



CANARY Training Tutorials



Acknowledgements

The National Homeland Security Research Center (NHSRC) would like to acknowledge the following organizations and individuals for their support in the development of the CANARY Training Tutorials:

U.S. EPA Office of Research and Development, NHSRC

Terra Haxton

Regan Murray

Jennifer Hagar (ORISE Fellow)

U.S. EPA Office of Water, Water Security Division

Steve Allgeier

Katie Umberg

Sandia National Laboratories (IA DW-89-92291401)

Samantha Cafferky

David Hart

Sean Hollister

Sean McKenna

Questions concerning this document or its application should be addressed to:

Terra Haxton

USEPA/NHSRC (NG 16)

26 W. Martin Luther King Drive

Cincinnati, OH 45268

(513) 569-7810

haxton.terra@epa.gov

Disclaimer

The U.S. Environmental Protection Agency (EPA) through its Office of Research and Development funded and collaborated in the research described here under Inter-Agency Agreement DW-89-92291401 with the Department of Energy's Sandia National Laboratories. This document has been subjected to the Agency's review and has been approved for publication. EPA does not endorse the purchase or sale of any commercial products or services.

This report was prepared as an account of work sponsored by an agency of the United States Government. Accordingly, the United States Government retains a nonexclusive, royalty free license to publish or reproduce the published form of this contribution, or allow others to do so for United States Government purposes.

Sandia Corporation, the United States Government, any agency thereof, and their employees do not make any warranty, express or implied, or assume any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately-owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by Sandia Corporation, the United States Government, or any agency thereof. The views and opinions expressed herein do not necessarily state or reflect those of Sandia Corporation, the United States Government or any agency thereof.

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Table of Contents

Acknowledgements	2
Table of Contents	4
List of Figures	5
List of Tables	8
List of Acronyms	9
1. Introduction	10
2. Configuration File Tutorial	13
2.1 Editing the Configuration File	13
2.2 Valid Range	16
2.3 Evaluation Type	19
2.4 History Window	25
2.5 Outlier Threshold	28
2.6 Event Threshold	30
2.7 Multiple Locations	31
3. Optimizing CANARY Configurations Tutorial	36
3.1 Multiple Algorithms	36
3.2 Consensus Algorithm	50
3.3 Binomial Event Discriminator	57
4. Database Driven Input/Output Tutorial	60
4.1 Obtaining JDBC Driver	60
4.2 Modifying Configuration File to Use Databases	62
5. Composite Signals Tutorials	69
5.1 Suppressing an Alarm After a Calibration Event	69
5.2 Integrating Pump Flow Operational Data into Event Detection	77
5.3 Integrating Tank Level Operational Information Into Event Detection	88
References	97
Appendix A: Frequently Asked Questions (FAQs)	98
Appendix B: Binomial Distribution Function Exercise	101
Appendix C: Configuration File Quick Reference	107
Appendix D: File Types	116
Glossary	117

List of Figures

Figure 1: YML file example viewed in UltraEdit.....	14
Figure 2: YML file example viewed in WordPad.	15
Figure 3: YML file example viewed in NotePad++.	16
Figure 4: <i>Signals</i> configuration section with default <i>valid range</i> parameters.	17
Figure 5: Signal plots produced using the default <i>valid range</i> parameters.....	18
Figure 6: <i>Signals</i> configuration section with the changed <i>valid range</i> parameters.	19
Figure 7: Signal plots produced using the <i>valid range</i> parameters in Figure 6.	19
Figure 8: <i>Signals</i> configuration section with the <i>evaluation type</i> parameter as OP.	20
Figure 9: Operational signal plot.	20
Figure 10: <i>Signals</i> configuration section with the <i>evaluation type</i> parameter as WQ.....	21
Figure 11: Water quality signal plot.	21
Figure 12: <i>Signals</i> configuration section with the <i>evaluation type</i> parameter as CAL.	21
Figure 13: Output graph without calibration signal.	22
Figure 14: <i>Signals</i> configuration section with the <i>evaluation type</i> parameter switched to OP.	23
Figure 15: Output graph with the <i>evaluation type</i> parameter switched to OP.....	24
Figure 16: <i>Algorithms</i> configuration section with a <i>history window</i> parameter of 144.....	25
Figure 17: Output graph produced when the <i>history window</i> parameter is 144.	26
Figure 18: <i>Algorithms</i> configuration section with a <i>history window</i> parameter of 36.....	27
Figure 19: Output graph produced when the <i>history window</i> parameter is 36.	28
Figure 20: <i>Algorithms</i> configuration section with an <i>outlier threshold</i> parameter of 0.8.	29
Figure 21: Event probability plot produced when the <i>outlier threshold</i> parameter is 0.8.	29
Figure 22: <i>Algorithms</i> configuration section with an <i>outlier threshold</i> parameter of 0.99.	30
Figure 23: Event probability plot produced when the <i>outlier threshold</i> parameter is 0.99.	30
Figure 24: <i>Algorithms</i> configuration section with an <i>event threshold</i> parameter of 0.85.....	30
Figure 25: Event probability plot produced when the <i>event threshold</i> parameter is 0.85.	31
Figure 26: <i>Algorithms</i> configuration section with an <i>event threshold</i> parameter of 0.99.....	31
Figure 27: Event probability plot produced when the <i>event threshold</i> parameter is 0.99.	31
Figure 28: <i>Monitoring stations</i> configuration section.	33
Figure 29: Output EDSF files.....	33
Figure 30: Station A output graph.	34
Figure 31: Station B output graph.....	35
Figure 32: <i>Algorithms</i> configuration section with a <i>history window</i> parameter of 36.....	38
Figure 33: <i>Monitoring stations</i> section with multiple algorithms enabled.	39
Figure 34: Output graph when the <i>history window</i> parameter is 36.	40
Figure 35: <i>Algorithms</i> configuration section with a <i>history window</i> parameter of 72.....	41
Figure 36: Output graph when the <i>history window</i> parameter is 72.	42
Figure 37: <i>Algorithms</i> configuration section with a <i>history window</i> parameter of 108.....	44
Figure 38: Output graph when the <i>history window</i> parameter is 108.	45
Figure 39: <i>Algorithms</i> configuration section with a <i>history window</i> parameter of 144.....	46
Figure 40: Output graph when the <i>history window</i> parameter is 144.	47
Figure 41: <i>Algorithms</i> configuration section with a <i>history window</i> parameter of 180.....	48
Figure 42: Output graph when the <i>history window</i> parameter is 180.	49
Figure 43: CANARY total events graph.....	50
Figure 44: <i>Signals</i> configuration section.	52

Figure 45: <i>Algorithms</i> configuration section with the consensus algorithm defined.....	52
Figure 46: Monitoring configuration section with algorithm B2 activated.	53
Figure 47: Output graph produced using the set point algorithm, SPPE.	54
Figure 48: <i>Algorithms</i> configuration section with three algorithms defined.	55
Figure 49: <i>Monitoring stations</i> configuration section with the consensus algorithm enabled.	55
Figure 50: Output graph produced when using the consensus algorithm.	56
Figure 51: <i>Algorithms</i> configuration section with a BED <i>window</i> parameter of 20.	57
Figure 52: <i>Algorithms</i> configuration section with a BED <i>window</i> parameter of 6.....	58
Figure 53: Output graph produced using a BED <i>window</i> parameter of 20.....	58
Figure 54: Output graph produced using a BED <i>window</i> parameter of 6.....	59
Figure 55: JDBC driver zip file location.....	60
Figure 56: Unzipping JDBC driver file.	61
Figure 57: Unzipped file location.	61
Figure 58: Copying JAR files.	62
Figure 59: New JAR file location.	62
Figure 60: <i>Data sources</i> configuration section with the <i>type</i> parameter set as database.....	63
Figure 61: <i>Data sources</i> configuration section with the <i>location</i> parameter listing URL of database.....	64
Figure 62: <i>Data sources</i> configuration section with the <i>timestep options</i> parameters.	65
Figure 63: Data sources configuration section with the <i>database options</i> parameters.....	66
Figure 64: <i>Data sources</i> configuration section with <i>database options login</i> parameters.....	67
Figure 65: <i>Data sources</i> configuration section with the optional <i>database options</i> parameter, <i>input format</i>	68
Figure 66: Output graph produced for January 3, 2011.....	70
Figure 67: Output plot produced for January 6, 2011.....	71
Figure 68: <i>Signals</i> configuration section for original calibration signal, RAW_CAL.	71
Figure 69: <i>Signals</i> configuration section for composite calibration signal.....	72
Figure 70: <i>Algorithms</i> and <i>monitoring stations</i> configuration sections.....	73
Figure 71: Output graph produced with composite signal.....	74
Figure 72: <i>Signals</i> configuration section for CAL_TIME_OUT_CTFD_A.....	75
Figure 73: <i>Signals</i> configuration section for FINAL_CAL_CTFD.....	76
Figure 74: Output graph produced with alarm suppression composite signal enabled.	77
Figure 75: <i>Monitoring stations</i> configuration section.	78
Figure 76: Output graph produced of Station D data with a calibration period and three alarms.	79
Figure 77: <i>Monitoring stations</i> configuration section with three pump signals for Station D.	80
Figure 78: Output graph produced with the additional three operational signals.....	81
Figure 79: <i>Signals</i> configuration section with added composite signals.	82
Figure 80: Output graph produced with three operation signals and event probability.....	83
Figure 81: <i>Signals</i> configuration section with changed calibration signal.	83
Figure 82: <i>Signals</i> configuration section with the composite calibration signal.	84
Figure 83: Output plots produced using composite calibration signal.	85
Figure 84: <i>Signals</i> configuration section with the modified composite signal.....	86
Figure 85: Output plots produced using modified composite signal.	86
Figure 86: Output plots produced using modified composite signal for January 25th only.....	87
Figure 87: <i>Signals</i> configuration section with final composite signal.....	87
Figure 88: Output plots produced using final composite signal for entire week.	88
Figure 89: Output plots produced using final composite signal for January 25 th	88

Figure 90: <i>Monitoring stations</i> configuration section.	89
Figure 91: Output graph produced using initial configuration file.	90
Figure 92: <i>Signals</i> configuration section defining composite signal, REL_TANK_LVL.....	91
Figure 93: <i>Signals</i> configuration section defining composite signal, TANK_CL_CHANGE.	92
Figure 94: <i>Monitoring stations</i> configuration section composite signal enabled.	93
Figure 95: Output graph produced when using composite signal.....	94
Figure 96: <i>Signals</i> configuration section with the modified TANK_CL_CHANGE signal.	95
Figure 97: Output graph produced when using modified signal.....	96
Figure 98: Example of comments within a configuration file.	98
Figure 99: NFAILURES column using a NTRIALS of 20.	102
Figure 100: Binomial distribution function using a NTRIALS of 20 and a PFAIL of 0.5.....	103
Figure 101: Probability of event values for a NTRIALS of 20.	104
Figure 102: Probability of event graph using a NTRIALS of 20.	105
Figure 103: Probability of event values for a NTRIALS of 6.	105
Figure 104: Probability of event graph using a NTRIALS of 6.	106
Figure 105: Example of <i>canary</i> section using BATCH mode.	108
Figure 106: Example of <i>canary</i> section using a database connection.	108
Figure 107: Example of <i>timing options</i> section.	109
Figure 108: Example of <i>data sources</i> section using a CSV file.	111
Figure 109: Example of <i>signals</i> section using a chlorine signal.....	113
Figure 110: Example of <i>algorithms</i> section using the LPCF algorithm.	114
Figure 111: Example of <i>monitoring stations</i> section.	115

List of Tables

Table 1: Key Parts of <i>CANARY Training Tutorials</i>	10
Table 2: Folder Directories for <i>CANARY Training Tutorials</i>	11
Table 3: CANARY Configuration File Sections	107
Table 4: Input Parameters for <i>canary</i> Section of the CANARY Configuration File.....	107
Table 5: Input Parameters for <i>timing options</i> Section of the CANARY Configuration File.....	108
Table 6: Input Parameters for <i>data sources</i> Section of the CANARY Configuration File	109
Table 7: Input Parameters for <i>signals</i> Section of the CANARY Configuration File	112
Table 8: Input Parameters for <i>algorithms</i> Section of the CANARY Configuration File	113
Table 9: Input Parameters for <i>monitoring stations</i> Section of the CANARY Configuration File	114

List of Acronyms

ALM	Alarm signal
CAL	Calibration signal
BED	Binomial Event Discriminator
CANARY	Name of the CANARY software, not an acronym!
CAVE	Consensus AVErage, one of the consensus algorithms contained in CANARY
CL2	Residual or free chlorine concentration in water (water quality parameter)
CMAX	Consensus MAXimum, one of the consensus algorithms contained in CANARY
COND	Specific conductivity of water (water quality parameter)
CSV	Comma-Separated Value data file
EDS	Event Detection System
EDSC	File extension for CANARY pattern matching cluster file (e.g., cluster_file.edsc)
EDSD	File extension for CANARY output file (e.g., output_file.edsd)
EDSY	File extension for CANARY configuration file (e.g., config_file.edsy)
EPA	Environmental Protection Agency
FAQ	Frequently Asked Questions
LPCF	Linear Prediction Coefficient Filter, one of the algorithms contained in CANARY
MVNN	Multivariate Nearest Neighbor, one of the algorithms contained in CANARY
ORP	Oxidation reduction potential
PH	pH is a measure of the acidity of water (water quality parameter)
SCADA	Supervisory Control and Data Acquisition
SPPE	Set point proximity exponential distribution algorithm contained in CANARY
TEMP	Temperature of water (water quality parameter)
TOC	Total organic carbon concentration in water (water quality parameter)
TURB	Turbidity in water (water quality parameter)
YML	File extension for CANARY configuration file (e.g., config_file.yml)

1. Introduction

The CANARY event detection software was developed to enhance the detection of contaminants in drinking water. Many drinking water utilities collect real-time data from sensors located throughout the water distribution network. The sensors measure water quality parameters such as pH, residual chlorine, total organic carbon, and specific conductance. CANARY analyzes these data rapidly to identify anomalous or abnormal periods of water quality that might indicate contamination incidents. CANARY has operated in several water utilities around the world since it was released publicly in 2009. The software has also been used as a research platform for testing and developing new capabilities for event detection system (EDS) algorithms and sensor performance. *CANARY Training Tutorials* provide practical examples of the full range of CANARY's capabilities. Some basic functions are used each time the software is used, and others are used mainly in specific situations. Each example is documented with a step-by-step approach using text and screen-capture figures illustrating the changes made to the configuration file parameters and discussing the impact of those changes on the CANARY output. Table 1 lists all of the key sections of the *CANARY Training Tutorials*.

Table 1: Key Parts of *CANARY Training Tutorials*

Section 2	Specifying parameters for the CANARY configuration file
Section 3	Optimizing CANARY parameter values for maximum performance at multi-sensor monitoring stations
Section 4	Running CANARY in real-time with input from a database
Section 5	Creating composite signals in CANARY based on combinations of sensor data
Appendix A	Frequently Asked Questions (FAQs)
Appendix B	Binomial Distribution Function Exercise
Appendix C	Configuration File Quick Reference
Appendix D	File Types
Glossary	Definitions of commonly used terms in CANARY

All of the input and output files used in *CANARY Training Tutorials* are available to download from the same location as this document and the CANARY Trac site (<https://software.sandia.gov/trac/canary/downloader/download/category/6>). The *CANARY_Training_Tutorials.zip* file contains two folders, *Tutorial_Files* and *Tutorial_Results*. The *Tutorial_Files* folder contains the input files for the examples covered in the *CANARY Training Tutorials* and can be used to replicate the examples. The *Tutorial_Results* folder contains the input and output for these examples and can be used to verify the example results. This folder is Read-only. The main directories of these folders are listed in Table 2.

Table 2: Folder Directories for *CANARY Training Tutorials*

Main Directory in Tutorial_Files	Subdirectories in Tutorial_Files	Corresponding Training Tutorials Section
Configuration_Tutorial	Valid_Range\Default Valid_Range\Change	2.2 Valid Range
	Evaluation_Type\Temp_wq Evaluation_Type\Temp_op Evaluation_Type\Cal_cal Evaluation_Type\Cal_op	2.3 Evaluation Type
	History_Window\Initial History_Window\Change	2.4 History Window
	Outlier_Threshold\Initial Outlier_Threshold\Change	2.5 Outlier Threshold
	Event_Threshold\Initial Event_Threshold\Change	2.6 Event Threshold
	Multiple_Locations	2.7 Multiple Locations
Optimizing_Tutorials	Multiple_Algorithms\HW_36 Multiple_Algorithms\HW_72 Multiple_Algorithms\HW_108 Multiple_Algorithms\HW_144 Multiple_Algorithms\HW_108	3.1 Multiple Algorithms
	Consensus_Algorithm\Initial Consensus_Algorithm\Join	3.2 Consensus Algorithm
	BED\Window_6 BED\Window_20	3.3 Binomial Event Discriminator
Database_Tutorial		4.1 Obtaining JDBC Driver, 4.2 Modifying Configuration File to Use Databases
Composite_Tutorials	Composite_Signals_1\Initial Composite_Signals_1\Flip_Cal Composite_Signals_1\Suppress	5.1 Suppressing an Alarm After a Calibration Event
	Composite_Signals_2\Initial Composite_Signals_2\Step 1 Composite_Signals_2\Step 2 Composite_Signals_2\Step 3 Composite_Signals_2\Step 4 Composite_Signals_2\Step 5	5.2 Integrating Pump Flow Operational Data into Event Detection
	Composite_Signals_3\Initial Composite_Signals_3\Step 1 Composite_Signals_3\Step 2	5.3 Integrating Tank Level Operational Information Into Event Detection

The user is encouraged to examine other existing CANARY software documentation:

- *CANARY Quick Start Guide* describes how to install and run CANARY (U.S. EPA 2012).
- *CANARY User's Manual* provides a concise description of the configuration file and the parameters used in the event detection algorithms (Hart and McKenna 2012).
- *Water Quality Event Detection Systems for Drinking Water Contamination Warning Systems* (Murray et al. 2010) defines the motivation for the development of CANARY and the theory underlying the software's mathematical and statistical algorithms.

Presentations from CANARY webinars are available in Adobe PDF format at this site: <https://software.sandia.gov/trac/canary/downloader/download/category/6>. Users will be required to fill out a short registration form.

2. Configuration File Tutorial

The configuration file is one of the required CANARY input files. It specifies all the parameter values needed to run CANARY for a particular analysis. In addition, the software requires a data source: either a comma-separated value (CSV) file that contains the sensor data, or a link to a database containing the data (for many water utilities, part of a Supervisory Control and Data Acquisition (SCADA) system).

The parameters specified in the configuration file communicate specific details about the type of data source to be used, the length of the analysis and other timing options, the water quality, operational and alarm/calibration signals contained in the data sources, and the algorithms and associated parameter values used to analyze the data.

This section provides a tutorial designed to increase familiarity with the CANARY configuration file and the parameters contained within it. In particular, these tutorials examine several of the specific parameters that need to be defined for a CANARY run in the *signals*, *algorithms*, or *monitoring stations* sections of the configuration file.

- In the *signals* section of the configuration file, the tutorials examine the valid range of the water quality (WQ), operational (OP), and calibration (CAL) signals and the type of an input signal.
- For the *algorithms* section of the configuration file, the tutorial examines the size of the *history window*, the *outlier threshold*, and the *event threshold*. The last tutorial in this section explains how to add multiple monitoring stations to a single configuration file under the *monitoring stations* section.

The associated files are found in the “Tutorial_Files\Configuration_Tutorial\” directory.

Additional information on the configuration file and these parameters is provided in Appendix D: Configuration File Quick Reference of this document and Section 5 of the CANARY User’s Manual (Hart and McKenna 2012).

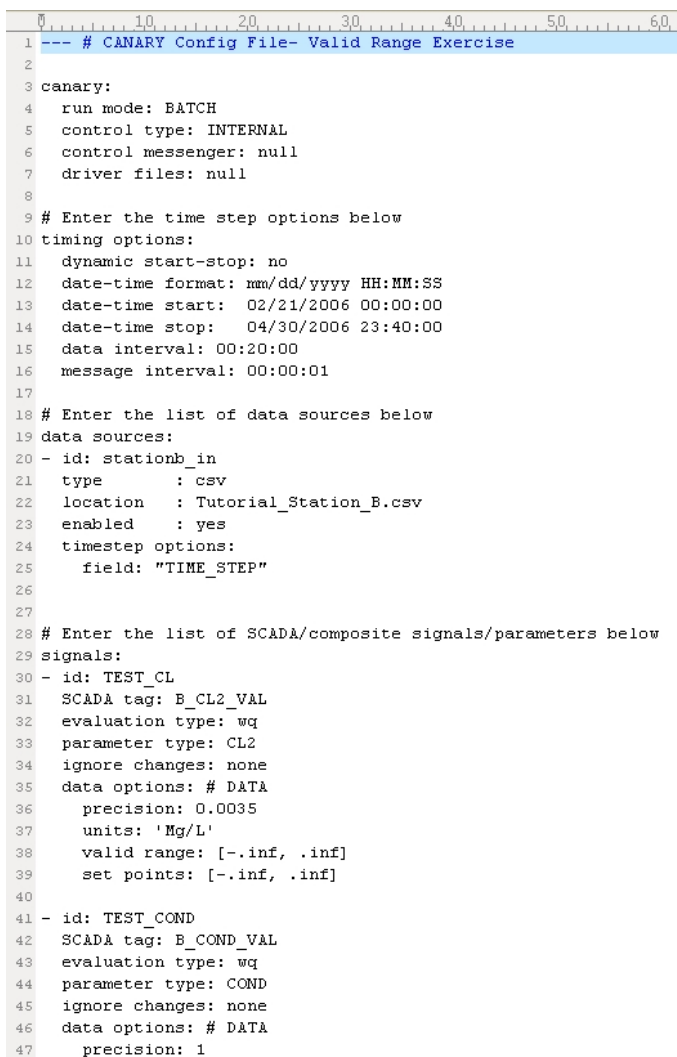
2.1 Editing the Configuration File

Configuration files are written in the YML markup language. YML files are commonly used for software configuration because their format allows for a large amount of detailed information to be read and edited. YML files are formatted with a single parameter value defined on each line, specified by the parameter name and the parameter value. Groups of parameters that go together to jointly define functionality within CANARY are indicated by indented lines. Descriptive comments are added following a number sign (#) and are ignored by CANARY. The configuration files end with either an “.yml” (YML) or “.edsy” (EDSY) extension.¹ The EDSY extension hooks the file into the right-click abilities of the Windows ® operating system; otherwise it is equivalent to the YML extension.

¹ This is a change from previous CANARY configuration files that were written in the XML markup language and ended with an “.edsx” (EDSX) extension. However, CANARY is backward compatible and can read EDSX files and translate into EDSY files.

Any text editor can be used to view and edit the CANARY configuration file. An example configuration file is shown using three different editors: UltraEdit, WordPad, and Notepad++ in Figure 1, Figure 2, and Figure 3, respectively. The formatting of the configuration files, including the indentations, is easy to see in all three editors. The Notepad++ editor recognizes the YML format and automatically uses different colors to highlight different components of the configuration file, for example, comments are shown in green.

The file shown in each figure is stationB_VR_CL.yml and is found in the directory “Tutorial_Files\Configuration_Tutorial\Valid_Range.” As shown, the file specifies that this CANARY run is in batch mode using historical data contained in a CSV file. The signals to be analyzed by CANARY include residual chlorine (CL2) and conductivity (COND) data collected every 20 minutes from 02/21/2006 - 04/30/2006.



```

1 --- # CANARY Config File- Valid Range Exercise
2
3 canary:
4   run mode: BATCH
5   control type: INTERNAL
6   control messenger: null
7   driver files: null
8
9 # Enter the time step options below
10 timing options:
11   dynamic start-stop: no
12   date-time format: mm/dd/yyyy HH:MM:SS
13   date-time start: 02/21/2006 00:00:00
14   date-time stop: 04/30/2006 23:40:00
15   data interval: 00:20:00
16   message interval: 00:00:01
17
18 # Enter the list of data sources below
19 data sources:
20 - id: stationB_in
21   type : csv
22   location : Tutorial_Station_B.csv
23   enabled : yes
24   timestep options:
25     field: "TIME_STEP"
26
27
28 # Enter the list of SCADA/composite signals/parameters below
29 signals:
30 - id: TEST_CL
31   SCADA tag: B_CL2_VAL
32   evaluation type: wq
33   parameter type: CL2
34   ignore changes: none
35   data options: # DATA
36     precision: 0.0035
37     units: 'Mg/L'
38     valid range: [-.inf, .inf]
39     set points: [-.inf, .inf]
40
41 - id: TEST_COND
42   SCADA tag: B_COND_VAL
43   evaluation type: wq
44   parameter type: COND
45   ignore changes: none
46   data options: # DATA
47     precision: 1

```

Figure 1: YML file example viewed in UltraEdit.

```

--- # CANARY Config File- Valid Range Exercise

canary:
  run mode: BATCH
  control type: INTERNAL
  control messenger: null
  driver files: null

# Enter the time step options below
timing options:
  dynamic start-stop: no
  date-time format: mm/dd/yyyy HH:MM:SS
  date-time start: 02/21/2006 00:00:00
  date-time stop: 04/30/2006 23:40:00
  data interval: 00:20:00
  message interval: 00:00:01

# Enter the list of data sources below
data sources:
- id: stationb_in
  type : csv
  location : Tutorial_Station_B.csv
  enabled : yes
  timestep options:
    field: "TIME_STEP"

# Enter the list of SCADA/composite signals/parameters below
signals:
- id: TEST_CL
  SCADA tag: B_CL2_VAL
  evaluation type: wq
  parameter type: CL2
  ignore changes: none
  data options: # DATA
    precision: 0.0035
    units: 'Mg/L'
    valid range: [-.inf, .inf]
    set points: [-.inf, .inf]

- id: TEST_COND
  SCADA tag: B_COND_VAL
  evaluation type: wq
  parameter type: COND
  ignore changes: none
  data options: # DATA
    precision: 1
    units: '(\mu)S/cm'
    valid range: [-.inf, .inf]
    set points: [-.inf, .inf]

```

Figure 2: YML file example viewed in WordPad.

```

1  --- # CANARY Config File- Valid Range Exercise
2
3  canary:
4      run mode: BATCH
5      control type: INTERNAL
6      control messenger: null
7      driver files: null
8
9  # Enter the time step options below
10 timing options:
11     dynamic start-stop: off
12     date-time format: mm/dd/yyyy HH:MM:SS
13     date-time start: 02/21/2006 00:00:00
14     date-time stop: 04/30/2006 23:40:00
15     data interval: 00:20:00
16     message interval: 00:00:01
17
18 # Enter the list of data sources below
19 data sources:
20     id: stationb_in
21     type: csv
22     location: Tutorial_Station_B.csv
23     enabled: yes
24     timestep options:
25         field: "TIME_STEP"
26
27
28 # Enter the list of SCADA/composite signals/parameters below
29 signals:
30     id: TEST_CL
31     SCADA tag: B_CL2_VAL
32     evaluation type: wq
33     parameter type: CL2
34     ignore changes: none
35     data options:
36         precision: 0.0035
37         units: 'Mg/L'
38         valid range: [-.inf, .inf]
39         set points: [-.inf, .inf]

```

Figure 3: YML file example viewed in NotePad++.

2.2 Valid Range

This tutorial examines the *valid range* parameter which defines a range of acceptable values for each input data signal in the configuration file. This parameter also provides the minimum and maximum bounds on the y-axis of CANARY's graphical output. However, if the data is within a smaller range, the graphical output adjusts to this smaller range on the y-axis (i.e., the graph zooms in on the data) in order to examine the smaller fluctuations in the data. Values outside the *valid range* are ignored by CANARY. This parameter can be used to indicate when data should be ignored as a result of sensor or data transmission malfunctions. Thus, the *valid range* parameter value should be selected with care.

If the *valid range* parameter is omitted or left blank, then the values are set to the default, [-.inf, .inf], where .inf stands for infinity. This default range does not restrict signal values and so all values will be read and analyzed by CANARY. However, the default can allow for unrealistic values to be included in the analysis. For example, the pH concentration is always above or equal to zero and less than 14, then, the *valid range* might be defined as [0, 14].

Figure 4 shows part of the stationB_VR_CL.yml configuration file found in the “Tutorial_Files\Configuration_Tutorial\Valid_Range\Default” directory. For the CL2, COND and PH signals defined, the *valid range* parameter is set to the default value of `[-.inf, .inf]`. When this configuration file is run in CANARY, the y-axis of the graphical output for each signal is automatically scaled to the minimum and maximum value of that signal within the time period of the plot, as shown in Figure 5 (the blue dots/lines represent the water quality data that contributed to the identification of an event at that time). In this case, chlorine varies between 0 and 3 mg/L, while pH varies from 7.2 to 8. This automatic scaling might work for many signals, but sometimes the scale of the plot might be too broad to provide details of the signal fluctuations. The *valid range* parameter can be used to adjust the scale of the plots. However, reduction in the range might cause some data to be ignored in the CANARY analysis.

```
signals:
- id: TEST_CL
  SCADA tag: B_CL2_VAL
  evaluation type: wq
  parameter type: CL2
  ignore changes: none
  data options: # DATA
    precision: 0.0035
    units: 'Mg/L'
    valid range: [-.inf, .inf]
    set points: [-.inf, .inf]

- id: TEST_COND
  SCADA tag: B_COND_VAL
  evaluation type: wq
  parameter type: COND
  ignore changes: none
  data options: # DATA
    precision: 1
    units: '{\mu}S/cm'
    valid range: [-.inf, .inf]
    set points: [-.inf, .inf]

- id: TEST_PH
  SCADA tag: B_PH_VAL
  evaluation type: wq
  parameter type: PH
  ignore changes: none
  data options: # DATA
    precision: 0.01
    units: 'pH'
    valid range: [-.inf, .inf]
    set points: [-.inf, .inf]
```

Figure 4: *Signals* configuration section with default *valid range* parameters.

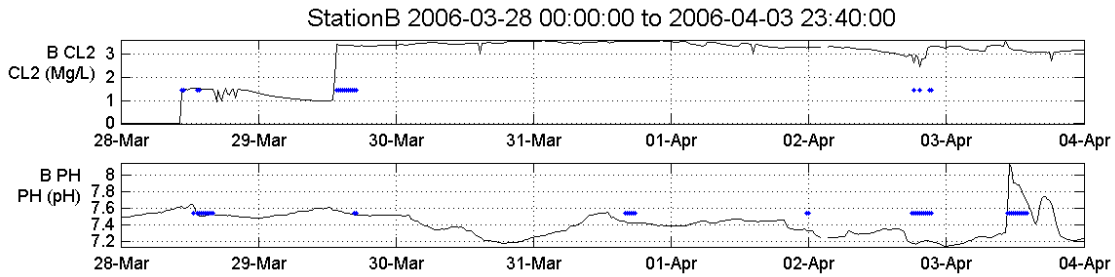


Figure 5: Signal plots produced using the default *valid range* parameters.

Figure 6 shows part of the *signals* section of the stationB_VR_CL_.yml configuration file found in the “Tutorial_Files\Configuration_Tutorial\Valid_Range\Change” directory. The *valid range* parameter for the CL2 signal was changed from $[-\text{inf}, \text{inf}]$ to $[0.5, \text{inf}]$, and from $[-\text{inf}, \text{inf}]$ to $[7.0, 7.5]$ for the PH signal. After running the changed YML file in CANARY, the results are shown in Figure 7. When the data exceeds the *valid range*, the plot is marked with a pink triangle on the upper or lower boundary according to which value was exceeded. Multiple pink triangles are shown in Figure 7; in order to capture all of the data in the plot, the *valid range* parameter needs to be broadened. It is useful to recall that the *valid range* parameter value needs to be selected with care, because it determines which data is analyzed by CANARY and which data is included in the graphical output.

Another feature of the CANARY graphing output is shown in Figure 5 and Figure 7; when all signals drop to a value of zero, the station is considered offline, the data is not analyzed, and these values are not shown in the plot. In Figure 5 and Figure 7, the one hour of missing data early on April 2nd is evident by the small gaps in the plots.

```

signals:
|- id: TEST_CL
  SCADA tag: B_CL2_VAL
  evaluation type: wq
  parameter type: CL2
  ignore changes: none
| data options: # DATA
  precision: 0.0035
  units: 'Mg/L'
  valid range: [0.5, .inf]
  set points: [-.inf, .inf]

|- id: TEST_COND
  SCADA tag: B_COND_VAL
  evaluation type: wq
  parameter type: COND
  ignore changes: none
| data options: # DATA
  precision: 1
  units: '(\mu)S/cm'
  valid range: [-.inf, .inf]
  set points: [-.inf, .inf]

|- id: TEST_PH
  SCADA tag: B_PH_VAL
  evaluation type: wq
  parameter type: PH
  ignore changes: none
| data options: # DATA
  precision: 0.01
  units: 'pH'
  valid range: [7.0, 7.5]
  set points: [-.inf, .inf]

```

Figure 6: Signals configuration section with the changed *valid range* parameters.

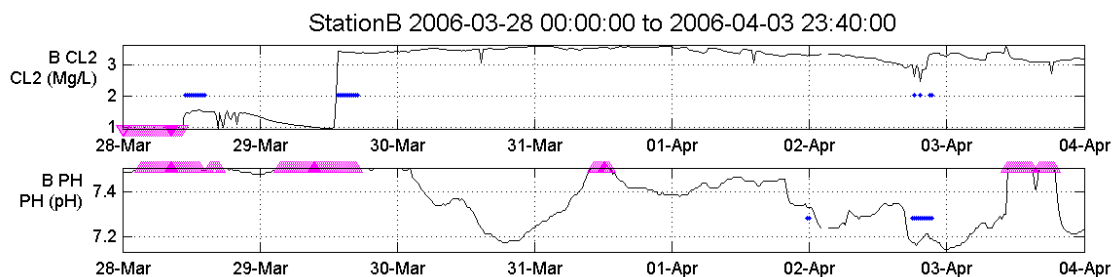


Figure 7: Signal plots produced using the *valid range* parameters in Figure 6.

