Initial Findings regarding EmNet CSO ID Tool

Introduction:

This report discusses initial efforts at developing a software tool that searches EmNet's CSO MySQL database for wet and dry CSO overflow events. The purpose of this report is to inform EmNet of the proposed algorithms, to provide some initial data regarding how well they work, and to present plans for future development of the software tool.

The purpose of the software tool under development (CSOIDtool) is to automatically probe EmNET's MySQL database for potential anomalies in the data. The particular anomalies of concern are Dry CSO events and sensor failures. To achieve this purpose a C++ program was written using Visual Studio 2005. This program connects to the EmNet database, extracts the sensor data for a specified group of sensors over a specified time interval, and then identifies time intervals over which the flow appears to be anomalous.

2.0 Program (CSOIDtool) Description

The current program prototype examines data from CSO39 sensors S1 and S3 from mid November 2008 to the end of June 2009. This particular set of sensors monitors a pumping station at CSO39. This system is particularly easy to start with because it has a single input and single output so that overflow events are relatively easy to identify directly by inspecting the data. The geometry for this set of sensors is shown below in figure 1.

The CSOIDtool program was written in C++ using Microsoft's Visual Studio 2005 development program. The inputs to the program are

1) Set of "input" sensors, specified by name. In the case of the CSO39 example, the input sensor is `G.21.1 S3`.
2) Set of "output" sensors, specified by name. In the case of the CSO39 example, the output sensor is `G21.1 S1`.
3) An interval of time specified by a start date-time and stop date-time. For the CSO39 example the start date was 2008-11-25 0:0:0 and the stop date was 2009-6-25 0:0:0.

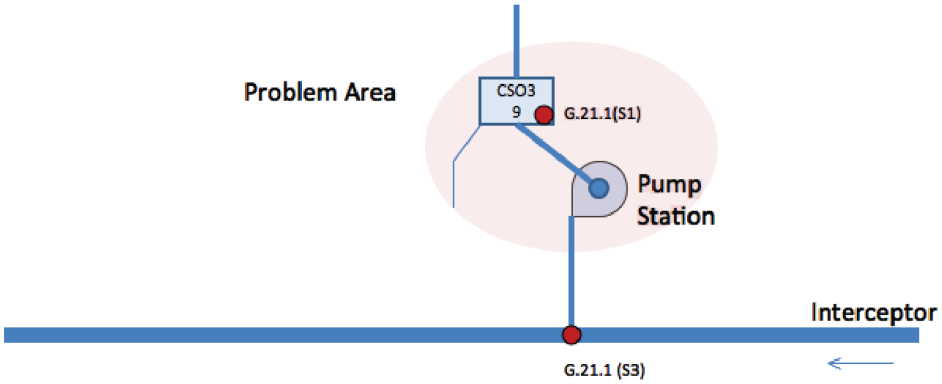


Fig. 1. Pumping Station at CSO39

The outputs generated by the program are

1) A listing on the program console summarizing the program's findings. These findings are shown in the appendix (`console_output.txt`). It simply lists the number of events identified in the output and input sensor histories. Each output event is then classified as being either a DRY or WET overflow depending upon whether or not the output event can be correlated to an appropriate input event.
2) Two files (`output_meas.txt` and `input_meas.txt`) which contain a dump of the raw and processed sensor data from the output and input nodes, respectively.
3) Two files (`CSO_resultINx.txt` and `CSO_resultOUTx.txt`) containing a dump of the identified CSO event's sensor data for its output and input node's, respectively. A pair of files is generated for each CSOevent identified by the program. In the case of this example (see appendix), 20 CSOevents were identified.

The program CSOIDtool is organized as follows. The program first connects to the EmNET database. It then queries the database, using the names of the input/output senosrs to obtain the sensor ID numbers, scale and offset coefficients. The results from this query are then used to initialize a data structure holding the data characterizing the sensor parameters. The program uses the specified start and stop dates to query the database for the specified sensor data. When fetching the raw sensor data, the program automatically uses a syntactical pattern recognition algorithm (described below) to classify the sensor history into intervals of "WET" and "DRY" flows. The resulting "classified" sensor data is then stored in a data structure. The program then

correlates the input and output data streams to identify wet and dry CSO overflow events. Each CSO overflow event is then written to a file. Currently we're using Matlab to plot the program outputs.

The first step in identifying CSO events involves identifying intervals in the input/output sensor data over which the flow is "DRY" or "WET". After some experimentation it was found that the standard deviation of the raw sensor data, computed over a suitable averaging window provided a reliable way of identifying WET flows. A "WET" flow is defined as an interval over which the water depth measured by the sensor is indicative of a high volume flow, usually seen during "WET" weather events.

