP практически не отличаются [31], поэтому в приведённой таблице этот алгоритм не присутствует. В таблице не приведены данные для SOMP, так как точность его работы относительно OMP отличается не больше, чем на 5% в обе стороны, и в сравнении работы словарей его результаты излишни.

- x^2
- $arctg(10 \cdot x)$
- $0.2 \cdot ch(5 \cdot x)$

Для каждой функции были взяты следующие пары параметров:

- 10 атомов, 100 узлов сетки;
- 100 атомов, 1000 узлов сетки;
- 1000 атомов, 10000 узлов сетки;
- 10000 атомов, 100000 узлов сетки.

Далее идут результаты тестов, в таблице указана ошибка RMSE (Root Mean Square Error), вычислявшаяся по формуле

$$\sqrt{\frac{\sum_{i=1}^{N}(R_i^S - R_i^A)^2}{N}},$$

где R^S — значение входного сигнала в точке, R^A — значение аппроксимации в этой же точке, N — размер сигнала.

Таблица 1 - Ошибка RMSE для аппроксимаций синтетических функций при помощи OMP

Словарь	Функция	10 100	100 1000	1000 10^4	10^4 10^5
gabor	x^2	$8.35{\cdot}e^{-5}$	$7.32{\cdot}e^{-6}$	$7.47{\cdot}e^{-6}$	$7.32{\cdot}e^{-6}$
DCT	x^2	$4.1{\cdot}e^{-5}$	$7.32{\cdot}e^{-6}$	$3.83{\cdot}e^{-6}$	$2.15{\cdot}e^{-6}$
B-Spline	x^2	$1.04{\cdot}e^{-5}$	$\mathbf{3.05}{\cdot}e^{-6}$	$\mathbf{8.76}{\cdot}e^{-7}$	$5.14{\cdot}e^{-7}$
Hyp Spline	x^2	$1.01{\cdot}e^{-5}$	$2.95{\cdot}e^{-6}$	$8.75{\cdot}e^{-7}$	$5.12{\cdot}e^{-7}$
Trig Spline	x^2	$1.01{\cdot}e^{-5}$	$\mathbf{2.7}{\cdot}e^{-6}$	$\mathbf{8.68}{\cdot}e^{-7}$	$5.12{\cdot}e^{-7}$
gabor	$arctg(10 \cdot x)$	0.0029	0.0016	0.0016	0.0014
DCT	$arctg(10 \cdot x)$	0.0024	0.00061	$\mathbf{1.92}{\cdot}e^{-5}$	$5.2{\cdot}e^{-6}$
B-Spline	$arctg(10 \cdot x)$	0.00068	$2.65{\cdot}e^{-6}$	$\mathbf{9.99}{\cdot}e^{-7}$	$4.23{\cdot}e^{-7}$
Hyp Spline	$arctg(10 \cdot x)$	0.00067	$2.9{\cdot}e^{-6}$	$9.99{\cdot}e^{-7}$	$4.23{\cdot}e^{-7}$
Trig Spline	$arctg(10 \cdot x)$	0.00068	$2.53{\cdot}e^{-6}$	$\mathbf{9.96}{\cdot}e^{-7}$	$4.22{\cdot}e^{-7}$
gabor	$0,2 \cdot ch(5 \cdot x)$	0.04754	0.00267	0.0022	0.002
DCT	$0,2 \cdot ch(5 \cdot x)$	0.0147	0.00103	0.000142	0.000025
B-Spline	$0,2 \cdot ch(5 \cdot x)$	0.00463	$3.08{\cdot}e^{-6}$	$9.84{\cdot}e^{-7}$	$4.89{\cdot}e^{-7}$
Hyp Spline	$0,2 \cdot ch(5 \cdot x)$	0.00445	$3.07{\cdot}e^{-6}$	$9.74{\cdot}e^{-7}$	$4.9{\cdot}e^{-7}$
Trig Spline	$0,2 \cdot ch(5 \cdot x)$	0.00483	$3.07{\cdot}e^{-6}$	$9.78{\cdot}e^{-7}$	$4.88{\cdot}e^{-7}$

Из приведённых экспериментов видно, что для всех выбранных функций сплайновые словари дали более точный результат, чем другие словари. Разница между словарями, сгенерированными из сплайнов с разными порождающими вектор-функциями, численно не видна, однако в таблице выделены места, где словари из тригонометрических или гиперболических сплайнов показывают результат лучше, чем словари из B-сплайнов.