2.3 Evaluation Type

This tutorial examines the *evaluation type* parameter which classifies input signals into one of four different types: water quality (WQ), operational (OP), alarm (ALM) or calibration (CAL) signals. Any signal must be classified into one of these types, but typically, water quality signals (e.g., pH, residual chlorine, total organic carbon, specific conductance) are those that directly

measure an aspect of the quality of the water, although pressure, valve status, or other measurable quantity could be labeled as a WQ type signal and CANARY could use the signal to see if an event has occurred. Operational signals typically include water temperature, water levels in tanks, pressures, flow rates, valve settings, or pump operations. Alarm signals generally indicate a performance or maintenance issue (e.g., the sensor is out of reagents or a membrane needs cleaning). A calibration signal denotes a period in which all sensors at the monitoring station are currently undergoing calibration or maintenance.

The types of signals are treated differently in the event detection calculations. Events are identified based on WQ signals only; OP, ALM, and CAL signals cannot be used to identify events. OP signals can be used indirectly to identify events as part of a composite signal or as part of pattern matching. ALM or CAL signals are used to indicate when water quality data should be ignored.

The different signals are also treated differently in the output graphs. When the *evaluation type* parameter is set as OP, the y-axis label on the plot is shown in green, indicating it is an operational signal and not used to identify events. When the *evaluation type* parameter is set as CAL, the plot is not shown in the output graph. As an example, in the StationB_wq.yml file located in the “Tutorial_Files\Configuration_Tutorial\Evaluation_Type\Temp_op” directory, temperature is defined as an operational signal. Figure 8 shows part of the *signals* section of the YML file and the temperature plot is shown in Figure 9 with the green y-axis label given to all OP signals.

```
- id: TEST_TEMP
  SCADA tag: B_TEMP_VAL
  evaluation type: op
  parameter type: TEMP
  ignore changes: none
  data options: # DATA
    precision: 0.1
    units: '^oF'
    valid range: [32, .inf]
    set points: [-.inf, .inf]
```

Figure 8: Signals configuration section with the *evaluation type* parameter as OP.

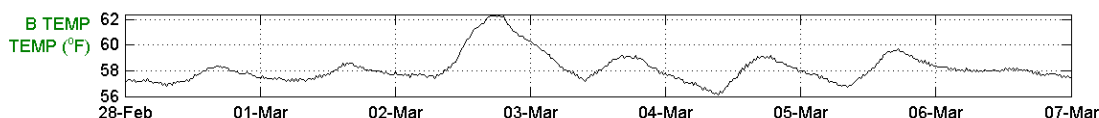


Figure 9: Operational signal plot.

If the *evaluation type* parameter is set to WQ as in Figure 10, the color of the y-axis label changes from green to black in the output plot (Figure 11) indicating that the data is included in the detection analysis. This change was made to *signals* section of the StationB_wq.yml file located in the directory, “Tutorial_Files\Configuration_Tutorial\Evaluation_Type\Temp_wq.”


```

- id: TEST_TEMP
  SCADA tag: B_TEMP_VAL
  evaluation type: wq
  parameter type: TEMP
  ignore changes: none
  data options: # DATA
    precision: 0.1
    units: '^oF'
    valid range: [32, .inf]
    set points: [-.inf, .inf]

```

Figure 10: Signals configuration section with the *evaluation type* parameter as WQ.

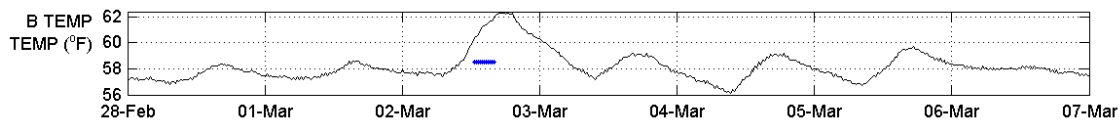


Figure 11: Water quality signal plot.

Signals defined as calibration (CAL) or alarm (ALM) are not displayed in the output graphs. In Figure 12, the calibration signal CAL_StationB is defined as *evaluation type* CAL in the StationB_cal-to-op.yml file in the “Tutorial_Files\Configuration_Tutorial\Evaluation_Type\Cal_cal” directory. For alarm or calibration signals, the *alarm options* parameters must also be defined as opposed to the *data options* parameters for water quality signals (Figure 10) and operational signals (Figure 8). Figure 13 shows the output graph, which does not include the calibration signal CAL_StationB as calibration signals are not plotted. This figure shows all of the water quality and operational signal plots as well as the event probability plot. The probability of event is calculated with CANARY using algorithms and parameter values specified in the configuration file.

```

- id: CAL_StationB
  SCADA tag: CAL_StationB
  evaluation type: cal
  parameter type: QUALITY
  ignore changes: none
  alarm options: #ALARM
    value when active: 1

```

Figure 12: Signals configuration section with the *evaluation type* parameter as CAL.

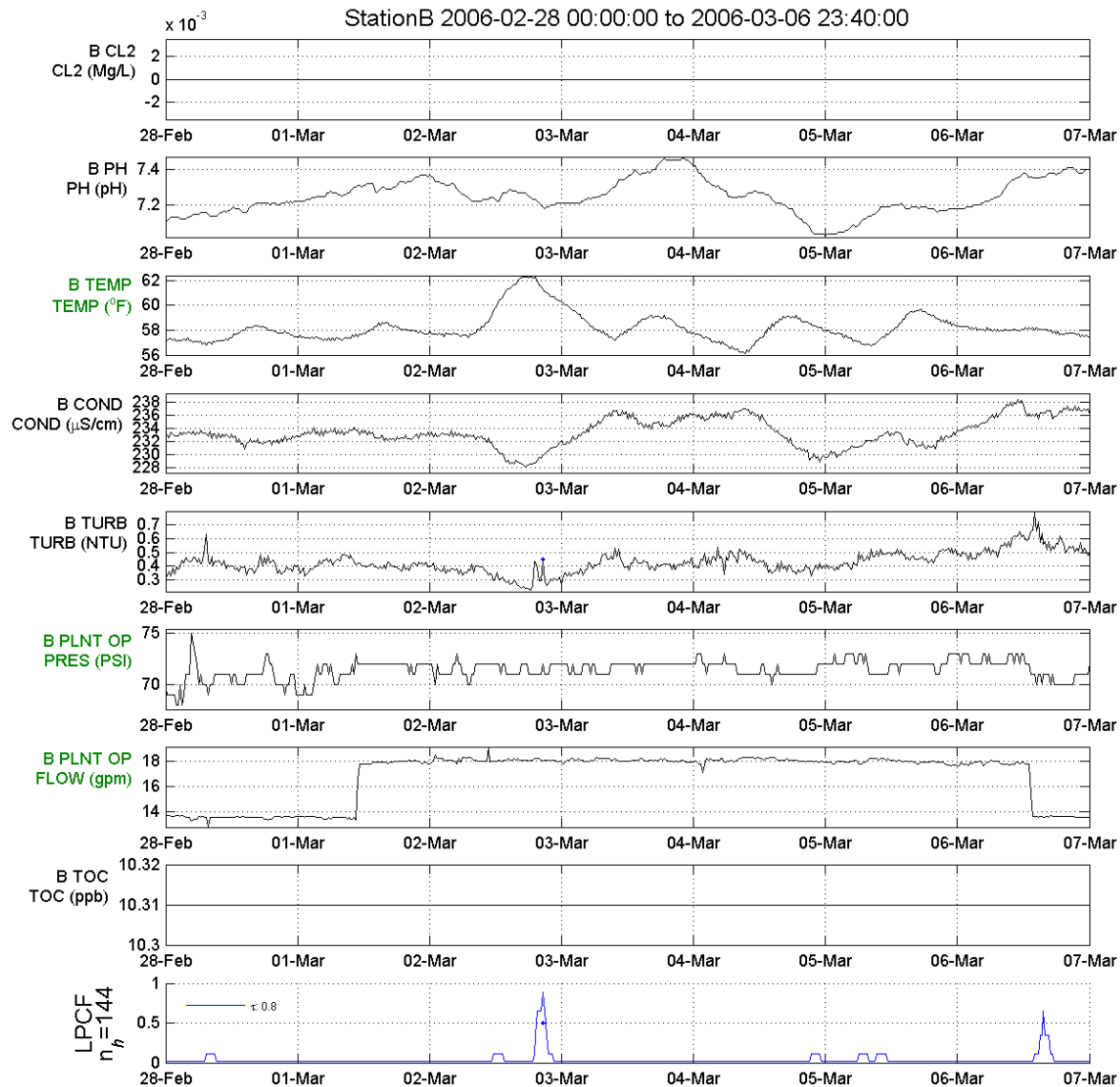


Figure 13: Output graph without calibration signal.

In Figure 14, the *evaluation type* parameter for the CAL_StationB signal is changed to OP in the StationB_cal-to-op.yml file in the directory “Tutorial_Files\Configuration_Tutorial\Evaluation_Type\Cal_op.” Figure 15 displays the output graph with the CAL_StationB signal at the top in green. Note the calibration is zero during this one week period as there are no calibration events during this week.

```
- id: CAL_StationB
  SCADA tag: CAL_StationB
  evaluation type: op
  parameter type: QUALITY
  ignore changes: none
  data options: # DATA
    precision: 0.0001
    units: ''
    valid range: [-.inf, .inf]
    set points: [-.inf, .inf]
```

Figure 14: *Signals* configuration section with the *evaluation type* parameter switched to OP.

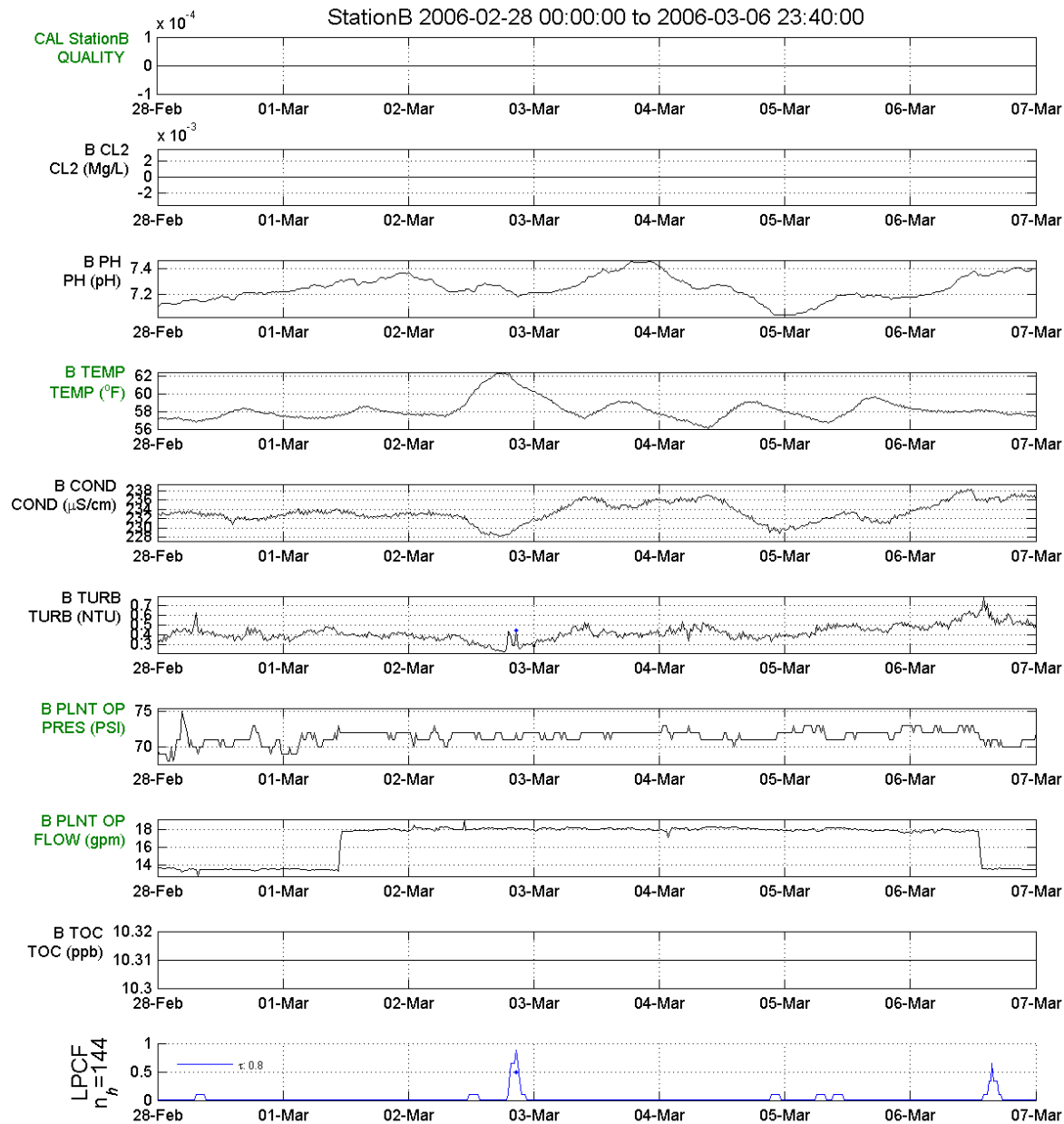


Figure 15: Output graph with the *evaluation type* parameter switched to OP.

A simple, useful trick to verify that a calibration signal is performing as expected is to temporarily change the *evaluation type* parameter from CAL to OP and examine the output graph. After verification, the *evaluation type* parameter needs to be switched back to CAL. This trick is especially useful when constructing composite signals as discussed later in this document. It is important to note that multiple CAL signals can be defined in a configuration file, but only one can be enabled for each station. If multiple calibration signals are needed, the signals must be changed to operational signals (OP) and combined using a calibration composite signal (CAL). Section 0 describes the process in more detail.

2.4 History Window

This tutorial examines the *history window* parameter which specifies the number of previous data points used to predict the next value of a water quality signal. This is a parameter in the *algorithms* section of the configuration file that determines how much historical data is used by CANARY at any given time step, to make a prediction about the next time step. The *history window* is a fixed length but moves forward in time: for example, if the *history window* parameter is two days, on Monday, CANARY will use data from Saturday and Sunday to make predictions, and on Tuesday, CANARY will use data from Sunday and Monday.

The value assigned to the *history window* parameter is applied to all water quality signals included in the monitoring station definition in the configuration file. In general, there is an optimal value for the *history window* parameter value that yields more accurate predictions. When applying CANARY to multiple water utility monitoring stations, a window size of 1.5 to 2.0 days has proven to be the most accurate; smaller or larger values resulted in decreased accuracy. For many water quality signals, there are diurnal patterns of variability, and so a *history window* parameter value of one day or more makes sense. While including a large number of days might seem like it would increase accuracy, for many water quality signals, data from weeks or months ago do not add much additional value to the analysis. For the example shown in Figure 16, the time steps are 20 minutes long and thus a *history window* parameter value of 144 is equivalent to two days.

```
algorithms:
|- id: test
  type: LPCF
  history window: 144
  outlier threshold: 0.8
  event threshold: 0.85
  event timeout: 12
  event window save: 30
| BED:
  window: 6
  outlier probability: 0.5
```

Figure 16: *Algorithms* configuration section with a *history window* parameter of 144.

Figure 17 shows the results of running CANARY with the stationB_.yml file contained in the “Tutorial_Files\Configuration_Tutorial\History_Window\HW_144” directory. The plot shows one week of CANARY results. The label “LPCF $n_h = 144$,” shown on the side of the probability of event plot, defines the algorithm used (LPCF) and the number of time steps in the history window ($n_h = 144$). Two events, on March 23rd and March 27th, are identified by the blue lines (when the probability of an event as predicted by LPCF exceeds the threshold value). The first event is due to an increase in the turbidity signal of station B (B TURB), while the second is due to increases in both the turbidity (B TURB) and chlorine (B CL2) signals. The probability of an event also rises above zero at several other times during this week but does not exceed the threshold and so is not labeled an event.

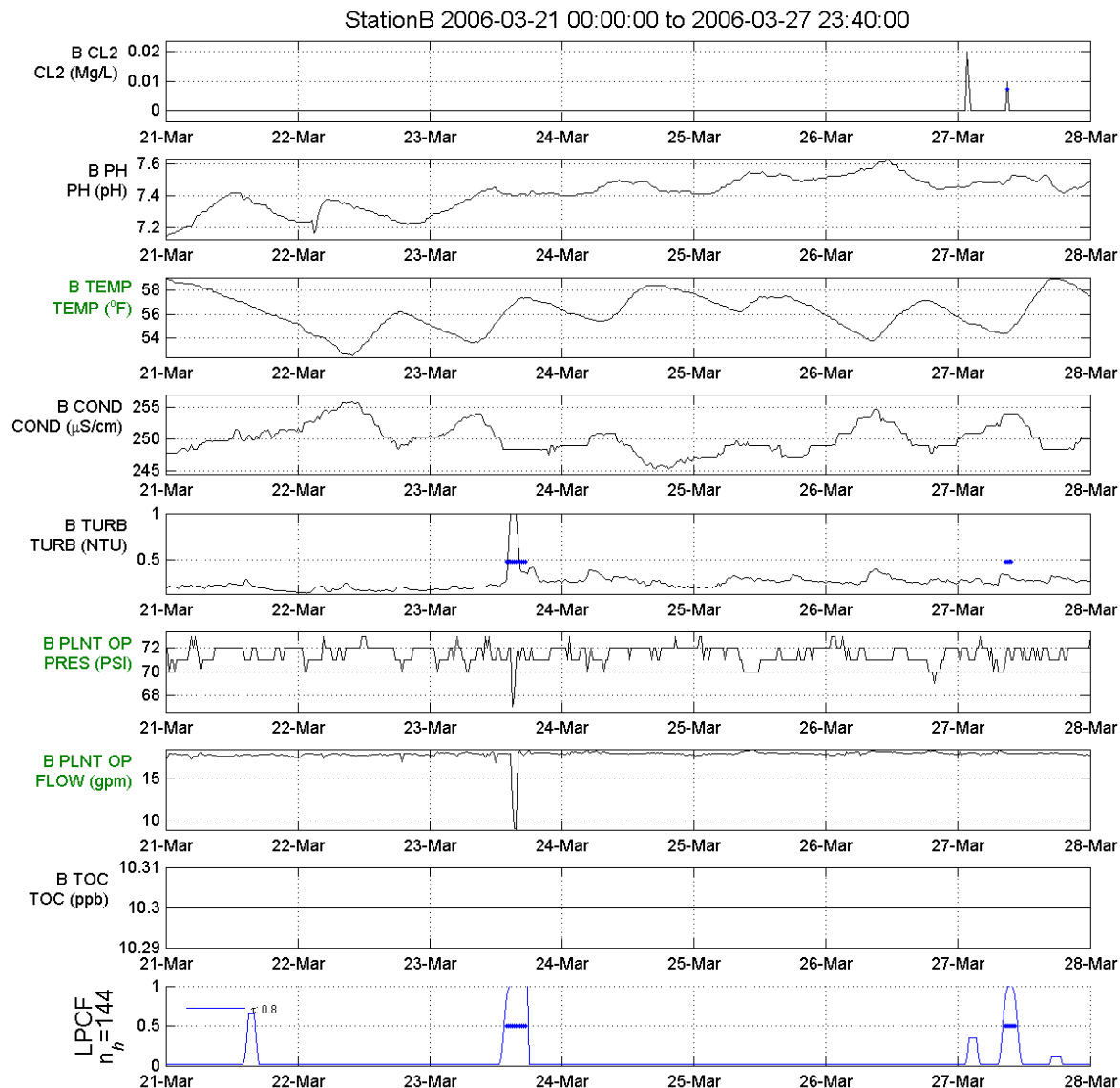


Figure 17: Output graph produced when the *history window* parameter is 144.

In order to evaluate the impact of the *history window* parameter on the results, the value is reduced from 144 to 36 (2 days to 12 hours) and the analysis is redone. Figure 18 highlights the change made to the configuration file, stationB_.yml file contained in the “Tutorial_Files\Configuration_Tutorial\History_Window\HW_36” directory. All other parameters within the configuration file are kept the same. Figure 19 shows the results of running CANARY on the modified YML file. Instead of identifying 2 events, 12 events are now identified. With the small history window, CANARY is not able to accurately predict future water quality signals. Every new data value is far from CANARY’s prediction and so the algorithms identify multiple events. This illustrates the importance of selecting appropriate values for each of the parameters in the configuration file: a CANARY user might assume these 12 events indicate real water quality anomalies instead of realizing that the configuration parameters were not selected appropriately. A *history window* parameter value of half a day is

not appropriate for this monitoring station.

```
algorithms:  
- id: test  
  type: LPCE  
  history window: 36  
  outlier threshold: 0.8  
  event threshold: 0.85  
  event timeout: 12  
  event window save: 30  
  BED: #BED  
    window: 6  
    outlier probability: 0.5
```

Figure 18: *Algorithms* configuration section with a *history window* parameter of 36.

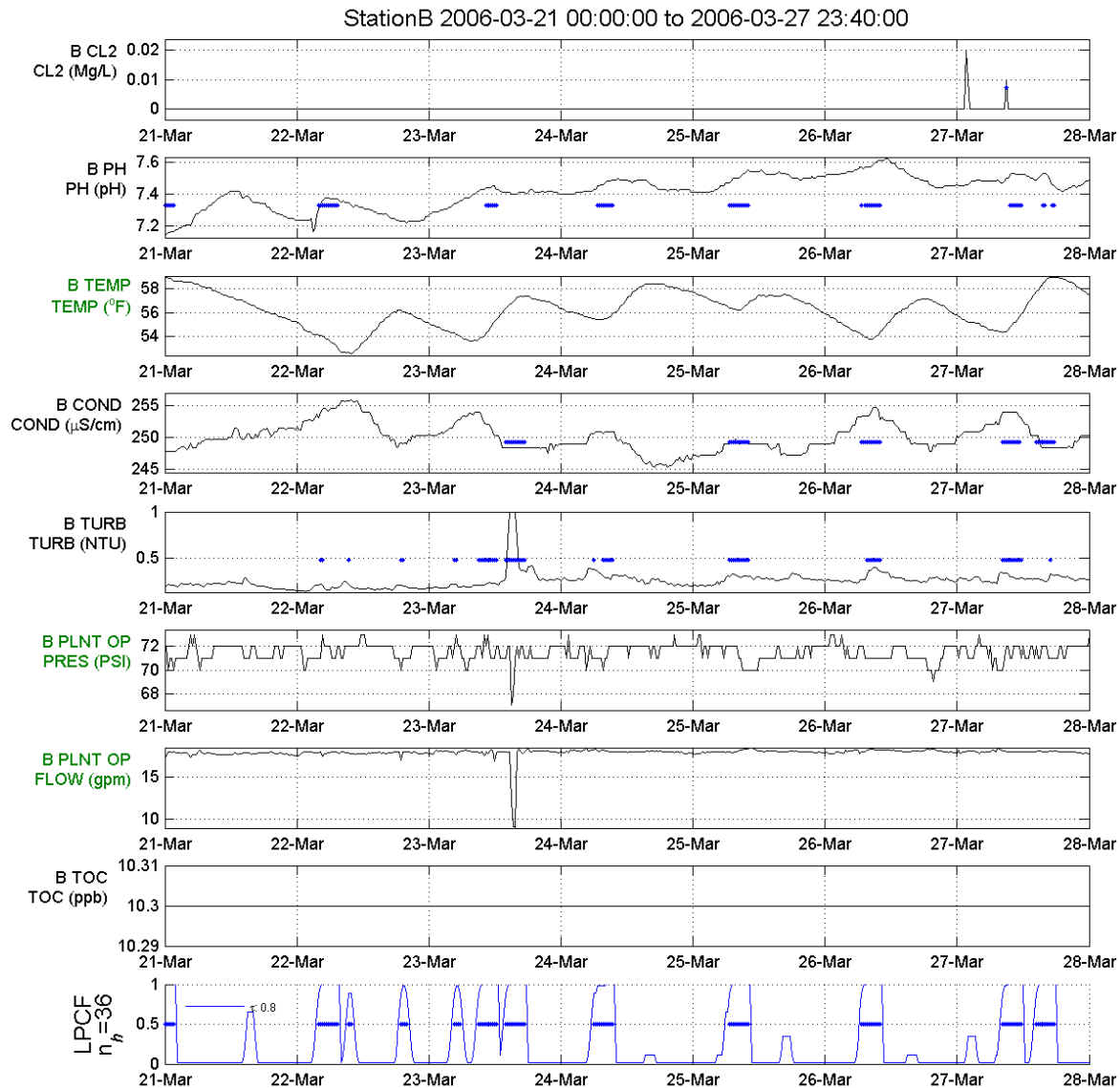


Figure 19: Output graph produced when the *history window* parameter is 36.

2.5 Outlier Threshold

This tutorial examines the *outlier threshold* which determines if an observed data value is considered to be an outlier or within the normal background range of values. The residual is the difference between the predicted water quality value and the observed water quality value at a single time step. If the absolute value of the residual is larger than the *outlier threshold* parameter, it is an outlier. The *outlier threshold* parameter is defined in units of standard deviations. Comparison of the size of the residual to the *outlier threshold* parameter is done at every time step for every water quality signal and the maximum residual value across all signals is retained for comparison to the *outlier threshold* parameter. The *algorithms* section of the initial configuration file is shown in Figure 20, with the *outlier threshold* parameter of 0.80 highlighted. This is part of the stationB.yml file in the directory “Tutorial_Files\Configuration_Tutorial\Outlier_Threshold\Initial.” The resulting event

probability plot is shown in Figure 21, where two distinct events are identified.

```
algorithms:
|- id: test
  type: LPCF
  history window: 144
  outlier threshold: 0.8
  event threshold: 0.85
  event timeout: 12
  event window save: 30
| BED: # BED
  window: 6
  outlier probability: 0.5
```

Figure 20: Algorithms configuration section with an *outlier threshold* parameter of 0.8.

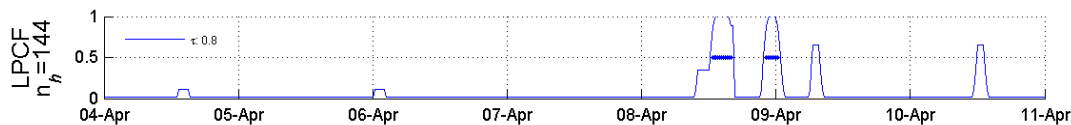


Figure 21: Event probability plot produced when the *outlier threshold* parameter is 0.8.

Increasing the value of the *outlier threshold* parameter will reduce the number of time steps classified as outliers and, thus, the number of events, which makes the event detection algorithm less sensitive in terms of detecting significant changes in the water quality signal. In the stationB_OT.99.yml file in the directory “Tutorial_Files\Configuration_Tutorial\Outlier_Threshold\Change,” the value of the *outlier threshold* parameter is increased from 0.8 to 0.99 as highlighted in Figure 22. The probability of an event plot produced from running CANARY with this increased value is shown in Figure 23. Only one event, on April 8th, is identified with this increased threshold value compared to the three events identified with the smaller threshold value.

Changing the *outlier threshold* and the *history window* parameter values affects the sensitivity of CANARY at a specified monitoring station. As a general guide, the *history window* parameter should be set large enough to include two days of previous data and the *outlier threshold* parameter should be adjusted to obtain the desired event detection sensitivity at the monitoring station. Typically, the *outlier threshold* parameter will be near 1.0. Additional examples of adjusting the *outlier threshold* and *history window* parameters are covered later in this document.

```

algorithms:
- id: test
  type: LPCF
  history window: 144
  outlier threshold: 0.99
  event threshold: 0.85
  event timeout: 12
  event window save: 30
  BED: # BED
    window: 6
    outlier probability: 0.5

```

Figure 22: *Algorithms* configuration section with an *outlier threshold* parameter of 0.99.

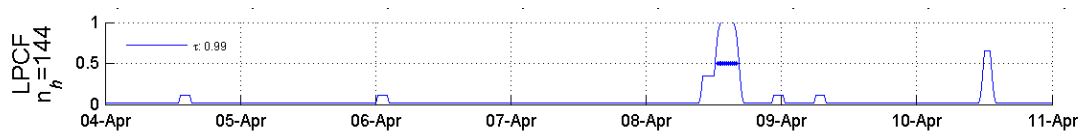


Figure 23: Event probability plot produced when the *outlier threshold* parameter is 0.99.

2.6 Event Threshold

This tutorial examines the *event threshold* parameter which defines the maximum event probability that must be exceeded before a group of outliers is identified as an event. Note the *outlier threshold* parameter is related to a single outlier value, while the *event threshold* parameter is for a group of outliers. For the initial run, the *event threshold* parameter is set to 0.85 (highlighted in Figure 24 which shows the *algorithms* section of the *station_et.yml* file in the “Tutorial_Files\Configuration_Tutorial\ Event_Threshold\Initial” directory).

```

algorithms:
|- id: test
  type: LPCF
  history window: 144
  outlier threshold: 0.8
  event threshold: 0.85
  event timeout: 12
  event window save: 30
| BED: # BED
  window: 6
  outlier probability: 0.5

```

Figure 24: *Algorithms* configuration section with an *event threshold* parameter of 0.85.

Figure 25 shows event probabilities for one week of data using an *event threshold* parameter of 0.85. A total of eight distinct events were identified during this week.

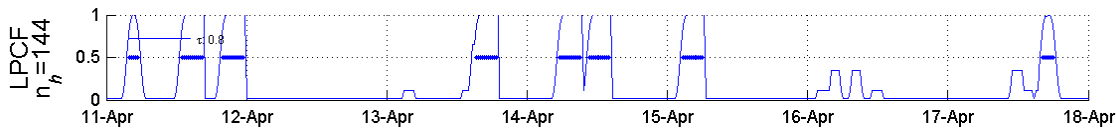


Figure 25: Event probability plot produced when the *event threshold* parameter is 0.85.

When the *event threshold* parameter is increased to 0.99, as highlighted in Figure 26, only six events are identified in Figure 27, two less than at the previous setting. As the threshold is increased, CANARY is less sensitive, detecting fewer events. This illustrates the importance of selecting the appropriate value for the *event threshold* parameter with the desired detection sensitivity.

```
algorithms:
|- id: test
  type: LPCF
  history window: 144
  outlier threshold: 0.8
  event threshold: 0.99
  event timeout: 12
  event window save: 30
| BED: # BED
  window: 6
  outlier probability: 0.5
```

Figure 26: *Algorithms* configuration section with an *event threshold* parameter of 0.99.

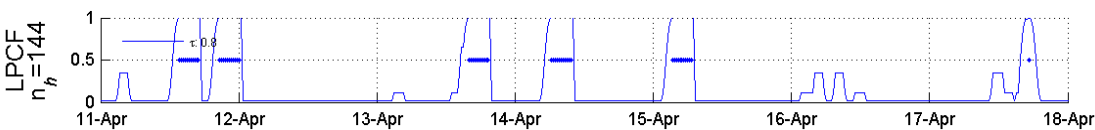


Figure 27: Event probability plot produced when the *event threshold* parameter is 0.99.

2.7 Multiple Locations

Multiple monitoring stations can be included in a single configuration file. This is common in online mode when multiple stations are run from a single launch of CANARY. For example, a water utility might have five monitoring stations that contain three sensors each (pH, CL2, and COND). All of the monitoring data is transmitted wirelessly to the utility's SCADA database. A single CANARY configuration file can be created that defines all five stations and CANARY can be run in online mode analyzing the data from all five stations simultaneously. Defining multiple monitoring stations in a single configuration file is less common for offline (BATCH) mode, but it can be useful if the data for multiple stations are stored in a single CSV formatted file.

To combine two or more stations in a single configuration file, the two station's configuration details must be combined. If a configuration file already exists for each station, this process consists of adding one of the configuration files into the other. When combining the

configuration files, both stations need to be listed under the *monitoring stations* configuration section, all input signals for both stations need to be listed with the correct names, and each station needs to be assigned an algorithm.

Figure 28 shows the example configuration file, MultiStation.yml found in the “Tutorial_Files\ Configuration_Tutorial\Multiple_Locations” directory. This file defines two stations that both use the same test algorithm; Station B has more input signals than Station A.

When a configuration file with multiple monitoring stations is run, multiple EDSD output files are created; one for each station and one with results from all of the stations. Figure 29 shows the output EDSD files created when using the configuration file shown in Figure 28.

The single EDSD file with the combined results can be used to graph each station separately. The EDSD files generated are specific to each station and so are the resulting graphs. Figure 30 and Figure 31 are examples of graphs for Stations A and B, respectively. Since the number of signals within each station is different, the resulting number of plots is also different. The graphs are from the same week and both show variability in their respective signals and different number of events.

```

monitoring stations:
|- id: StationB
  station id number:
  station tag name: StationB
  location id number: 2
  enabled: yes
  inputs:
    - id: stationB_in
  outputs:
    - id: stationA_out
  signals:
    - id: PRESS_CAL
    - id: B_CHLORINE1
    - id: B_PH
    - id: B_ORP
    - id: B_TEMP
    - id: B_COND
    - id: B_PRESS
    - id: DEL_PRESS
  algorithms:
    - id: test

|- id: StationA
  station id number:
  station tag name: StationA
  location id number: 1
  enabled: yes
  inputs:
    - id: stationA_in
  outputs:
    - id: stationA_out
  signals:
    - id: CAL_StationA
    - id: A_CHLORINE
    - id: A_PH
    - id: A_TEMP
    - id: A_COND
    - id: A_TOC
  algorithms:
    - id: test

```

Figure 28: Monitoring stations configuration section.




Name	Size	Type ▲
 MultiStation.edsd	354 KB	CANARY Data File
 MultiStation.StationA.edsd	80 KB	CANARY Data File
 MultiStation.StationB.edsd	347 KB	CANARY Data File

Figure 29: Output EDSD files.