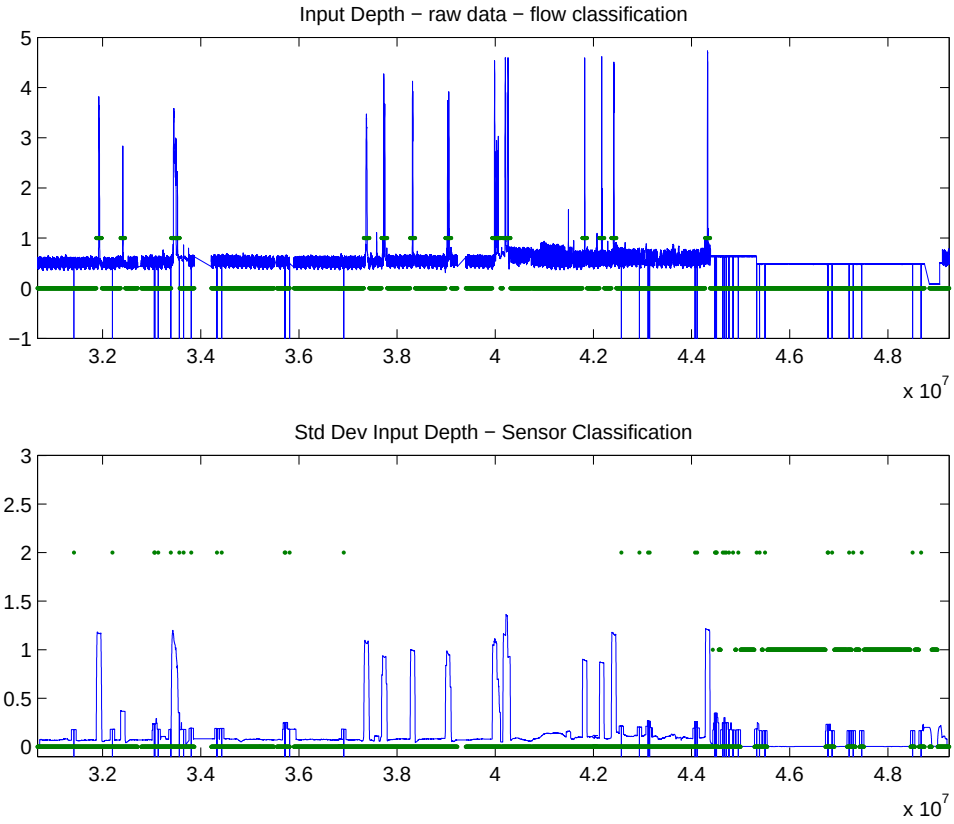


Fig. 2. Classified Input Sensor Stream

Figure 2 plots the data for the input sensor. The top plot shows the "raw" sensor data (blue) recovered from the EmNet database. The green points in the plot show the "classification" of the input flow. A value of 0 indicates a DRY flow and a value of 1 indicates a WET flow.

The bottom plot shows the standard deviation of the sensor data averaged over a window of suitable length. The green points in this plot show the classification of the sensor data. A value of 0 indicates that the sensor data is NORMAL. A value of 1 indicates that the sensor data is ANOMALOUS. Such anomalies occur when the variance of the data is too small to suggest that the sensor is functioning normally. A value of 2 indicates that the sensor data point is an OUTLIER and should not be used in computing means and variances. OUTLIER's are identified as points where the sensor value is below -1.

