Инженерный вестник Дона, №1 (2025)
ivdon.ru/ru/magazine/archive/n1y2025/9760

Реализация адаптации экстремальной фильтрации к реальному времени

А.А. Приймак

Пензенский государственный университет, Пенза

Аннотация: В работе описывается метод экстремальной фильтрации и подходы автора, позволяющие адаптировать его к работе в режиме реального времени — покадровая обработка и метод с подгрузкой сигнала. Далее, предоставляются решения, с помощью которых возможно реализовать перечисленное на реальных устройствах, основанные на протоколах HTTP, WS и библиотеке Multiprocessing. После приводятся результаты, а также делаются выводы о пригодности подходов и решений автора к работе на реальных устройствах.
Ключевые слова: экстремальная фильтрация, метод покадровой обработки сигнала, метод с подгрузкой значений, Multiprocessing, HTTP, WebSocket, REST, JSON, Python, микроконтроллеры, одноплатные компьютеры.

Введение

В системах диагностики и распознавания, для оценки текущего состояния, используют параметры, вычисляемые по частотам и амплитудам составляющих сигнала. Для параметров формируют диагностические признаки, свидетельствующие о разных событиях и процессах в системе.

Ранее, для анализа сигнала, применяли методы спектрального анализа, параметрические методы, например метод Прони [1,2] или метод Писаренко. Однако, они имеют высокую трудоёмкость, что плохо для систем, функционирующих в режиме реального времени (РВ).

Метод экстремальной фильтрации (ЭФ) [3-5] позволяет получить составляющие сигнала и рассчитать для них параметры быстрее, так как он проще аналогов. ЭФ лучше подходит для систем РВ, где нужна минимальная задержка в получении результатов.

ЭФ используется относительно давно, она предполагает: анализ экстремумов, расчёт знакопеременных и сглаженных составляющих сигнала. Знакопеременные и сгоаженные составляющие вычисляются по формулам: $x_{pi} = -0.25x_{эi-1} + 0.5x_{эi} – 0.25x_{эi+1}$ и $x_{ci} = 0.25x_{эi-1} + 0.5x_{эi} + 0.25x_{эi+1}$, соответственно.

Изначально, метод ЭФ рассчитан на работу с конечными данными. Чтобы использовать его для данных, поступающих в режиме РВ, автором предложены специальные методы адаптации, одним из которых является покадровая обработка сигнала (ПКД) [3], а другим — метод с подгрузкой значений (ПДЗ).

ПКД предполагает одновременное накопление значений поступающего сигнала и их анализ, с применением ЭФ. Условием работоспособности ПКД, является не превышение времени анализа буфера над временем накопления в него значений ($t_{буф} > t_{аб}$). Если условие не выполняется, то решением может быть разделение буфера на участки меньшей длины.

Другим методом, позволяющим адаптировать ЭФ к режиму РВ, является метод ПДЗ («скользящее окно» или *FIFO*). Для данного метода, условие нормальной работы состоит в не превышении времени анализа одного значения сигнала ($t_{аз}$) над шагом дискретизации ($dt > t_{аз}$).

Актуальным является вопрос реализации описанных подходов на реальных устройствах. С учётом использования языка *Python 3.11*, автором предложены несколько вариантов решения задачи.

1. Параллелизм с библиотекой Multiprocessing

Метод ПКД требует обеспечения одновременного накопления значений и их анализа. Для этого можно использовать пакеты *Multiprocessing, Multithreading* [6,7] или библиотеку *asyncio*.

Выполняемые программой операции, можно разделить на две группы: *CPU — bound* и *I/O — bound*. Время выполнения первых сильно зависит от мощности процессора, а время выполнения вторых, главным образом, определяется продолжительностью операций ввода — вывода.

В рамках решаемой задачи, рационально использовать пакет *Multiprocessing*, в виду большого количества операций типа *CPU — bound*.

Пакет *Multiprocessing* позволяет «обойти» глобальную блокировку интерпретатора — *Gil*, которая есть в языке *Python*. Эта блокировка не даёт выполнять одновременно несколько потоков, входящих в один процесс.

Минусом использования библиотеки *Multiprocessing* являются высокие требования к мощности процессора. Хотя, если устройство имеет несколько ядер, на каждом из которых, параллельно, выполняется свой процесс, то быстродействие возрастает. Но, рост не пропорционален количеству ядер.

При работе программы, написанной автором, получены результаты, которые представлены в таблице 1. Программа ориентирована на выделение двух, самых высокочастотных, составляющих сигнала.

2. Параллелизм с архитектурой клиент-сервер и протоколом HTTP

Обеспечить одновременное накопление сигнала и его анализ, для ПКД, можно также с помощью, создания приложения, использующего клиент — серверную архитектуру, с обменом информацией по протоколу *HTTP 1.1*.

