1. Метод вычисления Time of Flight

Методы измерения ToF являются важным инструментом для ультразвуковой томографии, особенно для клинического применения.

Существует математический метод, разработанный для таких целей. Он называется методом информационного критерия Акаике [1].

Метод позволяет обнаружить точку (AIC ToF), на которой происходит момент заметного возрастания значений в некотором окне заранее определённой ширины. Сама точка определяется через вычисление сумм натуральных логарифмов выборочных дисперсий двух пограничных сегментов окна, движущихся в противоположном направлении. Центр окна предварительно определяется максимальным амплитудным значением во входных данных.

Расчёт AIC производится по формуле:

$$AIC(k) = k \cdot log\left(var\big(S(1,k)\big)\right) + (N-k) \cdot log\left(var\big(S(k+1,N)\big)\right) \quad (1)$$

, где S(1,k) и S(k+1,N) – сегменты, полученные разбиением точкой k.

var() – выборочная дисперсия сегментов S:

$$var\big(S(i,j)\big) = \delta_{j-1}^2 = \frac{1}{j} \cdot \sum_{l=i}^{j} (S(l) - \bar{S})^2 \, , i \le j; i,j = 1..N \quad (2)$$

, где $\bar{S}$ – среднее значение S(i, j) сигнала на промежутке от i до j.

Точка минимума AIC выбирается в качестве искомого ToF, позиция которой соответствует позиции во входных данных и вычисляется по формуле:

$$ToF = \min\big(AIC(k)\big) \text{, при } 1 \le k \le N \ (3)$$

2. Используемые инструменты

В рамках данной работы в целях апробации и тестирования математических методов был использован дистрибутив Python Anaconda3, включающий в себя набор свободно распространяемых программных библиотек. Основная особенность дистрибутива — оригинальный менеджер разрешения зависимостей conda с графическим интерфейсом Anaconda Navigator, что позволяет отказаться от стандартных менеджеров пакетов (таких, как pip для Python). Дистрибутив скачивается единожды, и вся последующая конфигурация, в том числе установка дополнительных модулей, может проводиться в offline режиме.

Среда разработки Jupyter Notebook, входящая в состав дистрибутива, запускается в как локальный сервер, к которому можно подключиться из веб-браузера. Приложение Jupyter Notebook хранит в одном файле программный код, графики, комментарии, изображения, формулы, что позволяет в реальном времени легко и удобно писать и отлаживать программы.

В данной работе использовалось расширение NumPy, добавляющее поддержку больших многомерных массивов и матриц, вместе с большой библиотекой высокоуровневых математических функций для операций с этими массивами.

3. Определение входных данных

Метод информационного критерия Акаике в ультразвуковой томографии является основным и несложен в реализации в виде алгоритма.

В ходе работы после анализа графиков выяснилось, что оптимальная ширина окна, в котором производится поиск ToF, равна 120 тактам, так как в большинстве случаев сигнал имеет ровно такую продолжительность. Это значение в дальнейшем определяет ширину окна в алгоритме ToF, реализуемом на ПЛИС.

Программа считывает значения сигналов типа integer шириной 16 бит из файла с экспериментальными данными размером 31ГБ, полученными с датчиков после сканирования фантомного объекта. Целочисленный тип данных хорошо подходит для вычислений на ПЛИС и был использован в текущей работе.

В файле бинарного формата располагаются потоковые данные с одного среза, в котором производилось 2048 проходов излучения. На каждый такой проход записано 3750 значений с 2048 приёмников.

Так как файл эксперимента содержит значения, записанные из потока данных, то из них выбирается определённый срез, позиция которого определяется номером излучателя и датчика. Затем с помощью np.reshape срез распределяется по ячейкам массива test_data и транспонируется функцией np.transpose.

С помощью функции np.amax вычисляется максимальное значение массива test_data. Это значение определяет индекс, который ищется итеративным обходом по массиву. Затем делается вырезка, равная окну в 128 значений, центр которой определяется вычисленным индексом и записывается в массив values, который подаётся на вход вычислений AIC.

Вычисления AIC проходят в цикле с числом итераций imin+1...imax-1, где imin и imax – минимальный и максимальный размер массива values, исходя из того, что по методу Акаике (1) в дисперсии включены индексы 1 и N.