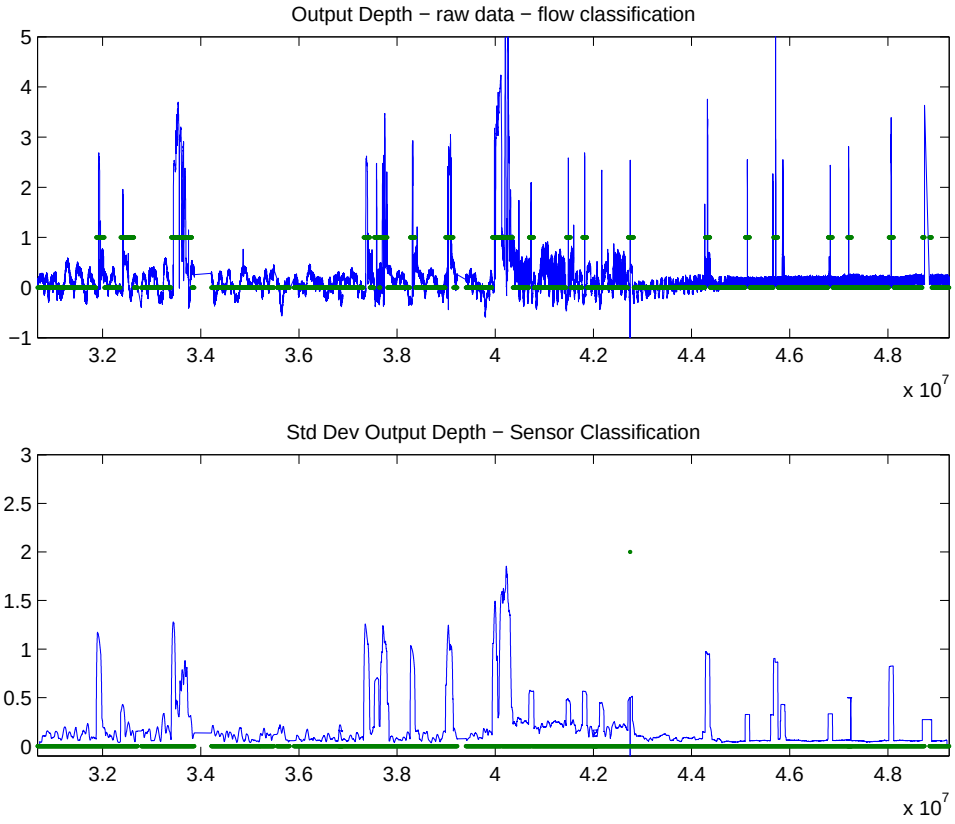


Fig. 3. Classified Output Sensor Streams

Figure 3 plots the data for the output sensor. These plots follow the same format as the plots in figure 2.

In reviewing the data shown in figures 2 and 3, it became apparent that the standard deviation (when computed over a suitable window length) provided a good way of identifying potential WET periods in the sensor data. In our case the window length was chosen to be 150 samples.

Window lengths less than 150 appeared to introduce a great deal of fluctuation in the computed averages. Window lengths much longer than 150 results in too much smoothing.

3.0 CSOID Algorithms

While visual inspection of the plots clearly shows when the wet flows occur, we must develop a computer algorithm that ”automates” this classification process. One way of automating this classification process is to simply choose a threshold, and declare the flow to be wet whenever the standard deviation of the flow is above this threshold. In general, however, this approach identifies ”wet points” in the flow as opposed to ”wet intervals”. In our case, we want to identify ”continuous intervals” (rather than points) over which the flow is ”wet”.

To address this issue, we employed a syntactical approach to interval recognition. We first pass the stream of standard deviations through an edge filter. This is, essentially, a filter that computes a derivative to the slope of the time series over a suitable window. In this case, a window of length 10 was used to compute the derivative. We then sequentially process the points in the trace. The current data point being examined will mark the beginning of a wet flow when the ”derivative” of the trace at that point is greater than a specified positive constant and the previously processed point is DRY. The next data point will also be declared WET as long as the standard deviation is above a specified threshold level. That threshold level is chosen based on the peak and minimum levels seen during the WET event and just before the WET event. In order to prevent excessive fragmentation of the interval, we also require a minimum interval length of at least 5 samples. The next data point is marked as DRY (thereby marking the end of the wet interval) if the standard deviation of the trace falls below the specified threshold and the prior point was marked as WET.

The results of this identification algorithm are shown in figures 2 and 3. These results clearly show that the proposed method accurately identifies most peaks in the sensor data. It appears that only one output sensor peak just after 4.2e7 seconds was not picked up by this algorithm.

Determining whether or not a CSO event is ”wet” or ”dry” involves simply matching the wet output flow events to suitable wet input flow events. For this example, we were able to classify each WET output flow as indicative of a ”wet” or ”dry CSO event provided we could find a wet input flow whose interval of support had a non-null intersection with the interval of support for the wet output flow. This type of logic was easy to automate in the CSOIDtool program and we

used it to automate the generation of output files capturing the sensor traces of each classified CSO event.

4.0 Results

In this example, we identified 20 wet output flow events and 13 wet input flow events, which resulted in potentially 7 dry CSO events being declared. The program automatically picked out the sensor data pertaining to the identified CSO events. The plots of this data are shown in the appendix. Each page of the appendix shows input/output flows for 4 CSO events.

In particular, we see that CSO events 0-1, 3, 5-8, 11, and 13 were identified as being WET events, whereas events 2, 4, 9-10, 12, and 14-19 were identified as being dry.