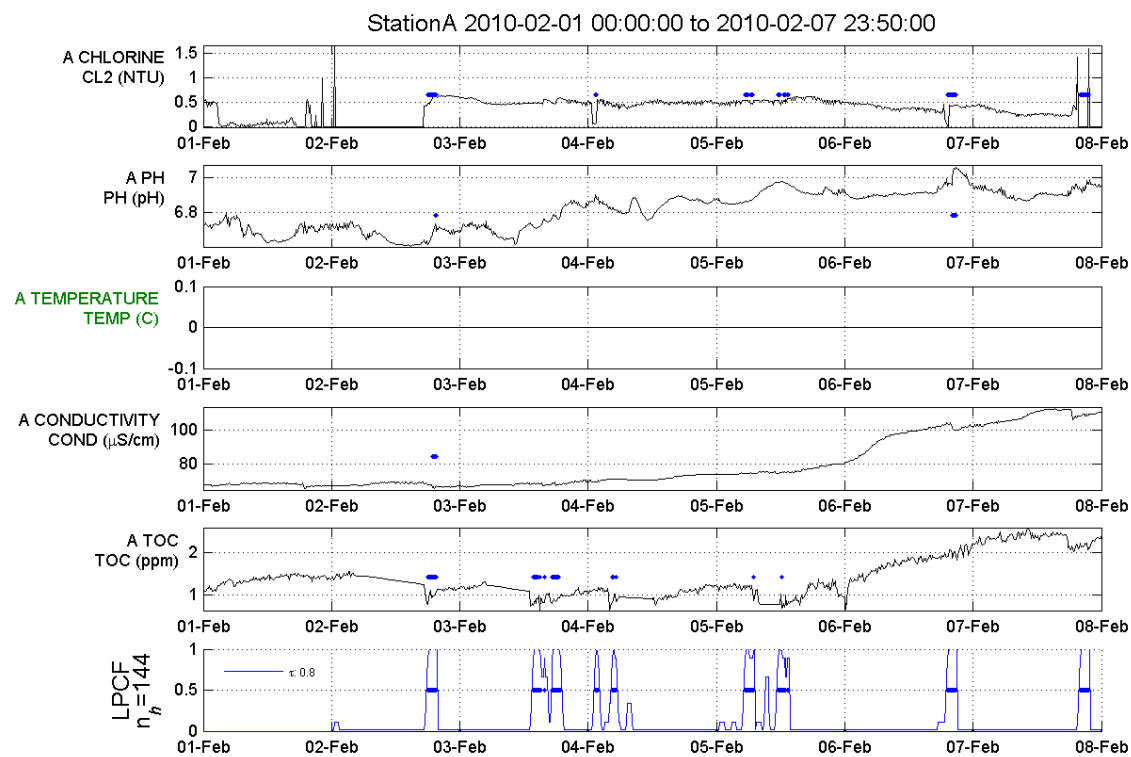


Figure 30: Station A output graph.

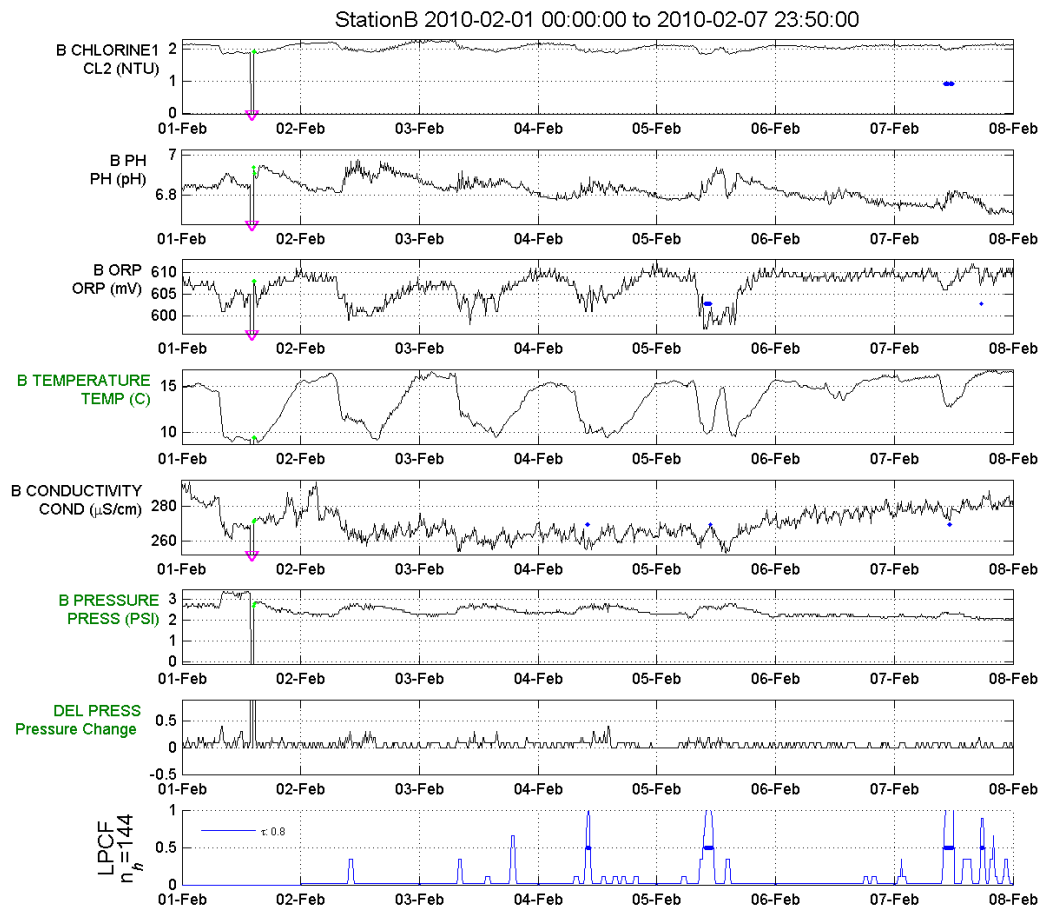


Figure 31: Station B output graph.

3. Optimizing CANARY Configurations Tutorial

Optimization of CANARY configuration parameters is an iterative process to determine the ideal set of configuration parameters. Typically, the goal of optimizing the configuration parameters is to minimize false positive detections while maximizing detection of true events. False positive detections occur when CANARY identifies an event that the user does not consider to be a real event. The configuration parameters can be adjusted to minimize false positives if historical data is available that contains no real events. True detection occurs when CANARY detects an event that the user has determined is a real event. In order to configure CANARY to maximize detection of true events, data containing real events must be available. In most cases, data containing multiple large-scale real incidents is not available; however, data containing events such as pipe breaks or other routine water quality events should be used to help maximize the true detection rate.

Determining the ideal set of parameters often requires a number of offline runs on historical data with different parameter settings. The results are compared to determine which parameters are best for each monitoring station. The computational burden of these runs can be decreased by defining different algorithms, each with a different parameterization, in the same configuration file and using a single CANARY run to obtain results.

Typically, the optimization process is conducted for each monitoring station separately, and involves:

- selecting the most appropriate algorithm (e.g., LPCF, MVNN, or one of the consensus algorithms)
- selecting the ideal parameter values for the algorithm
- adjusting the Binomial Event Discriminator (BED) parameters to increase/decrease the sensitivity of the alarm detection as well as to adjust the delay between the onset of an event and the identification of that event.

Algorithms are discussed in Section 2.4.2 of the CANARY User's Manual (Hart and McKenna 2012). The files used for the tutorials in this section are located in the "Tutorial_Files\Optimizing_Tutorials" directory.

3.1 Multiple Algorithms

This tutorial examines different combinations of the *history window* and *outlier threshold* parameters for the LPCF algorithm at a single monitoring station location. The goal is to select values of the *history window* and *outlier threshold* parameters that minimize false positives and maximize detection at this monitoring station. The LPCF algorithm type has already been selected for use, and the BED parameters have been set (BED parameters are discussed further in Section 3.3 and Appendix B of this document, and Section 2.5.1 of the CANARY User's Manual (Hart and McKenna 2012). For this tutorial, the *outlier threshold* parameter is varied from 0.6 to 1.0 and the *history window* parameter is varied from 36 to 180.

For this example, it is unknown if the data over this time period contains true water quality events. It is assumed that the data set is representative of normal background conditions with no true water quality events. Therefore, in this example, the combination of parameter values that

minimizes the number of detected events should be selected in order to minimize the number of false positives. Ideally, if data with real events were available for this example, the true detection rate could be maximized. For more information about optimizing configuration, see Murray et al. 2010.

For the first exercise, the configuration file, `stationB_multialgorithm_HW36.yml`, and the historical data file, `Tutorial_Station_B.csv`, are located in the directory “`Tutorial_Files\Optimizing_Tutorials\Multiple_Algorithms\HW_36`”. To begin, the *history window* parameter is set to 36 time steps. The data interval in the *timing options* section of the YML file is set to 20 minutes; therefore, 36 time steps is the equivalent of 12 hours. Figure 32 shows the *algorithms* section of the YML file where the *history window* parameter is set to 36 and five test algorithms are defined with *id* parameters of test1 through test5. The only differences in the test algorithms are the values of the *outlier threshold* parameter, which range from 0.60 (algorithm test1) to 1.0 (algorithm test5). Four of these definitions are shown in Figure 32 (algorithm test5 is not shown). The *monitoring stations* section of the YML with the five algorithms enabled is shown in Figure 33.

```

algorithms:
|- id: test1
  type: LPCF
  history window: 36
  outlier threshold: 0.6
  event threshold: 0.85
  event timeout: 12
  event window save: 30
| BED: # BED
  window: 6
  outlier probability: 0.5

|- id: test2
  type: LPCF
  history window: 36
  outlier threshold: 0.7
  event threshold: 0.85
  event timeout: 12
  event window save: 30
| BED: # BED
  window: 6
  outlier probability: 0.5

|- id: test3
  type: LPCF
  history window: 36
  outlier threshold: 0.8
  event threshold: 0.85
  event timeout: 12
  event window save: 30
| BED: # BED
  window: 6
  outlier probability: 0.5

|- id: test4
  type: LPCF
  history window: 36
  outlier threshold: 0.9
  event threshold: 0.85
  event timeout: 12
  event window save: 30
| BED: # BED
  window: 6
  outlier probability: 0.5

```

Figure 32: *Algorithms* configuration section with a *history window* parameter of 36.

```

monitoring stations:
|- id: StationB
  station id number:
  station tag name: StationB
  location id number: -1
  enabled: yes
| inputs:
  - id: stationb_in
| outputs:
| signals:
  - id: CAL_StationB
  - id: TEST_CL
  - id: TEST_PH
  - id: TEST_TEMP
  - id: TEST_COND
  - id: TEST_TURB
  - id: TEST_PRES_PLNT
  - id: TEST_FLOW_PLNT
  - id: TEST_TOC
    cluster: no
| algorithms:
  - id: test1
  - id: test2
  - id: test3
  - id: test4
  - id: test5

```

Figure 33: Monitoring stations section with multiple algorithms enabled.

Figure 34 shows the output graph when CANARY is run on this YAML file. The graphical output produced is similar to the case with a single algorithm, except that an additional plot is added to the bottom of the graph for each of the five test algorithms. The *outlier threshold* parameter increases from algorithms test1 to test5 (top to bottom in the figure). As the threshold increases, fewer data points are considered to be outliers, and thus fewer events are detected. Thus the sensitivity of the detection decreases as the *outlier threshold* parameter increases.

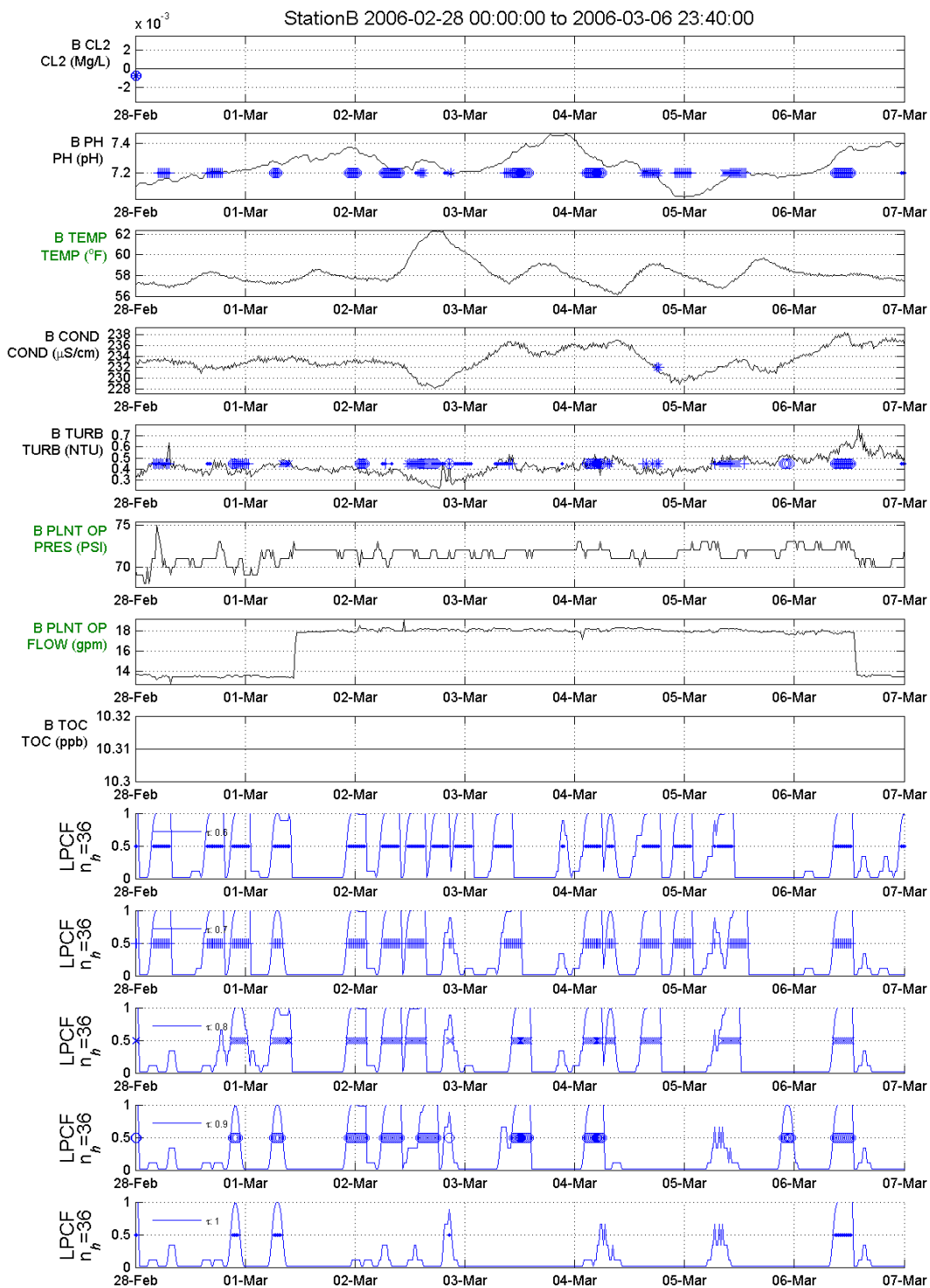


Figure 34: Output graph when the *history window* parameter is 36.

For the second exercise, the *history window* parameter is doubled to 72 time steps, or 1 day. The configuration file, stationB_multialgorithm_HW72.yml, and the historical data file,

Tutorial_Station_B.csv, are located in the directory “Tutorial_Files\Optimizing_Tutorials\Multiple_Algorithms\HW_72”. Figure 35 shows the *algorithms* section of the configuration file with the *history window* parameter set to 72 and the five corresponding algorithms definitions.

```
algorithms:
|- id: test1
  type: LPCF
  history window: 72
  outlier threshold: 0.6
  event threshold: 0.85
  event timeout: 12
  event window save: 30
| BED: # BED
  window: 6
  outlier probability: 0.5

|- id: test2
  type: LPCF
  history window: 72
  outlier threshold: 0.7
  event threshold: 0.85
  event timeout: 12
  event window save: 30
| BED: # BED
  window: 6
  outlier probability: 0.5

|- id: test3
  type: LPCF
  history window: 72
  outlier threshold: 0.8
  event threshold: 0.85
  event timeout: 12
  event window save: 30
| BED: # BED
  window: 6
  outlier probability: 0.5

|- id: test4
  type: LPCF
  history window: 72
  outlier threshold: 0.9
  event threshold: 0.85
  event timeout: 12
  event window save: 30
| BED: # BED
  window: 6
  outlier probability: 0.5
```

Figure 35: *Algorithms* configuration section with a *history window* parameter of 72.

Figure 36 shows the output graph when the *history window* parameter is 72. Note the decreased

sensitivity of the event detection relative to the previous exercise when the *history window* parameter was equal to 36 (Figure 34). For this larger *history window* parameter, only three events are identified at the two highest *outlier thresholds* parameters (bottom two plots).

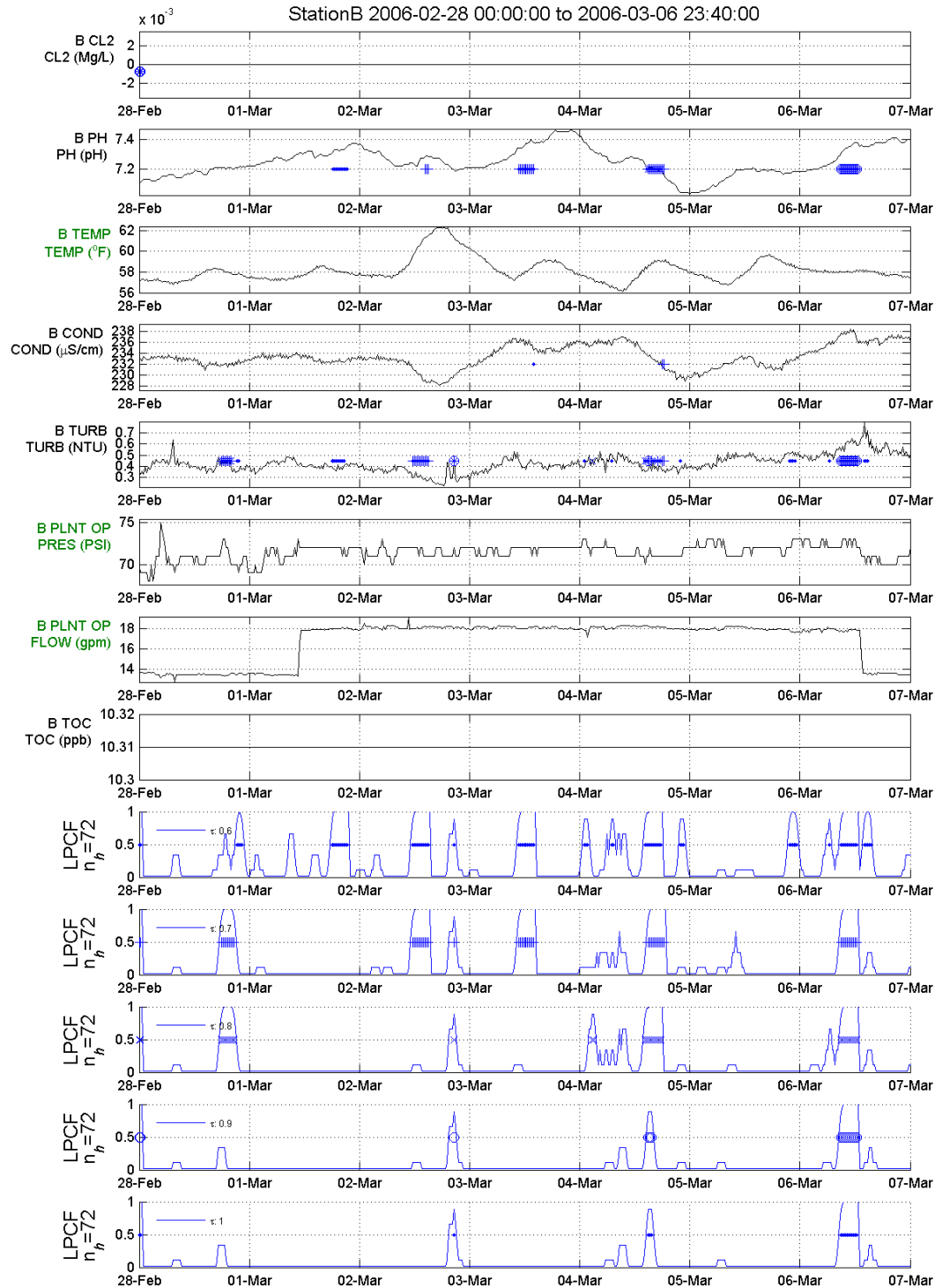


Figure 36: Output graph when the *history window* parameter is 72.

For the third exercise, the *history window* parameter is set to 108 time steps (or 1.5 days). The configuration file, stationB_multialgorithm_HW108.yml, and the historical data file, Tutorial_Station_B.csv, are located in the directory “Tutorial_Files\Optimizing_Tutorials\Multiple_Algorithms\HW_108”. Figure 37 shows the *algorithms* section of the YML file with a *history window* parameter set to 108 and the five corresponding algorithms definitions.

```

algorithms:
]- id: test1
  type: LPCF
  history window: 108
  outlier threshold: 0.6
  event threshold: 0.85
  event timeout: 12
  event window save: 30
] BED: # BED
  window: 6
  outlier probability: 0.5
.
]- id: test2
  type: LPCF
  history window: 108
  outlier threshold: 0.7
  event threshold: 0.85
  event timeout: 12
  event window save: 30
] BED: # BED
  window: 6
  outlier probability: 0.5
.
]- id: test3
  type: LPCF
  history window: 108
  outlier threshold: 0.8
  event threshold: 0.85
  event timeout: 12
  event window save: 30
] BED: # BED
  window: 6
  outlier probability: 0.5
.
]- id: test4
  type: LPCF
  history window: 108
  outlier threshold: 0.9
  event threshold: 0.85
  event timeout: 12
  event window save: 30
] BED: # BED
  window: 6
  outlier probability: 0.5

```

Figure 37: *Algorithms* configuration section with a *history window* parameter of 108.

Figure 38 shows the corresponding output graph. The longer *history window* parameter leads to even fewer events detected (Figure 38), but the decrease is not as dramatic as seen for the initial increase in the *history window* parameter from 36 to 72 time steps.

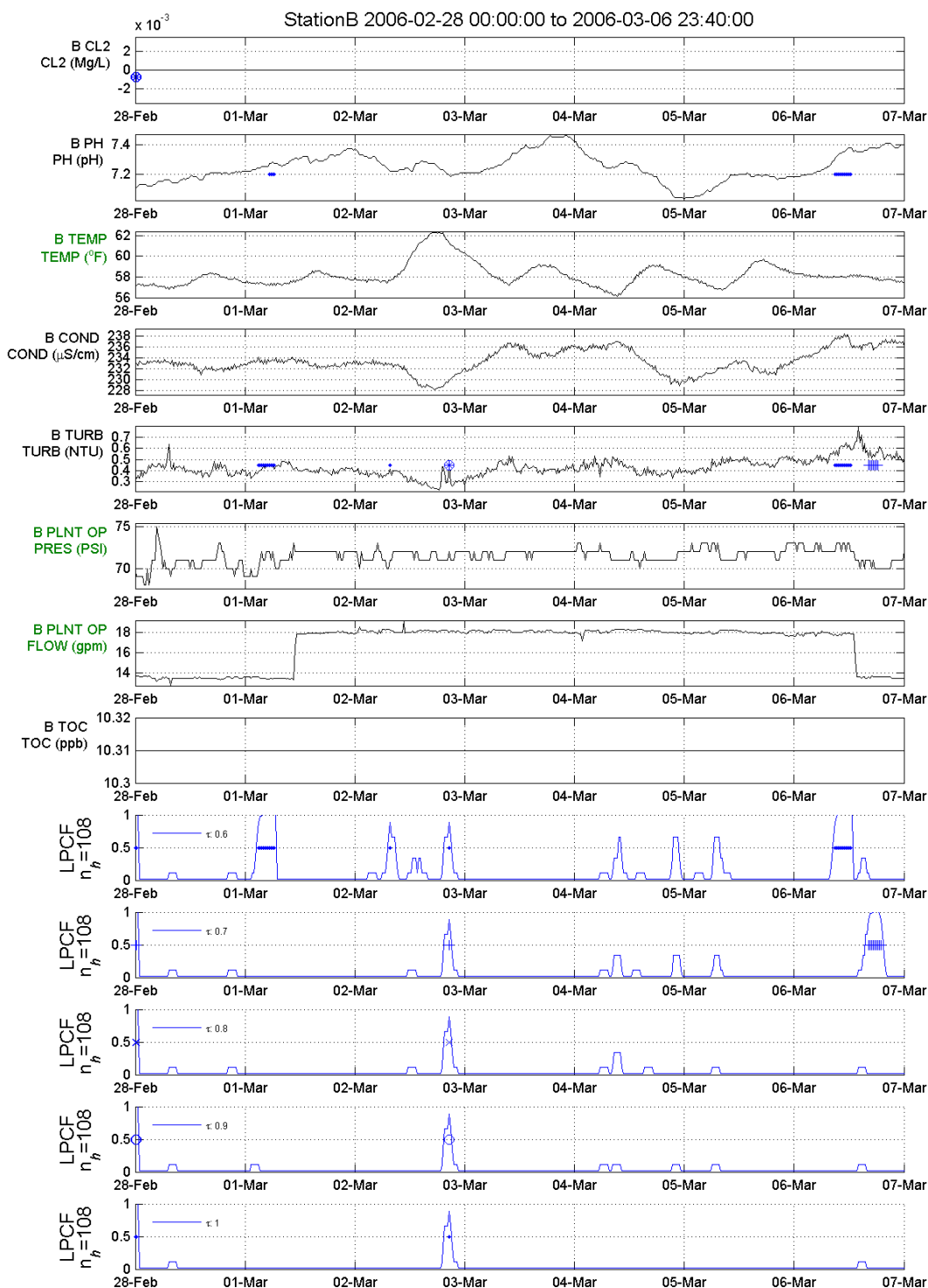


Figure 38: Output graph when the *history window* parameter is 108.

For the fourth exercise, the *history window* parameter is set to 144 (2 days). The configuration file, stationB_multialgorithm_HW144.yml, and the historical data file, Tutorial_Station_B.csv, are located in the “Tutorial_Files\Optimizing_Tutorials\Multiple_Algorithms\HW_144”

directory. Figure 39 shows the *algorithms* section of the configuration file with a *history window* parameter of 144 and the five corresponding algorithms definitions.

```
algorithms:
- id: test1
  type: LPCF
  history window: 144
  outlier threshold: 0.6
  event threshold: 0.85
  event timeout: 12
  event window save: 30
  BED: # BED
    window: 6
    outlier probability: 0.5

- id: test2
  type: LPCF
  history window: 144
  outlier threshold: 0.7
  event threshold: 0.85
  event timeout: 12
  event window save: 30
  BED: # BED
    window: 6
    outlier probability: 0.5

- id: test3
  type: LPCF
  history window: 144
  outlier threshold: 0.8
  event threshold: 0.85
  event timeout: 12
  event window save: 30
  BED: # BED
    window: 6
    outlier probability: 0.5

- id: test4
  type: LPCF
  history window: 144
  outlier threshold: 0.9
  event threshold: 0.85
  event timeout: 12
  event window save: 30
  BED: # BED
    window: 6
    outlier probability: 0.5
```

Figure 39: *Algorithms* configuration section with a *history window* parameter of 144.

Figure 40 shows the output graph when the *history window* parameter is 144 time steps using five different *outlier threshold* parameters. As the *outlier threshold* parameter increases, the number and duration of events decreases.

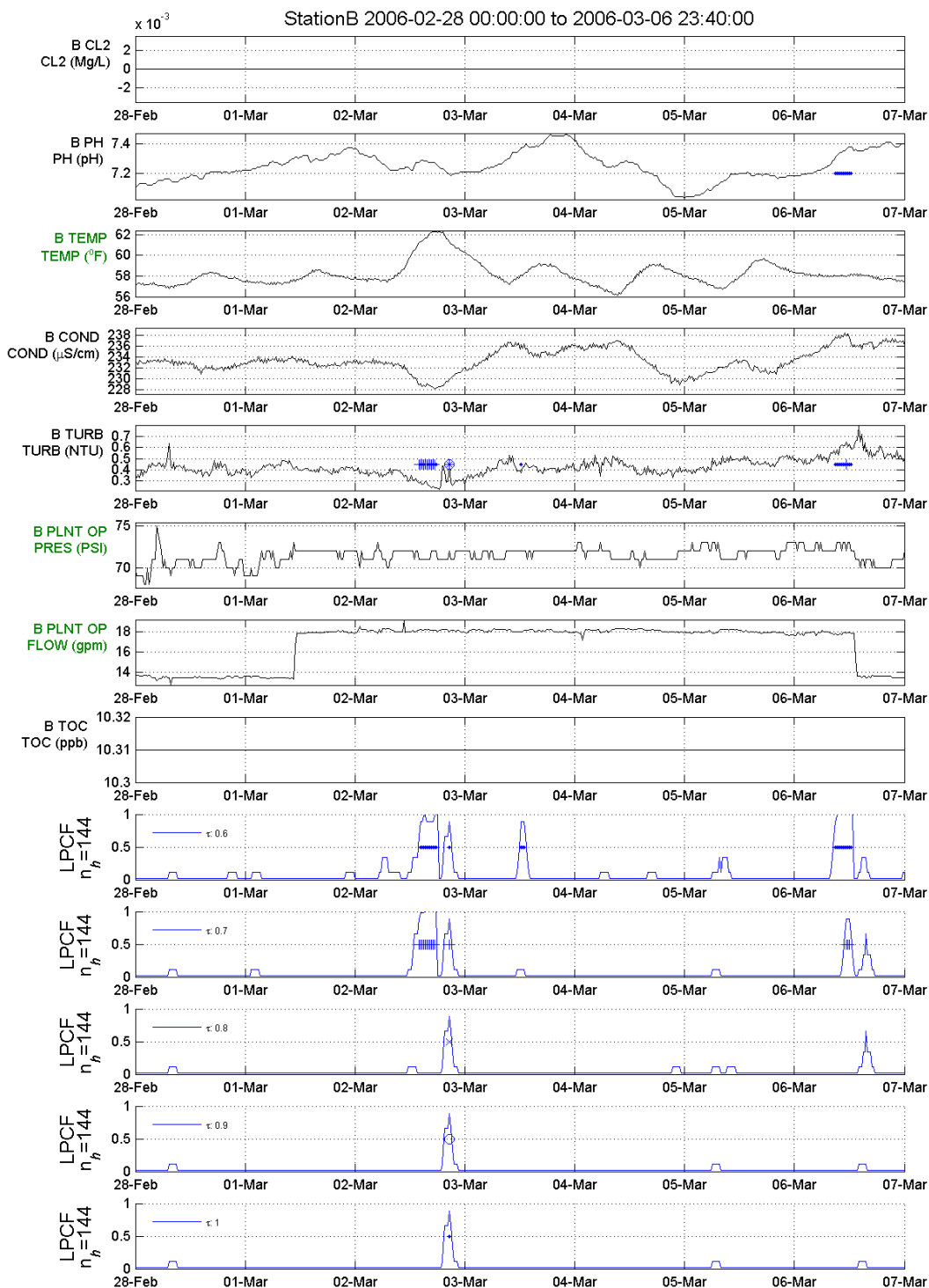


Figure 40: Output graph when the *history window* parameter is 144.

For the fifth exercise, the *history window* parameter is set to 180 time steps (2.5 days). The configuration file, `stationB_multialgorithm_HW180.yml`, and the historical data file, `Tutorial_Station_B.csv`, are located in the directory

“Tutorial_Files\Optimizing_Tutorials\Multiple_Algorithms\HW_180”. Figure 41 shows the *algorithms* section of the configuration file with a *history window* parameter of 180 and the five corresponding algorithms definitions.

```
algorithms:
- id: test1
  type: LPCF
  history window: 180
  outlier threshold: 0.6
  event threshold: 0.85
  event timeout: 12
  event window save: 30
  BED: # BED
    window: 6
    outlier probability: 0.5

- id: test2
  type: LPCF
  history window: 180
  outlier threshold: 0.7
  event threshold: 0.85
  event timeout: 12
  event window save: 30
  BED: # BED
    window: 6
    outlier probability: 0.5

- id: test3
  type: LPCF
  history window: 180
  outlier threshold: 0.8
  event threshold: 0.85
  event timeout: 12
  event window save: 30
  BED: # BED
    window: 6
    outlier probability: 0.5

- id: test4
  type: LPCF
  history window: 180
  outlier threshold: 0.9
  event threshold: 0.85
  event timeout: 12
  event window save: 30
  BED: # BED
    window: 6
    outlier probability: 0.5
```

Figure 41: *Algorithms* configuration section with a *history window* parameter of 180.

Figure 42 shows the output graph when the *history window* parameter is 180 time steps. Note that there is only one event identified for all five algorithms.