6.2. Аудиосигналы

Для тестирования на аудиосигналах был подготовлен датасет различных аудиофайлов. В него вошли порядка 12 тысяч файлов формата wav длительностью около 3 секунд, среди них есть записи слов и коротких фраз на английском с фоновым шумом и файлы с представлением гравитационных волн чёрных дыр в wav формате. Датасет доступен по ссылке https://github.com/Duletov/AudioFiles. Результаты, полученные при тестировании точности на аудиофайлах `DC_a01.wav` - `DC\_a20.wav` со скоростью потока 705 кбит в секунду представлены в таблице. Для каждого сигнала указана средняя ошибка точности RMSE.

Таблица 2 - Ошибка RMSE для аппроксимаций аудиосигналов при помощи OMP

Словарь	Ошибка
DCT	0,017817
B-Spline	0,015323
Hyp Spline	0,0144351
Trig Spline	0,014920

Как видно из приведённой выше таблицы, при тестировании на аудиофайлах длиной больше 1 секунды сплайновые словари показывают лучший результат. Разница между словарями, построенными на минимальных сплайнах, оказалась мала.

6.3. Сравнение скорости

Для сравнения скорости работы алгоритмов использовались следующие настройки запуска:

- Функция $arctg(10 \cdot x)$ на отрезке $[0, 1]$
- 3000 атомов, 50 000 точек сетки

Время замерялось от точки входа в алгоритм и до выхода в основное тело программы. Для параллельных версий алгоритмов проходили

замеры с использованием 1, 2 и 4 ядер. Скорости записи ответа в файл и вывода на экран не учитывались. Время замерялось в миллисекундах, в таблице представлено значение в секундах. Запуск повторялся 10 раз, в таблице представлено среднее время и стандартное отклонение $\sigma = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}}$, где n - количество измерений, x_i - измерение времени, $\overline{x}$ - среднее арифметическое.

Таблица 3 - Сравнение скорости работы алгоритмов

Алгоритм	Кол. ядер	Время	Ст. откл.
OMP	-	171,3 с	0,4
OMP параллельная версия	1	174,2 с	0,2
OMP параллельная версия	2	106,3 с	0,1
OMP параллельная версия	4	72,5 с	0,2
SOMP	1	31,4 с	0,1
SOMP	2	26,8 с	0,3
SOMP	4	23,4 с	0,1

Таким образом мы можем сделать вывод, что разработанный алгоритм работает быстрее исходной реализации OMP и её параллельной версии.

7. Заключение

В рамках поставленной цели были выполнены следующие задачи:

1. Изучены работы в области разреженной аппроксимации и использующиеся в них алгоритмы.

2. Подготовлен набор тестовых сигналов для проведения экспериментов и реализован соответствующий модуль обработки.

3. Ускорены используемые алгоритмы при помощи параллельных вычислений.

4. Разработан новый алгоритм разреженной аппроксимации для использования преимуществ сплайновых словарей.

5. Проведены численные эксперименты на подготовленными сигналами и проведена оценка эффективности использования словарей из минимальных сплайнов в задачах разреженной аппроксимации.

Список литературы

[1] О. О. Луковенкова Моделирование и обработка импульсных сигналов геоакустической эмиссии на базе разреженной аппроксимации // диссертация на соискание учёной степени ктн. 2016.

[2] R. Rubinstein, A.M. Bruckstein, M. Elad Dictionaries for Sparse Representation Modeling // Proceedings of the IEEE. 2010. Vol. 98(6).

[3] L. Rebollo-Neira, A. Plastino Sparse representation of Gravitational Sound // Journal of Sound and Vibration. 2007. Vol. 417, pp. 306-–314.

[4] M. Elad Sparse and redundant representations // From theory to applications in signal and image processing. 2010. 376 p.

[5] M. Andrecut Sparse random approximation and lossy compression // IAENG International Journal of Computer Science. 2011. Vol. 38, pp. 205–214.

[6] L. Rebollo-Neira A dedicated greedy pursuit algorithm for sparse spectral representation of music sound // Journal of the Acoustical Society of America. 2016. Vol. 140, no. 4, pp. 2933–2943.

[7] P. Breen Algorithms for Sparse Approximation // Year 4 Project, School of Mathematics University of Edinburgh. 2009.

[8] M. Unser, E. Soubies, F. Soulez, M. McCann, L. Donati GlobalBioIm: A Unifying Computational Framework for Solving Inverse Problems // Imaging and Applied Optics 2017 (3D, AIO, COSI, IS, MATH, pcAOP). 2017.

[9] L. Rebollo-Neira, J. Bowley, A.G. Constantinides, A. Plastino Self Contained Encrypted Image Folding // Physica A: Statistical Mechanics and its Applications. 2012. 391 p.