In reviewing the plots, we see that most WET CSO events have time histories as shown for CSO event 0. In this case there are well-defined peaks in both the input and output flows that overlap. Some of the WET CSO events seem to violate this trend. In particular, we see that CSO event 2, 7, and 8 have abnormal profiles. While these CSO events have wet input flows, we note that the some of the output flows are abnormally shaped. This occurs because the pump has stopped working properly. While this program does not automate the identification of such abnormal CSO events, it should be possible to modify the program so these events are also identified. This would be done using pattern recognition techniques similar to the syntactic methods used to identify the wet-flow periods.

In reviewing the plots, we see that most DRY CSO events have been correctly identified. Namely, there are ”wet” output flows that have no corresponding wet input flows. The only exception to this is seen in CSO event 4, where this is an abnormally shaped input flow pulse that was not detected by our algorithm. Another interesting observation occurs with regard to DRY events 12, 14-19. CSO event 12 shows an abnormal pulse shape on the output flow that may be due to problems with the sensor. CSO event 14-18 show an abnormally low level of variation in the input sensor. This may be indicative of a failure on the input sensor. The last CSO event 19 appears to be a result of a large outlier in the sensor data, again indicative of sensor failure.

5.0 Recommendations

We believe the preliminary results obtained with CSOIDtool are promising. These results

suggest that automating the identification of CSO events and sensor failure in the EmNET database can be reduced to a rather simple set of syntactical pattern recognition problems that are relatively easy to implement. While this report confines its attention to a relatively simple input-output set of flows, it should be possible to extend this approach to the two more complicated system shown below in figure 4.

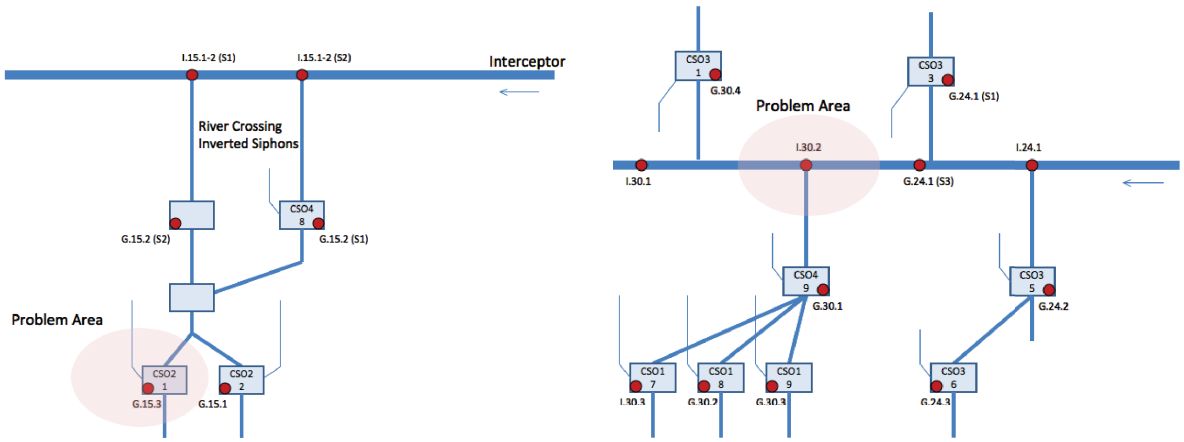


Fig. 4. More Examples

These systems can also be treated as a set of input-output flows. In the lefthand system, The input sensor flow is the sum of sensors `G.15.2(S2)` and `G.15.2(S1)` and the output is the sum of sensors `G.15.3` and `G.15.1`. In the righthand system we have a number of input-output relations that can be examined to check for possible CSO events. So the approach identified in this report should be easily extended to other scenarios. This extension requires prior identification of the appropriate input-output systems that the program is going to check. This can be done ahead of time using knowledge of the sewer system layout.

The bigger challenge involves improving the pattern recognition algorithms being used in this scheme. Right now we are triggering the detection of the wet-flow interval using the standard deviation of the sensor data. A somewhat better approach may be adopted using a matched filtering approach in which we use the impulse response of the system is used to compute the sufficient statistic required to detect the wet-flow periods. It would also be valuable to use the syntactical idea to detect abnormal wet-flow outputs.