Клиентская и серверная часть могут находиться на, физически разных, устройствах. «Клиентами» могут быть, например, микроконтроллеры (с поддержкой WiFi или LAN), с подключенными к их АЦП датчиками, а сервером — персональный компьютер, на который стекаются данные.

Передачу можно построить на принципах *REST* и отправлять данные с клиентской части, в формате *JSON*, с помощью простых, асинхронных, POST – запросов. Асинхронность важна для параллелизма, она позволяет не ожидать ответа на запрос от сервера, продолжая выполнение программы.

Для достижения параллельности накопления и обработки, в клиентской части, следует предусмотреть буфер, в котором будут накапливаться значения сигнала. Во время его заполнения, серверная часть осуществляет анализ, с использованием метода ЭФ. По окончанию заполнения буфера на клиенте, происходит передача данных серверу и очистка буфера.

Автором, на языке *Python* (с библиотеками *FastAPI Asyncio*, *HTTPx*, *JSON* [8,9]), создано приложение использующее протокол *HTTP*. В таблице 1 и на рисунке 1 представлены результаты работы этого приложения.

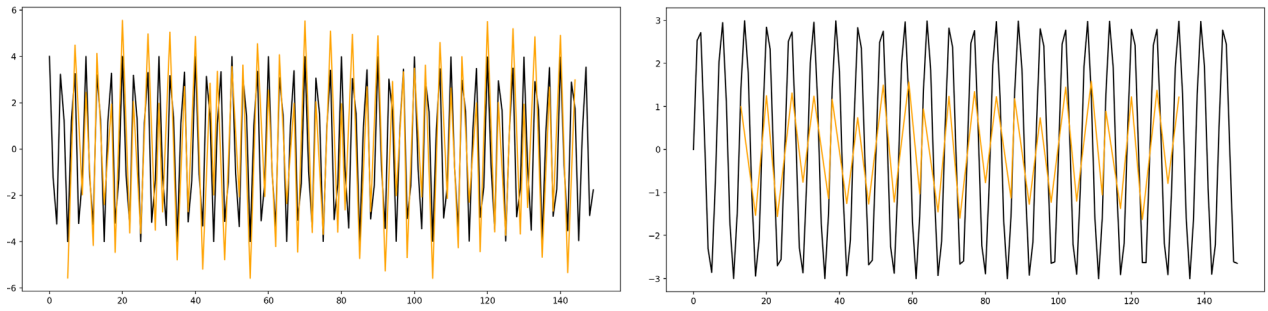

Рис. 1 — первая (слева) и вторая (справа): ЭФ, ПКД, HTTP

Показаны первая (слева) и вторая (справа), высокочастотные составляющие, выделенные из фрагмента сигнала, находящегося в буфере длинною $n = 150$ точек. Видно совпадение известных составляющих с выделенными, что говорит о правильности работы.

3. Параллелизм с архитектурой клиент-сервер и протоколом WS

Протокол *WebSocket (WS)* [10], так же как и *HTTP*, может быть полезен при реализации методов автора в виде клиент — серверного приложения. В отличии, от *HTTP*, протокол *WS* можно использовать и для метода с подгрузкой значений сигнала, а не только для реализации покадровой обработки. Причина этого состоит в том, что метод ПДЗ, предполагает постоянную отправку единичных значений — отправлять, беспрерывно, *POST* – запросы на сервер, с содержимым в одно значение, не рационально.

Автором, на языке *Python* (с библиотеками *WebSockets* и *JSON*), создано два приложения, использующих протокол *WS*. Первое из них реализует метод с ПДЗ, а второе — метод ПКД.

В качестве «клиентов», аналогично протоколу HTTP, могут выступать микроконтроллеры, одноплатные компьютеры и другие устройства в сети.

Для приложения, реализующего метод с ПДЗ на протоколе *WS*, можно заметить, что время, которое требуется на передачу одного значения, составляет около $t_n = 10{,}1$ *мс*, а время затрачиваемое на анализ этого же значения, составляет, около $t_{аб}, = 1$ *мс*. Таким образом, условие работоспособности, для метода с ПДЗ, на протоколе *WS*, соблюдается.

В таблице 1 и на рисунке 2 даны результаты работы ЭФ, адаптированной к РВ, с помощью метода с ПДЗ, с использованием протокола *WS*. Показаны первая (слева) и вторая (справа), высокочастотные составляющие, выделенные из фрагмента сигнала, находящегося в скользящем окне, длинною $n = 150$ точек. Видно совпадение известных составляющих с выделенными, что говорит о правильности работы.