[10] M.A. Hameed Comparative analysis of Orthogonal Matching Pursuit and Least Angle Regression // Michigan State University, Master of Science Thesis. 2012.

[11] D. Donoho, Y. Tsaig, I. Drori, J.L. Starck Sparse Solution of Underdetermined Linear Equations by Stagewise Orthogonal Matching Pursuit // IEEE Transactions on Information Theory. 2006. Vol. 58(2), pp. 1094–1121.

[12] L. Pu, Z. Jiangtao, X. Kewen, Z. Qiao, H. Ziping Research on Improvement of Stagewise Weak Orthogonal Matching Pursuit Algorithm // Pattern Recognition and Image Analysis. 2019. Vol. 29(4), pp. 613-–620.

[13] Y. Zhang, G. Sun Stagewise Arithmetic Orthogonal Matching Pursuit // International Journal of Wireless Information Networks. 2018. Vol. 25(2), pp. 221–228.

[14] S. Yao, Q. Guan, S. Wang Fast sparsity adaptive matching pursuit algorithm for large-scale image reconstruction // Journal of Wireless Communications and Networking. 2018. Vol. 78.

[15] J. Xiang, H. Yue, Y. Xiangjun Orthogonal Matching Pursuit Algorithm Via Improved Matching Criterion // Proceedings of the 2nd International Conference on Digital Signal Processing - ICDSP. 2018. P. 27–31.

[16] http://www.nonlinear-approx.info/code/matlab/uniform.html, дата обращения 15.05.2022

[17] M. Andrle, L. Rebollo-Neira Cardinal B-spline dictionaries on a compact interval // Applied and Computational Harmonic Analysis. 2005. Vol. 18(3), pp. 336–346.

[18] S. Chu, S. Narayanan, C.-C.J. Kuo. Environmental Sound Recognition With Time–Frequency Audio Features // IEEE Transactions On Audio, Speech And Language Processing. 2009. Vol. 17(6), pp. 1142 –1158.

[19] R. Tibshirani Regression shrinkage and selection via the lasso // Journal of the Royal Statistical Society. Series B (Methodological). 1996. P. 267–288.

[20] Z. Chen, W. Yang, J. Li, T. Yi, J. Wu, D. Wang Bridge influence line identification based on adaptive B-spline basis dictionary and sparse regularization // Struct Control Health Monitoring. 2019. Vol. 26(6).

[21] И.Г. Бурова, Ю.К. Демьянович Минимальные сплайны и их приложения // Издательство СПбГУ, 2010. 365 с.

[22] Ю.К. Демьянович, А.А. Макаров Необходимые и достаточные условия неотрицательности координатных тригонометрических сплайнов второго порядка // Вестник СПбГУ. Математика. Механика. Астрономия. 2017. Т. 4(62), вып. 1, стр. 9–16.

[23] Е.К. Куликов, А.А. Макаров Об аппроксимации гиперболическими сплайнами // Записки научных семинаров ПОМИ. 2018. Т. 472, стр. 179–194.

[24] O. Kosogorov, A. Makarov On Some Piecewise Quadratic Spline Functions // Lecture Notes In Computer Science, Numerical Analysis and Apolications. 2017. P. 448–-455.

[25] Ray Maleh Improved RIP Analysis of Orthogonal Matching Pursuit // L-3 Communications Mission Integration Division. 2011.

[26] http://www.nonlinear-approx.info/examples/node2.html, дата обращения 15.05.2022

[27] Pere L. Gilabert, D. Lopez-Bueno, G. Montoro Spectral Weighting Orthogonal Matching Pursuit Algorithm for Enhanced Out-of-Band Digital Predistortion Linearization // IEEE Transactions on Circuits and Systems II: Express Briefs. 2019. P. 1277 - 1281.

[28] L. Rebollo-Neira, I. Sanches A simple scheme for compressing sparse representation of melodic music // Electronics letters. 2016. Vol. 00.

[29] M. Suchithra, P. Sukanya, P. Prabha, O.K. Sikha, V. Sowmya, K.P. Soman An experimental study on application of Orthogonal Matching Pursuit algorithm for image denoising // 2013 International Mutli-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s). 2013. P. 729-736.

[30] L. Rebollo-Neira, J. Bowley Sparse Representation of Astronomical Images // J. Opt. Soc. Am. A 30. 2013. P. 758-768.

[31] Д.Е. Дулетов Анализ эффективности применения сплайновых словарей в задачах разреженной аппроксимации // Курсовая работа, Программная инженерия, СПбГУ. 2021.

[32] S. Mallat and Z. Zhang Matching pursuit with time-frequency dictionaries // IEEE Transaction on Signal Processing. 1993. Vol. 41, pp. 3397-3415.