Appendix: console_output.txt

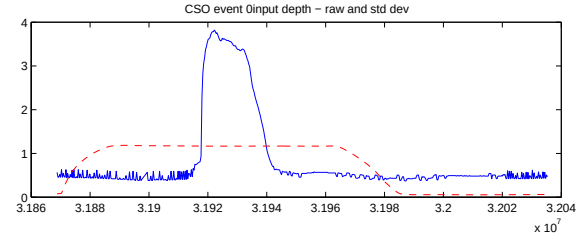
```
CSO OVERFLOW SUMMARY
        13 Input Events and 20 Output Events


CSO EVENT 0
        duration = 547 samples
        starting 2008-12-8 20:26:0
        ending   2008-12-10 18:46:0
        WET cso event
CSO EVENT 1
        duration = 915 samples
        starting 2008-12-14 14:11:0
        ending   2008-12-17 18:31:0
        WET cso event
CSO EVENT 2
        duration = 1404 samples
        starting 2008-12-26 13:6:0
        ending   2008-12-31 9:56:0
        DRY cso event
CSO EVENT 3
        duration = 421 samples
        starting 2009-2-6 21:56:0
        ending   2009-2-8 9:16:0
        WET cso event
CSO EVENT 4
        duration = 327 samples
        starting 2009-2-9 10:11:0
        ending   2009-2-10 13:46:0
        DRY cso event
CSO EVENT 5
```

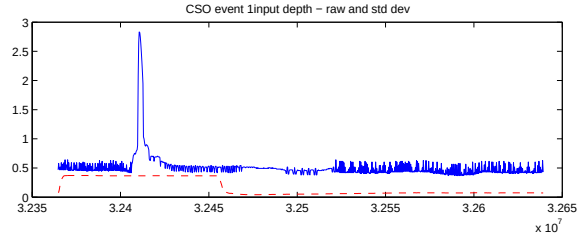
```
        duration = 479 samples
        starting 2009-2-10 20:46:0
        ending   2009-2-12 12:41:0
        WET cso event
CSO EVENT 6
        duration = 357 samples
        starting 2009-2-17 20:56:0
        ending   2009-2-19 2:41:0
        WET cso event
CSO EVENT 7
        duration = 538 samples
        starting 2009-2-26 5:6:0
        ending   2009-2-28 1:56:0
        WET cso event
CSO EVENT 8
        duration = 1371 samples
        starting 2009-3-7 5:31:0
        ending   2009-3-12 2:41:0
        WET cso event
CSO EVENT 9
        duration = 293 samples
        starting 2009-3-15 21:56:0
        ending   2009-3-17 1:41:0
        DRY cso event
CSO EVENT 10
        duration = 305 samples
        starting 2009-3-24 16:21:0
        ending   2009-3-25 17:46:0
        DRY cso event
CSO EVENT 11
        duration = 310 samples
```

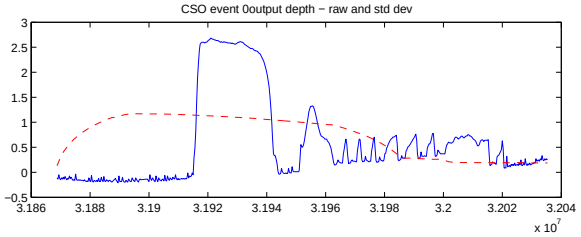
```
        starting 2009-3-28 12:26:0
        ending   2009-3-29 14:16:0
        WET cso event
CSO EVENT 12
        duration = 393 samples
        starting 2009-4-9 6:1:0
        ending   2009-4-10 15:11:0
        DRY cso event
CSO EVENT 13
        duration = 335 samples
        starting 2009-4-27 11:6:0
        ending   2009-4-28 15:1:0
        WET cso event
CSO EVENT 14
        duration = 306 samples
        starting 2009-5-6 21:31:0
        ending   2009-5-7 23:1:0
        DRY cso event
CSO EVENT 15
        duration = 317 samples
        starting 2009-5-13 12:56:0
        ending   2009-5-14 15:21:0
        DRY cso event
CSO EVENT 16
        duration = 305 samples
        starting 2009-5-26 10:56:0
        ending   2009-5-27 12:11:0
        DRY cso event
CSO EVENT 17
        duration = 313 samples
        starting 2009-5-31 18:16:0
```

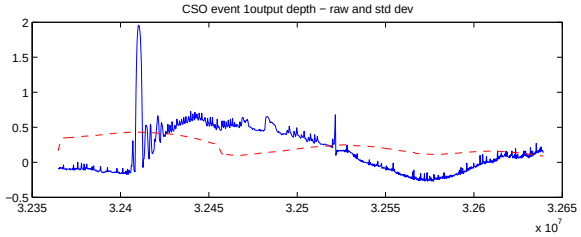
```
        ending   2009-6-1 20:31:0
        DRY cso event
CSO EVENT 18
        duration = 333 samples
        starting 2009-6-10 18:16:0
        ending   2009-6-11 22:6:0
        DRY cso event
CSO EVENT 19
        duration = 301 samples
        starting 2009-6-18 16:26:0
        ending   2009-6-20 21:1:0
        DRY cso event
TOTAL NUMBER OF CSO EVENTS = 20
```