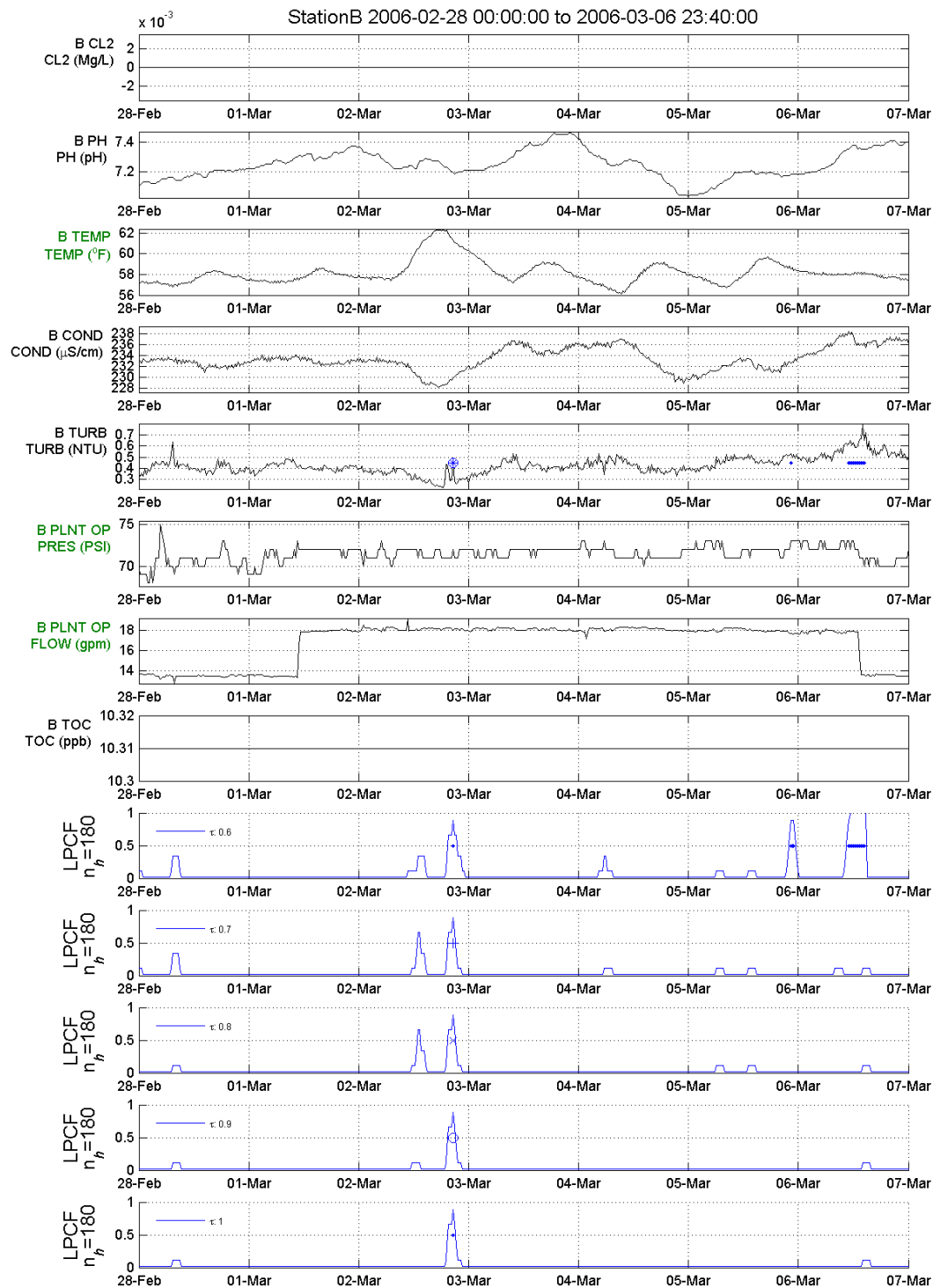


Figure 42: Output graph when the *history window* parameter is 180.

By analyzing the number of events detected for the combinations of *history window* and *outlier threshold* parameter values, the sensitivity of CANARY to these parameters for this monitoring

station can be examined. Figure 43 shows the number of events identified for each combination of *history window* and *outlier threshold* parameters for the data set from a single monitoring station over a period of 69 days. Increasing both the *history window* and *outlier threshold* parameters decrease the number of events detected. The decrease in the number of events is a nearly linear function of the *outlier threshold* parameter, while the events decrease more rapidly with an increasing *history window* parameter. If the data set is representative of normal background conditions with no true water quality events, then the combination of parameter values that minimizes the number of detected events should be selected. This would serve to minimize the number of false positives. However, since data with real events is not available for this test, it might be more effective to allow for a slightly larger number of detected events in order to ensure detection is sensitive enough to detect true hazardous contamination events. For more information about optimizing configuration, see Murray et al. 2010.

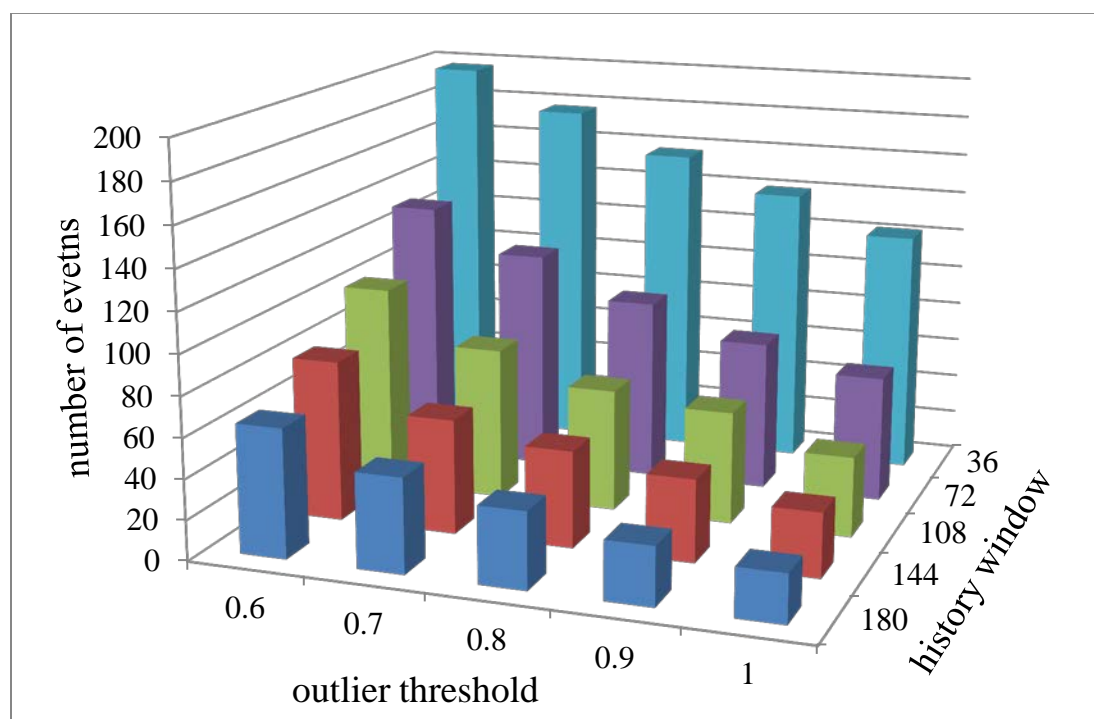


Figure 43: CANARY total events graph.

3.2 Consensus Algorithm

In some cases, the combination of two event detection algorithms provides improved results relative to a single event detection algorithm. As an example, the LPCF algorithm identifies significant changes in the relative value of a signal, but it does not provide information on the actual value of the signal. In other words, the LPCF algorithm does not care what the data value is, only how quickly it changes. Under the LPCF algorithm, a water quality signal could gradually decrease from one to zero over a week without any events being detected by CANARY. In contrast to the LPCF algorithm, a set-point algorithm detects events if the actual values of the water quality signal exceed upper or lower bounds. However, it cannot detect changes in water quality, no matter how sudden, that do not exceed either the set-point thresholds (minimum or maximum). The consensus algorithm feature within CANARY allows

both algorithms to be used simultaneously. Two consensus algorithm options (CAVE, which averages event probabilities, and CMAX, which takes the maximum event probability) are available within CANARY (see details in Section 2.4.3.5 the CANARY User's Manual (Hart and McKenna 2012)).

This tutorial examines the application of the CMAX consensus algorithm, which combines the set-point algorithm, SPPE, with the LPCF algorithm. The configuration files *CLDY_Initial.yml* and *CLDY_Join.yml* are used for this tutorial and are located in the directory "Tutorial_Files\Optimizing_Tutorials\Consensus_Algorithm". The monitoring station data used in this tutorial is contained in the data file, *CLDY_train.csv*, and includes six primary water quality parameters: residual chlorine (CL2), specific conductivity (COND), pH (PH), temperature (TEMP), turbidity (TURB), and total organic carbon (TOC). The data were collected at a two minute sampling frequency. An alarm status for each sensor, indicating a nonfunctioning sensor, was recorded. In addition, a large number of signals regarding the status of pumps and valves along with flow rates, tank levels, and pressures were recorded. A calibration signal for the entire station was provided.

The SPPE algorithm is chosen for the set-point algorithm (see Section 2.4.2.3 of the CANARY User's Manual (Hart and McKenna 2012) for additional details). In a set-point algorithm, as a water quality signal value gets closer to either the minimum or maximum set-point value, the probability of an event becomes closer to 1.0. The difference between the water quality signal value and either set-point value is measured in units of the *precision* parameter that is part of each signal definition. Figure 44 shows the signal section of the *CLDY_initial.yml* file in which three signals (CL2, COND, and PH) are defined each with distinct precision settings (0.005, 1, and 0.01). Set-point algorithms require the *set points* parameter to be set in the signal definition. This parameter defines the minimum and maximum set-point values, and the default parameter values are [-.inf, .inf]. The set-point data values for the TEST_CL and the TEST_PH signals are shown in Figure 44, which are distinct from the *valid range* parameter data values. The remaining signals, including TEST_COND, are set at the default set-point values of [-.inf, .inf], and therefore do not contribute to the set point algorithm.

```

signals:
- id: TEST_CL
  SCADA tag: CLDY_CL2X_VAL
  evaluation type: wq
  parameter type: CL2
  ignore changes: none
  data options:
    precision: 0.005
    units: 'mg/L'
    valid range: [0.0, 4]
    set points: [0.2, 1.5]

- id: TEST_COND
  SCADA tag: CLDY_COND_VAL
  evaluation type: wq
  parameter type: COND
  ignore changes: none
  data options:
    precision: 1
    units: '(\mu)S/cm'
    valid range: [300, 500]
    set points: [-.inf, .inf]

- id: TEST_PH
  SCADA tag: CLDY_PHXX_VAL
  evaluation type: wq
  parameter type: PH
  ignore changes: none
  data options:
    precision: 0.01
    units: 'pH'
    valid range: [6, 10]
    set points: [8, 9.5]

```

Figure 44: *Signals* configuration section.

The *algorithms* section of the CLDY_initial.yml file is shown in Figure 45. One algorithm is defined: B2 is the SPPE set-point algorithm with a *history window* parameter of 10 and an *outlier threshold* parameter of 80. Figure 46 shows the *monitoring stations* section of the configuration file which identifies the signals and algorithm to use in the analysis.

```

algorithms:
|- id: B2
  type: SPPE
  history window: 10
  outlier threshold: 80
  event threshold: 0.995
  event timeout: 30
  event window save: 30

```

Figure 45: *Algorithms* configuration section with the consensus algorithm defined.


```

monitoring stations:
|- id: StationD
   station id number:
   station tag name: StationD
   location id number: -1
   enabled: yes
| inputs:
  - id: stationd_in
  outputs:
| signals:
  - id: TEST_CL
  - id: TEST_COND
  - id: TEST_PH
  - id: TEST_TEMP
  - id: TEST_TOC
  - id: TEST_RES_CL
  - id: TEST_TANK_ELEV
  - id: TEST_TANK_FLOW
  - id: TEST_CAL
| algorithms:
  - id: B2

```

Figure 46: Monitoring configuration section with algorithm B2 activated.

Figure 47 shows the output graph from running this YAML file in CANARY. During this one week, the algorithm identifies two water quality events; in both cases, TEST_PH exceeds its upper set point value. The event probability plot at the bottom of Figure 47 indicates other times where the signals get close to one of the set-point values but do not exceed the set point to trigger an event. The pink triangle in the TEST_PH plot shows when the values exceed the valid range, here set to be [6, 10].

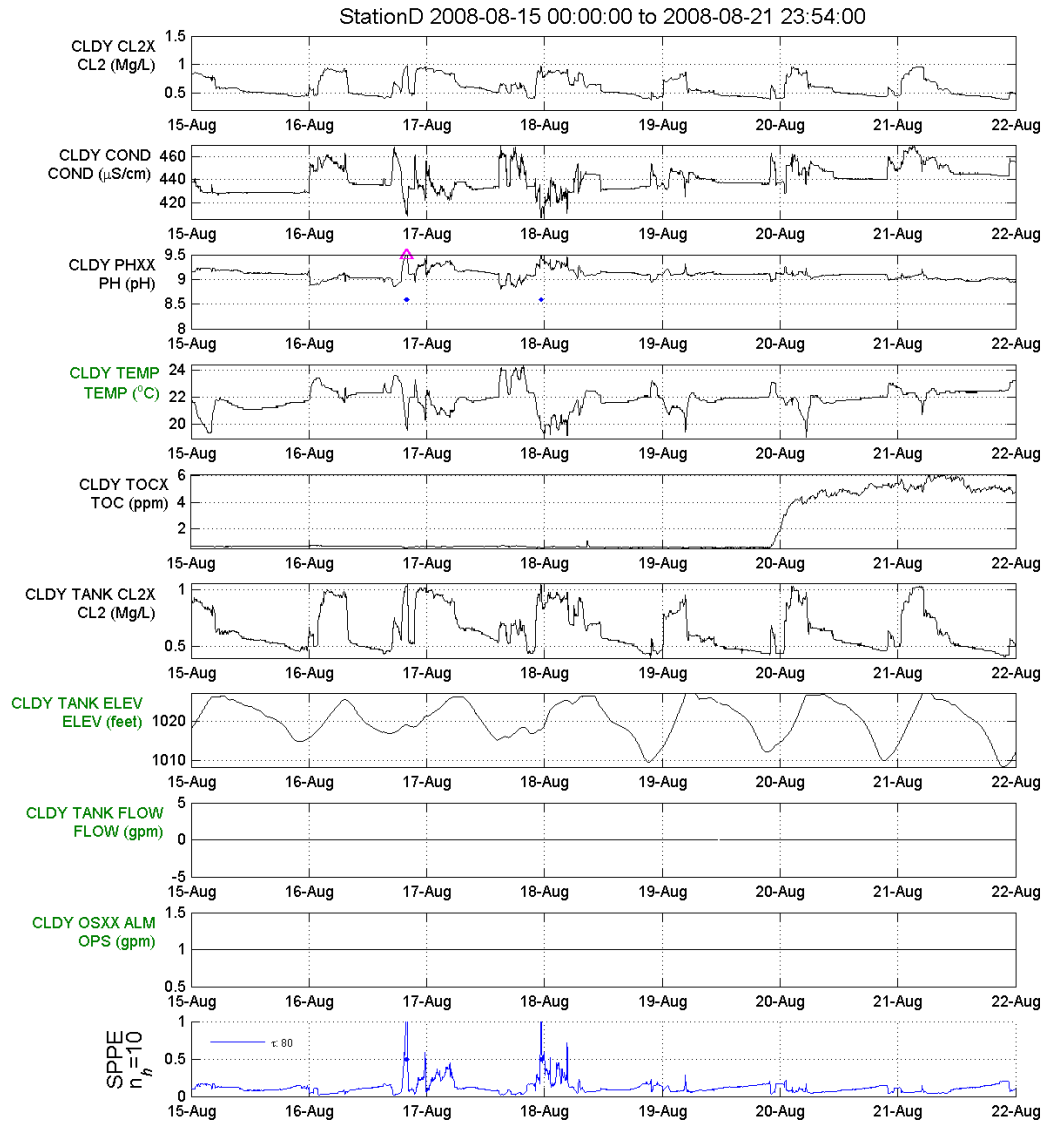


Figure 47: Output graph produced using the set point algorithm, SPPE.

In the configuration file, CLDY_Join.yml found in the directory “\Tutorial_Files\Optimizing_Tutorials\Consensus_Algorithm\Join”, as shown in Figure 48, the *algorithms* section defines three algorithms: B1 is the LPCF algorithm; B2 is the SPPE set-point algorithm; JOIN is the consensus algorithm of *type* CMAX that combines the first two algorithms together. CMAX retains the maximum event probability between two algorithms. To activate the consensus algorithm in the analysis, the algorithm’s name, JOIN, is listed under the *algorithms* parameter in the *monitoring stations* section (Figure 49), as shown in CLDY_Join.yml.

```

algorithms:
- id: B1
  type: LPCF
  history window: 700
  outlier threshold: 0.8
  event threshold: 0.995
  event timeout: 30
  event window save: 30
  BED:
    window: 12
    outlier probability: 0.5

- id: B2
  type: SPPE
  history window: 10
  outlier threshold: 80
  event threshold: 0.995
  event timeout: 30
  event window save: 30

- id: JOIN
  type: CMAX
  history window: 10
  outlier threshold: 20
  event threshold: 0.995
  event timeout: 30
  event window save: 30
  use algorithm inputs:
    - id: B1
    - id: B2

```

Figure 48: *Algorithms* configuration section with three algorithms defined.

```

monitoring stations:
]- id: StationD
  station id number:
  station tag name: StationD
  location id number: -1
  enabled: yes
] inputs:
  - id: stationd_in
outputs:
] signals:
  - id: TEST_CL
  - id: TEST_COND
  - id: TEST_PH
  - id: TEST_TEMP
  - id: TEST_TOC
  - id: TEST_RES_CL
  - id: TEST_TANK_ELEV
  - id: TEST_TANK_FLOW
  - id: TEST_CAL
] algorithms:
  - id: B1
  - id: B2
  - id: JOIN

```

Figure 49: *Monitoring stations* configuration section with the consensus algorithm enabled.

Figure 50 shows the output graph with the results from all three algorithms for the same week as shown in Figure 47. The event probability is calculated by each individual algorithm, as well as the combined algorithm. The LPCF algorithm identifies multiple events while the set point algorithm identifies only one event. The consensus algorithm used here, CMAX, combines the results from both algorithms and identifies the events identified by both separate algorithms.

In summary, the consensus algorithm allows the simultaneous use of multiple algorithms for water quality monitoring. This feature combines the strengths of different algorithms (e.g., set points for absolute signal values and LPCF for relative changes in the signal values) to improve confidence in the event detection results.

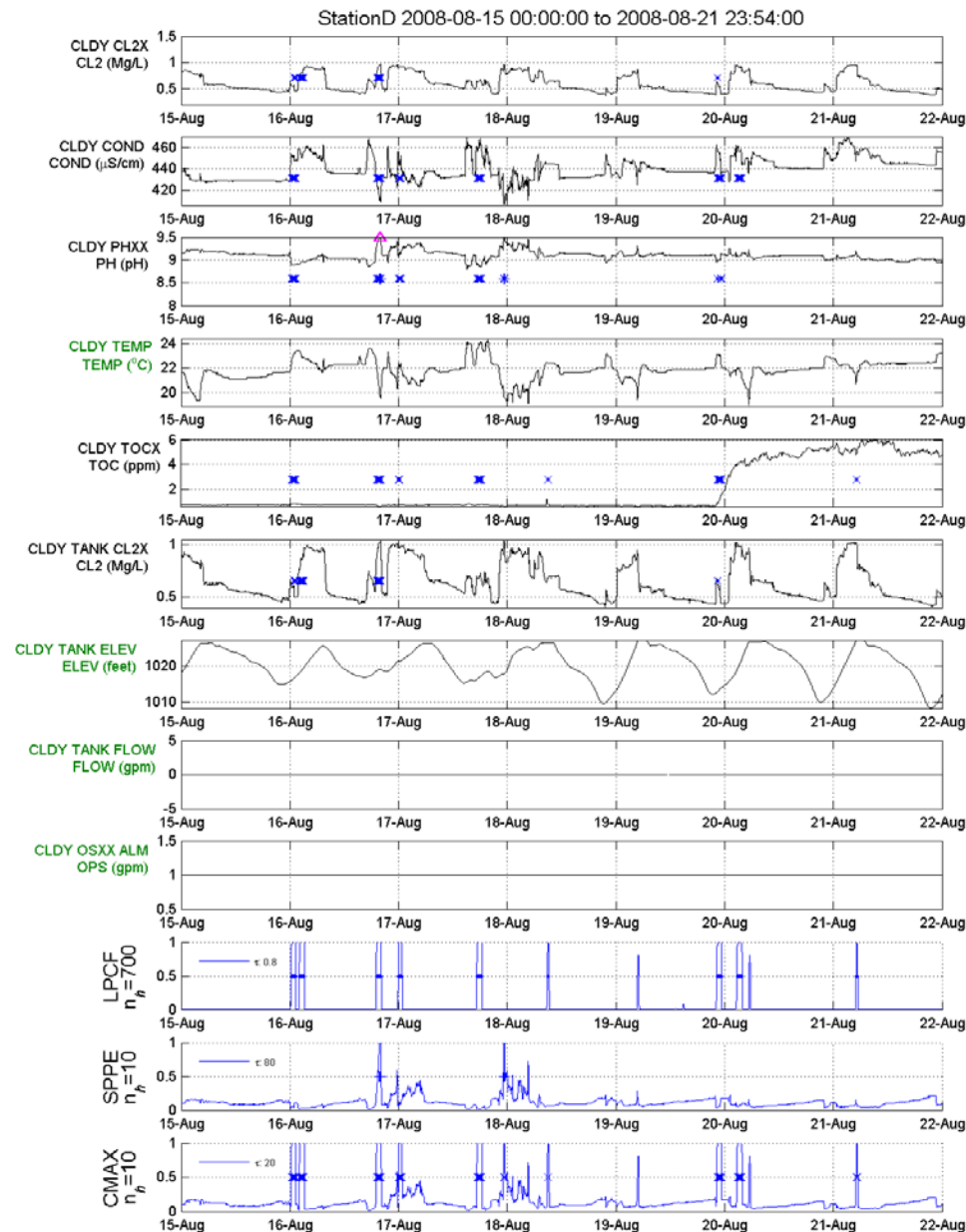


Figure 50: Output graph produced when using the consensus algorithm.

3.3 Binomial Event Discriminator

Typical utility water quality data has significant, short-lived spikes in the values, which often last only one or two time steps. Some of these spikes are due to errors in the sensors or in the transmission of data; in most cases, they do not indicate a true water quality event. In order to ignore these spikes, the Binomial Event Discriminator (BED) function within CANARY aggregates evidence of an event (i.e., outliers) over multiple consecutive time steps before identifying an event.

The BED is based on a binomial failure model that looks at the number of outliers (NFAILURES) within a certain number of time steps in a user defined window (NTRIALS-window) given a fixed probability of an outlier occurring at any time step (see Appendix B: Binomial Distribution Function Exercise in this document for more details). If there are more outliers within the window than would be predicted, then this increases the likelihood that an event is occurring. A CANARY user can adjust the BED *window* parameter to increase or decrease the sensitivity of detection.

The BED function works in conjunction with another algorithm. For example, Figure 51 shows the *algorithms* section of StationB.yml found in the “\Tutorial_Files\Optimizing_Tutorials\BED” directory. The test algorithm is defined to be of type LPCF and uses the BED function with a window of 20 time steps and an outlier probability of 0.5. The LPCF algorithm is used to identify outliers and then BED calculates the continuous probability that an event has occurred at that time step and ensures that enough outliers are present within the window before identifying an event. If the BED probability is greater than the algorithm *event threshold* parameter, then CANARY identifies an event.

This tutorial examines the BED function within CANARY by considering the influence of the BED *window* parameter. By changing the BED *window* parameter, the user can increase or decrease CANARY’s sensitivity. Figure 51 shows the *algorithms* section of StationB.yml found in the “\Tutorial_Files\Optimizing_Tutorials\BED\Window_20” directory and Figure 52 shows the *algorithms* section of StationB.yml found in the “\Tutorial_Files\Optimizing_Tutorials\BED\Window_6” directory. These files define the BED *window* parameter as 20 and 6 time steps long, respectively. The output graphs for each configuration are shown in Figure 53 and Figure 54.

```
algorithms:
- id: test
  type: LPCF
  history window: 144
  outlier threshold: 0.8
  event threshold: 0.85
  event timeout: 12
  event window save: 30
  BED: #BED
    window: 20
    outlier probability: 0.5
```

Figure 51: *Algorithms* configuration section with a BED *window* parameter of 20.

```

algorithms:
|- id: test
  type: LPCF
  history window: 144
  outlier threshold: 0.8
  event threshold: 0.85
  event timeout: 12
  event window save: 30
| BED: # BED
  window: 6
  outlier probability: 0.5

```

Figure 52: Algorithms configuration section with a BED window parameter of 6.

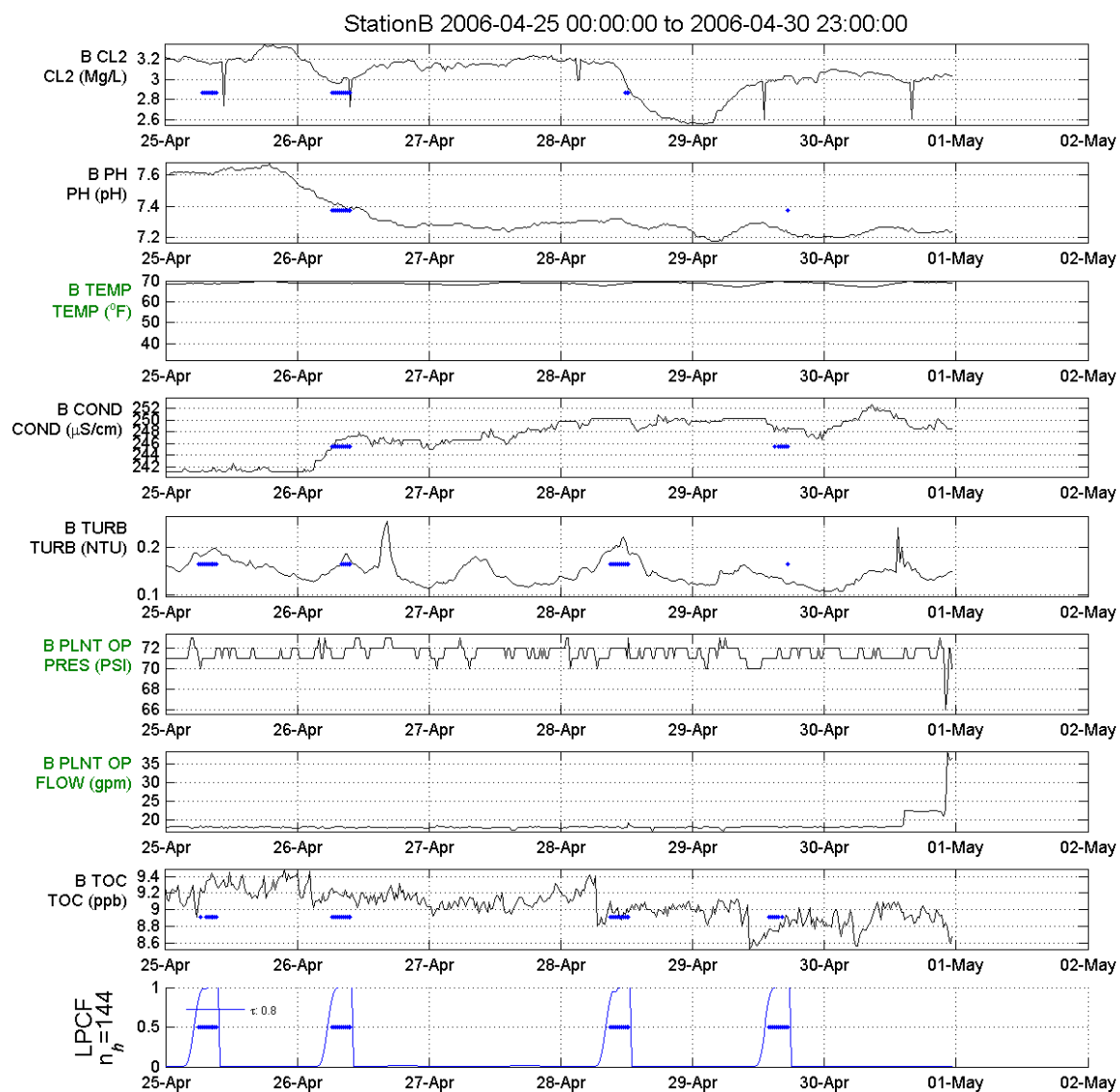


Figure 53: Output graph produced using a BED window parameter of 20.

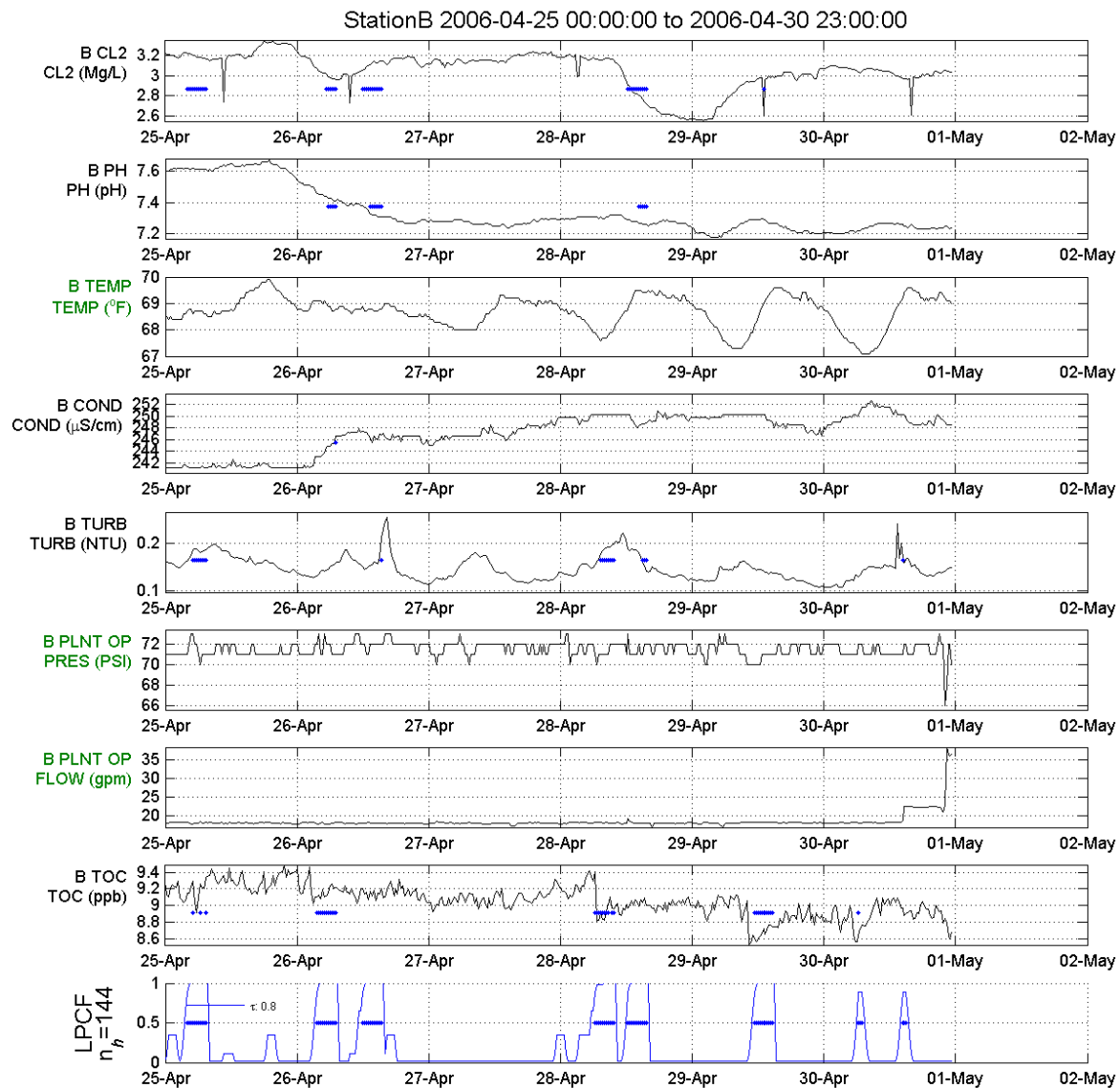


Figure 54: Output graph produced using a BED window parameter of 6.

With a BED window parameter of 20, Figure 53 shows fewer events than for a BED window parameter of 6 as in Figure 54. A shorter BED window parameter leads to faster event determinations, but also results in more events being identified, some of which might be false positives. A shorter BED window parameter leads to faster event determination, because fewer outliers are needed in order to classify a group of outliers as an event.

The tradeoff between faster determination and increased false positives is found in nearly all types of event detection systems. For each monitoring station and situation, the appropriate time to event determination and number of detections must be determined. Changing the values of the BED window, outlier threshold, and event threshold parameters gives the user the flexibility to set the CANARY sensitivity for any monitoring station.

4. Database Driven Input/Output Tutorial

This tutorial provides step-by-step instructions for connecting CANARY to a database. For most online applications of CANARY, a database that contains water quality signals (along with any associated operational and/or alarm signals) will need to be accessible to CANARY. This tutorial assumes that the user has some knowledge and experience in using databases and connecting databases to external programs. The configuration file used in this section, `example_SQLServer_2008_by_cols.yml`, is located in the “Tutorial_Files\Database_Tutorial” directory.

4.1 Obtaining JDBC Driver

- Download the JDBC driver for the user’s specific database. These files enable CANARY to communicate directly with the database, and they are typically available from your database vendor’s website. In Figure 55, the file `sqljdbc_3.0.1301.101_enu` was downloaded and saved to a directory on the user’s computer. This file is the JDBC driver file in a zipped format. This file should be unzipped within this same directory (as shown in Figure 56).