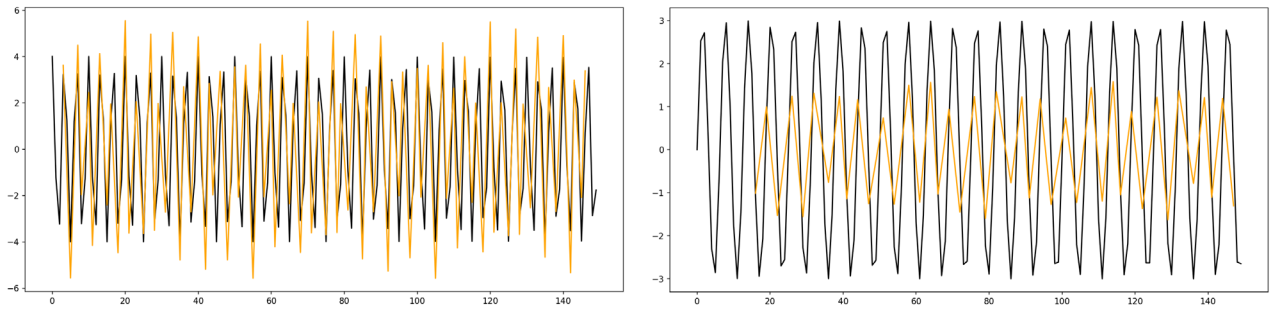

Рис. 2 — Первая (слева) и вторая (справа): ЭФ, ПДЗ, WS

В первой колонке таблицы 1 указана длина буфера — n. Во второй колонке — расчётные величины времени, потраченного на накопление значений в буфер и их передачу, при $dt = 0.01$, ($t_{буфр} = dt \cdot n$). В последующих колонках, для каждого из подходов, приведены, полученные на практике, величины времени, затрачиваемого: на накопление и передачу — $t_{буфп}$, на анализ поступивших из буфера значений — $t_{аб}$ (с помощью метода ЭФ).

Время анализа значений ($t_{аб}$) одинаково, так как, время, затрачиваемое на анализ сигнала, с помощью ЭФ, аналогично, потому что одинакова реализация самого метода ЭФ и количество выделяемых составляющих.

Таб. 1 — Результаты работы программ

n	$t_{буфр}$, c	*Multiprocessing, покадровая*		*HTTP, покадровая*		*WebSocket, покадровая*	
		$t_{буфп}$, c	$t_{аб}$, c	$t_{буфп}$, c	$t_{аб}$, c	$t_{буфп}$, c	$t_{аб}$, c
100	1	1,062056	0.000998	1,0761	0,000998	1,044128	0.000998
200	2	2,11077	0,001003	2,1234	0,001003	2,090541	0,001003
300	3	3,181699	0,001005	3,1764	0,001005	3,131814	0,001005
400	4	4,24	0,001011	4,2002	0,001011	4,176071	0,001011
500	5	5,288448	0,001017	5,2199	0,001017	5,230422	0,001017
600	6	6,374163	0,001022	6,2452	0,001022	6,286143	0,001022
700	7	7,40657	0,001027	7,2648	0,001027	7,324854	0,001027
800	8	8,467249	0,001501	8,2745	0,001501	8,350371	0,001501
900	9	9,504497	0,001810	9,2981	0,001810	9,421516	0,001810

4. Зависимость времени анализа от количества составляющих

Рассматривая, созданные автором, программы, может возникнуть вопрос о соблюдении условий работоспособности методов адаптации, при возрастании количества выделяемых составляющих.

Для систем обнаружения достаточно выделения двух, самых высокочастотных, составляющих. Однако, при решении ряда задач, может возникнуть необходимость получения всех составляющих сигнала, иначе не будет возможности чёткого понимания о быстропеременых процессах, протекающих в системе.

По результатам замеров времени, затрачиваемого на анализ сигнала, с помощью метода ЭФ, при разном количестве составляющих, была составлена таблица 2. В ней отображены величины времени (t_a), которые потребовались для выполнения процедуры анализа участка сигнала, длиной $n = 900$, при выделении одной, двух, трёх или четырёх составляющих. Шаг дискретизации, между поступающими значениями, составил $dt = 0.01$ c.

Из таблицы видно, что сколько бы не выделялось составляющих, время затрачиваемое на анализ значений мало, относительно величин времени, требуемого на их получение. Запас позволяет уменьшить *dt* или увеличить *n*.

Таб. 2 — Время анализа *n* – составляющих

Количество составляющих, n	1	2	3	4
t_a, *c*	0.001012	0.001782	0.001989	0.002622

Выводы

Результатом работы является успешная реализация методов автора по адаптации ЭФ к режиму РВ. Это подтверждается соблюдением условий работоспособности, как для метода покадровой обработки, так и для метода с подгрузкой сигнала — анализ быстрее накопления.

Большое практическое значение имеют приложения, использующие протоколы *HTTP* и *WS*, потому что они решают проблему передачи информации с датчиков на сервер, где ведётся мониторинг состояния системы.