В цикле производятся вычисления левой и правой дисперсии с помощью функции np.var. Затем от каждой дисперсии берётся логарифм функцией np.log, результат умножается на коэффициенты k и N-k-1 (по формуле (1)) и суммируется. Каждое из полученных значений записывается в массив aic, одновременно с чем сохраняются в переменные последнее определённое минимальное значение minAIC и текущий индекс. По окончанию проходов цикла, последний индекс, сохранённый вместе со значением minAIC определяет точку на графиках AIC и входных данных

Результаты вычислений на данных одного из датчиков приведены на следующих графиках:

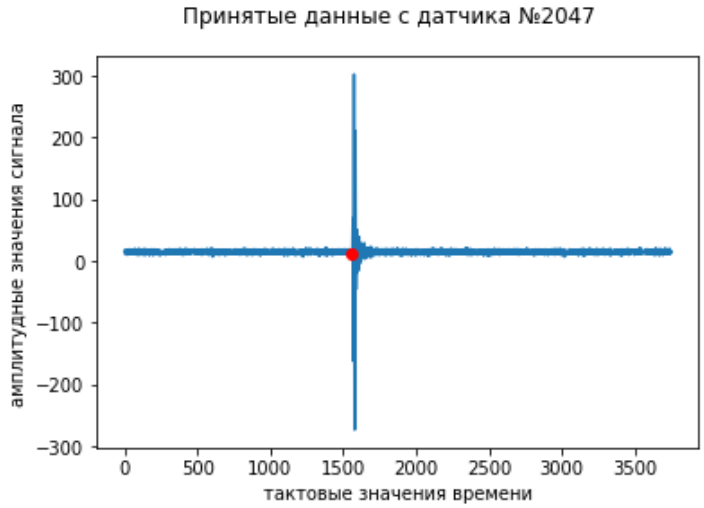
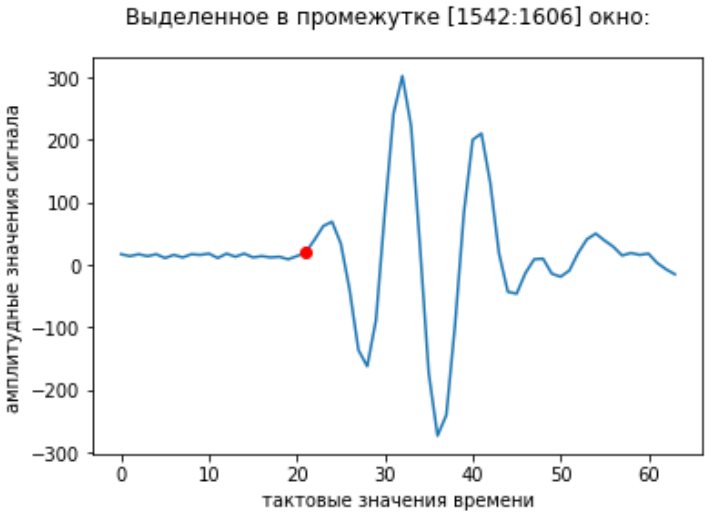
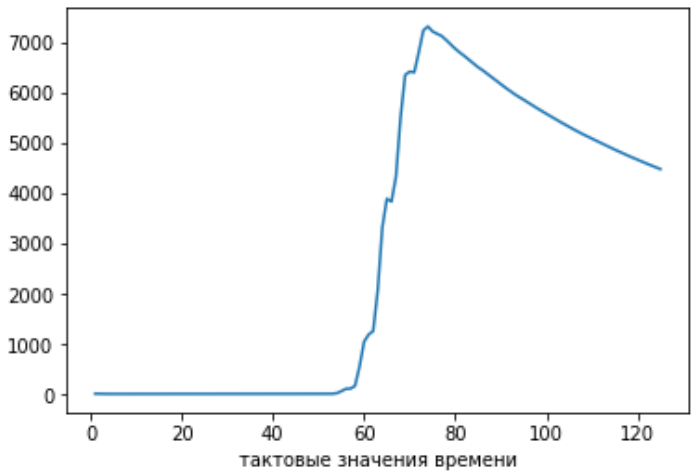