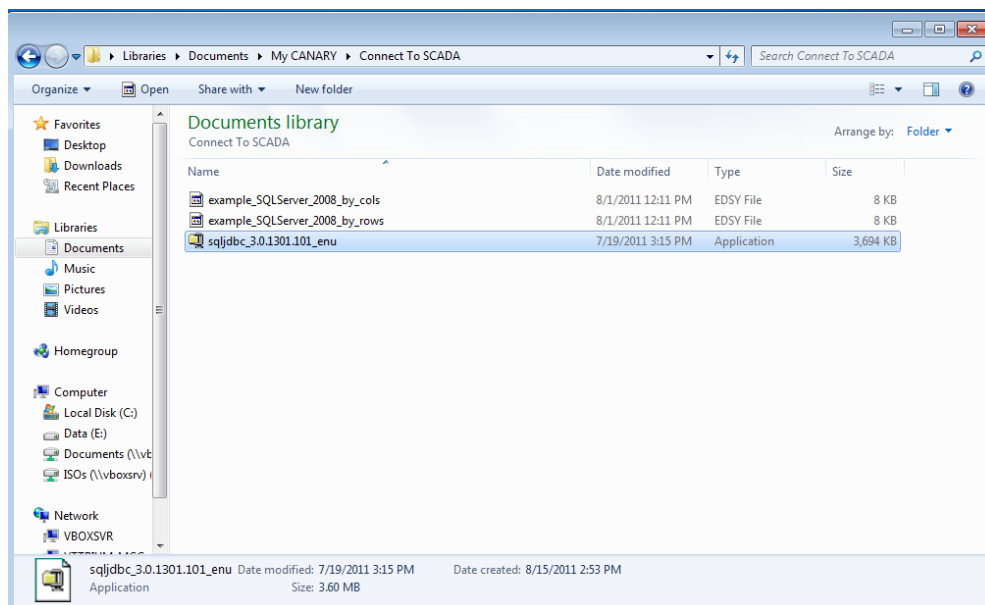


Figure 55: JDBC driver zip file location.

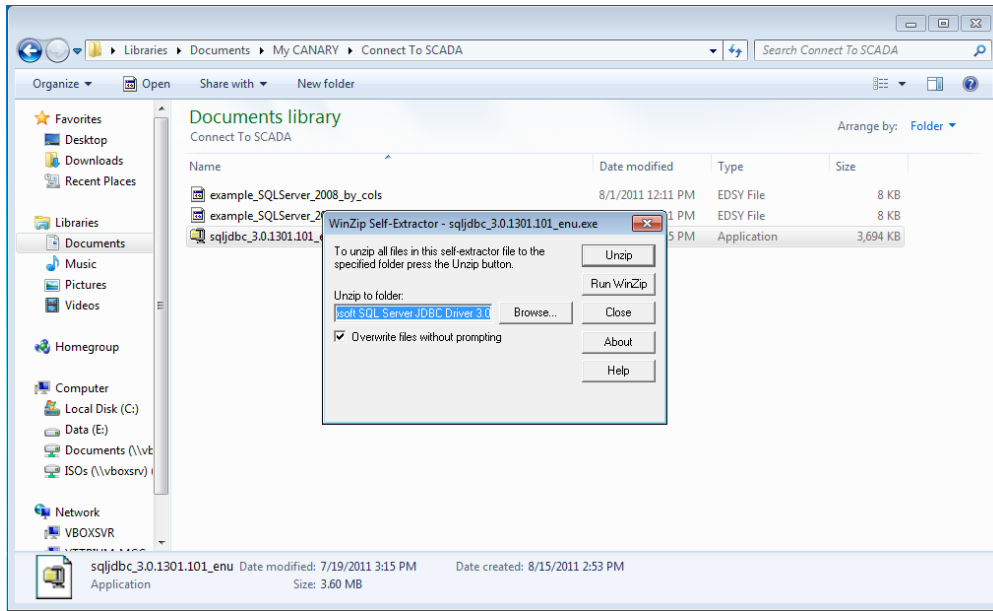


Figure 56: Unzipping JDBC driver file.

- Access the folder with the unzipped files (Figure 57) and then select the desired database JAR file.

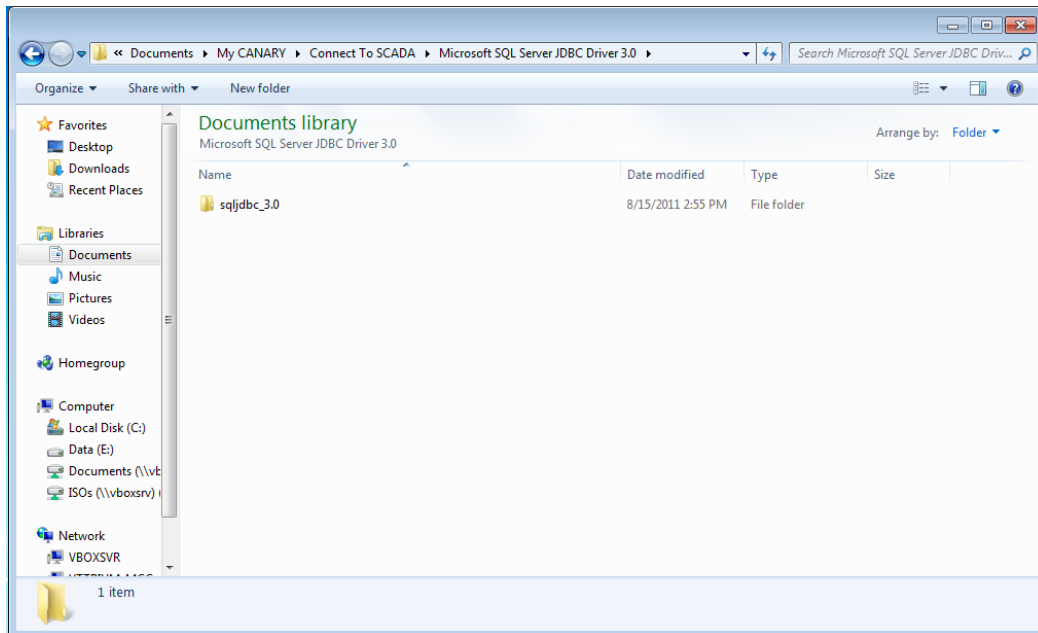


Figure 57: Unzipped file location.

- Find and copy the JAR file(s) located in the unzipped JDBC driver folder (Figure 58). For this example, two JAR files need to be copied.

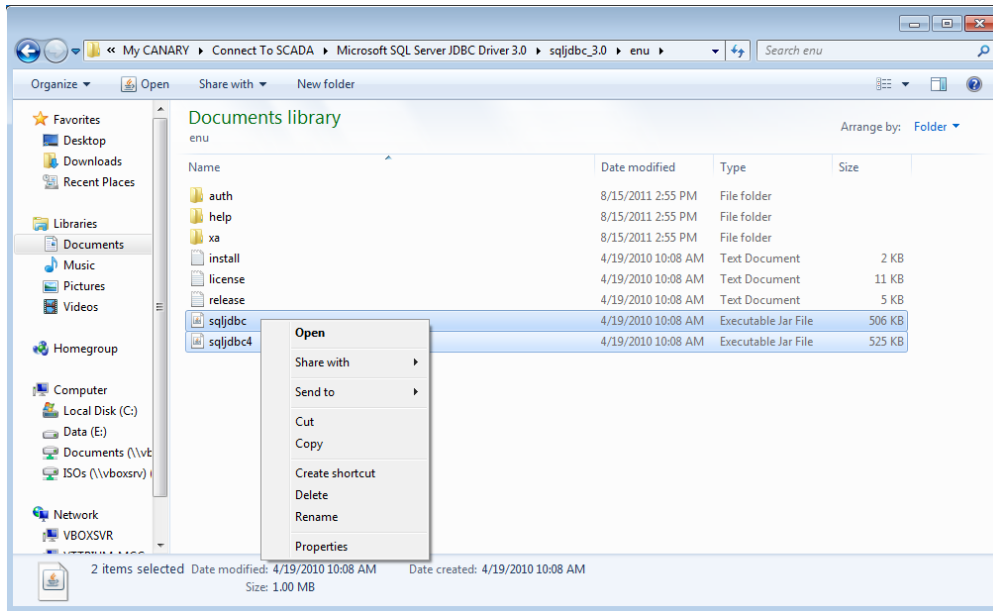


Figure 58: Copying JAR files.

- Paste the JAR file(s) to the \Program Files\CANARY\lib directory (Figure 59).

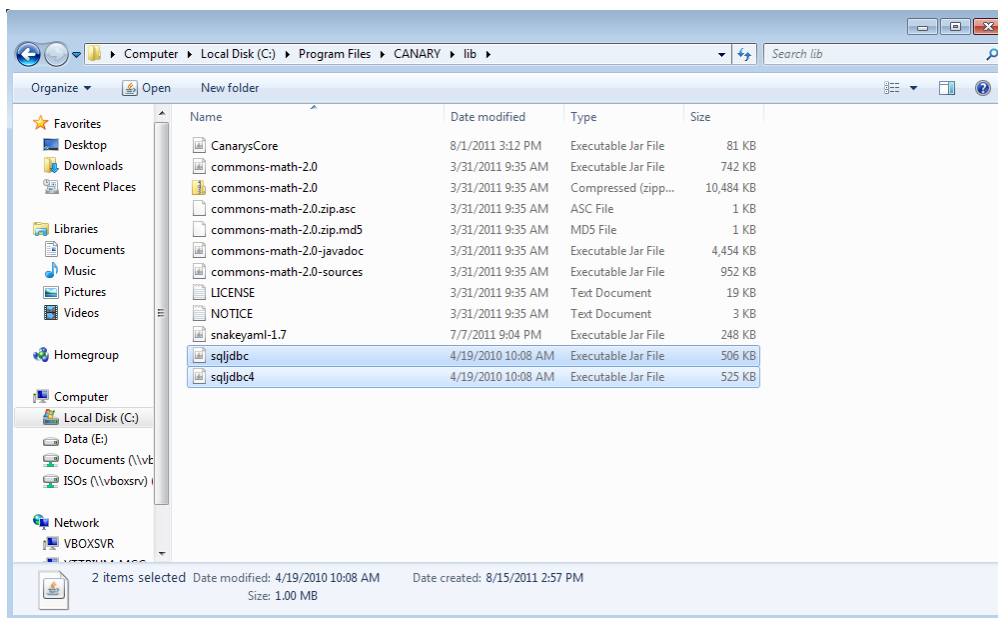


Figure 59: New JAR file location.

4.2 Modifying Configuration File to Use Databases

The configuration file tells CANARY that the input data is found within a database and several options need to be defined. Open the configuration (YML) file in a text editor application.

- Go to the *data sources* section near the top of the file and set the *type* parameter to DB (see the highlighted section in Figure 60).

```

# CANARY Config File - Database driven tutorial example

canary:
  run mode: BATCH
  control type: INTERNAL
  control messenger: null
  driver files: null

# Enter the time step options below
timing options:
  dynamic start-stop: off
  date-time format: mm/dd/yyyy HH:MM:SS
  date-time start: 02/01/2008 00:00:00
  date-time stop: 02/28/2008 23:58:00
  data interval: 00:02:00
  message interval: 00:00:01

# Enter the list of data sources below
data sources:
  - id: database_sqlserver
    type: db
    location: jdbc:sqlserver://127.0.0.1;instanceName=SQLEXPRESS;databaseName=WDS_SIM
    enabled: yes
    timestep options:
      field: "TIME_STEP"
      format: "120"
      conversion function: "CONVERT(datetime,"
    database options:
      time drift: 0
      JDBC2 class name: com.microsoft.sqlserver.jdbc.SQLServerDataSource
      input table: "dbo.test_station_d"
      input format: row based
      output table: "dbo.test_output_d"
    login info:
      prompt for login: no
      username: sa
      password: SYSTEM

```

Figure 60: Data sources configuration section with the *type* parameter set as database.

- Set the *location* parameter to the user's database URL location (Figure 61). The format for this line is jdbc:(database vendor identifier):/(IP address); (database specific options).

```

# CANARY Config File - Database driven tutorial example

canary:
  run mode: BATCH
  control type: INTERNAL
  control messenger: null
  driver files: null

# Enter the time step options below
timing options:
  dynamic start-stop: off
  date-time format: mm/dd/yyyy HH:MM:SS
  date-time start: 02/01/2008 00:00:00
  date-time stop: 02/28/2008 23:58:00
  data interval: 00:02:00
  message interval: 00:00:01

# Enter the list of data sources below
data sources:
  - id: database_sqlserver
    type: db
    location: jdbc:sqlserver://127.0.0.1;instanceName=SQLEXPRESS;databaseName=WDS_SIM
    enabled: yes
    timestep options:
      field: "TIME_STEP"
      format: "120"
      conversion function: "CONVERT(datetime,"
    database options:
      time drift: 0
      JDBC2 class name: com.microsoft.sqlserver.jdbc.SQLServerDataSource
      input table: "dbo.test_station_d"
      input format: row based
      output table: "dbo.test_output_d"
      login info:
        prompt for login: no
        username: sa
        password: SYSTEM

```

Figure 61: Data sources configuration section with the *location* parameter listing URL of database.

- Set the *timestep options* parameters (Figure 62). The *timestep options* parameters are *field*, *format*, and *conversion function*. The *field* parameter is typically “Time_Step” and it must correspond to the name of the time step column within the database table. The *format* parameter is defined by the database and it should match the value in the *date-time format* parameter in the *timing options* section. The *conversion function* parameter is specified by the database and should include any preceding commas or open parenthesis.

```

# CANARY Config File - Database driven tutorial example

canary:
  run mode: BATCH
  control type: INTERNAL
  control messenger: null
  driver files: null

# Enter the time step options below
timing options:
  dynamic start-stop: off
  date-time format: mm/dd/yyyy HH:MM:SS
  date-time start: 02/01/2008 00:00:00
  date-time stop: 02/28/2008 23:58:00
  data interval: 00:02:00
  message interval: 00:00:01

# Enter the list of data sources below
data sources:
  - id: database_sqlserver
    type: db
    location: jdbc:sqlserver://127.0.0.1;instanceName=SQLEXPRESS;databaseName=WDS_SIM
    enabled: yes
    timestep options:
      field: "TIME_STEP"
      format: "120"
      conversion function: "CONVERT(datetime,"
    database options:
      time drift: 0
      JDBC2 class name: com.microsoft.sqlserver.jdbc.SQLServerDataSource
      input table: "dbo.test_station_d"
      input format: row based
      output table: "dbo.test_output_d"
      login info:
        prompt for login: no
        username: sa
        password: SYSTEM

```

Figure 62: Data sources configuration section with the *timestep options* parameters.

- Set the *database options* parameters (Figure 63). The *database options* include many parameters, but the main ones which need to be defined are *time drift*, *JDBC2 class name*, *input table*, and *output table*. The *time drift* parameter is database and computer specific and is the difference in time between the local computer time and the time on the database machine in days. The *JDBC2 class name* parameter is a Java class name referenced in the JDBC database documentation and it will always have DataSource in the last part of the file name. The *input table* and *output table* parameters are database specific and user defined.

```

# CANARY Config File - Database driven tutorial example

canary:
  run mode: BATCH
  control type: INTERNAL
  control messenger: null
  driver files: null

# Enter the time step options below
timing options:
  dynamic start-stop: off
  date-time format: mm/dd/yyyy HH:MM:SS
  date-time start: 02/01/2008 00:00:00
  date-time stop: 02/28/2008 23:58:00
  data interval: 00:02:00
  message interval: 00:00:01

# Enter the list of data sources below
data sources:
  - id: database_sqlserver
    type: db
    location: jdbc:sqlserver://127.0.0.1;instanceName=SQLEXPRESS;databaseName=WDS_SIM
    enabled: yes
    timestep options:
      field: "TIME_STEP"
      format: "120"
      conversion function: "CONVERT(datetime, "
    database options:
      time drift: 0
      JDBC2 class name: com.microsoft.sqlserver.jdbc.SQLServerDataSource
      input table: "dbo.test_station_d"
      input format: row based
      output table: "dbo.test_output_d"
    login info:
      prompt for login: no
      username: sa
      password: SYSTEM

```

Figure 63: Data sources configuration section with the *database options* parameters.

- Under the *database options* parameter, set the *login info* parameters. These parameters are *prompt for login*, *username*, and *password*. If the *prompt for login* parameter is set to yes, then the *username* and *password* parameters are not required and can be removed from the configuration file. If the parameters remain, then the parameter values must be blank. If *prompt for login* is set to no, the user must define the *username* and *password* parameters (Figure 64).

```

# CANARY Config File - Database driven tutorial example

canary:
  run mode: BATCH
  control type: INTERNAL
  control messenger: null
  driver files: null

# Enter the time step options below
timing options:
  dynamic start-stop: off
  date-time format: mm/dd/yyyy HH:MM:SS
  date-time start: 02/01/2008 00:00:00
  date-time stop: 02/28/2008 23:58:00
  data interval: 00:02:00
  message interval: 00:00:01

# Enter the list of data sources below
data sources:
  - id: database_sqlserver
    type: db
    location: jdbc:sqlserver://127.0.0.1;instanceName=SQLEXPRESS;databaseName=WDS_SIM
    enabled: yes
    timestep options:
      field: "TIME_STEP"
      format: "120"
      conversion function: "CONVERT(datetime,"
    database options:
      time drift: 0
      JDBC2 class name: com.microsoft.sqlserver.jdbc.SQLServerDataSource
      input table: "dbo.test_station_d"
      input format: row based
      output table: "dbo.test_output_d"
    login info:
      prompt for login: no
      username: sa
      password: SYSTEM

```

Figure 64: Data sources configuration section with *database options login* parameters.

- Additional optional parameters can be set by the user under the *database options* parameter. These options include *input format* and *output format*. If the *input format* parameter is omitted, then the format is column based (for other options, see Section 5.3 of the CANARY User's Manual (Hart and McKenna 2012)). The example in Figure 65 shows the option for row based input. If the *output format* parameter is omitted, then the default format of the database table is used.

```

# CANARY Config File - Database driven tutorial example

canary:
  run mode: BATCH
  control type: INTERNAL
  control messenger: null
  driver files: null

# Enter the time step options below
timing options:
  dynamic start-stop: off
  date-time format: mm/dd/yyyy HH:MM:SS
  date-time start: 02/01/2008 00:00:00
  date-time stop: 02/28/2008 23:58:00
  data interval: 00:02:00
  message interval: 00:00:01

# Enter the list of data sources below
data sources:
  - id: database_sqlserver
    type: db
    location: jdbc:sqlserver://127.0.0.1;instanceName=SQLEXPRESS;databaseName=WDS_SIM
    enabled: yes
    timestep options:
      field: "TIME_STEP"
      format: "120"
      conversion function: "CONVERT(datetime,"
    database options:
      time drift: 0
      JDBC2 class name: com.microsoft.sqlserver.jdbc.SQLServerDataSource
      input table: "dbo.test_station_d"
      input format: row based
      output table: "dbo.test_output_d"
    login info:
      prompt for login: no
      username: sa
      password: SYSTEM

```

Figure 65: *Data sources* configuration section with the optional *database options* parameter, *input format*.

- To test the database connection, save and run the configuration file in CANARY.

5. Composite Signals Tutorials

The composite signals capability allows users to combine and modify input signals in order to enhance detection. Composite signals are created through simple mathematical operations and other logic statements. They can be combinations of water quality, operational, and/or calibration signals. Composite signals can be used to create new calibration signals or to integrate operational information into the event detection process.

Three tutorials that highlight the development of different composite signals are provided. The files associated with these tutorials are located in the “Tutorial_Files\Composite_Tutorials” directory. The three tutorials are:

- Creation of a calibration signal to suppress alarms for a fixed time period after a calibration event.
- Integration of flow data from a single pump into a composite signal.
- Integration of tank levels and tank outlet water quality into a composite signal.

5.1 Suppressing an Alarm After a Calibration Event

Alarms after a calibration event are a common issue in utilities when there is a significant difference in the water quality values before and after calibrating the sensor(s). Suppressing alarms for a set time period following a calibration event allows the algorithm data windows to be filled with new data prior to re-starting online event detection. A composite signal can be defined to extend the signal calibration time period by a set amount. This tutorial demonstrates this approach.

In this tutorial, the input data is contained in the file, CTFD_caltest_mod.csv, which is located in the “Tutorial_Files\Composite_Tutorials\Composite_Signals_1\Initial” directory. This file contains data from a single monitoring station that collected data every two minutes for six water quality signals: residual chlorine (CL2), specific conductivity (COND), oxidation reduction potential (ORP), pH (PH), temperature (TEMP), and total organic carbon (TOC). A calibration signal is applied to all sensors at the monitoring station in a 0/1 format. For this calibration signal, 0 indicates calibration and 1 indicates normal operations. Note that this logic is the reverse of that typically used in distribution systems.

Figure 66 shows the water quality signal plots as well as the CANARY LPCF detection plot for a single day, January 3, 2011, based on running the configuration file, CTFD_composite_A_mod.yml. The YML file specifies the LPCF algorithm with a *history window* parameter of 1080 time steps (1.5 days) and an *event threshold* parameter of 0.90 in the *algorithms* section. It also specifies the six signals to be included in the analysis in the *monitoring stations* section, as well as the calibration signal RAW_CAL. The calibration alarm is turned on around 10:30 AM on January 3, 2011, and stays on until 11:04 AM as shown by the green bars in Figure 66. The TOC value drops to zero quickly and stays there throughout the rest of the day. Since the calibration signal was used as part of the analysis, the calibration periods are ignored and CANARY does not identify an event during this period.

In contrast, Figure 67 shows the water quality signal plots and detection results for January 6,

2011. In this case, the calibration alarm is on from 9:44 AM to 10:32 AM. The TOC signal takes about 14 minutes before it starts to return to normal values, first bouncing up rapidly to a value near 3 and then stabilizing down to a value near 1. During this transition period, CANARY identifies an event. As the calibration period has ended, this data is not ignored by CANARY.

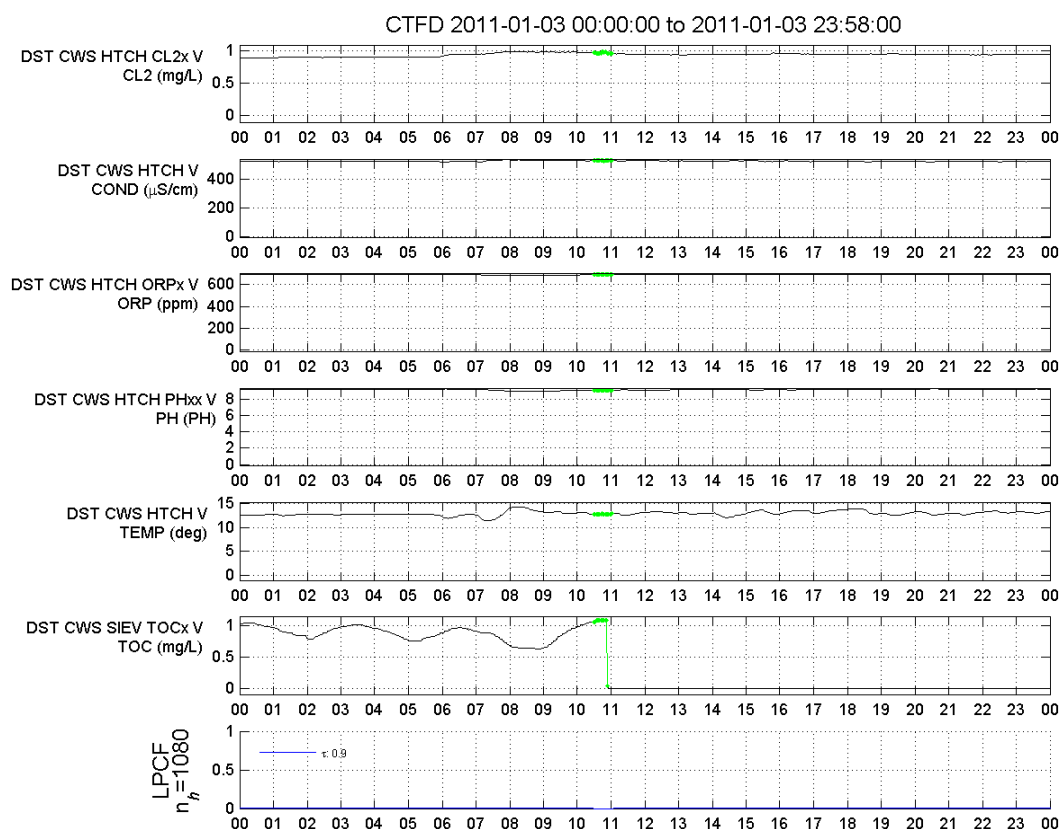


Figure 66: Output graph produced for January 3, 2011.

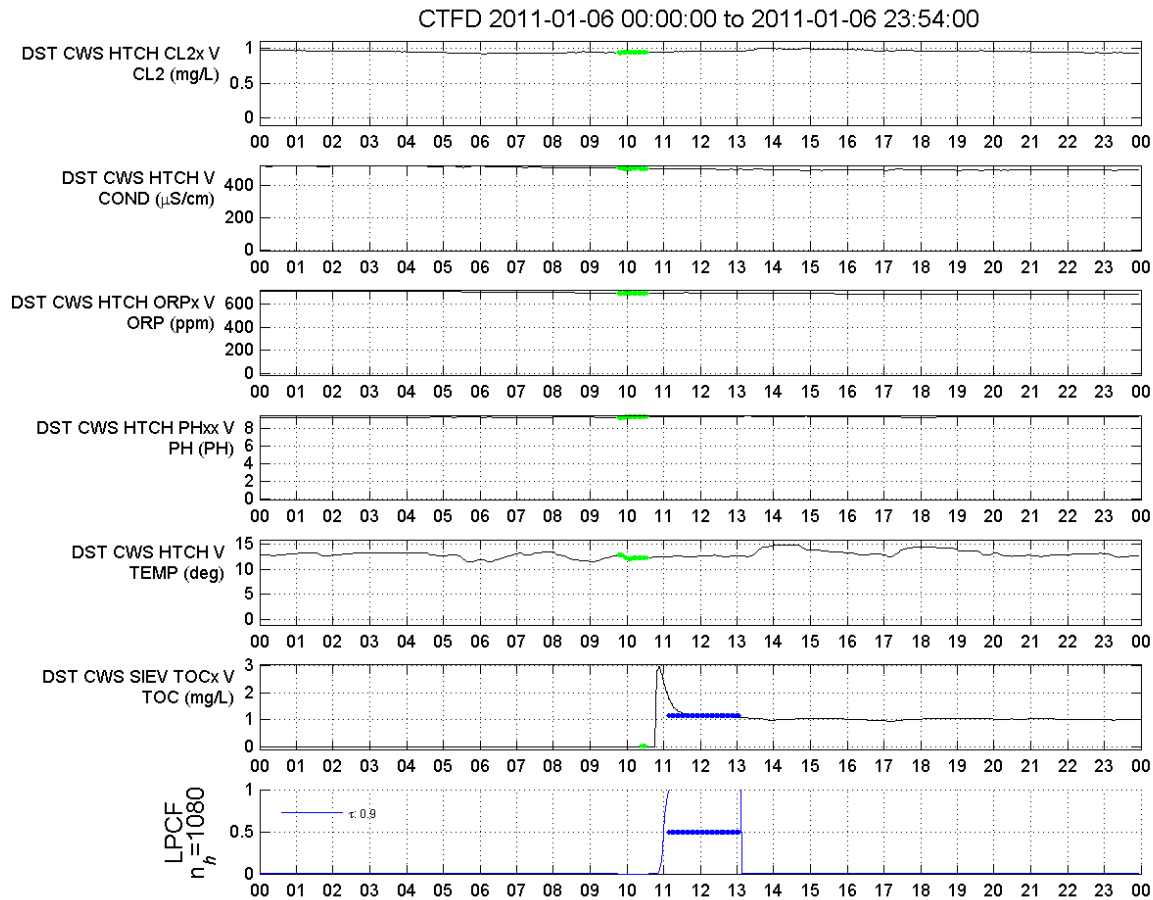


Figure 67: Output plot produced for January 6, 2011.

This event is clearly a byproduct of the calibration period and should not be identified as a real event. Either users can use their expert judgement to determine that this is a false alarm and ignore it, or they can attempt to use another feature of CANARY to suppress these types of events that follow calibration periods. In what follows, the second option is demonstrated.

In order to suppress events identified following calibration periods, a new composite signal is created by extending the original calibration signal, RAW_CAL, by thirty time steps. A portion of the *signals* section of the YML file is shown in Figure 68 where the calibration signal is defined.

```
- id: RAW_CAL
  SCADA tag: DST_CWS_CTFD_MON_OSxx_CMD
  evaluation type: cal
  parameter type: Raw CAL
  ignore changes: none
  alarm options:
    value when active: 0
```

Figure 68: *Signals* configuration section for original calibration signal, RAW_CAL.

A new composite signal is created in the *signals* section of the YAML file to reverse the calibration flags, so that 0 indicates normal operations and 1 indicates calibration (Figure 69). At each time step, the composite rule subtracts 1.0 from the RAW_CAL signal and then takes the absolute value of the resulting number. Note that the signal *evaluation type* parameter is set to OP for two reasons. First, so that the signal can be graphed and second, so a new composite signal can be defined as the calibration signal. The new YAML file, CTFD_caltest_mod.csv, is found in the “Tutorial_Files\Composite_Tutorials\Composite_Signals_1\Flip_CAL” directory.

```
- id: FLIP_CAL
  SCADA tag: FLIP_CAL
  evaluation type: op
  parameter type: Flipped Calib
  ignore changes: none
  data options: # DATA
    precision: 0.5
    units: ''
    valid range: [0.0, 1.0]
    set points: [-.inf, .inf]
  composite rules: |
    @RAW_CAL[0]
    {1}
    -
    abs
```

Figure 69: Signals configuration section for composite calibration signal.

```

# Enter the list of event detection algorithms below
algorithms:
- id: CTFD_ALG
  type: LPCF
  history window: 1080
  outlier threshold: 0.9
  event threshold: 0.95
  event timeout: 60
  event window save: 30
  BED:
    window: 15
    outlier probability: 0.5

# Enter the list of monitoring stations below
monitoring stations:
- id: CTFD
  station id number:
  station tag name: CTFD
  location id number: -1
  enabled: yes
  inputs:
    - id: files_in
  outputs:
  signals:
    - id: RAW_CAL
    - id: FLIP_CAL
    - id: CTFD_H2OxHTCH_CL2x_V
    - id: CTFD_H2OxHTCH_COND_V
    - id: CTFD_H2OxHTCH_ORPx_V
    - id: CTFD_H2OxHTCH_PHxx_V
    - id: CTFD_H2OxHTCH_TEMP_V
    - id: CTFD_H2OxSIEV_TOCx_V
  algorithms:
    - id: CTFD_ALG

```

Figure 70: Algorithms and monitoring stations configuration sections.

The RAW_CAL and FLIP_CAL signals are plotted in Figure 71 with the rest of the data by switching the signal *evaluation type* parameter from CAL to OP. Since these two signals are now operational (OP) signals, they have green y-axis labels to distinguish them from the water quality (WQ) signals. Note that these signals must be included in the *monitoring stations* section of the YML file in order to appear on the graphs, see Figure 70. The newly created composite signal is working correctly, since the FLIP_CAL signal is a mirror image of the RAW_CAL signal.

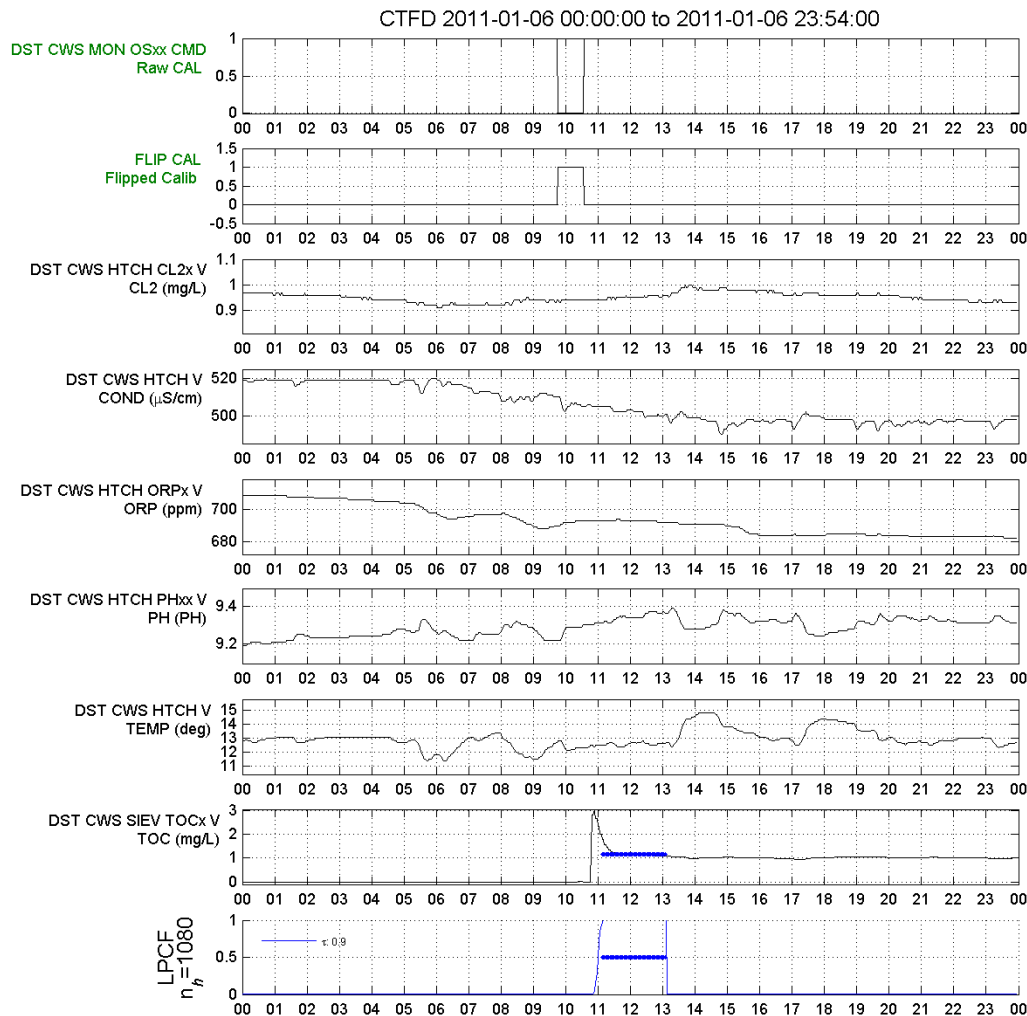


Figure 71: Output graph produced with composite signal.