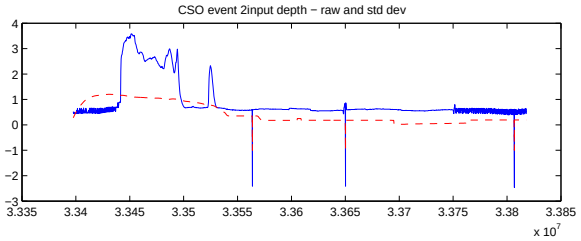
Appendix: CSO EVENT 0 to 3

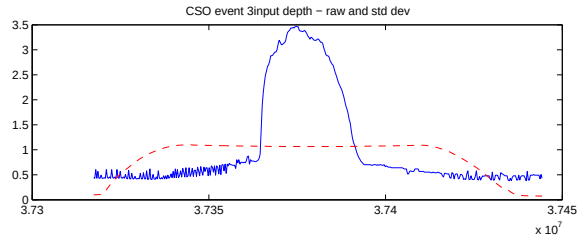


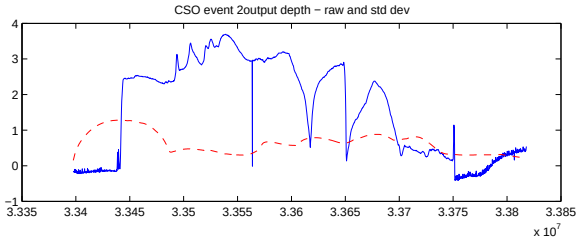


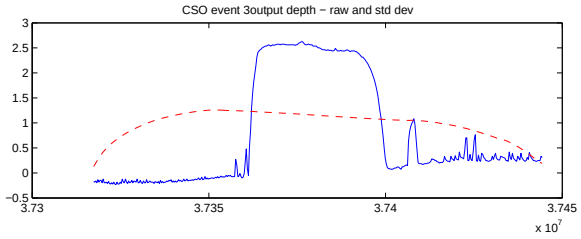




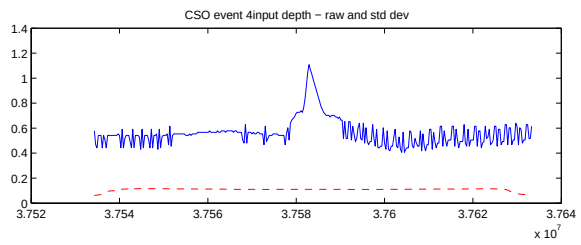


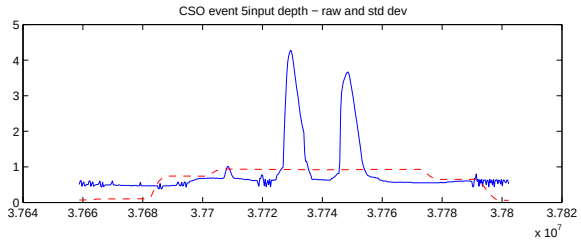


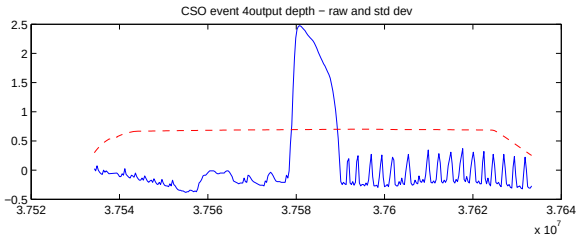


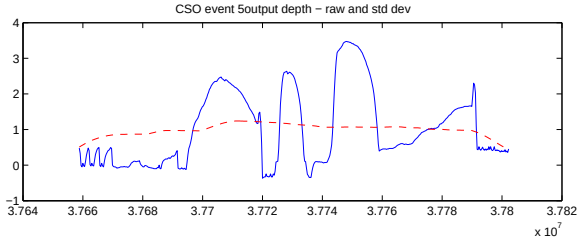


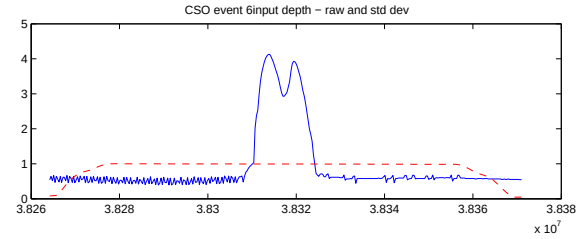
Appendix: CSO EVENT 4 to 7