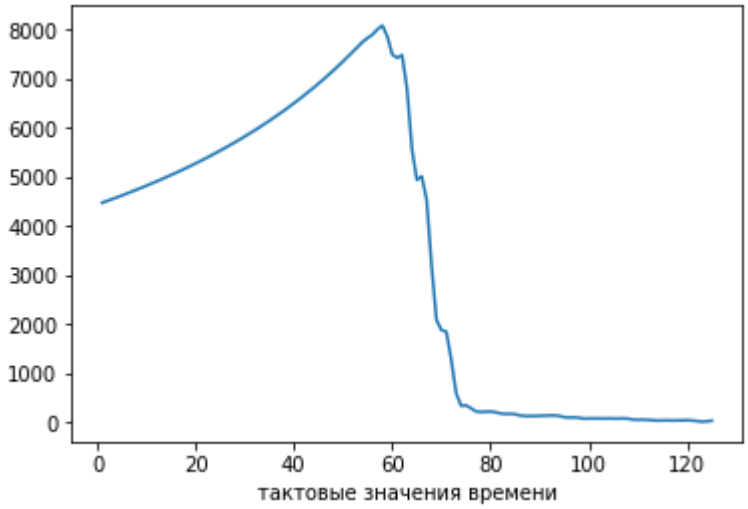
График значений входных данных



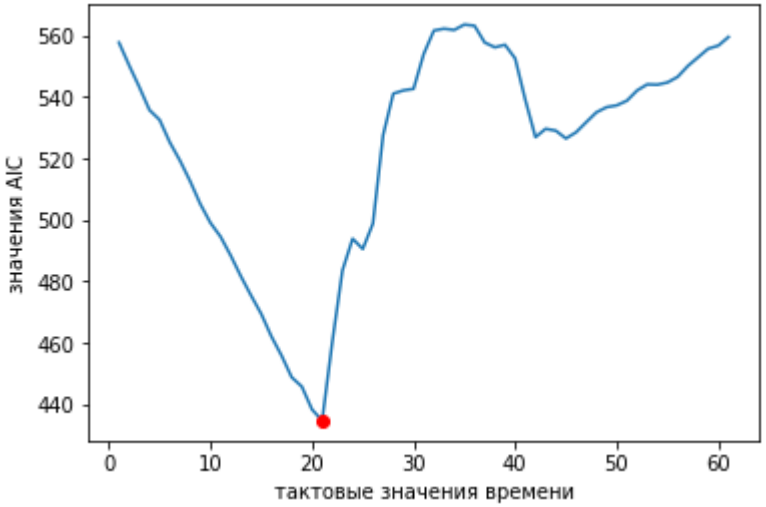
Выделенное по максимальному значению окно



Вычисленные значения левой дисперсии



Вычисленные значения правой дисперсии



Вычисленные программой значения AIC

В результате можно заметить, что среди входных данных обнаруживается значительно превосходящий фоновый шум пик сигнала, данные окна, выделенного на позиции этого пика не превосходят 300 единиц, а график AIC имеет хорошо выраженный минимум, по которому можно легко определить min AIC.

Ввиду использования в программе математических библиотек, с параметрами, позволяющими производить вычисления с низкой погрешностью, полученные значения можно считать эталонными и ими можно пользоваться в дальнейшем.

4. Преобразование выборочных дисперсий метода AIC ToF

Поскольку вычисления происходят в цикле с использованием функций библиотеки NumPy, важно заметить, что выборочная дисперсия в формуле (2) использует среднее значение $\bar{S}$, которое делает невозможным потоковую обработку данных до окончания их поступления. Такие задачи решаются преобразованием выражения, в котором есть среднее, по формуле сокращённого умножения, что позволяет привести исходное выражение к потоковым суммам. В итоге, выборочные дисперсии в формуле (1) можно заменить выражениями:

$$var\big(S(1,k)\big) = \left(\frac{\sum_{l=1}^{k}\big(S(l)\big)^2}{k} - \frac{\big(\sum_{l=1}^{k} S(l)\big)^2}{k^2}\right) \quad (3)$$

$$var\big(S(k+1,N)\big) = \left(\frac{\sum_{l=1}^{k+1}\big(S(l)\big)^2}{N-k+1} - \frac{\big(\sum_{l=1}^{k+1} S(l)\big)^2}{(N-k+1)^2}\right) \quad (4)$$

В программе предварительно вычисляются суммы $\sum_{l=1}^{k+1} S(l)$ и суммы квадратов $\sum_{l=1}^{k+1}\big(S(l)\big)^2$, которые записываются в массивы runningSum и runningSumSq соответственно. Значения из этих массивов используются в следующем цикле при вычислении выборочных дисперсий, соответствующих формулам (3) и (4). Т.е. происходит деление сумм на коэффициенты, определяющиеся номером итерации и вычитание полученного этим делением результата. Все значения используют тип данных np.float.

После чего необходимо было проверить точность вычислений с таким подходом с помощью сравнения с эталоном. Для большей наглядности можно проверять на сильно зашумлённых данных.