To suppress alarms for a user-specified time period after calibration, a new composite signal that uses the FLIP_CAL signal and stays as a 1 for 30 time steps beyond the most recent calibration event can be used. The new YAML file is found in the “Tutorial_Files\Composite_Tutorials\Composite_Signals_1\Suppress” directory. Because of a constraint that limits the total number of parenthetical statements within a composite signal to 16 (32 pairs), the composite signal definition needs to be broken down into smaller parts to cover all 30 time steps. The first composite signal covers the first 10 time steps, the second covers the second 10 time steps, and the third covers the third 10 time steps. In Figure 72, the first of three composite signals that provide the basis of alarm suppression over 30 consecutive time steps is defined.

```

- id: CAL_TIME_OUT_CTFD_A
  SCADA tag: CAL_TIME_OUT_CTFD_A
  evaluation type: op
  parameter type: Calibration Time Out
  ignore changes: none
  data options: # DATA
    precision: 1
    units: ''
    valid range: [-.inf, .inf]
    set points: [-.inf, .inf]
  composite rules: |
    @FLIP_CAL[0]
    @FLIP_CAL[1]
    -
    abs
    @FLIP_CAL[0]
    @FLIP_CAL[2]
    -
    abs
    max
    @FLIP_CAL[0]
    @FLIP_CAL[3]
    -
    abs
    max
    @FLIP_CAL[0]
    @FLIP_CAL[4]
    -
    abs
    max
    @FLIP_CAL[0]
    @FLIP_CAL[5]
    -
    abs
    max
    @FLIP_CAL[0]
    @FLIP_CAL[6]
    -
    abs
    max
    @FLIP_CAL[0]
    @FLIP_CAL[7]
    -
    abs
    max
    @FLIP_CAL[0]
    @FLIP_CAL[8]
    -
    abs
    max
    @FLIP_CAL[0]

```

Figure 72: Signals configuration section for CAL_TIME_OUT_CTFD_A.

Composite signals use a memory stack and only two numbers can be on the stack at any one time. In plain English, the logic above says:

- Subtract the previous value of FLIP_CAL (@FLIP_CAL[1]) from the current value of

FLIP_CAL (@FLIP_CAL[0]) and take the absolute value of the result. One value is on the stack.

- Subtract the second previous value of FLIP_CAL (@FLIP_CAL[2]) from the current value of FLIP_CAL and take the absolute value of the result. Two values are on the stack.
- Take the maximum of the two values on the stack. This leaves a single value, the maximum, on the stack.
- Subtract the third previous value of FLIP_CAL (@FLIP_CAL[3]) from the current value of FLIP_CAL and take the absolute value of the result. Two values are again on the stack.
- Take the maximum of the two values on the stack. This leaves a single value, the maximum, on the stack.
- Continue the same steps through the 10th previous value (@FLIP_CAL[10]). A single, final value on the stack is the maximum across all 10 comparisons.

The composite signal definition could also be written as the following equation:

$$\max_{i=1 \dots 10} \{ |FLIP_CAL(0) - FLIP_CAL(i)| \}$$

Two additional composite signals are created using the same exact commands given in Figure 72 except the numbers in the shift are 11-20 for the composite signal CAL_TIME_OUT_CTFD_B and 21-30 for the composite signal CAL_TIME_OUT_CTFD_C. These three composite signals track the maximum difference between the value of FLIP_CAL at the current time step and the value of FLIP_CAL at time steps ranging from 1 to 30 time steps prior to the current time step. The final composite signal, FINAL_CAL_CTFD, combines and retains the maximum value of any of these three, 10 time-step long composite signals (Figure 73). Note that this is the only signal in the YML file with an *evaluation type* parameter value set to CAL, and that this signal was included in the *monitoring stations* section of the YML file in order to be included in the detection analysis.

```
- id: FINAL_CAL_CTFD
  SCADA tag: FINAL_CAL_CTFD
  evaluation type: cal
  parameter type: Final Calibration
  ignore changes: none
  alarm options: # ALARM
    value when active: 1
  composite rules: |
    @CAL_TIME_OUT_CTFD_A[0]
    @CAL_TIME_OUT_CTFD_B[0]
    max
    @CAL_TIME_OUT_CTFD_C[0]
    max
```

Figure 73: Signals configuration section for FINAL_CAL_CTFD.

The results from running the configuration file with the new calibration signal are shown in Figure 74, in which periods of alarm suppression due to calibration are shown in green. By comparing Figure 71 and Figure 74, a few differences can be seen. The previously detected event

after the calibration period due to TOC suddenly increasing just before 10:00 AM is no longer detected, since the event occurs during a period of alarm suppression extending thirty time steps after calibration.

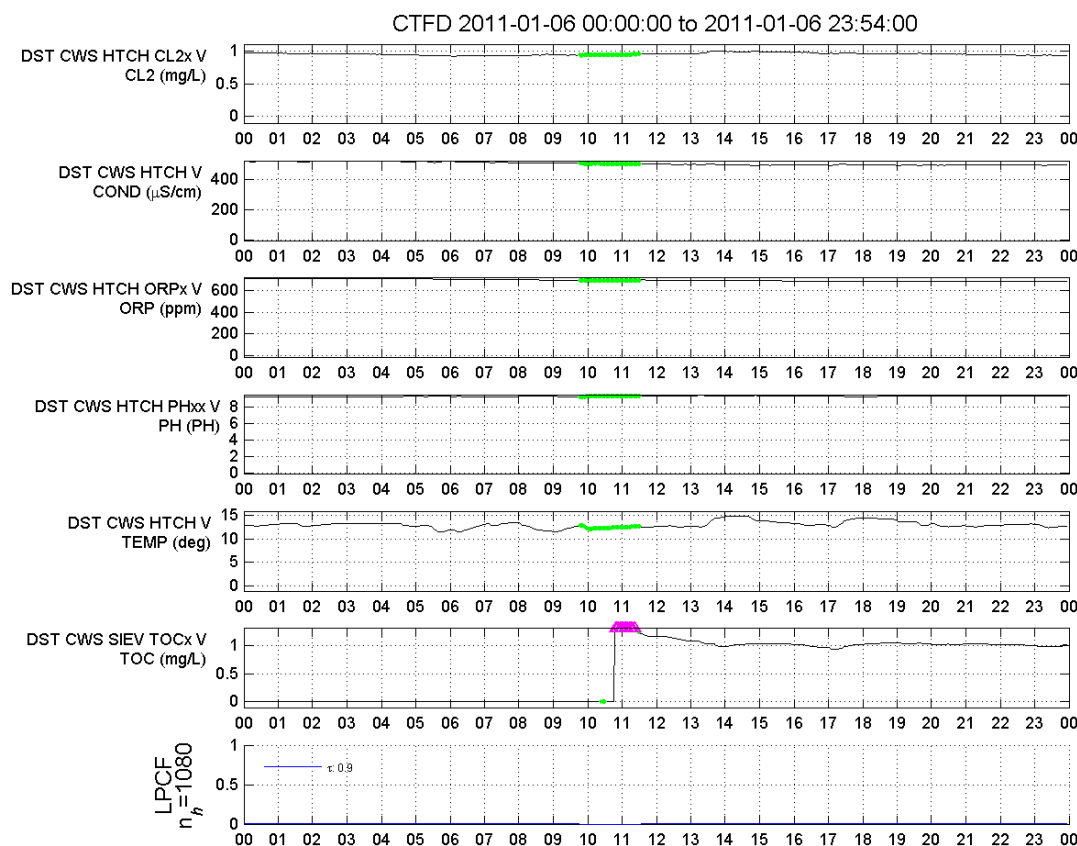


Figure 74: Output graph produced with alarm suppression composite signal enabled.

5.2 Integrating Pump Flow Operational Data into Event Detection

This tutorial demonstrates the creation of composite signals that integrate available operational data into the event detection process. Often water quality is monitored at the same locations that control operations in the distribution network (e.g., a pump station, tank, or valve). Operational alterations could cause changes in the water quality at the co-located monitoring station. If the changes in the operational control are recorded and available through the SCADA system, then they can be integrated into the event detection process.

The files associated with this tutorial are located in the “Tutorial_Files\Composite_Tutorials\Composite_Signals_2” directory. The data for this tutorial are found in the StationD_train.csv file which contains data from a four-month period in 2008. This monitoring station, Station D, recorded a number of primary water quality parameters every two minutes including: residual chlorine (CL2), specific conductivity (COND), pH (PH), temperature (TEMP), turbidity (TURB), and total organic carbon (TOC). The SCADA system at this site provides a calibration signal in a 0/1 format to indicate calibration time periods. The calibration signal applies to all sensors at the monitoring station.

This configuration file for this tutorial is called, StationD_Initial.yml. The *monitoring stations* section of the configuration file is shown in Figure 75 with seven signals included and one algorithm, B1. Algorithm B1 is of type LPCF with a *history window* of 1080 time steps (1.5 days) and an *event threshold* of 0.995.

```
monitoring stations:
|- id: StationD
  station id number:
  station tag name: StationD
  location id number: -1
  enabled: yes
| inputs:
  - id: stationd_in
| outputs:
| signals:
  - id: TEST_CAL
  - id: TEST_CL
  - id: TEST_PH
  - id: TEST_TEMP
  - id: TEST_COND
  - id: TEST_TURB
  - id: TEST_TOC
| algorithms:
  - id: B1
```

Figure 75: Monitoring stations configuration section.

The results for the second week of the analysis (1/23-1/29/2008) are shown in Figure 76. The bright green signal values (green boxes in Figure 76) in the afternoon of January 29th indicate a short calibration event at this station as identified by the TEST_CAL signal, which defines time periods of manual calibration. CANARY ignores water quality signals from this station during the indicated period of calibration.

During this two-week period, three events are identified based on changes in the CL2 and PH data. These alarms are considered to be false positives by the water utility as they are relatively small changes, fall within the normal range for these parameters, and occur on a regular basis throughout the four month period. It is hypothesized that the changes in the water quality values are caused by routine operational changes at this monitoring station.

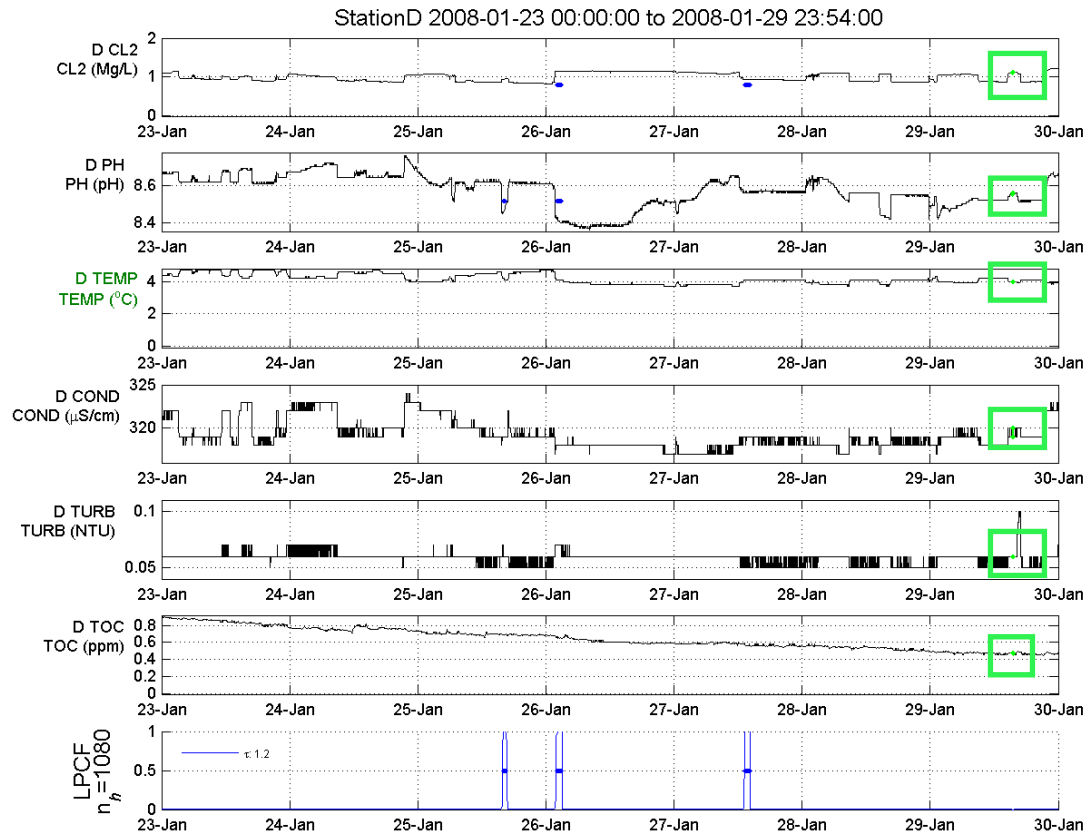


Figure 76: Output graph produced of Station D data with a calibration period and three alarms.

This monitoring station has a number of additional signals, such as, several pump signals and other water quality signals from mains that feed into this station. The *monitoring stations* section of the configuration file, StationD_Step1.yml, was altered to allow for the display of the three pump signals (Figure 77). By plotting these signals, a relationship between water quality signals, operational signals, and the identified events can be determined.

```

monitoring stations:
|- id: StationD
  station id number:
  station tag name: StationD
  location id number: -1
  enabled: yes
| inputs:
  - id: stationd_in
  outputs:
| signals:
  - id: TEST_CAL
  - id: TEST_CL
  - id: TEST_PH
  - id: TEST_TEMP
  - id: TEST_COND
  - id: TEST_TURB
  - id: TEST_TOC
  - id: TEST_PUMP_CONN_FLOW
  - id: TEST_PUMP1_FLOW
  - id: TEST_PUMP2_FLOW
| algorithms:
  - id: B1

```

Figure 77: Monitoring stations configuration section with three pump signals for Station D.

Figure 78 shows the same data as Figure 76 with the addition of the three pump signals. The operational signals are shown with green labels on the y-axis. Examination of Figure 78 shows that the water quality events might be linked to one or more of the changes in pumping rates. Although it is not exactly clear which pump could be responsible for the water quality events, it appears that changes in the pump on inlet 2, the D PUMP IN2 FLOW signal in Figure 78, might be causing the events. The timing of the identified events appears to correspond to large changes in this signal. In order to test this hypothesis, composite signals are added to the configuration file that will tell CANARY to ignore data during periods of large changes in the D PUMP IN2 FLOW signal.

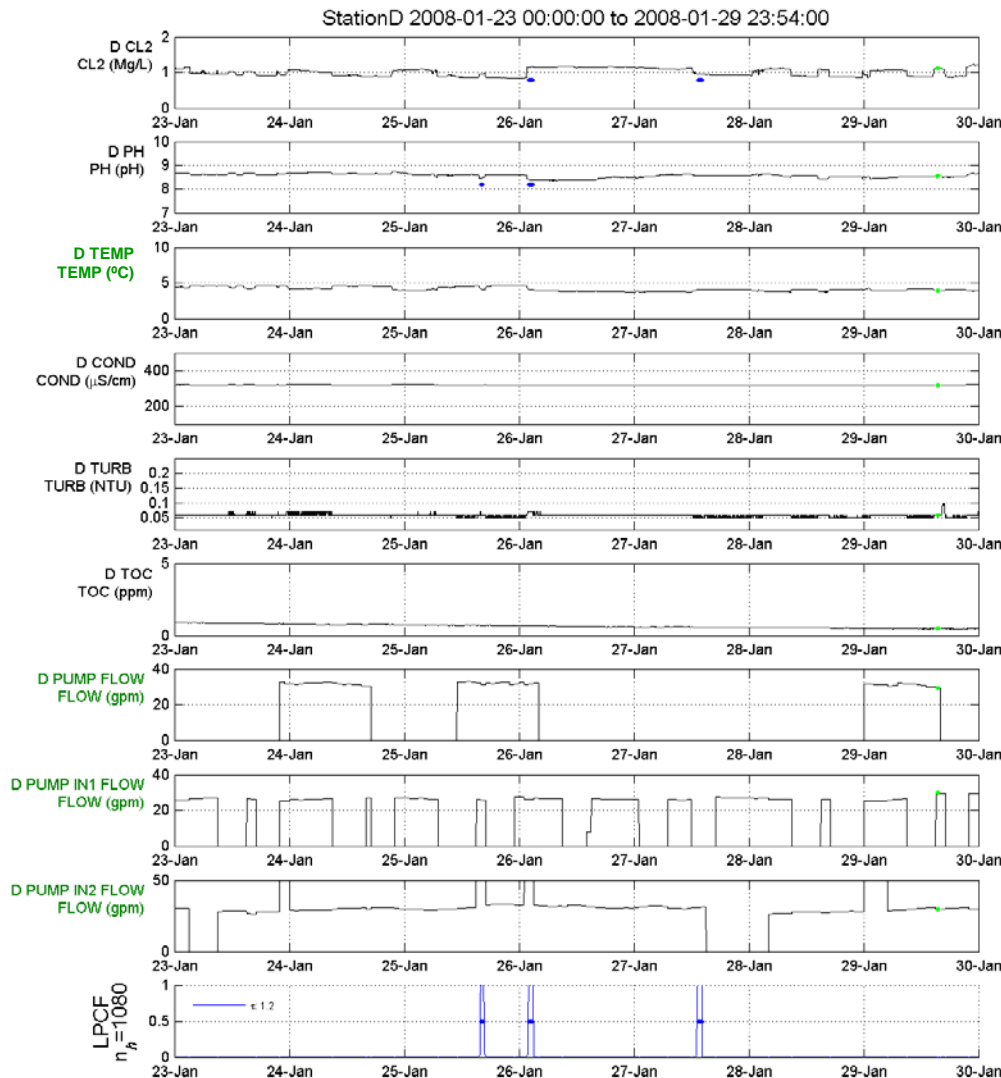


Figure 78: Output graph produced with the additional three operational signals.

In the StationD_Step2.yml file in the “Tutorial_Files\Composite_Tutorials\Composite_Signals_2\Step 2” directory, a new composite signal is created called CMB_PUMP_FLOW. This new signal measures changes in the flow rate over the last two time steps. Specifically, it is the maximum value of the absolute differences in the pump flow (TEST_PUMP2_FLOW) between the current time step [0] and the previous time step [1] or the current time step [0] and two time steps prior [2]. This composite signal is defined as an operational signal (OP).

Another composite signal is created called CAL_PUMP_FLOW and it compares the output of CMB_PUMP_FLOW (e.g., the change in the flow rate) at the current time to the constant value of 5 gpm. The greater than or equal operator (ge) returns a 1 if the value of CMB_PUMP_FLOW[0] is greater than or equal to 5 gpm and 0 otherwise. This signal is defined as an operational signal so that it is visible on CANARY’s output graph and to verify that is

working correctly. These additions to the file are shown in Figure 79.

```
- id: CMB_PUMP_FLOW
  SCADA tag: CMB_PUMP_FLOW
  evaluation type: op
  parameter type: Max_4Min_Change
  ignore changes: none
  data options: # DATA
    precision: 1
    units: ''
    valid range: [-.inf, .inf]
    set points: [-.inf, .inf]
  composite rules: |
    @TEST_PUMP2_FLOW[0]
    @TEST_PUMP2_FLOW[1]
    -
    abs
    @TEST_PUMP2_FLOW[0]
    @TEST_PUMP2_FLOW[2]
    -
    abs
    max

- id: CAL_PUMP_FLOW
  SCADA tag: CAL_PUMP_FLOW
  evaluation type: op
  parameter type: Pump2_Change
  ignore changes: none
  data options: # DATA
    precision: 0.0001
    units: ''
    valid range: [-.inf, .inf]
    set points: [-.inf, .inf]
  composite rules: |
    @CMB_PUMP_FLOW[0]
    {5}
    ge
```

Figure 79: Signals configuration section with added composite signals.

Figure 80 shows the three operational signals and the resulting probability of an event after running the YML file in CANARY. The top signal, D PUMP IN2 FLOW, is the original operational signal from the SCADA system. The second signal is the CMB_PUMP_FLOW composite signal, which calculates the maximum change in the D PUMP IN2 FLOW over the past two time steps (4 minutes). The last signal is the CAL_PUMP_FLOW composite signal, which has a value of 1 when changes in CMB_PUMP_FLOW exceed 5 gpm. From examination of Figure 80, all signals appear to be working as designed. In addition, the CMB_PUMP_FLOW signal value is typically near zero, but can range up to 30 gpm when there are sudden changes in the D PUMP IN2 FLOW signal.

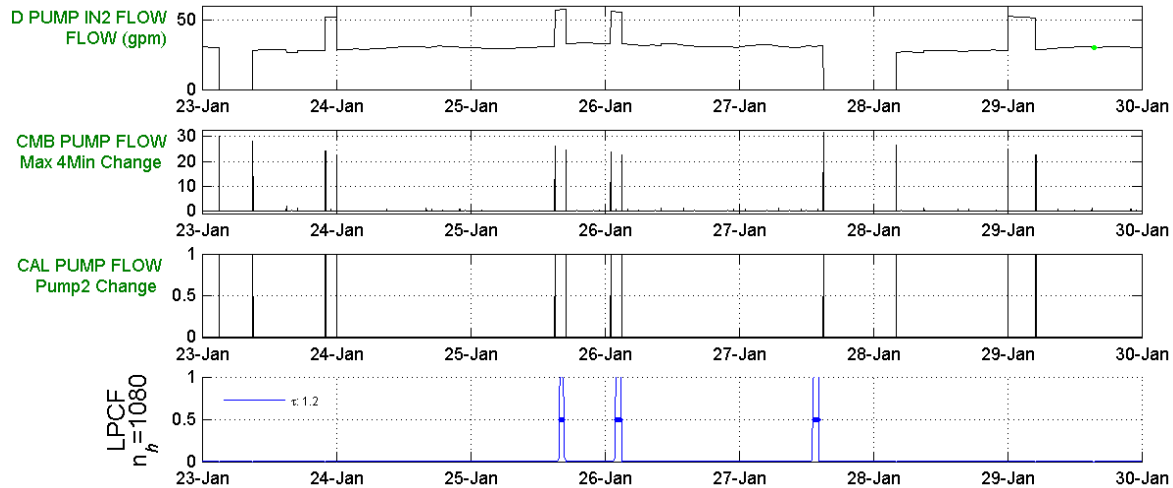


Figure 80: Output graph produced with three operation signals and event probability.

If it seems reasonable to ignore data during periods of large changes in the operational signal from one of the pumps, then another composite signal can be defined. This new signal, CAL_ALL, is defined to be a calibration signal which will trick CANARY into ignoring data at certain time periods. The new signal is created by modifying the existing calibration signal, TEST_CAL, whenever the composite signal CAL_PUMP_FLOW is non-zero. Only one calibration signal can be assigned per station, but this calibration signal can be a composite of other signals. The original TEST_CAL signal is combined with the newly created CAL_PUMP_FLOW into a final calibration signal, CAL_ALL.

Since CAL_ALL is going to be the new calibration signal for Station D, the original TEST_CAL signal's *evaluation type* parameter needs to be changed from CAL to OP. The *alarm options* parameters of the original TEST_CAL calibration signal do not apply for an operational signal and should not be used. These changes are made to the StationD_Step3.yml configuration file "Tutorial_Files\Composite_Tutorials\Composite_Signals_2\Step 3" directory. Figure 81 shows the *signals* section of the YML file in which *alarm options* parameters have been commented; alternatively, these three lines could be deleted from the configuration file.

```
- id: TEST_CAL
  SCADA tag: D_CAL
  evaluation type: op
  parameter type: CAL
  ignore changes: none
  data options: # DATA
    precision: 1
    units: ''
    valid range: [-.inf, .inf]
    set points: [-.inf, .inf]
  # alarm options: # ALARM
  #   value when active: 0
  #   scope: # entire station
```

Figure 81: Signals configuration section with changed calibration signal.

The CAL_ALL signal definition is shown in Figure 82 and it has the required *alarm options* parameters. In addition, the CAL_ALL signal needs to be added to the monitoring station section of the configuration file. The composite rule steps for this signal are:

- Subtract one from the current value of the TEST_CAL signal (@TEST_CAL[0]) and take the absolute value of the result. This converts the original TEST_CAL signal to 1 when active and 0 in the background.
- Add the result to the current value of the CAL_PUMP_FLOW signal (@CAL_PUMP_FLOW[0]) and compare to zero. If the final result is true (sum > 0), the signal value for this time step will be 1; otherwise it will be 0.

```
|- id: CAL_ALL
  SCADA tag: CAL_ALL
  evaluation type: cal
  parameter type: Aggregate_CAL
  ignore changes: none
| alarm options: # ALARM
  value when active: 1
| composite rules: |
  @TEST_CAL[0]
  (1)
  -
  abs
  @CAL_PUMP_FLOW[0]
  +
  (0)
  gt
```

Figure 82: Signals configuration section with the composite calibration signal.

Figure 83 shows the plots for the two inputs to the CAL_ALL signal and the probability of event after running CANARY with this new composite calibration signal. The calibration signal used in this analysis, CAL_ALL, is not plotted, but the calibration events it identifies are marked in green. The three water quality events are still identified by CANARY; thus, either these events are not related to the pump activity, or the timing of the events is slightly out of step with the changes in the water quality that caused CANARY to alarm. An additional look at the data in Figure 78 shows that events are occurring roughly one hour (30 time steps) after the change in status of D PUMP IN2 FLOW. This information can be included into the composite signal by increasing the length of the calibration signal.

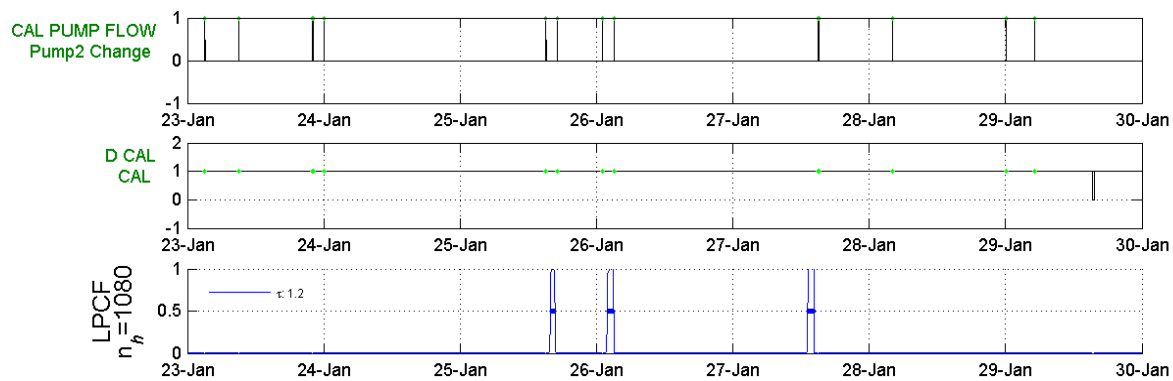


Figure 83: Output plots produced using composite calibration signal.

These changes to the CMB_PUMP_FLOW composite signal were made in the configuration file, StationD_Step4.yml, found in the “Tutorial_Files\Composite_Tutorials\Composite_Signals_2\Step 4” directory. Figure 84 displays the modified CMB_PUMP_FLOW signal in the YAML file. This change looks for differences between the current time step and 30 through 34 time steps (60 to 68 minutes) prior to the current time step. The results of this change are shown in Figure 85. This change in the delay of the calibration signal to an hour or more after the change in the pump status generates a calibration signal during the time of the first two events and so they are no longer identified as events. However, the third event occurs before a calibration period and thus is not clearly associated with the changes in the pump operations.

This tutorial accomplishes the goal of using operational data to decrease false positive detection of water quality events. The disadvantage of using composite signals to generate calibration signals is that additional time steps are classified as being in calibration and all water quality changes during those periods, not just those caused by the operational change, will go unnoticed. However, by limiting the period of calibration within the composite signal, the number of time steps considered to be in calibration can be minimized.

```

- id: CMB_PUMP_FLOW
  SCADA tag: CMB_PUMP_FLOW
  evaluation type: op
  parameter type: Max_4Min_Change
  ignore changes: none
  data options: # DATA
    precision: 1
    units: ''
    valid range: [-.inf, .inf]
    set points: [-.inf, .inf]
  composite rules: |
    @TEST_PUMP2_FLOW[0]
    @TEST_PUMP2_FLOW[30]
    -
    abs
    @TEST_PUMP2_FLOW[0]
    @TEST_PUMP2_FLOW[31]
    -
    abs
    @TEST_PUMP2_FLOW[0]
    @TEST_PUMP2_FLOW[32]
    -
    abs
    @TEST_PUMP2_FLOW[0]
    @TEST_PUMP2_FLOW[33]
    -
    abs
    @TEST_PUMP2_FLOW[0]
    @TEST_PUMP2_FLOW[34]
    -
    abs
    max

```

Figure 84: Signals configuration section with the modified composite signal.

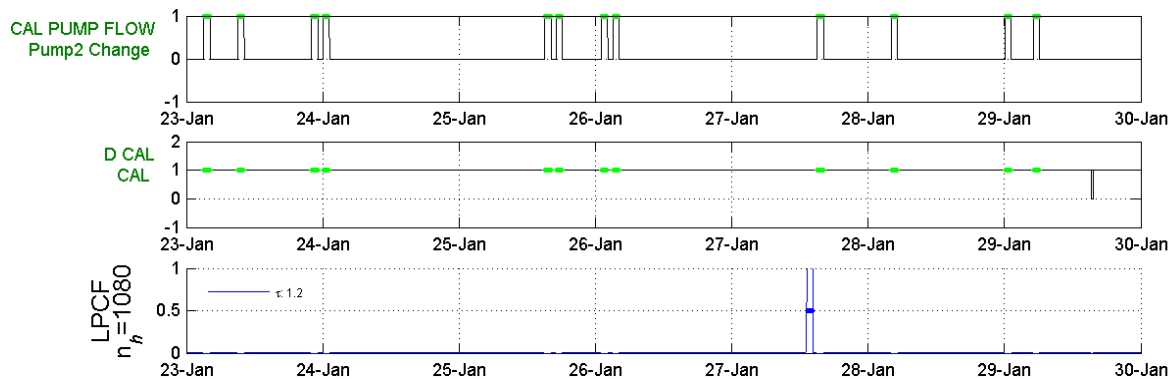


Figure 85: Output plots produced using modified composite signal.

Figure 86 shows the same data as Figure 85, but at a finer time resolution of only one day (January 25th). The periods of calibration are just over an hour long due to the additional delay of 60-68 minutes. A final change is made to the CMB_PUMP_FLOW to examine the differences in the pumping rates between 25 time steps prior to the current time and the same 30-34 time steps

prior to the current time as used previously (StationD_Step5.yml). These changes are shown in Figure 87 and the associated files are found in the directory, “Tutorial_Files\Composite_Tutorials\Composite_Signals_2\Step 5”.

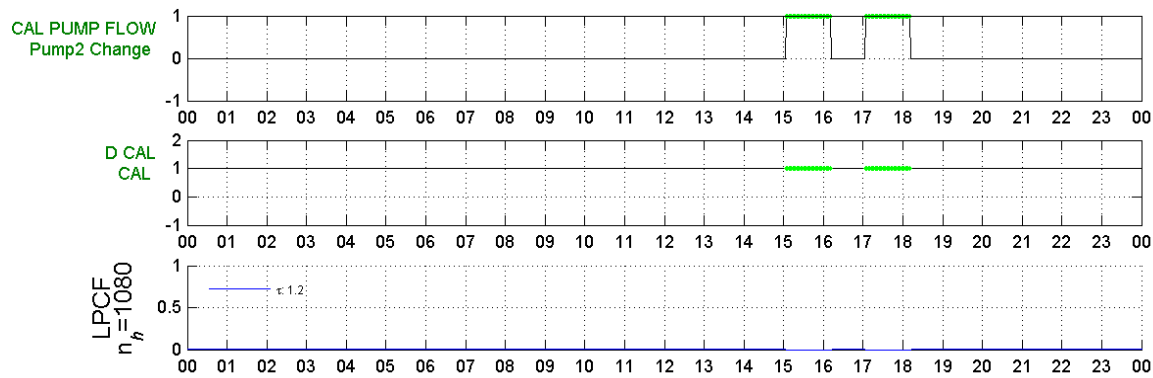


Figure 86: Output plots produced using modified composite signal for January 25th only.

```
- id: CMB_PUMP_FLOW
  SCADA tag: CMB_PUMP_FLOW
  evaluation type: op
  parameter type: Max_4Min_Change
  ignore changes: none
  data options: # DATA
    precision: 1
    units: ''
    valid range: [-.inf, .inf]
    set points: [-.inf, .inf]
  composite rules: |
    @TEST_PUMP2_FLOW[25]
    @TEST_PUMP2_FLOW[30]
  -
    abs
    @TEST_PUMP2_FLOW[25]
    @TEST_PUMP2_FLOW[31]
  -
    abs
    @TEST_PUMP2_FLOW[25]
    @TEST_PUMP2_FLOW[32]
  -
    abs
    @TEST_PUMP2_FLOW[25]
    @TEST_PUMP2_FLOW[33]
  -
    abs
    @TEST_PUMP2_FLOW[25]
    @TEST_PUMP2_FLOW[34]
  -
    abs
    max
```

Figure 87: Signals configuration section with final composite signal.

The results from this final composite signal modification are shown for the entire week and January 25th in Figure 88 and Figure 89, respectively. The total number of time steps within the total calibration period is cut from 34 down to 9. The change in the size of the calibration period is most clearly seen in Figure 89. The timing of the delay might not be quite optimal, as there are still some non-zero event probabilities in the lowest graphs, but these levels are below the event threshold needed to define a water quality event.