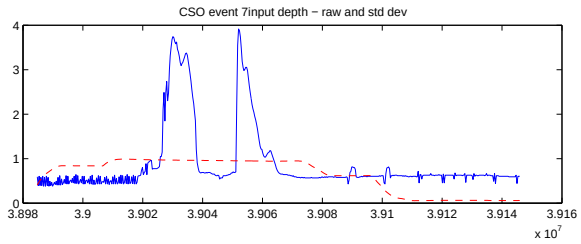


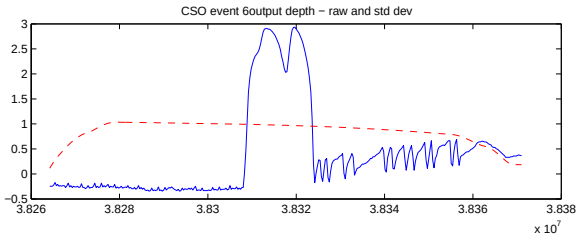


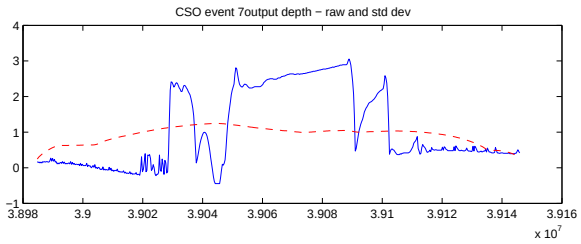




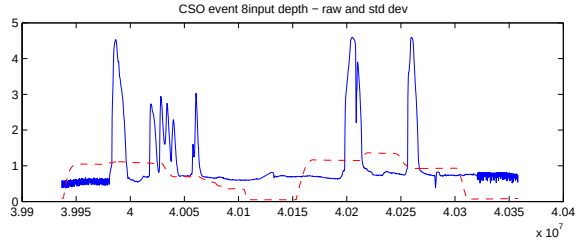


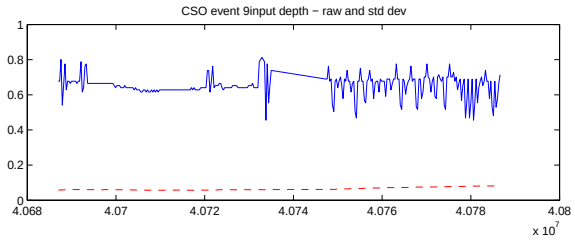


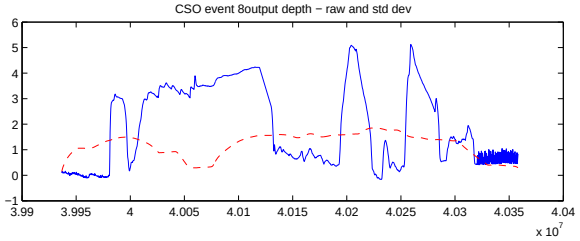


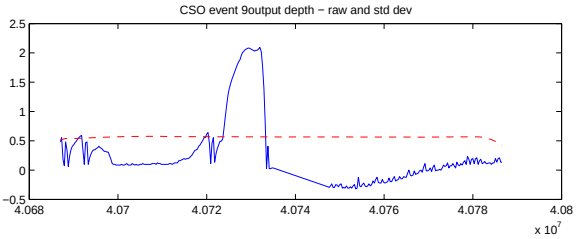


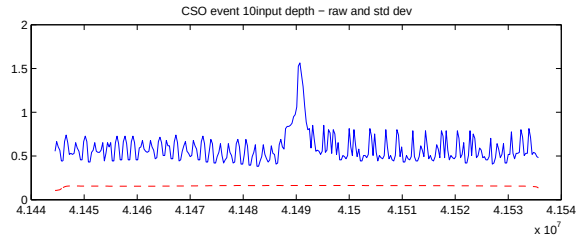
Appendix: CSO EVENT 8 to 11

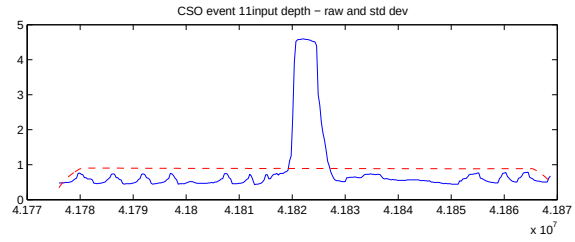


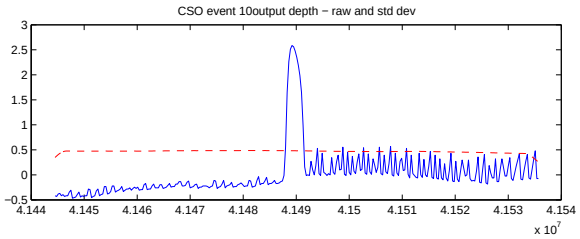