Особый интерес вызывают клиент — серверные приложения на протоколе *WS*. Они обладает двухсторонней связью, с возможностью инициативы от клиента. То есть, клиенту не нужно постоянно отправлять запрос на сервер, чтобы получить от него какую — либо информацию. В любой момент, сервер может сам отправить сообщение в канал, оказав, таким образом, управляющее воздействие на клиент, к которому могут быть подключены различные исполнительные механизмы.

Также, протокол *WS* хорошо подходит для реализации метода с подгрузкой сигнала. Преимуществом этого метода, перед покадровой обработкой. выражено в том, что параметры составляющих, в последнем, могут быть получены лишь после полного заполнения буфера, а в первом — они доступны сразу.

Литература

1. Иосифов В.П., Гулынина Е.В., Иосифова Л.Г. Модифицированный метод оценки спектральных характеристик с применением дискретных преобразований Уолша и метода параметрического спектрального анализа Прони // Инженерный вестник Дона. 2019. №5. URL: ivdon.ru/ru/magazine/archive/N5y2019/5963.

2. Мясникова Н.В., Дудкин В.А. Использование метода Прони для анализа сейсмических сигналов идущего человека // Известия вузов. Поволжский регион. Технические науки. 2009. №4.

3. Лысова Н. В., Мясникова Н. В. Регулирование веса бумажного полотна на основе метода экстремальной фильтрации для систем реального времени с помощью покадровой обработки сигнала // Инженерный вестник Дона. 2019. №5. URL: ivdon.ru/ru/magazine/archive/N5y2019/5882.

4. Мясникова Н.В., Берестень М.П. Разложение на эмпирические моды на основе экстремальной фильтрации // Цифровая обработка сигналов. 2014 №4. с. 13 — 17.

5. Мясникова Н.В., Берестень М.П., Цыпин Б.В. Экспресс-анализ сигналов в инженерных задачах // М.: Физматлит, 2016. 184 с.

6. Lambertson Christine Mastering python: A comprehensive guide to programming. Lulu.com, 2023. 280 P. ISBN 9781312513518.

7. Яворски, Зиаде. Python. Лучшие практики и инструменты // СПб.: Питер, 2021. 560 с.

8. Abdulazeez Abdulazeez Adeshina. Building python web APIs with FastAPI: A Fast-paced guide to building high-performance, robust web APIs with very little boilerplate code. Packt Publishing, 2022. 218 P. ISBN 9781801074513, 1801074518.

9. Сигалов Д.И., Жабицкий М.Г. Создание системы непрерывного анализа качества трансляции с камер видеонаблюдения на строительных площадках

через алгоритм детектирования Кэнни // International Journal of Open Information Technologies. Vol. 12, No. 8. 2024. pp. 94-104.

10. Rodrigues Diego Manual of python for web development 2024, Edition: Modern frameworks and tools. N.p., 2024. 226 P.

References

1. Iosifov V.P., Gulynina Ye.V., Iosifova L.G. Inzhenernyj vestnik Dona. 2019. №5(56). URL: ivdon.ru/ru/magazine/archive/N5y2019/5963.

2. Myasnikova N.V., Dudkin V.A., Izvestiya vuzov. Povolzhskiy region. Tehnicheskie nauki [University news. Volga region. Technical science]. 2009. №4.

3. Lysova N.V., Myasnikova N.V. Inzenernyj vestnik Dona. 2019. №5(56). URL: ivdon.ru/ru/magazine/archive/N5y2019/5882.

4. Myasnikova N.V., Beresten M.P. Tsifrovaya obrabotka signalov [Digital signal processing]. 2014 №4. pp. 13-17.

5. Myasnikova N.V., Beresten' M.P., Tsypin B.V. Ekspress-analiz signalov v inzhenernykh zadachakh [Express analysis of signals in engineering problems]. M.: Fizmatlit, 2016. 184 P.

6. Lambertson Christine Mastering python: A comprehensive guide to programming. Lulu.com, 2023. 280 P. ISBN 9781312513518.

7. Yavorski, Ziade Python. Luchshiye praktiki i instrumenty [Best Practices and Tools]. SPb.: Piter, 2021. 560 P.

8. Abdulazeez Abdulazeez Adeshina. Building python web APIs with FastAPI: A Fast-paced guide to building high-performance, robust web APIs with very little boilerplate code. Packt Publishing, 2022. 218 P. ISBN 9781801074513.

9. Sigalov D.I., Zhabitskiy M.G. International Journal of Open Information Technologies. Vol. 12, No. 8. 2024. pp. 94-104.

10. Rodrigues Diego Manual of python for web development 2024, Edition: Modern frameworks and tools. N.p., 2024. 226 P.

Дата поступления: 10.11.2024

Дата публикации: 28.12.2024