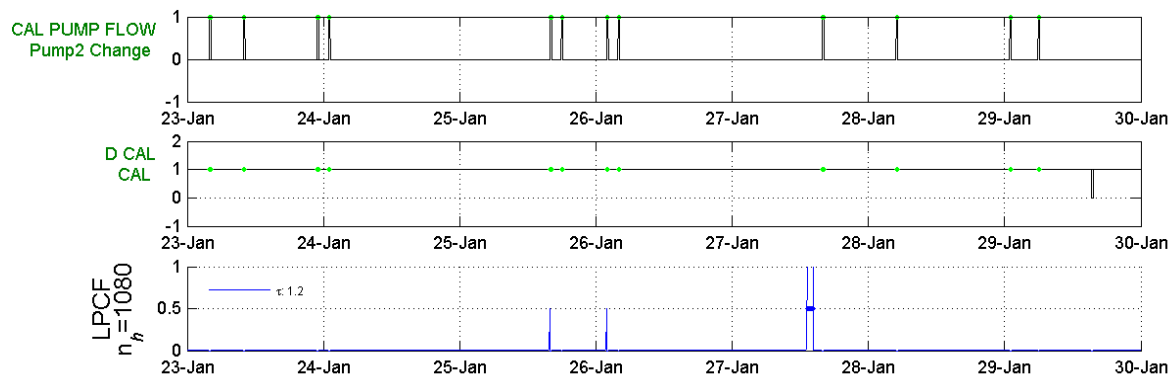


Figure 88: Output plots produced using final composite signal for entire week.

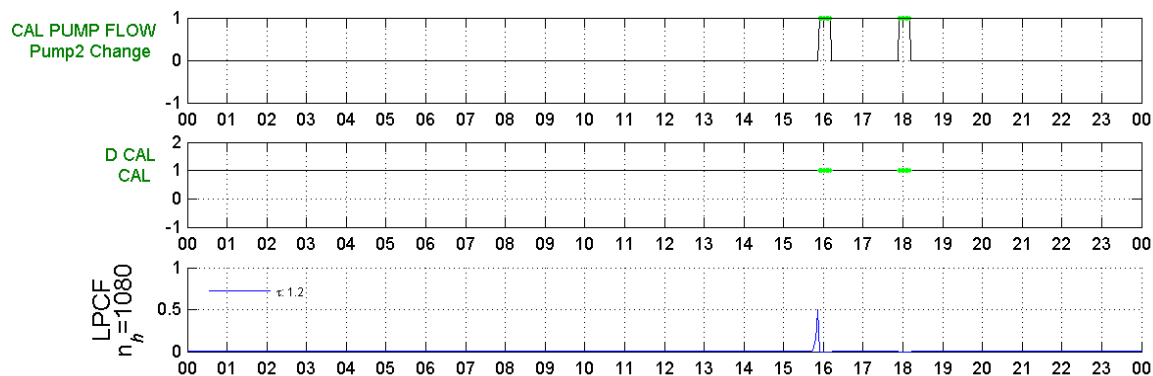


Figure 89: Output plots produced using final composite signal for January 25th.

Additional adjustments to the composite signals can be made to increase or decrease the delay and/or length of the calibration signal created to meet additional needs. Also, information from other operational data streams can be integrated into the water quality event detection process following this example.

5.3 Integrating Tank Level Operational Information Into Event Detection

This tutorial demonstrates how composite signals integrate operational data into the event detection process as a method for reducing false positives by using operational data to create simulated calibration time periods. While this is similar to the previous tutorial, the approach is different.

The data used in this tutorial are found in the file, CLDY_train.csv, found in the “Tutorial_Files\Composite_Tutorials\Composite_Signals_3” directory. The file contains data from a single water quality monitoring station which includes six primary water quality parameters collected every two minutes over a six-week period. The water quality parameters are residual chlorine (CL2), specific conductivity (COND), pH (PH), temperature (TEMP), turbidity (TURB), and total organic carbon (TOC). An alarm status for each sensor, which indicates that the data is unreliable, is also included. A calibration signal for the entire station is provided.

The data at this location is highly variable. Operational data might or might not be useful to help reduce the false positive event detection rate. Therefore, the data file also provides a large number of signals regarding the status of pumps and valves, along with flow rates, tank levels, and pressures, measured from other locations in the same water distribution system. In addition, a secondary chlorine sensor monitors the residual chlorine levels at the tank outlet that feeds water into the main where the water quality monitoring station is located.

The large amount of operational data and the lack of knowledge regarding the correlation between the operations and the water quality at this monitoring station make this a complicated problem. A large number of possible combinations of operational signals could be used to suppress false alarms. A subset of these possible combinations is examined here, with the understanding that other combinations could provide results of equal or improved quality. The configuration file for this tutorial is CLDY_Initial.yml and is found in the “Tutorial_Files\Composite_Tutorials\Composite_Signals_3\Initial” directory. A set of parameters for the LPCF and BED algorithms was developed to detect events. The *monitoring stations* configuration section is shown in the Figure 90.

```
monitoring stations:
|- id: StationD
  station id number:
  station tag name: StationD
  location id number: -1
  enabled: yes
| inputs:
  - id: stationd_in
| outputs:
| signals:
  - id: TEST_CL
  - id: TEST_COND
  - id: TEST_PH
  - id: TEST_TEMP
  - id: TEST_TOC
  - id: TEST_RES_CL
  - id: TEST_TANK_ELEV
  - id: TEST_TANK_FLOW
  - id: TEST_CAL
  - id: TEST_UPS_ALM
| algorithms:
  - id: B1
```

Figure 90: Monitoring stations configuration section.

Figure 91 shows the output produced by CANARY with this configuration file. Five distinct events in the first week of the analysis (08/08-08/14/2008) are identified with multiple water

quality signals contributing to each event. No calibration events were defined during this time period, so there are no bright green time bars on the graph. The TEST_UPS_ALM signal is the only calibration signal in the data set (not shown). Three operation signals that might be correlated with the water quality changes are shown, including the CLDY_TANK_FLOW, CLDY_TANK_ELEV and CLDY_OSXX_ALM signals. The CLDY_TANK_FLOW signal does not provide any useful information for this week. The CLDY_TANK_CL2X signal is the residual chlorine level coming out of the tank into the main and it might provide some useful information on reducing event detections. Additionally, the fluctuations in the CLDY_TANK_ELEV signal could be of use.

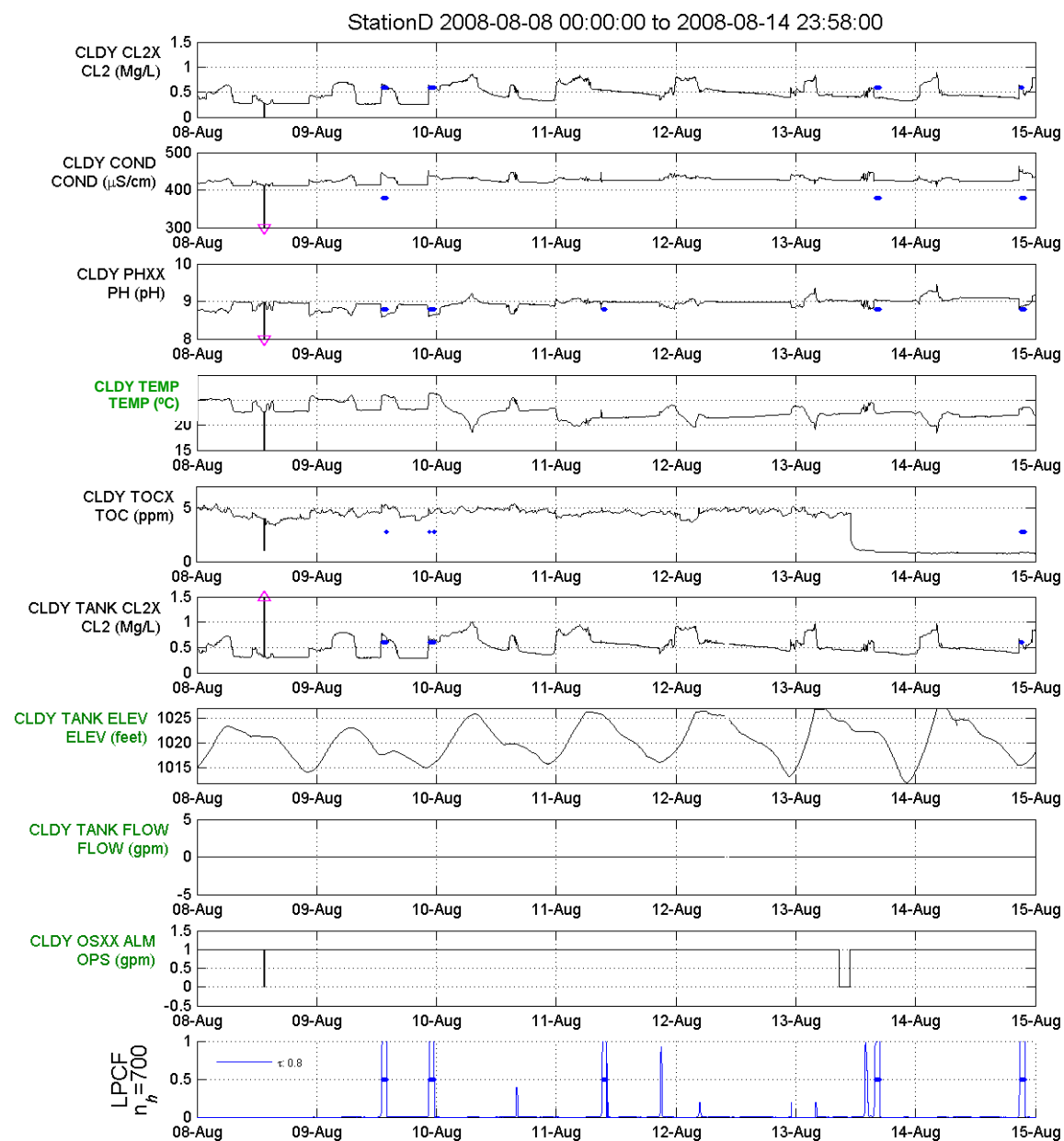


Figure 91: Output graph produced using initial configuration file.

The *signals* section of the configuration file, CLDY_Step1.yml, found in the directory “Tutorial_Files\Composite_Tutorials\Composite_Signals_3\Step1” was altered to include two new composite signals. The first new composite signal, REL_TANK_LVL, normalizes the water elevation level of the tank to a value between 0 and 1. The second new composite signal, TANK_CL_CHANGE, identifies significant changes within the tank outlet residual chlorine value over the previous two time steps. The *signals* sections of the YML file for these signals are shown in Figure 92 and Figure 93.

```
|- id: REL_TANK_LVL
  SCADA tag: REL_TANK_LVL
  evaluation type: op
  parameter type: Calibration Time Out
  ignore changes: none
| data options: # DATA
  precision: 1
  units: ''
  valid range: [0.0, 1.0]
  set points: [-.inf, .inf]
| composite rules: |
  @TEST_TANK_ELEV[0]
  {1.009630e+003}
  -
  {1.027150e+003}
  {1.009630e+003}
  -
  /
```

Figure 92: *Signals* configuration section defining composite signal, REL_TANK_LVL.

In order to build the composite signal that defines the relative tank level, it is necessary to know the minimum (1009.630 ft) and maximum (1027.150 ft) tank levels. The relative tank level is the current value of the tank level (TEST_TANK_ELEV[0]) minus the minimum value (1009.630) divided by the full tank level range (1027.150-1009.630). This transforms the REL_TANK_LVL into a value between 0 and 1.

```

|- id: TANK_CL_CHANGE
  SCADA tag: TANK_CL_CHANGE
  evaluation type: op
  parameter type: Tank Chlorine Change
  ignore changes: none
  # alarm options: #ALARM
  #   value when active: 1
  #   scope: # entire station
| data options: # DATA
  precision: 0.0001
  units: ''
  valid range: [-.inf, .inf]
  set points: [-.inf, .inf]
| composite rules: |
  @TEST_RES_CL[0]
  @TEST_RES_CL[1]
  -
  abs
  @TEST_RES_CL[0]
  @TEST_RES_CL[2]
  -
  abs
  @TEST_RES_CL[0]
  @TEST_RES_CL[3]
  -
  abs
  max
  {1.200000e-001}
  gt

```

Figure 93: Signals configuration section defining composite signal, TANK_CL_CHANGE.

The TANK_CL_CHANGE signal identifies the maximum absolute value of the difference in the TEST_RES_CL signal between the current time step and any of the previous three time steps. This maximum value is then compared to 0.12 and any value greater is a 1, otherwise it is a 0. Thus, TANK_CL_CHANGE is equal to 1 when large changes have recently occurred in the chlorine residual. This composite signal is of *evaluation type* CAL and therefore the *alarm options* parameter information is included in the signal definition. However, since the signal needs to be viewed in the output graph, the *evaluation type* parameter is set to OP and the *alarm options* parameter information is commented out with the # symbols.

Additional changes to the *monitoring stations* section of the configuration file include removing the two operational signals, TEST_TANK_FLOW and CLDY_OSXX_ALM, which do not provide any useful information about the events during this time period. Also, the TEST_TOC signal does not appear to be functioning properly during this time period. In the *signals* section, the TEST_RES_CL signal is converted from the *evaluation type* parameter of WQ to OP for graphing purposes. The two new composite signals are added to the *monitoring stations* section of the configuration file as shown in Figure 94. The main goal at this point is to plot these signals to look for a relationship between them and the observed events.


```

monitoring stations:
|- id: StationD
   station id number:
   station tag name: StationD
   location id number: -1
   enabled: yes
| inputs:
|   - id: stationd_in
| outputs:
| signals:
|   - id: TEST_CL
|   - id: TEST_COND
|   - id: TEST_PH
|   - id: TEST_TEMP
|   - id: TEST_RES_CL
|   - id: REL_TANK_LVL
|   - id: TANK_CL_CHANGE
|   - id: TEST_UPS_ALM
| algorithms:
|   - id: B1

```

Figure 94: *Monitoring stations* configuration section composite signal enabled.

Figure 95 shows the new output graph with the additional composite signals. Examination of the figure shows that the changes in the tank outflow residual chlorine appear to be indicators of events. The relationship between the relative tank level and the events is less clear; however, relative tank levels at or near 1.0 could be the cause of some events. The total number of events for this week is now eight, since the removal of the TOC signal has caused some events that were combined previously to now show up as individual events.

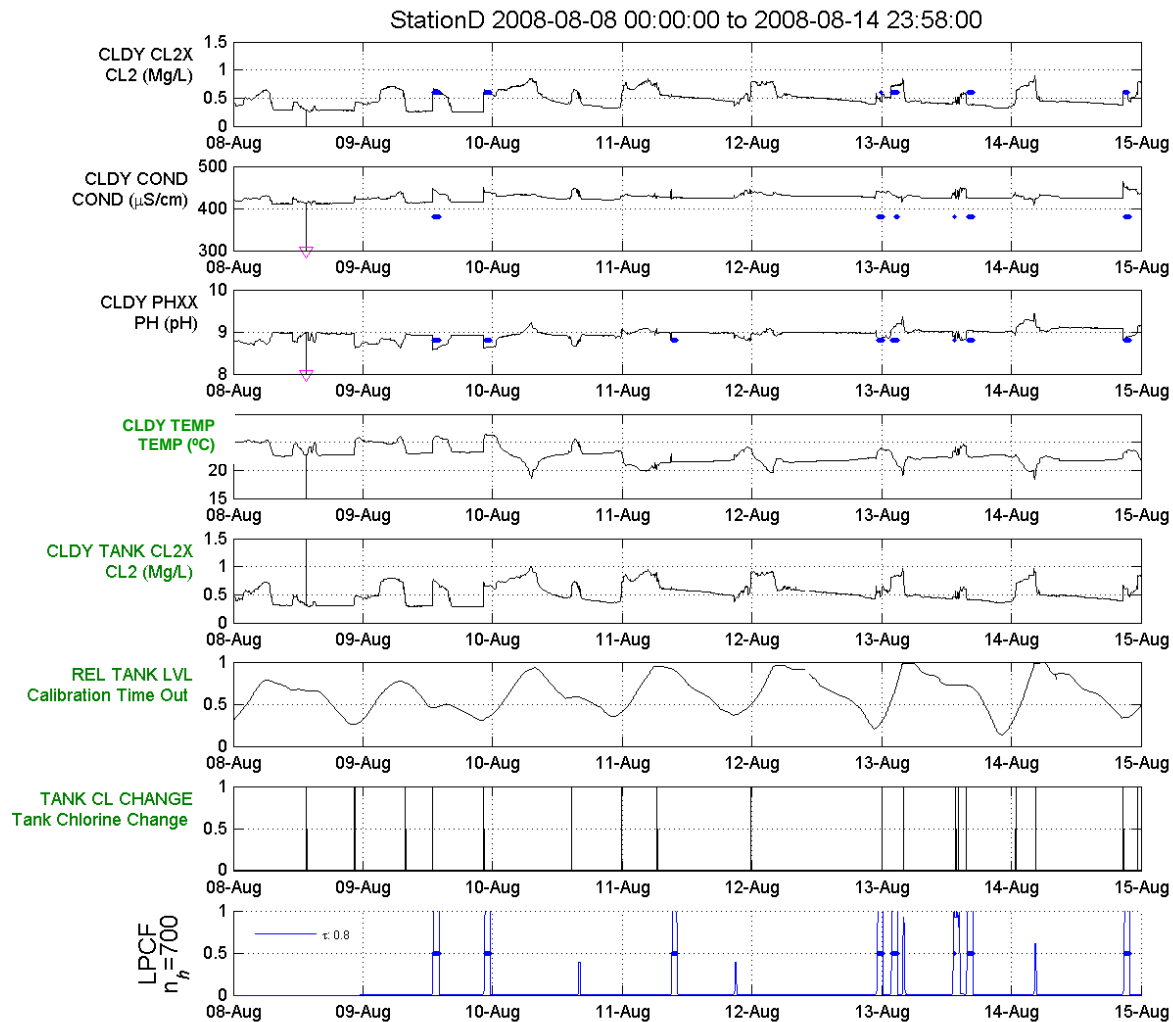


Figure 95: Output graph produced when using composite signal.

In order to test the hypotheses that the water quality events might be linked to changes in the tank chlorine residual, the TANK_CL_CHANGE signal's *evaluation type* parameter is converted back to CAL. In addition, the three lines that previously commented out when the *evaluation type* parameter was OP need to be uncommented. The revised *signals* section of the CLDY_Step2.yml file found in the directory "Tutorial_Files\Composite_Tutorials\Composite_Signals_3\Step2" for the TANK_CL_CHANGE signal is shown in Figure 96. The relative tank level is left as an operation signal and so does not affect the CANARY analysis directly. The *monitoring stations* section does not need to be changed, since the TANK_CL_CHANGE signal is already defined.

```

- id: TANK_CL_CHANGE
  SCADA tag: TANK_CL_CHANGE
  evaluation type: cal
  parameter type: Tank Chlorine Change
  ignore changes: none
  alarm options: # ALARM
    value when active: 1
  composite rules: |
    @TEST_RES_CL[0]
    @TEST_RES_CL[1]
    -
    abs
    @TEST_RES_CL[0]
    @TEST_RES_CL[2]
    -
    abs
    @TEST_RES_CL[0]
    @TEST_RES_CL[3]
    -
    abs
    max
    {1.200000e-001}
    gt

```

Figure 96: *Signals* configuration section with the modified TANK_CL_CHANGE signal.

Figure 97 shows the new output graph produced when using the TANK_CL_CHANGE as a calibration signal. Multiple calibration events occur based on this new composite signal and are shown in green. Only two water quality events are identified. Examination of these two events shows that the first one, on August 11th, is caused by a short-lived change in the TEST_COND and TEST_PH signals and is not correlated with changes in the residual chlorine level out of the tank (TEST_RES_CL signal). In addition, the temperature signal, TEST_TEMP, changes at this same time. The second event, at the end of August 12th, is associated with changes in the TEST_RES_CL signal. The TANK_CL_CHANGE composite signal could be modified to be longer in order to remove this event.

The relative tank level, as defined in the REL_TANK_LVL signal, is not a direct predictor of events. A relationship does exist between periods of tank filling and sudden changes in the TEST_RES_CL values, but the current calibration signal, TANK_CL_CHANGE, already uses the TEST_RES_CL values directly, and the REL_TANK_LVL signal would only add redundant information to the current calibration signal. For this reason, the REL_TANK_LVL signal is not added to the calibration process.

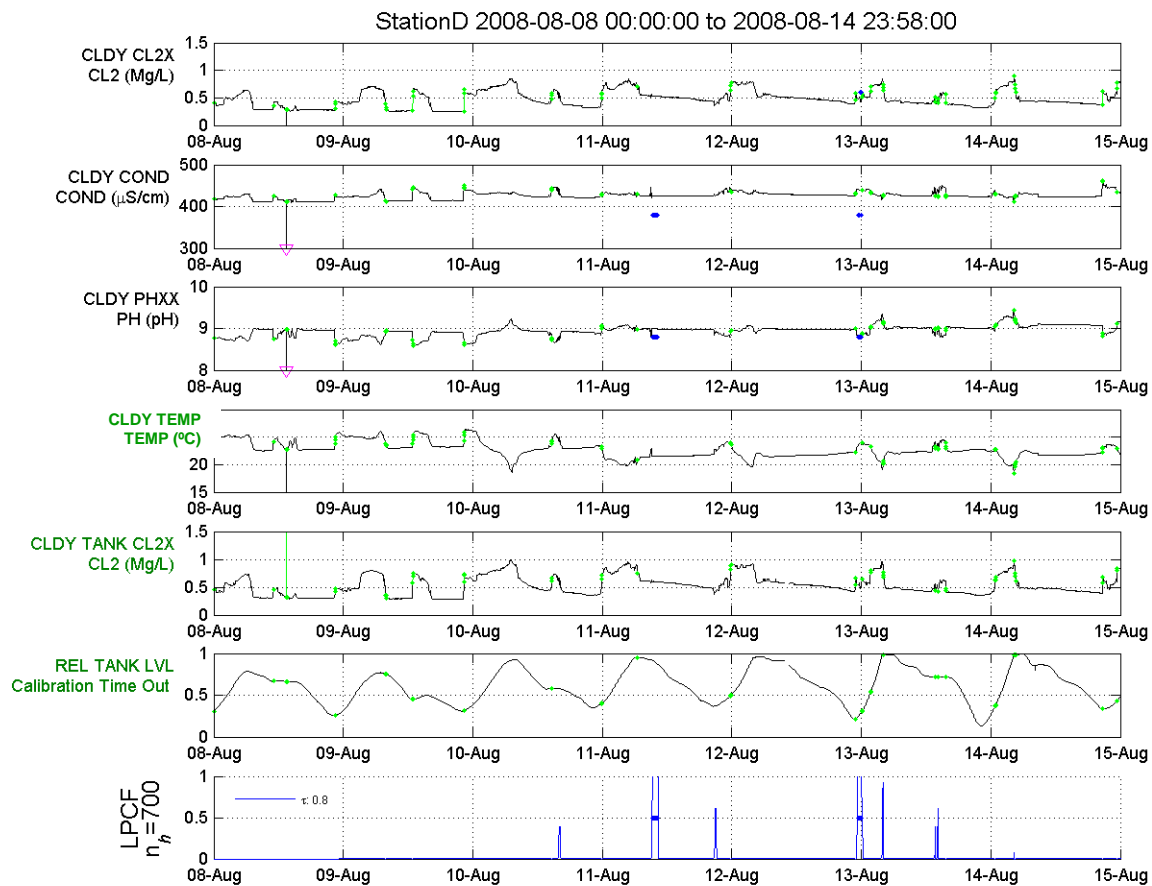


Figure 97: Output graph produced when using modified signal.

In summary, adding a calibration signal that used the residual chlorine level of a tank outflow was effective in reducing unwanted water quality events for this monitoring station downstream of the tank. This tutorial also shows how to use two constant elevation values to define the relative water tank level. Additional information, such as the rate of change (slope) of the tank level could be calculated within a composite signal and added to the event detection process, but that level of sophistication is not needed here.

References

Hart, D. B. and McKenna, S. A. 2012. *CANARY User's Manual, Version 4.3.2*. Washington, D,C.: U.S. Environmental Protection Agency. EPA/600/R/08/040B.

McKenna, S. A., Vugrin, E. D., Hart, D. B, and Aumer, R. A. 2013. Multivariate Trajectory Clustering for False Positive Reduction in Online Event Detection. *Journal of Water Resources Planning and Management*. 139(1): 3-12.

Murray, R., Haxton, T., McKenna, S., Hart, D., Klise, K., Koch, M., Vugrin, E., Martin, S., Wilson, M., and Cruz, V. 2010. *Water Quality Event Detection Systems for Drinking Water Contamination Warning Systems: Development, Testing, and Application of CANARY*. Washington, D,C.: U.S. Environmental Protection Agency. EPA/600/R-10/036.

U.S. EPA. 2012. *CANARY Quick Start Guide*. Washington, D.C.: U.S. Environmental Protection Agency. EPA/600/R-12/010.

Appendix A: Frequently Asked Questions (FAQs)

This section provides answers to frequently asked questions from each of section of *CANARY Training Tutorials*.

CONFIGURATION FILE FAQs

Q. *What are the spacing requirements within an YML file? What conventions need to be paid attention to when editing the configuration file?*

A. Extra lines should be removed, and nothing should be indented using tabs. Indentation should only be done using spaces.

Q. *Can explanatory notes be added to the configuration file?*

A. Yes. Notes are added as comments within YML formatted files. Comments are denoted with a leading # on each line in the configuration file. Two types of comments are shown in Figure 98. The # alarm options is a comment denoting that the *alarm options* parameter is a heading in the YML for alarm information. Another comment is # entire station, which identifies that the *scope* parameter with no value is applied to the entire station. The TANK_CL_CHANGE signal was originally defined as a calibration signal and as such it has the *alarm options* parameter information. The signal was changed to *evaluation type* of OP and the *alarm options* parameters are not needed. Rather than removing these parameters, as they might be used again in the future, the entire *alarm options* parameter block is commented out by adding a # to the start of each line.

```
- id: TANK_CL_CHANGE
  SCADA tag: TANK_CL_CHANGE
  evaluation type: op
  parameter type: Tank Chlorine Change
  ignore changes: none
# alarm options:
#   value when active: 1
#   scope: #entire station
  data options:
    precision: 0.0001
    units: ''
    valid range: [-.inf, .inf]
    set points: [-.inf, .inf]
  composite rules: |
    @TEST_RES_CL[0]
    @TEST_RES_CL[1]
    -
    abs
    @TEST_RES_CL[0]
    @TEST_RES_CL[2]
    -
    abs
    @TEST_RES_CL[0]
    @TEST_RES_CL[3]
    -
    abs
    max
    {1.200000e-001}
    gt
```

Figure 98: Example of comments within a configuration file.

OPTIMIZING CANARY CONFIGURATION FILE FAQs

Q. *Will multiple locations be on the same graph?*

A. No. The EDSD files generated are specific to each station and so are the resulting graphs. This feature allows one configuration file to be used for all the stations, but the results are kept separate on a station by station basis.

Q. *What if BED is not used? Will there still be values on an event probability plot in the output graphs?*

A. If BED is turned off, the resulting probability is the absolute value of the largest residual across all sensors for that time step. It is no longer an actual probability, but a residual with values greater than 1.0. The status column will now indicate an event for every time step where the probability is greater than *event threshold* parameter.

This is a little bit of apples versus oranges in that CANARY is now comparing a residual value against a probability threshold, but it makes sense in that whatever is called probability of event should be consistently compared against the value of *event threshold* parameter. This is why the organization of the configuration file has the parameter definition for *event threshold* parameter outside of the BED parameters.

If a true probability of an event is wanted when using the LPCF or MVNN algorithms, BED must be used. The SPPE and SPPB algorithms will produce a true probability of event value without using the BED. The probability of an event is calculated by the BED using the approach detailed in Murray et al. (2010).

DATABASE DRIVEN INPUT/OUTPUT FAQs

Q. *What do positive or negative values of the time drift parameter mean?*

A. The *time drift* parameter accounts for the discrepancies between the clocks on two different systems. Positive values mean that the computer running CANARY has a clock value **behind** the database (e.g., CANARY's clock says 12:30 but the database says 12:35). A negative value means that CANARY's clock is **ahead** of the database. Because of latency in writing data values to a database, it is a good idea to set this value so that CANARY is looking for new data a minute or two *after* the database writes the data – this way there is less likely to be data loss because of missed reads.

Q. *What are the units of the time drift parameter?*

A. The *time drift* parameter is defined in fractions of a day. It is specified in decimal format.

Q. *My database JDBC driver has several classes, which class is the right one?*

A. The class will have DataSource in its name, probably at the end of the class name. Pool time classes are okay. The main thing to ensure is that this is a JDBC2 DataSource class, which

should be listed in the class ancestors.

Q. *My database vendor has several different JAR files on their website, which one do I download?*

A. This depends on which version of the database is being used. For example, Oracle™ has different drivers for versions 10 and 11 of their full database and different versions for their free software as well. Many of the newest installations will have a *jdbc*.jar file somewhere in the installed Program Files folder, but when in doubt, ask the database administrator, the contractor, or employee in charge of maintaining the database. Again, CANARY needs the JDBC2 drivers.

Q. *How do I set the URL for my database?*

A. The address will look similar to the addresses in the examples. However, this is very database specific, and not every example can be shown. When in doubt, talk to the local database contact.

Q. *I do not want my username and password in the configuration file. Does this mean I have to type it in every time I use CANARY?*

A. Yes. The user will either need to protect the configuration file with access controls or enter a password every time, because CANARY does not have encryption capabilities. Even with access controls, it is always a good idea to make the CANARY user have the least privileges possible to protect both systems and the database.

COMPOSITE SIGNALS FAQs

Q. *What are the mathematical operations that can be used in composite signals?*

A. A description of the permissible operations and the symbols that define them are included in the CANARY User's Manual (Hart and McKenna 2012).

Q. *When should I use pattern matching with trajectory clustering versus using composite signals to decrease the events caused by water quality patterns?*

A. If operational data are available that provide cues to when a water quality event is about to happen, then composite signals are generally easier to implement and refine. When operational data are not available, or not closely tied to the change in water quality (as can occur when the operational change and the monitoring station are not located together), then pattern matching can be implemented.

Appendix B: Binomial Distribution Function Exercise

A useful exercise to help understand how CANARY's BED parameters interact is to examine the effect of the different parameters within a spreadsheet by using the binomial distribution function. The BED is based on a binomial failure model that looks at the number of failures (NFAILURES) within a certain number of trials (NTRIALS) given that the chance of a failure occurring in any single trial (PFAIL) is constant. In terms of the CANARY event detection process, NFAILURE is an outlier, NTRIALS is the number of time steps within a user-defined window, and PFAIL is the chance of an outlier occurring at any time step. If there are more outliers within the window than would be predicted, then this increases the likelihood that an event is occurring.

Two parameters control the sensitivity of the event detection when using the BED, the size of the BED *window* and the BED *event threshold*. While the *event threshold* parameter is not directly one of the BED parameters, it uses the output of the BED to determine when an event occurs. By changing the BED *window* parameter, the user can increase or decrease CANARY's sensitivity. For this tutorial, Microsoft® Office Excel® was used; however, any spreadsheet software that includes a binomial distribution function can be used. The spreadsheet function used to compute the probability of an event is the following:

Probability of event = BINOMDIST(NFAILURES, NTRIALS, PFAIL, CDF)

Where the relationship between the parameters in the spreadsheet function and those in CANARY are:

NTRIALS = BED *window*

PFAIL = BED *outlier probability*

CDF = 1.0 (Not an input to CANARY)

NFAILURES = The number of outlier time steps that occur within the BED window. This number is calculated within CANARY.

For this tutorial, complete the following steps:

- Open a Microsoft® Office Excel® file and make a column that contains the numbers 0 to 20. This column will represent the NFAILURES (number of outliers) in the function (Figure 99).

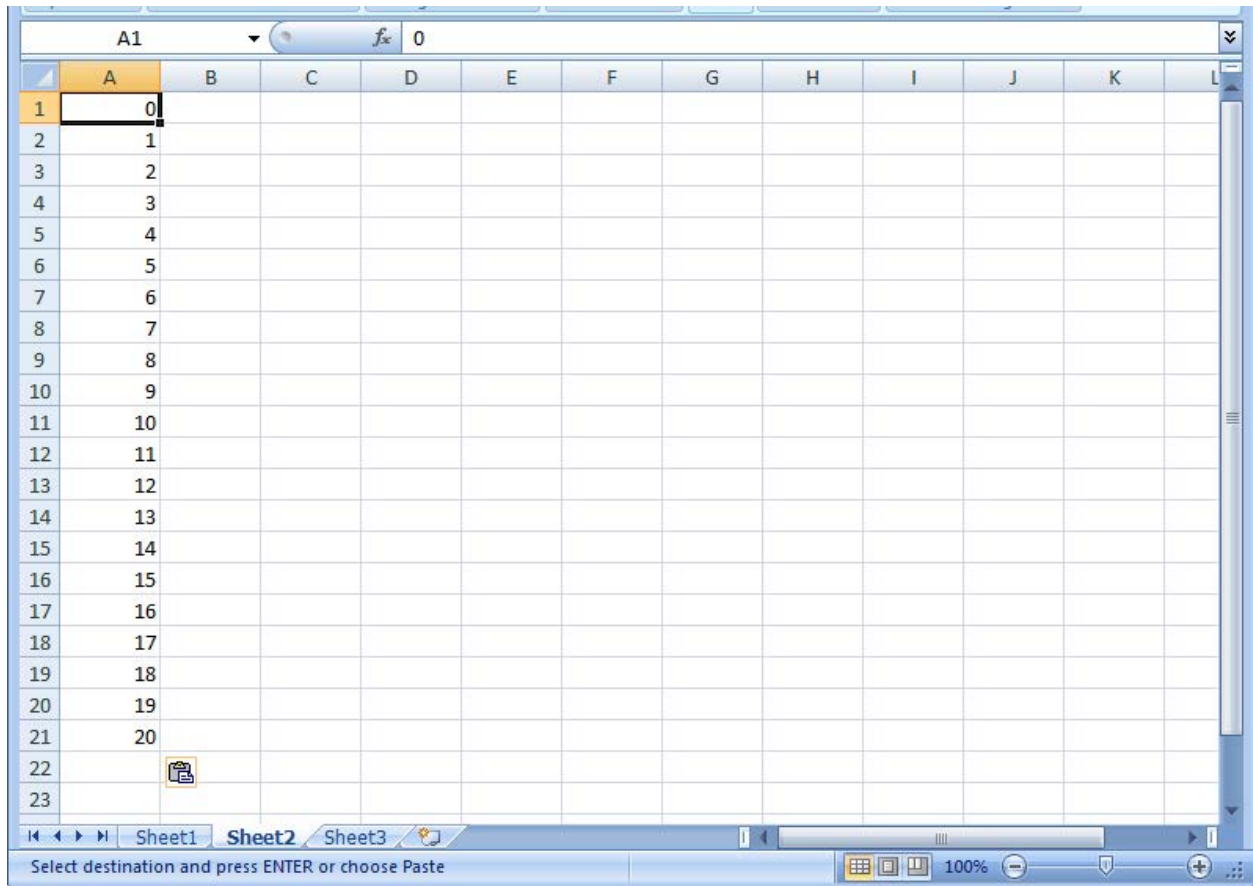


Figure 99: NFAILURES column using a NTRIALS of 20.