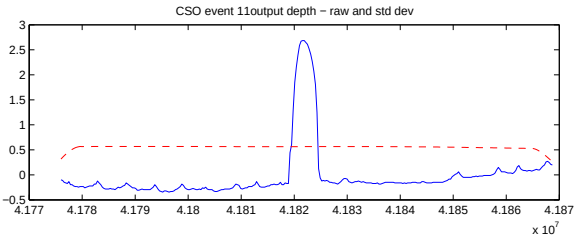




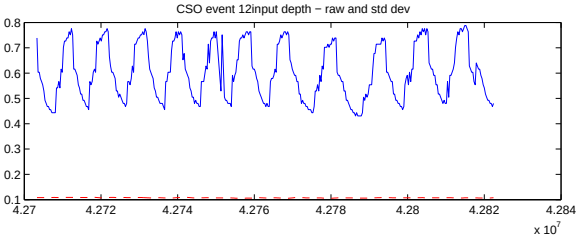


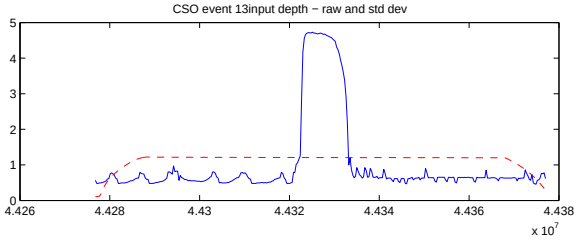


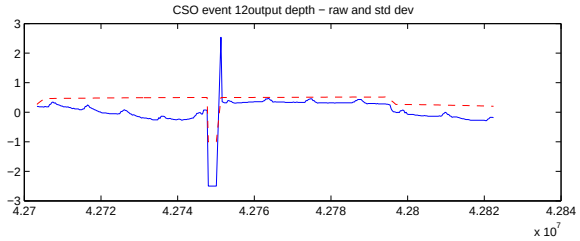


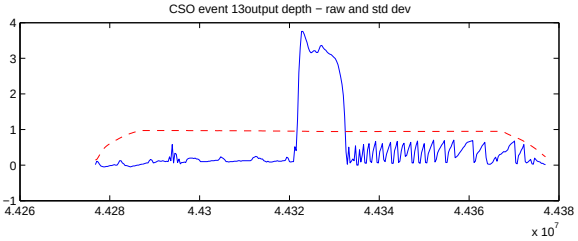


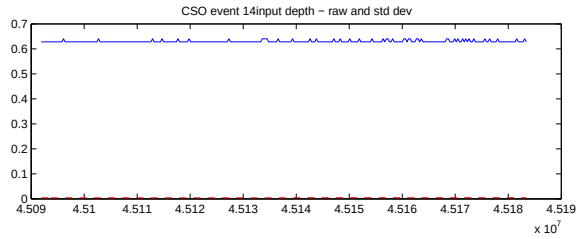
Appendix: CSO EVENT 12 to 15

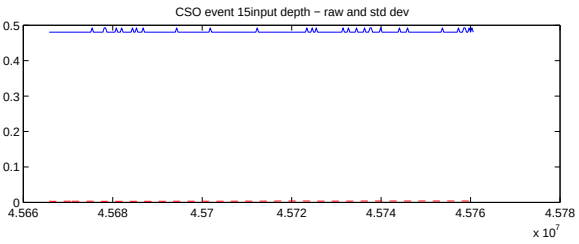


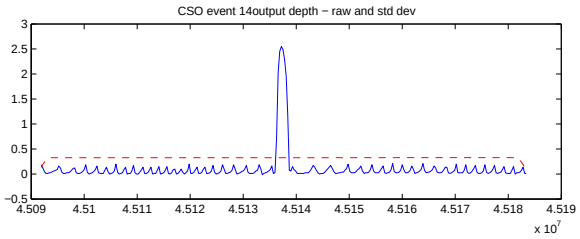


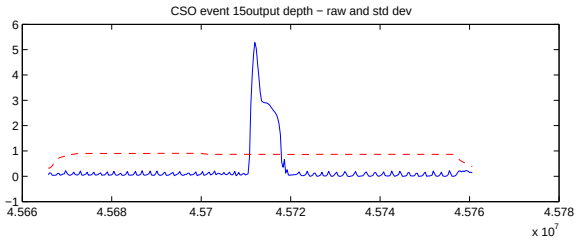












Appendix: CSO EVENT 16 to 19