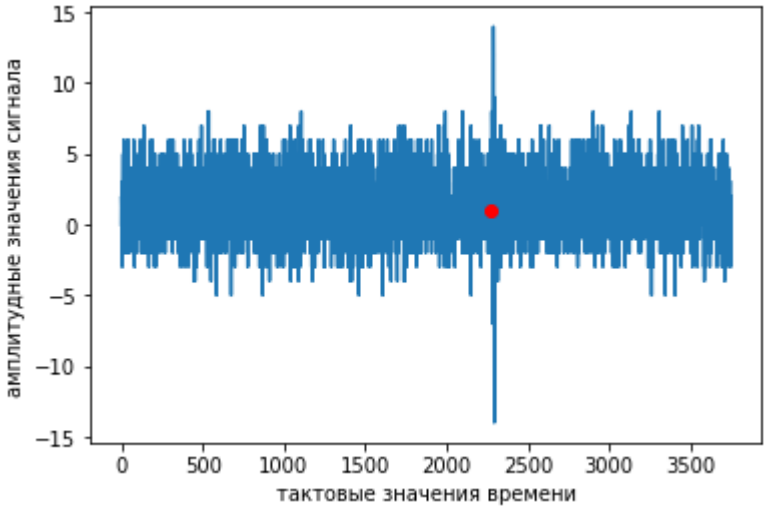


График зашумленных входных данных

Выбранное из данных окно

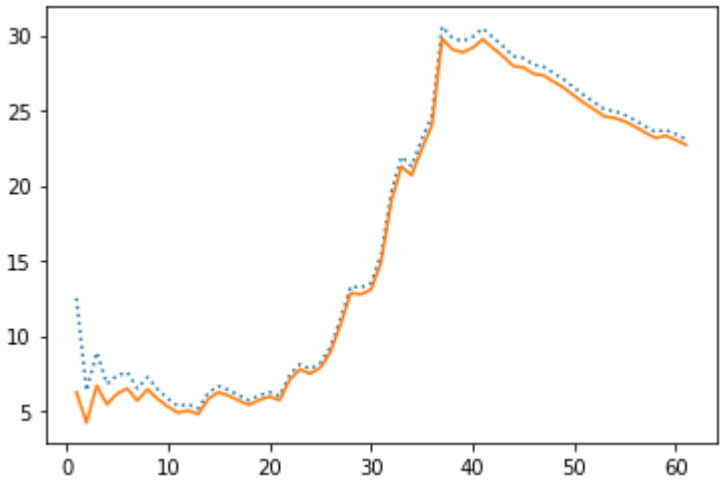


График левой дисперсии

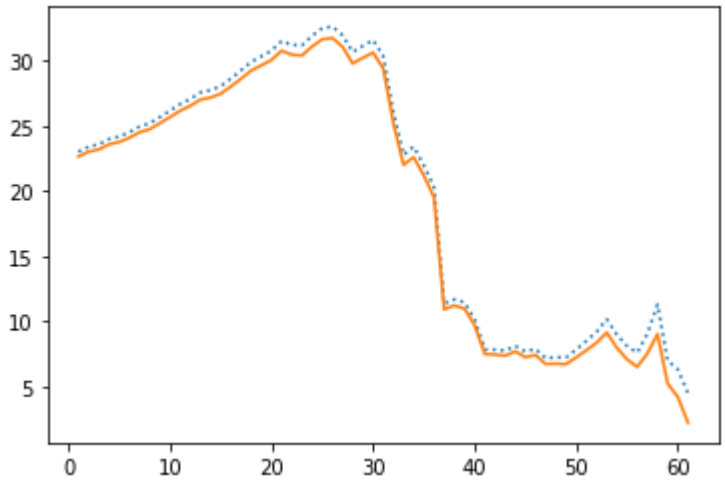


График правой дисперсии

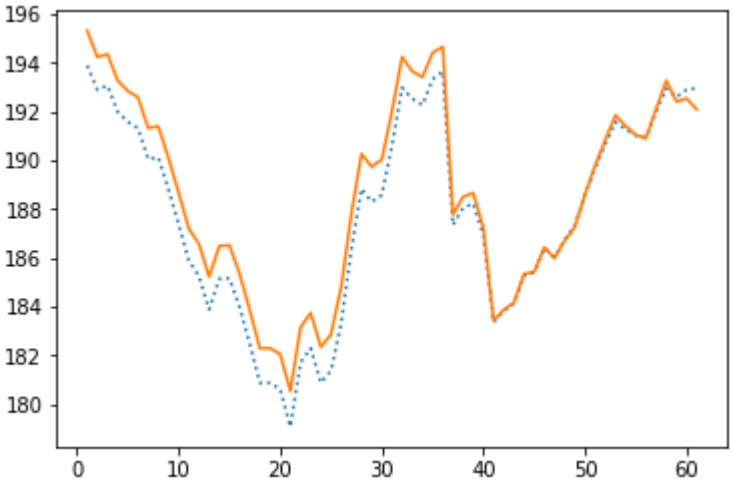


График значений AIC

Алгоритм вычисления преобразованных дисперсий в сравнении с эталонными значениями выдаёт схожий по форме график значений AIC, несмотря на небольшое отклонение по оси Y и учитывая сильную зашумлённость входных данных, что не ухудшает точность определения AIC ToF.