- In the B1 cell, type the following function: = BINOMDIST(A1, 20, 0.5, 1) (Figure 100). This function uses the Column A values as the number of outliers within the BED window.

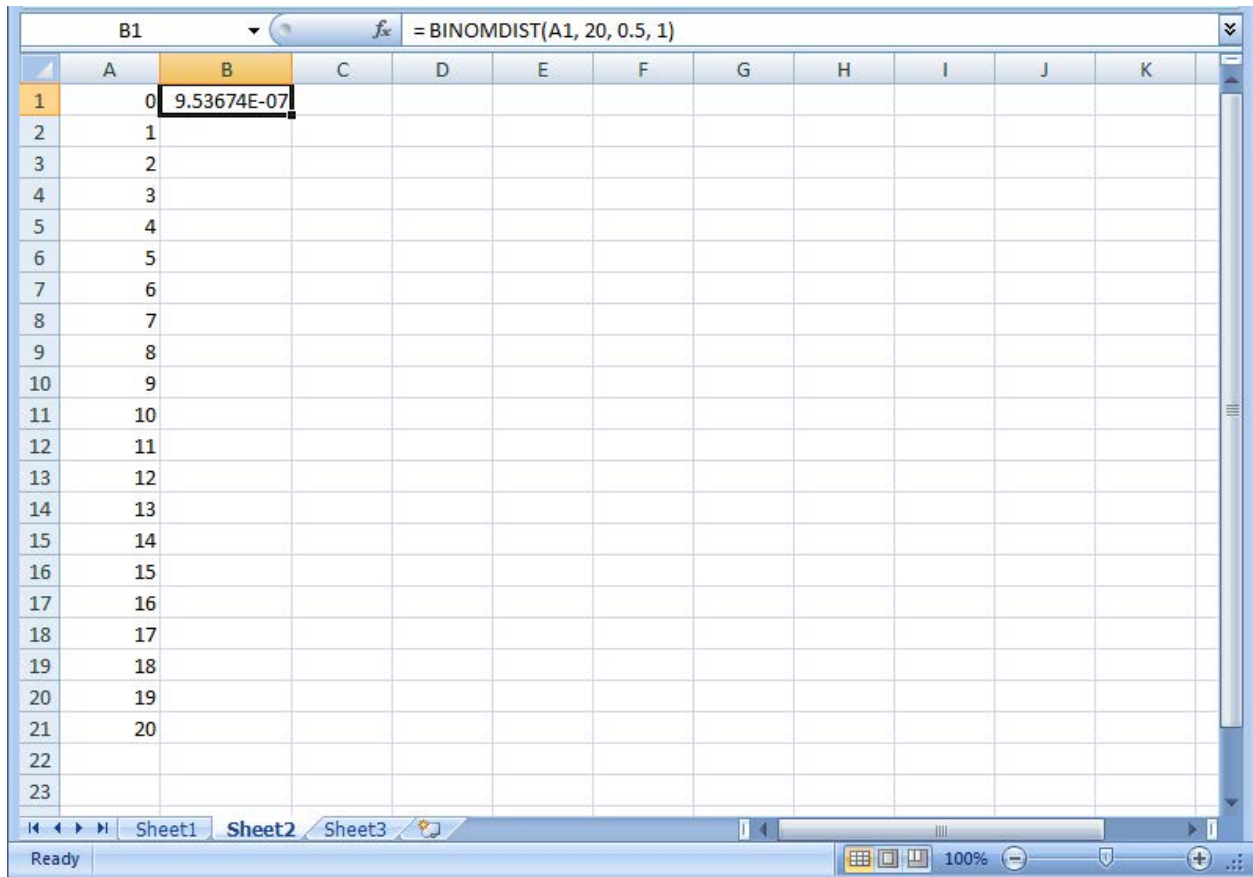


Figure 100: Binomial distribution function using a NTRIALS of 20 and a PFAIL of 0.5.

- Copy the B1 cell to cells B2 through B21 (Figure 101).

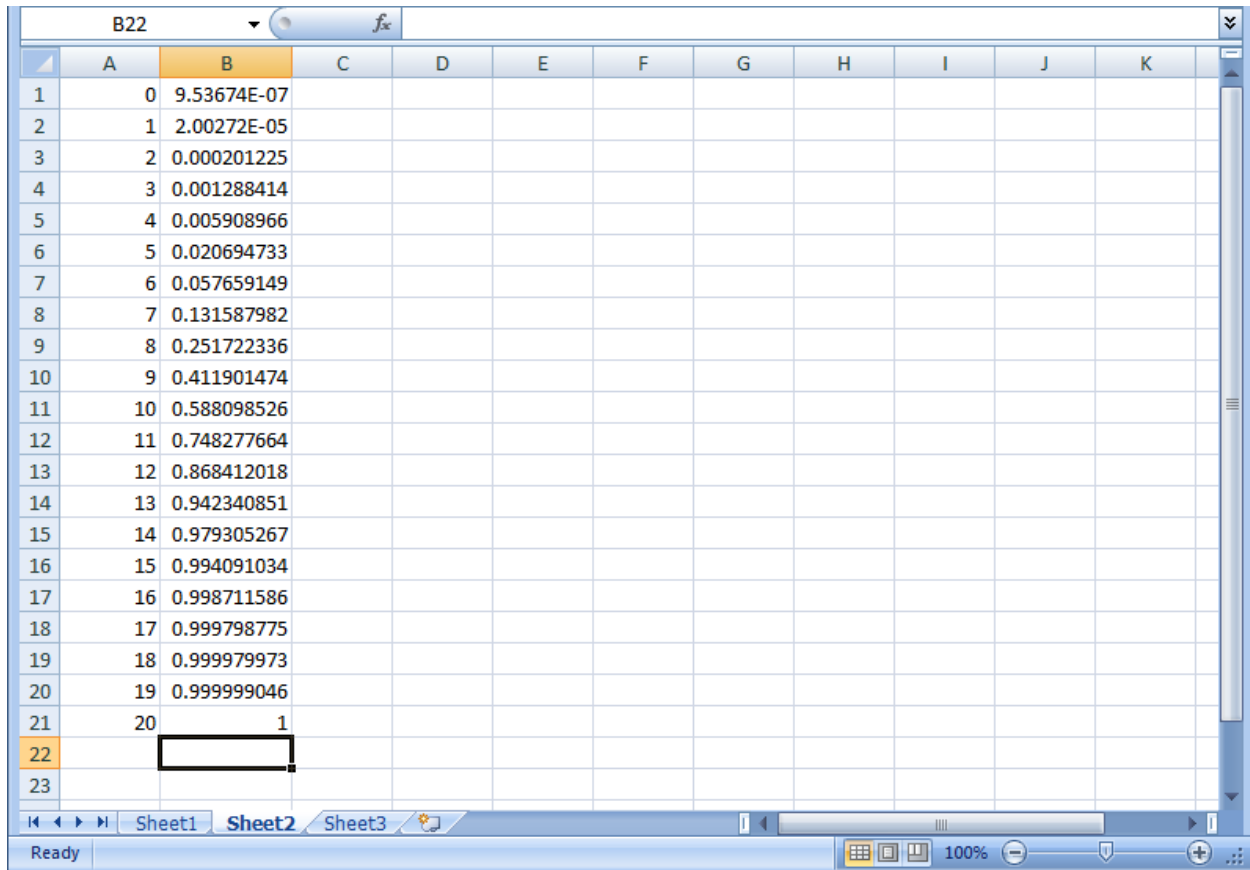


Figure 101: Probability of event values for a NTRIALS of 20.

The Column B values are the calculated probability of an event, which increase as the number of outlier time steps within the BED window increases, until it becomes 100% at 20 outliers. The probability of event values approach 1.0 asymptotically; even with 15 of the time steps in the BED window being outliers, the probability of event value is already above 0.99. A graph of this can be seen in Figure 102. The sensitivity of the event detection can also be modified by using different *event threshold* parameter values. For example, if the *event threshold* parameter is set to 0.85, 12 or more outliers (failures) within the BED *window* parameter are needed to cause an event as shown in Figure 102. If the *event threshold* parameter is increased to 0.99, and all other parameter values are held constant, 15 or more outliers within the window are needed to cause an event.

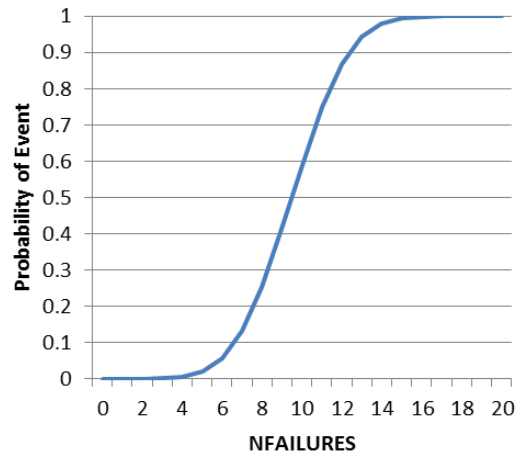


Figure 102: Probability of event graph using a NTRIALS of 20.

To increase the sensitivity of the BED function, the size of NTRIALS (BED window) is decreased. The previous steps are repeated using a NTRIALS of 6 instead of 20. The function is the following = BINOMDIST(A1, 6, 0.5, 1). The probability of event values using a NTRIALS of 6 is shown in (Figure 103).

	A	B	C	D	E	F	G	H	I	J	K
1	0	9.53674E-07		0	0.015625						
2	1	2.00272E-05		1	0.109375						
3	2	0.000201225		2	0.34375						
4	3	0.001288414		3	0.65625						
5	4	0.005908966		4	0.890625						
6	5	0.020694733		5	0.984375						
7	6	0.057659149		6	1						
8	7	0.131587982		7	#NUM!						
9	8	0.251722336		8	#NUM!						
10	9	0.411901474		9	#NUM!						
11	10	0.588098526		10	#NUM!						
12	11	0.748277664		11	#NUM!						
13	12	0.868412018		12	#NUM!						
14	13	0.942340851		13	#NUM!						
15	14	0.979305267		14	#NUM!						
16	15	0.994091034		15	#NUM!						
17	16	0.998711586		16	#NUM!						
18	17	0.999798775		17	#NUM!						
19	18	0.999979973		18	#NUM!						
20	19	0.999999046		19	#NUM!						
21	20	1		20	#NUM!						
22											
23											

Figure 103: Probability of event values for a NTRIALS of 6.

With a shorter NTRIALS (BED window) of 6, the probability of an event increases much faster than with a NTRIALS of 20. Using *event threshold* parameters of 0.85 and 0.99 would now cause an event after four or five outliers. As seen in Figure 104, a 100% probability is reached at 6 outliers instead of 20. Having more outliers than the length of the window is not possible (i.e., 7 outliers out of 6 time steps is impossible); therefore, the #NUM error appears in the lower cells of Column E.

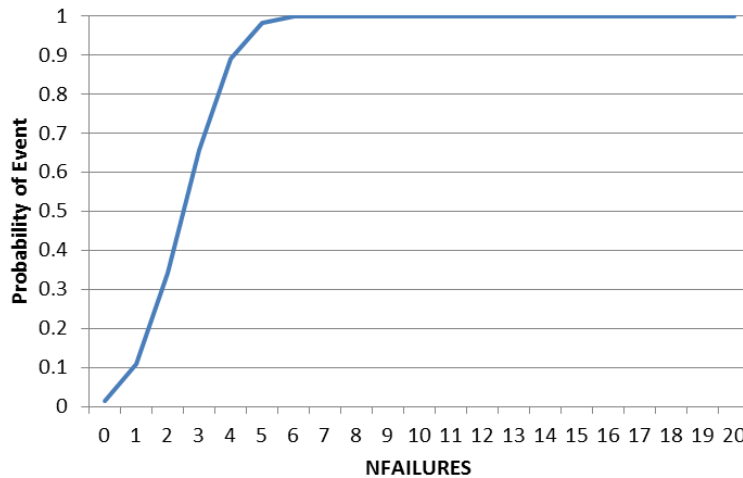


Figure 104: Probability of event graph using a NTRIALS of 6.

Appendix C: Configuration File Quick Reference

This section is intended to provide a quick reference to create or edit CANARY configuration files. The input parameters associated with each section of the configuration file along and section examples based on the tutorial files are provided. Example configuration files can be found in the examples folder created after installing CANARY (...\\My CANARY\\examples\\), or in the tutorial archive (ZIP) that corresponds to *CANARY Training Tutorials*. For more information regarding CANARY configuration files, see Sections 5 and Appendix A of the CANARY User's Manual (Hart and McKenna 2012).

The main sections of the CANARY configuration file are shown in Table 3. The parameters for each section along with their options, syntax, and additional information are shown in following tables. Defaults, if applicable, are in **bold** in the Options column. If a parameter is optional, then a checkmark is in the Optional column.

Table 3: CANARY Configuration File Sections

Section	Contents
canary	Basic information about the CANARY run
timing options	Date and time range of information, data intervals
data sources	Input file or database information
signals	Lists the signal information from the input file or database
algorithms	Specifies the event detection algorithm
monitoring stations	Identifies the monitoring stations, signals, and algorithms for analysis

The first section of the CANARY configuration file is the *canary* section. This section provides the basic information on how a CANARY analysis is executed. Table 4 shows the input parameters for this section. Section 5.1 of the CANARY User's Manual (Hart and McKenna 2012) provides more details on these parameters.

Table 4: Input Parameters for *canary* Section of the CANARY Configuration File

Input Parameters	Options	Optional
run mode:	BATCH, REALTIME, EDDIES	
control type:	INTERNAL, EXTERNAL, EDDIES	
control messenger:	null (for INTERNAL control type), else defines data sources for EXTERNAL or EDDIES control type	✓
use continue:	Specify a restart file to pre-populate data (e.g., continue.edsd). Used primarily with REALTIME.	✓
data provided:	NEW VALUES, ALL VALUES	✓
driver files:	null , or specify any driver files required to run. Precede each	

Input Parameters	Options	Optional
	entry with a minus (-) symbol.	

Figure 105 shows a basic example of a *canary* section using BATCH mode. Figure 106 highlights the use of the *driver files* parameter for connecting CANARY to a database. Note the indentation levels for each type of input and the use of the minus (-) symbol to reference the location of the database specific driver file. The required database driver file must be downloaded from the database developer's website or installation CD. Its location on your machine must also be specified. In this example, it is stored in the lib subfolder within CANARY's install location. This was done for convenience. Section 4 of *CANARY Training Tutorials* provides more information on connecting CANARY with databases.

```
canary:
  run mode: BATCH
  control type: INTERNAL
  control messenger: null
  driver files: null
```

Figure 105: Example of *canary* section using BATCH mode.

```
canary:
  run mode: BATCH
  control type: INTERNAL
  control messenger: null
  driver files:
    - C:\Program Files (x86)\CANARY\lib\database_specific-bin.jar
```

Figure 106: Example of *canary* section using a database connection.

The second section of the CANARY configuration file is the *timing options* section. This section provides the date and time information for the data to be analyzed. Table 5 shows the input parameters for this section. Section 5.2 of the CANARY User's Manual (Hart and McKenna 2012) provides more details on the *timing options* section.

Table 5: Input Parameters for *timing options* Section of the CANARY Configuration File

Input Parameters	Options	Optional
dynamic start-stop:	off (usually used by BATCH and EDDIES) on (usually used in REALTIME)	
date-time format:	Common formats: US standard – mm/dd/yyyy HH:MM AM European Std – dd/mm/yyyy HH:MM:SS ISO Std – yyyyymmddTHHMMSS ODBC Database canonical – yyyy-mm-dd HH:MM:SS	
date-time start:	Input start time of analysis	
date-time stop:	Input stop time of analysis	
data interval:	HH:MM:SS Format (e.g., 5 minutes = 00:05:00)	

Input Parameters	Options	Optional
message interval:	Time to wait for input, or sleep time. Generally, should be smaller than <i>data interval</i> .	

Figure 107 contains an example of the *timing options* section. This example is requesting analysis from May 28th 2013 to July 1st 2013 at a data interval of 2 minutes.

```
# Enter the time step options below
timing options:
  dynamic start-stop: off
  date-time start: 2013-05-28 00:00:00
  date-time stop: 2013-07-01 00:00:00
  date-time format: yyyy-mm-dd HH:MM:SS
  data interval: 00:02:00
  message interval: 00:00:10
```

Figure 107: Example of *timing options* section.

The third section of the CANARY configuration file is the *data sources* section, which provides the input and output information for a CANARY analysis. Table 6 shows the input parameters for this section. Section 5.3 of the CANARY User’s Manual (Hart and McKenna 2012) provides more details on the *data sources* section.

Table 6: Input Parameters for *data sources* Section of the CANARY Configuration File

Input Parameters	Options	Optional
- id:	Internal identifier. Text string. No spaces are allowed (use dash or underscore) and field is case sensitive.	
type:	CSV, DB (or JDBC), EDDIES, XML	
location:	File name or URL of data source (can contain “..” or full path to direct to local files not located in the same folder as the YML)	
enabled:	yes or no	
configFile:	Replaced by <i>database options</i> :	✓
timestep options:	field:	Specify the column header of the date/time information
	format:	Can be omitted if it matches the global date format
	conversion function:	Required to convert string formats in databases
database options:	time drift:	Units of fractional day (e.g., 0.75=18hr)
	JDBC2 class name:	Java class inside the driver
	input table:	Specify the table or view inside the database that contains the input data
	input format:	default , row-based or custom (case sensitive)

Input Parameters	Options	Optional
	input fields:	<ul style="list-style-type: none"> • <u>time-step</u>: (field in which the time stamp of the new data is listed) • <u>parameter tag</u>: (field in which the SCADA tag is listed) • <u>parameter value</u>: (field in which the SCADA value is listed) • <u>parameter quality</u>: (field in which the quality flag might be listed. This must be a string that is either “Normal” or “Bad” (case insensitive).)
	output table:	Specify the location of the table inside the database into which the CANARY results will be placed
	output format:	default , extended, custom (case sensitive)
	output fields:	<p>Starts the output fields subsection. Can be turned off with no or false. <u>custom</u> – no default field names are provided for any field <u>default</u> or <u>extended</u> can use the commands below to override default fields</p> <ul style="list-style-type: none"> • write conditions: (all) • time-step: (field for time/date information) • instance id: (defines the field for the CANARY instance ID that created the values) • station id: (station definition that provided the results) • algorithm id: (field for the algorithm that generated the results) • parameter tag: (single value of the SCADA parameter tag that caused the event) • parameter residual: (field for the residual of the parameter. Omit unless specifying parameter tag.) • parameter type: (field that contains the parameter type.

Input Parameters	Options	Optional
		<p>Only valid when using parameter tag.)</p> <ul style="list-style-type: none"> • event code: (field for the integer event code) • event probability: (field for the event probability) • contributing parameters: (field for whitespace delimited contributing parameters) • comments: (field for analysis comments) • pattern match id: (field for the pattern with the highest match pattern) • pattern match probability: (field for the probability of highest match probability) • secondary match id: (field for the pattern with the 2nd highest match pattern) • secondary match probability: (field for the probability of 2nd highest match probability) • tertiary match id: (field for the pattern with the 3rd highest match probability) • tertiary match probability: (field for the probability of 3rd highest match pattern)
	login info:	<p>prompt for login: yes or no</p> <p>username:</p> <p>password:</p>

Figure 108 shows an example of the *data sources* section for a CSV data file. Section 4 of *CANARY Training Tutorials* provides more information regarding database connections.

```
# Enter the list of data sources below
data sources:
- id: stationb_in
  type      : csv
  location   : Tutorial_Station_B.csv
  enabled    : yes
  timestep options:
    field: "TIME_STEP"
```

Figure 108: Example of *data sources* section using a CSV file.

The fourth section of the CANARY configuration file is the *signals* section, which defines the data signals for a CANARY analysis. Table 7 shows the input parameters for this section. Each signal must begin with the *id* parameter and subsequent lines should match the indentation of this parameter. Section 5.4 of the CANARY User’s Manual (Hart and McKenna 2012) provides more details on the *signals* section.

Table 7: Input Parameters for *signals* Section of the CANARY Configuration File

Input Parameters	Options	Optional
- id:	Case sensitive string. Cannot include spaces or symbols. Must be unique and cannot be a portion of another id. (e.g., ‘CL_PUMP1’ and ‘CL_PUMP1_1’ will produce an error, but ‘CL_PUMP_1_1’ will not.)	
SCADA tag:	Signal name given in the CSV or database file. Must match the column header, database field, or the database table value.	
evaluation type:	WQ (water quality) OP (operations) – Can also contain calibration data, if multiple calibration signals occur in one station ALM (alarm) CAL (calibration) – Only one per station	
parameter type:	Short text used as axis label for CANARY plots (e.g., pH or CL2). In EDDIES mode, they must match EDDIES definitions.	
ignore changes:	all, increases, decreases, both, none	✓
data options:	Needed for all water quality (WQ) and operational (OP) signal types	
precision:	Specify the noise threshold for the signal. Tied to sensor precision.	
units:	Label for CANARY plots. Recognizes LaTeX character strings (e.g., $\{\mu\} = \mu$).	
valid range:	Values outside the valid range are treated as originating from a faulty sensor and are ignored. Two unit vector (e.g., $[-.inf, .inf]$ or $[0, 2]$).	
set points:	Used for event detection. If either value is outside the valid range then the set point alarm does not occur. Two unit vector.	
alarm options:	Used with alarm (ALM) and calibration (CAL) signals	
value when active:	1 or 0 (The majority of utilities define normal operations as 0 and alarm/calibration as 1. The opposite can also be handled.)	
scope:	SCADA Tag. Specifies which signal (WQ or OP types) the alarm signal applies. Blank for CAL.	

Input Parameters	Options	Optional
composite rules:	Defines a composite signal. Should be followed by a pipe command (). Commands use Reverse Polish Notation (RPN). Limited to about 12 calculations per 'composite rule:' command.	

Figure 109 contains a portion of the *signals* section. This specific example is for a chlorine signal that is stored under the *SCADA tag* parameter B_CL2_VAL but will be displayed in CANARY as TEST_CL (*id* parameter).

```
# Enter the list of SCADA/composite signals/parameters below
signals:
- id: TEST_CL
  SCADA tag: B_CL2_VAL
  evaluation type: wq
  parameter type: CL2
  ignore changes: none
  data options:
    precision: 0.0035
    units: 'Mg/L'
    valid range: [-.inf, .inf]
    set points: [-.inf, .inf]
```

Figure 109: Example of *signals* section using a chlorine signal.

The fifth section of the CANARY configuration file is the *algorithms* section, which defines the algorithms to be used for a CANARY analysis. Table 8 shows the input parameters for this section. Each algorithm must begin with the *id* parameter and subsequent lines should match the indentation of this parameter. Section 5.5 of the CANARY User's Manual (Hart and McKenna 2012) provides more details on the *algorithms* section.

Table 8: Input Parameters for *algorithms* Section of the CANARY Configuration File

Input Parameters	Options	Optional
- id:	Internal identifier. Text string. No spaces are allowed (use dash or underscore) and field is case sensitive. Same indent behavior as the <i>signals</i> section.	
type:	LPCF, MVNN, SPPE, SPPB, JAVA, CAVE, CMAX	
history window:	Number of prior time steps to include in history. Rule of thumb is to include 1.5 to 2 days of previous data (e.g., 1 day = 1440 minutes => 1440/2 minutes/time step = 720).	
outlier threshold:	Number of standard deviations of prediction error that must be met or exceeded before declaring a value an outlier	
event threshold:	Specify the probability of an event that must be exceeded prior to declaring a series of outliers an event	
event timeout:	Number of consecutive time steps after an event is detected before the alarm is automatically silenced	

Input Parameters	Options	Optional
event window save:	Amount of time to be saved prior to an event being reported. Used only for plotting the identified events in post processing.	
BED:	<div> <div>window:</div> <div>Size of the binomial window. Must be less than history window. Typical values are 4 to 18 steps.</div> </div> <div> <div>outlier probability:</div> <div>Should be left at 0.5. The probability of failure for each binomial trial.</div> </div>	
external algorithm class:	Can only be used if external algorithm has been created. It is the class name of the external algorithm being called.	✓
external algorithm config:	Defines the external algorithm configuration. The XML configuration should be added here in double-quotes.	✓
cluster library file:	File name of the clustering data file to use with the algorithm	✓
use algorithm inputs:	Specifies which algorithm to use in the CAVE or CMAX consensus algorithms. List of algorithm ids. Each line starting with a (-) symbol.	

Figure 110 displays an *algorithms* section example that defines the algorithm as LPCF using the BED option.

```
# Enter the list of event detection algorithms below
algorithms:
- id: test
  type: LPCF
  history window: 144
  outlier threshold: 0.8
  event threshold: 0.85
  event timeout: 12
  event window save: 30
  BED:
    window: 6
    outlier probability: 0.5
```

Figure 110: Example of *algorithms* section using the LPCF algorithm.

The last section of the CANARY configuration file is the *monitoring stations* section. This section defines the monitoring station location, the signals, and the algorithms to be used for a CANARY analysis. Table 9 shows the input parameters for this section. Each algorithm must begin with the *id* parameter and subsequent lines should match the indentation of this parameter. Section 5.6 of the CANARY User’s Manual (Hart and McKenna 2012) provides more details on the *monitoring stations* section.

Table 9: Input Parameters for *monitoring stations* Section of the CANARY Configuration File

Input Parameters	Options	Optional
------------------	---------	----------

Input Parameters	Options	Optional
- id:	Basis for naming output files. Text string. No spaces are allowed (use dash or underscore) and field is case sensitive. Same indent behavior as the <i>signals</i> section.	
location id number:	Integer. User-defined physical location.	
station tag name:	Defines a station tag name for use in the LOCATION_ID field of output tables. If omitted the station id string will be used instead.	
station id number:	Configuration number for the station. Can be used to differentiate between two different substations or two different algorithms using the same signal data.	
enabled:	yes or no. This allows one configuration file to have multiple analysis steps configured for a data stream, but only run a subset at any given time.	
inputs:	'- id:' (Must match the id from the <i>data sources</i> section)	
outputs:	'- id:' (Must match the id from the <i>data sources</i> section)	✓
signals:	'- id:'	Must match the id in the <i>signals</i> section
	cluster:	yes or no
algorithms:	'- id:'	Must match the id in the <i>algorithms</i> section
	cluster:	yes or no

Figure 111 shows a *monitoring stations* section example. Each of the *id* parameters refers to the internal names that correspond to those in the *data sources*, *signals* and *algorithms* sections.

```
# Enter the list of monitoring stations below
monitoring stations:
- id: StationB
  station id number:
  station tag name: StationB
  location id number: -1
  enabled: yes
  inputs:
    - id: stationb_in
  outputs:
  signals:
    - id: CAL_StationB
    - id: TEST_CL
    - id: TEST_PH
    - id: TEST_TEMP
    - id: TEST_COND
    - id: TEST_TURB
    - id: TEST_PRES_PLNT
    - id: TEST_FLOW_PLNT
    - id: TEST_TOC
    cluster: no
  algorithms:
    - id: test
```

Figure 111: Example of *monitoring stations* section.

Appendix D: File Types

Input files

- **.yaml** or **.edsy** – file that specifies the configuration details; double click this type of file to run CANARY, or right-click and choose "Edit" to open the configuration editor in a text editor. Older configuration files were written in XML format and ended in EDSX. These older files can still be read by CANARY. The EDSY extension hooks the file into the right-click abilities of the Windows® operating system; otherwise it is equivalent to the YML
- **csv** – file that provides input data for analysis from a spreadsheet. If the data is to be read from a database, then the YML file must contain instructions for accessing the database.

Output files

- **out.yaml** – copy of the YML configuration file which was run.
- **heartbeat.dat** – file that details the specific date and time CANARY was run.
- **.edsd** – output files for each monitoring station and one that summarizes all of the output; graphed when double clicked.
- **status.log** – file that tracks the history of all the actions taken at each time step.
- **station.log** – file that tracks the history of the CANARY analysis on each day.
- **CONTROL.msg.log** – file that tracks the history of messages passed between analysis outputs and Control, the part of the code that controls when actions are executed; will open in a text editor when double-clicked.
- **Test Station.Summary.txt** – summary of the CANARY run that logs each detected event and summarizes the inputs to the analysis.
- **.png** – image file used by CANARY that displays graphed data. This file is created by right clicking the EDSd file and selecting Graph Data.

Glossary

- **baseline-change** – A long-term sustained change in the average behavior of a signal. The length of time that is required to be considered a baseline-change is definable by the user. The initial change will trigger an event alarm, however after the user-defined length of time the alarms will be suppressed for two reasons: (1) The CANARY alarm is no longer needed because operators have identified the cause or initiated action to find the cause; or (2) because enough time has passed, operators would like CANARY to begin using the new baseline-value to calculate future events. Not all baseline-changes last long enough to be considered baseline-changes.
- **binomial event discriminator (BED)** – A statistical algorithm that is used to determine how many outliers are needed in a given time-frame to constitute an event alarm. Parameters for this algorithm are specified by the user. (See Section 3.3 and Appendix B of this document, or the CANARY User’s Manual Section 2.5.1 for more information.)
- **clustering** – A process of identifying normal changes in water quality patterns. Cluster files are used to keep a library of pattern information to identify “normal events” and decrease false alarms. Typically one or more operation signals provide useful information to a clustering algorithm.
- **data-interval** – (also called *sampling interval*) The frequency at which CANARY expects data signals. A common data interval is two minutes (data interval: 00:02:00). Only data that is most recent, or matches the data-interval best, will be used by CANARY for analysis.
- **evaluation type** – This parameter defines the type of data signal. The options are WQ (water-quality), OP (operations), ALM (data alarm), and CAL (calibration). Only signals defined as WQ are used to detect events. Only one CAL signal can be used by each station. To use multiple calibration signals, define the signals as OP signals, and combine them into a single CAL signal using a composite signal.
- **event** – A period of sustained anomalous activity (i.e., a number of outliers occurring within a short period of time). The binomial event discriminator is used to determine the number of outliers in a given time-frame that will trigger an event alarm signal.
- **false alarm** – An event alarm signaled by CANARY that does not correlate to an actual event. Alarms may be caused due to calibration events, operational changes within the system or configuration parameters that are too sensitive.
- **monitoring station** – A set of sensors and their time series signals that are used together for event detection. These are made up of sensor hardware that is all located in the same physical testing site. Generally, a monitoring station includes some water quality signals, and may also transmit operations or alarm signals.
- **normalization window** – The user-definable time-frame in which short-term statistical

calculations are made. Data within the normalization window is normalized to have a mean of zero and a standard deviation of 1.0. The duty cycle of a given location should be considered when specifying the normalization window. For example, a site below a tank that fills and drains on a daily schedule may need a window size of slightly longer than one day, while an in-network location may only need three to six hours of history.

- **outlier** – A data point at a single time step that is anomalous (i.e., deviates by more than a threshold value) relative to the background or predicted behavior for that signal at a given time step. Multiple outliers in a given time-frame may constitute an event.
- **precision** – The precision limit of sensor hardware. Used by CANARY to help reduce false alarms by telling CANARY the smallest change that can be reported by a sensor (e.g., a change from one time step to another of one (1) in a conductivity signal does not have the same implications as a change of one in a pH signal).
- **probability threshold (τ_B)** – The probability of an event that must be exceeded before an event is signaled by CANARY.
- **residual** – The value calculated by CANARY that is equal to the difference between the measured and predicted water quality signal value at a single time step. The absolute value of this calculated residual is compared to the threshold value to determine if the data point is an outlier.
- **signal** – A data stream, usually from a SCADA system. Signals may include water quality (WQ) data (e.g., residual or free chlorine concentration) and operations (OP) data (e.g., tank levels, or flow rates). Sensor hardware may also provide error data to the SCADA controller, which can be used by CANARY as an alarm (ALM) signal.
- **sub-station** – A monitoring station that is physically located at the same place as other monitoring stations. CANARY uses sub-stations to differentiate testing locations. For example, if a main supply tank has two outlet pipes that are being monitored; each outlet would be considered its own sub-station.
- **threshold (τ_A)** – A value used to determine if a data point is considered an outlier by CANARY. This value is calculated relative to residuals calculated for that signal. Threshold has units of standard deviation, σ , not the units of the raw signal data. The standard deviation is calculated within a *normalization window*. This allows outlier determinations for all signal data to be made using a single threshold value.
- **water quality pattern** – A recurring trend in one or more water quality signals that would normally be significant enough to trigger an alarm, but which is considered by the user to be a “normal event”. Such patterns could be caused by routine operations such as daily demand changes, pumps or plants turning on or off, or water treatment activities. If a pattern is regular enough, it can be identified, stored in a library, and used as a recognized pattern to help eliminate false alarms associated with normal activities.

SCIENCE



PRESORTED STANDARD
POSTAGE & FEES PAID
EPA
PERMIT NO. G-35

Office of Research and Development (8101R)
Washington, DC 20460

Official Business
Penalty for Private Use
\$300